This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

ActionVLAD: Learning spatio-temporal aggregation for action classification

Rohit Girdhar[1*] Deva Ramanan[1] Abhinav Gupta[1] Josef Sivic[2,3*] Bryan Russell[2]
[1]Robotics Institute, Carnegie Mellon University [2]Adobe Research [3]INRIA
http://rohitgirdhar.github.io/ActionVLAD

Abstract

In this work, we introduce a new video representation for action classification that aggregates local convolutional features across the entire spatio-temporal extent of the video. We do so by integrating state-of-the-art two-stream networks [42] with learnable spatio-temporal feature aggregation [6]. The resulting architecture is end-to-end trainable for whole-video classification. We investigate different strategies for pooling across space and time and combining signals from the different streams. We find that: (i) it is important to pool jointly across space and time, but (ii) appearance and motion streams are best aggregated into their own separate representations. Finally, we show that our representation outperforms the two-stream base architecture by a large margin (13% relative) as well as outperforms other baselines with comparable base architectures on HMDB51, UCF101, and Charades video classification benchmarks.

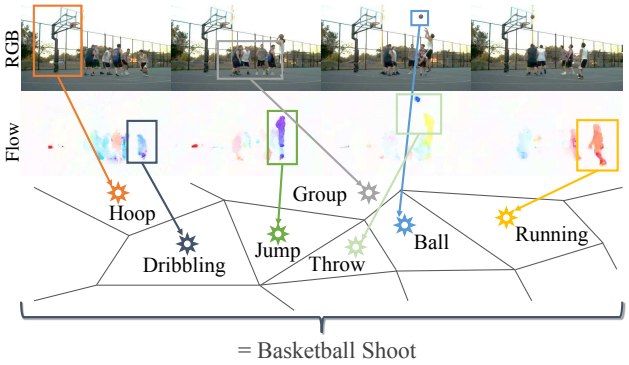


Figure 1: How do we represent actions in a video? We propose ActionVLAD, a spatio-temporal aggregation of a set of action primitives over the appearance and motion streams of a video. For example, a basketball shoot may be represented as an aggregation of appearance features corresponding to 'group of players', 'ball' and 'basketball hoop'; and motion features corresponding to 'run', 'jump', and 'shoot'. We show examples of primitives our model learns to represent videos in Fig. 6.

1. Introduction

Human action recognition is one of the fundamental problems in computer vision with applications ranging from video navigation and movie editing to human-robot collaboration. While there has been great progress in classification of objects in still images using convolutional neural networks (CNNs) [19, 20, 43, 47], this has not been the case for action recognition. CNN-based representations [15, 51, 58, 59, 63] have not yet significantly outperformed the best hand-engineered descriptors [12, 53]. This is partly due to missing large-scale video datasets similar in size and variety to ImageNet [39]. Current video datasets are still rather small [28, 41, 44] containing only on the order of tens of thousands of videos and a few hundred classes. In addition, those classes may be specific to certain domains, such as sports [44], and the dataset may contain noisy labels [26]. Another key open question is: what is the appropriate *spatiotemporal representation* for modeling videos? Most recent video representations for action recognition are primarily based on two different CNN architectures: (1) 3D spatio-temporal convolutions [49, 51] that potentially learn complicated spatio-temporal dependencies but have been so far hard to scale in terms of recognition performance; (2) Two-stream architectures [42] that decompose the video into motion and appearance streams, and train separate CNNs for each stream, fusing the outputs in the end. While both approaches have seen rapid progress, two-stream architectures have generally outperformed the spatio-temporal convolution because they can easily exploit the new ultra-deep architectures [19, 47] and models pre-trained for still-image classification.

However, two-stream architectures largely disregard the long-term temporal structure of the video and essentially learn a classifier that operates on individual frames or short blocks of few (up to 10) frames [42], possibly enforcing consensus of classification scores over different segments of the video [58]. At test time, T (typically 25) uniformly

*Work done at Adobe Research during RG's summer internship

sampled frames (with their motion descriptors) are classified independently and the classification scores are averaged to get the final prediction. This approach raises the question whether such temporal averaging is capable of modelling the complex spatio-temporal structure of human actions. This problem is exaggerated when the same sub-actions are shared among multiple action classes. For example, consider a complex composite action of a 'basketball shoot' shown in Figure 1. Given only few consecutive frames of video, it can be easily confused with other actions, such as 'running', 'dribbling', 'jumping' and 'throwing'. Using late fusion or averaging is not the optimal solution since it requires frames belonging to same sub-action to be assigned to multiple classes. What we need is a global feature descriptor for the video which can aggregate evidence *over the entire video* about both the appearance of the scene and the motion of people without requiring every frame to be uniquely assigned to a single action.

To address this issue we develop an end-to-end *trainable* video-level representation that aggregates convolutional descriptors across different portions of the imaged scene and across the entire temporal span of the video. At the core of this representation is a spatio-temporal extension of the NetVLAD aggregation layer [6] that has been shown to work well for instance-level recognition tasks in still images. We call this new layer **ActionVLAD**. Extending NetVLAD to video brings the following two main challenges. First, what is the best way to aggregate frame-level features across time into a video-level representation? To address this, we investigate aggregation at different levels of the network ranging from output probabilities to different layers of convolutional descriptors and show that aggregating the last layers of convolutional descriptors performs best. Second, how to best combine the signals from the different streams in a multi-stream architecture? To address this, we investigate different strategies for aggregating features from spatial and temporal streams and show, somewhat surprisingly, that best results are obtained by aggregating spatial and temporal streams into their separate single video-level representations. We support our investigations with quantitative experimental results together with qualitative visualizations providing the intuition for the obtained results.

