AdaScan: Adaptive Scan Pooling in Deep Convolutional Neural Networks for Human Action Recognition in Videos

Amlan Kar[1,*] Nishant Rai[1,*] Karan Sikka[2,3,†] Gaurav Sharma[1]
[1]IIT Kanpur[‡] [2]SRI International [3]UCSD

Abstract

We propose a novel method for temporally pooling frames in a video for the task of human action recognition. The method is motivated by the observation that there are only a small number of frames which, together, contain sufficient information to discriminate an action class present in a video, from the rest. The proposed method learns to pool such discriminative and informative frames, while discarding a majority of the non-informative frames in a single temporal scan of the video. Our algorithm does so by continuously predicting the discriminative importance of each video frame and subsequently pooling them in a deep learning framework. We show the effectiveness of our proposed pooling method on standard benchmarks where it consistently improves on baseline pooling methods, with both RGB and optical flow based Convolutional networks. Further, in combination with complementary video representations, we show results that are competitive with respect to the state-of-the-art results on two challenging and publicly available benchmark datasets.

1. Introduction

Rapid increase in the number of digital cameras, notably in cellphones, and cheap internet with high data speeds, has resulted in a massive increase in the number of videos uploaded onto the internet [3]. Most of such videos, e.g. on social networking websites, have humans as their central subjects. Automatically predicting the semantic content of videos, e.g. the action the human is performing, thus, becomes highly relevant for searching and indexing in this fast growing database. In order to perform action recognition in such videos, algorithms are required that are both easy and fast to train and, at the same time, are robust to noise, given the real world nature of such videos.

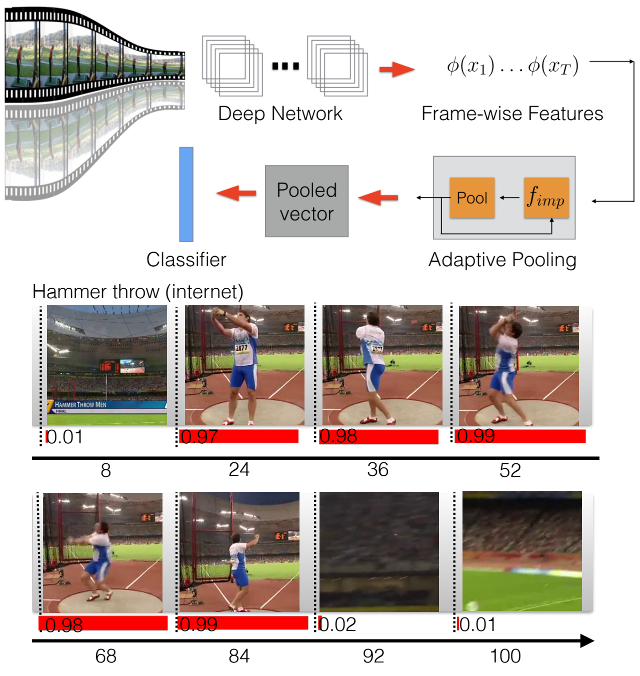

Figure 1: (Top) Illustration of proposed `AdaScan` . It first extracts deep features for each frame in a video and then passes them to the proposed Adaptive Pooling module, which recursively pools them while taking into account their discriminative importances—which are predicted inside the network. The final pooled vector is then used for classification. (Bottom) Predicted discriminative importance for a video that was downloaded from the internet[1] and ran through `AdaScan` trained on UCF101. The numbers and bars on the bottom indicate the predicted importance $\in [0, 1]$ and the timeline gives the relative frame position in percentile (see Section 4.4).

A popular framework for performing human action recognition in videos is using a temporal pooling operation to 'squash' the information from different frames in a video into a summary vector. Mean and max pool-

*Amlan Kar and Nishant Rai contributed equally to this work.
†Part of the work was done when Karan Sikka was with UCSD. `karan.sikka@sri.com`
‡`{amlan, nishantr, grv}@cse.iitk.ac.in`

[1]Video downloaded from `https://www.youtube.com/watch?v=KnHUAc20WEU` and cropped from 3–18 seconds

ing, i.e. taking the average or the coordinatewise max of the (features of the) frames, are popular choices, both with classic 'shallow' as well as recent 'deep' methods [31, 40, 18, 43]. However, these pooling methods consider all frames equally and are not robust to noise, i.e. to the presence of video frames that do not correspond to the target action [22, 7, 1, 20, 48, 52]. This results in a loss in performance as noted by many host algorithms, with both shallow and deep pipelines e.g. [2, 7, 30, 20]. Several methods have proposed solutions to circumvent the limitations of these pooling methods. Such solutions either use Latent Variable Models [22, 36, 9, 30, 19], which require an additional inference step during learning, or employ a variant of Recurrent Neural Networks (RNN) [29, 50] which have intermediate hidden states that are not immediately interpretable. In this work we propose a novel video pooling algorithm that learns to dynamically pool video frames for action classification, in an end-to-end learnable manner, while producing interpretable intermediate 'states'. We name our algorithm `AdaScan` since it is able to both adaptively pool video frames, and make class predictions in a single temporal scan of the video. As shown in Figure 1, our algorithm internally predicts the discriminative importance of each frame in a video and uses these states for pooling. The proposed algorithm is set in a weakly supervised setting for action classification in videos, where labels are provided only at video-level and not at frame-level [22, 52, 30, 20, 2]. This problem is extremely relevant due to the difficulty and non-scalability of obtaining frame-level labels. The problem is also very challenging as potentially noisy and untrimmed videos may contain distractive frames that do not belong to the same action class as the overall video.

