Learning Discriminative Video Representations Using Adversarial Perturbations

Jue Wang[1*][0000−0001−8546−4522] and Anoop Cherian[2][0000−0002−5566−0351]

[1]Data61/CSIRO, ANU, Canberra [2]MERL Cambridge, MA
jue.wang@anu.edu.au cherian@merl.com

Abstract. Adversarial perturbations are noise-like patterns that can subtly change the data, while failing an otherwise accurate classifier. In this paper, we propose to use such perturbations for improving the robustness of video representations. To this end, given a well-trained deep model for per-frame video recognition, we first generate adversarial noise adapted to this model. Using the original data features from the full video sequence and their perturbed counterparts, as two separate bags, we develop a binary classification problem that learns a set of discriminative hyperplanes – as a subspace – that will separate the two bags from each other. This subspace is then used as a descriptor for the video, dubbed *discriminative subspace pooling*. As the perturbed features belong to data classes that are likely to be confused with the original features, the discriminative subspace will characterize parts of the feature space that are more representative of the original data, and thus may provide robust video representations. To learn such descriptors, we formulate a subspace learning objective on the Stiefel manifold and resort to Riemannian optimization methods for solving it efficiently. We provide experiments on several video datasets and demonstrate state-of-the-art results.

1 Introduction

Deep learning has enabled significant advancements in several areas of computer vision; however, the sub-area of video-based recognition continues to be elusive. In comparison to image data, the volumetric nature of video data makes it significantly more difficult to design models that can remain within the limitations of existing hardware and the available training datasets. Typical ways to adapt image-based deep models to videos are to resort to recurrent deep architectures or use three-dimensional spatio-temporal convolutional filters [8, 50, 42]. Due to hardware limitations, the 3D filters cannot be arbitrarily long. As a result, they usually have fixed temporal receptive fields (of a few frames) [50]. While recurrent networks, such as LSTM and GRU, have shown promising results on video tasks [60, 33, 3], training them is often difficult, and so far their performance has been inferior to models that look at parts of the video followed by a late fusion [8, 41].

[*] Work done while interning at MERL.

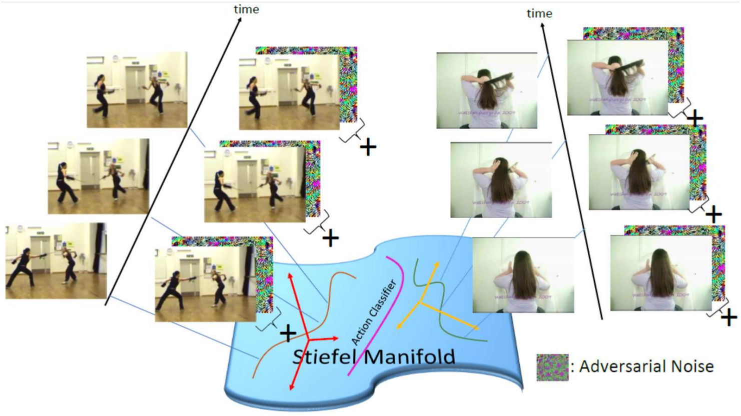


Fig. 1. A graphical illustration of our discriminative subspace pooling with adversarial noise. For every video sequence (as CNN features), our scheme generates a positive bag (with these features) and a negative bag by adding adversarial perturbations to the features. Next, we learn discriminative temporally-ordered hyperplanes that separate the two bags. We use orthogonality constraints on these hyperplanes and use them as representations for the video. As such representations belong to a Stiefel manifold, we use a classifier on this manifold for video recognition.

While, better CNN architectures, such as the recent I3D framework [8], is essential for pushing the state-of-the-art on video tasks, it is also important to have efficient representation learning schemes that can capture the long-term temporal video dynamics from predictions generated by a temporally local model. Recent efforts in this direction, such as rank pooling, temporal segment networks and temporal relation networks [55, 10, 22, 18, 21, 5, 45], aim to incorporate temporal dynamics over clip-level features. However, such models often ignore the noise in the videos, and use representations that adhere to a plausible criteria. For example, in the rank pooling scheme [10, 22, 21, 5, 20], it is assumed that the features from each frame are temporally-ordered, and learns a representation that preserves such order – however without accounting for whether the learned representation fits to data foreground or background.

In this paper, we present a novel pooling framework for temporally-ordered feature summarization. In contrast to prior works, we assume that per-frame video features consist of noisy parts that could confuse a classifier in a downstream task, such as for example, action recognition. A robust representation, in this setting, will be one that could avoid the classifier from using these vulnerable features for making predictions. However, finding these features is challenging as well. To this end, we resort to some intuitions made in a few works recently in the area of adversarial perturbations [34, 36, 35, 56]. Such perturbations are noise-like patterns that, when added to data, can fail an otherwise well-trained highly accurate classifier. Such perturbations are usually subtle, and in image recognition tasks, are quasi-imperceptible to a human. It was shown in several recent works that such noise can be learned from data. Specifically, by taking gradient ascent on a minimizing learning objective, one can produce such perturbations that will push the data points to the class boundaries, thereby making the classifier to mis-classify. Given that the strength (norm) of this noise is often bounded, it is highly likely that such noise will find minimum strength patterns that select features that are most susceptible to mis-classification. To this end, we use the recent universal adversarial perturbation generation scheme [35].

Once the perturbations are learned (and fixed) for the dataset, we use it to learn robust representations for the video. To this end, for features from every frame, we make two bags, one consisting of the original features, while the other one consisting of features perturbed by noise. Next, we learn a discriminative hyperplane that separates the bags in a max-margin framework. Such a hyperplane, which in our case is produced by a primal support vector machine (SVM), finds decision boundaries that could well-separate the bags; the resulting hyperplane is a single vector and is a weighted combination of all the data points in the bags. Given that the data features are non-linear, and given that a kernelized SVM might not scale well with sequence lengths, we propose to instead use multiple hyperplanes for the classification task, by stacking several such hyperplanes into a column matrix. We propose to use this matrix as our data representation for the video sequence.

