Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Deeply-Supervised CNN Model for Action Recognition with Trainable Feature Aggregation

Yang Li, Kan Li*, Xinxin Wang
School of Computer Science, Beijing Institute of Technology, Beijing, China
{yanglee, likan, wangxx}@bit.edu.cn

Abstract

In this paper, we propose a deeply-supervised CNN model for action recognition that fully exploits powerful hierarchical features of CNNs. In this model, we build multi-level video representations by applying our proposed aggregation module at different convolutional layers. Moreover, we train this model in a deep supervision manner, which brings improvement in both performance and efficiency. Meanwhile, in order to capture the temporal structure as well as preserve more details about actions, we propose a trainable aggregation module. It models the temporal evolution of each spatial location and projects them into a semantic space using the Vector of Locally Aggregated Descriptors (VLAD) technique. This deeply-supervised CNN model integrating the powerful aggregation module provides a promising solution to recognize actions in videos. We conduct experiments on two action recognition datasets: HMDB51 and UCF101. Results show that our model outperforms the state-of-the-art methods.

1 Introduction

Human action recognition is one of the fundamental problems in computer vision with a wide range of applications including video surveillance, behavior analysis and video retrieval. Although it has drawn quite researchers' attention and made considerable progress over the past few years, some critical challenges remain. One of the key challenges is how to build appropriate spatiotemporal representations for videos to recognize human actions.

To address the challenge, most recent researchers [Karpathy *et al.*, 2014; Tran *et al.*, 2015; Wang *et al.*, 2016] focused on applying Deep Convolutional Neural Networks (CNNs) to learn more powerful video representations. However, since current CNN architectures lack the capability to model the entire video, they usually break an entire video into several short segments (i.e., frames or clips), and then extract CNN features for each segment separately and finally aggregate the multiple segment features into an integrated representation. Based on this idea, plenty of CNN models with different aggregation strategies have been proposed and achieved great improvement in the accuracy of action recognition.

According to the aggregation strategy, these models are divided into two classes. Some models aggregate multiple segment features using Bag-of-Words (BoW) based methods. They decompose the segment features extracted from CNNs as a collection of local spatial features and encode them into an orderless video-level representation using a dictionary [Girdhar *et al.*, 2017; Duta *et al.*, 2017]. Although these models can encode entire video into a semantic and compact representation, they ignore the video temporal structure that reveals the evolution process of frame appearance over time. Different from the above models constructing orderless representations, the other models work on explicitly exploiting the temporal structure of videos using, for example, Recurrent Neural Networks (RNNs) [Srivastava *et al.*, 2015] or ranking functions [Bilen *et al.*, 2016]. In most of these models, the inputs to RNNs or ranking functions are the features extracted from the fully-connected layer of CNNs. As the inputs are high-level features that compress much information, the constructed representations are limited in describing fine details about actions.

In our work, we propose a deeply-supervised CNN model integrating a new aggregation module that not only constructs semantic video-level representations but also captures the temporal structure of videos. In detail, the proposed model aggregates frame features at multiple convolutional layers to fully exploit the powerful hierarchical representations of CNNs. This is motivated by the recent research [Bau *et al.*, 2017] which shows that the convolutional feature maps are usually complementary since visual semantic concepts corresponding to different actions can emerge in different convolutional layers. Moreover, in our model, the video representations at different layers are learned in a deep supervision manner [Lee *et al.*, 2015], and the final prediction is obtained via the learnable ensemble of multiple scores predicted from different layers. In this way, we exploit multi-level video representations in a single network, which brings improvement in both performance and efficiency for action recognition.

The core of our proposed model is the aggregation module. It extracts the Local Evolution Descriptors (LEDs) in a video and constructs a robust video representation by projecting them into a semantic space. Different from the conventional

*Corresponding Author.

methods modeling the temporal evolution in frame-level features, our method models the temporal evolution for each spatial location in the video and constructs a set of LEDs. Then we encode them into a meta-action based representation using the VLAD technique. This is based on the fact that an action is composed of some dynamic action primitives, which are regarded as meta-actions in our work. For example, "shoot ball" can be represented as an aggregation of some subitems corresponding to "legs:jumping", "arms:throwing", "torso:moving up-down", "basketball:flying". We argue that the meta-action based representation is more discriminative for action recognition, because it not only captures the temporal evolution of the entire video but also contains more semantic information about actions.

