Video Representation Learning Using Discriminative Pooling

Jue Wang[1,3] Anoop Cherian[2,3,4] Fatih Porikli[2,3] Stephen Gould[2,3]
[1]Data61/CSIRO, [2]Australian Centre for Robotic Vision
[3]The Australian National University, Canberra, Australia, [4]MERL Cambridge, MA
firstname.lastname@anu.edu.au

Abstract

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 this end, we learn a (nonlinear) hyperplane that separates this unknown, yet discriminative, feature from the rest. Applying multiple instance learning in a large-margin 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 serve as robust representations for pooling of the features. We formulate a joint objective and an efficient solver that learns these hyperplanes per video and the corresponding action classifiers over the hyperplanes. Our pooling scheme is end-to-end trainable within a deep framework. We report results from experiments on three benchmark datasets spanning a variety of challenges and demonstrate state-of-the-art performance across these tasks.*

1. Introduction

We are witnessing an astronomical increase of video data on the web. 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 (most notably on object detection and image tagging) and is now pushing forward the state-of-the-art in action recognition and video understanding. However, current solutions 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 [10, 11, 12, 13, 18, 41, 42, 54].

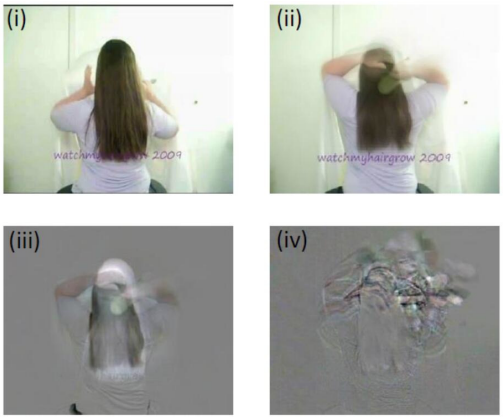

Figure 1. A visualization of discriminative pooling applied to RGB frames in a sequence. (i) a sample frame, (ii) average pooling all frames, (iii) dynamic image by rank pooling [2], and (iv) our SVM pooling. Our representation captures more details of the actions as it learns to discriminate parts of the foreground against a background set. In this case, we used the average-pooled frames as the background.

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. However, video data can be of arbitrary length and scaling up image-based CNN architectures to yet another dimension of complexity is not an easy task as the number of parameters in such models will be significantly higher. This demands more advanced computational infrastructures and greater quantities of clean training data [5]. Instead, 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-stream model [13, 41, 42, 53, 55], the CNNs are

trained to independently predict actions from short video clips (consisting of single frames or stacks of about ten optical flow frames); these predictions are then pooled to generate a prediction for the full sequence – typically using average/max pooling or classifying using a linear SVM. While average pooling gives equal weights to all the predictions, max pooling is sensitive to outliers, and a classifier may be confused by predictions from background actions. Several works try to tackle this problem by using different pooling strategies [2, 14, 53, 54, 60], which achieve some improvement compared with the baseline algorithm.

To this end, we observe that not all predictions on the short video snippets are equally informative, yet some of them must be [36]. This allows us to cast the problem in a multiple instance learning (MIL) framework, where we assume that some of the features in one sequence are indeed useful, while the rest are not. We assume all the CNN features from a sequence (containing both the good and bad features) to represent a positive bag, while features from the known background or noisy frames as a negative bag. We then formulate a binary classification problem of separating as many good features as possible in the positive bag using a discriminative classifier. The decision boundary of the classifier thus learned is then used as a descriptor for the entire video sequence, which we call the SVM Pooled (SVMP) descriptor. Subsequently, the SVMP descriptor is used in an action classification setup. We also provide a joint objective that learns both the SVMP descriptors and the action classifiers, which generalizes the applicability of our method to both CNN features and hand-crafted ones. In a pure CNN setup, our pooling method can be implemented alongside the rest of the CNN layers and trained in an end-to-end manner.

Compared to other popular pooling schemes, our proposed method offers several benefits: First, it produces a compact representation of a video sequence of arbitrary length by characterizing the classifiability of its features against a background set. Second, it is robust to classifier outliers thanks to the SVM formulation. Last, it is computationally efficient. To provide intuitions, in Figure 1, we provide a visualization of our descriptor applied directly on video frames. As is clear, SVMP captures the essence of action dynamics in more detail in comparison to prior works.

