Discriminative Video Representation Learning Using Support Vector Classifiers

Wang, J.; Cherian, A.

Abstract

Most popular deep models for action recognition in videos generate independent predictions for short clips, which are then pooled heuristically to assign an action label to the full video segment. As not all frames may characterize the underlying action—indeed, many are common across multiple actions—pooling schemes that impose equal importance on all frames might be unfavorable. In an attempt to tackle this problem, we propose discriminative pooling, based on the notion that among the deep features generated on all short clips, there is at least one that characterizes the action. To identify these useful features, we resort to a negative bag consisting of features that are known to be irrelevant, for example, they are sampled either from datasets that are unrelated to our actions of interest or are CNN features produced via random noise as input. With the features from the video as a positive bag and the irrelevant features as the negative bag, we cast an objective to learn a (nonlinear) hyperplane that separates the unknown useful features from the rest in a multiple instance learning formulation within a support vector machine setup. We use the parameters of this separating hyperplane as a descriptor for the full video segment. Since these parameters are directly related to the support vectors in a max-margin framework, they can be treated as a weighted average pooling of the features from the bags, with zero weights given to non-support vectors. Our pooling scheme is end-to-end trainable within a deep learning framework. We report results from experiments on eight computer vision benchmark datasets spanning a variety of video-related tasks and demonstrate state-of-the-art performance across these tasks.

Discriminative Video Representation Learning Using Support Vector Classifiers

Jue Wang   Anoop Cherian

Abstract—Most popular deep models for action recognition in videos generate independent predictions for short clips, which are then pooled heuristically to assign an action label to the full video segment. As not all frames may characterize the underlying action—indeed, many are common across multiple actions—pooling schemes that impose equal importance on all frames might be unfavorable. In an attempt to tackle this problem, we propose *discriminative pooling*, based on the notion that among the deep features generated on all short clips, there is at least one that characterizes the action. To identify these useful features, we resort to a negative bag consisting of features that are known to be irrelevant, for example, they are sampled either from datasets that are unrelated to our actions of interest or are CNN features produced via random noise as input. With the features from the video as a positive bag and the irrelevant features as the negative bag, we cast an objective to learn a (nonlinear) hyperplane that separates the unknown useful features from the rest in a multiple instance learning formulation within a support vector machine setup. We use the parameters of this separating hyperplane as a descriptor for the full video segment. Since these parameters are directly related to the support vectors in a max-margin framework, they can be treated as a weighted average pooling of the features from the bags, with zero weights given to non-support vectors. Our pooling scheme is end-to-end trainable within a deep learning framework. We report results from experiments on eight computer vision benchmark datasets spanning a variety of video-related tasks and demonstrate state-of-the-art performance across these tasks.

Index Terms—video representation, video data mining, discriminative pooling, action recognition, deep learning.

◆

1 INTRODUCTION

WE are witnessing an astronomical increase of video data around us. This data deluge has brought out the problem of effective video representation – specifically, their semantic content – to the forefront of computer vision research. The resurgence of convolutional neural networks (CNN) has enabled significant progress to be made on several problems in computer vision [1], [2] and is now pushing forward the state-of-the-art in action recognition and video understanding as well [3], [4], [5], [6]. Even so, current solutions for video representation are still far from being practically useful, arguably due to the volumetric nature of this data modality and the complex nature of real-world human actions.

Using effective architectures, CNNs are often found to extract features from images that perform well on recognition tasks. Leveraging this know-how, deep learning solutions for video action recognition have so far been straightforward extensions of image-based models [6], [7], [8]. However, applying such models directly on video data is not an easy task as the video can be arbitrarily long, to address which a CNN may need to be scaled up by yet another dimension of complexity, which could increase the number of parameters sharply. This demands more advanced computational infrastructures and greater quantities of clean training data [3], [9]. To overcome this problem, the trend has been on converting the video data to short temporal segments consisting of one to a few frames, on which the existing image-based CNN models are trained. For example, in the popular two-
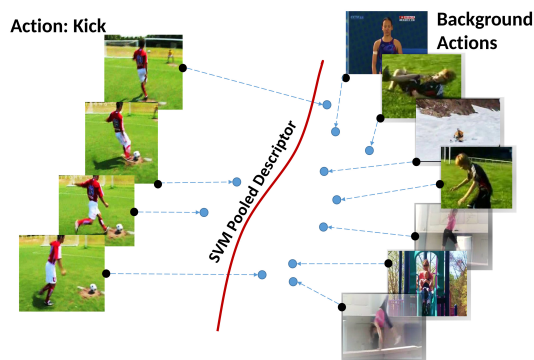

Fig. 1: A illustration of our discriminative pooling scheme. Our main idea is to learn a representation for the positive bag (left) of CNN features from the video of interest. To extract useful features from this video, we use a negative bag (right) of features from videos that are known to contain irrelevant/noise features. The representation learning problem is cast as a binary (non)-linear classification problem in an SVM setting; the hyperplane found via the optimization (which is a linear combination of support vectors) is used as the representation of the positive bag, which we call the *SVM pooled descriptor*.

stream model [7], [10], [11], [12], [13], the CNNs are trained to independently predict actions from short video clips (consisting of single frames or stacks of about ten optical flow frames) or a snippet of about 64 frames as in the recent I3D architecture [3]; these predictions are then pooled to generate a prediction for the full sequence – typically using average/max pooling. While average pooling gives equal weights to all the predictions, max pooling may be sensitive to outliers. There have also been recent approaches that learn representations over features produced by, say a two-stream model, such as the temporal relation networks of

- *Jue Wang is with the Research School of Engineering, The Australian National University, ACT 2601, Australia. E-mail: jue.wang@anu.edu.au*

- *Anoop Cherian is with Mistubishi Electric Research Labs (MERL), Cambridge, MA, E-mail: cherian@merl.com*

Zhou et al. [6], the rank pooling and its variants Bilen et al., [14], Fernando et al., [15], and Cherian et al., [16], [17] that capture the action dynamics, higher-order statistics of CNN features Cherian et al., [18], [19], CNN features along motion trajectories Wang et al., [12] and temporal segments Wang et al., [20], to name a few. However, none of these methods avoid learning meaningless information from the noise/background within the video, explicitly modeling which and demonstrating its benefits, are the main contributions of this paper.

To this end, we observe that not all predictions on the short video snippets are equally informative, yet some of them must be [21]. This allows us to cast the problem in a multiple instance learning (MIL) framework, where we assume that some of the features in s given sequence are indeed useful, while the rest are not. We assume all the CNN features from a sequence (containing both the good and the bad features) to represent a positive bag, while CNN features from unrelated video frames or synthetically generated random noise frames as a negative bag. We would ideally want the features in the negative bag to be correlated well to the uninformative features in the positive bag. We then formulate a binary classification problem of separating as many good features as possible in the positive bag using a discriminative classifier (we use a support vector machine (SVM) for this purpose). The decision boundary of this classifier thus learned is then used as a descriptor for the entire video sequence, which we call the SVM Pooled (SVMP) descriptor. To accommodate the fact that we are dealing with temporally-ordered data in the positive bag, we also explore learning our representations with partial ordering relations. An illustration of our SVMP scheme is shown in the Figure 1.

Our SVMP scheme/descriptor shares several properties of standard pooled descriptors, however also showcases several important advantages. For example, similar to other pooling schemes, SVM pooling results in a compact and fixed length representation of videos of arbitrary length. However differently, our pooling gives different weights to different features, and thus may be seen as a type of weighted average pooling, by filtering out features that are perhaps irrelevant for action recognition. Further, given that our setup uses a max-margin encoding of the features, the pooled descriptor is relatively stable with respect to data perturbations and outliers. Our scheme is agnostic to the feature extractor part of the system, for example, it could be applied to the intermediate features from any CNN model or even hand-crafted features. Moreover, the temporal dynamics of actions are explicitly encoded in the formulation. The scheme is fast to implement using publicly available SVM solvers, and also could be trained in an end-to-end manner within a CNN setup.

To evaluate our SVMP scheme, we provide extensive experiments on various datasets spanning a diverse set of tasks, namely action recognition and forecasting on HMDB-51 [22], UCF-101 [23], Kinetics-600 [24] and Charades [25]; skeleton-based action recognition on MSR action-3D [26], and NTU-RGBD [27]; image-set verification on the PubFig dataset [28], and video-texture recognition on the YUP++ dataset [29]. We outperform standard pooling methods on these datasets by a significant margin (between 3–14%) and demonstrate superior performance against state-of-the-art results by 1–5%.

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

- We introduce the concept of multiple instance learning (MIL) into a binary SVM classification problem for learn-ing video descriptors.
- We propose SVM pooling that captures and summarizes the discriminative features in a video sequence while explicitly encoding the action dynamics.
- We explore variants of our optimization problem and present progressively cheaper inference schemes, includ-ing a joint pooling and classification objective, as well as an end-to-end learnable CNN architecture.
- We demonstrate the usefulness of our SVMP descriptor by applying it on eight popular vision benchmarks spanning diverse input data modalities and CNN architectures.

2 RELATED WORK

The problem of video representation learning has received significant interest over the past decades. Thus, we restrict our literature review to some of the more recent methods, and defer the interested reader to excellent surveys on the topics such as [30], [31], [32].