However, there is a practical problem with our descriptor; each such descriptor is local to its respective sequences and thus may not be comparable between videos. To this end, we make additional restrictions on the hyperplanes – regularizing them to be orthogonal, resulting in our representation being subspaces. Such subspaces mathematically belong to the so-called Stiefel manifold [6]. We formulate a novel objective on this manifold for learning such subspaces on video features. Further, as each feature is not independent of the previous ones, we make additional temporal constraints. We provide efficient Riemannian optimization algorithms for solving our objective, specifically using the Riemannian conjugate gradient scheme that has been used in several other recent works [10, 25, 28]. Our overall pipeline is graphically illustrated in Figure 1.

We present experiments on three video recognition tasks, namely (i) action recognition, (ii) dynamic texture recognition, and (iii) 3D skeleton based action recognition. On all the experiments, we show that our scheme leads to state-of-the-art results, often improving the accuracy between 3–14%.

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

- We introduce adversarial perturbations into the video recognition setting for learning robust video representations.
- We formulate a binary classification problem to learn temporally-ordered discriminative subspaces that separate the data features from their perturbed counterparts.
- We provide efficient Riemannian optimization schemes for solving our objective on the Stiefel manifold.
- Our experiments on three datasets demonstrate state-of-the-art results.

2 Related work

Traditional video learning methods use hand-crafted features (from a few frames) – such as dense trajectories, HOG, HOF, etc. [52] – to capture the appearance and the video dynamics, and summarize them using a bag-of-words representation or more elegantly using Fisher vectors [38]. With the success of deep learning methods, feeding video data as RGB frames, optical flow subsequences, RGB differences, or 3D skeleton data directly into CNNs is preferred. One successful such approach is the two-stream model (and its variants) [42, 18, 17, 27] that use video segments (of a few frames) to train deep models, the predictions from the segments are fused via average pooling to generate a video level prediction. There are also extensions of this approach that directly learn models in an end-to-end manner [17]. While, such models are appealing to capture the video dynamics, it demands memory for storing the intermediate feature maps of the entire sequence, which may be impractical for long sequences. Recurrent models [2, 13, 14, 31, 46, 57] have been explored for solving this issue, that can learn to filter useful information while streaming the videos through them, but they are often found difficult to train [37]; perhaps due to the need to back-propagate over time. Using 3D convolutional kernels [8, 50] is another idea that proves to be promising, but bring along more parameters. The above architectures are usually trained for improving the classification accuracy, however, do not consider the robustness of their internal representations – accounting for which may improve their generalizability to unseen test data. To this end, we explore the vulnerable factors in a model (via generating adversarial perturbations [35]), and learn representations that are resilient to such factors in a network-agnostic manner.

Our main inspiration comes from the recent work of Moosavi et al. [35] that show the existence of quasi-imperceptible image perturbations that can fool a well-trained CNN model. They provide a systematic procedure to learn such perturbations in an image-agnostic way. In Xie et al. [56], such perturbations are used to improve the robustness of an object detection system. Similar ideas have been explored in [34, 36, 58]. In Sun et al. [48], a latent model is used to explicitly localize discriminative video segments. In Chang et al. [9], a semantic pooling scheme is introduced for localizing events in untrimmed videos. While these schemes share similar motivation as ours, the problem setup and formulations are entirely different.

On the representation learning front of our contribution, there are a few prior pooling schemes that are similar in the sense that they also use the parameters of an optimization functional as a representation. The most related work is rank-pooling and its variants [22, 21, 20, 47, 4, 11, 53] that use a rank-SVM for capturing the video temporal evolution. Similar to ours, Cherian et al. [10] propose to use a subspace to represent video sequences. However, none of these methods ensure if the temporal-ordering constraints capture useful video content or capture some temporally-varying noise. To overcome this issue, Wang et al [54] proposes a representation using the decision boundaries of a support vector machine classifier that separates data features from independently sampled noise. In this paper, we revisit this problem in the setting of data dependent noise generation via an adversarial noise design and learns a non-linear decision boundary using Riemannian optimization; our learned representations per sequence are more expressive and leads to significant performance benefits.

3 Proposed Method

Let us assume $X = \langle x_1, x_2, ..., x_n \rangle$ be a sequence of video features, where $x_i \in \mathbb{R}^d$ represents the feature from the i-th frame. We use 'frame' in a loose sense; it could mean a single RGB frame or a sequence of a few RGB or optical flow frames (as in the two stream [43] or the I3D architectures [8]) or a 3D skeleton. The feature representation x_i could be the outputs from intermediate layers of a CNN. As alluded to in the introduction, our key idea is the following. We look forward to an effective representation of X that is (i) compact, (ii) preserves characteristics that are beneficial for the downstream task (such as video dynamics), and (iii) efficient to compute. Recent methods such as generalized rank pooling [10] have similar motivations and propose a formulation that learns compact temporal descriptors that are closer to the original data in ℓ_2 norm. However, such a reconstructive objective may also capture noise, thus leading to sub-optimal performance. Instead, we take a different approach. Specifically, we assume to have access to some noise features $Z = \{z_1, z_2, ..., z_m\}$, each $z_i \in \mathbb{R}^d$. Let us call X the positive bag, with a label $y = +1$ and Z the negative bag with label $y = -1$. Our main goal is to find a discriminative hyperplane that separates the two bags; these hyperplanes can then be used as the representation for the bags.