We provide extensive experimental evidence on various datasets for different tasks, such as action recognition/anticipation/detection on the HMDB-51 and Charades datasets and skeleton-based action recognition on NTU-RGBD. We outperform baseline results on these datasets by a significant margin (between 3–11%) and beat all the previously reported results as well (between 1–4%).

To set the stage for introducing our method, we briefly review relevant literature on prior works in the next section.

2. Related Work

Traditional methods for video action recognition typically use hand-crafted local features, such as dense trajectories, HOG, HOF, etc. [51], or mid-level representations on them, such as Fisher Vectors [34]. With the resurgence of deep learning methods for object recognition [21], 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 [11, 12, 18, 20, 41]. Although the architecture of these networks are different, the core idea is to embed the video data into a semantic feature space, and then recognize the actions either by aggregating the individual features per frame using some statistic (such as max or average) or directly training a CNN based end-to-end classifier [11]. 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. This problem may be tackled using recurrent models [1, 8, 9, 25, 45, 60], however such models are usually harder to train [32]. Another promising direction is to use 3D convolutional filters [5, 47], but would need more parameters and large amounts of clean data for pretraining. In contrast to all these approaches, we look at the problem from that of choosing the correct set of frames automatically that are discriminative in recognizing the actions. A scheme similar to ours is the recent work of Wang et al., [54], however they use manually-defined video segmentation for equally-spaced snippet sampling.

Typically, pooling schemes consolidate input data into compact representations. Instead, we use the parameters of the data modeling function, i.e., the SVM decision boundary, as our representation. Note that such a hyperplane is of the same dimensionality as the data and well-known as a weighted combination of each data point, where the weight captures how discriminative each point is. There have been other recent works that use parameters of machine learning algorithms for the purpose of pooling, such as rank pooling [14], generalized rank pooling [6], dynamic images [2] and dynamic flow [52]. However, while 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 preselected (but arbitrary) bag of negative features. In this respect, our pooling scheme is also different from Exemplar-SVMs [29, 56, 61] that learns feature filters per data sample and then use these filters for feature extraction.

An important component of our scheme is the MIL scheme, which is a popular data selection technique [7, 26, 57, 58, 62]. In the context of action recognition, schemes

similar in motivation have been suggested before. For example, Satkin and Hebert [35] explore the effect of temporal cropping of videos to regions of actions; however, assumes these regions are continuous. Nowozin et al. [31] represent videos as sequences of discretized spatio-temporal 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. [27] 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 [46] propose a latent variable based model to explicitly localize discriminative video segments where events take place. Vahdat et al. present a compositional model in [48] for video event detection 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 [15] and reduce the formulation to standard SVM setup which can be solved rapidly. In the $\propto$-SVMs of Yu et al., [23, 59], the positive bags are assumed to have a fixed fraction of positives, which is a criterion we also assume in our framework. However, our optimization setup and our goals are different. Specifically, our goal is to learn a video representation for recognition, while [59] tackles the problem of action detection.

3. Proposed Method

In this section, we describe our method for learning SVMP descriptors and the action classifiers. The overall pipeline is illustrated in Figure 2.

Let us assume we are given a dataset of N video sequences $\mathcal{X}^+ = \left\{X_1^+, X_2^+, ..., X_N^+\right\}$, where each X_i^+ is sequence with a set of frame level features, *i.e.*, $X_i^+ = \left\{\mathbf{x}_1^{i+}, \mathbf{x}_2^{i+}, ..., \mathbf{x}_n^{i+}\right\}$, each $\mathbf{x}_k^{i+} \in \mathbb{R}^p$. We assume that each X_i^+ is associated with an action class label $y_i^+ \in \{1, 2, ..., d\}$. 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 $\mathcal{X}^- = \left\{X_1^-, X_2^-, ...X_M^-\right\}$ belonging to actions different from those in $\mathcal{X}^+$, where each $X_j^- = \left\{\mathbf{x}_1^{j-}, \mathbf{x}_2^{j-}, ..., \mathbf{x}_n^{j-}\right\}$ are the features associated with a negative bag, each $\mathbf{x}_k^{j-} \in \mathbb{R}^p$. For simplicity, we assume all sequences have the same number n of features. Further our scheme is feature-agnostic, i.e., they 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). We provide below an MIL formulation for (i) and a joint objective combining (i) and learning (ii).

