Improving Human Action Recognition by Non-action Classification

Yang Wang and Minh Hoai
Stony Brook University, Stony Brook, NY 11794, USA
{wang33, minhhoai}@cs.stonybrook.edu

Abstract

In this paper we consider the task of recognizing human actions in realistic video where human actions are dominated by irrelevant factors. We first study the benefits of removing non-action video segments, which are the ones that do not portray any human action. We then learn a non-action classifier and use it to down-weight irrelevant video segments. The non-action classifier is trained using Action-Thread, a dataset with shot-level annotation for the occurrence or absence of a human action. The non-action classifier can be used to identify non-action shots with high precision and subsequently used to improve the performance of action recognition systems.

1. Introduction

The ability to recognize human actions in video has many potential applications in a wide range of fields, ranging from entertainment and robotics to security and health-care. However, human action recognition [1, 8, 13–15, 29, 37] is tremendously challenging for computers due to the complexity of video data and the subtlety of human actions. Most current recognition systems flounder on the inability to separate human actions from the irrelevant factors that usually dominate subtle human actions. This is particularly problematic for human action recognition in TV material, where a single human action may be dispersively portrayed in a video clip that also contains video shots for setting the scene and advancing dialog. For example, consider the video clips from the Hollywood2 dataset [22] depicted in Figure 1. These video clips are considered as ‘clean’ examples for their portrayed actions, but 3 out of 6 shots do not depict the actions of interest at all. Although recognizing human actions in TV material is an important and active area of research, existing approaches often assume the contiguity of human action in video clip and ignore the existence of irrelevant video shots.

In this paper, we first present our findings on the benefits of having purified action clips where irrelevant video shots are removed. We will then propose a dataset and a method to learn a non-action classifier, one that can be used to remove or down-weight the contribution of video segments that are unlikely to depict a human action.

Of course, identifying all non-action video segments is an ill-posed problem. First, there is no definition for what a general human action is. Second, even for the classes of actions that are commonly considered such as hug and handshake, the temporal extent of an action is highly ambiguous. For example, when is the precise moment of a hug? When two people start opening their arms or when the two bodies are in contact? Because of the ambiguities in human actions, our aim in this paper is to identify video segments that are unlikely related to the actions of our interest. Many of those segments can be unarguably identified, e.g., video shots that contain no people, show the close-up face of a character, exhibit little motion, or are part of a dialog (some examples are shown in Figure 1). However, instead of manually defining what a non-action segment should be, in this paper we will use supervised learning and train a non-action classifier using video data that has shot-level annotation. The classifier is based on Support Vector Machines [34] and appearance and motion features. More specifically, we will combine Fisher Vector encoding [27] of Dense Trajectory Descriptors [37] and the deep learning features of a Two-stream ConvNet [29].

It should be noted that we propose to learn a non-action classifier for generic human actions. This has several benefits over an action-specific classifier that only aims to identify video segments that are irrelevant to a specific action.
First, a generic non-action classifier is universal; it can be used to improve the recognition performance of action classes that do not have detailed training annotation. Second, even when detailed annotation exists, it would still be difficult to obtain a good action-specific segment classifier. To some extent, having a good classifier that can remove segments that are not related to a specific class is equivalent to having a good action recognizer already. Thus, an action-specific classifier brings no complementary benefits, while a generic action classifier does and can be used to increase the signal-to-noise ratio of actions in video clips.

The rest of this paper is structured as follows. We first review some related topics in Section 2. Section 3 presents the empirical evidence that pruning irrelevant shots leads to significant improvement on human action recognition. This is followed by the experiment on learning and evaluating a non-action classifier in Section 4. In Section 5, we propose an approach for using the non-action classifier for human action recognition and describe the performance gains in several experiments.

