

Few-Shot Learning of Video Action Recognition Only Based on Video Contents

Yang Bo

Yangdi Lu

Wenbo He

McMaster University

boy2@mcmaster.ca

Abstract

The success of video action recognition based on Deep Neural Networks (DNNs) is highly dependent on a large number of manually labeled videos. In this paper, we introduce a supervised learning approach to recognize video actions with very few training videos. Specifically, we propose Temporal Attention Vectors (TAVs) which adapt various length videos to preserve the temporal information of the entire video. We evaluate the TAVs on UCF101 and HMDB51. Without training any deep 3D or 2D frame feature extractors on video datasets (only pre-trained on ImageNet), the TAVs only introduce 2.1M parameters but outperforms the state-of-the-art video action recognition benchmarks with very few labeled training videos (e.g. 92% on UCF101 and 59% on HMDB51, with 10 and 8 training videos per class, respectively). Furthermore, our approach can still achieve competitive results on full datasets (97.1% on UCF101 and 77% on HMDB51).

1. Introduction

The use of Deep Neural Networks (DNNs) in the field of computer vision has expanded significantly in recent years. For video action recognition, several frameworks [5, 6, 35] have shown outstanding performance. The success of these approaches largely sustained by the manual annotation of the large-scale datasets. However, it is still challenging to recognize human actions with very few manually labeled training videos. There are several attempts to deal with this problem. Srivastava *et al.* [30] trained a Long Short-term Memory (LSTM) with the fixed number of unlabeled video frames to predict future frames of that video then fine-tune it for the supervised video action recognition. However, it still needs a large number of videos to train which brings huge computational overhead. Zhu *et al.* [45], Mettes *et al.* [24] and Jain *et al.* [12] attempted to recognize actions without any observed data or with only few labeled data (Zero/Few-Shot Learning). These approaches classify the actions by measuring the similarity (e.g. Euclidean distance) between the visual representations and the semantic

representations in the embedding space. However, these approaches require additional linguistic contexts or visual representations of objects and the accuracy is heavily dependent on the precise representation of the additional information.

In this paper, we aim to adopt supervised learning to recognize human actions with very few manually annotated training videos. Recent works such as [7, 36, 41] pointed out that long term dynamics and temporal patterns are very important cues for the recognition of actions. The key challenge is to generate a video descriptor that precisely captures the important video-wide temporal correlation among frame features with a small number of training parameters. Some current Convolutional Neural Networks (CNNs) based researches are either using sub-sampling [41] or longer video clips [36] to capture the video-wide temporal dynamics. However, these works only preserve partial temporal information of the video. Some other approaches adapt ranking functions [7] or Recurrent Neural Networks (RNNs) [1, 4] to preserve the temporal information. However, they introduce too many training parameters that cannot be trained well in the circumstance of only having very few training videos. Another work [9] generates the video descriptor by clustering the pixels in all frame features maps both spatially and temporally with k-means clustering algorithm. However, this approach cannot preserve the temporal information of videos.

In this paper, we propose Temporal Attention Vectors (TAVs) which adapt to various lengths of videos to encode the correlation among frame features. The frame features are either manually defined or generated by an ImageNet pre-trained CNN. The initial value of TAVs are manually defined and each TAV highlights a certain period of the video by giving them higher temporal weights than others. After that, the TAVs are aggregated by their importance scores which are learned by a shallow CNN. There are three keys advantages to use TAVs. First, the TAVs capture the video-wide dynamics of the video. Unlike the current end-to-end trainable frameworks, the TAVs encode the temporal correlation between all frames, which is important especially with only very few training videos. Second, the TAVs