3.1. 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 $\mathcal{X}^-$ (as 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 bag X_i^+ from all features in $\mathcal{X}^-$, then we cast the following objective, which is a variant of the sparse MIL (SMIL) [3], normalized set kernel (NSK) [15], and $\propto$-SVM [59] formulations:

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

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

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

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

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

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 easily as described in the sequel and supported by our experimental results.

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

3.2. Learning Video Level Classifiers

Given a dataset of sequences $\mathcal{X}^+$ and a negative bag $\mathcal{X}^-$, we propose to learn the SVMP descriptors per sequence and the classifiers on $\mathcal{X}^+$ jointly as a multi-class structured SVM problem which includes the MIL problem $P1$ as a

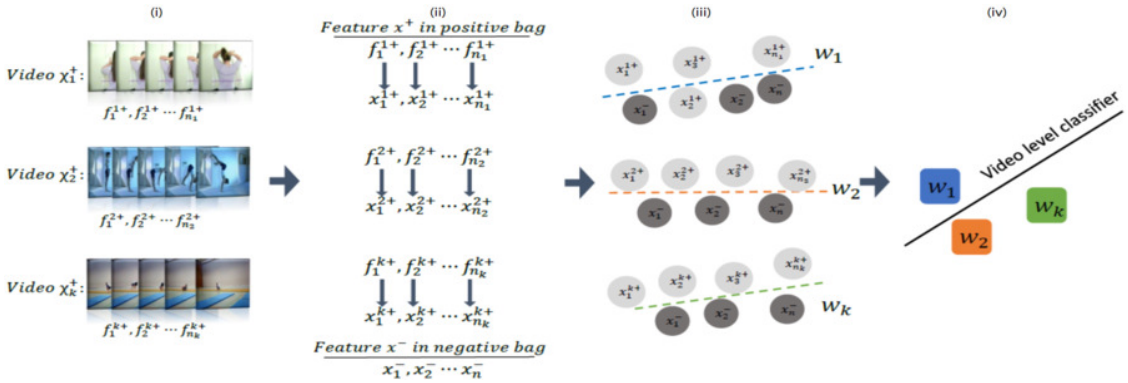

Figure 2. Illustration of our SVM pooling pipeline: (i) Extraction of frames f_i from videos, (ii) converting frames f_i into features x_i, (iii) learning decision boundaries w_j, one for every sequence, on its respective features x_i, and (iv) using w_j as descriptor in a video classifier.

sub-objective. The joint formulation is as follows:

$$\min_{w,b,Z} P2 := \sum_{i=1}^{N} \|w_i\|^2 + \sum_{j=1}^{d} \|Z_j\|^2 + C_2 \sum_{i,l=1}^{N} \gamma_{il} + C_1 \sum_{i=1}^{N} \sum_{k=1}^{(M+1)n} \zeta_{ik} \quad (6)$$

$$Z_j^T \left(\begin{bmatrix} w \\ b \end{bmatrix}_i - \begin{bmatrix} w \\ b \end{bmatrix}_l \right) \geq \Delta(y_i^+, y_l^+) - \gamma_{il}, \quad (7)$$

$$\text{where } y_i^+ = j, \forall y_l^+ \in \mathrm{I}_d, \text{ and } \forall j \in \mathrm{I}_d,$$

$$\theta(\mathbf{x};\eta)\left(w_i^T \mathbf{x} + b_i\right) \geq 1 - \zeta_{ik},\ \mathbf{x} \in X_i^+ \bigcup \mathcal{X}^-, \forall i = \mathrm{I}_N,$$

$$\text{and} \quad \theta(\mathbf{x};\eta) \in \{+1, -1\}, \gamma_{il} \geq 0,\ \zeta_{ik} \geq 0,$$

where $\theta(x.;\eta)$ is as defined in (3) and (4). The function $\Delta(y, z)$ computes the similarity between the ground truth labels y and z. The formulation $P2$ jointly optimizes the computations of SVMP descriptors per sequence (w_i, b_i) and the parameters Z of d video level classifiers, in a one-versus-rest fashion as described in (7). The constant C_2 is a regularization parameter on the action classifiers and γ represents the respective slack variables per sequence. For brevity, we use I_m to represent the set $\{1, 2, ..., m\}$.