We comprehensively evaluate our model on two public challenging datasets: HMDB51 [Kuehne *et al.*, 2011], and UCF101 [Soomro *et al.*, 2012], where the proposed model produces more accurate prediction results than state-of-the-art. We highlight our contributions as follows:

1. We propose a deeply-supervised CNN model, which fully exploits powerful hierarchical features of CNNs and constructs multi-level video representations to improve the performance of action recognition.
2. In our model, we present a trainable aggregation module to learn video-level action representations. It captures more details about actions through modeling the temporal evolution of each spatial location and applies VLAD technique to encode them into the semantic and compact meta-action based representation.
3. The proposed model is evaluated on two action recognition datasets. The results demonstrate that our model can build robust and discriminative video representations and achieve better performance than the state-of-the-art models.

2 Related Work

In this section, we provide a brief overview about CNN based action recognition approaches and aggregation methods for building video-level representations.

2.1 Action Recognition with CNNs

Recently, plenty of works focus on using CNNs for action recognition. Karpathy et al. [Karpathy *et al.*, 2014] trained a deep network on the Sports-1M activities dataset. It operated on individual frames and turned out to be less accurate than the state-of-the-art hand-craft representation. To capture the motion information in videos, Simonyan et al. [Simonyan and Zisserman, 2014] proposed a two-stream network, which takes RGB frames and optical flows as input respectively and fused the predictions from the two streams as the final output. Based on the two-stream architecture, several improved methods [Feichtenhofer *et al.*, 2016b; 2016a; Wang *et al.*, 2016; Carreira and Zisserman, 2017; Zhang *et al.*, 2016] have been proposed. For example, Feichtenhofer et al. [Feichtenhofer *et al.*, 2016b] developed an architecture that can fuse spatial and temporal cues at several levels of granularity in feature abstraction and achieved great improvements. Except for the two-stream based methods, there are also other aspects of explorations. Tran et al. [Tran *et al.*, 2015] explored 3D ConvNets on video streams for spatiotemporal feature learning for clips. In this way, they avoid to calculate the optical flow explicitly and achieved good performance. Carreira et al. [Carreira and Zisserman, 2017] introduced a new Two-Stream Inflated 3D ConvNet (I3D) that expanded filters and pooling kernels of ConvNets into 3D, which made it possible to learn seamless spatiotemporal features.

2.2 Building Video-level Representations

Modeling video-level temporal structure is crucial for better understanding of actions in videos. Recently, plenty of methods have been proposed to build video-level representations based on CNN features. Some of them [Girdhar *et al.*, 2017; Duta *et al.*, 2017; Lan *et al.*, 2017] integrates BoW based encoding techniques, such as Fisher Vector and VLAD, into CNN architectures to learn video-level representations. For example, Girdhar et al. [Girdhar *et al.*, 2017] aggregated local convolutional features across the entire extent of the video using VLAD encoding method. The other methods build video-level representations explicitly considering the temporal structure of videos. Most of them [Srivastava *et al.*, 2015; Richard *et al.*, 2017] feed the CNN features of multiple video segments into a RNN with LSTM [Donahue *et al.*, 2015] or GRU [Cho *et al.*, 2014] units. For example, Srivastava et al. [Srivastava *et al.*, 2015] used the multi-layer LSTM network to learn action representations. Richard et al. [Richard *et al.*, 2017] divided all videos into small blocks and eventually modeled long and complex video information using GRU. In addition, Bilen et al. [Bilen *et al.*, 2016] proposed a temporal pooling method, which captures the temporal evolution of a video by looking for a function that is capable of ordering the frames of the video temporally.

3 Method

We present an overview of our proposed model before going into details. The architecture of our model is depicted in Figure 1. Multi-level video representations are constructed by applying our proposed aggregation module at different convolutional layers. Then multiple scores are predicted through the corresponding classifiers and fused to produce the final result.