2. Related Work

Action recognition is a well studied problem with standard datasets [4, 8, 18, 28, 41, 44] focused on tasks such as classification [15, 42, 53, 58, 59] and temporal or spatio-temporal localization [25, 60, 62]. Action recognition is, however, hard due to the large intra-class variability of different actions and difficulty of annotating large-scale training datasets. As a result, the performance of automatic recognition methods is still far below the capability of human vision. In this paper we focus on the problem of action classification, i.e., classifying a given video clip into one of K given actions classes. We review the main approaches to this problem below followed by a brief review of feature aggregation.

Dense trajectories: Up until recently, the dominating video representation for action recognition has been based on extracting appearance (such as histograms of image gradients [10]) and motion features (such as histogram of flow [11]) along densely sampled point trajectories in video. The descriptors are then aggregated into a bag-of-visual-words like representation resulting in a fixed-length descriptor vector for each video [52, 54]. The representation can be further improved by compensating for unwanted camera motions [53]. This type of representation, though shallow, is still relevant today and is in fact part of the existing state-of-the-art systems [15, 51, 55]. We build on this work by performing a video-level aggregation of descriptors where both the descriptors and the parameters for aggregation are jointly learned in a discriminative fashion.

Convolutional neural networks: Recent work has shown several promising directions in learning video representations directly from data using convolutional neural networks. For example, Karpathy *et al.* [26] showed the first large-scale experiment on training deep convolutional neural networks from a large video dataset, Sports-1M. Simonyan and Zisserman[42] proposed the two-stream architecture, thereby decomposing a video into appearance and motion information. Wang *et al.* [58] further improved the two-stream architecture by enforcing consensus over predictions in individual frames. Another line of work has investigated video representations based on spatio-temporal convolutions [49, 51], but these methods have been so far hard to scale to long videos (maximum of 120 frames in [51]), limiting their ability to learn over the entire video.

Modelling long-term temporal structure: Some methods explicitly model the temporal structure of the video using, for example, grammars [36, 40] but are often limited to constrained set-ups such as sports [21], cooking [38], or surveillance [5]. More related to our approach, the temporal structure of the video can be also represented implicitly by an appropriate aggregation of descriptors across the video [16, 29, 31, 34, 55, 61]. For example, Ng *et al.* [31] combine information across frames using LSTMs. Xu *et al.* [61] use features from fc7 ImageNet pre-trained model and use VLAD [23] for aggregation, and show improvements for video retrieval on TRECVID datasets [8]. However, their method is not end-to-end trainable and is used as a post-processing step. Other works in event detection and action classification rely on pooling hand crafted features over segments of videos [17, 30, 32, 37, 48]. Others have also investigated pooling convolutional descriptors from video using, for example, variants of Fisher Vectors [29, 34] or pool-

ing along point trajectories [55]. In contrast to these methods, we develop an end-to-end trainable video architecture that combines recent advances in two-stream architectures with a trainable spatio-temporal extension of the NetVLAD aggregation layer, which to the best of our knowledge, has not be done before. In addition, we compare favourably the performance of our approach with the above pooling methods in Section 4.6.

Feature aggregation: Our work is also related to feature aggregation such as vectors of locally aggregated descriptors (VLAD) [23] and Fisher vectors (FV) [35, 46]. Traditionally, these aggregation techniques have been applied to keypoint descriptors as a post processing step, and only recently have been extended to end-to-end training within a convolutional neural network for representing still images [6]. We build on this work and extend it to an end-to-end trainable video representation for action classification by feature aggregation over space and time.

Contributions: The contribution of this paper are threefold: (1) We develop a powerful video-level representation by integrating trainable spatio-temporal aggregation with state-of-the-art two-stream networks. (2) We investigate different strategies for pooling across space and time as well as combining signals from the different streams providing insights and experimental evidence for the different design choices. (3) We show that our final representation outperforms the two-stream base architecture by a large margin (13% relative) as well as outperforms other baselines with comparable base architectures on HMDB51, UCF101, and Charades video classification benchmarks.

3. Video-level two-stream architecture

We seek to learn a representation for videos that is trainable end-to-end for action recognition. To achieve this we introduce an architecture outlined in Figure 2. In detail, we sample frames from the entire video, and aggregate features from the appearance (RGB) and motion (flow) streams using a vocabulary of "action words" into a single video-level fixed-length vector. This representation is then passed through a classifier that outputs the final classification scores. The parameters of the aggregation layer – the set of "action words" – are learnt together with the feature extractors in a discriminative fashion for the end task of action classification.

In the following we first describe the learnable spatio-temporal aggregation layer (Sec. 3.1). We then discuss the possible placements of the aggregation layer in the overall architecture (Sec. 3.2) and strategies for combining appearance and motion streams (Sec. 3.3). Finally, we give the implementation details (Sec. 3.4).

