Embedding Sequential Information into Spatiotemporal Features for Action Recognition

Yuancheng Ye[1] and Yingli Tian[1,2]
[1]The Graduate Center, City University of New York
yye@gradcenter.cuny.edu

[2]The City College, City University of New York
ytian@ccny.cuny.edu

Abstract

In this paper, we introduce a novel framework for video-based action recognition, which incorporates the sequential information with the spatiotemporal features. Specifically, the spatiotemporal features are extracted from the sliced clips of videos, and then a recurrent neural network is applied to embed the sequential information into the final feature representation of the video. In contrast to most current deep learning methods for the video-based tasks, our framework incorporates both long-term dependencies and spatiotemporal information of the clips in the video. To extract the spatiotemporal features from the clips, both dense trajectories (DT) and a newly proposed 3D neural network, C3D, are applied in our experiments. Our proposed framework is evaluated on the benchmark datasets of UCF101 and HMDB51, and achieves comparable performance compared with the state-of-the-art results.

1. Introduction

Nowadays, there are overwhelming images and videos on the Internet, which draws more and more attentions about how to make use of them in various tasks of computer vision. Thanks to many innovative techniques, such as crowdsourcing, some large scale datasets are organized from chaotic pools, *e.g*. ImageNet [3] and Sport1M [12], which largely facilitate a lot of research work in computer vision. However, traditional methods usually cannot fully take advantage of large scale datasets, and have difficulties for transferring learned knowledge from one dataset to another. Since the seminal paper [13] which makes use of the powerful computation ability of GPUs, deep neural networks (DNNs) have enjoyed a renaissance in various topics of computer vision, such as image classification[22, 6], object detection [7, 8], image description [5, 11], and etc. With carefully designed structures, DNNs can learn general and discriminative models from large scale datasets. Moreover, DNNs can propagate the knowledge learned from one dataset to another by a finetune process, which renders DNNs the ability to utilize the information learned from large scale sources to tackle the problems in relatively small datasets.

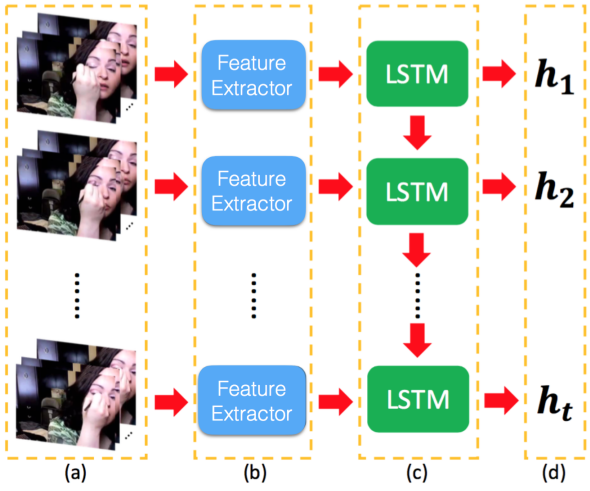

Figure 1. Proposed hybrid action recognition framework. (a) Video clips sliced with a fixed length. (b) Extracting spatiotemporal features from each video clip. (c) Learning sequential information of these clips through the LSTM model. (d) The feature sequence corresponding to the input clips respectively.

Convolutional neural networks [16] (CNNs) are specific DNN structures, which have enjoyed great success in many image-based tasks. Many endeavors have been made to extend CNNs to the video domain. However, this task is

difficult mainly due to three major problems: 1) The volume of video data is much bigger than image. How to efficiently handle video data in the limited GPU memory is not tractable. 2) Compared to two dimensional image data, videos have additional temporal dimension. It is important to take into account the temporal information, but most CNN techniques are only image-based. 3) Videos contain different number of frames. Unlike the images, for which a simple resizing process can transform them into a uniform size, video interpolation or extrapolation process may lose important temporal information about the video.

In addition to CNNs, recurrent neural networks (RNNs) have also been applied to many computer vision tasks. One of the major attributes of RNNs is that they can preserve or discard information based on long term memories, which enables them to take into account the sequential dependencies of the inputs. Another advantage of RNNs is their ability to handle inputs with variant lengths. Despite all the merits presented above, there is a significant limitation in the structure of RNNs, which is how to back propagate error information in a long range to update weights in layers. Many versions of RNNs have been proposed to alleviate this problem, among them the long short term machine [9] (LSTM) is one of the most popular.

