T-C3D: Temporal Convolutional 3D
Network for Real-Time Action Recognition

Kun Liu,1 Wu Liu,1 Chuang Gan,2 Mingkui Tan,3 Huadong Ma1
1Beijing Key Lab of Intelligent Telecommunication Software and Multimedia,
Beijing University of Posts and Telecommunications, Beijing 100876, China;
2Tsinghua University, Beijing, China; 3South China University of Technology, Guangzhou, China
Email: {liu_kun, liuwu, mhd}@bupt.edu.cn; ganchuang1990@gmail.com; mingkuitan@scut.edu.cn

Abstract

Video-based action recognition with deep neural networks has shown remarkable progress. However, most of the existing approaches are too computationally expensive due to the complex network architecture. To address these problems, we propose a new real-time action recognition architecture, called Temporal Convolutional 3D Network (T-C3D), which learns video action representations in a hierarchical multi-granularity manner. Specifically, we combine a residual 3D convolutional neural network which captures complementary information on the appearance of a single frame and the motion between consecutive frames with a new temporal encoding method to explore the temporal dynamics of the whole video. Thus heavy calculations are avoided when doing the inference, which enables the method to be capable of real-time processing. On two challenging benchmark datasets, UCF101 and HMDB51, our method is significantly better than state-of-the-art real-time methods by over 5.4% in terms of accuracy and 2 times faster in terms of inference speed (969 frames per second), demonstrating comparable recognition performance to the state-of-the-art methods. The source code for the complete system as well as the pre-trained models are publicly available at https://github.com/tc3d.

1 Introduction

Video-based action recognition is to enable the computers to recognize the human actions automatically in real-world videos. It has attracted considerable attention from the academic community (Simonyan and Zisserman 2014; Tran et al. 2015) to industry applications such as video classification (Karpathy et al. 2014) and behavior analysis in public security systems (Wang and Schmid 2013). The task of human action recognition in videos, however, is still very challenging due to several reasons. First, the video is naturally an information-intensive media with a number of complexities, e.g., scale variations, cluttered background, viewpoint changes, camera motions, and so on. Second, unlike action recognition in a still image, video-based action recognition should have the ability of characterizing both short-term small motions and long-term temporal evolutions of appearances. Some actions can be reliably distinguished through the motion computed from consecutive frames (i.e., short-term motions), but there are also certain similar actions that require the overall features of long-term video (i.e., long-term motions), where the short-term information on short clips is almost the same.

To illustrate the above issues, we show some example videos from UCF101 dataset in Figure 1. As shown in Figure 1, “Playing Piano” and “Archery” can be easily recognized through the appearance information of a static frame or small motion between continual frames. However, sometimes short clips are not sufficient for distinguishing similar classes (high jump vs long jump). In this situation, the video-level representations must be taken into consideration. Therefore, it is important to exploit the complementary nature of the single static image, the short and long-term temporal evolution, and the video-level representations, which, however, is a great challenge in video processing. Last but not least, due to the highly complex nature of the videos, it often requires expensive computational cost to deal with a video. In practice, this is often the most critical bottleneck for video-based ac-

![Figure 1: Example videos from four classes of UCF101.](a) Playing Piano and Archery

(b) High Jump and Long Jump

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
Action recognition has been widely explored in the last decade. We briefly group previous works related to ours into two categories: 1) action recognition with hand-engineered features, and 2) CNNs for action recognition.

To depict the temporal motion in videos for action recognition, many works try to devise effective features. Early researches propose some video representations which are derived from the image domain and extended to measure the temporal dimension of 3D volumes, such as 3D Histogram of Gradient (HOG3D) (Klaser, Marszałek, and Schmid 2008) and 3D Scale-Invariant Feature Transform (SIFT-3D) (Scovanner, Ali, and Shah 2007). Besides, several works focus on designing local spatio-temporal features. In particular, Wang et al. (Wang and Schmid 2013) propose a state-of-the-art hand-crafted feature named Improved Dense Trajectories (IDT), which extracts several descriptors (HOG, HOF and MBH) and tracks them in a dense optical flow field. However, some features designed by human beings are not discriminative enough to model the video. Part of them are too computationally expensive to process at real-time despite the impressive performance.