For aggregating frame features from the convolutional layers, we propose an aggregation module (see Figure 1 (b)). It extracts the Local Evolution Descriptors (LEDs) of a video, and encodes them into a video-level representation. The aggregation module brings two advantages: (1) capturing more details about the action through modeling the temporal evolution of each spatial location; and (2) projecting the local descriptors into a semantic space, leading to more discriminative video-level representations.

For obtaining the multi-level predictions, supervision is directly fed into corresponding convolutional layers (see Figure 1 (a)). Such deep supervision learning strategy boosts the performance via: (1) directly constructing multi-level video representations and producing multiple predictions; and (2)

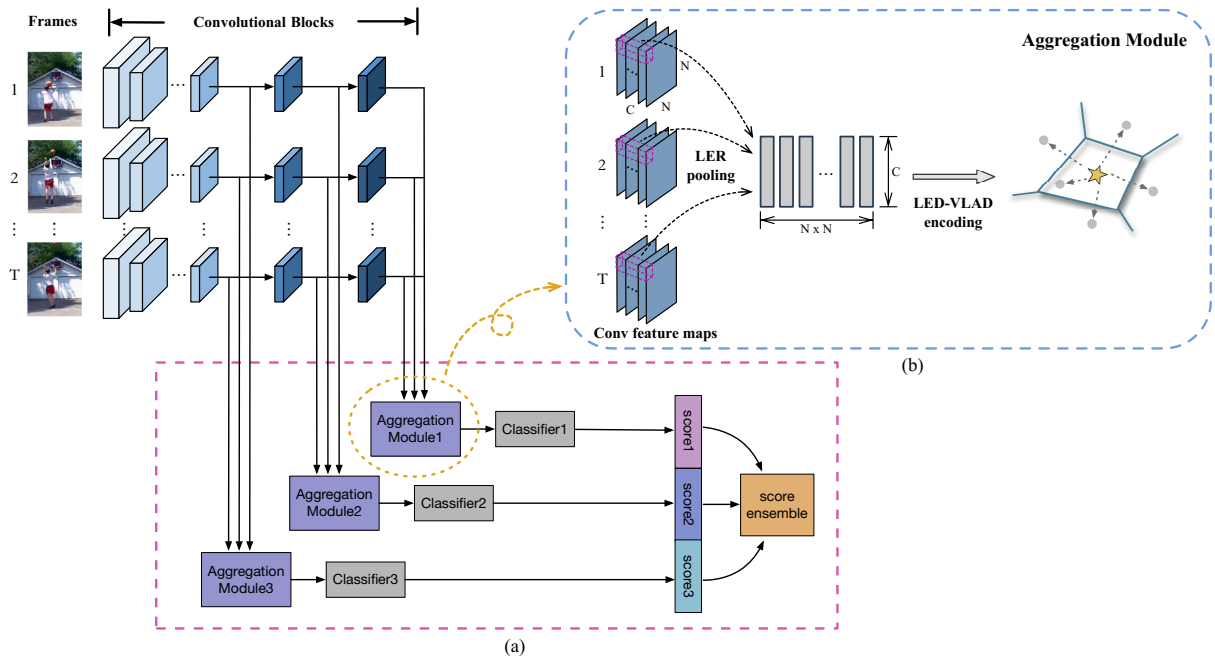


Figure 1: Architecture of our proposed model. (a) Our model constructs multi-level video representations by applying our proposed aggregation module at different convolutional layers, and learns to fuse the multiple scores predicted from different layers. (b) The proposed aggregation module models the temporal evolution of each spatial location and encodes them into a semantic video-level representation.

improving discrimination of intermediate layers, thus gaining improvement of overall performance.

3.1 Proposed Aggregation Module

The architecture of our proposed aggregation module is depicted in Figure 1 (b). The entire aggregation process can be decomposed into two cascaded stages, namely **local evolution descriptor extraction** and **local evolution descriptor encoding**. According to the two stages, we design an aggregation module with two operations: Local Evolution Rank (LER) pooling and LED-VLAD encoding. We will discuss the details in following sections.