An obvious question is how such a hyperplane can be a good data representation? To answer this, let us consider the following standard SVM formulation with a single discriminator $w \in \mathbb{R}^d$:

$$\min_{w, \xi \geq 0} \frac{1}{2} \|w\|^2 + \sum_{\theta \in X \cup Z} \left[\max(0, 1 - y(\theta) w^\top \theta + \xi_\theta) + C \xi_\theta \right], \tag{1}$$

where with a slight abuse of notation, we assume $y(\theta) \in \{+1, -1\}$ is the label of θ, ξ are the slack variables, and C is a regularization constant on the slacks. Given the positive and negative bags, the above objective learns a linear classification boundary that could separate the two bags with a classification accuracy of say γ. If the two bags are easily separable, then the number of support vectors used to construct the separating hyperplane might be a few and thus may not capture a weighted combination of a majority of the points in the bags - as a result, the learned hyperplane would not be representative of the bags. However, if the negative bag Z is suitably selected and we demand a high γ, we may turn (1) into a difficult optimization problem and would demand the solver to overfit the decision boundary to the bags; this overfitting creates a significantly better summarized representation, as it may need to span a larger portion of the bags to satisfy the γ accuracy.[1] This overfitting of the hyperplane is our key idea,

[1] Here regularization parameter C is mainly assumed to help avoid outliers.

```
Input: Feature points $x_{ij}$, Network weighting $W$, fooling rate $\psi$, cross entropy
       loss with softmax funtion $f(.)$, normalization operator $N(.)$.
Output: Adversarial noise vector $\epsilon$.
Initialization: $\epsilon \leftarrow 0$.
repeat
    $\Delta\epsilon \leftarrow \arg\min_r \|r\|_2 - \sum_{ij} f(W^\top(x_{ij}), W^\top(x_{ij} + \epsilon + r))$;
    $\epsilon \leftarrow N(\epsilon + \Delta\epsilon)$;
until $Accuracy \leq 1 - \psi$;
return $v$
```

Algorithm 1: Optimization step for solving adversarial noise.

that allows to avoid using data features that are susceptible to perturbations, while summarizing the rest.

There are two key challenges to be addressed in developing such a representation, namely (i) an appropriate noise distribution for the negative bag, and (ii) a formulation to learn the separating hyperplanes. We explore and address these challenges below.

3.1 Finding Noise Patterns

As alluded to above, having good noise distributions that help us identify the vulnerable parts of the feature space is important for our scheme to perform well. To this end, we resort to the recent idea of universal adversarial perturbations (UAP) [35]. This scheme is dataset-agnostic and provides a systematic and mathematically grounded formulation for generating adversarial noise that when added to the original features is highly-likely to mis-classify a pre-trained classifier. Further, this scheme is computationally efficient and requires less data for building relatively generalizable universal perturbations.

Precisely, suppose $\mathcal{X}$ denotes our dataset, let h be a CNN trained on $\mathcal{X}$ such that $h(x)$ for $x \in \mathcal{X}$ is a class label predicted by h. Universal perturbations are noise vectors ϵ found by solving the following objective:

$$\min_{\epsilon} \|\epsilon\| \text{ s.t. } h(x+\epsilon) \neq h(x), \forall x \in \mathcal{X}, \tag{2}$$

where $\|\epsilon\|$ is a suitable normalization on ϵ such that its magnitude remains small, and thus will not change x significantly. In [35], it is argued that this norm-bound restricts the optimization problem in (2) to look for the minimal perturbation ϵ that will move the data points towards the class boundaries; i.e., selecting features that are most vulnerable – which is precisely the type of noise we need in our representation learning framework.

To this end, we extend the scheme described in [35], to our setting. Differently to their work, we aim to learn a UAP on high-level CNN features as detailed in Alg. 1 above, where the x_{ij} refers to the i^{th} frame in the j^{th} video. We use the classification accuracy before and after adding the noise as our optimization criteria as captured by maximizing the cross-entropy loss.

3.2 Discriminative Subspace Pooling

Once a "challenging" noise distribution is chosen, the next step is to find a summarization technique for the given video features. While one could use a simple discriminative classifier, such as described in (1) to achieve this, such a linear classifier might not be sufficiently powerful to separate the potentially non-linear CNN features and their adversarial perturbations. An alternative is to resort to non-linear decision boundaries using a kernelized SVM; however that may make our approach less scalable and poses challenges for end-to-end learning. Thus, we look forward to a representation within the span of data features, while having more capacity for separating non-linear features.

Our main idea is to use a subspace of discriminative directions (as against a single one as in (1)) for separating the two bags such that every feature x_i is classified by at least one of the hyperplanes to the correct class label. Such a scheme can be looked upon as an approximation to a non-linear decision boundary by a set of linear ones, each one separating portions of the data. Mathematically, suppose $W \in \mathbb{R}^{d\times p}$ is a matrix with each hyperplane as its columns, then we seek to optimize:

$$\min_{W,\xi} \Omega(W) + \sum_{\theta \in X \cup Z} \left[\max\left(0, 1 - \max\left(\mathbf{y}(\theta) \odot \mathbf{W}^\top \theta\right) - \xi_\theta\right) + C\xi_\theta\right], \qquad (3)$$

where $\mathbf{y}$ is a vector with the label y repeated p times along its rows. The quantity Ω is a suitable regularization for W, of which one possibility is to use $\Omega(W) = W^\top W = \mathbf{I}_p$, in which case W spans a p dimensional subspace of $\mathbb{R}^d$. Enforcing such subspace constraints (orthonormality) on these hyperplanes are often empirically seen to demonstrate better performance as is also observed in [10]. The operator $\odot$ is the element-wise multiplication and the quantity $\max(\mathbf{y}(\theta) \odot \mathbf{W}^\top\theta)$ captures the maximum value of the element-wise multiplication, signifying that if at least one hyperplane classifies θ correctly, then the hinge-loss will be zero.