Algorithms based on the Multiple Instance Learning (MIL) framework try to solve this problem by alternating between spotting relevant frames in videos and (re-)learning the model. Despite obtaining promising results, MIL is (i) prone to overfitting, and (ii) by design, fails to take into account the contributions of multiple frames together, as noted recently [30, 19]. More recently, Long Short Term Memory (LSTM) networks have also been used for video classification. They encode videos using a recurrent operation and produce hidden vectors as the final representations of the videos [29, 50, 6]. Despite being able to model reasonably long-term temporal dependencies, LSTMs are not very robust to noise and have been shown to benefit from explicit, albeit automatic, removal of noisy frames [10, 52]. The proposed algorithm does not require such external noisy frame pruning as it does so by itself while optimizing the classification performance in a holistic fashion.

In summary we make the following contributions. (1) We propose a novel approach for human action classification in videos that (i) is able to identify informative frames in the video and only pool those, while discarding others, (ii) is end-to-end trainable along with the representation of the images, with the final objective of discriminative classification, and (iii) works in an inductive setting, i.e. given the training set it learns a parametrized function to pool novel videos independently, without requiring the whole training set, or any subset thereof, at test time. (2) We validate the proposed method on two challenging publicly available video benchmark datasets and show that (i) it consistently outperforms relevant pooling baselines and (ii) obtains state-of-the-art performance when combined with complimentary representations of videos. (3) We also analyze qualitative results to gain insights to the proposed algorithm and show that our algorithm achieves high performance while only pooling from a subset of the frames.

2. Related Work

Many earlier approaches relied on using a Bag of Words (BoW) based pipeline. Such methods typically extracted local spatio-temporal features and encoded them using a dictionary [18, 41, 5, 28, 25, 40]. One of the first works [18] described a video with BoW histograms that encoded Histograms of Gradients (HoG) and Histograms of Flow (HoF) features over 3D interest points. Later works improved upon this pipeline in several ways [23, 47] by using dense sampling for feature extraction [41], describing trajectories instead of 3D points [13, 39], and using better pooling and encoding methods [47, 25, 23]. Improving upon these methods Wang et al. [40] proposed the Improved Dense Trajectories (iDT) approach that showed significant improvement over previous methods by using a combination of motion stabilized dense trajectories, histogram based features and Fisher Vector (FV) encodings with spatio-temporal pyramids. Some recent methods have improved upon this pipeline by either using multi-layer fisher vectors [24] or stacking them at multiple temporal scales [17]. All of these approaches rely on the usage of various local features combined with standard pooling operators.

While the above methods worked with an orderless representation, another class of methods worked on explicitly exploiting the spatial and temporal structure of human activities. Out of these, a set of methods have used latent structured SVMs for modeling the temporal structure in human activities. These methods typically alternate between identifying discriminative frames (or segments) in a video (inference step) and learning their model parameters. Niebles et al. [22] modeled an activity as a composition of latent temporal segments with anchor positions that were inferred during the inference step. Tang et al. [36] improved upon Niebles et al. by proposing a more flexible approach using a variable duration HMM that factored each video into latent states with variable durations. Other approaches have also used MIL and its variants to model discriminative frames in a video, with or without a temporal structure [26, 30, 8, 51, 42, 20, 27]. Most related to our work is the dynamic pooling appoach used by Li et al. [20] who used a scoring function to identify discriminative frames in a video and then pooled over only these frames. In contrast, our method does not solve an inference problem, and instead explicitly predicts the discriminative importance of each frame and pools them in a single scan. Our work is also inspired by an early work by Satkin et al. [27] who identified the best temporal boundary of an action, defined as the minimum number of frames required to classify this action, and obtained a final representation by pooling over these frames.

Despite the popularity of deep Convolutional Neural Networks (CNN) in image classification, it is only recently that deep methods have achieved performance comparable to shallow methods for video action classification. Early approaches used 3D convolutions for action recognition [12, 14]; while these showed decent results on the task, the top performances were still obtained by the traditional non-deep methods. Simonyan et al. [31] proposed the two-stream deep network that combined a spatial network (trained on RGB frames) and a temporal network (trained on stacked flow frames) for action recognition. Ng et al. [50] highlighted a drawback in the two-stream network that uses a standard image CNN instead of a specialized network for training videos. This results in the two-stream network not being able to capture long-term temporal information. They proposed two deep networks for action classification by (i) adding standard temporal pooling operations in the network, and (ii) using LSTMs for feature pooling. Recent methods have also explored the use of LSTMs for both predicting action classes [21, 29, 34, 21] and video caption generation [6, 49]. Some of these techniques have also combined attention with LSTM to focus on specific parts of a video (generally spatially) during state transitions [21, 29, 49]. Our work bears similarity to these attention based frameworks in predicting the relevance of different parts of the data. However it differs in several aspects: (i) The attention or disriminative importance utilized in our work is defined over temporal dimension vs. the usual spatial dimension, (ii) we predict this importance score in an online fashion, for each frame, based on the current frame and already pooled features, instead of predicting them together for all the frames [49], and (iii) ours is a simple formulation that combines the prediction with standard mean pooling operation to dynamically pool frame-wise video features. Our work is also related to LSTMs through its recursive formulation but differs in producing a clearly interpretable intermediate state along with the importance of each frame vs. LSTM's generally non-interpretable hidden states. It is also worth mentioning the work on Rank Pooling and Dynamic Image Networks that use a ranking function to pool a video [1, 7]. However, compared to current methods their approach entails a non-trivial intermediate step that requires solving a ranking formulation for pooling each vector.