Local Evolution Descriptor Extraction

Given T frames $[I_1, I_2, ..., I_T]$ of a video V and the frame at time t is represented as I_t, we can obtain a convolutional feature map for each frame from the last convolution layer of the convolutional blocks. We denote the size of the feature map as $N \times N \times C$, where $N \times N$ refers to the spatial dimensions of the feature map and C is the channel size. For extracting local spatial features, we individually take each spatial location and concatenate the values along all channels. In this way, we obtain $N \times N$ local spatial features for each feature map.

Inspired by the recent progress in video analysis that constructs video representations by learning to order the frames [Fernando *et al.*, 2015], we propose a method to generate the LED by encoding the temporal ordering of a sequence of local spatial features. Specifically, we chronologically order the local spatial features for a specific spatial location i and denote them as $[r_{i1}, r_{i2}, ..., r_{iT}]$, where $i = \{1, ..., N \times N\}$ and $r_{it} \in \mathbb{R}^C$. Then we define a ranking function that associates time t with a score $S(t, i \mid e) = e^T d_{it}$, where $e \in \mathbb{R}^C$ is a vector of parameters and $d_{it} = \frac{1}{t} \times \sum_{\tau=1}^{t} r_{i\tau}$ is the average of the local spatial features up to time t. In order to model the evolution of the spatial location i, we set the constraint that the later time is associated with a larger score, i.e. $q > t \Rightarrow S(q, i \mid e) > S(t, i \mid e)$. Thus, the function parameters e will reflect the rank of these local spatial features. Learning e is posed as a convex optimization problem:

$$
\begin{aligned}
& e^\star = \underset{e}{\operatorname{argmin}}\, E(e), \\
& E(e) = \frac{\lambda}{2} \|e\|^2 + \\
& \frac{2}{T(T-1)} \times \sum_{q>t} \max\left\{0, 1 - S\left(q, i \mid e\right) + S\left(t, i \mid e\right)\right\}.
\end{aligned}
\tag{1}
$$

The first term in this objective function is the usual quadratic regularizer. The second term is a hinge-loss soft-counting how many pairs $q > t$ are incorrectly ranked by the scoring function. By optimizing this function, we can map a sequence of local spatial features to a single vector $e^\star$. This vector contains the information to rank these local spatial features, and we define this vector as Local Evolution Descriptor (LED).

In order to simplify the process of seeking $e^\star$, we use the approximate technique proposed by [Bilen *et al.*, 2016] to solve the optimization problem of Eq. 1. Finally, the solution to the Eq. 1 can be reduced to:

$$
e^\star = \sum_{t=1}^{T} \alpha_t r_{it}, \tag{2}
$$

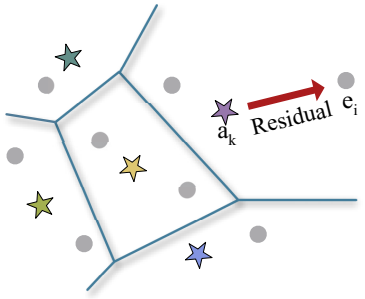


Figure 2: Illustration of LED-VLAD encoding. Points correspond to LEDs extracted from a video. Stars correspond to meta-action words that are initialized by clustering LEDs extracted from training videos. LED-VLAD aggregates the residuals between LEDs and meta-action words to construct the video representations.

where the coefficients $\alpha_t = 2(T-t+1)-(T+1)(H_T-H_{t-1})$ and the t-th Harmonic number $H_t = \sum_{t=1}^{T} \frac{1}{t}$. Eq. 2 means a weighted combination of the local spatial features.

Based on the approximate solution of the ranking function, we design the LER pooling layer and consider it as an intermediate layer of the CNN model. It takes T convolutional feature maps with size of $N \times N \times C$ and outputs $N \times N$ LED vectors with C dimensions.

Local Evolution Descriptor Encoding

Through the LER pooling, we obtain a collection of LEDs $[e_1, e_2, ..., e_{N\times N}]$ for a video. In this step, we aggregate these LEDs to construct the final video representation. The most intuitive way is to concatenate them in a vector. However, the vector dimension will be very high, which will induce heavy computational cost. In our work, we propose LED-VLAD method to encode the LEDs of a video into a compact and semantic representation.