2. Related Works

In this work, we propose to learn a non-action classifier to predict whether a video subsequence is an action instance [19]. This is related to defining and measuring objectness in image windows [2, 3, 33], which can assist some common visual tasks like object proposal and detection. Learning such high-level concept often relies on well-defined visual features such as saliency[17], color, edges [40] and super-pixels. There have been some recent attempts [5, 7] to extend objectness to actionness. They often measure the actionness by fusing different feature channels such as space-time saliency [24], optical flow [7], body configuration [16] and deep learning features [9], sometimes with human input like eye fixation [23]. However, compared to objectness, actionness in videos is still not sufficiently explored due to the computational intensity in video space and the subtlety of human actions.

3. Benefits of pruning irrelevant shots

We now present our findings on the statistics of non-action shots in a typical human action dataset and the benefits of removing them for human action recognition. We will defer the description of a method for classifying non-action shots to the next section. In this section, we assume there is an oracle for identifying non-action shots.

3.1. ActionThread dataset

For the studies in this section, we consider the ActionThread dataset [12]. This is a typical human action dataset of which the method to collect human action samples is similar to that of many human action datasets, including Hollywood2 [22], TVHID [25], and Hollywood3D [10]. The ActionThread dataset consists of video samples for 13 actions, a superset of the actions considered in Hollywood2 [22] and TVHID [25]. The video samples were automatically located and extracted around human action occurrences using script mining in 15 different TV series. They are split into training and test sets such that the two subsets do not share samples from the same TV series.

Amazon Mechanical Turk (AMT) workers were asked to annotate the occurrence of human actions shot-by-shot for each video. Each video shot of the dataset was labeled by three AMT workers. Most (86.3%) of the shots received the same annotation by three AMT workers. We then manually reviewed and carefully relabeled those shots with conflicting annotations. Of those shots we relabeled, around 60% were consistent with the majority vote. Videos without action occurrences were eliminated. Finally, we have a dataset of 3,035 videos for 13 actions. Table 1 shows detailed statistics for the refined ActionThread dataset. On average, one video contains roughly 6.5 shots, 60% of which are non-action shots.

3.2. Action recognition system

We hypothesize that removing non-action shots will improve the recognition performance. We study this hypothesis with a popular action recognition method that holds the state-of-the-art performance on many datasets. This method is based on Dense Trajectory Descriptors [37], Fisher Vector encoding [27], and Least-Squares Support Vector Machines [31].
Table 2. Benefits of pruning irrelevant shots. This shows the Average Precision (AP) of a popular action recognition method, with and without pruning the non-action shots. The last column displays the performance gain for pruning non-action shots. The performance gain is significant, as large as 34% for DriveCar.

<table>
<thead>
<tr>
<th>Action</th>
<th>No pruning</th>
<th>Pruning</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ans.Phone</td>
<td>29.2</td>
<td>42.0</td>
<td>12.8</td>
</tr>
<tr>
<td>DriveCar</td>
<td>41.4</td>
<td>75.5</td>
<td>34.1</td>
</tr>
<tr>
<td>Eat</td>
<td>33.2</td>
<td>56.1</td>
<td>22.9</td>
</tr>
<tr>
<td>Fight</td>
<td>63.0</td>
<td>70.6</td>
<td>7.6</td>
</tr>
<tr>
<td>GetOutCar</td>
<td>23.8</td>
<td>35.8</td>
<td>12.0</td>
</tr>
<tr>
<td>ShakeHand</td>
<td>42.9</td>
<td>58.4</td>
<td>15.5</td>
</tr>
<tr>
<td>Hug</td>
<td>50.1</td>
<td>59.4</td>
<td>9.3</td>
</tr>
<tr>
<td>Kiss</td>
<td>65.0</td>
<td>72.2</td>
<td>7.2</td>
</tr>
<tr>
<td>Run</td>
<td>85.2</td>
<td>93.7</td>
<td>8.5</td>
</tr>
<tr>
<td>SitDown</td>
<td>60.9</td>
<td>76.5</td>
<td>15.6</td>
</tr>
<tr>
<td>SitUp</td>
<td>9.5</td>
<td>13.2</td>
<td>3.7</td>
</tr>
<tr>
<td>StandUp</td>
<td>40.4</td>
<td>58.5</td>
<td>18.1</td>
</tr>
<tr>
<td>HighFive</td>
<td>44.7</td>
<td>55.1</td>
<td>10.4</td>
</tr>
<tr>
<td>Mean</td>
<td>45.3</td>
<td>59.0</td>
<td>13.7</td>
</tr>
</tbody>
</table>