As to the one-stream CNNs, networks always pay more attention to spatial information. Several works (Tran et al. 2015; 2017; Varol, Laptev, and Schmid 2017) employ 3D convolution operator with the input of a short clip to model the temporal motion in video. However, this kind of methods have not yet significantly outperformed the traditional methods for action recognition in video. This is partly due to lack the capacity to model long-term features utilizing 3D-CNN. The other reason might be underachieving large-scale video datasets comparable in size and variety to ImageNet. The appearance of large-scale and well-labeled video datasets, like Sports-1M (Karpathy et al. 2014) and Kinetics (Kay et al. 2017), bring the opportunities to promote researches in this area. Motivated by above observations, we extend 3D-CNN with temporal encoding framework to model the entire video character and pre-train the network on a large-scale and clean dataset to fully unleash the potential of 3D-CNN.

The two-stream architecture is proposed in (Simonyan and Zisserman 2014) where spatial net captures single RGB image appearance and temporal net depicts the motion among a short clip with the input of ten optical flow maps. This might be the first work demonstrates that the deep model is more accurate than hand-engineered features, such as dense trajectories-based representation. Recently several attempts (Girdhar et al. 2017; Wang et al. 2016) have been made to improve the two-stream network from different aspects. Very recently, Kar et al. (Kar et al. 2017) describe an adaptive temporal pooling method that learns to pool discriminative and informative frames and discards majority of the redundant and non-discriminative frames in the
video. The approach proposed in (Girdhar et al. 2017) provides an end-to-end trainable architecture for spatiotemporal video feature aggregation, where the inherent visual vocabularies (VLAD) are learned directly from the loss function. Despite good performance, all of two-stream based methods are too computationally expensive to meet the real-time requirement due to the heavy calculation of optical flows.

For real-time action recognition, Zhang et al. (Zhang et al. 2016) replace the optical flow with motion vectors to deploy the algorithm at real-time. This method transfers knowledge from high quality optical flow to motion vector encoding representation. While this work accelerates the speed of deep learning methods for action recognition, it is cumbersome owing to the calculation of optical flow during the training phase. Besides, the performance is not really superior. Compared with it, our approach totally avoids calculating the optical flow and only requires RGB frames to train the network. Meanwhile, our model can achieve a superior recognition performance and a faster speed.

## 3 Temporal 3D Convolutional Network

Figure 2 illustrates the framework of the proposed T-C3D network. Each input video is firstly divided into $S$ parts. Then several frames are selected from each part to make up a clip. Next, $S$ clips represented the entire video are fed into the 3D-CNNs respectively. The 3D-CNN extends the 2D-CNN at the temporal dimension, which is more suitable for capturing the three-dimensional data feature of video.

The 3D-CNNs on all clips share the same weights. Furthermore, the feature maps or class scores of different clips are fused by an aggregation function to yield segmental consensus, which is a video-level prediction. Compared with previous works, T-C3D optimizes and updates its parameter through the video-level score rather than the clip-level prediction. The process can be formulated as Equation 1:

$$Y_v = H(Q(F(C_1; W); F(C_2; W); \ldots; F(C_S; W)))$$  \hspace{1cm} (1)

in which $Y_v$ represents the final class score of the video $v$, $F(C_s; W)$ is the function describing the 3D-CNN with weights $W$, and yields feature map of clip $C_s$, such as last convolutional layer, fully connected layer, and probability of...
all action categories. The aggregation function $Q$ fuses the output from multiple clips to obtain a discriminative representation of the entire video. Based on this representation, the prediction function $H$ produces the probability values of each action category for the whole video. Specially, in our work, the output of $F$ is the last fully connected layer while $H$ is the widely used softmax function. Multiple alternatives of aggregation function $Q$ are exploited in the Section 4.4.