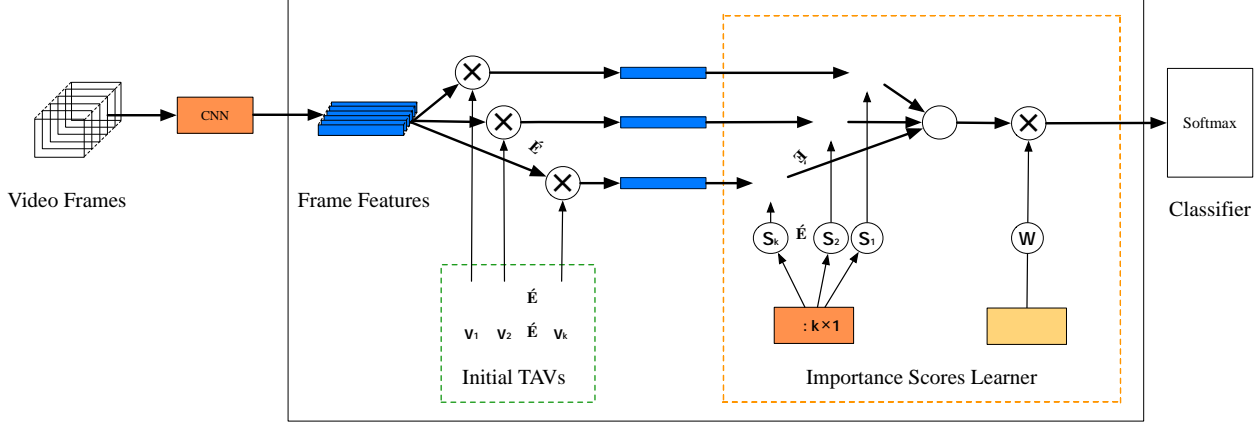


Figure 1: The working procedure of TAVs. The CNN which is used as the frame feature extractor pre-trained on ImageNet and the weights are frozen during the task, *e.g.* not trained on the video dataset. “ \times ” denotes matrix multiplication, “ $+$ ” denotes the element-wise multiplication and “ $+$ ” denotes the addition. v_k denotes the initialized TAVs, S_k denotes the importance score for the k th TAV and W denotes the linear embedding weights matrix. $\text{conv}_{k \times 1}$ denotes $k \times 1$ convolution and emb denotes the linear embedding operation. Here, we show the base importance learner, the inflate-shrink learner can be done by adding one convolution layer with $32 \ 1 \times 1$ filters. We use a fully connected layer with softmax function as the classifier.

can model complex temporal patterns. The static temporal weights cannot correctly express the temporal patterns of actions. For example, actions “run” and “long jump” have different patterns, we only need to focus on “run” action for the prior one but need focus on both “run” and “jump” for the latter one. By studying the importance scores, the TAVs can simulate different temporal patterns. Third, the TAVs only introduce few training parameters. With insufficient training samples, a deep neural network (many training parameters) is hard to learn the true temporal patterns of actions. Only few training parameters (importance scores) makes TAVs could capture more accurate temporal pattern than the deep network with very few training videos.

2. Related Works

Capturing the spatiotemporal information of videos for action recognition has been a well-studied research domain. Historically, researchers have mostly focused on the handcrafted spatiotemporal features of Space-Time Interest Points (STIP) [20]. Most successful examples are 3D Histogram of Gradient (HOG3D) [17], Histogram of Optical Flow (HOF) [21], and Motion Boundary Histogram (MBH) [3]. Also, the trajectory-based approaches [14, 23, 32, 38, 39] have shown a significant improvement in action recognition.

More recently, the CNN-based end-to-end fashions have been widely applied to the video action recognition area and shown the outstanding performance. These approaches can be roughly separated into two categories. The first category, which extends the CNNs to a third, temporal dimension by

replacing the 2D filters with 3D ones [2, 13, 34, 35, 37] to capture the spatiotemporal information from fixed length video clips. The second category initially processes color and optical flow information in parallel for subsequent late fusion of their separate classification scores [28]. Several improvements were proposed based on this work. For example, Wang *et al.* [40] extracted deep features and conducted trajectory constrained pooling to aggregate convolutional features as video representations. Feichtenhofer *et al.* [6] tried different two-stream fusion approaches to fuse the two streams. Carreira and Zisserman [2] recently introduced a model (I3D) that combines two-stream processing with 3D convolutions.

An alternative solution models the temporal structure of video by various pooling approaches [8, 15, 43], rank functions [7], k-means clustering [9] and different distributions [25, 26, 27]. Recurrent Neural Networks have also been used to encode temporal information for learning video representations [4, 30, 31, 33, 42]. Donahue *et al.* [4] used the LSTM together with CNN to either output an action label or a video description. Srivastava *et al.* [30] proposed to learn video descriptions with the encoder-decoder LSTM in an unsupervised manner.

Our work is similar to the temporal structure filters in [26]. They train Cauchy distributions to detect multi-actions in videos. The centers and width of Cauchy distributions are initialized by some trainable and uniformly selected “seed” between -1 to 1 then scaled according to the length of videos and exponential function, respectively. On the contrary, our approach initializes both center and width according to the length of video and studies the importance scores instead of learning the distribution parameters themselves.

We compare temporal structure filters with TAVs in Section 5.

3. Temporal Attention Vectors

To recognize actions with very few training samples, the video representation should preserve video-wide temporal information meanwhile the generation procedure of the representation should involve only small number of training parameters.

We now describe the TAVs which are shown in Figure 1. We denote a video as $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathbb{R}^{W \times H \times C}$ and $t = [1 : T]$. We use W and H to represent the width and height of the frame respectively. Each frame either represents an RGB image ($C = 3$) or a horizontal / vertical optical flow image ($C = 1$). The length T usually varies for different videos. We use $(\mathbf{x}_t) \in \mathbb{R}^D$ to represent the feature of frame \mathbf{x}_t , where \cdot represents the operations of a CNN (the frame feature extractor). The video representation is generated with two steps: i). The frame features $(\mathbf{x}_1), (\mathbf{x}_2), \dots, (\mathbf{x}_T)$ are aggregated with the elements of K TAVs $\{v_1, v_2, \dots, v_K\}$, where $v_k \in \mathbb{R}^T$ and initial values are manually defined. ii). All aggregated features are added with the importance scores s for each TAV then linear embedded with weight W to form the video representation. s and W are learned by a convolution layer and a fully connected layer, respectively. More formally, we rewrite the above operations as follows:

$$\mathbf{D} = \mathbf{W} \sum_{k=1}^K \sum_{t=1}^T s_k (\mathbf{x}_t) v_{t,k}, \quad (1)$$

where $v_{t,k}$ is the t th element in the k th TAV v_k , s_k is the importance score for the k th TAV and \cdot denotes the matrix multiplication.