3.2.1 Trajectory features

The feature representation is based on improved Dense-Trajectory Descriptors (DTDs) [37]. DTD extracts dense trajectories and encodes gradient and motion cues along trajectories. Each trajectory leads to four feature vectors: Trajectory, HOG, HOF, and MBH, which have dimensions of 30, 96, 108, and 192 respectively. The procedure for extracting DTDs is the same as [12, 13], and we refer the reader to [12, 13, 37] for more details.

Note that each trajectory has a temporal span of 15 frames, and the temporal location of each trajectory is taken as the index of the middle frame (the 8th frame). For efficiency, we only extract trajectory descriptors for each video clip once. If an experiment requires pruning some segments of the video clip, we simply remove the trajectories that are associated with the frames inside the segments.

3.2.2 Fisher Vector encoding

To encode features, we use Fisher Vector [27]. A Fisher Vector encodes both first and second order statistics between the feature descriptors and a Gaussian Mixture Model (GMM). In [37], Fisher Vector shows an improved performance over bag of features for action classification. Following [27, 37], we first reduce the dimension of DTDs by a factor of two using Principal Component Analysis (PCA). We set the number of Gaussians to \( k = 256 \) and randomly sample a subset of 1,000,000 features from the training sets to learn the GMM. There is one GMM for each feature type. A video sequence is represented by a \( 2dk \) dimensional Fisher Vector for each descriptor type, where \( d \) is the descriptor dimension after performing PCA. As in [27, 37], we apply power \((\alpha = 0.5)\) and \(L_2\) normalization to the Fisher Vectors. We combine all descriptor types by concatenating their normalized Fisher Vectors, leading to a single feature vector of 109,056 dimensions.

3.2.3 Least-Squares SVM

For recognition, we use Least-Squares Support Vector Machines (LSSVM) [31]. LSSVM, also known as kernel Ridge regression [28], has been shown to perform equally well as SVM in many classification benchmarks [11, 32, 35, 36]. LSSVM has a closed-form solution, which is a computational advantage over SVM [34]. We train 13 binary LSSVM classifiers for 13 action classes. For each action class, we train a one-versus-rest classifier where positive training examples are action samples from the class in consideration and negative training examples are action samples from other classes.

3.3. Results

Table 2 shows the Average Precision (AP) of the aforementioned action recognition method with and without pruning non-action shots. Here we measure performance using Average Precision, which is an accepted standard for action recognition (e.g., [16, 22, 26]). As can be seen, the ability to prune non-action shots leads to significant performance gain, which is shown in the last column of Table 2. The average performance gain is 13.7%, and it is as high as 34.1% for DriveCar.

Table 2 shows the benefits of an ideal situation where we can identify all non-action shots. This is perhaps unrealistic in practice, but we can still expect some improvement in the recognition performance even when we cannot remove all
irrelevant shots. Figure 2 shows the mean average precision for recognizing 13 action classes when the percentage for removing non-action shots is varied from 0 to 100%. Given a percentage \( p \) (0% \( \leq p \leq 100\% \)), we randomly remove a non-action shot from the video with probability \( p \). We repeat this experiment 20 times and compute the mean and standard deviation for the mean average precision. As can be observed, the performance gain increases as the removal percentage grows. The performance gain is significant even when we can only eliminate 40% of the non-action shots.

4. Non-action Classification

Having confirmed that removing non-action shots leads to large performance gain in the action recognition task, we describe in this section our approach for learning a classifier to differentiate between action shots from non-action shots.