3.3. Efficient Optimization

The problem $P2$ is not convex due to the function $\theta(\mathbf{x};\eta)$ that needs to select a set from the positive bags that satisfy the criteria in (5). Also, note that the sub-problem $P1$ could be posed as a mixed-integer quadratic program (MIQP), which is known to be in NP [24]. While, there are efficient approximate solutions for this problem (such as [30]), the solution must be scalable to large number of both high-dimensional features generated by a CNN and low-dimensional local features. To this end, we propose the following relaxation.

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. However, the correct value of C_1 depends on each sequence. Thus, in Algorithm (1), 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
  $C_1 \leftarrow \lambda C_1;$
  $[w, b] \leftarrow \arg\min_{w,b} \mathrm{SVM}(X^+,\ \mathcal{X}^-,\ C_1);$
  $\hat{X}^+ \leftarrow \left\{\mathbf{x} \in X^+ \mid w^T\mathbf{x} + b \geq 0\right\};$
until $\frac{|\hat{X}^+|}{|X^+|} \geq \eta;$
return $[w, b]$

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

Each step of Algorithm (1) 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. If $\hat{X}_p^+$ is one such $\hat{X}^+$. Then, $P1$ using $\hat{X}_p^+$ is just the SVM formulation and is thus convex. That is, if we enumerate all such $\hat{X}_p^+$ that satisfies the constraint using η, then the objective for each such $\hat{X}_p^+$ is an SVM problem, that could be solved using standard efficient solvers. Instead of enumerating all such bags $\hat{X}_p^+$, in Alg. 1, 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 $P1$, due to the convexity of SVM, it can be shown that the 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).

Considering $P2$, it is non-convex in Z and (w_i, b_i)'s jointly. However, it is convex in Z when fixing $(w_i, b_i), \forall i \in \{1, 2, ..., N\}$. Thus, under the above conditions, if we need to run only one iteration of $P1$, then $P2$ becomes convex in either variables separately, and thus we could solve it using block coordinate descent (BCD) towards a local minimum. Algorithm 2 depicts the iterations. Note that there is a coupling between the data point decision boundaries (w_i, b_i) and the action classifier decision boundaries Z_j in (7), either of which are fixed when optimizing over the other using BCD. When optimizing over (w_i, b_i), $Z_j^T \begin{bmatrix} w \\ b \end{bmatrix}_l$ (in (7)) is a constant, and we use $\Delta(y_i^+, y_i^+) = 1$, in which case the problem is equivalent to assuming Z as a *virtual* positive data point in the positive bag. We make use of this observation in Algorithm 2 by including Z in the positive bag. Note that these virtual Z points are updated in place rather than adding new points in every iteration.

Input: $\mathcal{X}^+, \mathcal{X}^-, \eta$
repeat
  `/* compute SVMP descriptors for all sequences */`
  for $X_i^+ \in \mathcal{X}^+$ **do**
    $[w_i, b_i] \leftarrow \arg\min_{w,b} \text{SVM}(X_i^+, \mathcal{X}^-, C)$;
  end
  $Z \leftarrow$ Solve P2 fixing $\begin{bmatrix} w \\ b \end{bmatrix}_i, \forall i = \{1, ..., N\}$;
  `/* Z is added to` X_i^+ `so that SVM could be used to satify (7) */`
  $X_i^+ \leftarrow X_i^+ \cup Z$
until *until convergence*;
return Z

Algorithm 2: A block-coordinate scheme for $P2$

When using decision boundaries as data descriptors, a natural question can be regarding the identifiability of the sequences using this descriptor, especially if the negative bag is randomly sampled. To circumvent this issue, we propose two workarounds, namely (i) to use the same negative bag for all the sequences, and (ii) assume all features (including positives and negatives) are centralized with respect to a global data mean.

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 [43], 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:

$$f(.) = \sum_{\mathbf{x} \in X^+ \cup \mathcal{X}^-} \alpha_{\mathbf{x}} K(., \mathbf{x}), \tag{8}$$

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 [49], 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.