Recall that we work with video data, and thus there are temporal characteristics of this data modality that may need to be captured by our representation. In fact, recent works show that such temporal ordering constraints indeed results in better performance, e.g., in action recognition [10, 21, 5, 4]. However, one well-known issue with such ordered pooling techniques is that they impose a global temporal order on all frames jointly. Such holistic ordering ignores the repetitive nature of human actions, for example, in actions such as clapping or hand-waving. As a result, it may lead the pooled descriptor to overfit to non-repetitive features in the video data, which might be corresponding to noise/background. Usually a slack variable is introduced in the optimization to handle such repetitions, however its effectiveness is questionable. To this end, we propose a simple temporal segmentation based ordering constraints, where we first segment a video sequence into multiple non-overlapping temporal segments $\mathcal{T}_0, \mathcal{T}_1, ...\mathcal{T}_{\lfloor n/\delta \rfloor}$, and then enforce ordering constraints only within the segments. We find the segment length δ as the minimum number of consecutive frames that do not result in a repeat in the action features.

With the subspace constraints on W and introducing temporal segment-based ordering constraints on the video features, our complete **order-constrained discriminative subspace pooling optimization** can be written as:

$$\min_{\substack{W^\top W=\mathbf{I}_p,\\ \xi,\zeta\geq 0}} \sum_{\theta\in X\cup Z}\left[\max\left(0,1-\max\left(\mathbf{y}(\theta)\odot\mathbf{W}^\top\theta\right)-\xi_\theta\right)\right]+C_1\sum_{\theta\in X\cup Z}\xi_\theta+C_2\sum_{i<j}\zeta_{ij}, \quad (4)$$

$$\left\|W^\top x_i\right\|^2+1\leq\left\|W^\top x_j\right\|^2+\zeta_{ij},\quad i<j,\forall(i,j)\in\mathcal{T}_k,\text{where} \quad (5)$$

$$\mathcal{T}_k=\left\{k\delta+1,k\delta+2,...,\min(n,(k+1)\delta)\right\},\forall k\in\{0,1,...,\lfloor n/\delta\rfloor\} \quad (6)$$

$$\delta=b^*-a^*,\ \text{where}\ (a^*,b^*)=\operatorname*{arg\,min}_{a,b>a}\left\|x_a-x_b\right\|, \quad (7)$$

where (5) captures the temporal order, while the last two equations define the temporal segments, and computes the appropriate segment length δ, respectively. Note that, the temporal segmentation part could be done offline, by using all videos in the dataset, and selecting a δ which is the mean. In the next section, we present a scheme for optimizing W by solving the objective in (4)and (5).

Once each video sequence is encoded by a subspace descriptor, we use a classifier on the Stiefel manifold for recognition. Specifically, we use the standard exponential projection metric kernel [10, 26] to capture the similarity between two such representations, which are then classified using a kernelized SVM.

3.3 Efficient Optimization

The orthogonality constraints on W results in a non-convex optimization problem that may seem difficult to solve at first glance. However, note that such subspaces belong to well-studied objects in differential geometry. Specifically, they are elements of the Stiefel manifold $\mathcal{S}(d,p)$ (p subspaces in $\mathbb{R}^d$), which are a type of Riemannian manifolds with positive curvature [6]. There exists several well-known optimization techniques for solving objectives defined on this manifold [1], one efficient scheme is Riemannian conjugate gradient (RCG) [44]. This method is similar to the conjugate gradient scheme in Euclidean spaces, except that in the case of curved-manifold-valued objects, the gradients should adhere to the geometry (curvature) of the manifold (such as orthogonal columns in our case), which can be achieved via suitable projection operations (called exponential maps). However, such projections may be costly. Fortunately, there are well-known approximate projection methods, termed *retractions* that could achieve these projections efficiently without losing on the accuracy. Thus, tying up all together, for using RCG on our problem, the only part that we need to derive is the Euclidean gradient of our objective with respect to W. To this end, rewriting (5) as a hinge loss on (4), our objective on W and its gradient are:

$$\min_{W\in\mathcal{S}(d,p)} g(W):=\sum_{\theta\in X\cup Z}\left[\max\left(0,1-\max\left(\mathbf{y}(\theta)\odot\mathbf{W}^\top\theta\right)-\xi_\theta\right)\right]$$
$$+\frac{1}{n(n-1)}\sum_{i<j}\max(0,1+\left\|W^\top x_i\right\|^2-\left\|W^\top x_j\right\|^2-\zeta_{ij}), \quad (8)$$

$$\frac{\partial g}{\partial W} = \sum_{\theta \in X \cup Z} A(W;\theta, y(\theta)) + \frac{1}{n(n-1)} \sum_{i<j} B(W; x_i, x_j), \text{where} \tag{9}$$

$$A(W;\theta, y(\theta)) = \begin{cases} 0, \quad \text{if } \max(y(\theta) \odot W^\top\theta - \xi_\theta) \geq 1 \\ -\left[\mathbf{0}_{d\times r\text{-}1}\; y(\theta)\theta\; \mathbf{0}_{d\times p-r}\right],\ r = \arg\max_q y(\theta) \odot W_q^\top\theta, \text{ else} \end{cases} \tag{10}$$

$$B(W; x_i, x_j) = \begin{cases} 0, \quad \text{if } \left\|W^\top x_j\right\|^2 \geq 1 + \left\|W^\top x_i\right\|^2 - \zeta_{ij} \\ 2(x_i x_i^\top - x_j x_j^\top)W, \quad \text{else.} \end{cases} \tag{11}$$

In the definition of $A(W)$, we use $W_q^\top$ to denote the q-th column of W. To reduce clutter in the derivations, we have avoided including the terms using $\mathcal{T}$. Assuming the matrices of the form xx^T can be computed offline, on careful scrutiny we see that the cost of gradient computations on each data pair is only $O(d^2p)$ for $B(W)$ and $O(dp)$ for the discriminative part $A(W)$. If we include temporal segmentation with k segments, the complexity for $B(W)$ is $O(d^2p/k)$.