2.1 Video Representation Using Shallow Features

Traditional methods for video action recognition typically use hand-crafted local features, such as dense trajectories, HOG, HOF, etc. [33], which model videos by combining dense sampling with feature tracking. However, the camera motion, as one of the video natures, usually result in non-static video background and hurt the quality of features. To tackle this problem, Wang et al. [34] improved the performance of dense trajectories by removing background trajectories and warping optical flow. Based on the improved dense trajectories, high-level representations are designed via pooling appearance and flow features along these trajectories, and have been found to be useful to capture human actions. For example, Sadanand et al. [35] propose Action Bank, which converts the individual action detector into semantic and viewpoint space. Similarly, Bag of words model [36], Fisher vector [37], and VLAD [38] representations are mid-level rep-resentations built on such hand-crafted features with the aim of summarizing local descriptors into a single vector representation. In Peng et al. [39], a detailed survey of these ideas is presented. In comparison to these classic representation learning schemes, our proposed setup is grounded on discriminatively separating useful data from the rest.

2.2 Video Representation Using Deep Features

With the resurgence of deep learning methods for object recog-nition [40], there have been several attempts to adapt these models to action recognition. Recent practice is to feed the video data, including RGB frames, optical flow subsequences, and 3D skeleton data into a deep (recurrent) network to train it in a supervised manner. Successful methods following this approach are the two-stream models and their extensions [4], [7], [29], [41], [42]. As apparent from its name, it has two streams, spatial stream is to capture the appearance information from RGB frames and temporal stream is to learn the motion dynamics from stacked optical flow. And then, they apply early or late fusion strategy to predict the final label. Although the architecture of these networks are different, the core idea is to split the video into short clips and embed them into a semantic feature space, and then recognize the actions either by aggregating the individual features per clip using some statistic (such as max or average) or directly training a CNN based end-to-end classifier [4]. While the latter schemes

are appealing, they usually need to store the feature maps from all the frames in memory which may be prohibitive for longer sequences. Moreover, this kind of training strategy may fail to capture the long-term dynamics in the video sequence. To tackle this problem, some recurrent models [43], [44], [45], [46], [47], [48] are proposed, which use long-short term memory (LSTM) or gate recurrent unit (GRU) to embed the temporal relationship among frames by using logistic gates and hidden states. However, the recurrent neural networks are usually hard to train [49] due to the exploding and vanishing gradient problem. Temporal Segment Network (TSN) [20] and Temporal Relation Network (TRN) [6] provide alternative solutions that are easier to train.

Another promising solution is to use 3D convolutional filters [3], [50], [51], [52], [53]. Compared to 2D filters, 3D filters can capture both spatial and temporal video structure. However, feeding the entire video into the CNNs may be computationally prohibitive. Further, 3D kernels bring more parameters into the architecture; as a result, may demand large and clean data for effective training [3]. While, an effective CNN architecture that can extract useful action-related features is essential to make progress in video understanding, we focus on the other aspect of the problem – that is, given a CNN architecture how well can we summarize the features it produces for improving action recognition. To this end, our efforts in this paper can be seen as complimentary to these recent approaches.

2.3 Video Representation Using Pooling Schemes

Typically, pooling schemes consolidate input data into compact representations based on some data statistic that summarizes the useful content. For example, average and max pooling captures zero-th and first order statistics. There are also works that use higher-order pooling, such as Cherian and Gould [54] using second-order, Cherian et al. [19] using third-order, and Girdhar et al., [55] proposing a video variant of the VLAD encoding which is approximately a mixture model. A recent trend in pooling schemes, which we also follow in this paper, is to use the parameters of a data modeling function, as the representation. For example, rank pooling [15] proposes to use the parameters of a support vector regressor as a video representation. In Bilen et al., [14], rank pooling is extended towards an early frame-level fusion, dubbed *dynamic images*; Wang et al. [56], extends this idea to use optical flow, which they call *dynamic flow* representation. Cherian et al. [17] generalized rank pooling to include multiple hyperplanes as a subspace, enabling a richer characterization of the spatio-temporal details of the video. This idea was further extended to non-linear feature representations via kernelized rank pooling in [16]. However, while most of these methods optimize a rank-SVM based regression formulation, our motivation and formulation are different. We use the parameters of a binary SVM to be the video level descriptor, which is trained to classify the frame level features from a pre-selected (but arbitrary) bag of negative features. Similar works are Exemplar-SVMs [57], [58], [59], that learn feature filters per data sample and then use these filters for feature extraction. However, in this paper, we use the decision boundary of the SVM to be the video level descriptor, that separate as many discriminative features as possible in each sequence while implicitly encoding the temporal order of these features.

2.4 Multiple Instance Learning

An important component of our algorithm is the MIL scheme, which is a popular data selection technique [60], [61], [62], [63], [64]. In the context of video representation, schemes similar in motivation have been suggested before. For example, Satkin and Hebert [65] explore the effect of temporal cropping of videos to regions of actions; however, it assumes these regions are continuous. Nowozin et al. [66] represent videos as sequences of discretized spatiotemporal sets and reduces the recognition task into a max-gain sequence finding problem on these sets using an LPBoost classifier. Similar to ours, Li et al. [67] propose an MIL setup for complex activity recognition using a dynamic pooling operator – a binary vector that selects input frames to be part of an action, which is learned by reducing the MIL problem to a set of linear programs. Chen and Nevatia [68] propose a latent variable based model to explicitly localize discriminative video segments where events take place. Vahdat et al. present a compositional model in [69] for video event detection, which is presented using a multiple kernel learning based latent SVM. While all these schemes share similar motivations as ours, we cast our MIL problem in the setting of normalized set kernels [70] and reduce the formulation to standard SVM setup which can be solved rapidly. In the $\propto$-SVMs of Yu et al., [71], [72], the positive bags are assumed to have a fixed fraction of positives, which is a criterion we also assume in our framework. However, the negative bag selection, optimization setup and our goals are different; specifically, our goal is to learn a video representation for any subsequent task including recognition, anticipation, and detection, while the framework in [71] is designed for event detection. And we generate the negative bag by using CNN features generated via inputing random noise images to the network.

The current paper is an extension of our published conference paper [73] and differs in the following ways. Apart from the more elaborate literature survey we present, we also provide extensions of our pooling scheme, specifically by incorporating temporal-ordering constraints. We provide detailed derivations of our end-to-end pooling variant. We further present elaborate experiments on five more datasets in addition to the three datasets that we used in [73], including a large scale action recognition experiment using the recently proposed Kinetics-600 dataset.

3 PROPOSED METHOD

In this section, we first describe the problem of learning SVMP descriptors and introduce three different ways to solve it. Before proceeding, we provide a snapshot of our main idea and problem setup graphically in Figure 2. Starting from frames (or flow images) in positive and negative bags, these frames are first passed through some CNN model for feature generation. These features are then passed to our SVMP module that learns (non-linear) hyperplanes separating the features from the positive bag against the ones from the negative bag, the latter is assumed fixed for all videos. These hyperplane representations are then used to train an action classifier at the video level. In the following, we formalize these ideas concretely.

3.1 Problem Setup

Let us assume we are given a dataset of N video sequences $\mathcal{X}^+ = \{X_1^+, X_2^+, ..., X_N^+\}$, where each X_i^+ is a set of frame level features, $i.e.$, $X_i^+ = \{\mathbf{x}_1^{i+}, \mathbf{x}_2^{i+}, ..., \mathbf{x}_n^{i+}\}$, each $\mathbf{x}_k^{i+} \in \mathbb{R}^p$. We assume that each X_i^+ is associated with an action class label
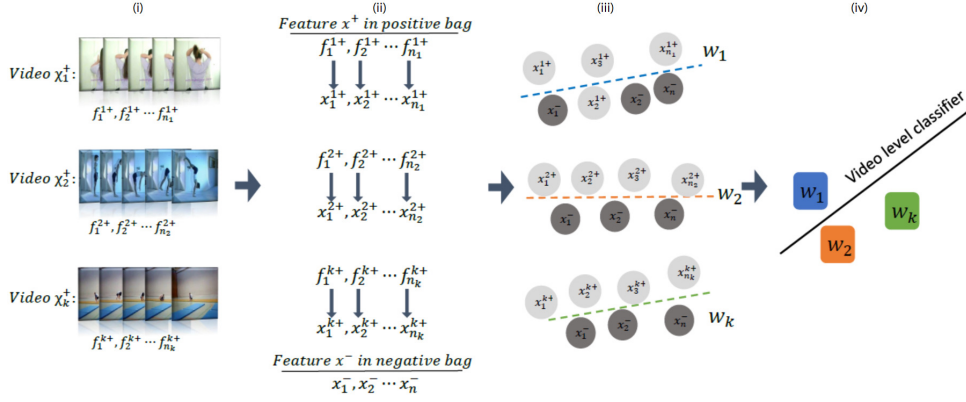
Fig. 2: Illustration of our SVM Pooling scheme. (i) Extraction of frames from videos, (ii) Converting frames f into feature x, (iii) Learning decision boundary w from feature x, and (iv) Using w as video descriptor.

$y_i^+ \in \{1, 2, ..., d\}$. Further, the $+$ sign denotes that the features and the sequences represent a positive bag. We also assume that we have access to a set of sequences $\mathcal{X}^- = \{X_1^-, X_2^-, ...X_M^-\}$ belonging to actions different from those in $\mathcal{X}^+$, where each $X_j^- = \{\mathbf{x}_1^{j-}, \mathbf{x}_2^{j-}, ..., \mathbf{x}_n^{j-}\}$ are the features associated with a negative bag, each $\mathbf{x}_k^{j-} \in \mathbb{R}^p$. For simplicity, we assume all sequences have same number n of features. Further note that our scheme is agnostic to the type of features, i.e., the feature may be from a CNN or are hand-crafted.