3.1. Trainable Spatio-Temporal Aggregation

Consider $x_{i,t} \in R^D$, a D-dimensional local descriptor extracted from spatial location $i \in \{1 \dots N\}$ and frame $t \in \{1 \dots T\}$ of a video. We would like to aggregate these descriptors both spatially and temporally over the whole video while preserving their informative content. This is achieved by first dividing the descriptor space R^D into K cells using a vocabulary of K "action words" represented by anchor points $\{c_k\}$ (Fig. 3 (c)). Each video descriptor $x_{i,t}$ is then assigned to one of the cells and represented by a residual vector $x_{it} - c_k$ recording the *difference* between the descriptor and the anchor point. The difference vectors are then summed across the entire video as

$$V[j,k] = \sum_{t=1}^{T} \sum_{i=1}^{N} \underbrace{\frac{e^{-\alpha||x_{it}-c_k||^2}}{\sum_{k'} e^{-\alpha||x_{it}-c_{k'}||^2}}}_{\text{Soft-assignment}} \underbrace{(x_{it}[j] - c_k[j])}_{\text{Residual}}, \tag{1}$$

where $x_{it}[j]$ and $c_k[j]$ are the j-th components of the descriptor vector x_{it} and anchor c_k, respectively, and α is a tunable hyper-parameter. Note that the first term in (1) represents the soft-assignment of descriptor x_{it} to cell k and the second term, $x_{it}[j] - c_k[j]$, is the residual between the descriptor and the anchor point of cell k. The two summing operators represent aggregation over time and space, respectively. The output is a matrix V, where k-th column $V[\cdot, k]$ represents the aggregated descriptor in the k-th cell. The columns of the matrix are then intra-normalized [7], stacked, and L2-normalized [23] into a single descriptor $v \in R^{KD}$ of the entire video.

The intuition is that the residual vectors record the differences of the extracted descriptors from the "typical actions" (or sub-actions) represented by anchors c_k. The residual vectors are then aggregated across the entire video by computing their sum inside each of the cells. Crucially, all parameters, including the feature extractor, action words $\{c_k\}$, and classifier, are jointly learnt from the data in an end-to-end manner so as to better discriminate target actions. This is because the spatio-temporal aggregation described in (1) is differentiable and allows for back-propagating error gradients to lower layers of the network. Note that the outlined aggregation is a spatio-temporal extension of the NetVLAD [6] aggregation, where we, in contrast to [6], introduce the sum across time t. We refer to our spatio-temporal extension as **ActionVLAD**.

Discussion: It is worth noting the differences of the above aggregation compared to the more common average or max-pooling (Figure 3). Average or max-pooling represent the entire distribution of points as only a single descriptor which can be sub-optimal for representing an entire video composed of multiple sub-actions. In contrast, the proposed video aggregation represents an entire distribution of descriptors with multiple sub-actions by splitting

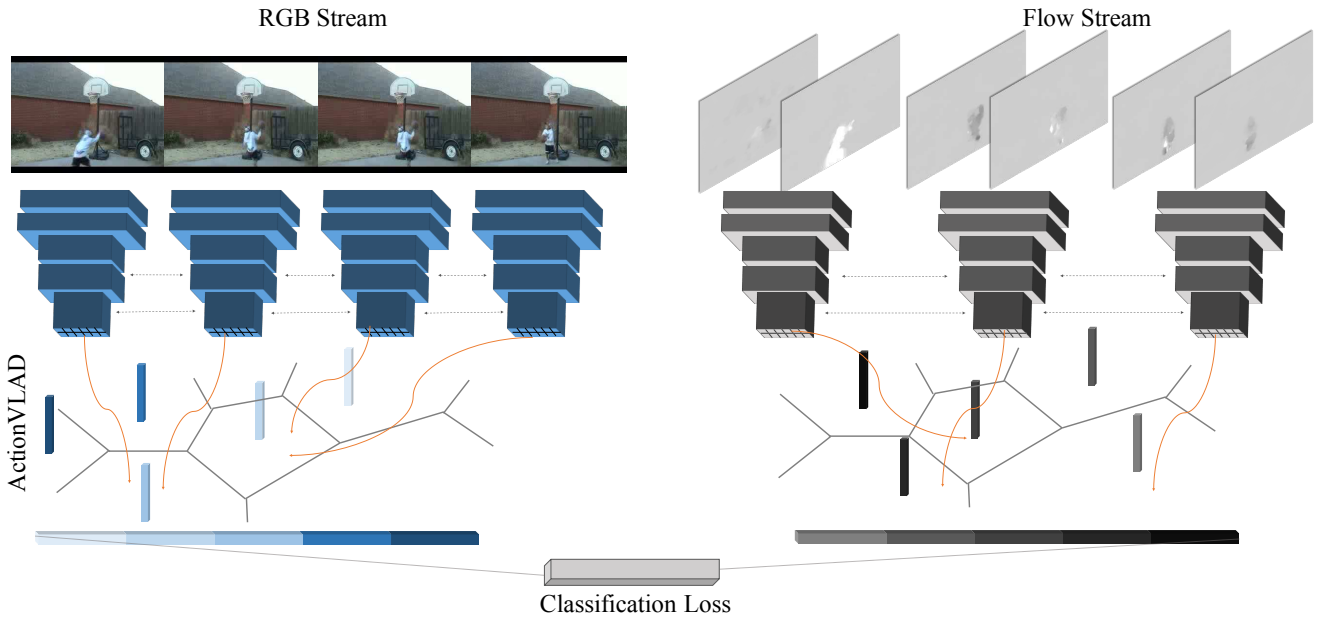


Figure 2: Our network architecture. We use a standard CNN architecture (VGG-16) to extract features from sampled appearance and motion frames from the video. These features are then pooled across space and time using the ActionVLAD pooling layer, which is trainable end to end with a classification loss. We also experiment with ActionVLAD to fuse the two streams (Sec. 3.3).