The TAVs is able to study the temporal pattern with very few training videos because of the follows. i). It encodes video-wide temporal information by using multiple TAVs which adapt to the various lengths of frame sequences and not introduce any training parameters. ii). It is able to select the important temporal information by studying the importance scores. In the following of this section, we provide the details of how to initialize the TAVs and the architecture of the shallow CNN that is used to study the importance scores of the TAVs.

3.1. Initialization of the TAVs

The static pooling approaches (e.g. average pooling) fail to preserve the sequential information of the frame features. We propose to encode the video-wide temporal information by calculating the weighted sum of all frame features. The weights are the elements of the TAVs which adapt to various length of videos. We introduce several initialization approaches of the TAVs below and leave the experiments in

appendix.

Random. One simplest way to initialize the TAVs is uniformly choosing them from the interval $(0, 1)$ (Figure 2a). More formally, let L denotes a number larger or equal to the frames number of the longest video in a dataset, we can choose the temporal attention weights as follows:

$$\begin{aligned} a_k &= [a_{1,k}, a_{2,k}, \dots, a_{L,k}], \quad k = [1 : K], \\ a_{l,k} &\sim U(0, 1), \quad l = [1 : L], \\ v_k &= [v_{1,k}, v_{2,k}, \dots, v_{T,k}], \\ v_{t,k} &= \frac{e^{a_{t,k}}}{\sum_{t=1}^T e^{a_{t,k}}}. \end{aligned} \quad (2)$$

We normalize the weights in v_k such that let they sum up to 1 to eliminate the impact brought by the various length of videos. For example, as the video length increases, the weights for each frame feature become smaller. Therefore, the same action represented with the various number of frames will not be misclassified.

Single Switching Distinguishable (SSD). An alternative way is using the combination of a constant sequence and the strictly monotone sequences as the initial values of the TAVs, which are shown in Figure 2b, to distinguish a single pair of frames switching (e.g. $\mathbf{X}_1 = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ and $\mathbf{X}_2 = (\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_2)$). The elements in the constant sequence are all c and the elements in strictly monotone sequences are randomly chosen from $U(0, 1)$ then sort with ascending or descending orders. Same as the random initialization, we normalize the weights in v_k . The approach can be formally represented as follows:

$$\begin{aligned} v_1 &= [c, c, \dots, c], \\ a_k &= [a_{1,k}, a_{2,k}, \dots, a_{L,k}], \quad k = [2 : K], \\ a_{l,k} &\sim U(0, 1), \quad l = [1 : L], \\ &\text{sort}(a_k), \\ v_k &= [v_{1,k}, v_{2,k}, \dots, v_{T,k}], \\ v_{t,k} &= \frac{e^{a_{t,k}}}{\sum_{t=1}^T e^{a_{t,k}}}. \end{aligned} \quad (3)$$

By using these TAVs, it is obvious that the switching order of single pair of frames could be detected by the TAVs based on a reasonable assumption that the frame features of two different frames are different.

Dynamic Gaussian (DG). This TAVs initialization approach is based on the following observations: i). The “key” frames (the most important frames for classification task) that are used to represent an action are consecutive. ii). These key frames form only a clip rather than the entire video. iii). The number of key frames that are needed to represent different actions varies. Based on the above observations, we propose a way to initialize the TAVs from the Probability Density Function (PDF) of Gaussian Distributions. The results are shown in Figure 2c. We evenly

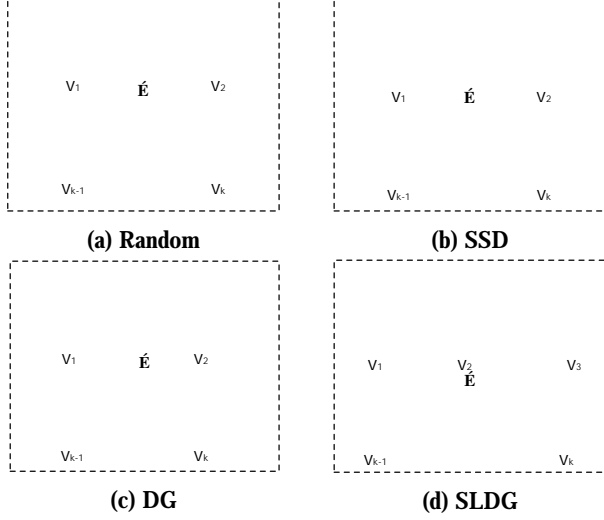


Figure 2: The illustration of TAVs which are initialized by different approaches.

divide a video into several clips and use the TAVs highlight each of them. The TAVs give the higher temporal weights for the frame features in the clip and lower weights for the frame features out of that clip. The mean and standard deviation are dynamically selected based on the length of the video clips. Specifically, we evenly partition the frame number sequence $(1, 2, \dots, T)$ of the video X into K chunks $\{C_1, C_2, \dots, C_K\}$. For each chunk, we choose the element in the middle position as the mean μ_k and use the length of the chunk $\text{len}(C_k)$ divide by a factor as the standard deviation σ_k . More formally, these operations can be formulated as:

$$\begin{aligned} v_k &= [v_{1,k}, v_{2,k}, \dots, v_{T,k}], k = [1 : K], \\ v_{t,k} &= \text{PDF}(t|\mu_k, \sigma_k^2), t = [1 : T], \\ \mu_k &= C_k(\frac{\text{len}(C_k)}{2}), \sigma_k = \frac{\text{len}(C_k)}{2}. \end{aligned} \quad (4)$$

The DG initialized TAVs highlight different clips cross the entire video, but each of them encodes the video-wide temporal information.