End-to-End Learning: The proposed scheme can be used in an end-to-end CNN learning setup where the representations can be learned jointly with the CNN weights. In this case, CNN backpropogation would need gradients with respect to the solutions of an argmin problem defined in (4), which may seem difficult. However, there exist well-founded techniques [12], [15][Chapter 5] to address such problems, specifically in the CNN setting [23] and such techniques can be directly applied to our setup. However, since gradient derivations using these techniques will require review of some well-known theoretical results that could be a digression from the course of this paper, we provide them in the supplementary materials.

4 Experiments

In this section, we demonstrate the utility of our discriminative subspace pooling (DSP) on several standard vision tasks (including action recognition, skeleton-based video classification, and dynamic video understanding), and on diverse CNN architectures such as ResNet-152, Temporal Convolutional Network (TCN), and Inception-ResNet-v2. We implement our pooling scheme using the ManOpt Matlab package [7] and use the RCG optimizer with the Hestenes-Stiefel's [24] update rule. We found that the optimization produces useful representations in about 50 iterations and takes about 5 milli-seconds per frame on a single core 2.6GHz CPU. We set the slack regularization constant $C = 1$. As for the CNN features, we used public code for the respective architectures to extract the features. Generating the adversarial perturbation plays a key role in our algorithm, as it is used to generate our negative bag for learning the discriminative hyperplanes. We follow the experimental setting in [35] to generate UAP noise for each model by solving the energy function as depicted in Alg. 1. Differently from [35], we generate the perturbation in the shape of the high level CNN feature instead of an RGB image. We review below our the datasets, their evaluation protocols, the CNN features next.

4.1 Datasets, CNN Architectures, and Feature Extraction

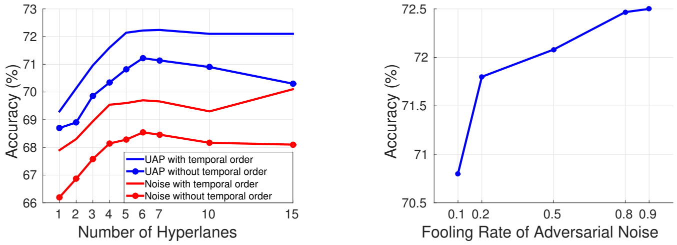
HMDB-51 [29]: is a popular video benchmark for human action recognition, consisting of 6766 Internet videos over 51 classes; each video is about 20 – 1000 frames. The standard evaluation protocol reports average classification accuracy on three-folds. To extract features, we train a two-stream ResNet-152 model (as in [42]) taking as input RGB frames (in the spatial stream) and a stack of optical flow frames (in the temporal stream). We use features from the pool5 layer of each stream as input to DSP, which are sequences of 2048D vectors.

NTU-RGBD [39]: is by far the largest 3D skeleton-based video action recognition dataset. It has 56,880 video sequences across 60 classes, 40 subjects, and 80 views. The videos have on average 70 frames and consist of people performing various actions; each frame annotated for 25 3D human skeletal keypoints (some videos have multiple people). According to different subjects and camera views, two evaluation protocols are used, namely cross-view and cross-subject evaluation [39]. We use the scheme in Shahroudy et al. [39] as our baseline in which a temporal CNN (with residual units) is applied on the raw skeleton data. We use the 256D features from the bottleneck layer (before their global average pooling layer) as input to our scheme.

YUP++ dataset [18]: is a recent dataset for dynamic video-texture understanding. It has 20 scene classes with 60 videos in each class. Importantly, half of the sequences in each class are collected by a static camera and the rest are recorded by a moving camera. The latter is divided into two sub-datasets, YUP++ stationary and YUP++ moving. As described in the [18], we apply the same 1/9 train-test ratio for evaluation. There are about 100-150 frames per sequence. We train an Inception-ResNet-v2 on the respective training set to generate the features and fine-tune a network that was pre-trained on the ImageNet dataset. In detail, we apply the 1/9 train-test ratio and follow the standard supervised training procedure of image-based tasks; following which we extract frame-level features (1536D) from the second-last fully-connected layer.

4.2 Parameter Analysis

Evaluating the Choice of Noise: As is clear by now, the noise patterns should be properly chosen, as it will affect how well the discriminative hyperplanes characterize useful video features. To investigate the quality of UAP features, we compare it with the baseline of choosing noise from a Gaussian distribution with the data mean and standard deviation computed on the respective video dataset (as done in the work of Wang et al. [54]). We repeat this experiment 10-times on the HMDB-51 split-1 features. In Figure 2(a), we plot the average classification accuracy after our pooling operation against an increasing number of hyperplanes in the subspaces. As is clear, using UAP significantly improves the performance against the alternative, substantiating our intuition. Further, we also find that using more hyperplanes is beneficial, suggesting that adding UAP to the features leads to a non-linear problem requiring more than a single discriminator to capture the informative content.



(a) (b)

Fig. 2. Analysis of the hyper parameters used in our scheme. All experiments use ResNet-152 features on HMDB-51 split-1 with a fooling rate of 0.8 in (a) and 6 hyperplanes in (b). See text for details.