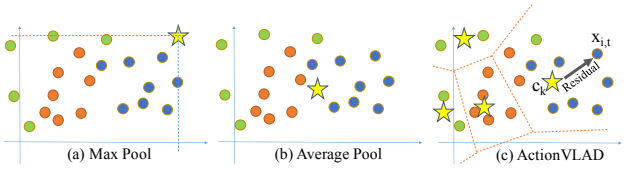


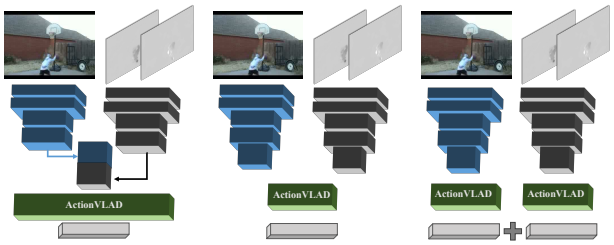
Figure 3: Different pooling strategies for a collection of diverse features. Points correspond to features from a video, and different colors correspond to different sub-actions in the video. While (a) max or (b) average pooling are good for similar features, they do not not adequately capture the complete distribution of features. Our representation (c) clusters the appearance and motion features and aggregates their residuals from nearest cluster centers.

the descriptor space into cells and pooling inside each of the cells. In theory, a hidden layer between a descriptor map and pooling operation could also split up the descriptor space into half-spaces (by making use of ReLU activations) before pooling. However, it appears difficult to train hidden layers with dimensions comparable to ours $KD = 32,768$. We posit that the ActionVLAD framework imposes strong regularization constraints that makes learning of such massive models practical with limited training data (as is the case for action classification).

3.2. Which layer to aggregate?

In theory, the spatio-temporal aggregation layer described above can be placed at any level of the network to pool the corresponding feature maps. In this section we describe the different possible choices that will later guide our experimental investigation. In detail, we build upon the two-stream architecture introduced in Simonyan and Zisserman [42] over a VGG16 network [43]. Here we consider only the appearance stream, but discuss the different ways of combining the appearance and motion streams with our aggregation in Section 3.3.

Two-stream models first train a frame-level classifier using all the frames from all videos, and at test time, average the predictions from T uniformly sampled frames [42, 58]. We use this base network (pre-trained on frame-level) as a feature generator that provides input from different frames to our trainable ActionVLAD pooling layer. But, which layer's activations do we pool? We consider two main choices. First, we consider pooling the output of fully connected (FC) layers. Those are represented as 1×1 spatial feature maps with 4096-dimensional output for each of the T frames of the video. In other words we pool one 4096-dimensional descriptor from each of the T frames of the video. Second, we consider pooling features from the convolutional layers (we consider conv4_3 and conv5_3). For conv5_3, for example, those are represented by 14×14 spatial feature maps each with 512-dimensional descriptors for each of the T frames, i.e. we pool 196 512-dimensional descriptors from each of the T frames. As we show in Section 4.3, we obtain best performance by pooling features at the highest convolutional layer (conv5_3 for VGG-16).



(a) Concat Fusion (b) Early Fusion (c) Late Fusion

Figure 4: Different strategies for combining the appearance and motion streams.

3.3. How to combine Flow and RGB streams?

ActionVLAD can be also used to pool features across the different streams of input modalities. In our case we consider appearance and motion streams [42], but in theory pooling can be done across any number of other data streams, such as warped flow or RGB differences [58]. There are several possible ways to combine the appearance and motion streams to obtain a jointly trainable representation. We explore the most salient ones in this section and outline them in Figure 4.

Single ActionVLAD layer over concatenated appearance and motion features (Concat Fusion). In this case, we concatenate the corresponding output feature maps from appearance and motion, essentially assuming their spatial correspondence. We place a single ActionVLAD layer on top of this concatenated feature map, as shown in Figure 4(a). This allows correlations between appearance and flow features to be exploited for codebook construction.

Single ActionVLAD layer over all appearance and motion features (Early Fusion). We also experiment with pooling all features from appearance and motion streams using a single ActionVLAD layer, as illustrated in Figure 4(b). This encourages the model to learn a single descriptor space x_{ij} for both appearance and motion features, and hence exploit redundancy in the features.

Late Fusion. This strategy, shown Figure 4(c), follows the standard testing practice of weighted average of the appearance and motion last layer features. Hence, we have two separate ActionVLAD layers, one for each stream. This allows for both ActionVLAD layers to learn specialized representations for each input modality.

We compare the performance of the different aggregation techniques in Section 4.5.

3.4. Implementation Details

We train all our networks with a single-layer linear classifier on top of our ActionVLAD representation described above. Throughout, we use $K = 64$ and a high value for $\alpha = 1000.0$, similar to [6]. Since the output feature dimensionality can be large, we use a dropout of 0.5 over the representation to avoid overfitting to small action classification datasets. We train the network with cross-entropy loss, where the probabilities are obtained through a softmax. Similar to [6], we decouple ActionVLAD parameters $\{c_k\}$ used to compute the soft assignment and the residual from (1) to simplify learning (though both sets of parameters are identically initialized with the same cluster centers).