Our goals are two-fold, namely (i) to learn a classifier decision boundary for every sequence in $\mathcal{X}^+$ that separates a fraction η of them from the features in $\mathcal{X}^-$ and (ii) to learn video level classifiers on the classes in the positive bags that are represented by the learned decision boundaries in (i). In the following, we will provide a multiple instance learning formulation for achieving (i), and a joint objective combining (i) and learning (ii). However, before presenting our scheme, we believe it may be useful to gain some insights into the main motivations for our scheme.

As alluded to above, given the positive and negative bags, our goal is to learn a linear (or non-linear) classification boundary that could separate the two bags with a classification accuracy of $\eta\%$ – this classification boundary is used as the descriptor for the positive bag. Referring to the conceptual illustration in Figure 3(a), when no negative bag is present, there are several ways to find a decision hyperplane in a max-margin setup that could potentially satisfy the η constraint. However, there is no guarantee that these hyperplanes are useful for action recognition. Instead, by introducing a negative bag, which is almost certainly to contain irrelevant features, it may be easier for the decision boundary to identify useless features from the rest; the latter containing useful action related features, as shown in Figure 3(b). This is precisely our intuitions for proposing this scheme.

3.2 Learning Decision Boundaries

As described above, our goal in this section is to generate a descriptor for each sequence $X^+ \in \mathcal{X}^+$; this descriptor we define to be the learned parameters of a hyperplane that separates the features $\mathbf{x}^+ \in X^+$ from all features in $\mathcal{X}^-$. We do not want to warrant that all $\mathbf{x}^+$ can be separated from $\mathcal{X}^-$ (since several of them may belong to a background class), however we assume that at least a fixed fraction η of them are classifiable. Mathematically, suppose the tuple (w_i, b_i) represents the parameters of a max-margin hyperplane separating some of the features in a positive
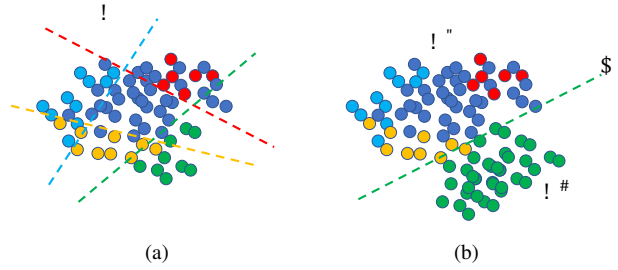

Fig. 3: An illustration of our overall idea. (a) the input data points, and the plausible hyperplanes satisfying some η constraint, (b) when noise $\mathcal{X}^-$ is introduced (green dots), it helps identify noisy features/data dimensions, towards producing a hyperplane w that classifies useful data from noise, while satisfying the η constraint.

bag X_i^+ from all features in $\mathcal{X}^-$, then we cast the following objective, which is a variant of the sparse MIL (SMIL) [74], normalized set kernel (NSK) [70], and $\propto$-SVM [72] formulations:

$$\underset{w_i \in \mathbb{R}^p, b_i \in \mathbb{R}, \zeta \geq 0}{\arg\min} P1(w_i, b_i) := \frac{1}{2}\|w_i\|^2 + C_1 \sum_{k=1}^{(M+1)n} \zeta_k \quad (1)$$

$$\text{subject to } \theta(\mathbf{x}; \eta)\left(w_i^T \mathbf{x} + b_i\right) \geq 1 - \zeta_k \quad (2)$$

$$\theta(\mathbf{x}; \eta) = -1, \forall \mathbf{x} \in \left\{X_i^+ \bigcup \mathcal{X}^-\right\} \backslash \hat{X}_i^+ \quad (3)$$

$$\theta(\hat{\mathbf{x}}; \eta) = 1, \forall \hat{\mathbf{x}} \in \hat{X}_i^+ \quad (4)$$

$$\left|\hat{X}_i^+\right| \geq \eta \left|X_i^+\right|. \quad (5)$$

In the above formulation, we assume that there is a subset $\hat{X}_i^+ \subset X_i^+$ that is classifiable, while the rest of the positive bag need not be, as captured by the ratio in (5). The variables ζ capture the non-negative slacks weighted by a regularization parameter C_1, and the function θ provides the label of the respective features. Unlike SMIL or NSK objectives, that assumes the individual features $\mathbf{x}$ are summable, our problem is non-convex due to the unknown set $\hat{X}^+$. However, this is not a serious deterrent to the usefulness of our formulation and can be tackled as described in the sequel and supported by our experimental results.

As our formulation is built on an SVM objective, we call this specific discriminative pooling scheme as *SVM pooling* and formally define the descriptor for a sequence as:

Definition 1 (SVM Pooling Desc.). Given a sequence X of features $\mathbf{x} \in \mathbb{R}^p$ and a negative dataset $\mathcal{X}^-$, we define the *SVM*

Pooling (SVMP) descriptor as $\mathrm{SVMP}(X) = [w, b]^T \in \mathbb{R}^{p+1}$, where the tuple (w, b) is obtained as the solution of problem $P1$ defined in (1).

3.3 Optimization Solutions

The problem $P1$ could be posed as a mixed-integer quadratic program (MIQP), which is unfortunately known to be in NP [75]. The problem $P1$ is also non-convex due to the proportionality constraint η, and given that the labels $\theta(\mathbf{x}; \eta)$ are unknown. Towards a practically useful approximate solution circumventing these difficulties, we present three optimization strategies below.

3.3.1 Exhaustive Enumeration

A naïve way to solve problem $P1$ could be to enumerate all the possible $\theta(\mathbf{x}; \eta)$ that meet a given η constraint, which reduces solving the problem $P1$ to the classical SVM problem for each instantiation of the plausible θ assignments. In such a setting, for a given sequence, we can rewrite (1) as:

$$\underset{w_i \in \mathbb{R}^p, b_i \in \mathbb{R}, \zeta \geq 0}{\arg\min} \frac{1}{2} \|w_i\|^2 + C_1 \sum_{k=1}^{(M+1)n} \zeta_k \\ + \max(0, 1 - \zeta_k - \theta(\mathbf{x}; \eta)(w_i^T \mathbf{x} + b_i)), \quad (6)$$

where the constraints are included via the hinge loss. Once these subproblems are solved, we could compare the optimal solutions for the various subsets of the positive bag, and pick the best solution with smallest objective value. As is apparent, this naïve strategy becomes problematic for longer sequences or when η is not suitably chosen.

3.3.2 Alternating algorithm

This is a variant of the scheme proposed in [72]. Instead of enumerating all possible $\theta(\mathbf{x}; \eta)$ as above, the main idea here is to fix $\theta(\mathbf{x}; \eta)$ or $[w, b]$ alternately and optimize the other. The detailed algorithm is shown in the Alg. 1.

Input: $X^+, \mathcal{X}^-, \eta$
Initialize θ according to η;
repeat
    Fix θ to solve $[w, b] \leftarrow \mathrm{SVM}(X^+, \mathcal{X}^-, \theta)$;
    Fix $[w, b]$ to solve θ:
    Reinitialize $\theta_i \leftarrow -1, \forall i \in (i, n)$;
    for $i = 1 \rightarrow n$ **do**
        Set $\theta_i \leftarrow 1$;
        record the reduction of Objective
    end
    Sort and select the top R reductions, $R = \eta n$;
    Get θ according to the sorting;
until *Reduction is smaller than a threshold (10^{-4});*
return $[w, b]$

Algorithm 1: Alternating solution to the MIL problem $P1$

In the Algorithm 1, fixing θ to solve $[w, b]$ is a standard SVMP problem as in the enumeration algorithm above. When fixing $[w, b]$ to solve θ, we apply a similar strategy as in [72]; i.e., to initialize all labels in θ as -1, and then to turn each θ_i to $+1$ and record the reduction in the objective. Next, we sort these reductions to get the top R best reductions, where $R = \eta n$. A higher reduction implies it may lead to a smaller objective. Next, these top R θ_i will be set to $+1$ in θ. While, there is no theoretical

guarantee for this scheme to converge to a fixed point, empirically we observe a useful convergence, which we limit via a suitable threshold.

3.3.3 Parameter-tuning algorithm

As is clear, both the above schemes may be computationally expensive in general. We note that the regularization parameter C_1 in (1) controls the positiveness of the slack variables ζ, thereby influencing the training error rate. A smaller value of C_1 allows more data points to be misclassified. If we make the assumption that useful features from the sequences are easily classifiable compared to background features, then a smaller value of C_1 could help find the decision hyperplane easily (further assuming the negative bag is suitably chosen). However, the correct value of C_1 depends on each sequence. Thus, in Algorithm (2), we propose a heuristic scheme to find the SVMP descriptor for a given sequence X^+ by iteratively tuning C_1 such that at least a fraction η of the features in the positive bag are classified as positive.