Evaluating Temporal Constraints: Next, we evaluate the merit of including temporal-ordering constraints in the DSP objective, viz. (4). In Figure 2(a), we plot the accuracy with and without such temporal order, using the same settings as in the above experiment. As is clear, embedding temporal constraint will help the discriminative subspace capture representations that are related to the video dynamics, thereby showing better accuracy. In terms of the number of hyperplanes, the accuracy increases about 3% from one hyperplane to when using six hyperplanes, and drops around 0.5% from 6 hyperplanes to 15 hyperplanes, suggesting that the number of hyperplanes (6 in this case) is sufficient for representing most sequences.

UAP Fooling Rate: In Figure 2(b), we analyze the fooling rate of UAP that controls the quality of the adversary to confuse the trained classifier. The higher the fooling rate is, the more it will mix the information of the feature in different classes. As would be expected, we see that increasing the fooling rate from 0.1 to 0.9 increases the performance of our pooling scheme as well. Interestingly, our algorithm could perform relatively well without requiring a very high value of the fooling rate. From [35], a lower fooling rate would reduce the amount of data needed for generating the adversarial noise, making their algorithm computationally cheaper. Further, comparing Figures 2(a) and 2(b), we see that incorporating a UAP noise that has a fooling rate of even 10% does show substantial improvements in DSP performance against using Gaussian random noise (70.8% in Figure 2(b) against 69.8% in Figure 2(a)).

Experimental Settings: Going by our observations in the above analysis, for all the experiments in the sequel, we use six subspaces in our pooling scheme, use temporal ordering constraints in our objective, and use a fooling rate of 0.8 in UAP. Further, as mentioned earlier, we use an exponential projection metric kernel [11] for the final classification of the subspace descriptors using a kernel SVM. Results using end-to-end learning are provided in the supplementary materials.

	HMDB-51			NTU-RGBD		YUP++	
	Spatial	Temporal	Two-stream	Cross-subject	Cross-view	Stationary	Moving
AP	46.7% [19]	60.0% [19]	63.8% [19]	74.3% [45]	83.1% [45]	85.1%	76.5%
MP	45.1%	58.5%	60.6%	65.4%	78.5%	81.8%	72.4%
DSP	**58.5%**	**67.0%**	**72.5%**	**81.6%**	**88.7%**	**95.1%**	**88.3%**

Table 1. The accuracy comparison between our Discriminative subspace pooling (DSP) with standard Average pooling (AP) and Max pooling (MP).

HMDB-51	
Method	Accuracy
Temporal Seg. n/w [55]	69.4%
TS I3D [8]	80.9%
ST-ResNet [16]	66.4%
ST-ResNet+IDT [16]	70.3%
STM Network [17]	68.9%
STM Network+IDT [17]	72.2%
ShuttleNet+MIFS [40]	71.7%
GRP [10]	70.9%
SVMP [54]	71.0%
L^2STM [49]	66.2%
Ours(TS ResNet)	**72.4%**
Ours(TS ResNet+IDT)	**74.3%**
Ours(TS I3D)	**81.5%**

NTU-RGBD		
Method	Cross-Subject	Cross-View
VA-LSTM [59]	79.4%	87.6%
TS-LSTM [30]	74.6%	81.3%
ST-LSTM+Trust Gate [32]	69.2%	77.7%
SVMP [54]	78.5%	86.4%
GRP [10]	76.0%	85.1%
Res-TCN [45]	74.3%	83.1%
Ours	**81.6%**	**88.7%**

YUP++		
Method	Stationary	Moving
TRN [18]	92.4%	81.5%
SVMP [54]	92.5%	83.1%
GRP [10]	92.9%	83.6%
Ours	**95.1%**	**88.3%**

Table 2. Comparisons to the state-of-the-art on each dataset following their respective official evaluation protocols. We used three splits for HMDB-51. 'TS' refers to 'Two-Stream'.

4.3 Experimental Results

Compared with standard pooling: In Table 1, we show the performance of DSP on the three datasets and compare to standard pooling methods such as average pooling and max pooling. As is clear, we outperform the baseline results by a large margin. Specifically, we achieve 9% improvement on the HMDB-51 dataset split-1 and $5\% - 8\%$ improvement on the NTU-RGBD dataset. On these two datasets, we simply apply our pooling method on the CNN features extracted from the pre-trained model. We achieve a substantial boost (of up to 12%) after applying our scheme.

Comparisons to the State of the Art: In Table 2, we compare DSP to the state-of-the-art results on each dataset. On the HMDB-51 dataset, we also report accuracy when DSP is combined hand-crafted features (computed using dense trajectories [51] and summarized as Fisher vectors (IDT-FV)). As the results show, our scheme achieves significant improvements over the state of the art. For example, without IDT-FV, our scheme is 3% better than than the next best scheme [55] (69.4% vs. 72.4% ours). Incorporating IDT-FV improves this to 74.3% which is again better than other schemes. We note that the I3D architecture [8] was introduced recently that is pre-trained on the larger Kinectics dataset and when fine-tuned on the HMDB-51 leads to about 80.9% accuracy. To understand the advantages of DSP on pooling I3D model generated features, we applied our scheme to their bottleneck features (extracted using the public code provided by the authors) from the fine-tuned model. We find that our scheme further improves I3D by about 0.6% showing that there is still room for improvement for this model. On the other two datasets, NTU-RGBD and YUP++, we find that our scheme leads to about 5–7% and 3–6% improvements respectively, and outperforms prior schemes based on recurrent networks and temporal relation models, suggesting that our pooling scheme captures spatio-temporal cues much better than recurrent models.

Run Time Analysis: In Figure 3, we compare the run time of DSP with similar methods such as rank pooling, dynamic images, and GRP. We used the Matlab implementations of other schemes and used the same hardware platform (2.6GHz Intel CPU single core) for our comparisons. To be fair, we used a single hyperplane in DSP. As the plot shows, our scheme is similar in computations to rank pooling and GRP.