We use $T = 25$ frames per video (for both flow and RGB) to learn and evaluate our video representation, in order to be comparable to standard practice [42, 58]. Flow is represented using 10 consecutive x and y direction flow maps, to get a 20-channel input. Since our model is trained at a video level, we can fit very few videos every iteration due to limited GPU memory and CPU preprocessing capability. To maintain reasonable batch sizes, we use slow updates by averaging gradients over multiple GPU iterations. We clip gradients at 5.0 L2 norm. Data augmentation is done at the video level by performing random cropping/flipping of all the RGB and flow frames correspondingly. When training ActionVLAD, we use the Adam solver [27] with $\epsilon = 10^{-4}$. This is required as the ActionVLAD output is L2 normalized and we need a lower ϵ value for reasonably fast convergence. We perform training in a two-step approach. In the first step, we initialize and fix the VLAD cluster centers, and only train the linear softmax classifier with a learning rate of 0.01. In the second step, we jointly finetune both the linear classifier and the ActionVLAD cluster centers with a learning rate of 10^{-4}. Our experiments show that this uniformly gives a significant boost in validation accuracy (Table 1), indicating that ActionVLAD does adapt clusters to better represent the videos. When training ActionVLAD over conv5_3, we keep layers before conv5_1 fixed to avoid overfitting to small action classification datasets. This also helps by having a smaller GPU memory footprint and faster training. That said, our model is completely capable of end-to-end training, which can be exploited for larger, more complex datasets. We implemented our models in TensorFlow [3] and have released our code and pretrained models [1].

4. Experiments

In this section we experiment with the various network architectures proposed above on standard action classification benchmarks.

4.1. Datasets and Evaluation

We evaluate our models on two popular trimmed action classification benchmarks, UCF101 [44] and HMDB51 [28]. UCF101 consists of 13320 sports video clips from 101 action classes, and HMDB51 contains 6766 realistic and varied video clips from 51 action classes. We follow the evaluation scheme from THUMOS13 chal-

Table 1: Evaluation of training ActionVLAD representation using VGG-16 architecture on HMDB51 split 1.

Method	Appearance	Motion
Two Stream [15]	47.1	55.2
Non-trained ActionVLAD + Linear Classifier	44.9	55.6
Trained ActionVLAD	51.2	58.4

lenge [24] and employ the provided three train/test splits for evaluation. We use the split 1 for ablative analysis and report final performance averaged over all 3 splits. Finally, we also evaluate our model on an untrimmed dataset, Charades [41]. Since a video in Charades can have multiple labels, the evaluation is performed using mAP and weighted average precision (wAP), where AP of each class is weighted by class size.

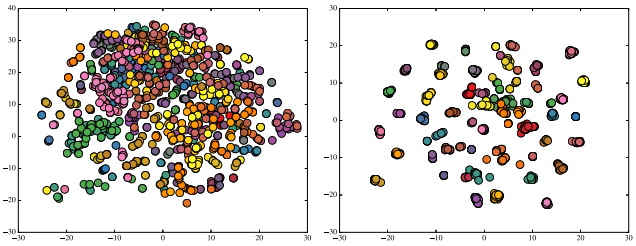
4.2. Training ActionVLAD Representation

To first motivate the strength of the trainable ActionVLAD representation, we compare it to non-trained ActionVLAD extracted from video. In particular, we train only a single classifier over the ActionVLAD layer (over conv5_3) initialized by k-means (and kept fixed). Next, we start from the above model and also train the parameters of the layers including and before ActionVLAD, upto conv5_1. As shown in Table 1, removing the last 2-layer non-linear classifier from the two-stream model and training a single linear layer over the (fixed) VLAD pooled descriptor already gets close to the two-stream performance. This improves even further when the parameters of ActionVLAD are trained together with the preceding layers. Throughout, we use the default value of $K = 64$ following [6]. Initial experiments (HMDB51 split 1 RGB) have shown stable performance on varying K (49.1%, 51.2% and 51.1% for $K = 32$, 64 and 128 respectively). Reducing K to 1, however, leads to much lower performance, 43.2%.

We also visualize the classes for which we obtained the highest improvements compared to the standard two-stream model. We see the largest gains for classes such as 'punch', which were often confused with 'kick' or 'hit'; 'climb_stairs', often confused with 'pick' or 'walk'; 'hit', often confused with 'golf'; and 'drink', often confused with 'eat' or 'kiss'. This is expected, as these classes are easy to confuse given only local visual evidence. For example, hitting and punching are easily confused when looking at few frames but can be disambiguated when aggregating information over the whole video. We present the whole confusion matrix highlighting the pairs of classes with the highest changes of performance in the appendix [2].

4.3. Where to ActionVLAD?

Here we evaluate the different positions in the network where we can insert the ActionVLAD layer. Specifically, we compare placing ActionVLAD after the last two convolutional layers (conv4_3 and conv5_3) and after the last fully-connected layer (fc7). In each case, we train until the block of layers just before the ActionVLAD layer; so conv4_1 to loss in case of conv4, and fc7 to loss in case of fc7. Results are shown in Table 2a and clearly show that best performance is obtained by aggregating the last convolutional layer (conv5_3). fc6 features obtain performance similar to fc7, getting 42.7% for RGB.



conv5_3 fc7

Figure 5: tSNE embedding of subsampled conv and fc features from a set of videos. Features belonging to the same video have the same color. Note that the fc features are already very similar and hence cluster together, while the conv5_3 features are more spread out and mixed.