Short-Long Dynamic Gaussian (SLDG). This initialization approach is inspired by the DG approach. The original DG can only highlight certain range of temporal information of a video. However, we are able to highlight shorter or longer ranges of temporal information by decreasing or increasing the standard deviation value in the DG generation approach, respectively (Fig 2d). More formally, we define a set of chunk numbers $m = \{m_1, m_2, \dots, m_N\}$. Then we repeat the DG vector generation approach N times and use m_n as the chunk number in the n th iteration. Thus, the total number of temporal vectors is $m_1 + m_2 + \dots + m_N$ and the value of standard deviation is dynamically defined by m_n .

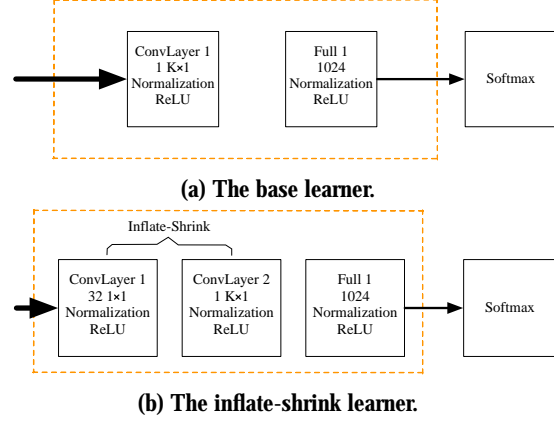


Figure 3: The architecture of the importance score learner.

3.2. Importance Score Learner

The importance score learner is used to generate video descriptor which contains important temporal information based on the aggregated frame features. We propose two types of learner. i). Base learner: scores of TAVs are directly learned by a single convolution layer (Figure 3a). ii). Inflate-shrink learner: The dimensionality of aggregated frame features is first expanded using 1×1 convolution then compressed into a single one as shown in Figure 3b.

The base learner has a single convolutional layer with one $K \times 1$ filter and the inflate-shrink learner has 2 convolutional layers with 32 and 1 filter response maps with 1×1 and $K \times 1$ filters for the first and second convolutional layers, respectively. All convolutional layers are followed by a batch normalization layer and a rectified linear unit (ReLU). Filter stride for all dimensions is 1 for convolution operations. Then we apply 1 fully connected layers of sizes 1024 for both learners. We use ReLU after the fully connected layers.

4. Implementation Details

The architecture of entire framework is shown in Figure 1. We adopt all convolution blocks and the global pooling layer of the ImageNet [18] pre-trained ResNet-152 [11] and I3D [2] as the backbone networks (frame feature extractors). The weights for the extractors are shared and frozen for all experiments. The final video descriptor is predicted using a fully connected layer with softmax function.

4.1. Input Configuration

We use the same input configurations for both training and testing in our experiments. The frame extractor generates the features for both RGB frames and the optical flow images. The optical flow images [44] are pre-computed and stored as JPEG images (with displacement vectors > 20 pixels clipped). We follow the data argumentation from

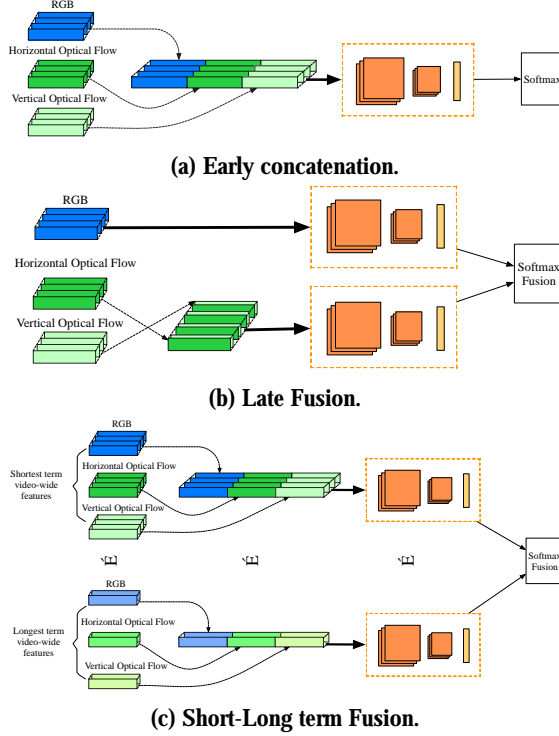


Figure 4: Three spatial and temporal fusion approaches.