In this paper, we propose a novel hybrid framework, which combines RNNs with extracted spatiotemporal features, for action recognition task. As illustrated by Figure 1, the input of our proposed framework is a sequence of sliced clips. Then the spatiotemporal features are extracted from each clip. After that the LSTMs are applied to learn sequential information among these clips. Finally, a sequence of output is produced corresponding to the sequence of video clips in the input. Our framework is similar to [5], which is the pioneering work to integrate CNNs with RNNs. However, their work is frame-based and only takes into account the sequential information of the frames. Our framework, which combines extracted spatiotemporal features with RNNs, can model both the spatiotemporal information of the sliced clips and the sequential characteristics of these clips in the whole video.

For the algorithms of extracting spatiotemporal features from the sliced clips, both dense trajectories [24] and C3D [23] are incorporated in our system. The dense trajectories are produced by tracking the interested points, and the spatiotemporal features are extracted along each trajectory. Compared with the algorithm of dense trajectories, C3D is a newly proposed model which constructs a 3D convolutional neural network to learn the spatiotemporal features from each video clip.

The contributions of our paper are three folds. Firstly, by integrating spatiotemporal features with LSTMs, the sequential information of the sliced clips can be modeled, which improves the discriminative power of the final feature representations of the videos. Secondly, because the input of our framework is a sequence of sliced clips, the whole video can be easily presented in a relatively short sequence, which makes our algorithm very efficient when comes to long videos. Finally, an extensive analysis of the influence of sequential length and step size for RNNs to handle variable lengths of inputs is studied.

The rest of the paper is organized as follows. Section 2 introduces related work about action recognition and techniques of deep learning in the video-based applications. Section 3 provides detailed background information of this paper. Specifically, Section 3.1 provides a brief introduction of the LSTM and the dense trajectories and C3D methods implemented in our framework for extracting the spatiotemporal features are discussed in Sections 3.2 and 3.3 respectfully. The details of our proposed framework are illustrated in the Section 4. Section 5 demonstrates the experimental results of our framework conducted on the benchmark datasets: UCF101 and HMDB51. Finally, conclusions are presented in the Section 6.

2. Related work

In general, an action recognition framework comprises four major components: feature extraction, feature encoding, pooling and normalization, and classification. The feature extraction is to locate distinct interested points or trajectories in videos and obtain their descriptors based on the appearance and motion information around them. The criteria of how to distinguish those interested points or trajectories are manually defined. The feature encoding is to encode generated features into feature representations. The common encoding algorithms for action recognition are bag-of-visual words (BOV) and Fisher Vector (FV). BOV entails codebook vocabularies usually computed by the K-means algorithm, and then the extracted features are assigned to the correspondent clusters. Fisher Vector incorporates Gaussian Mixture Models (GMMs) to encode each extracted feature. For the pooling and normalization component, the encoded features are aggregated into one feature representation for each video. For classification, various methods have been applied based on the characteristics of the generated features.

By extending the Harris corner detector to the 3D domain, Laptev and Lindeberg proposed the STIPs [15] by extracting sparse interest points in the space-time domain. Based on this idea, many other algorithms [2, 4, 27] have been proposed. Instead of extracting sparse points in videos, Wang *et al*. [25] first introduced improved dense trajectories and the corresponding descriptors around the trajectories to tackle the action recognition task. Recently, [26] proposed a novel algorithm to integrate detected trajectories with CNN learned features. Although some interested points or trajectories and their corresponding descrip-

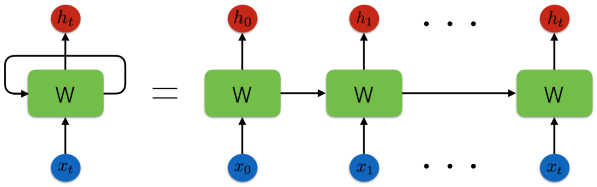

Figure 2. Illustration of recurrent neural networks (RNNs). The right side is the unrolled equivalent of the left side. This chain-like structure indicates that the sequential information can be modelled by RNNs.

tors have very good performance on most benchmark action recognition datasets, the appealing attributes of DNNs propel researchers to design efficient and effective DNN structures to perform action recognition tasks on large scale datasets.

An intuitive way to extend image-based CNN structures to the video domain is to perform the finetune and classification process on each frame independently, and then conduct a later fusion, such as average scoring, to predict the action class of the video. Despite of its simple implementation, this method achieves comparable results to many carefully designed algorithms. To incorporate temporal information in the video, [20] introduces a two-stream framework. One stream is based on RGB images and the other is based on the stacked optical flows. Although this work proposes an innovative way to learn temporal information using CNN structure, in essence it is still image-based since the third dimension of stacked optical flows collapses immediately after the first convolutional layer. 3D CNN structure [10] provides another perspective to apply deep learning techniques to video-based tasks. Instead of sticking to the 2D convolutional operations, temporal information can be learned from the 3D convolutions with both spatial and temporal stride. The C3D model is proposed based on this idea. Because of the limited GPU computing resources, instead of taking the whole video as the input, the C3D model only operates on the sliced clips with fixed length, and later fusion methods are performed to obtain the final category of the entire video. This compromised implementation trick may neglect the sequential dependencies of those isolated clips.