Specifically, we first cluster the LEDs extracted from all training videos and consider the K cluster centers $[a_1, a_2, ..., a_K]$ as a vocabulary with K meta-action words. Then we divide the descriptor space $\mathbb{R}^C$ into K cells using the meta-action words, as shown in Figure 2. To encode the extracted LEDs of a video, each LED e_i is assigned to one of the cells and represented by a residual vector $e_i - a_k$, which records the difference between the LED and the meta-action represented by anchor a_k. These residual vectors are then summed across the entire video as:

$$h_k = \sum_{i=1}^{N\times N} \frac{e^{-\alpha\|e_i-a_k\|^2}}{\sum_{k'} e^{-\alpha\|e_i-a_{k'}\|^2}} (e_i - a_k), \tag{3}$$

where the first term in Eq. 3 is the soft-assignment of descriptor e_i to a_k and α is a tunable hyper-parameter; the second term is the residual between the LED and the k-th anchor point. The result h_k represents the aggregated descriptor in the k-th cell. Finally, K aggregated descriptors are stacked into a single vector $\nu \in \mathbb{R}^{C\times K}$ to represent the video.

Since the encoding method in Eq. 3 is differentiable and allows for back-propagating error gradients to lower layers of the network, we design the LED-VLAD layer. Moreover we incorporate it with the LER pooling layer to construct our aggregation module.

3.2 Action Recognition Model

CNNs are powerful visual models that are capable of capturing hierarchical features. In order to fully exploit the powerful hierarchical representations, we propose a deeply-supervised CNN model that exploits multiple convolutional layer information to improve the accuracy of action recognition.

Specifically, we select M convolutional layers from the standard CNN architecture. Each selected layer is then associated with an aggregation module and classifier to explicitly predict action scores. For combining all the parameters of the convolution and aggregation modules of the networks, we define:

$$\begin{aligned} W &= \left\{W^1_{conv}, ..., W^L_{conv}, W^{l_1}_{agg}, ..., W^{l_M}_{agg}\right\}, \\ w_c &= \left\{w^1_c, ..., w^M_c\right\}, \end{aligned} \tag{4}$$

where L denotes the total number of convolutional layers. The parameters for the m-th aggregation module are denoted as $W^{l_m}_{agg}$, and the parameters for the m-th classifier are denoted as w^m_c. We define a loss function that merges all the output layer classification error:

$$\Theta(W, w_c) = \sum_{m=1}^{M} L(W, w^m_c), \tag{5}$$

where L denotes the video-level loss function for action classification. Given a video V, its ground truth label is $g \in A$ where A defines all class labels. The number of the class labels is defined as Z. Then L is defined as the cross-entropy loss:

$$\begin{aligned} &L(W, w^m_c) = \\ &-\sum_{i=1}^{Z} [g = A_i] \, logP(s^m = A_i \mid V, W, w^m_c), \end{aligned} \tag{6}$$

where s^m indicates the predicted action label from the m-th convolutional layer, [ ] is the Iverson bracket notation for a boolean statement.

For fusing the multi-level predictions, a class-wise ensemble method is proposed. In this method, multiple score values for each category are summed according to corresponding weights to take full advantage of the complementarity of multi-level information. Here, we define w_f as the fusion weights, and F indicates the fused prediction. The loss function for the ensemble layer is defined as:

$$\begin{aligned} &\Phi(W, w_c, w_f) = \\ &-\sum_{i=1}^{Z} [g = A_i] \, logP(F = A_i \mid V, W, w^m_c, w_f). \end{aligned} \tag{7}$$

Then all the parameters W, w_c and w_f can be learned via minimizing the following objective function over all training set:

$$\begin{aligned} &(W, w_c, w_f) = \\ &argmin \left(\frac{1}{M}\Theta(W, w_c) + \Phi(W, w_c, w_f)\right). \end{aligned} \tag{8}$$

Given a test video, we can use the trained model to predict the action category.