[41]. We randomly crop from the four corners and the center of input images and sample the width and height of each crop randomly as $W, H \in \{256, 224, 192, 168\}$, followed by re-sizing to 224×224 . The argumentation is applied for both original and horizontal flipped images.

4.2. Two Streams Fusion

Since our framework adopts the two-stream architecture which takes RGB and optical flow fields as inputs. We consider three ways to fuse the spatial and temporal streams. The detailed comparison is shown in appendix.

Early concatenation. As shown in Figure 4a, the importance scores of both spatial and temporal TAVs are studied by single learner. In this case, the input of the learner $F \in \mathbb{R}^{K \times 3D}$.

Late fusion. In contrast with the early concatenation approach, we study the importance scores of spatial and temporal TAVs separately by two learners. The input for the temporal learner is $F \in \mathbb{R}^{K \times D \times 2}$. The class scores of the spatial and temporal streams are combined by late fusion as the final class scores as illustrated in Figure 4b.

Short-Long term fusion. This fusion approach is designed for the SLDG initialized TAVs. As shown in Figure 4c, this approach learn the importance score of TAVs for different temporal periods separately by adopting multiple learners. The spatial and temporal TAVs share same importance score (same as the early concatenation). The class scores of the different terms are then late fused as the

final class score.

4.3. Training and Testing

The training procedures are depended on different fusion approaches. For the early concatenation and short-long term fusion, we use the batch size of 64 which are randomly selected (uniform across all video samples). The initial learning rate is set to 10^{-5} and reduce by a factor of 10 after the validation error saturates. The training stopped when the learning rate reaches 10^{-8} . For the late fusion, we first separately train both streams then train them together as in [28]. The learning rate starts at 10^{-4} and is reduced by a factor of 10 two times after the validation error saturates. When train two streams together, we set the learning rate as 10^{-5} and reduce by a factor of 10 after the validation error increases. We stop the training when learning rate reaches 10^{-8} . The batch size is 128 for single stream training and 64 for two streams together. The kernel and bias weights are all initialized by Xavier initialization [10]. The learner weights are learned using the Adam algorithm [16]. During the testing, the class scores for the whole video are obtained by averaging the scores across the inputs.

5. Experiments and Results

In this section, we first introduce the evaluation datasets then we provide detailed analysis of the effectiveness of different TAVs initialization, fusion approaches and the inflate-shrink structure. Finally, we compare the performance of our method to the state-of-the-art on the datasets with change in the size of training videos and the full datasets.

5.1. Datasets

We evaluate our approach on two popular action recognition datasets. First, UCF101 [29], which consists of 13320 action videos in 101 categories. The second dataset is HMDB51 [19], which contains 6766 videos that have been annotated for 51 actions. For both datasets, we use two different set of training/testing splits: i). the official splits, ii). the smaller splits. The official splits is provided by the datasets. For the smaller splits, we uniformly choose various number of videos from the official training splits for each action and the testing splits are the same as the official one.

There are two main reasons that we choose UCF101 and HMDB51: **a.** our object is action recognition under very few training data scenario. If there are enough data (e.g. Kinetics [2]), the 3D-CNNs (e.g. I3D, I2+1D networks) are the better choices, **b.** we only find [30] have done the similar task. For a fair comparison, we choose the same datasets as them.

5.2. Evaluation of the Effectiveness of TAVs and Importance Score Learners

In this section, we investigate the effectiveness of TAVs and importance score learners (Section 3), the average accuracies are reported in Table 1. We choose the DG initialized TAVs ($K = 4$) to encode the frame features and evaluate the performances. The value of α is set to 2 since we find it results the best performance. In the following experiments, we use $\alpha = 2$ as default for DG vectors. We follow the training and testing procedures as described in Section 4. In Table 1, we can see applying DG initialized TAVs

Aggregator	Backbone	ResNet-152 [11]	I3D [2]
Average pooling		64.4	86.1
DG		68.1	89.1
DG + Base learner		70	89.5
DG + Inflate-Shrink learner		70.1	90.6

Table 1: Average accuracy (%) on the UCF101 split 1 (RGB) with different importance score learners and backbones. The experiments are repeated 10 times and each time 10 training videos are uniformly chosen for each class.

gives 3.7% and 3% accuracy improvements for ResNet-152 and I3D backbones compare to the average pooling, respectively. The accuracy is further increased by applying the score learner. For example, the performance improves from 68.1% to 70% when use the base learner for DG+ResNet152, and further increases 0.1% when apply the inflate-shrink learner. When use I3D as the backbone, the inflate-shrink learner yields 1.5% improvements compare to only apply the pure DG initialized TAVs. In the following experiments, we use the inflate-shrink learner as the default choice.

We also evaluate the effectiveness of hyper-parameters (TAVs number, initialization approaches, fusion approaches etc.) with few training parameters. Please see appendix for detail.