As described previously, RNNs can handle inputs with variable lengths. Both [17] and [5] propose to input CNN features of each frame into RNN structures and achieve good results. The former one emphasizes the pooling strategies and how to fuse different features, while the latter put more attention on how to train an end-to-end DNN structure that integrates CNNs with RNNs. To overcome the shortcoming of RNNs that the error is hard to back propagate in the long range, both algorithms choose LSTM as the RNN structure. The function of RNNs can be viewed as embedding sequential information into the input sequence.

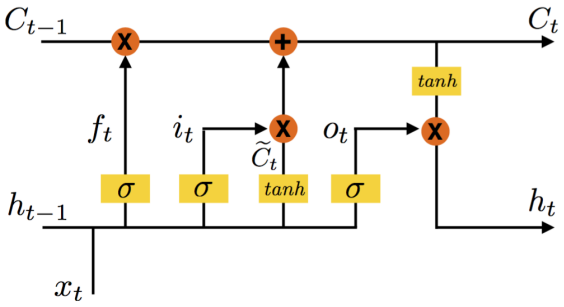

Figure 3. Illustration of the LSTM memory unit implemented in our framework. The meanings of symbols in the figure are demonstrated in equations (1) - (6).

Therefore the intention of both [17] and [5] is to explore the sequential information of the independent frames. This approach neglects the spatiotemporal attributes within the video, and to complement this shortcoming both studies apply optical flow information as an additional channel and performs a later fusion on these two channels. As demonstrated in both studies, by fusing the optical flow information the accuracy of action recognition can be significantly improved. This phenomenon indicates that the discriminative power of feature representations of videos can be improved by combining the sequential and spatiotemporal information.

Inspired by the above observation, we propose a spatiotemporal-LSTM hybrid framework for action recognition which takes advantage of both spatiotemporal features and LSTMs. Specifically, both dense trajectories and C3D are utilized in our framework to extract spatiotemporal features of sliced clips. Then the extracted spatiotemporal features are fed into the LSTM model to learn sequential information between these features. The experimental results demonstrate that by incorporating both spatiotemporal and sequential information, the performance can be improved.

3. Background

In this section, three important components in our proposed framework are introduced briefly: LSTM, dense trajectories and C3D.

3.1. LSTM

Before being applied to the vision tasks, recurrent neural networks have succeeded in the area of speech recognition, machine translation, and etc. One of the most notable advantages of RNNs is to handle inputs with variant lengths. A simple diagram of RNN model is illustrated in the Figure 2. This diagram shows that the RNN structures can model the sequential information of the inputs. The initial models of RNNs have a common problem about back

propagating errors to update layers in the long range. In the paper [9], LSTM is introduced to alleviate this problem.

LSTM has the ability to preserve or discard information by carefully designed gate mechanisms. The illustration of LSTM memory unit is showed in the Figure 3. The following formulas demonstrate how the hidden layer (output of the LSTM memory unit) is computed:

$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \quad (1)$$
$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \quad (2)$$
$$\widetilde{C}_t = tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c) \quad (3)$$
$$C_t = f_t \otimes C_{t-1} \oplus i_t \otimes \widetilde{C}_t \quad (4)$$
$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \quad (5)$$
$$h_t = o_t \otimes tanh(C_t) \quad (6)$$

In the above equations, W_* denotes the weight matrix of the corresponding layer and b_* is the bias vector. In our implementation, all bias vectors are set to 0, and therefore the weight matrices are the only parameters to be learned. $\sigma(x) = (1+e^{-x})^{-1}$ represents the sigmoid function, which squashes each element of the input vector into the range $[0, 1]$. $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is the hyperbolic tangent function, which makes sure that every element of the output vector falls in the range $[-1, 1]$. Eq. (1) describes the behavior of the forget gate layer, which dictates which elements in the input vectors, h_{t-1} and x_t, should be preserved. The input gate layer is simulated by Eq.(2), which decides which values in the newly created vector, described by Eq. (3), should be updated. The update process is showed in Eq. (4). Eq. (5) presents the output layer in the LSTM memory unit, which dictates which elements of the vector output by Eq. (6) should be kept in the hidden layer h_t.