We believe that this is due to two reasons. First, pooling at a fully-connected layer prevents ActionVLAD from modelling spatial information, as these layers already compress a lot of information. Second, fc7 features are more semantic and hence features from different frames are already similar to each other not taking advantage of the modelling capacity of the ActionVLAD layer, i.e. they would often fall into the same cell. To verify this hypothesis we visualize the conv5_3 and fc7 appearance features from the same frames using a tSNE [50] embedding in Figure 5. The plots clearly show that fc7 features from the same video (shown in the same color) are already similar to each other, while conv5_3 features are much more varied and ready to benefit from the ability of the ActionVLAD layer to capture complex distributions in the feature space as discussed in Sec. 3.1.

4.4. Baseline aggregation techniques

Next, we compare our ActionVLAD aggregation with baseline aggregation techniques. As discussed earlier, average or max pooling reduce the distribution of features in the video into one single point in the feature space, which is suboptimal for the entire video. This is supported by our results in Table 2b, where we see a significant drop in performance using average/max pooling over con5_3 features, even compared to the baseline two-stream architecture.

Table 2: Evaluation of (a) ActionVLAD at different positions in a VGG-16 network; and (b) ActionVLAD compared to other pooling strategies, on HMDB51 split 1.

(a) Position of ActionVLAD.

Method	RGB	Flow
2-Stream	47.1	55.2
conv4_3	45.0	53.5
conv5_3	51.2	58.4
fc7	43.3	53.1

(b) Different pooling strategies.

Method	RGB	Flow
2-Stream	47.1	55.2
Avg	41.6	53.4
Max	41.5	54.6
ActionVLAD	51.2	58.4

Table 3: Comparison of (a) Different fusion techniques described in Sec. 3.3 on HMDB split 1; and (b) Comparison of two-stream with ActionVLAD, averaged over 3-splits of HMDB.

(a) Fusion Type

Method	Val acc
Concat	56.0
Early	64.8
Late	66.9

(b) Overall Comparison

Stream	2-Stream [59]	Ours
RGB	42.2	49.8
Flow	55.0	59.1
Late Fuse	58.5	66.3

4.5. Combining Motion and Appearance

We compare the different combination strategies in Table 3a. We observe that late fusion performs best. To further verify this observation, we show a tSNE plot of conv5 features from appearance and motion streams in the appendix [2]. The two features are well separated, indicating there is potentially complementary information that is best exploited by fusing later in the network. In contrast, concat fusion limits the modelling power of the model as it uses the same number of cells to capture a larger portion of the feature space. Finally, we compare our overall performance on three splits of HMDB51 with the two-stream baseline in Table 3b. We see significant improvements over each input modality as well as over the final (late-fused) vector.

4.6. Comparison against State of the Art

In Table 4, we compare our approach to a variety of recent action recognition methods that use a comparable base architecture to ours (VGG-16). Our model outperforms all previous approaches on HMDB51 and UCF101, using comparable base architectures when combined with iDT. Note that similar to 10-crop testing in two-stream models, our model is also capable of pooling features from multiple crops at test time. We report our final performance using 5 crops, or 125 total images, per video. Other methods such as [14] and [58] based on ultra-deep architectures such as ResNet [19] and Inception [22] respectively obtain higher performance. However, it is interesting to note that our model still outperforms some of these ultra-deep two-stream baselines, such as ResNet-50 (61.2% on HMDB51 and 91.7% on UCF101) and ResNet-152 (63.8% and 91.8%) [14][1], while employing only a VGG-16 network with ActionVLAD (66.9% and 92.7%). We evaluate our method on Charades [41] in Tab. 5, and outperform all previous reported methods (details in appendix [2]).

Table 4: Comparison with the state of the art on UCF101 and HMDB51 datasets averaged over 3 splits. First section compares all ConvNet-based methods reported using VGG-16 or comparable models. Second section compares methods that use iDT [53], and third section reports methods using ultra deep architectures, multi-modal inputs (more than RGB+Flow) and hybrid methods.

	UCF101	HMDB51
Spatio-Temporal ConvNet [26]	65.4	-
LRCN [13]	82.9	-
C3D [49]	85.2	-
Factorized ConvNet [45]	88.1	59.1
VideoDarwin [16]	-	63.7
Two-Stream + LSTM [31] (GoogLeNet)	88.6	-
Two-Stream ConvNet [42] (VGG-M)	88.0	59.4
Two-Stream ConvNet [57, 59] (VGG-16)	91.4	58.5
Two-Stream Fusion [15] (VGG-16)	92.5	65.4
TDD+FV [55]	90.3	63.2
RNN+FV [29]	88.0	54.3
Transformations [59]	92.4	62.0
LTC [51]	91.7	64.8
KVMF [63]	**93.1**	63.3
ActionVLAD (LateFuse, VGG-16)	92.7	**66.9**
DT+MVSV [9]	83.5	55.9
iDT+FV [53]	85.9	57.2
iDT+HSV [33]	87.9	61.1
MoFAP [56]	88.3	61.7
C3D+iDT [49]	90.4	-
Two-Stream Fusion+iDT [15]	93.5	69.2
LTC+iDT [51]	92.7	67.2
ActionVLAD (VGG-16) + iDT	**93.6**	**69.8**
TSN (BN-Inception, 3-modality) [58]	94.2	69.4
DT+Hybrid architectures [12]	92.5	70.4
ST-ResNet+iDT [14]	94.6	70.3

Table 5: Comparison with the state of the art on Charades [41] using mAP and weighted-AP (wAP) metrics.