4 Experiments

In this section, we report the experimental results to evaluate the proposed model for action recognition. We first introduce the action recognition benchmark datasets and evaluation metrics used to evaluate our model. Then, we show the implementation details of our model and present a detailed analysis of the results with different model settings. Finally, the results of our model and qualitative comparison with other state-of-the-art models are presented.

4.1 Datasets

We evaluate the proposed models on two popular action classification benchmarks, **HMDB51** [Kuehne *et al.*, 2011] and **UCF101** [Soomro *et al.*, 2012]. The HMDB51 dataset comprises of 51 human action categories, such as "backhand flip" and "swing baseball bat" and spans over 6766 videos. The videos are downloaded from Youtube, and each video contains a single action. The UCF101 dataset comprises of 101 human action categories like "Apply Eye Makeup" and "Rock Climbing" and spans over 13320 videos. The videos are realistic and relatively clean. Both datasets are split into three parts, and we report final performance averaged over all three splits.

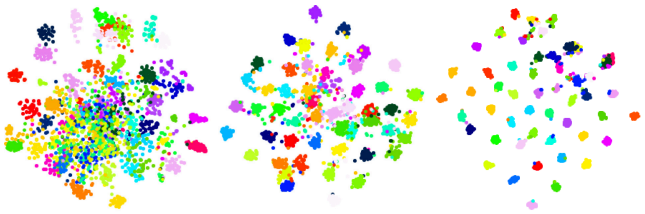
4.2 Implementation Details

In this section, we show the implementation details of our model. The input of our model is the RGB frames that are uniformly sampled from videos. In order to determine the value of T, we evaluated the classification accuracy of trained models with different values of T ($T = 5, 10, 20$ and 30). We found that the accuracy did not improve significantly when T = 20 or 30 compared with T = 10. It illustrates that T=10 is enough to capture the required temporal dependencies. Thus, we finally selected T = 10, which is a good trade-off between accuracy and efficiency. In our model, we choose the Inception with Batch Normalization (BN-Inception) [Ioffe and Szegedy, 2015] as the basic network architecture.

For initialing the convolutional parameters of our model, we first pre-train the network for action recognition using average pooling aggregation strategy at the $Mixed5\_c$ convolutional layer, which is the top convolutional layer of BN-Inception network. Then we transfer the pre-trained convolutional parameters to our model. For the parameters of the aggregation module, we experiment with different values of K ($K = 16, 32$ and 64) and obtain the best performance when $K = 32$. Moreover, we initialize the meta-action words by clustering the LEDs extracted from training videos using the k-means method. The cluster centers are jointly trained with the classifier and convolutional parameters, which helps to improve the final accuracy. At last, the weights of the classifiers are randomly initialized from a Gaussian distribution with zero mean and standard deviation of 0.01.

	Split1	Split2	Split3	Average
Avg Pooling	48.9	48.1	48.3	48.4
VLAD	51.4	49.2	48.6	49.7
Our Aggregation Module	57.2	55.8	55.1	56.0

Table 1: Comparisons with several baseline aggregation strategies.



(a) Avg-pooling (b) VLAD (c) Our Aggregation Module

Figure 3: Embedding visualizations of the features generated through three different aggregation strategies using tSNE. Each video is visualized as a point, and videos belonging to the same action have the same color. The features learned using our aggregation module are more semantically separable compared to the other strategies, which suggests that our aggregation strategy is a better way to learn video representations.

For capturing multiple convolutional layer information, we select $M = 3$ feature maps generated respectively from $Mixed5\_a, Mixed5\_b$ (the names of two relatively bottom convolutional layers), and $Mixed5\_c$ layers of the BN-Inception. Since recognizing actions usually requires high-level semantic information like object parts or body parts, we select the convolutional layers from the top layers that contain high-level information. When the model is trained, the network parameters are learned by back-propagating the loss function defined in Eq. 8 using the Adam [Kingma and Ba, 2014] with $\epsilon = 1e-4$. The network is trained with a momentum of 0.9 and a weight of decay of $4e-5$. We train it on a PC with 3.4GHz CPU, a TITAN X GPU, and 64G RAM.

4.3 Results and Performance Analysis

In this section, we present a detailed analysis of our model in two aspects: multi-frame feature aggregation strategy and multi-level prediction ensemble techniques.