3. Proposed Approach

We now describe the proposed approach, that we call `AdaScan` (Adaptive Scan Pooling Network), in detail. We denote a video as

$$X = [\mathbf{x}_1, \ldots, \mathbf{x}_T], \mathbf{x}_t \in \mathbb{R}^{224\times224\times K}, \quad (1)$$

with each frame $\mathbf{x}_t$ either represented as RGB images ($K = 3$), or as a stack of optical flow images of neighbouring frames [31] ($K = 20$ in our experiments). We work in a supervised classification setting with a training set

$$\mathcal{X} = \{(X_i, y_i)\}_{i=1}^{N} \subset \mathbb{R}^{224\times224\times K\times T} \times \{1, \ldots, C\}, \quad (2)$$

where X_i is a training video and y_i is its class label (from one of the C possible classes). In the following, we drop the subscript i, wherever it is not required, for brevity.

`AdaScan` is a deep CNN augmented with an specialized pooling module (referred to as 'Adaptive Pooling') that scans a video and dynamically pools the features of select frames to generate a final pooled vector for the video, adapted to the given task of action classification. As shown in Figure 1, our model consists of three modules that are connected to each other sequentially. These three modules serve the following purposes, respectively: (i) feature extraction, (ii) adaptive pooling, and (iii) label prediction. The feature extractor module comprises of all the convolutional layers along with the first fully connected (FC-6) layer of the VGG-16 network of Simonyan et al. [32]. This module is responsible for extracting deep features from each frame $\mathbf{x}_t$ of a video, resulting in a fixed dimensional vector, denoted as $\phi(\mathbf{x}_t) \in \mathbb{R}^{4096}$. The purpose of the Adaptive Pooling module is to selectively pool the frame features by aggregating information from only those frames that are discriminative for the final task, while ignoring the rest. It does so by recursively predicting a score that quantifies the discriminative importance of the current frame, based on (i) the features of the current frame, and (ii) the pooled vector so far. It then uses this score to update the pooled vector (described formally in the next section). This way it aggregates discriminative information only by pooling select frames, whose indices might differ for different videos, to generate the final dynamically pooled vector for the video. This final vector is then normalized using an ℓ_2 normalization layer and the class labels are (predicted) using a FC layer with softmax function. We now describe the adaptive pooling module of `AdaScan` in more detail and thereafter provide details regarding the loss function and learning procedure.

3.1. Adaptive Pooling

This is the key module of the approach which dynamically pools the features of the frames of a video. It does a temporal scan over the video and pools the frames by inferring the discriminative importance of the current frame feature given the feature vector and the pooled video vector so far. In the context of video classification, we want the predicted discriminative importance of a frame to be high if the frame contains information positively correlated to the class of the video, and possibly negatively correlated to the rest of the classes, and low if the frame is either redundant, w.r.t. already pooled frames, or does not contain any useful information for the classification task. We note that this definition of importance is similar to the notion of discriminativeness of a particular part of the data as used in prior MIL based methods. However, contrary to MIL based methods, which effectively weight the frames with a one-hot vector, our algorithm is naturally able to focus on more than one frame in a video, if required, while explicitly outputting the importances of all the frames in an online fashion.

Let us denote the adaptive pooled vector till the initial t frames for a video X as $\psi(X, t)$. The aim is now to compute the vector after pooling all the T frames in a video i.e. $\psi(X, T)$. The Adaptive Pooling module implements the pooling by recursively computing two operations. The first operation, denoted as f_{imp}, predicts the discriminative importance, $\gamma_{t+1} \in [0, 1]$, for the next i.e. $(t+1)^{th}$ frame given its CNN feature, $\phi(\mathbf{x}_{t+1})$, and the pooled features till time t, $\psi(X, t)$. We denote the importance scores of the frames of a video as a sequence of reals $\Gamma = \{\gamma_1, \dots, \gamma_T\} \in [0, 1]$. The second operation is a weighted mean pooling operation that calculates the new pooled features $\psi(X, t+1)$ by aggregating the previously pooled features with the features from current frame and its predicted importance. The operations are formulated as:

$$\gamma_{t+1} = f_{imp}(\psi(X, t), \phi(\mathbf{x}_{t+1})) \tag{3}$$

$$\psi(X, t+1) = \frac{1}{\hat{\gamma}_{t+1}} \left(\hat{\gamma}_t \psi(X, t) + \gamma_{t+1} \phi(\mathbf{x}_{t+1})\right) \tag{4}$$

$$\text{where,} \quad \hat{\gamma}_p = \sum_{k=1}^{p} \gamma_k \tag{5}$$

Effectively, at t^{th} step the above operation does a *weighted* mean pooling of all the frames of a video, with the weights of the frame features being the predicted discriminative importance scores $\gamma_1, \dots, \gamma_t$.