	mAP	wAP
Two-stream + iDT (best reported) [41]	18.6	-
RGB stream (BN-inception, TSN [56] style training)	16.8	23.1
ActionVLAD (RGB only, BN-inception)	17.6	25.1
ActionVLAD (RGB only, BN-inception) + iDT	**21.0**	**29.9**

4.7. Qualitative Analysis

Finally, we visualize what our model has learned. We first visualize the learned 'action words', to understand the primitives our model uses to represent actions. We randomly picked few thousand frames from HMDB videos, and computed ActionVLAD assignment maps for conv5_3 features by taking the max over soft-assignment in Eq. 1. These maps define which features get soft-assigned to which of the 64 'action words'. Then for each action word, we visualize the frames that contain that action word, and highlight regions that correspond to center of receptive field

[1] Reported on http://www.robots.ox.ac.uk/~vgg/software/two_stream_action/

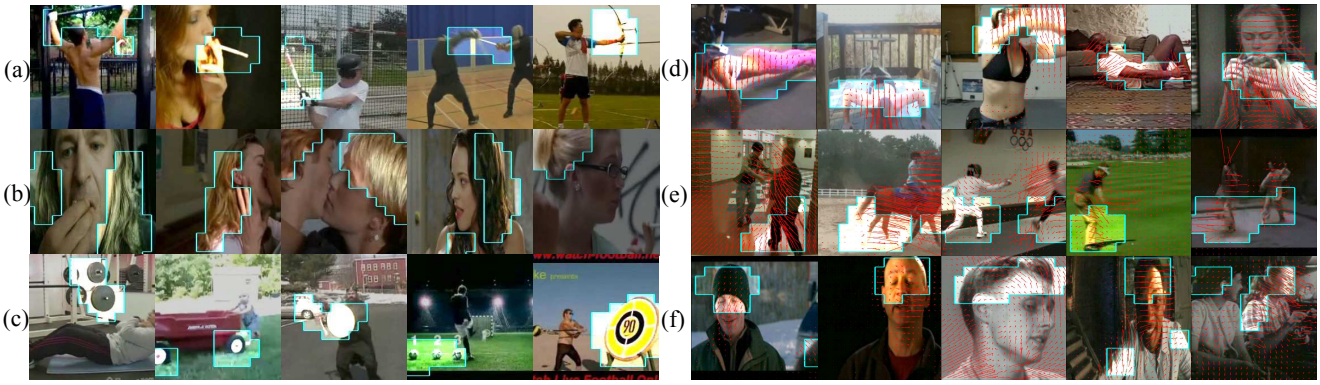


Figure 6: Visualization of 'action words' our ActionVLAD model learns when trained for appearance and motion modalities. Each row shows several frames from videos where bright regions correspond to the center of the receptive field for conv5_3 features that get assigned to one specific 'action word' cell. In detail, (a) shows an 'action word' that looks for human hands holding rod-like objects, such as a pull-up bar, baseball bat or archery bow. (b) looks for human hair. (c) looks for circular objects like wheels and target boards. (d)-(f) shows similar action words for the flow stream. These are more complex, as each word looks at shape and motion of the flow field over 10 frames. Here we show some easy to interpret cases, such as (d) up-down motion, (e) linear motion for legs and (f) head motion.

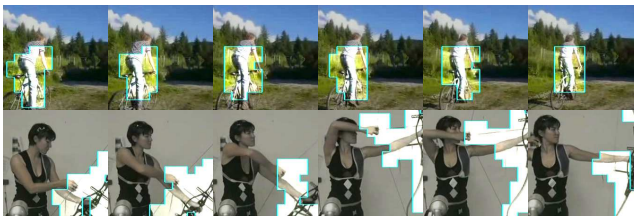


Figure 7: Assignment of image regions to two specific appearance 'action words' over several video frames. ActionVLAD representation "tracks" these visual and flow patches over the video.

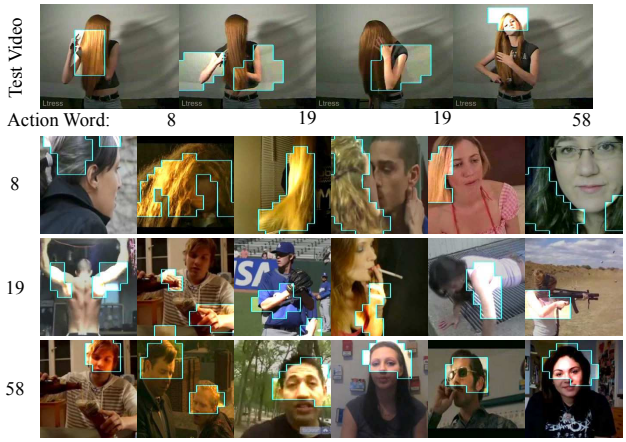


Figure 8: Top contributing action words for classifying this video. The softmax score for this 'brushing hair' video is obtained (in appearance stream) from features corresponding to 'action words' that seem to focus on faces, eyes, hair and hand/leg clusters. It was incorrectly classified as 'clap' by the baseline two-stream architecture but is correctly classified as 'brushing hair' by our method.

of conv5_3 neuron assigned to that word. Fig. 6 shows some such action words and our interpretation for those words. We visualize these assignment maps over videos in Fig. 7.