As demonstrated by the above procedures, LSTM integrates many carefully designed gate layers to make the decision about whether keeping or ignoring specific elements of input vectors. Moreover, the uppermost line in the Figure 3 represents the cell state, which preserves the learned sequential information during the whole process.

3.2. Dense Trajectories

The algorithm of dense trajectories is proposed by Wang *et al*. [25]. Firstly, the interested points are located in each frame by the methods introduced in the paper [19] on a set of different scales. Then these points are tracked through a fixed number of frames, which produce a trajectory for each corresponding interest point. These trajectories are pruned by a set of criteria, e.g. variance.

Along each resulting trajectory, a descriptor is generated based on both spatial and temporal attributes around that trajectory. Specifically, the neighborhood around each trajectory is split into spatiotemporal cells. As shown in Figure 4, in each cell, the histogram of gradient (HOG), histogram of optical flow (HOF) and motion boundary histogram (MBH) are calculated. After that, BOV is utilized in our experiments to encode trajectory descriptors.

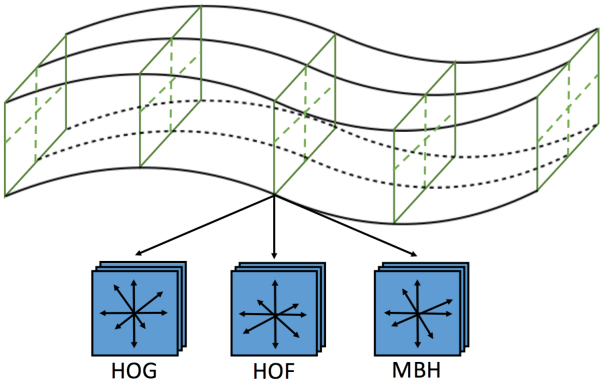

Figure 4. Illustration of trajectory-aligned descriptors. A tube around each trajectory is split into spatiotemporal cells with computed features of histogram of gradient (HOG), histogram of optical flow (HOF) and motion boundary histogram (MBH).

3.3. C3D

The C3D [23] is a newly developed 3D convolutional neural network structure. It slices the whole video into fixed length clips, and then conducts the finetune and classification processes on these clips independently.

In the 2D CNNs, the dimension of each feature map is $n \times h \times w$, where n stands for the number of filters in the corresponding convolutional layer, h and w represents the height and width of the feature map. The spatial size of the filter in the 2D convolutional process is defined manually, while the third dimension is automatically set to the value of the first dimension of the feature maps in the previous layer, which is also the number of filters in that layer.

In the 3D CNNs, the dimension of the feature maps produced by each convolutional layer is $n \times l \times h \times w$, where the additional parameter l stands for the number of frames. For the 3D filters in the 3D convolutional process, in addition to the spatial size, temporal length should also be set manually. The fourth dimension of the 3D filter is also automatically set to the first dimension of the feature maps in the previous convolutional layer. Figure 5 illustrates the 3D convolutional process in the C3D method. Moreover, instead of only pooling on the spatial domain, the pooling processes in the 3D CNNs pool the features in a small cuboid. Please note that if the temporal lengths of all 3D pooling layers are set to 2, then the number of 3D pooling layers in the 3D CNN structure should be $log_2 n$, where n is the length of sliced clips. After all the convolutional processes, a feature vector is generated and then fed into the fully connected layers. As observed from Table 1, the network structure in

layer	conv1a	pool1	conv2a	pool2	conv3a	conv3b	pool3	conv4a	conv4b	pool4	conv5a	conv5b	pool5	fc6	fc7
size	$3\times3\times3$	$1\times2\times2$	$3\times3\times3$	$2\times2\times2$	$3\times3\times3$	$3\times3\times3$	$2\times2\times2$	$3\times3\times3$	$3\times3\times3$	$2\times2\times2$	$3\times3\times3$	$3\times3\times3$	$2\times2\times2$	-	-
spatial stride	1	2	1	2	1	1	2	1	1	2	1	1	2	-	-
temporal stride	1	1	1	2	1	1	2	1	1	2	1	1	2	-	-
channel number	64	64	128	128	256	256	256	512	512	512	512	512	512	4096	4096

Table 1. The network architecture in the C3D model.

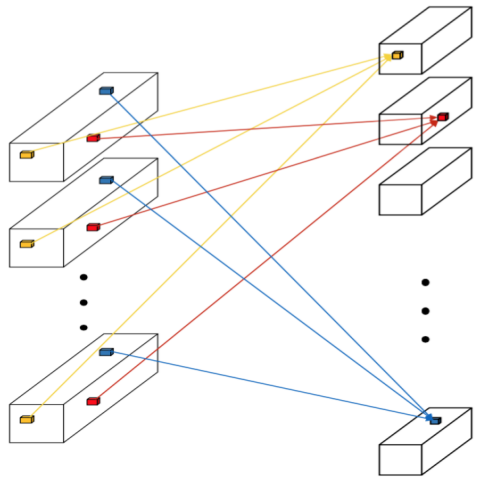

Figure 5. Illustration of the 3D convolutional process in the C3D algorithm. Different color cuboids represent different 3D filter kernels.