Input: $X^+, \mathcal{X}^-, \eta$
$C_1 \leftarrow \epsilon, \lambda > 1$;
repeat
    $C_1 \leftarrow \lambda C_1$;
    $[w, b] \leftarrow \arg\min_{w,b} \mathrm{SVM}(X^+, \mathcal{X}^-, C_1)$;
    $\hat{X}^+ \leftarrow \{\mathbf{x} \in X^+ \mid w^T \mathbf{x} + b \geq 0\}$;
until $\frac{|\hat{X}^+|}{|X^+|} \geq \eta$;
return $[w, b]$

Algorithm 2: Parameter-tuning solution for MIL problem $P1$

A natural question here is how optimal is this heuristic? Note that, each step of Algorithm (2) solves a standard SVM objective. Suppose we have an oracle that could give us a fixed value C for C_1 that works for all action sequences for a fixed η. As is clear, there could be multiple combinations of data points in $\hat{X}^+$ that could satisfy this η (as we explored in the Enumeration algorithm above). If $\hat{X}_p^+$ is one such $\hat{X}^+$. Then, $P1$ using $\hat{X}_p^+$ is just the SVM formulation and is thus convex. Different from previous algorithms, in Alg. 2, we adjust the SVM classification rate to η, which is easier to implement. Assuming we find a C_1 that satisfies the η-constraint using $P1$, then due to the convexity of SVM, it can be shown that the optimizing objective of P1 will be the same in both cases (exhaustive enumeration and our proposed regularization adjustment), albeit the solution $\hat{X}_p^+$ might differ (there could be multiple solutions).

3.4 Nonlinear Extensions

In problem $P1$, we assume a linear decision boundary generating SVMP descriptors. However, looking back at our solutions in Algorithms (1) and (2), it is clear that we are dealing with standard SVM formulations to solve our relaxed objectives. In the light of this, instead of using linear hyperplanes for classification, we may use nonlinear decision boundaries by using the kernel trick to embed the data in a Hilbert space for better representation. Assuming $\mathcal{X} = \mathcal{X}^+ \cup \mathcal{X}^-$, by the Representer theorem [76], it is well-known that for a kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$, the decision function f for the SVM problem P1 will be of the form:

$$f(.) = \sum_{\mathbf{x} \in X^+ \cup \mathcal{X}^-} \alpha_{\mathbf{x}} K(., \mathbf{x}), \quad (7)$$

where $\alpha_{\mathbf{x}}$ are the parameters of the non-linear decision boundaries. However, from an implementation perspective, such a direct kernelization may be problematic, as we will need to store the training set to construct the kernel. We avoid this issue by restricting our formulation to use only homogeneous kernels [77], as such kernels have explicit linear feature map embeddings on which a linear SVM can be trained directly. This leads to exactly the same formulations as in (1), except that now our features $\mathbf{x}$ are obtained via a homogeneous kernel map. In the sequel, we call such a descriptor a *nonlinear SVM pooling* (NSVMP) descriptor.

3.5 Temporally-Ordered Extensions

In the formulations we proposed above, there are no explicit constraints to enforce the temporal order of features in the SVMP descriptor. This is because, in the above formulations, we assume the features themselves capture the temporal order already. For example, the temporal stream in a two-stream model is already trained on a densely-sampled stack of consecutive optical flow frames. However, motivated by several recent works [14], [15], [17], [56], we extend our Equation (1) by including ordering constraints as:

$$w^T \mathbf{x}_j^{i+} + \delta \leq w^T \mathbf{x}_k^{i+}, \quad \forall j < k; \mathbf{x}_j^{i+}, \mathbf{x}_k^{i+} \in \hat{X}_i^+ \quad (8)$$

where we reuse the notation defined above and define $\delta > 0$ as a margin enforcing the order. In the sequel, we use this temporally-ordered variant of SVMP for our video representation. Note that with the ordering constraints enforced, it is difficult to use the enumerative or alternating schemes for finding the SVMP descriptors, instead we use Alg. 2 by replacing the SVM solver by a custom solver [78].

4 END-TO-END CNN LEARNING

In this section, we address the problem of training a CNN end-to-end with SVM pooling as an intermediate layer – the main challenge is to derive the gradients of SVMP for efficient back-propagation. This challenge is amplified by the fact that we use the parameters of the decision hyperplane to generate our pooling descriptor, this hyperplane is obtained via a non-differentiable argmin function (refer to (1)). However, fortunately, there is well-developed theory addressing such cases using the implicit function theorem [79], and several recent works towards this end in the CNN setting [80]. We follow these approaches and derive the gradients of SVMP below.

4.1 Discriminative Pooling Layer

In Figure 4, we describe two ways to insert the discriminative pooling layer into the CNN pipeline, namely (i) inserting SVMP at some intermediate layer and (ii) inserting SVMP at the end of the network just before the final classifier layer. While the latter pools smaller dimensional features, computing the gradients will be faster (as will be clear shortly). However, the last layer might only have discriminative action features alone, and might miss other spatio-temporal features that could be useful for discriminative pooling. This is inline with our observations in our experiments in Section 5 that suggest that applying discriminative pooling after pool5 or fc6 layers is significantly more useful than at the end of the fc8 layer. This choice of inserting the pooling layer between some intermediate layers of the CNN leads to the first choice. Figure 4 also provides the gradients that need to be computed for back-propagation in either case. The only new component of

this gradient is that for the argmin problem of pooling, which we derive below.

4.2 Gradients Derivations for SVMP

Assume a CNN f taking a sequence S as input. Let f_L denote the L-th CNN layer and let X_L denote the feature maps generated by this layer for all frames in S. We assume these features go into an SVMP pooling layer and produces as output a descriptor w (using a precomputed set of negative feature maps), which is then passed to subsequent CNN layers for action classification. Mathematically, let $g(z) = \arg\min_w \text{SVMP}(X_{L-1})$ define the SVM pooling layer, which we re-define using hinge-loss in the objective $f(z, w)$ as:

$$\text{SVMP}(X_{L-1}) = \frac{1}{2} \|w\|^2 + \frac{\lambda}{2} \sum_{z \in X_{L-1}} \max\left(0, \theta(z; \eta) w^T z - 1\right)^2.$$

As is by now clear, with regard to a CNN learning setup, we are dealing with a bilevel optimization problem here – that is, optimizing for the CNN parameters via stochastic gradient descent in the outer optimization, which requires the gradient of an argmin inner optimization with respect to its optimum, i.e., we need to compute the gradient of $g(z)$ with respect to the data z. By applying Lemma 3.3 of [80], this gradient of the argmin at an optimum SVMP solution w^* can be shown to be the following:

$$\nabla_z g(z)|_{w=w^*} = -\nabla_{ww} \text{SVMP}(X_{L-1})^{-1} \nabla_{zw} \text{SVMP}(X_{L-1}),$$

where the first term captures the inverse of the Hessian evaluated at w^* and the second term is the second-order derivative wrt z and w. Substituting for the components, we have the gradient at $w = w^*$ as:

$$-\left(\mathbf{I} + \lambda \sum_{\forall j: \theta_j w^T z_j > 1} (\theta_j z_j)(\theta_j z_j)^T\right)^{-1} \left[\lambda \sum_{\forall j: \theta_j w^T z_j > 1} \mathrm{D}\left(\theta_j^2 w^T z_j - \theta_j\right) + \theta_j^2 w z_j^T\right] \tag{9}$$

where for brevity, we use $\theta_j = \theta(z_j; \eta)$, and D is a diagonal matrix, whose i-th entry as $D_{ii} = \theta_i^2 w^T z_i - \theta_i$.

5 EXPERIMENTS

In this section, we explore the utility of discriminative pooling on several vision tasks, namely (i) action recognition using video and skeletal features, (ii) localizing actions in videos, (iii) image set verification, and (iv) recognizing dynamic texture videos. We introduce the respective datasets and experimental protocols in the next.

5.1 Datasets

HMDB-51 [22] and UCF-101 [23]: are two popular benchmarks for video action recognition. Both datasets consist of trimmed videos downloaded from the Internet. HMDB-51 has 51 action classes and 6766 videos, while UCF-101 has 101 classes and 13320 videos. Both datasets are evaluated using 3-fold cross-validation and mean classification accuracy is reported. For these datasets, we analyze different combinations of features on multiple CNN frameworks.

Charades [25]: is an untrimmed and multi-action dataset, containing 11,848 videos split into 7985 for training, 1863 for validation, and 2,000 for testing. It has 157 action categories, with several fine-grained categories. In the classification task, we follow the evaluation protocol of [25], using the output probability of the classifier to be the score of the sequence. In the detection task,

$$X_{L-2} \xrightarrow{} \boxed{f_{L-1}(X_{L-2}, Z_{L-1})} \xrightarrow{X_{L-1}} \boxed{SVMP(X_{L-1})} \xrightarrow{} w^*(X_{L-1}) \xrightarrow{} \boxed{f_L(w^*, Z_L)} \xrightarrow{X_L} loss(softmax(X_L), \theta)$$

$$\frac{\partial\, loss}{\partial Z_{L-1}} = \frac{\partial\, loss(\mathrm{softmax}(X_L), \theta)}{\partial X_L} \frac{\partial f_L(w^*, Z_L)}{\partial w^*} \frac{\partial SVMP(X_{L-1})}{\partial X_{L-1}} \frac{\partial f_{L-1}(X_{L-1}, Z_{L-1})}{\partial Z_{L-1}}$$

$$X_{L-2} \xrightarrow{} \boxed{f_{L-1}(X_{L-2}, Z_{L-1})} \xrightarrow{X_{L-1}} \boxed{f_L(X_{L-1}, Z_L)} \xrightarrow{X_L} \boxed{SVMP(X_L)} \xrightarrow{} w^*(X_L) \xrightarrow{} loss(softmax(w^*), \theta)$$