4.1. Feature computation

As also mentioned in the introduction, many non-action shots can be identified based on the size of the human characters, the amount of the motion, and the context of the shot in a longer video sequence (e.g., part of a dialog). To capture the discriminative information for classification, we propose to combine DTDs [37] and deep-learned features from a Two-stream ConvNet [29]. These features lead to the state-of-the-art performance in many datasets [18, 22, 30]. Recent experiments [38] have also suggested that they are complimentary to each other.

4.1.1 Dense Trajectory Features

Dense Trajectory Features are extracted and used with Fisher Vector encoding as described in Sections 3.2.1 and 3.2.2. Note that each trajectory has a temporal span of 15 frames, and we assign each trajectory to the middle frame (the \( i^{th} \) frame). Each frame is therefore associated with a set of trajectories. We compute an unnormalized Fisher Vector for each frame, and we have a sequence of frame-wise Fisher Vectors. Since unnormalized Fisher Vector is additive, the unnormalized Fisher Vector for a set of frames is the sum of unnormalized Fisher Vectors for individual frames. Thus, given a sequence of frame-wise unnormalized Fisher Vectors for a video clip, we can efficiently compute the unnormalized Fisher Vector for any subsequence of the video clip. Finally the unnormalized Fisher Vector can be normalized to obtain the DTD feature representation for the subsequence.

4.1.2 Deep-learning features

We use deep-learning features from a Two-stream ConvNet [29], which was proposed recently for human action recognition. In this paper, we use the pre-trained Two-Stream ConvNet provided by Wang et al. [38] as a generic feature extractor. The model is trained on Split1 of UCF-101 dataset [30]. This model contains both a spatial and a temporal ConvNet [29]. The spatial ConvNet is based on VGG-M-2048 model [4] and fine-tuned with image frames from videos; the temporal ConvNet have a similar structure, but its input is a set of 10 consecutive optical flow maps.

Video frames are extracted at 25fps and subsequently resized to 256x340 pixels, having the aspect ratio of 4:3. Given the sequence of frames, we calculate the dense optical flow map between pairs of consecutive frames. We use the GPU version of TVL1 algorithm [39], for its efficiency and accuracy. Following [29], we rescale and discretize the optical flow values into the integer range [0, 255], and store as image using JPEG compression. This greatly reduces the storage size.

**Spatial features.** The spatial ConvNet is based on the VGG-M-2048 model [4]. It requires input as an image region of size \( 224 \times 224 \times 3 \) and outputs a 4096-dim feature vector at the FC6 layer. Note in VGG-M-2048, FC6 is a fully connected layer with 4096 dimensions, as opposed to FC7 which has 2048 dimensions. To compute the feature vector for a video frame, we extract FC6 feature vectors at the center area and four corners of the frame as well as their left-right flipped images. Thus we obtain 10 feature vectors and average them to get a single 4096-dim spatial feature for a frame. Because consecutive frames are often similar, we only compute the spatial feature vector at every five frames. The feature vector for a set of frames (either from a contiguous video sequence or from the union of multiple disjoint video sequences) is the average of the feature vectors computed for the individual frames in the set.

**Temporal features.** The computation of temporal feature vectors is similar to that of spatial feature vectors. The only difference is that the input to the temporal ConvNet must be a set of 10 consecutive optical flow maps. So, to compute the temporal feature vector for a frame \( t \), we use the 10-frame volume that is centered at the frame \( t \).
4.1.3 Feature representation for a video shot

Consider a video shot \([s, e]\) which is a part of a longer video clip \([1, n]\). We first compute \(f_{\text{in}}\) the feature vector for the video shot by concatenating the DTD feature vector, the spatial ConvNet feature vector, and the temporal ConvNet feature vector. Note that the DTD feature vector is power and \(L_2\)-normalized, while the spatial and temporal ConvNet feature vectors are \(L_2\)-normalized. In addition to \(f_{\text{in}}\), we also compute two feature vectors \(f_{\text{out}}\) and \(f_{\text{all}}\) for the video frames outside \([s, e]\) and all the video frames, respectively. The ultimate feature vector to represent a video shot is taken as \([f_{\text{in}}, f_{\text{out}}, f_{\text{all}}]\), as illustrated in Figure 3. This feature vector encodes the appearance and motion of the video shot as well as its relative comparison with other video shots in its surrounding context.