We implement the attention prediction function $f_{imp}(\cdot)$ as a Multilayer Perceptron (MLP) with three layers. As the underlying operations for $f_{imp}(\cdot)$ rely only on standard linear and non-linear operations, they are both fast to compute and can be incorporated easily inside a CNN network for end-to-end learning. In order for $f_{imp}(\cdot)$ to consider both the importance and non-redundancy of a frame we feed the difference between the current pooled features and features from the next frame to the Adaptive Pooling module. We found this simple modification, of feeding the difference, to not only help reject redundant frames but also improve generalization. We believe this is due to the fact that the residual might be allowing the Adaptive Pooling module to explicitly focus on unseen features while making a decision on whether to pool them (additively) or not.

Owing to its design, our algorithm is able to maintain the simplicity of a mean pooling operation while predicting and adapting to the content of each incoming frame. Moreover at every timestep we can easily interpret both the discriminative importance and the pooled vector for a video, leading to an immediate extension to an online/streaming setting, which is not the case for most recent methods.

3.2. Loss Function and Learning

We formulate the loss function using a standard cross entropy loss $\mathcal{L}_{CE}$ between the predicted and true labels. In order to direct the model towards selecting few frames from a video, we add an entropy based regularizer $\mathcal{L}_E$ over the predicted scores, making the full objective as

$$\mathcal{L}(X, y) = \mathcal{L}_{CE}(X, y) + \lambda \mathcal{L}_E(\Gamma) \tag{6}$$

$$\mathcal{L}_E(\Gamma) = -\sum_k \frac{e^{\gamma_k}}{N} \log\left(\frac{e^{\gamma_k}}{N}\right) \tag{7}$$

$$\gamma_k, \lambda \geq 0, N = \sum_t e^{\gamma_t} \tag{8}$$

The regularizer minimizes the entropy over the normalized(using softmax) discriminative scores. Such a regularizer encourages a peaky distribution of the importances, i.e. it helps select only the discriminative frames and discard the non discriminative ones when used with a discriminative loss. We also experimented with the popular sparsity promoting ℓ_1 regularizer, but found it to be too aggressive as it led to selection of very few frames, which adversely affected the performance. The parameter λ is a trade-off parameter which balances between a sparse selection of frames and better minimization of the cross entropy classification loss term. If we set λ to relatively high values we expect fewer number of frames being selected, which would make the classification task harder e.g. single frame per video would make it same as image classification. While, if the value of λ is relatively low, the model is expected to select larger number of frames and also possibly overfit. We show empirical results with varying λ in the experimental Section 4.2.3.

4. Experimental Results

We empirically evaluate our approach on two challenging publicly available human action classification datasets.

We first briefly describe these datasets, along with their experimental protocol and evaluation metrics. We then provide information regarding implementation of our work. Thereafter we compare our algorithm with popular competetive baseline methods. We also study the effect of the regularization used in `AdaScan` and compare our approach with previous state-of-the-art methods on the two datasets. We finally discuss qualitative results to provide important insights to the proposed method.

HMDB51[2] [16] dataset contains around 6800 video clips from 51 action classes. These action classes cover wide range of actions – facial actions, facial action with object manipulations, general body movement, and general body movements with human interactions. This dataset is challenging as it contains many poor quality video with significant camera motions and also the number of samples are not enough to effectively train a deep network [31, 44]. We report classification accuracy for 51 classes across 3 splits provided by the authors [16].

UCF101[3] [33] dataset contains 13320 videos from 101 action classes that are divided into 5 categories- human-object interaction, body-movement only, human-human interaction, playing musical instruments and sports. Action classification in this datasets is challenging owing to variations in pose, camera motion, viewpoint and spatio-temporal extent of an action. Owing to these challenges and higher number of samples this dataset is often used for evaluation in previous works. We report classification accuracy for 101 classes across the 3 train/test splits provided by the authors [33].

4.1. Implementation Details

To implement `AdaScan` , we follow Simonyan et al. [31], and use a two-stream network that consists of a spatial and a temporal 16 layer VGG network [32]. We generate a 20 channel optical flow input, for the temporal network, by stacking both X and Y direction optical flows from 5 neighbouring frames in both directions [31, 44]. We extract the optical flow using the tool[4] provided by Wang et al. [44], that uses TV-L1 algorithm and discretizes the optical flow fields in the range of $[0, 255]$ by a linear transformation. As described in Section 3 our network trains on a input video containing multiple frames instead of a single frame as was done in the two-stream network [31]. Since videos vary in the number of frames and fitting an entire video on a standard GPU is not possible in all cases, we prepare our input by uniformly sampling 25 frames from each video. We augment our training data by following the multiscale cropping technique suggested by [44]. For testing, we use 5 random samples of 25 frames extracted from the video, and use 5 crops of 224×224 along with their flipped versions. We take the mean of these predictions for the final prediction for a sample.

We implement the Adaptive Pooling layer's $f_{imp}(\cdot)$ function, as described in Section 3, using a three layer MLP with $\tanh$ non linearities and sigmoid activation at the final layer. We set the initial state of the pooled vector to be same as the features of the first frame. We found this initialization to be stable as compared to initialization with a random vector. We initialize the components of the Adaptive Pooling module using initialization proposed by Glorot et al. [11]. We also found using the residual of the pooled and current frame vector as input to the Adaptive Pooling module to work better than their concatenation.