$$\frac{\partial\, loss}{\partial Z_L} = \frac{\partial\, loss(\mathrm{softmax}(w^*(X_L)), \theta)}{\partial w^*} \frac{\partial SVMP(X_L)}{\partial X_L} \frac{\partial f_L(X_{L-1}, Z_L)}{\partial Z_L}$$
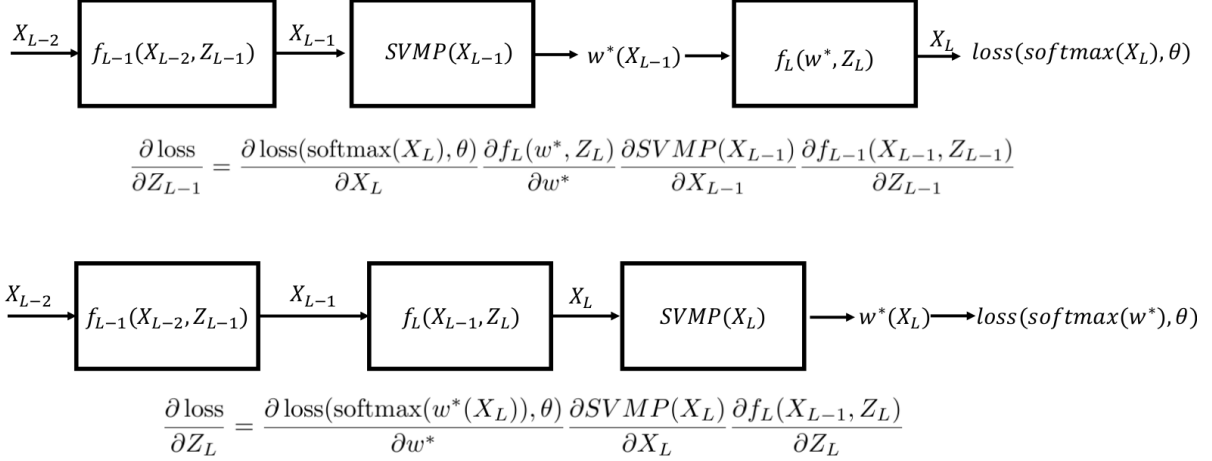
Fig. 4: Two possible ways to insert SVM pooling layer within a standard CNN architecture. In the first option (top), we insert the SVMP layer between fully connected layers, while in the latter we include it before the final classifier layer. The choice of $L-1$ layer for the former is arbitrary. We also show the corresponding partial gradients with respect to weights of the layer penultimate to the SVM pooling layer. Except for the gradients $\frac{\partial SVMP(X)}{\partial X}$, other gradients are the standard ones. Here Z_ℓ represents the weights of the ℓ-th layer of the network.

we follow the 'post-processing' protocol described in [81], which uses the averaged prediction score of a small temporal window around each temporal pivot. Using the provided two-stream fc7 feature[1], we evaluate the performance on both tasks using mean average precision (mAP) on the validation set.

Kinetics-600 [24]: is one of the largest dataset for action recognition. It consists of 500K trimmed video clips over 600 action classes with at least 600 video clips in each class. Each video clip is at least 10 seconds long with a single action class label. We apply our SVMP scheme on the CNN features (2048-D) extracted from the I3D network [3].

MSR Action3D [26] and NTU-RGBD [27]: are two popular action datasets providing 3D skeleton data. Specifically, MSR Action3D has 567 short sequences with 10 subjects and 20 actions, while NTU-RGBD has 56,000 videos and 60 actions performed by 40 people from 80 different view points. NTU-RGBD is by far the largest public dataset for depth-based action recognition. To analyze the performance of SVMP on non-linear features, we use a lie-algebra encoding of the skeletal data as proposed in [82] for the MSR dataset. As for NTU-RGBD, we use a temporal CNN as in [42], but uses SVMP instead of their global average pooling.

Public Figures Face Database (PubFig) [28]: contains 60,000 real-life images of 200 people. All the images are collected directly from the Internet without any post-processing, which make the images in each fold have large variations in lighting, backgrounds, and camera views. Unlike video-based datasets, PubFig images are non-sequential. To generate features, we fine-tune a ResFace-101 network [83] on this dataset and follow the evaluation protocol of [41].

YUP++ dataset [29]: is recent dataset for dynamic scene understanding. It has 20 scene classes, such as Beach, Fireworks, Waterfall, Railway, etc. There are 60 videos in each class. Half of the videos are recorded by a static camera and the other half by a moving camera. Accordingly, it is divided into two sub-datasets, *YUP++ moving camera* and *YUP++ static camera*. We use the latest Inception-ResNet-v2 model [84] to generate features (from last dense layer) from RGB frames and evaluate the performance according to the setting in [29], which use a 10/90 train-test ratio.
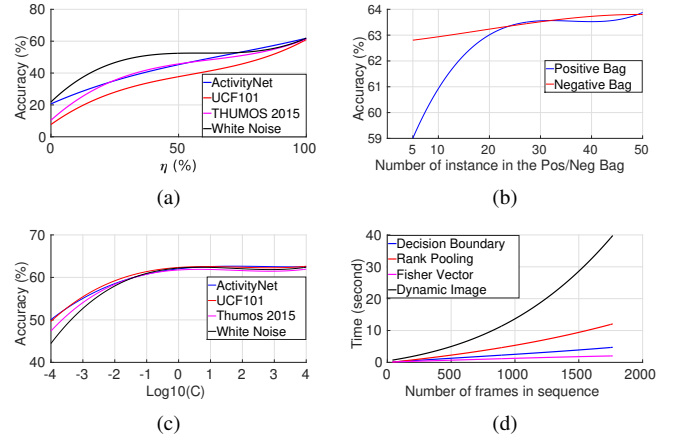
(a)

(b)

(c)

(d)

Fig. 5: Analysis of the parameters used in our scheme. All experiments use VGG features from fc6 dense layer. See text for details.

5.2 Parameter Analysis

In this section, we analyze the influence of each of the parameters in our scheme.

Selecting Negative Bags: An important step in our algorithm is the selection of the positive and negative bags in the MIL problem. We randomly sample the required number of frames (say, 50) from each sequence/fold in the training/testing set to define the positive bags. In terms of the negative bags, we need to select samples that are unrelated to the ones in the positive bags. We explored four different negatives in this regard to understand the impact of this selection. We compare our experiments on the HMDB-51 (and UCF101) datasets. Our considered the following choices for the negative bgs: clips from (ithe ActivityNet dataset [85] unrelated to HMDB-51, (ii) the UCF-101 dataset unrelated to HMDB-51, (iii) the Thumos Challenge background sequences[2], and (iv) synthesized random white noise image sequences. For (i) and (ii), we use 50 frames each from randomly selected videos, one from every unrelated class, and for (iv) we used 50 synthesized white noise images, and randomly generated stack of optical flow

---

1. http://vuchallenge.org/charades.html

2. http://www.thumos.info/home.html

images. Specifically, for the latter, we pass white noise RGB images to the same CNN models and extract the feature from the last fully-connected layer. As for hand-crafted or geometry features used in our other experiments (such as action recognition on human pose sequences), we directly use the white noise as the negative bag. As shown in Figure 5(a), the white noise negative is seen to showcase better performance for both lower and higher value of η parameter.

To understand this trend, in Figure 6, we show TSNE plots visualizing the deep CNN features for the negative bag variants. Given that the CNNs are trained on real-world image data and we extract features from the layer before the last linear layer, it is expected that these features be linearly separable (as seen in Figure 6(a) and 6(b)). However, we believe using random noise inputs may be activating combinations of filters in the CNN that are never co-activated during training, resulting in features that are highly non-linear (as Figure 6(c) shows). Thus, when requiring SVMP to learn linear/non-linear decision boundaries to classify video features against these "noise" features perhaps forces the optimizer to select those dimensions in the inputs (positive bag) that are more correlated with actions in the videos, thereby empowering the descriptor to be more useful for classification.

In Figure 7, we show the TSNE visualizations of SVMP descriptors comparing to average pooling and max pooling on data from 10-classes of HDMB-51 dataset. The visualization shows that SVMP leads to better separated clusters, substantiating that SVMP is learning discriminative representations.
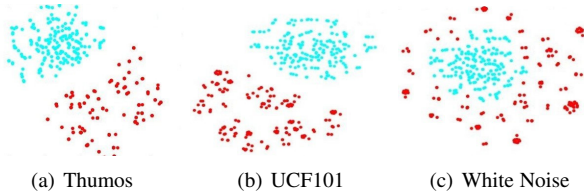

(a) Thumos   (b) UCF101   (c) White Noise

Fig. 6: T-SNE plots of positive (blue) and negative bags (red) when using negatives from: (a) Thumos, (b) UCF101, and (c) white noise.

Choosing Hyperparameters: The three important parameters in our scheme are (i) the η deciding the quality of an SVMP descriptor, (ii) $C_1 = C$ used in Algorithm 2 when finding SVMP per sequence, and (iii) sizes of the positive and negative bags. To study (i) and (ii), we plot in Figures 5(c) and 5(a) for HMDB-51 dataset, classification accuracy when C is increased from 10^{-4} to 10^4 in steps and when η is increased from 0-100% and respectively. We repeat this experiment for all the different choices of negative bags. As is clear, increasing these parameters reduces the training error, but may lead to overfitting. However, Figure 5(b) shows that increasing C increases the accuracy of the SVMP descriptor, implying that the CNN features are already equipped with discriminative properties for action recognition. However,
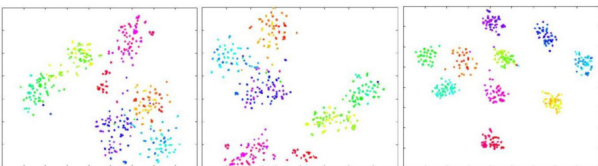

Fig. 7: T-SNE visualizations of SVMP and other pooling methods on sequences from the HMDB51 dataset (10 classes used). From left to right, Average Pooling, Max Pooling, and SVMP.