C3D has 8 convolutional layers, 5 max-pooling layers, and 2 fully connected layers. The sizes of all 3D convolutional kernels are $3 \times 3 \times 3$, and the stride of these kernels are all 1 in both spatial and temporal domain. The sizes of all pooling kernels are $2 \times 2 \times 2$, except for the first one, which is $1 \times 2 \times 2$.

Due to the limitation of GPU computing resources, the C3D model only operates on the video clips with fixed length. Although by conducting later fusion on these clips, comparable results can be achieved, important sequential information between sliced clips is neglected.

4. Spatiotemporal-LSTM hybrid framework

Based on the above discussion, we propose the spatiotemporal-LSTM hybrid framework, which can capture both spatiotemporal and sequential information of sliced video clips.

In our proposed framework, the video is sliced into fixed length clips, from which spatiotemporal features are extracted. After that these features are fed into the LSTM model to embed the sequential information to the final video representation.

For the spatiotemporal feature extractor, two algorithms are applied in our framework. The first is the dense trajectories (DT) algorithm, which is one of the state-of-the-art traditional methods for action recognition. Another is the C3D model which is a newly proposed deep learning technique for action classification. As demonstrated in the experiments, by concatenating the features generated by the DT algorithm and the C3D model, the performance can be improved. This phenomenon is consistent with the experimental results reported in the paper [23], in which by fusing the features generated by the improved dense trajectories (iDT) algorithm[25] and the C3D model, better accuracy of action recognition can be achieved.

In our implementation, the dense trajectories algorithm is utilized instead of iDT, since iDT entails the results of human detections before extracting trajectories from the video. Compared with the original version of dense trajectories algorithm, two modifications have been made in our framework. First, to be consistent with the clip length in the C3D model, the trajectory length is changed to 16. Second, to generate the optical flows, we apply the Epicflow [18], which is one of the state-of-the-art optical flow generating algorithms. To make the dimension of the encoded feature representation to be consistent with the one produced by the C3D model, BOV is implemented instead of Fisher Vector.

In addition to train a C3D network on the RGB frames of videos, we also train a motion C3D network on the optical flow images. The optical flow images are generated by stacking the x-component, the y-component and the magnitude of the flow on three channels respectively. Each element in the image is then multiplied by 16 and converted to the closest integer between 0 and 255. This practice has demonstrated good performance in many other studies [5, 17]. As observed in the experimental results, by fusing the features generated by both RGB and motion C3D models, the performance can be improved, which indicates that complementary information is provided by training deep neural networks on the optical flow channels.

The features extracted by both RGB and motion C3D networks are concatenated with the features generated by the dense trajectories algorithm to form the feature representations of the clips. Before fed these feature representation into the LSTM model, a fully connected layer is also added to smooth the transition process.

In our framework, the LSTM model serves as a sequential information encoder. The video $\mathbb{V}$ is sliced with a certain overlap into clips $\langle c_1, c_2, \ldots, c_n \rangle$. These sliced video clips are represented by the spatiotemporal features extracted from them:

$$\langle c_1, c_2, \ldots, c_n \rangle \mapsto \langle f_1, f_2, \ldots, f_n \rangle, \tag{7}$$