5.3. Comparison with state-of-the-art with Very Few Training Videos

Methods	# Parameter
St Multiplier (two-stream) [6]	85.8M
LSTM [30]	83.8M
TAVs (ours)	2.1M

Table 2: Parameter number of Spatiotemporal Multiplier Networks, unsupervised LSTM and TAVs

In this section, we compare our model to other frameworks on UCF101 and HMDB51 with change in the size of

the labeled training set. We use the same training sets for all models. During the comparison of the TAVs initialization approaches, we find the SLDG initialization with $K = 3$ (# of TAVs) and $m = (2, 1)$ (# of chunks for each iteration) gives best results. We believe this is because the SLDG catch both short and long temporal information of video. In all following experiments, we use SLDG with this setting as default choice.

The first model is the Spatiotemporal Multiplier network [6]. The comparisons are shown in Figure 5a and Figure 5b. Our model outperforms the Spatiotemporal Multiplier network when only giving few labeled training videos. For example, with only 10 labeled training video per class, our model achieves 78.5% on UCF101 and 45.8% on HMDB51 which are 7.5% and 10.3% higher than the accuracy of Spatiotemporal Multiplier network, respectively. Overall, our model outperforms the Spatiotemporal Multiplier network when the number of training videos per class is less than 60 and 40 on UCF101 and HMDB51, respectively. We also try to use ResNet-152 for both spatial and temporal stream but the accuracy is much lower than original model with very few training samples. We believe the reason is the 152 layers ResNet contains much more parameters than 50 layers one which is not suitable for the few shot task. The second framework is the unsupervised LSTM [30] which is pre-trained on a 300 hours YouTube data then transformed to the supervised learning. For fair comparisons, our framework also only use the RGB frames as inputs. The results are shown in Figure 5c and Figure 5d. We notice that our model slightly underperforms the LSTM when the size of the training set is extremely small (1 or 2 videos per class). We believe that is due to the LSTM is pre-trained on a large number of videos and this is confirmed when we using Kinetics pre-trained I3D as frame feature extractor. As the size of the labeled dataset grows, the gap becomes smaller. When the number of training video per class is 4, our model has accuracy 58.2% on UCF101 and 29.5% on HMDB51 which are higher than the unsupervised LSTM on both datasets. We calculate the training parameters that are introduced by the importance score learner, since the only training procedure is applied on the learner. As shown in Table 2, the TAVs only introduce 2.1M parameters, which is 97.5% and 97.4% smaller compare to the Spatiotemporal Multiplier Network and unsupervised LSTM, respectively.

Instead of using 2D network as the backbone, we also evaluate the TAVs with 3D frame feature extractor. We use I3D [2] as the backbone and compare the performance of TAVs with base I3D and Temporal Structure Filter (TSF) [26] with few training videos. The results are shown in Table 3. We use the authors provided codes for base I3D and implement the TSF according to the paper. For the experiments, all layers before the 3D average pooling layer are frozen. We repeat the experiments 3 times and report

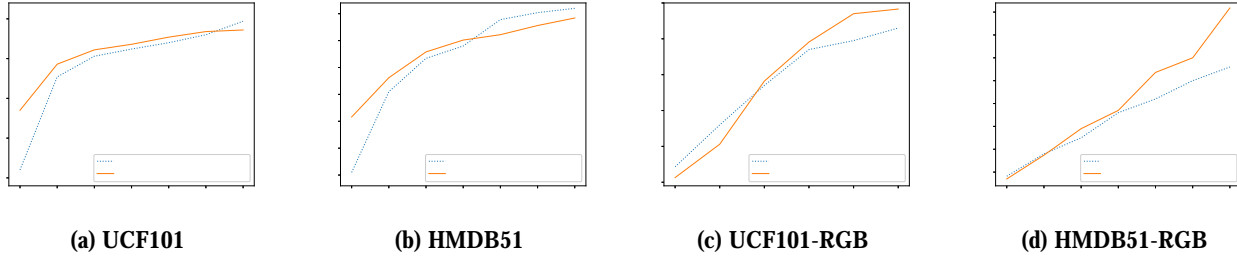


Figure 5: Comparisons with the Spatiotemporal Multiplier Network ((a) and (b)) and fine-tuned unsupervised LSTM ((c) and (d)) for action recognition with change in the size of the labeled training set on UCF101 and HMDB51 split 1. The training videos are uniformly chosen for each action class. For the comparison with St Multiplier Network, the experiments are repeated 5 times and the average accuracy (%) are reported for both TAVs and St Multiplier Network. For comparison with the fine-tuned unsupervised LSTM, only RGB frames are used and the average accuracy of 10 times experiments are reported.