We initialize the spatial network for training UCF101 from VGG-16 model [32] trained on ImageNet [4]. For training the temporal network on UCF101, we initialize its convolutional layers with the 16000 iteration snapshot provided by Wang et al. [44]. For training HMDB51 we initialize both the spatial and temporal network by borrowing the convolutional layer weights from the corresponding network trained on UCF101. During experiments we observed that reinitializing the Adaptive Pooling module randomly performed better than initializing with the weights from the network trained on UCF101. We also tried initializing the network trained on HMDB51 with the snapshot provided by [44] and with an ImageNet pre-trained model but found their performance to be worse. Interestingly, from the two other trials, the model initialized with ImageNet performed better, showing that training on individual frames for video classification might lead to less generic features due to the noise injected by the irrelevant frames for an action class. We found it extremely important to use separate learning rates for training the Adaptive Pooling module and fine-tuning the Convolutional layers. We use the Adam solver[15] with learning rates set to $1e-3$ for the Adaptive Pooling module and $1e-6$ for the Convolutional layers. We use dropout with high (drop) probabilities ($= 0.8$) both after the FC-6 layer and the Adaptive Pooling module and found it essential for training. We run the training for 6 epochs for the spatial network on both datasets. We train the temporal network, for 2 epochs on UCF101 and 6 epochs on HMDB51. We implement our network using the tensorflow toolkit[5].

Baselines and complementary features. For a fair comparison with standard pooling approaches, we implement three baselines methods using the same deep network as `AdaScan` with end-to-end learning. We implement mean and max pooling by replacing the Adaptive Pooling module with mean and max operations. For implementing MIL,

[2] `http://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/`
[3] `http://crcv.ucf.edu/data/UCF101.php`
[4] `https://github.com/wanglimin/dense_flow`

[5] https://www.tensorflow.org

Network	Max Pool	MIL	Mean Pool	AdaScan
Spatial	77.2	76.7	78.0	**79.1**
Temporal	80.3	79.1	80.8	**81.7**

Table 1: Comparison with baselines on UCF101 - Split 1 in terms of multiclass classification accuracies.

we first compute classwise scores for each frame in a video and then take a max over the classwise scores across all the frames prior to the softmax layer. For complimentary features we compute results with improved dense trajectories (iDT) [40] and 3D convolutional (C3D) features [37] and report performance using weighted late fusion. We extract the iDT features using the executables provided by the authors [40] and use human bounding boxes for HMDB51 but not for UCF101. We extract FV for both datasets using the implementation provided by Chen et al. [35]. For each low-level feature[6], their implementation first uses Principal Component Analysis (PCA) to reduce the dimensionality to half and then trains a Gaussian Mixture Models (GMM). The GMM dictionaries, of size 512, are used to extract FV by using the `vlfeat` library [38]. The final FV is formed by applying both power normalization and ℓ_2 normalization to per features FV and concatenating them. Although Chen et al. have only provided the GMMs and PCA matrices for UCF101, we also use them for extracting FVs for HMDB51. For computing C3D features we use the Caffe implementation provided by Tran et al. [37] and extract features from the FC-6 layer over a 16 frame window. We compute final feature for each video by max pooling all the features followed by ℓ_2 normalization.

4.2. Quantitative Results

4.2.1 Comparison with Pooling Methods

Table 1 gives the performances of AdaScan along with three other commonly used pooling methods as baselines i.e. max pooling (coordinate-wise max), MIL (multiple instance learning) and mean pooling, on the Split 1 of the UCF101 dataset. MIL is the weakest, followed by max pooling and then mean pooling ($76.7, 77.2, 78.0$ resp. for spatial network and $79.1, 80.3, 80.8$ for the temporal one), while the proposed AdaScan does the best (79.1 and 81.7 for spatial and temporal networks resp.). The trends observed here were typical — we observed that, with our implementations, among the three baselines, mean pooling was consistently performing better on different settings. This could be the case since MIL is known to overfit as a result on focussing only on a single frame in a video [30, 19], while max pooling seems to fail to summarize relevant parts of an actions (and thus overfit) [7]. Hence, in the following experiments we mainly compare with mean pooling.

[6]Trajectory, HOG, HOF, Motion Boundary Histograms (X and Y)

	Spatial network		Temporal network	
Split	Mean Pool	AdaScan	Mean Pool	AdaScan
1	78.0	**79.1**	80.8	**82.3**
2	77.2	**78.2**	82.7	**84.1**
3	77.4	**78.4**	83.7	83.7
Avg	77.6	**78.6**	82.4	**83.4**

UCF101 [33]

	Spatial network		Temporal network	
Split	Mean Pool	AdaScan	Mean Pool	AdaScan
1	41.3	**41.8**	48.8	**49.3**
2	40.3	**41.0**	48.8	**49.8**
3	41.3	**41.4**	48.3	**48.5**
Avg	40.9	**41.4**	48.6	**49.2**

HMDB51 [16]

Table 2: Comparison of AdaScan with mean pooling. We report multiclass classification accuracies.