The differentiability of temporal encoding method allows T-C3D to be easily optimized using back propagation. Based on standard categorical cross-entropy loss, the final loss function regarding the segmental consensus $G = Q(F(C_1; W); F(C_2; W); \ldots; F(C_S; W))$ is formulated as

$$L(y, G) = \frac{1}{N} \sum_{i=1}^{N} y_i (G_i - \log \sum_{j=1}^{N} \exp G_j),$$

where $N$ is the number of action classes and $y_i$ is the ground truth label concerning class $i$. $S$ is a hyperparameter and we perform hyperparameter search to evaluate the effects of the value of $S$ in next section. More importantly, the aggregation function $Q$ is of significant importance because it not only integrates the clips’ information into video-level feature but also determines the differentiability of the whole pipeline. In our experiments, we extensively study multiple differentiable aggregation function alternatives, such as mean pooling, max pooling, and attention pooling. Differentiable aggregation functions allow us to realize the multiple clips to jointly update the network weights $W$ with standard back-propagation algorithms. In the back-propagation process, the gradients of network weights $W$ concerning to the loss value $L$ can be formed as

$$\frac{\partial L(y, G)}{\partial W} = \frac{\partial L}{\partial W} \sum_{s=1}^{S} \frac{\partial Q}{\partial F(C_s)} \frac{\partial F(C_s)}{\partial W},$$

where $S$ is number of clips adopted by T-C3D. In Equation 3, the weights are optimized through the segmental consensus $G$ derived from all clip-level prediction. Updated parameters in this manner, T-C3D learns network weights from the overall video rather than a single clip. Next, we will introduce the implementation of the model in detail.

### 3.2 Video Components Generation

Different from the still images, videos are dynamic and with varying sequences. To exploit the good manners to model the overall video, we first uniformly divide the video into several parts in the temporal dimension. Then a clip is constituted by sampling several frames from each part with two popular sample schemes. The first scheme uniformly divides the video snippet generated in the previous step into a certain number of fragments and randomly selects one frame from each fragment to constitute the final clip. The second method randomly chooses a certain number of consecutive frames from the snippet to construct the final clip. In essence, the first sampling approach randomly selects non-consecutive frames distributed evenly throughout the video to represent the whole video. The second method uniformly chooses $S$ clips from the entire video and each clip consists of a certain number of sequential frames. In next section, we give the comparison on classification accuracy of these two different down-sampling methods.

### 3.3 3D Convolutional Neural Network

Different from 2D-CNN’s outstanding achievement on various visual tasks in still images, 3D-CNN is likely to fit for the videos which can be considered as the expansion of images in the temporal dimension. Convolutional 3D Network (C3D) (Tran et al. 2015) is one of the typical works that employ the 3D-CNN to extract both the spatial information and temporal cues with the input of sixteen raw RGB frames. However, sixteen raw RGB frames cannot model the long-term information. Then, Long-term Temporal Convolutions (LTC) (Varol, Laptev, and Schmid 2017) improve the C3D by feeding 3D-CNN with longer continuous RGB frame sequences and corresponding optical flow maps, ranging from 20 to 100 frames. All above works demonstrate that 3D-CNN is a promising direction for video-based action recognition. In our work, we extend the aforementioned 3D-CNN works from the following aspects.

Firstly, inspired by the amazing image classification accuracy obtained by the deep residual CNN, we adopt a deeper 3D-CNN network with residual block. More specially, we employ the 3D ResNet with 17 convolutional layers and one fully connected layer according to previous work on ConvNet architecture search (Tran et al. 2017). Experiments demonstrate that the deeper residual 3D-CNN can extract richer and stronger spatio-temporal feature hierarchies from the given multiple frames.

Secondly, pre-training the parameters of CNN (Gan et al. 2015; 2016b; Liu, Liu, and Ma 2017; Ma 2017) on a large-scale dataset has been proven greatly crucial for various visual tasks, e.g., object detection, image classification, semantic segmentation, and so on. For 3D-CNN, previous works such as LTC have shown that 3D models pre-trained on Sports-1M achieve higher classification accuracy than the models trained from scratch. In this paper, we first follow the strategy introduced in C3D and pre-train our model on Sports-1M. Although Sports-1M has over than one million videos, it contains amounts of noise since it is not manually labeled. Very recently, Kay (Kay et al. 2017) et.al propose a large-scale and clean dataset, called Kinetics, which covers 400 human action categories with at least 400 video clips for each action. To activate the neuron in the 3D-CNN as much as possible, we make efforts to train the 3D-CNN on Kinetics with the temporal encoding method. Experiments show that pre-training on Kinetics significantly boosts the performance.