To verify how the 'action words' help in classification, we consider an example video in Fig. 8 that was originally misclassified by the two-stream model but gets a high softmax score for the correct class with our ActionVLAD model. For ease of visualization, we only consider the RGB model in this example. We first compute the ActionVLAD feature for this video, extract the linear classifier weights for the correct class, and accumulate the contribution of each 'action word' in the final score. We show a visualization for some of the distinct top contributing words. The visualization for each word follows same format as in Fig. 6. We see that this 'brushing hair' video is represented by spatial 'action words' focussing on faces, eyes, hair, and hands.

5. Conclusion

We have developed a successful approach for spatiotemporal video feature aggregation for action classification. Our method is end-to-end trainable and outperforms most prior approaches based on the two-stream VGG-16 architecture on the HMDB51 and UCF101 datasets. Our approach is general and may be applied to future video architectures as an ActionVLAD CNN layer, which may prove helpful for related tasks such as (spatio) temporal localization of human actions in long videos.

Acknowledgements: The authors would like to thank Gül Varol and Gunnar Atli Sigurdsson for help with iDT. DR is supported by NSF Grant 1618903, Google, and the Intel Science and Technology Center for Visual Cloud Systems (ISTC-VCS).

References

[1] Project webpage (code/models). http://rohitgirdhar.github.io/ActionVLAD. 5

[2] Supplementary material (appendix) for the paper. https://arxiv.org/abs/1704.02895. 6, 7

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 5

[4] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *CoRR*, abs/1609.08675, 2016. 2

[5] M. R. Amer, S. Todorovic, A. Fern, and S. Zhu. Monte carlo tree search for scheduling activity recognition. In *ICCV*, 2013. 2

[6] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016. 1, 2, 3, 5, 6

[7] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013. 3

[8] G. Awad, J. Fiscus, M. Michel, D. Joy, W. Kraaij, A. F. Smeaton, G. Qunot, M. Eskevich, R. Aly, G. J. F. Jones, R. Ordelman, B. Huet, and M. Larson. Trecvid 2016: Evaluating video search, video event detection, localization, and hyperlinking. In *TRECVID*, 2016. 2

[9] Z. Cai, L. Wang, X. Peng, and Y. Qiao. Multi-view super vector for action recognition. In *CVPR*, 2014. 7

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2

[11] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006. 2

[12] C. R. de Souza, A. Gaidon, E. Vig, and A. M. López. Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition. In *ECCV*, 2016. 1, 7

[13] J. Donahue, L. A. Hendricks, S. Guadarrama, S. V. M. Rohrbach, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 7

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

[15] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 1, 2, 6, 7

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

[17] A. Gaidon, Z. Harchaoui, , and C. Schmid. Temporal localization of actions with actoms. In *PAMI*, 2013. 2

[18] A. Gorban, H. Idrees, Y.-G. Jiang, A. R. Zamir, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes, 2015. 2

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016. 1, 7

[20] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 1

[21] M. S. Ibrahim, S. Muralidharan, Z. Deng, A. Vahdat, and G. Mori. A hierarchical deep temporal model for group activity recognition. In *CVPR*, 2016. 2

[22] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015. 7

[23] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 2, 3

[24] Y. Jiang, J. Liu, A. Roshan Zamir, I. Laptev, M. Piccardi, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://www.thumos.info/, 2013. 6

[25] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, 2016. 2

[26] 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. 1, 2, 7

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[28] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 1, 2, 5

[29] G. Lev, G. Sadeh, B. Klein, and L. Wolf. Rnn fisher vectors for action recognition and image annotation. In *ECCV*, 2016. 2, 7

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

[31] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2, 7

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

[33] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CVIU*, 2016. 7

[34] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014. 2

[35] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 3

[36] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *CVPR*, 2014. 2

[37] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *CVPR*, 2013. 2

[38] M. Rohrbach, M. Regneri, M. Andriluka, S. Amin, M. Pinkal, and B. Schiele. Script data for attribute-based recognition of composite activities. In *ECCV*, 2012. 2

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 1

[40] M. S. Ryoo and J. K. Aggarwal. Recognition of composite human activities through context-free grammar based representation. In *CVPR*, 2006. 2

[41] 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. 1, 2, 6, 7

[42] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2, 4, 5, 7

[43] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 4

[44] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR-12-01*, 2012. 1, 2, 5

[45] L. Sun, K. Jia, D.-Y. Yeung, and B. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015. 7

[46] V. Sydorov, M. Sakurada, and C. Lampert. Deep fisher kernels and end to end learning of the Fisher kernel GMM parameters. In *CVPR*, 2014. 3

[47] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. 2016. 1

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

[49] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2, 7

[50] L. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-sne. *JMLR*, 2008. 6

[51] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *CoRR*, abs/1604.04494, 2016. 1, 2, 7

[52] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action Recognition by Dense Trajectories. In *CVPR*, 2011. 2

[53] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1, 2, 7

[54] 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

[55] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 2, 3, 7

[56] L. Wang, Y. Qiao, and X. Tang. Mofap: A multi-level representation for action recognition. *IJCV*, 2016. 7

[57] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *CoRR*, abs/1507.02159, 2015. 7

[58] 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, 4, 5, 7

[59] X. Wang, A. Farhadi, and A. Gupta. Actions ˜ transformations. In *CVPR*, 2016. 1, 2, 7

[60] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, 2015. 2

[61] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, 2015. 2

[62] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *arXiv preprint arXiv:1507.05738*, 2015. 2

[63] W. Zhu, J. Hu, G. Sun, J. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *CVPR*, 2016. 1, 7