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 backpropagation. Assume a CNN f taking a sequence S as input. Let f_L denote the L-th CNN layer and let Y_L denote the feature maps generated by this layer for all frames in S. We assume these features go into an SVMP 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 classification. Mathematically, let $g(w) = \arg\min_w \text{SVMP}(Y_L)$ define the SVM pooling layer, which we re-define the hinge-loss as:

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

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 $g(w)$ with respect to the data z. By applying Lemma 3.3 of [17], this gradient of the argmin at an optimum SVMP solution w^* can be shown to be the following:

$$\nabla_z g(w^*) = -\nabla_{ww} \text{SVMP}(Y_L)^{-1} \nabla_{zw} \text{SVMP}(Y_L),$$

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:

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

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 tasks, namely (i) action recognition using video and skeletal features, (ii) action anticipation, and (iii) localizing actions in videos. We introduce these datasets briefly next along with details of the features used, followed by an analysis of the parameters of our pooling scheme, before furnishing our results against state-of-the-art.

5.1. Datasets

HMDB-51 [22]: is a popular benchmark for video action recognition, consisting of trimmed videos downloaded from the Internet. The dataset contains 51 action classes and 6766 videos. The recognition results are evaluated using 3-fold cross-validation and mean classification accuracy is reported. For this dataset, we analyze different combinations of features on multiple CNN frameworks.

Charades [39]: is an untrimmed 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, including several fine-grained classes. In the classification task, we follow the evaluation protocol of [39], using the output probability of the classifier to be the score of the sequence. In the detection task, we use 'post-processing' protocol described in [38], which uses the averaged prediction score of a small temporal window around temporal pivots. The dataset provides two-stream VGG-16 fc7 features which we use in our method.[1] The performance on detection and recognition tasks are evaluated using mean average precision (mAP) on the validation set.

NTU-RGBD [37]: is by far the largest action datasets providing 3D skeleton data. It has 56,000 videos and 60 actions performed by 40 people from 80 different views. We use the temporal CNN proposed in [20] to generate features, but uses SVMP instead of their global average pooling.

5.2. Parameter Analysis

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

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

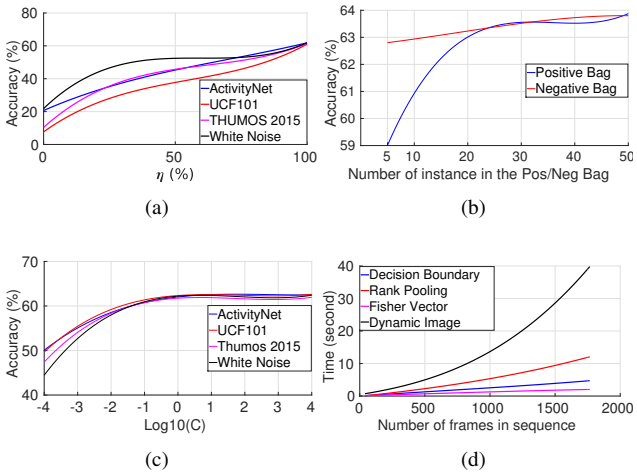

Figure 3. Analysis of the parameters in our scheme. All experiments use VGG features from fc6 layer. See text for details.

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 (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 and apply them on the HMDB-51 dataset split-1. They are samples from (i) ActivityNet dataset [4] unrelated to HMDB-51, (ii) UCF-101 dataset unrelated to HMDB-51, (iii) Thumos Challenge background sequence[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 images. As shown in Figure 3(a), the white noise negative shows better performance for both lower and higher value of η parameter. So, we use it in our experiments for other datasets.

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 1 when finding SVMP per sequence, and (iii) sizes of the positive and negative bags. To study (i) and (ii), we plot in Figures 3(c) and 3(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 3(b) shows that increasing C increases the accuracy of

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

Table 1. Comparisons using various features on HMDB-51 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)

the SVMP descriptor, implying that the CNN features are already equipped with discriminative properties for action recognition. However, 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 3(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 3(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 3(d), we compare the time it took on average to generate SVMP descriptors for an increasing number of frames in a sequence. For comparison, we plot the running times for some of the recent pooling schemes such as rank pooling [2, 14] and the Fisher vectors [51]. 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 in contrast to some of the recent popular pooling methods.