# Training Videos	I3D [2]		I3D+TSF [26]		I3D+SLDG(2,1)	
	Round Acc	Average Acc	Round Acc	Average Acc	Round Acc	Average Acc
UCF101						
1	20.3 / 24.4 / 25.6	23.4	45.8 / 52.6 / 46.7	48.4	65.8 / 68.2 / 71	68.3
2	39.4 / 39.8 / 39.6	39.6	61.7 / 63.2 / 61.5	62.1	82.4 / 83.4 / 83.3	83
4	63.3 / 63.1 / 62.9	63.1	74.1 / 69.2 / 65.2	70	88.4 / 88.4 / 88.1	88.3
10	74.4 / 74.9 / 74.4	74.6	84.7 / 85.8 / 83	84.5	91.8 / 92.5 / 91.7	92
20	83.3 / 83 / 83.4	83.2	88.6 / 85.5 / 86.4	86.3	93.3 / 93.2 / 93.4	93.3
50	90.5 / 90.2 / 90.4	90.4	90.7 / 90.4 / 87.8	89.6	94.1 / 93.8 / 94	94
HMDB51						
1	13.1 / 13.8 / 14.2	13.7	28.3 / 25.4 / 27.1	27	34.7 / 35.8 / 37.4	36
2	18.6 / 18.2 / 18.3	18.4	36.5 / 34.1 / 34.8	35.1	44.4 / 42.5 / 43.3	43.4
4	30 / 31.2 / 29.7	30.3	46.1 / 44.5 / 42.6	44.4	53.3 / 56.3 / 49.8	53.1
8	42.9 / 44 / 43.3	43.4	49.5 / 52 / 49.5	50.3	56.5 / 61.1 / 59.6	59
16	53.2 / 54.1 / 53.9	53.7	52.9 / 50.2 / 52.9	52	62.1 / 63.9 / 63.5	63.2
32	61.1 / 61.8 / 61	61.3	56 / 55.2 / 56.2	55.8	65.2 / 66.3 / 65	65.5
64	68.6 / 68.8 / 68.2	68.5	56.9 / 58.2 / 59	58	67.4 / 67.6 / 67.1	67.4

Table 3: Comparisons with I3D (left column) and Temporal Structure Filter (TSF) (middle column) for action recognition with different number of training samples on UCF101 and HMDB51 split 1 (RGB). The experiments are repeated 3 times and each time the training videos are uniformly chosen for each class. We report both round accuracies (left part of each column) and average accuracy (right part of each column).

both round accuracies and average accuracy. In Table 3, we can see that as the number of training videos increases, the accuracies for all three frameworks also increase. The TSF outperforms the base I3D when the number of training videos < 50 and < 8 on UCF101 and HMDB51, respectively. Our approach outperforms the base I3D and TSF with all small training sets except when the number of training videos = 64 on HMDB51. We try to fully fine-tune the I3D on both datasets. However, the results are much lower than only train the fully connected layer with few training videos. Furthermore, we also implement the TSF according to the code provided by the authors (initialize the width of Cauchy distribution according to the length of videos which is different to the paper) and evaluate on HMDB51 with 8

training videos per class. It achieves 53.8 over three rounds but still 5.2% less than ours. We also compare our approach with I3D on Diving48 dataset [22], see details in the appendix.

6. Comparison with the state-of-the-art on Full Datasets

Finally, we compare our model to the state-of-the-art action recognition results on full UCF101 and HMDB51 datasets. The performance is summarized in Table 4. The table is divided into three sets. The first set compares models that use only RGB data. The second set compares models that use optical flow features only. Models in the third

Methods	Backbone	Pretrain	Fully Fine-tuning	UCF101	HMDB51
Two-stream (spatial) [28]	Two-stream	ImageNet	Yes	73	40.5
Two-stream (spatial) [6]	ResNet-152	ImageNet	Yes	83.4	46.7
Unsupervised LSTM (spatial) [30]	LSTM	300 hrs vids of Sports-1M	No	75.8	44.0
I3D [2]	I3D	ImageNet+Kinetics	Yes	95.6	74.8
TSF [26]	I3D	ImageNet+Kinetics	No	91.1	60
Ours	ResNet-152	ImageNet	No	83.2	58.9
Ours	I3D	ImageNet+Kinetics	No	95	69.8
Two-stream (temporal) [28]	Two-stream	-	Yes	83	54.6
Two-stream (temporal) [6]	ResNet-152	-	Yes	87.2	60
Unsupervised LSTM (temporal) [30]	LSTM	300 hrs vids of Sports-1M	No	77.7	-
I3D [2]	I3D	ImageNet+Kinetics	Yes	96.7	77.1
TSF [26]	I3D	ImageNet+Kinetics	No	92.3	62.8
Ours	ResNet-152	ImageNet	No	78.5	51.3
Ours	I3D	ImageNet+Kinetics	No	96	73.4
Two-stream [28]	Two-stream	ImageNet (spatial)	Yes	88	59.4
Two-stream [6]	ResNet-152	ImageNet (spatial)	Yes	91.8	63.8
St Multiplier [6]	ResNet-50,152	ImageNet	Yes	94.2	68.9
Unsupervised LSTM [30]	LSTM	ImageNet	No	84.3	-
ActionVLAD [9]	VGG16	ImageNet	Yes	92.7	66.9
I3D [2]	I3D	ImageNet+Kinetics	Yes	98	80.7
TSF [26]	I3D	ImageNet+Kinetics	No	93.3	63.8
Ours	ResNet-152	ImageNet	No	89.8	64.2
Ours	I3D	ImageNet+Kinetics	No	97.1	77

Table 4: Comparison with state-of-the-art action recognition models on full UCF101 and HMDB51. The fully fine-tuning: Yes indicates the backbone is end-to-end fine-tuned on UCF101 and HMDB51, No indicates the backbone is frozen during the experiments.

set use both.