TABLE 1: Comparison between Algorithms 1 and 2 in HMDB-51 split-1.

Method	Accuracy	Avg. Time (sec)/Video
Alternating Algorithm (Alg. 1)	**69.8%**	2.4
Parameter-tuning Algorithm (Alg. 2)	69.5%	**0.2**

beyond $C = 10$, a gradual decrease in performance is witnessed, suggesting overfitting to bad features in the positive bag. Thus, we use $C = 10$ (and $\eta = 0.9$) in the experiments to follow. To decide the bag sizes for MIL, we plot in Figure 5(b), performance against increasing size of the positive bag, while keeping the negative bag size at 50 and vice versa; i.e., for the red line in Figure 5(b), we fix the number of instances in the positive bag at 50; we see that the accuracy raises with the cardinality of the negative bag. A similar trend, albeit less prominent is seen when we repeat the experiment with the negative bag size, suggesting that about 30 frames per bag is sufficient to get a useful descriptor.

Running Time: In Figure 5(d), we compare the time it took on average to generate SVMP descriptors for an increasing number of frames in a sequence on the UCF101 dataset. For comparison, we plot the running times for some of the recent pooling schemes such as rank pooling [14], [15] and the Fisher vectors [34]. The plot shows that while our scheme is slightly more expensive than standard Fisher vectors (using the VLFeat[3]), it is significantly cheaper to generate SVMP descriptors than some of the recent popular pooling methods. To be comparable, we use publicly available code of SVM in SVMP as well as in rank pooling.

5.3 Experiments on HMDB-51 and UCF-101

Following recent trends, we use a two-stream CNN model in two popular architectures, the VGG-16 and the ResNet-152 [10], [11]. For the UCF101 dataset, we directly use publicly available models from [10]. For the HMDB dataset, we fine-tune a two-stream VGG/ResNet model trained for the UCF101 dataset.

SVMP Optimization Schemes: We proposed three different optimization strategies for solving our formulation (Section 3.3). The enumerative solution is trivial and non-practical. Thus, we will only compare Algorithms 1 and 2 in terms of the performance and efficiency. In Table 1, we show the result between the two on fc6 features from a VGG-16 model. It is clear that the alternating solution is slightly better than parameter-tuning solution; however, is also more computationally expensive. Considering the efficiency, especially for the large-scale datasets, we use parameter-tuning solution in the following experiments.

SVMP on Different CNN Features: We generate SVMP descriptors from different intermediate layers of the CNN models and compare their performance. Specifically, features from each layer are used as the positive bags and SVMP descriptors computed using Alg. 1 against the chosen set of negative bags. In Table 2, we report results on split-1 of the HMDB dataset and find that the combination of fc6 and pool5 gives the best performance for the VGG-16 model, while pool5 features alone show good performance using ResNet. We thus use these feature combinations for experiments to follow.

Linear vs Non-Linear SVMP: We analyze the complementary nature of SVMP and its non-linear extension NSVMP (using a homogeneous kernel) on HMDB-51 and UCF-101 split1. The results are provided in Table 3, and clearly show that the combination leads to significant improvements consistently on both datasets.

3. http://www.vlfeat.org/

TABLE 2: Comparison of SVMP descriptors using various CNN Features on HMDB split-1.

Feature/ model	Accuracy independently	Accuracy when combined with:
pool5 (vgg-16)	57.9%	**63.8%** (fc6)
fc6 (vgg-16)	63.3%	-
fc7 (vgg-16)	56.1%	57.1% (fc6)
fc8 (vgg-16)	52.4%	58.6% (fc6)
softmax (vgg-16)	41.0%	46.2% (fc6)
pool5 (ResNet-152)	**69.5%**	-
fc1000 (ResNet-152)	61.1%	68.8% (pool5)

TABLE 3: Comparison between SVMP and NSVMP on split-1.

	HMDB-51		UCF-101	
	VGG	ResNet	VGG	ResNet
linear-SVMP	63.8%	69.5%	91.6%	92.2%
nonlinear-SVMP	64.4%	69.8%	92.0%	93.1%
Combination	**66.1%**	**71.0%**	**92.2%**	**94.0%**

End-to-End Learning and Ordered-SVMP: In Table 4[4], we compare to the end-to-end learning setting as described in Section 4. For end-to-end learning, we insert our discriminative pooling layer after the 'fc6' layer in VGG-16 model and the 'pool5' layer in ResNet model. We also present results when using the temporal ordering constraint (TC) into the SVMP formulation to build the ordered-SVMP. From the results, it appears that although the soft-attention scheme performs better than average pooling, it is inferior to SVMP itself; which is unsurprising given it does not use a max-margin optimization. Further, our end-to-end SVMP layer is able to achieve similar (but slightly inferior) performance to SVMP, which perhaps is due to the need to approximate the Hessian. As the table shows, we found that the temporal ranking is indeed useful for improving the performance of naïve SVMP. Thus, in the following experiments, we use SVMP with temporal ranking for all video-based tasks.

TABLE 4: Comparison to standard pooling methods on split-1. TC is short for Temporal Constraint, E2E is short for end-to-end learning.

	HMDB-51		UCF-101	
	VGG	ResNet	VGG	ResNet
Spatial Stream-AP [10], [86]	47.1%	46.7%	82.6%	83.4%
Spatial Stream-SVMP	58.3%	57.4%	85.7%	87.6%
Spatial Stream-SVMP(E2E)	56.4%	55.1%	83.2%	85.7%
Spatial Stream-SVMP+TC	**59.4%**	**57.9%**	**86.6%**	**88.9%**
Temporal Stream-AP [10], [86]	55.2%	60.0%	86.3%	87.2%
Temporal Stream-SVMP	61.8%	65.7%	88.2%	89.8%
Temporal Stream-SVMP(E2E)	58.3%	63.2%	87.1%	87.8%
Temporal Stream-SVMP+TC	**62.6%**	**67.1%**	**88.8%**	**90.9%**
Two-Stream-AP [10], [86]	58.2%	63.8%	90.6%	91.8%
Two-Stream-SVMP	66.1%	71.0%	92.2%	94.2%
Two-Stream-SVMP(E2E)	63.5%	68.4%	90.6%	92.3%
Two-Stream-SVMP+TC	**67.2%**	**71.3%**	**92.5%**	**94.8%**

SVMP Image: In Figure 8, we visualize SVMP descriptor when applied directly on raw video frames. We compare the resulting image against those from other schemes such as the dynamic images of [14]. It is clear that SVMP captures the essence of action dynamics in more detail. To understand the action information present in these images, we trained an action classifier directly on these images, as is done on Dynamic images in [14]. We use the BVLC CaffeNet [87] as the CNN – same the one used in [14]. The results are shown in the Table 5 on split-1 of JHMDB (a subset of HMDB-51, containing 21 classes) and UCF-101. As is clear, SVMP images are seen to outperform [14] by a significant

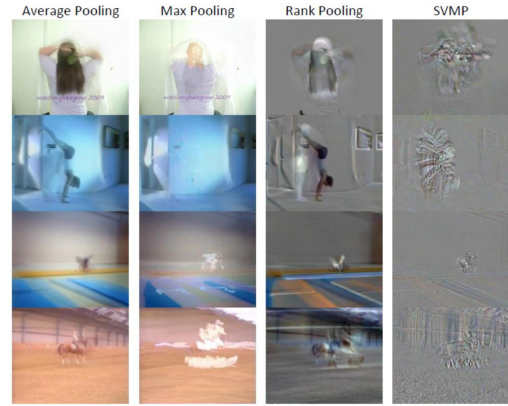4. All experiments in Table 4 use the same input features.

Fig. 8: Visualizations of various pooled descriptors.

TABLE 5: Recognition rates on split-1 of JHMDB and UCF-101.

Datasets	JHMDB	UCF-101
Mean image	31.3%	52.6%
Max image	28.6%	48.0%
Dynamic image [14]	35.8%	57.2%
SVMP image	**45.8%**	**65.4%**

margin, suggesting that SVMP captures more discriminative and useful action-related features. Howeer, we note that in contrast to dynamic images, our SVMP images do not intuitively look like motion images; this is perhaps because our scheme captures different information related to the actions, and we do not use smoothing (via running average) when generating them. The use of random noise features as the negative bag may be adding additional artifacts.

5.4 Action Recognition at Large Scale

Kinetics-600 is one the largest state-of-the-art dataset for action recognition on trimmed videos. For this experiment, we use the I3D network [3] (using the Inception-V3 architecture), as the baseline for feature generator. This model is pre-trained on ImageNet dataset [40] and stacks 64 continuous frames as inputs. Specifically, we extract the CNN features from the second last layer (Mix5c) and apply average pooling to reshape the feature from 4 x 7 x 7 x 1024 into 1024-D vector for each 64-chunk of RGB frames. For each video clip, we use a sliding window to generate a sequence of such features with a window size of 64 and a temporal stride of 8 frames. Then, we apply our proposed SVMP to generate video descriptors for action recognition. In Table 6, we make comparisons with the baseline result on the validation set of Kinetics-600, and indicates that SVMP can bring clear improvements even on the large-scale setting.