4.2.2 Detailed Comparison with Mean Pooling

Table 2 gives the detailed comparison between the best baseline of mean pooling with the proposed AdaScan , on the two datasets UCF101 and HMDB51, as well as, the two networks, spatial and temporal. We observe that the proposed AdaScan consistently performs better in all but one case out of the 12 cases. In the only case where it does not improve, it does not deteriorate either. The performance improvement is more with the UCF101 dataset, i.e. 77.6 to 78.6 for the spatial network and 82.4 to 83.4 for the temporal network, on average for the three splits of the datasets. The improvements for the HMDB51 dataset are relatively modest, i.e. 40.9 to 41.4 and 48.6 to 49.2 respectively. Such difference in improvement is to be somewhat expected. Firstly HMDB51 has fewer samples compared to UCF101 for training AdaScan . Also, while UCF101 dataset has actions related to sports, the HMDB51 dataset has actions from movies. Hence, while UCF101 actions are expected to have smaller sets of discriminative frames, e.g. throwing a basketball vs. just standing for it, compared to the full videos, HMDB51 classes are expected to have the discriminative information spread more evenly over all the frames. We could thus expect more improvements in the former case, as observed, by eliminating non-discriminative frames cf. the later where there is not much to discard. A similar trend can be seen in the classes that perform better with AdaScan cf. mean pooling and vice-versa (Figure 2). Classes such as "throw discuss" and "balance beam", which are expected to have the discriminative information concentrated on a few frames, do better with AdaScan while others such as "juggling balls" and "jump rope", where the action is continuously evolving or even periodic and the information is spread out in the whole of the video, do better with mean pooling.

Method	Two-stream	Very deep	LSTM	Attn	Add. Opti.	UCF101	HMDB51
Simonyan et al. [31]	✓					88.0	59.4
Wang et al. [44]	✓					88.0	59.4
Yue et al. [50]	✓	✓				88.2	–
Yue et al. [50]	✓	✓	✓			88.6	–
Wang et al. [43]	✓	✓				90.3	63.2
Sharma et al. [29]		✓	✓	✓		77.0*	41.3
Li et al. [21]	✓	✓	✓	✓		89.2	56.4
Bilen et al. [1]					✓	89.1	65.2
Wang et al. [46]	✓	✓			✓	92.4	62.0
Zhu et al. [52]	✓	✓			✓	93.1	63.3
Wang et al. [45]	✓	✓				94.2	69.4
Tran et al. [37]	3D convolutional filters					83.4	53.9
iDT [40]	shallow					84.3	58.4
MIFS [17]	shallow					88.5	63.8
AdaScan	✓	✓				89.4	54.9
+ iDT	late fusion					91.3	61.0
+ iDT + C3D	late fusion					93.2	66.9

Table 3: Comparison with existing methods (Attn. – Spatial Attention, Add. Opti. – Additional Optimization). (* Results are as reported by [21])

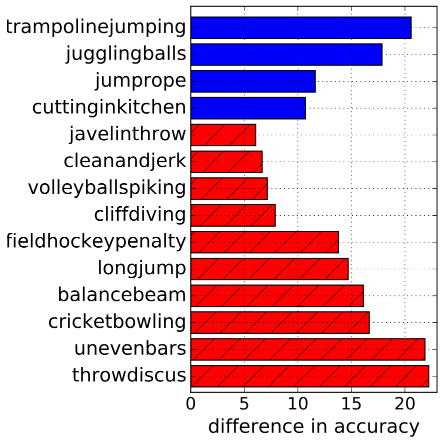

Figure 2: Comparison of AdaScan with mean pooling – example classes where mean pooling is better (blue, top four) and vice-versa (red, all but top four).

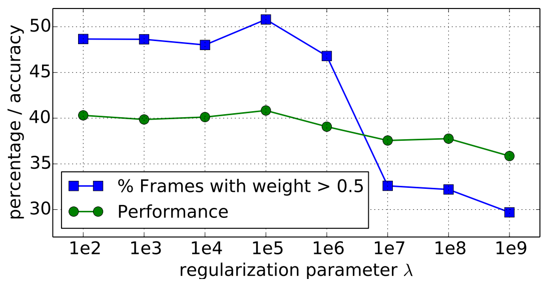

Figure 3: Effect of regularization parameter λ

4.2.3 Effect of Regularization Strength

As discussed in the Section 3.2 above, we have a hyperparameter $\lambda \in \mathbb{R}^+$ which controls the trade-off between noisy frame pruning and model fitting. We now discuss the effect of the λ hyperparameter. To study its effect we trained our spatial network with different λ values on the HMDB51 dataset for 3 epochs to produce the shown results. We see in Figure 3 that for very low regularization ($1e2$ to $1e4$), the model gives an importance (i.e. value of the coordinate corresponding to the frame in the normalized vector Γ of weights) of greater than 0.5 to only about 50% of frames, showing that the architecture in itself holds the capability to filter out frames, probably due to the residual nature of the input to the Adaptive Pooling module. As we increase regularization strength from $1e6$ to $1e7$ we see that we can achieve a drastic increase in sparsity by allowing only a small drop in performance. Subsequently, there is a constant increase in sparsity and corresponding drop in performance. The change in sparsity and performance reduces after $1e7$ because we clip gradients over a fixed norm, thus disallowing very high regularization gradients to flow back through the network. The λ hyperparameter therefore allows us to control the effective number of selected frames based on the importances predicted by the model.