On RGB data, our ResNet-152 based model performs 10.2% and 18.4% better than the original two-stream model on UCF101 and HMDB51, respectively. The ResNet-152 based two-stream framework performs slightly better than our model on UCF101, but ours do 12.2% better than theirs on HMDB51 even our frame feature extractor is not trained on the UCF101 and HMDB51 datasets. When switch the backbone to I3D, our model achieve 95% and 69.8% on UCF101 and HMDB51 without training the frame feature extractor, which are 4.9% and 9.8% higher and only 0.6% and 5% lower than TSF and base I3D, respectively.

When use ResNet-152 as the frame feature extractor, the performance of our model on optical flow data is just passable. We believe this is due to the feature extractor is only trained on ImageNet. As the distribution of optical flow is different from the RGB images, the extracted features can not correctly represent the optical flow images. This is proved when we use I3D as backbone (pre-training with optical flow features of Kinetics). Without training the I3D on UCF101 and HMDB51, our model achieve 96% on UCF101 and 73.4% on HMDB51.

When we combine predictions from the RGB and flow models, we obtain 89.8% and 64.2% on UCF101 and HMDB51 with ResNet-152, respectively. Our results are 2% lower and 0.4% better than the ResNet-152 based two-

stream network on UCF101 and HMDB51, respectively. The performance of our model is 4.5% and 4.7% lower compared to the St Multiplier network. However, it is interesting to note that our model outperforms TSF on HMDB51 even we use ResNet-152 instead of I3D as the backbone. When use I3D as frame feature extractor, the performance of our model is really closed to the base I3D which are only 0.9 and 2.7 lower than base I3D on UCF101 and HMDB51, respectively.

Our approach also outperform Temporal Segment Networks (TSN) [41] and partially fine-tuned I3D with only use RGB images on Diving48 dataset, please see details in the appendix.

7. Conclusion

In this paper, we propose TAVs to recognize human actions with very few labeled training videos. We analyze the performances of different initialized TAVs with optimized parameters on with very few training videos. The best performance is achieved by using the SLDG TAVs (details are in appendix). We show that the framework which adopts TAVs can learn discriminative video representation with very few labeled training samples. The performance is boost when apply stronger backbones (*e.g.* I3D). We believe our approach can be applied to other models to capture the temporal information of video in other tasks.

References

- [1] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.
- [2] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4724–4733. IEEE, 2017.
- [3] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *European conference on computer vision*, pages 428–441. Springer, 2006.
- [4] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.
- [5] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in neural information processing systems*, pages 3468–3476, 2016.
- [6] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7445–7454. IEEE, 2017.
- [7] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5378–5387, 2015.
- [8] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. In *Advances in Neural Information Processing Systems*, pages 34–45, 2017.
- [9] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, volume 2, page 3, 2017.
- [10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [12] M. Jain, J. C. van Gemert, T. Mensink, and C. G. Snoek. Objects2action: Classifying and localizing actions without any video example. In *Proceedings of the IEEE international conference on computer vision*, pages 4588–4596, 2015.
- [13] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.
- [14] Y.-G. Jiang, Q. Dai, X. Xue, W. Liu, and C.-W. Ngo. Trajectory-based modeling of human actions with motion reference points. In *European Conference on Computer Vision*, pages 425–438. Springer, 2012.
- [15] A. Kar, N. Rai, K. Sikka, and G. Sharma. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3376–3385, 2017.
- [16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [17] A. Klaser, M. Marszalek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, pages 275–1. British Machine Vision Association, 2008.
- [18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [19] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2556–2563. IEEE, 2011.
- [20] I. Laptev and T. Lindeberg. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.
- [21] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- [22] Y. Li, Y. Li, and N. Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 513–528, 2018.
- [23] P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 514–521. IEEE, 2009.
- [24] P. Mettes and C. G. Snoek. Spatial-aware object embeddings for zero-shot localization and classification of actions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4443–4452, 2017.
- [25] A. Piergiovanni, C. Fan, and M. S. Ryoo. Learning latent subevents in activity videos using temporal attention filters. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [26] A. Piergiovanni and M. S. Ryoo. Learning latent super-events to detect multiple activities in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5304–5313, 2018.
- [27] A. Piergiovanni and M. S. Ryoo. Temporal gaussian mixture layer for videos. *arXiv preprint arXiv:1803.06316*, 2018.
- [28] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.
- [29] 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.
- [30] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In

- International conference on machine learning*, pages 843–852, 2015.
- [31] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 371–380. ACM, 2015.
 - [32] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2004–2011. IEEE, 2009.
 - [33] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4597–4605, 2015.
 - [34] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497. IEEE, 2015.
 - [35] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.
 - [36] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2017.
 - [37] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1510–1517, 2018.
 - [38] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.
 - [39] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013.
 - [40] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4305–4314, 2015.
 - [41] 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.
 - [42] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 461–470. ACM, 2015.
 - [43] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.
 - [44] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*, pages 214–223. Springer, 2007.
 - [45] Y. Zhu, Y. Long, Y. Guan, S. Newsam, and L. Shao. Towards universal representation for unseen action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9436–9445, 2018.