4.2. Training a non-action classifier

We obtain a non-action classifier by training a Least-Squares SVM (Section 3.2.3) using data from the ActionThread dataset. This dataset is divided into disjoint train and test subsets, which contain 9724 and 9893 shots respectively. Within each subset, around 60% of the shots are non-action. The feature representation for each shot combines both DTD and Two-stream ConvNet, as described in the previous section.

4.3. Experiments and results

We measure the performance of non-action classification on the test set of the ActionThread dataset. The test set contains 1,514 videos, with 5877 non-action shots and 4016 action shots. Table 3 shows the Average Precision of the non-action classifier based on different features. DTD outperforms the Spatial and Temporal features of the Two-Stream ConvNet when they are used individually. When combined, the Spatial and Temporal ConvNets achieve comparable result to DTD. The best performance is achieved when all feature types are combined. From now on, we will use the combined feature vector in all of our experiments.

The non-action classifier using the combined feature vectors achieve the average precision of 86.1%. This classifier can be used to remove non-action shots and increase the signal-to-noise ratio of the action content in a video clip. In some cases, to minimize the chance of removing action shots, one might want to limit the number of shots removed for each video clip. Table 4 reports the Average Precision of the non-action classifier when the number of removed shots per video is limited to \(k\), with \(k = 1, 2, 3, 4\). As can be seen, limiting the number of removed shots per video can improve the average precision.

Figure 4 shows the distribution of non-action confidence scores on video shots of the test set. Each column represents an individual video; the red dots and blue pluses on that column respectively correspond to the non-action and action shots in the video. The points above have higher confidence scores than those below. For visualization, we align each video with the horizontal bar in the middle using the maximum score of the action shots. As can be seen, action and non-action shots are separated fairly well.

We also examine the average precision of the non-action classifier for individual videos. Figure 5 show the distribution of the average precisions computed for individual videos. As can be seen, the big proportion of the videos have the average precision of 1, which means a perfect separation between action and non-action shots.

### Table 3. Evaluation of non-action shot classification.

<table>
<thead>
<tr>
<th>Feature</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial (ConvNet)</td>
<td>80.8%</td>
</tr>
<tr>
<td>Temporal (ConvNet)</td>
<td>81.4%</td>
</tr>
<tr>
<td>Spatial+Temporal</td>
<td>84.1%</td>
</tr>
<tr>
<td>DTD</td>
<td>84.7%</td>
</tr>
<tr>
<td>DTD+Spatial+Temporal</td>
<td>86.1%</td>
</tr>
</tbody>
</table>

### Table 4. Average Precision of the non-action classifier when we remove at most \(k\) shots per video. \(k = \infty\) corresponds to no constraint on the number of removed shots per video.

<table>
<thead>
<tr>
<th>(k)</th>
<th>AP@k=1</th>
<th>AP@k=2</th>
<th>AP@k=3</th>
<th>AP@k=4</th>
<th>AP@(k=\infty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\infty)</td>
<td>92.6%</td>
<td>91.8%</td>
<td>90.9%</td>
<td>89.6%</td>
<td>86.1%</td>
</tr>
</tbody>
</table>

4.3.1 Leave-One-Action-Out Generalizability

So far, we train a non-action classifier based on non-action shots from videos of several human actions. The classifier
achieves the AP of 86.1%, which is encouraging. However, it is still unclear whether this classifier can be used to identify non-action shots in videos of an action that is not among the set of actions in the training data. To test this, we consider the performance of the leave-one-class-out classifiers. In particular, we consider 13 action classes of the ActionThread dataset in turn. For each action class, we train a non-action classifier on a reduced training dataset where the videos of the action class in consideration are removed. The obtained non-action classifier is used to identify the non-action shots in the videos of the left-out action class. We compare the average precision of this classifier and the classifier trained with all data. The results are shown in Figure 6. As can be seen, the leave-one-class-out classifiers are comparable to the non-action classifier trained on the full dataset. This demonstrates the ability to apply the non-action classifier to purify videos of unseen actions.