5.3. Experiments on HMDB-51

Following the recent trends, for this experiment, we use a two-stream CNN model in popular architectures, the VGG-16 and the ResNet-152 [13, 42]. We fine-tune a two-stream VGG/ResNet model trained for the UCF-101 dataset.

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 Algorithm 1 and 2 against the chosen set of negative bags. In Table 1, we report results on split-1 of the HMDB-51 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.

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

Table 2. Comparisons between SVMP and NSVMP on HMDB-51 split-1

	VGG	ResNet
Linear-SVMP	63.8%	69.5%
Non-linear-SVMP	64.4%	69.8%
Combination	**66.1%**	**71.0%**

Table 3. Comparison to standard pooling on HMDB-51 split-1

	VGG	ResNet
Spatial Stream-AP[10, 13]	47.1%	46.7%
Spatial Stream-MP	46.5%	45.1%
Spatial Stream-SVMP	**58.3%**	**57.4%**
Temporal Stream-AP [10, 13]	55.2%	60.0%
Temporal Stream-MP	54.8%	58.5%
Temporal Stream-SVMP	**61.8%**	**65.7%**
Two-Stream-AP [10, 13]	58.2%	63.8%
Two-Stream-MP	56.7%	60.6%
Two-Stream-SVMP	**66.1%**	**71.0%**

Table 4. Comparison of action anticipation on HMDB-51 split-1

	HMDB-51				
k/5	1/5	2/5	3/5	4/5	1
SVMP	**58.3%**	**65.5%**	**68.4%**	**70.1%**	**71.0%**
AP	48.6%	56.4%	59.9%	62.5%	63.8%
MP	46.2%	55.4%	56.3%	58.8%	60.6%

SVMP Extensions and Standard Pooling: We analyze the complementary nature of SVMP and its non-linear extension NSVMP (using a Chi-sq homogeneous kernel) on HMDB-51 split1. The results are provided in Table 2, and clearly show that the combination leads to significant improvements consistently on both datasets. Comparison between SVMP and standard pooling schemes (such as average (AP) and max (MP)) are reported in Table 3 using exactly the same set of features. As is clear, SVMP is significantly better than the other two pooling schemes.

SVMP for Action Anticipation We also evaluated the usefulness of SVMP for action anticipation. This is motivated by the intuition that SVMP might be able to learn generalizable decision boundaries when shown only a small part of the sequence – given the SVM is optimized in a max-margin framework. Specifically, we use $k \times \frac{1}{5}$ initial part of the sequences to be pooled by SVMP, ($k \in \{1, 2, 3, 4, 5\}$) which has to now predict the action in the full segment. We use the ResNet feature for this experiment. The results are provided in Table 4 and is clear that compared with others, the benefits of SVMP become higher, when only seeing a small fraction of the data, substantiating our intuition.

5.4. Recognition/Detection in Untrimmed Videos

As introduced in the Section 5.1, Charades is an untrimmed dataset with multiple actions in one sequence. We use the publicly available two-stream VGG features from the fc7 layer for this dataset. We applied our scheme on the provided training set (7985 videos), and report results (mAP) on the provided validation set (1863 videos)

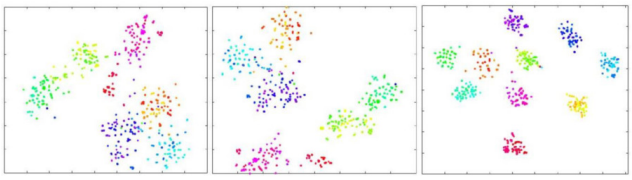

Figure 4. T-SNE visualizations of SVMP and other pooling methods. From left to right: average pooling, max pooling, and SVMP.

for the tasks of action classification and detection. In the classification task, we concatenate the two-stream features and apply a sliding pooling scheme to create multiple descriptors. Following the evaluation protocol in [39], we use the output probability of the classifier to be the score of the sequence. In the detection task, the standard evaluation setting is to use the prediction score of 25 equidistant time points in the sequence, which is not suitable for any pooling scheme. So, we consider another evaluation method with post-processing, proposed in [38]. This method uses the averaged prediction score of a small temporal window around each temporal pivots. Instead of average pooling, we apply the SVMP. From Table 5, it is clear that SVMP improves performance against other pooling schemes.