5.5 Action Recognition/Detection in untrimmed videos

We ues the Charades untrimmed dataset for this task. We use the publicly available two-stream VGG features from the fc7 layer for this dataset. We trained our models on the provided training set (7985 videos), and report results (mAP) on the provided validation set (1863 videos) for the tasks of action classification and

TABLE 6: Comparisons on Kinetics-600 dataset using I3D feature.

Method	Accuracy
AP [88]	71.9%
MP	67.8%
SVMP	**73.5%**

TABLE 7: Comparisons on Charades dataset.

Tasks	AP	MP	SVMP
Classification (mAP)	14.2%	15.3	**26.3%**
Detection (mAP)	10.9%	9.2	**15.1%**

TABLE 8: Accuracy comparison on different subsets of HMDB-51(H) and UCF-101(U) split-1 using I3D+ features.

Min # of frames	1	80	140	180	260
# of classes (H)	51	49	27	21	12
# of classes (U)	101	101	95	82	52
I3D (H)	79.6%	81.8%	84.1%	78.0%	77.3%
SVMP (H)	**80.0%**	**82.9%**	**84.8%**	**85.1%**	**86.8%**
I3D (U)	98.0%	98.0%	98.0%	95.9%	93.8%
SVMP (U)	**98.4%**	**98.9%**	**99.3%**	**98.5%**	**97.3%**

detection. In the classification task, we concatenate the two-stream features and apply a sliding window pooling scheme to create multiple descriptors. Following the evaluation protocol in [25], we use the output probability of the classifier to be the score of the sequence. In the detection task, we consider the evaluation method with post-processing proposed in [81], which uses the averaged prediction score of a temporal window around each temporal pivots. Instead of average pooling, we apply the SVMP. From Table 7, it is clear that SVMP improves performance against other pooling schemes by a significant margin; the reason for this is perhaps the following. During training, we use trimmed video clips, however, when testing, we extract features from every frame/clip in the untrimmed test video. As the network has seen only action-related frames during training, features from background frames may result in arbitrary predictions; and average pooling or max pooling on those features would hurt performance. When optimizing the binary classification problem between positive and negative bags for SVMP, the decision boundary would capture the most discriminative data support, leading to better summary of the useful features and leading to improved performance.

5.6 SVMP Evaluation on Other Tasks

In this section, we provide comprehensive evaluations justifying the usefulness of SVMP on non-video datasets and non-action tasks. We consider experiments on images sets recognition, skeleton-sequence based action recognition, and dynamic texture understanding.

MSR Action3D: In this experiment, we explore the usefulness of SVMP on non-linear geometric features. Specifically, we chose the scheme of Vemulapalli et al. [82] as the baseline that generates Lie algebra based skeleton encodings for action recognition. While they resort to a dynamic time warping kernel for the subsequent encoded skeleton pooling, we propose to use SVMP instead. We use the random noise with the dataset mean and deviation as the negative bag, which achieve better performance.

NTU-RGBD: On this dataset, we apply our SVMP scheme on the skeleton-based CNN features. Specifically, we use [42] as the baseline, which applies a temporal CNN with residual connections on the vectorized 3D skeleton data. We swap the global average pooling layer in [42] by SVM pooling layer. For the evaluation, we adopt the official cross-view and cross-subject protocols. What's interesting here is we try to explore whether the dimension of the feature point would affect the SVMP performance. During the SVMP, we use feature points with dimension from 150 to 4096. It seems only the number of data points would affect the performance of SVMP (from Charades dataset experiment), and it is not sensitive for the dimensionality.

PubFig: In this task, we evaluate the use of SVMP for image set representation. We follow the evaluation setting in [41] and create the descriptor for the training and testing by applying SVMP over ResFace-101 [83] features from every image in the PubFig dataset. Unlike the video-based tasks, all input features in this setting are useful and represent the same person; however their styles vary significantly, which implies the CNN features may be very different even if they are from the same person. This further demands that SVM pooling would need to find discriminative

TABLE 9: Comparison to the state of the art in each dataset, following the official evaluation protocol for each dataset.

HMDB-51 & UCF-101 (accuracy over 3 splits)		
Method	HMDB-51	UCF-101
Temporal segment networks [20]	69.4%	94.2%
AdaScan [89]	54.9%	89.4%
AdaScan + IDT + C3D [89]	66.9%	93.2%
ST ResNet [86]	66.4%	93.4%
ST ResNet + IDT [86]	70.3%	94.6%
ST Multiplier Network [4]	68.9%	94.2%
ST Multiplier Network + IDT [4]	72.2%	94.9%
Hierarchical rank pooling [90]	65.0%	90.7%
Two-stream I3D [3]	66.4%	93.4%
Two-stream I3D+ (Kinetics 300k) [3]	80.7%	98.0%
Ours (SVMP)	71.3%	94.6%
Ours (SVMP+IDT)	**72.6%**	**95.0%**
Ours (I3D+)	**81.8%**	**98.5%**
Kinetics-600		
Method	Accuracy	
I3D RGB [88]	71.3%	
Second-order Pooling [18]	54.7%	
Ours (SVMP)	**73.5%**	
Charades (mAP)		
Method	Classification	Detection
Two-stream [91]	14.3%	10.9%
ActionVlad + IDT [92]	21.0%	-
Asynchronous Temporal Fields [81]	22.4%	12.8%
Ours (SVMP)	26.3%	15.1%
Ours (SVMP+IDT)	**27.4%**	**16.3%**
MSR-Action3D		
Method	Accuracy	
Lie Group [82]	92.5%	
ST-LSTM + Trust Gate [93]	94.8%	
Ours (SVMP)	**95.5%**	
NTU-RGBD		
Method	Cross-Subject	Cross-View
Res-TCN [42]	74.3%	83.1%
ST-LSTM + Trust Gate [93]	69.2%	77.7%
Ours (SVMP)	**79.4%**	**87.6%**
PubFig		
Method	Accuracy	
Deep Reconstruction Models [41]	89.9%	
ESBC [94]	98.6%	
Ours (SVMP)	**99.3%**	
YUP++		
Method	Stationary	Moving
Temporal Residual Networks [29]	92.4%	81.5%
Ours (SVMP)	**92.9%**	**84.0%**

dimensions in the features that are correlated and invariant to the person identity.

YUP++: To investigate our SVMP scheme on deeper architectures, we use features from the latest Inception-ResNet-v2 model [84], which has achieved the state-of-the-art performance on the 2015 ILSVRC challenge. Specifically, we extract the RGB frames from videos and divide them into training and testing split according to the setting in [29] (using a 10/90 train test ratio). Like the standard image-based CNNs, the clip level label is used to train the network on every frame.

5.7 Comparisons to the State of the Art

In Table 9, we compare our best results against the state-of-the-art on each dataset using the standard evaluation protocols. For a fair comparison, we also report on SVMP combined with hand-crafted features (IDT-FV) [95] for HMDB-51. Our scheme outperforms other methods on all datasets by 1–4%. For example, on HMDB-51, our results are about 2-3% better than the next best method without IDT-FV. On Charades, we outperform previous methods by about 3% while faring well on the detection task against [81]. We also demonstrate significant performance (about 3-4%) improvement on NTU-RGBD and marginally better performance on MSR datasets on skeleton-based action recognition. Our results are superior (by 1-2%) on the PubFig and YUP++ datasets.

We further analyze the benefits of combining I3D+ with SVMP (instead of their proposed average pooling) on both HMDB-51 and UCF-101 datasets using the settings in [3]. However, we find that the improvement over average pooling in I3D+ is not significant; which we believe is because learning the SVMP descriptor needs to solve a learning problem implicitly, requiring sufficient number of training samples, i.e., number of frames in the sequence. The I3D network uses 64-frame chunks as one sample, thereby reducing the number of samples for SVMP, leading to sub-optimal learning. We analyze this hypothesis in Table 8; each column in this table represents performances on a data subset, filtered as per the minimum number of frames in their sequences. As is clear from the table, while SVMP performs on par with I3D+ when the sequences are shorter, it demonstrates significant benefits on subsets having longer sequences.

6 CONCLUSION

In this paper, we presented a simple, efficient, and powerful pooling scheme – SVM pooling – for video representation learning. We cast the pooling problem in a multiple instance learning framework, and seek to learn useful decision boundaries on video features against background/noise features. We provide an efficient scheme that jointly learns these decision boundaries and the action classifiers on them. Extensive experiments were showcased on eight challenging benchmark datasets, demonstrating state-of-the-art performance. Given the challenging nature of these datasets, we believe the benefits afforded by our scheme is a significant step towards the advancement of recognition systems designed to represent sets of images or videos.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*. IEEE, 2017.

[3] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *CVPR*, July 2017.

[4] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *CVPR*, 2017.

[5] J.-F. Hu, W.-S. Zheng, J. Pan, J. Lai, and J. Zhang, "Deep bilinear learning for rgb-d action recognition," in *ECCV*, September 2018.

[6] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, "Temporal relational reasoning in videos," *ECCV*, 2018.

[7] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014.

[8] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *PAMI*, vol. 35, no. 1, pp. 221–231, 2013.

[9] M. Monfort, B. Zhou, S. A. Bargal, A. Andonian, T. Yan, K. Ramakrishnan, L. Brown, Q. Fan, D. Gutfreund, C. Vondrick *et al.*, "Moments in time dataset: one million videos for event understanding," 2018.