4.3. Comparison with State-of-the-Art

Our model achieves performance competitive with the current state-of-the-art methods (Table 3) when combined with complementary video features on both UCF101 and HMDB51 datasets. We see that AdaScan itself either outperforms or is competitive w.r.t. other methods employing recurrent architectures (LSTMs) with only a single straightforward recurrent operation, without having to employ spatial attention, e.g. (on UCF101) 89.4 for AdaScan vs. $89.2, 77.0$ for [21, 29], or deep recurrent architectures with significant extra pre-training, like 88.6 for [50], demonstrating the effectiveness of the idea. We also show improvements over traditional shallow features, i.e. iDT [43] and MIFS [17], which is in tune with the recent trends in computer vision. Combined with complementary iDT features the performance of AdaScan increases to $91.3, 61.0$ from $89.4, 54.9$, which further goes up to $93.2, 66.9$ for the UCF101 and HMDB51 datasets respectively when combined with C3D features. These are competitive with the existing state-of-the-art results on these datasets.

4.4. Qualitative Results

Figure 4 shows some typical cases (four test videos from split 1 of UCF101) visualized with the output from the proposed AdaScan algorithm. Each frame in these videos is shown with the discriminative importance (value of the $\gamma_t \in [0, 1]$) predicted by AdaScan as a red bar on the bot-

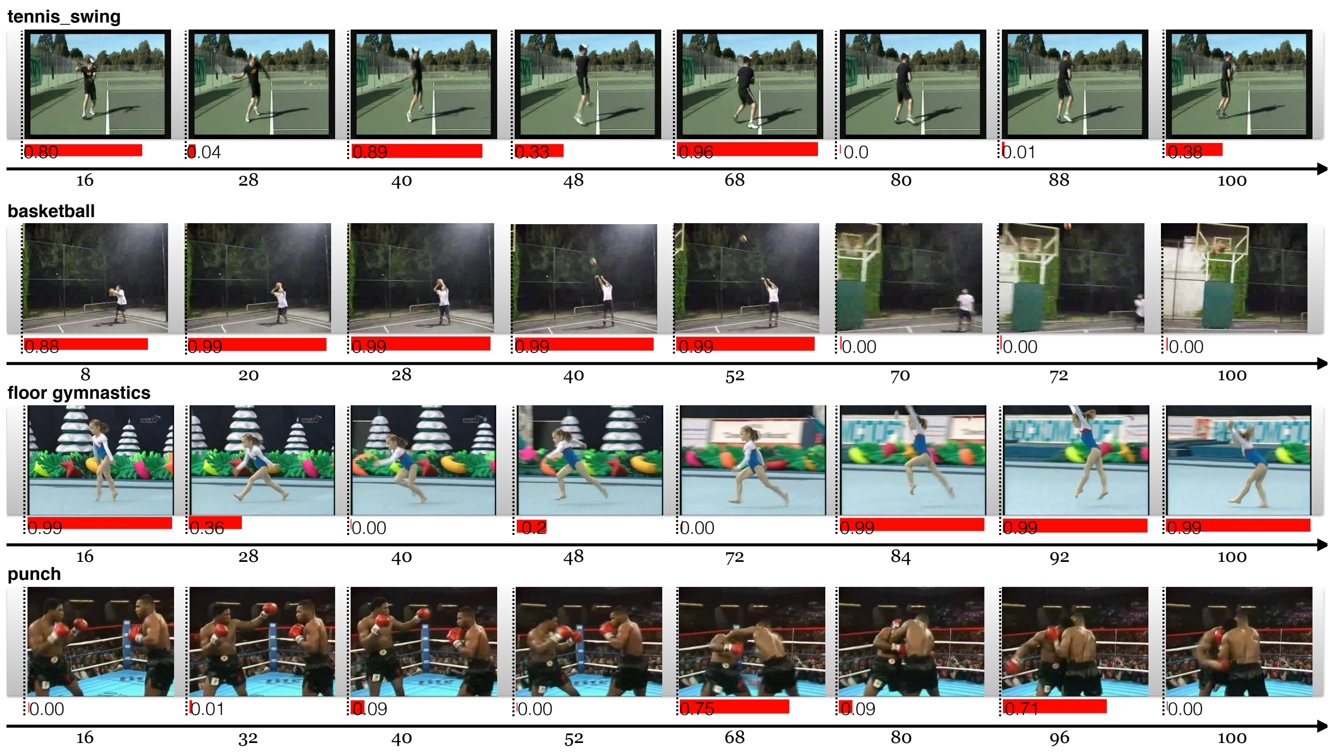

Figure 4: Visualizations of `AdaScan` frame selection. The numbers and red bars below the frames indicate the importance weights. The timeline gives the position of the frame percentile of total number of frames in the video (best seen in colour).

tom of the frame along with the relative (percentile) location of the frame in the whole video. In the "basketball" example we observe that `AdaScan` selects the right temporal boundaries of the action by assigning higher scores to frames containing the action. In the "tennis–swing" example, `AdaScan` selects around three segments in the clip that seem to correspond to (i) movement to reach the ball, (ii) hitting the shot and (iii) returning back to center of the court. We also see a similar trend in the "floor–gymnastics" example, where `AdaScan` selects the temporal parts corresponding to (i) initial preparation, (ii) running and (iii) the final gymnastic act. Such frame selections resonate with previous works that have highlighted the presence of generally 3 atomic actions (or actoms) in actions classes that can be temporally decomposed into finer actions [8]. We also see an interesting property in the "punch" example, where `AdaScan` assigns higher scores to frames where the boxers punch each other. Moreover, it assigns a moderate score of 0.2 to a frame where a boxer makes a failed punch attempt. We have also shown outputs on a video (in Figure 1) that contains "hammer throw" action and was downloaded from the internet. These visualizations strengthen our claim that `AdaScan` is able to adaptively pool frames in a video, by predicting discriminativeness of each frame, while removing frames that are redundant or non-discriminative. We further observe from these visualizations that `AdaScan` also implicitly learns to decompose actions from certain classes into simpler sub-events.