5.5. Skeletal Action Recognition in NTU-RGBD

For this experiment, we follow the two official evaluation protocols described in [37], i.e., the cross-view and cross-subject protocol. We use [20] as the baseline. This scheme applies a temporal CNN with residual connections on the 3D skeleton data. We swap the global average pooling layer in [20] by a Rank/SVM pooling layer. The result in Table 5 indicates that the SVMP works better than other pooling schemes on the skeleton-based features.

5.6. Visualization of SVMP

To gain further intuitions into the performance boost by SVMP, in Figure 4, we show TSNE visualizations comparing to average and max pooling on 10-classes from HDMB-51. The visualization shows that SVMP leads to better separated clusters, substantiating that it is learning much more discriminative representations than traditional methods.

5.7. Comparisons to the State of the Art

In Table 5, we compare our best result against the state-of-the-art results on each dataset using the respective standard evaluation protocols. For a fair comparison, we also report our best result combining with hand-crafted features (IDT-FV) [50] for HMDB-51. Our scheme obtains the state-of-the-art performance in all datasets and outperform other methods by 1–4%. We note that recently the two-stream I3D+ model[5], which is pre-trained on the larger Kinectics dataset (with more than 300K videos), achieves 80% on HMDB-51. However, without additional data, two-stream I3D is outperformed by our SVMP. Moreover, most of these methods could enjoy a further boost by applying our SVMP scheme. To substantiate this, we also show the I3D+ model to use SVMP (instead of their proposed average pooling) on HMDB-51 dataset using the settings in [5].

Table 5. Comparison to the state of the art in each dataset, following the official evaluation protocol for each dataset.

HMDB-51 (accuracy over 3 splits)	
Method	Accuracy
Temporal segment networks[54]	69.4%
AdaScan[19]	54.9%
AdaScan + IDT + C3D[19]	66.9%
ST ResNet[10]	66.4%
ST ResNet + IDT[10]	70.3%
ST Multiplier Network[11]	68.9%
ST Multiplier Network + IDT[11]	72.2%
Two-stream I3D[5]	66.4%
Two-stream I3D+ (Kinetics 300k)[5]	80.9%
SVMP (ResNet)	71.0%
SVMP (ResNet+IDT)	**72.6%**
SVMP (I3D+)	**81.3%**

Charades (mAP)		
Method	Classification	Detection
Two-stream VGG (Average Pooling) [40]	14.3%	10.9%
Two-stream VGG (Max Pooling) [40]	15.3%	9.2%
ActionVLAD + IDT[16]	21.0%	-
Asynchronous Temporal Fields [38]	22.4%	12.8%
SVMP(VGG)	25.1%	13.9%
SVMP(VGG+IDT)	**26.7%**	**14.2%**

NTU-RGBD		
Method	Cross-Subject	Cross-View
Res-TCN (Average Pooling)[20]	74.3%	83.1%
Res-TCN (Rank Pooling [2])	75.5%	83.9%
STA-LSTM [44]	73.4%	81.2%
ST-LSTM + Trust Gate[28]	69.2%	77.7%
Body-parts learning [33]	75.2%	83.1%
SVMP (Res-TCN)	**78.5%**	**86.4%**

6. Conclusion

In this paper, we presented a simple, efficient, and powerful pooling scheme, SVM pooling, for summarizing videos. We cast the pooling problem in a multiple instance learning framework, and seek to learn useful decision boundaries on the frame level features from each sequence against background/noise features. We provide an efficient scheme that jointly learns these decision boundaries and the action classifiers on them. We also extended the framework to deal with nonlinear decision boundaries and end-to-end CNN training. Extensive experiments were showcased on three 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 videos.