Multi-frame Feature Aggregation Strategy

In order to study the effect of our proposed aggregation module, we compare it with several baseline aggregation strategies on HMDB51 dataset. In this exploration, we do not consider to combine information from multiple convolutional layers. In order to compare the performance of different aggregation strategies, we use the convolutional feature maps from the $Mixed5\_b$ layer that has the best performance among the three candidate layers. We train three end-to-end action recognition networks with different aggregation strategies and predict the results respectively. The accuracies of different aggregation strategies on the three splits of HMDB51 are shown in Table 1. We see that our aggregation module achieves the best performance. Although VLAD can construct semantic representations for actions, it encodes the local spatial features into orderless representations that ignore the temporal structure of videos. This makes it difficult for VLAD method to distinguish the categories like "stand" and "sit" in HMDB51 dataset. Thus VLAD method achieves

	Mixed5_a	Mixed5_b	Mixed5_c
HMDB51	43.7	57.2	52.6
UCF101	71.5	86.5	73.2

Table 2: Accuracies of trained models that aggregate frame features at different convolutional layers.

	HMDB51	UCF101
Average	57.4	85.4
Weighted Average	57.9	86.7
Class-wise Ensemble	62.5	90.1

Table 3: Comparisons of different ensemble techniques.

limited improvement compared to the average pooling strategy. Our aggregation module not only captures the temporal structure of videos through the LER pooling but also encodes the LEDs into a semantic video-level representation. That is the reason why our aggregation module can achieve better improvements.

Furthermore, we qualitatively evaluate the aggregation strategies by visualizing the constructed features using t-SNE [Maaten and Hinton, 2008]. Figure 3 shows the feature embedding results. We can see that the features generated through Avg-pooling are very similar hence cluster together. The features generated through VLAD are relatively spread. At last, the features generated by our aggregation module are more semantically separable and mixed compared to the VLAD strategy. This shows that our aggregation module is a better way to learn video-level representations.

Multi-level Prediction Ensemble Technique

Here, we compare different ensemble techniques fusing the multi-level predictions. First, we show the classification accuracy of different trained models, which aggregates frame features at the $Mixed5\_a, Mixed5\_b$, and $Mixed5\_c$ layers respectively, on the split 1 of HMDB51 and UCF101 in Table. 2. Then, we train networks using different ensemble techniques to fuse the predictions from the multiple layers. Table 3 shows the results on the split 1 of HMDB51 and UCF101. The results show that the class-wise ensemble method, where multiple score values for each category are summed according to corresponding weights, presents the best performance among all these ensemble techniques. As discussed above, multi-level video presentations learned from our model are complementary. It often manifests as that the video representations constructed from different convolutional layers exhibit different recognition capability for the same action category. Thus, our ensemble method, which provides class-wise supervision to fuse predictions from multiple layers, can take full advantage of the complementary characteristic and obtains better performance.

4.4 Comparisons with State-of-the-art Methods

In this section, we compare our model with the state-of-the-art methods for action recognition. First, We compare our model with the previous RGB-image based methods in the upper part of Table 4. Spatiotemporal CNN [Karpathy *et al.*, 2014] and TSN (RGB-only) [Wang *et al.*, 2016] use the common pooling techniques. ActionVLAD [Girdhar *et al.*, 2017], ST-VLMPF [Duta *et al.*, 2017] and Attentional Pooling [Girdhar and Ramanan, 2017] use more complicated pooling and encoding techniques to construct video-level representations. C3D[Tran *et al.*, 2015] and I3D [Carreira and Zisserman, 2017] apply 3D convolution to learn temporal structure. We can see that our model obtains the highest accuracy with 62.4% on HMDB51 and 90.5% UCF101 respectively, which significantly outperforms the RGB-image based methods mentioned above. This demonstrates that our deeply-supervised model integrating our powerful aggregation module provides a promising way to recognize actions in videos.