and then the sequence of these features, in the same order

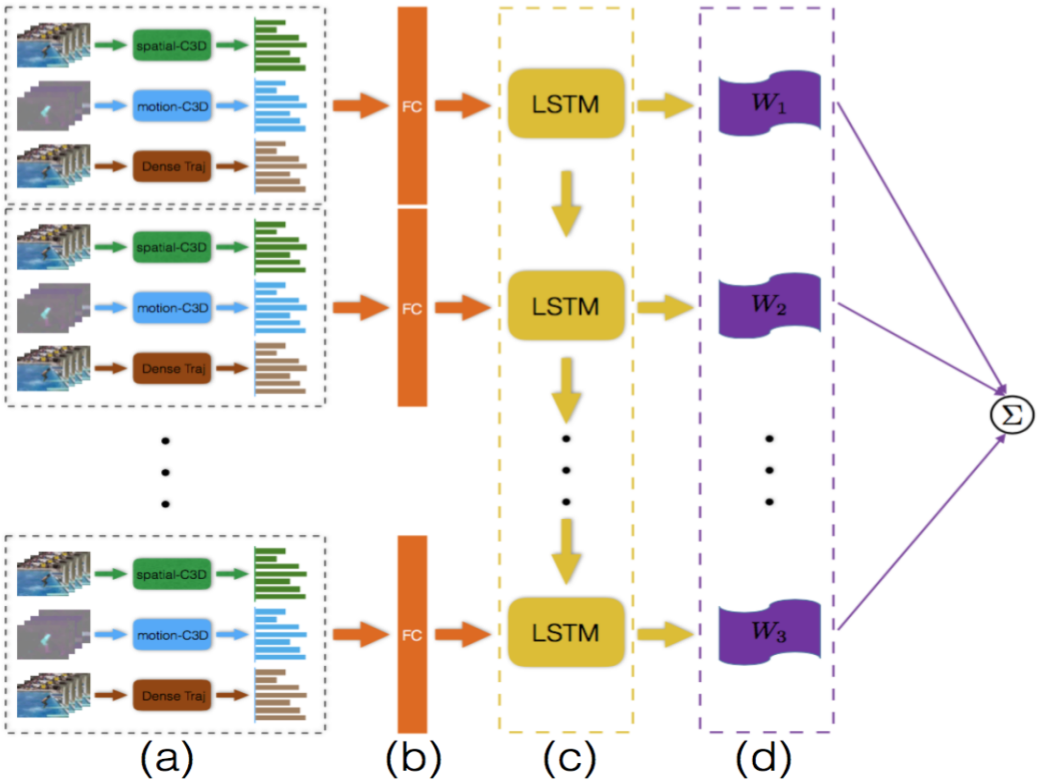

Figure 6. Spatiotemporal-LSTM framework: (a) Three types of spatiotemporal features are extracted from sliced clips by spatial-C3D, motion-C3D, and dense trajectories respectively. These three features are then ℓ_2 normalized and concatenated to form the feature representation of the video clip. (b) A fully connected layer is added to make the concatenated features more suitable for the LSTM model. (c) The features output by the fully connected layer are fed into the LSTM network in the same sequence order as the video clips. This process can embed the sequential information into the extracted spatiotemporal features. (d) The predictions made by the output of LSTM model. These predictions are averaged to get the final prediction of category of the video.

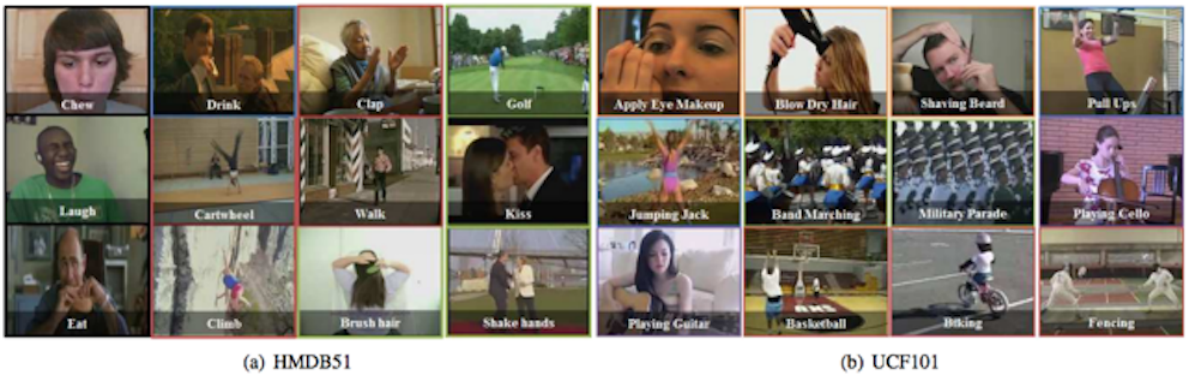

Figure 7. Examples of the frames of various concepts in the HMDB51 and UCF101 datasets.

as the video clips, is fed into the sequential information encoder:

$$\langle y_1, y_2, \ldots, y_n \rangle = LSTM(\langle f_1, f_2, \ldots, f_n \rangle), \quad (8)$$

Therefore, the final feature representation, $\langle y_1, y_2, \ldots, y_n \rangle$, output by our proposed spatiotemporal-LSTM framework includes both spatiotemporal attributes and the sequential information of these sliced clips.

For each element in the output feature sequence, $\langle y_1, y_2, \ldots, y_n \rangle$, a softmax classifier is applied, and the re-

sulting predictions are summed up to make the final prediction:

$$Score = \sum_{i=1}^{n} \varphi(y_i), \tag{9}$$

where φ represents the softmax classifier.