5. Non-action Classifier for Action Recognition

This section describes the benefits for using the non-action classifier for human action recognition. We also demonstrate the advantages of the generic non-action classifier over a set of action-specific non-action classifiers.

5.1. Action recognition with non-action classifier

Our proposed algorithm is based on the action recognition system described in Section 3.2, using dense trajectory descriptors, Fisher Vector encoding, and LSSVM. The key difference is the incorporation of the non-action classifier to down-weight irrelevant non-action video segments.

Suppose a video is represented by a set of segments \( \{W_i\} \) (discussed below). We first compute the normalized Fisher Vectors \( \{ \phi_i \} \) and non-action confidence scores \( \{ s_i \} \) of the segments. The feature vector for the video is taken as:

\[
\phi = \sum_i w_i \phi_i, \text{ with } w_i = \frac{e^{-\alpha s_i}}{\sum_j e^{-\alpha s_j}}
\]  

Here, we use \( s_i \) to weight the contribution of \( \phi_i \) using the softmax function. The higher \( s_i \), the lower the weight is. The parameter \( \alpha \) controls the balance between average pooling and max pooling. When \( \alpha \) is 0, all \( w_i \)'s are the same, and this becomes average pooling. If \( \alpha = \infty \), only one segment has the weight of 1, while the weights of others are 0. This is equivalent to max pooling. By tuning \( \alpha \), we can have a good balance between average pooling and max pooling. Here we propose to weight the contribution of video segments instead of removing non-action segments because the non-action classifier is imperfect (even though the AP is as high as 92.6% if we only remove the most confidence segment in each video).

Our approach for weighting the segment contribution is not limited to any segment generation scheme. Any segment proposal method, including random selection, shot-based division, or an action proposal method (if exists), can be incorporated in our framework. In this paper, we propose to use video segments that are generated by using a 25-frame sliding window. Because videos are at 25fps, each segment corresponds to one second of a video. One second is neither too short nor too long, and we can use the non-action classifier to determine its non-action score. Empirically, the action recognition performance is not sensitive to the segment length around one second. Notably, the segments are not taken as video shots for two reasons. First, the shot classifier itself is imperfect. Second, even for a shot that is not considered as non-action, it might be long and contains many parts that do not portray human action.

To verify the advantage of our action-weighted feature representation, we train 13 action classifiers using Least-Squares SVM with linear kernel as described in Section 3.2.3. Subsequently, we use another softmax function.
Table 5. Action recognition performance with generic or specific non-action classifiers.

<table>
<thead>
<tr>
<th>Feature</th>
<th>DTD</th>
<th>DTD + CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ans. Phone</td>
<td>None</td>
<td>29.2</td>
</tr>
<tr>
<td>DriveCar</td>
<td>None</td>
<td>41.4</td>
</tr>
<tr>
<td>Eat</td>
<td>None</td>
<td>33.2</td>
</tr>
<tr>
<td>Fight</td>
<td>None</td>
<td>63.0</td>
</tr>
<tr>
<td>GetOutCar</td>
<td>None</td>
<td>23.8</td>
</tr>
<tr>
<td>ShakeHand</td>
<td>None</td>
<td>42.9</td>
</tr>
<tr>
<td>Hug</td>
<td>None</td>
<td>50.1</td>
</tr>
<tr>
<td>Kiss</td>
<td>None</td>
<td>65.0</td>
</tr>
<tr>
<td>Run</td>
<td>None</td>
<td>85.2</td>
</tr>
<tr>
<td>SitDown</td>
<td>None</td>
<td>60.9</td>
</tr>
<tr>
<td>SitUp</td>
<td>None</td>
<td>9.5</td>
</tr>
<tr>
<td>StandUp</td>
<td>None</td>
<td>40.4</td>
</tr>
<tr>
<td>HighFive</td>
<td>None</td>
<td>44.7</td>
</tr>
</tbody>
</table>

Mean | 45.3 | 49.3 | 52.1 | 48.0 | 52.3 | 55.0 |

Table 6 compares the performance of VideoDarwin feature encodings with and without the non-action classifier. As can be seen, the non-action classifier provides benefits to this type of feature encoding.