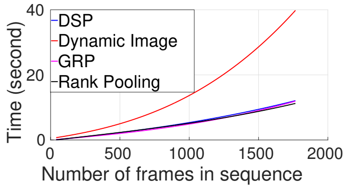


Fig. 3. Run time analysis of DSP against GRP [10], RP [21], and Dynamic Images [5].

#frames	1	80	100	140	160	180	260
#classes	51	49	34	27	23	21	12
AP [8]	**80.8**	81.8	86.1	84.1	82.3	78.0	**77.3**
DSP (ours)	**81.6**	82.8	88.5	88.0	86.1	83.3	**82.6**

Table 3. Comparison of I3D performance on sequences of increasing lengths in HMDB-51 split-1.

Analysis of Results on I3D Features: To understand why the improvement of DSP on I3D (80.9% against our 81.5%) is not significant (on HMDB-51) in comparison to our results on other datasets, we further explored the reasons. Apparently, the I3D scheme uses chunks of 64 frames as input to generate one feature output. However, to obtain DSP representations, we need a sufficient number of features per video sequence to solve the underlying Riemannian optimization problem adequately, which may be unavailable for shorter video clips. To this end, we re-categorized HMDB-51 into subsets of sequences according to their lengths. In Table 4.3, we show the performance on these subsets and the number of action classes for sequences in these subsets. As our results show, while the difference between average pool (AP) (as is done in [8]) and DSP is less significant when the sequences are smaller (<80 frames), it becomes significant (>5%) when the videos are longer (>260 frames). This clearly shows that DSP on I3D is significantly better than AP on I3D.

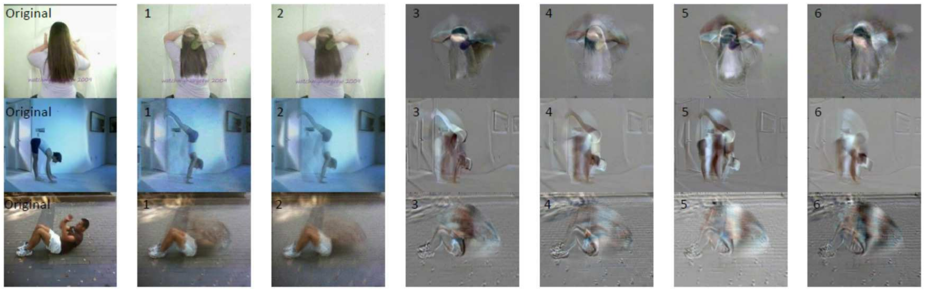


Fig. 4. Visualizations of our DSP descriptor (when applied on raw RGB frames) on an HMDB-51 video sequences. First column shows a sample frame from the video, second-to-seventh columns show the six hyperplanes produced by DSP. Interestingly, we find that each hyperplane captures different aspects of the sequences–first two mostly capture spatial, while the rest capture the temporal dynamics at increasing granularities.

Qualitative Results: In Figure 4, we visualize the hyperplanes that our scheme produces when applied to raw RGB frames from HMDB-51 videos – i.e., instead of CNN features, we directly feed the raw RGB frames into our DSP, with adversarial noise generated as suggested in [35]. We find that the subspaces capture spatial and temporal properties of the data separately; e.g., the first two hyperplanes seem to capture mostly the spatial cues in the video (such as the objects, background, etc.) while the rest capture mostly the temporal dynamics at greater granularities. Note that we do not provide any specific criteria to achieve this behavior, instead the scheme automatically seem to learn such hyperplanes corresponding to various levels of discriminative information. In the supplementary materials, we provide comparisons of this visualization against those generated by PCA and generalized rank pooling [10].

5 Conclusions

In this paper, we investigated the problem of representation learning for video sequences. Our main innovation is to generate and use synthetic noise, in the form of adversarial perturbations, for producing our representation. Assuming the video frames are encoded as CNN features, such perturbations are often seen to affect vulnerable parts of the features. Using such generated perturbations to our benefit, we propose a discriminative classifier, in a max-margin setup, via learning a set of hyperplanes as a subspace, that could separate our synthetic noise from data. As such hyperplanes need to fit to useful parts of the features for achieving good performance, it is reasonable to assume they capture data parts that are robust. We provided a non-linear objective for learning our subspace representation and explored efficient optimization schemes for computing it. Experiments on several datasets explored the effectiveness of each component in our scheme, demonstrating state-of-the-art performance on the benchmarks.