The whole structure of our proposed spatiotemporal-LSTM framework is presented in the Figure 6. As observed from the experiments, more discriminative power is rendered to the feature representations output by our framework, which contributes to better performance on the action recognition task.

5. Experiments

In this section, we first describe the action recognition datasets utilized in our experiments. Then the implementation details are presented. A parametric study is conducted to explore the impact of the step size and sequence length in the LSTM model. At last, the comparison of the results of our proposed spatiotemporal-LSTM framework with the baseline results is demonstrated.

5.1. Datasets

Our proposed spatiotemporal-LSTM framework is evaluated on two datasets: HMDB51 and UCF101. Examples of the frames of these two datasets are presented in the Figure 7. The details of the datasets are shown below:

HMDB51 dataset [14] consists of 51 action categories and 6766 videos. For each action class, there are 70 video samples for training and 30 video samples for testing. This dataset provides both original and stabilized videos, our experiments are conducted on the original version. There are three train/test data splits provided for this dataset, and an average accuracy on these three splits are reported in our experiments.

UCF101 dataset [21] is extended from the UCF50 dataset. It contains 101 action classes that can be divided into five types: human-object interaction, body-motion, human-human interaction, playing musical instruments, and sports. There are total 13320 video clips contained in this dataset, with fixed frame rate of 25 FPS and resolution of 320×240. Again, we report an average performance of our framework on the three train/test splits provided by this dataset.

5.2. Implementation Details

Around each trajectory, a volume of pixels, $N \times N \times L$, is selected to compute the corresponding descriptors. To extract trajectories with the same fixed length as the clip length in the C3D method, the lengths of trajectories are set to 16. The patch size, $N \times N$, is set to 32×32. This volume is then split into $n_\sigma \times n_\sigma \times n_\tau$ cells, where $n_\sigma \times n_\sigma$ is the spatial size and n_τ is the temporal size. In our experiments, n_σ is set to 2 and n_τ is set to 3. In each cell, HOG, HOF, MBHx and MBHy are computed with dimensions of 8, 9, 8 and 8 respectively. These histograms are then ℓ_2 normalized and concatenated. Therefore, for each trajectory, the dimension of the corresponding descriptor is $2 \times 2 \times 3 \times (8 + 9 + 8 + 8) = 396$.

To make the dimensions of final feature representations to be consistent with the output of the C3D model, the standard BOV is applied in our framework. The number of visual words is set to 4096, which is the same value as the dimension of the feature representations output by the fc7 layer in the C3D network. Descriptors of trajectories are then assigned to their closest vocabulary word using Euclidean distance. The resulting histograms of visual word occurrences are the feature representations.

For the C3D network, the details of architecture are presented in the Table 1. The C3D network is pre-trained on the sports1M dataset. The training parameters are the same as in the paper [23]. Two C3D networks are trained in our proposed framework: spatial-C3D on the RGB frames and motion-C3D on the optical flow images. The dimensions of the features extracted by spatial-C3D and motion-C3D are all 4096. The final spatiotemporal features of sliced video clips are the concatenations of the features computed from the dense trajectories algorithm, spatial-C3D and motion-C3D. Therefore, the final dimension of the extracted spatiotemporal features is $3 \times 4096 = 12288$.

To integrate the LSTM model with the extracted spatiotemporal features, a fully connected layer is added ahead of the LSTM network. We adopt the structure of the LSTM network presented in the paper [5].

Network	fc6	fc7
spatial-C3D-LSTM	82.5%	83.0%
motion-C3D-LSTM	79.2%	80.4%

Table 2. Comparison of the recognition accuracy of the features output by the fc6 and fc7 layers in C3D model on the UCF101 dataset.

5.3. Parametric Study

In this section, we present an ablation study of a set of the parameters in our proposed spatiotemporal-LSTM framework on the UCF101 dataset.

First, a comparison of the performance of the features output by the fc6 and fc7 layers in C3D network are presented in the Table 2. The spatial-C3D-LSTM and motion-C3D-LSTM are constructed by directly connecting the LSTM model to the output features of spatial-C3D and motion-C3D respectively. As observed from the table, the features output by the fc7 layer of C3D network consistently outperform the features output by the fc6 layer.

Since the LSTM network can take sequences of input

with variant lengths, we evaluate the impact of the sequential length and step size of the input features on the LSTM model. The results are shown in the Table 3.

step	length	accuracy
1	8	82.8%
	16	**83.0%**
	32	82.6%
	64	82.4%
8	8	81.6%
	16	82.3%
	32	82.0%
	64	81.5%
16	8	81.3%
	16	81.7%
	32	81.3%
	64	81.1%

Table 3. Evaluations of the impact of the sequential lengths and step sizes of the inputs on the LSTM model. All results are reported by spatial-C3D-LSTM on the UCF101 dataset. The best result is singled out with the bold font.