References

[1] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *Human Behavior Understanding*, pages 29–39. 2011. 2

[2] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016. 1, 2, 7, 8

[3] R. C. Bunescu and R. J. Mooney. Multiple instance learning for sparse positive bags. In *ICML*, 2007. 3

[4] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 6

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, July 2017. 1, 2, 8

[6] A. Cherian, B. Fernando, M. Harandi, and S. Gould. Generalized rank pooling for activity recognition. In *CVPR*, 2017. 2

[7] R. G. Cinbis, J. Verbeek, and C. Schmid. Weakly supervised object localization with multi-fold multiple instance learning. *PAMI*, 39(1):189–203, 2017. 2

[8] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 2

[9] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015. 2

[10] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016. 1, 7, 8

[11] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *CVPR*, 2017. 1, 2, 8

[12] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Temporal residual networks for dynamic scene recognition. In *CVPR*, 2017. 1, 2

[13] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 1, 7

[14] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015. 2, 7

[15] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola. Multi-instance kernels. In *ICML*, 2002. 3

[16] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017. 8

[17] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016. 5

[18] M. Hayat, M. Bennamoun, and S. An. Deep reconstruction models for image set classification. *PAMI*, 37(4):713–727, 2015. 1, 2

[19] 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. 8

[20] T. S. Kim and A. Reiter. Interpretable 3d human action analysis with temporal convolutional networks. *arXiv preprint arXiv:1704.04516*, 2017. 2, 6, 8

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[22] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011. 6

[23] K.-T. Lai, F. X. Yu, M.-S. Chen, and S.-F. Chang. Video event detection by inferring temporal instance labels. In *CVPR*, 2014. 3

[24] R. Lazimy. Mixed-integer quadratic programming. *Mathematical Programming*, 22(1):332–349, 1982. 4

[25] 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. 2

[26] W. Li and N. Vasconcelos. Multiple instance learning for soft bags via top instances. In *CVPR*, 2015. 2

[27] W. Li, Q. Yu, A. Divakaran, and N. Vasconcelos. Dynamic pooling for complex event recognition. In *ICCV*, 2013. 3

[28] J. Liu, A. Shahroudy, D. Xu, A. C. Kot, and G. Wang. Skeleton-based action recognition using spatio-temporal lstm network with trust gates. *arXiv preprint arXiv:1706.08276*, 2017. 8

[29] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011. 2

[30] R. Misener and C. A. Floudas. Glomiqo: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, 57(1):3–50, 2013. 4

[31] S. Nowozin, G. Bakir, and K. Tsuda. Discriminative subsequence mining for action classification. In *ICCV*, 2007. 3

[32] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013. 2

[33] H. Rahmani and M. Bennamoun. Learning action recognition model from depth and skeleton videos. In *ICCV*, 2017. 8

[34] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. 2

[35] S. Satkin and M. Hebert. Modeling the temporal extent of actions. In *ECCV*, 2010. 3

[36] K. Schindler and L. Van Gool. Action snippets: How many frames does human action recognition require? In *CVPR*, 2008. 2

[37] 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. 6, 8

[38] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, 2017. 6, 8

[39] 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. 6, 8

[40] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 8

[41] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2

[42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 7

[43] A. J. Smola and B. Schölkopf. *Learning with kernels*. Citeseer, 1998. 5

[44] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *AAAI*, 2017. 8

[45] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852, 2015. 2

[46] C. Sun and R. Nevatia. Discover: Discovering important segments for classification of video events and recounting. In *CVPR*, 2014. 3

[47] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. 2

[48] 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. 3

[49] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *PAMI*, 34(3):480–492, 2012. 5

[50] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013. 8

[51] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 2, 7

[52] J. Wang, A. Cherian, and F. Porikli. Ordered pooling of optical flow sequences for action recognition. *CoRR*, abs/1701.03246, 2017. 2

[53] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 1, 2

[54] 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. 1, 2, 8

[55] Y. Wang, J. Song, L. Wang, L. Van Gool, and O. Hilliges. Two-stream sr-cnns for action recognition in videos. In *BMVC*, 2016. 1

[56] G. Willems, J. H. Becker, T. Tuytelaars, and L. J. Van Gool. Exemplar-based action recognition in video. In *BMVC*, 2009. 2

[57] J. Wu, Y. Yu, C. Huang, and K. Yu. Deep multiple instance learning for image classification and auto-annotation. In *CVPR*, 2015. 2

[58] Y. Yi and M. Lin. Human action recognition with graph-based multiple-instance learning. *Pattern Recognition*, 53:148–162, 2016. 2

[59] 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. 3

[60] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2

[61] J. Zepeda and P. Perez. Exemplar svms as visual feature encoders. In *CVPR*, 2015. 2

[62] 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. 2