5.3. VideoDarwin with non-action classifier

To understand whether the benefits of using the non-action classifier is limited to the Fisher Vector encoding, we experiment with a method where we integrate the non-action classifier with VideoDarwin [8], a feature encoding method to capture the temporal evolution in a video sequence. VideoDarwin was proposed recently, and achieved the state-of-the-art performance on a number of datasets. It assumes every frame of a video clip carries some information about the action, and the total information about the action in a video segment correlates with the length. To capture this, VideoDarwin learns a Support Vector Regression (SVR) that maps the feature vector computed for each segment to its own length. More specifically, suppose \( \phi_i \) is the feature vector for the \( i^{th} \) frame. VideoDarwin learn the parameters \( \mathbf{u} \) of an SVR such that:

\[
SVR(\mathbf{u}) : \sum_{i=1}^{k} \phi_i \rightarrow \sum_{i=1}^{k} 1, \ k = 1..N
\]

The learned parameter vector \( \mathbf{u} \) is then used as the feature representation for the video clip. Notably, this formulation is based on the Matlab implementation\(^1\) of the authors, and it is slightly different from the formulation given in the paper [8] in which Ranking SVM is used instead.

However, the assumption of VideoDarwin that every frame carries some information about the action does not hold due to the existence of non-action shots. To address this problem, we propose VideoDarwin++, a reformulated version that incorporating the outputs of a non-action classifier. VideoDarwin++ learns the parameters \( \mathbf{u} \) of an SVR such that:

\[
SVR(\mathbf{u}) : \sum_{i=1}^{k} w_i \phi_i \rightarrow \sum_{i=1}^{k} w_i, \ k = 1..N
\]

Here the amount of information about an action in a segment does not solely depend on the length, but by the weights that are calculated based on the non-action scores.

Table 6 compares the performance of VideoDarwin features, with and without the using non-action classifier. As can be seen, the non-action classifier provides benefits to this type of feature encoding.

---

\(^1\)https://bitbucket.org/bfernando/videodarwin
5.4. Cross-dataset generalization

We have demonstrated the generalizability of the non-action classifier to videos in the test set and videos of left-out action categories in Section 4. Now we further study the benefits of the non-action classifier to action recognition task in completely different datasets.

5.4.1 Performance on Hollywood2

We first consider Hollywood2 dataset [22], which includes 12 actions and 1,707 videos collected from 69 Hollywood movies. We first divide videos into shots using a shot boundary detection algorithm based on HOG [6] and SIFT [21] features, then manually label each shot for action occurrence or absence. Only 31% are non-action shots, and this is ‘cleaner’ than ActionThread. We apply the non-action classifier learned from ActionThread onto the Hollywood2 dataset and report in Table 7 the action recognition results with and without using the non-action classifier.

5.4.2 Recognizing unseen action categories.

We also collected human action samples of six actions that are not contained in the ActionThread dataset. As can be seen, the benefits of the non-action classifier can be generalized to action categories that do not exist in the training set of the non-action classifier.

6. Conclusions

We have studied the benefits of removing non-action shots and proposed a method for detecting them. Our detector is based on Dense Trajectories Descriptors and Two-stream ConvNet features. This detector achieves an average precision of 86%, and it can be used to down-weight the contribution of irrelevant segments in the computation of a feature vector to represent a video clip. This approach significantly improves the performance of a recognition system. In our experiments, the improvement is equivalent to the ability to remove 65% of non-action shots without any false positive. Although the non-action classifier is far from perfect, it makes a good step towards the ultimate solution for human action recognition in realistic video.

Acknowledgment. This project is supported by the National Science Foundation Award IIS-1566248.
References