5. Conclusion

We presented an adaptive temporal pooling method, called `AdaScan` , for the task of human action recognition in videos. This was motivated by the observation that many frames are irrelevant for the recognition task as they are either redundant or non-discriminative. The proposed method addresses this, by learning to dynamically pool different frames for different videos. It does a single temporal scan of the video and pools frames in an online fashion. The formulation was based on predicting importance weights of the frames which determine their contributions to the final pooled descriptor. The weight distribution was also regularized with an entropy based regularizer which allowed us to control the sparsity of the pooling operation which in turn helped control the overfitting of the model. We validated the method on two challenging publicly available datasets of human actions, i.e. UCF101 [33] and HMDB51 [16]. We showed that the method outperforms baseline pooling methods of max pooling and mean pooling. It was also found to be better than Multiple Instance Learning (MIL) based deep networks. We also show improvements over previous deep networks that used LSTMs with a much simpler and interpretable recurrent operation. We also showed that the intuitions for the design of the methods were largely validated by qualitative results. Finally, in combination with complementary features, we also showed near state-of-the-art results with the proposed method.

6. Acknowledgements

The authors gratefully acknowledge John Graham from Calit2, UCSD, Robert Buffington from INC, UCSD, Vinay Namboodiri and Gaurav Pandey from IIT Kanpur for access to computational resources, Research-I foundation, IIT Kanpur for support, and Nvidia Corporation for donating a Titan X GPU.

References

[1] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016. 2, 3, 7

[2] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970, 2015. 2

[3] CISCO. White paper: Cisco vni forecast and methodology, 2015-2020. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html`, 2016. 1

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[5] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. IEEE, 2005. 2

[6] 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, 3

[7] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. *TPAMI*, 2016. 2, 3, 6

[8] A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal Localization of Actions with Actoms. *TPAMI*, 35(11):2782–2795, 2013. 3, 8

[9] A. Gaidon, Z. Harchaoui, and C. Schmid. Activity representation with motion hierarchies. *IJCV*, 107(3):219–238, 2014. 2

[10] C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. 2

[11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 5

[12] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, 35(1):221–231, 2013. 3

[13] Y.-G. Jiang, Q. Dai, X. Xue, W. Liu, and C.-W. Ngo. Trajectory-based modeling of human actions with motion reference points. In *ECCV*. Springer, 2012. 2

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 3

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[16] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 5, 6, 8

[17] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015. 2, 7

[18] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 2

[19] W. Li and N. Vasconcelos. Multiple instance learning for soft bags via top instances. In *CVPR*, 2015. 2, 6

[20] W. Li, Q. Yu, A. Divakaran, and N. Vasconcelos. Dynamic pooling for complex event recognition. In *CVPR*, 2013. 2, 3

[21] Z. Li, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. *arXiv preprint arXiv:1607.01794*, 2016. 3, 7

[22] J. C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010. 2

[23] 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 (CVIU)*, 2016. 2

[24] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, pages 581–595. Springer International Publishing, 2014. 2

[25] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010. 2

[26] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *CVPR*, 2013. 3

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

[28] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007. 2

[29] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015. 2, 3, 7

[30] K. Sikka and G. Sharma. Discriminatively trained latent ordinal model for video classification. *arXiv preprint arXiv:1608.02318*, 2016. 2, 3, 6

[31] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 3, 5, 7

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 3, 5

[33] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 5, 6, 8

[34] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR, abs/1502.04681*, 2, 2015. 3

[35] C. Sun and R. Nevatia. Large-scale web video event classification by use of fisher vectors. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 15–22. IEEE, 2013. 6

[36] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *CVPR*. IEEE, 2012. 2

[37] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 6, 7

[38] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008. 6

[39] 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. 2

[40] H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *IJCV*, pages 1–20, 2015. 2, 6, 7

[41] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009. 2

[42] L. Wang, Y. Qiao, and X. Tang. Mining motion atoms and phrases for complex action recognition. In *ICCV*, 2013. 3

[43] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 2, 7

[44] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 5, 7

[45] 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 *European Conference on Computer Vision*, pages 20–36. Springer, 2016. 7

[46] X. Wang, A. Farhadi, and A. Gupta. Actions transformations. In *CVPR*, 2016. 7

[47] X. Wang, L. Wang, and Y. Qiao. A comparative study of encoding, pooling and normalization methods for action recognition. In *ACCV*, 2012. 2

[48] Y. Wang and M. Hoai. Improving human action recognition by non-action classification. In *CVPR*, 2016. 2

[49] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *CVPR*, pages 4507–4515, 2015. 3

[50] 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, 3, 7

[51] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu. Action recognition with actons. In *ICCV*, 2013. 3

[52] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *CVPR*, 2016. 2, 7