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization algorithms on matrix manifolds. Princeton University Press (2009)
2. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A.: Sequential deep learning for human action recognition. In: Human Behavior Understanding, pp. 29–39 (2011)
3. Ballas, N., Yao, L., Pal, C., Courville, A.: Delving deeper into convolutional networks for learning video representations. In: ICLR (2016)
4. Bilen, H., Fernando, B., Gavves, E., Vedaldi, A.: Action recognition with dynamic image networks. PAMI (2017)
5. Bilen, H., Fernando, B., Gavves, E., Vedaldi, A., Gould, S.: Dynamic image networks for action recognition. In: CVPR (2016)
6. Boothby, W.M.: An introduction to differentiable manifolds and Riemannian geometry, vol. 120. Academic press (1986)
7. Boumal, N., Mishra, B., Absil, P.A., Sepulchre, R.: Manopt, a matlab toolbox for optimization on manifolds. JMLR **15**(1), 1455–1459 (2014)
8. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (July 2017)
9. Chang, X., Yu, Y.L., Yang, Y., Xing, E.P.: Semantic pooling for complex event analysis in untrimmed videos. PAMI **39**(8), 1617–1632 (2017)
10. Cherian, A., Fernando, B., Harandi, M., Gould, S.: Generalized rank pooling for activity recognition. In: CVPR (2017)
11. Cherian, A., Sra, S., Gould, S., Hartley, R.: Non-linear temporal subspace representations for activity recognition. In: CVPR (2018)
12. Chiang, A.C.: Fundamental methods of mathematical economics (1984)
13. Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: CVPR (2015)
14. Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. In: CVPR (2015)
15. Faugeras, O.: Three-dimensional computer vision: a geometric viewpoint. MIT press (1993)
16. Feichtenhofer, C., Pinz, A., Wildes, R.: Spatiotemporal residual networks for video action recognition. In: NIPS (2016)
17. Feichtenhofer, C., Pinz, A., Wildes, R.P.: Spatiotemporal multiplier networks for video action recognition. In: CVPR (2017)
18. Feichtenhofer, C., Pinz, A., Wildes, R.P.: Temporal residual networks for dynamic scene recognition. In: CVPR (2017)
19. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: CVPR (2016)
20. Fernando, B., Anderson, P., Hutter, M., Gould, S.: Discriminative hierarchical rank pooling for activity recognition. In: CVPR (2016)
21. Fernando, B., Gavves, E., Oramas, J.M., Ghodrati, A., Tuytelaars, T.: Modeling video evolution for action recognition. In: CVPR (2015)
22. Fernando, B., Gould, S.: Learning end-to-end video classification with rank-pooling. In: ICML (2016)
23. Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R.S., Guo, E.: On differentiating parameterized argmin and argmax problems with application to bi-level optimization. arXiv preprint arXiv:1607.05447 (2016)

24. Hager, W.W., Zhang, H.: A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM Journal on optimization **16**(1), 170–192 (2005)
25. Harandi, M.T., Salzmann, M., Hartley, R.: From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. In: ECCV (2014)
26. Harandi, M.T., Salzmann, M., Jayasumana, S., Hartley, R., Li, H.: Expanding the family of grassmannian kernels: An embedding perspective. In: ECCV (2014)
27. Hayat, M., Bennamoun, M., An, S.: Deep reconstruction models for image set classification. PAMI **37**(4), 713–727 (2015)
28. Huang, Z., Wang, R., Shan, S., Chen, X.: Projection metric learning on grassmann manifold with application to video based face recognition. In: CVPR (2015)
29. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: ICCV (2011)
30. Lee, I., Kim, D., Kang, S., Lee, S.: Ensemble deep learning for skeleton-based action recognition using temporal sliding LSTM networks. In: ICCV (2017)
31. Li, Q., Qiu, Z., Yao, T., Mei, T., Rui, Y., Luo, J.: Action recognition by learning deep multi-granular spatio-temporal video representation. In: ICMR (2016)
32. Liu, J., Shahroudy, A., Xu, D., Kot, A.C., Wang, G.: Skeleton-based action recognition using spatio-temporal LSTM network with trust gates. arXiv preprint arXiv:1706.08276 (2017)
33. Liu, J., Shahroudy, A., Xu, D., Wang, G.: Spatio-temporal LSTM with trust gates for 3d human action recognition. In: ECCV (2016)
34. Lu, J., Issaranon, T., Forsyth, D.: Safetynet: Detecting and rejecting adversarial examples robustly. In: ICCV (2017)
35. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations (2017)
36. Oh, S.J., Fritz, M., Schiele, B.: Adversarial image perturbation for privacy protection–a game theory perspective. In: ICCV (2017)
37. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: ICML (2013)
38. Sadanand, S., Corso, J.J.: Action bank: A high-level representation of activity in video. In: CVPR (2012)
39. Shahroudy, A., Liu, J., Ng, T.T., Wang, G.: NTU RGB+ D: A large scale dataset for 3d human activity analysis. In: CVPR (2016)
40. Shi, Y., Tian, Y., Wang, Y., Zeng, W., Huang, T.: Learning long-term dependencies for action recognition with a biologically-inspired deep network. In: ICCV (2017)
41. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
42. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS (2014)
43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
44. Smith, S.T.: Optimization techniques on riemannian manifolds. Fields institute communications **3**(3), 113–135 (1994)
45. Soo Kim, T., Reiter, A.: Interpretable 3d human action analysis with temporal convolutional networks. In: CVPR Workshops (2017)
46. Srivastava, N., Mansimov, E., Salakhutdinov, R.: Unsupervised learning of video representations using LSTMs. In: ICML (2015)
47. Su, B., Zhou, J., Ding, X., Wang, H., Wu, Y.: Hierarchical dynamic parsing and encoding for action recognition. In: ECCV (2016)

48. Sun, C., Nevatia, R.: Discover: Discovering important segments for classification of video events and recounting. In: CVPR (2014)
49. Sun, L., Jia, K., Chen, K., Yeung, D.Y., Shi, B.E., Savarese, S.: Lattice long short-term memory for human action recognition. In: ICCV (2017)
50. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: ICCV (2015)
51. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Dense trajectories and motion boundary descriptors for action recognition. IJCV **103**(1), 60–79 (2013)
52. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV (2013)
53. Wang, J., Cherian, A., Porikli, F.: Ordered pooling of optical flow sequences for action recognition. In: WACV. IEEE (2017)
54. Wang, J., Cherian, A., Porikli, F., Gould, S.: Video representation learning using discriminative pooling. In: CVPR (2018)
55. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: ECCV (2016)
56. Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., Yuille, A.: Adversarial examples for semantic segmentation and object detection. In: ICCV (2017)
57. Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR (2015)
58. Zhang, J., Zhang, T., Dai, Y., Harandi, M., Hartley, R.: Deep unsupervised saliency detection: A multiple noisy labeling perspective. In: CVPR (2018)
59. Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., Zheng, N.: View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In: ICCV (2017)
60. Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., Xie, X., et al.: Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In: AAAI (2016)