	HMDB-51	UCF-101
Spatiotemporal CNN [Karpathy *et al.*, 2014]	-	65.4
TSN (RGB-only) [Wang *et al.*, 2016]	-	85.7
ActionVLAD (RGB-only)[Girdhar *et al.*, 2017]	51.2	-
ST-VLMPF [Duta *et al.*, 2017]	49.8	81.8
Attentional Pooling [Girdhar and Ramanan, 2017]	52.2	-
C3D[Tran *et al.*, 2015]	51.6	82.3
I3D (RGB-only) [Carreira and Zisserman, 2017]	49.8	84.5
Our Model	**62.4**	**90.5**
Two-Stream [Simonyan and Zisserman, 2014]	59.4	88.0
Two-Stream fusion [Feichtenhofer *et al.*, 2016b]	65.4	92.5
ST-ResNet [Feichtenhofer *et al.*, 2016a]	66.4	93.4
ActionVLAD [Girdhar *et al.*, 2017]	66.9	92.7
Two-Stream CNN+LSTM [Yue-Hei Ng *et al.*, 2015]	-	88.6
Composite LSTM Model [Srivastava *et al.*, 2015]	44.0	84.3
TSN [Wang *et al.*, 2016]	69.4	94.2
Two-Stream I3D [Carreira and Zisserman, 2017]	66.4	93.4
Our Model + optical flow	**73.9**	**95.8**

Table 4: Comparisons with the state-of-the-art methods.

Most state-of-the-art models use two-stream framework, i.e. one stream is trained on RGB frames and the other on optical flows. For fair comparison, we compare our model combined with optical flow to the state-of-the-art two-stream methods, and the results in the lower part of Table 4 show our model outperforms all of the state-of-the-art methods.

5 Conclusions

We propose a deeply-supervised CNN model for action recognition. The proposed model learns multi-level video representations to fully exploit the complementary convolutional layer information. Moreover, in order to build more robust video-level representations, we introduce a new aggregation module that can capture detailed evolution information of videos and construct semantic action representations. Experiment results show that our model provides a better resolution to learn action representations and outperforms existing methods on two popular benchmark datasets.

Acknowledgments

This work was partly supported by Beijing Natural Science Foundation (4172054).

References

[Bau *et al.*, 2017] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, pages 3319–3327, 2017.

[Bilen *et al.*, 2016] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *CVPR*, pages 3034–3042, 2016.

[Carreira and Zisserman, 2017] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 4724–4733, 2017.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[Donahue *et al.*, 2015] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634, 2015.

[Duta *et al.*, 2017] Ionut Cosmin Duta, Bogdan Ionescu, Kiyoharu Aizawa, Nicu Sebe, et al. Spatio-temporal vector of locally max pooled features for action recognition in videos. In *CVPR*, pages 3205–3214, 2017.

[Feichtenhofer *et al.*, 2016a] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, pages 3468–3476, 2016.

[Feichtenhofer *et al.*, 2016b] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, pages 1933–1941, 2016.

[Fernando *et al.*, 2015] Basura Fernando, Stratis Gavves, Oramas Mogrovejo, José Antonio, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, pages 5378–5387, 2015.

[Girdhar and Ramanan, 2017] Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. In *NIPS*, pages 33–44, 2017.

[Girdhar *et al.*, 2017] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, pages 3165–3174, 2017.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[Karpathy *et al.*, 2014] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kuehne *et al.*, 2011] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011.

[Lan *et al.*, 2017] Zhenzhong Lan, Yi Zhu, Alexander G Hauptmann, and Shawn Newsam. Deep local video feature for action recognition. In *CVPRW*, pages 1219–1225, 2017.

[Lee *et al.*, 2015] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, pages 562–570, 2015.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[Richard *et al.*, 2017] Alexander Richard, Hildegard Kuehne, and Juergen Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *CVPR*, pages 1273–1282, 2017.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014.

[Soomro *et al.*, 2012] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[Srivastava *et al.*, 2015] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852, 2015.

[Tran *et al.*, 2015] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497, 2015.

[Wang *et al.*, 2016] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36, 2016.

[Yue-Hei Ng *et al.*, 2015] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702, 2015.

[Zhang *et al.*, 2016] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time action recognition with enhanced motion vector cnns. In *CVPR*, pages 2718–2726, 2016.