As shown in Table 3, the best performance is achieved with the sequential length of 16 and the step size of 1. The reason may be that, since the clip length is 16, by setting the sequential length to 16 and the step size to 1, the LSTM model can learn more compact features for the sequence of input clips.

5.4. Results and Analysis

In this section, we present the results of our proposed spatiotemporal framework, and compare with the performance of the state-of-the-art algorithms. The results are shown in the Table 4.

Compared with the C3D method, our spatial-C3D-LSTM model has an improvement of 0.7% on the UCF101 dataset, and 1.3% on the HMDB51 dataset. By combining the spatio-C3D and motion-C3D, the (spatial+motion)-C3D-LSTM improves 2.3% and 3.9% on the UCF101 and HMDB51 datasets respectively, when compared with C3D method.

By integrating the dense trajectories with LSTM network, improvements of 5.9% and 2.0% can be achieved on the UCF101 and HMDB51 datasets respectively, when compared with DT+BOV method which only adopts the dense trajectories algorithm.

Our spatiotemporal-LSTM framework concatenates all the features output by spatial-C3D, motion-C3D and dense trajectories to form the final spatiotemporal feature representation for each video clip. And then a sequence of these feature representations is input to the network, which is comprised of a fully connected layer and LSTM network. As observed from the results, our spatiotemporal-LSTM framework can have improvements 3.1% and 4.3% on the UCF101 and HMDB51 datasets respectively, when compared with the C3D method.

Methods	UCF101	HMDB51
Deep networks[12]	65.4%	-
LRCN [5]	82.9%	-
LSTM on long clips[17]	88.6%	-
Two-stream network [20]	88.0%	59.4%
TDD+FV[26]	**90.3%**	**63.2%**
STIP+BOV[1]	43.9%	23.0%
iDT+FV[25]	85.9%	57.2%
DT+BOV[24]	73.4%	46.6%
DT+FV[1]	81.4%	54.8%
C3D[23]	82.3%	49.9%
spatial-C3D-LSTM	83.0%	51.2%
motion-C3D-LSTM	80.4%	49.4%
(spatial + motion)-C3D-LSTM	84.6%	53.8%
DT-LSTM	79.2%	48.6%
spatiotemporal-LSTM	**85.4%**	**55.2%**

Table 4. Action recognition results on UCF101 and HMDB51. The results obtained by our proposed framework and the state-of-the-art algorithm are singled out.

Among our listed results, TDD+FV[26] achieves the best results. This method replaces the traditional descriptors with the CNN learned features in the dense trajectories algorithm. Because of limited time, we have not tried TDD in our framework, which is part of the future work.

Based on the above discussion, improvements can be achieved by embedding the sequential information into the spatiotemporal features. Therefore, our proposed spatiotemporal-LSTM framework can provide more discriminative and robust features compared with the ones only contain spatiotemporal or sequential information.

6. Conclusion

We have proposed a novel video feature extraction framework, which embeds the sequential information to the extracted spatiotemporal features. The experimental results demonstrate that our framework can achieve more accurate results on the UCF101 and HMDB51 datasets, when compared with the baseline algorithms (C3D and dense trajectories). By integrating the sequential information with the spatiotemporal information, the resulting features are more robust and effective to represent videos for action recognition task.

7. Acknowledgement

This work was supported in part by NSF grants EFRI-1137172, IIP-1343402, and IIS-1400802.

References

[1] Z. Cai, L. Wang, X. Peng, and Y. Qiao. Multi-view super vector for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 596–603, 2014.

[2] B. Chakraborty, M. B. Holte, T. B. Moeslund, and J. Gonzàlez. Selective spatio-temporal interest points. *Computer Vision and Image Understanding*, 116(3):396–410, 2012.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[4] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 65–72. IEEE, 2005.

[5] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014.

[6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision–ECCV 2014*, pages 346–361. Springer, 2014.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):221–231, 2013.

[11] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*, 2014.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1725–1732. IEEE, 2014.

[13] 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.

[14] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[15] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *arXiv preprint arXiv:1503.08909*, 2015.

[18] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172, 2015.

[19] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[20] 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.

[21] 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.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[23] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3d: generic features for video analysis. *arXiv preprint arXiv:1412.0767*, 2014.

[24] 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.

[25] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3551–3558. IEEE, 2013.

[26] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. *arXiv preprint arXiv:1505.04868*, 2015.

[27] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Computer Vision–ECCV 2008*, pages 650–663. Springer, 2008.