[10] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *CVPR*, 2016.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *CVPR*, 2015.

[13] Y. Wang, J. Song, L. Wang, L. Van Gool, and O. Hilliges, "Two-stream sr-cnns for action recognition in videos," in *BMVC*, 2016.

[14] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, "Dynamic image networks for action recognition," in *CVPR*, 2016.

[15] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *CVPR*, 2015.

[16] A. Cherian, S. Sra, S. Gould, and R. Hartley, "Non-linear temporal subspace representations for activity recognition," in *CVPR*, 2018.

[17] A. Cherian, B. Fernando, M. Harandi, and S. Gould, "Generalized rank pooling for activity recognition," in *CVPR*, 2017.

[18] A. Cherian and S. Gould, "Second-order temporal pooling for action recognition," *arXiv preprint arXiv:1704.06925*, 2017.

[19] A. Cherian, P. Koniusz, and S. Gould, "Higher-order pooling of cnn features via kernel linearization for action recognition," in *WACV*, 2017.

[20] 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.

[21] K. Schindler and L. Van Gool, "Action snippets: How many frames does human action recognition require?" in *CVPR*, 2008.

[22] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *ICCV*, 2011.

[23] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012.

[24] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[25] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in homes: Crowdsourcing data collection for activity understanding," in *ECCV*, 2016.

[26] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *CVPRW*, 2010.

[27] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," in *CVPR*, 2016.

[28] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *ICCV*, 2009.

[29] C. Feichtenhofer, A. Pinz, and R. Wildes, "Temporal residual networks for dynamic scene recognition," in *CVPR*, 2017.

[30] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *Image and vision computing*, vol. 60, pp. 4–21, 2017.

[31] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[32] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[33] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *CVPR*, 2011.

[34] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *ICCV*, 2013.

[35] S. Sadanand and J. J. Corso, "Action bank: A high-level representation of activity in video," in *CVPR*, 2012.

[36] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *ICCV*, 2003, p. 1470.

[37] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *ECCV*, 2010.

[38] H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *TPAMI*, vol. 34, no. 9, pp. 1704–1716, 2012.

[39] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Computer Vision and Image Understanding*, vol. 150, pp. 109–125, 2016.

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

[41] M. Hayat, M. Bennamoun, and S. An, "Deep reconstruction models for image set classification," *PAMI*, vol. 37, no. 4, pp. 713–727, 2015.

[42] T. S. Kim and A. Reiter, "Interpretable 3D human action analysis with temporal convolutional networks," 2017.

[43] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Human Behavior Understanding*, 2011, pp. 29–39.

[44] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venu-gopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR*, 2015.
[45] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *CVPR*, 2015.
[46] Q. Li, Z. Qiu, T. Yao, T. Mei, Y. Rui, and J. Luo, "Action recognition by learning deep multi-granular spatio-temporal video representation," in *ICMR*, 2016.
[47] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms." in *ICML*, 2015, pp. 843–852.
[48] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *CVPR*, 2015.
[49] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013.
[50] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *ICCV*, 2015.
[51] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *CVPR*, 2018.
[52] Y. Zhou, X. Sun, Z.-J. Zha, and W. Zeng, "Mict: Mixed 3d/2d convolutional tube for human action recognition," in *CVPR*, 2018.
[53] L. Wang, W. Li, W. Li, and L. Van Gool, "Appearance-and-relation networks for video classification," 2017.
[54] A. Cherian and S. Gould, "Second-order temporal pooling for action recognition," *IJCV*, 2018.
[55] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "Action-VLAD: Learning spatio-temporal aggregation for action classification," in *CVPR*, 2017.
[56] J. Wang, A. Cherian, and F. Porikli, "Dynamic pooling for complex event recognition," in *WACV*, 2017.
[57] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV*, 2011.
[58] G. Willems, J. H. Becker, T. Tuytelaars, and L. J. Van Gool, "Exemplar-based action recognition in video." in *BMVC*, 2009.
[59] J. Zepeda and P. Perez, "Exemplar svms as visual feature encoders," in *CVPR*, 2015.
[60] R. G. Cinbis, J. Verbeek, and C. Schmid, "Weakly supervised object localization with multi-fold multiple instance learning," *PAMI*, vol. 39, no. 1, pp. 189–203, 2017.
[61] W. Li and N. Vasconcelos, "Multiple instance learning for soft bags via top instances," in *CVPR*, 2015.
[62] J. Wu, Y. Yu, C. Huang, and K. Yu, "Deep multiple instance learning for image classification and auto-annotation," in *CVPR*, 2015.
[63] Y. Yi and M. Lin, "Human action recognition with graph-based multiple-instance learning," *Pattern Recognition*, vol. 53, pp. 148–162, 2016.
[64] D. Zhang, D. Meng, C. Li, L. Jiang, Q. Zhao, and J. Han, "A self-paced multiple-instance learning framework for co-saliency detection," in *ICCV*, 2015.
[65] S. Satkin and M. Hebert, "Modeling the temporal extent of actions," in *ECCV*, 2010.
[66] S. Nowozin, G. Bakir, and K. Tsuda, "Discriminative subsequence mining for action classification," in *ICCV*, 2007.
[67] W. Li, Q. Yu, A. Divakaran, and N. Vasconcelos, "Dynamic pooling for complex event recognition," in *ICCV*, 2013.
[68] C. Sun and R. Nevatia, "Discover: Discovering important segments for classification of video events and recounting," in *CVPR*, 2014.
[69] A. Vahdat, K. Cannons, G. Mori, S. Oh, and I. Kim, "Compositional models for video event detection: A multiple kernel learning latent variable approach," in *ICCV*, 2013.
[70] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels." in *ICML*, 2002.
[71] K.-T. Lai, F. X. Yu, M.-S. Chen, and S.-F. Chang, "Video event detection by inferring temporal instance labels," in *CVPR*, 2014.
[72] F. X. Yu, D. Liu, S. Kumar, T. Jebara, and S.-F. Chang, "$\propto$ svm for learning with label proportions," *arXiv preprint arXiv:1306.0886*, 2013.
[73] J. Wang, A. Cherian, F. Porikli, and S. Gould, "Video representation learning using discriminative pooling," in *CVPR*, 2018.
[74] R. C. Bunescu and R. J. Mooney, "Multiple instance learning for sparse positive bags," in *ICML*, 2007.
[75] R. Lazimy, "Mixed-integer quadratic programming," *Mathematical Programming*, vol. 22, no. 1, pp. 332–349, 1982.
[76] A. J. Smola and B. Schölkopf, *Learning with kernels.* Citeseer, 1998.
[77] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *PAMI*, vol. 34, no. 3, pp. 480–492, 2012.
[78] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a Matlab toolbox for optimization on manifolds," *Journal of Machine Learning Research*, vol. 15, pp. 1455–1459, 2014.
[79] A. L. Dontchev and R. T. Rockafellar, "Implicit functions and solution mappings," *Springer Monogr. Math.*, 2009.
[80] 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," 2016.
[81] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta, "Asynchronous temporal fields for action recognition," in *CVPR*, 2017.
[82] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3d skeletons as points in a lie group," in *CVPR*, 2014.
[83] I. Masi, A. Tran, T. Hassner, J. T. Leksut, and G. Medioni, "Do We Really Need to Collect Millions of Faces for Effective Face Recognition?" in *ECCV*, 2016.
[84] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, 2017.
[85] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *CVPR*, 2015.
[86] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal residual networks for video action recognition," in *NIPS*, 2016.
[87] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding." ACM, 2014.
[88] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, "A short note about kinetics-600," 2018.
[89] A. Kar, N. Rai, K. Sikka, and G. Sharma, "Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos," in *CVPR*, 2017.
[90] B. Fernando, P. Anderson, M. Hutter, and S. Gould, "Discriminative hierarchical rank pooling for activity recognition," in *CVPR*, 2016.
[91] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013.
[92] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "Actionvlad: Learning spatio-temporal aggregation for action classification," in *CVPR*, 2017.
[93] J. Liu, A. Shahroudy, D. Xu, A. C. Kot, and G. Wang, "Skeleton-based action recognition using spatio-temporal lstm network with trust gates," 2017.
[94] M. Hayat, S. H. Khan, and M. Bennamoun, "Empowering simple binary classifiers for image set based face recognition," *IJCV*, pp. 1–20, 2017.
[95] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *IJCV*, vol. 103, no. 1, pp. 60–79, 2013.
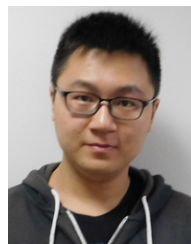
Jue Wang is a PhD student with the Research School of Engineering at the Australian National University since 2016. He is also associated with CSIRO's Data61 in Australia. From 2010-2014, he received his double bachelor degree (honors) in Electronic Engineering from Australian National University and Beijing Institute of Technology. His research interest are in the area of computer vision and machine learning.

Anoop Cherian is a Research Scientist with Mitsubishi Electric Research Labs (MERL) Cambridge, MA and an Adjunct Researcher affiliated to the Australian Centre for Robotic Vision (ACRV) at the Australian National University. Previously, he was a Postdoctoral Researcher in the LEAR team at INRIA at Grenoble. He received his B.Tech (honors) degree in computer science and Engineering from the National Institute of Technology, Calicut, India in 2002, his M.S. and Ph.D. degrees in computer science from the University of Minnesota, Minneapolis in 2010 and 2013 respectively. His research interests lie in the areas of computer vision and machine learning.