### 3.4 Aggregation Functions

As mentioned above, aggregation functions is a very curial component in the T-C3D framework. In this subsection, we provide a detailed description and insightful analysis of four aggregation functions, including average pooling, max pooling, weighted pooling, and attention pooling.

**Average Pooling.** In this aggregation function, we adopt average pooling to fuse the 3D-CNN output of the sampled
clips, i.e., \( g_i = \frac{1}{S} \sum_{s=1}^{S} F^i_s \), where \( F^i_s \) is the \( i^{th} \) element of \( F_s = F(C_s, W) \). The basic assumption of average pooling is to utilize the activations of all clips for action recognition and employ their mean responses as the overall video prediction. From this perspective, average pooling is able to jointly depict sequences of clips and obtain the visual feature from the entire video. However, some videos may contain noisy sequences that are irrelevant with the actions, in this case, averaging over these noisy clips cannot accurately model the action character, possibly leading to degradation of the recognition performance.

**Maximum Pooling.** Another widely used aggregation method is maximum pooling, where we perform a maximum operation over these clip-level outputs, i.e., \( g_i = \max_{s \in \{1, 2, \ldots, S\}} F^i_s \). The basic intuition of max pooling is to select the most discriminative clip for every action category and represent the whole video with this strongest response. Intuitively, it focuses on a single clip without taking the activations of other clips into consideration. In some cases, a single clip is not discriminative enough to capture the entire video information. To some degree, T-C3D degrades to the previous works which train the network with one clip per video when employing the max pooling. Therefore, this aggregating function drives the T-C3D to represent the entire video just with single clip, which violates the T-C3D’s assumption of modeling overall video.

**Weighted Pooling.** The goal of this aggregation function is to produce a set of linear weights to perform an element-wise weighted linear fusion among the outputs of each clip. Specifically, the aggregation function is defined as \( \sum_{s=1}^{S} \omega_s F^i_s \), where \( \omega_s \) is the weight for the \( s^{th} \) clip. In our experiments, both the network weights \( W \) and the fusion weights \( \omega \) are optimized simultaneously. We introduce this aggregation function based on the fact that action always consists of several phases and these different phases may have different influences in identifying action classes. This function combines the merits of both maximum pooling and evenly pooling, having ability of jointly depicting sequences of relevant clips while decreasing the bad effects of noisy clips. Specially, we adopt a convolutional layer with the kernel of \( S \times 1 \) to implement the function.

**Attention Pooling.** This aggregation function has the same goal as the weighted pooling method. It borrows the memory attention mechanism of a kind of end-to-end trainable memory network (Sukhbaatar et al. 2015) for our feature aggregation. The intuition therein is to employ a neural model to read external memories through a differentiable addressing/attention scheme. In our work, we consider the outputs of each clip as the memory and cast feature weighting as a memory addressing procedure. Formally, let \( F^i \) be the 3D-CNN feature map of \( s^{th} \) clip, then the aggregation module selects them with a kernel \( q \) via dot product, producing a sequence of corresponding weights \( e_s \). Then a softmax function operates on them to generate positive parameters \( \omega_s \) with \( \sum_{s=1}^{S} \omega_s = 1 \). These two steps can be formulated as the Equation 4 and Equation 5 respectively.

\[
\begin{align*}
    e_s &= q^T F^i_s, \\
    \omega_s &= \frac{\exp e_s}{\sum_{j=1}^{S} \exp e_j}.
\end{align*}
\]

Obviously, this aggregation module essentially chooses one point inside of the convex hull spanned by all the feature maps.

4 Experiments

In this section, we first describe the benchmark datasets and implementation details of the proposed framework: T-C3D. Then, we compare the performance and speed of our method with the state-of-the-art methods. After this, we explore various alternatives for learning T-C3D networks, such as generating snippets strategy, aggregation function, and weight initialization scheme.

4.1 Datasets and Evaluation Protocol

We empirically evaluate our T-C3D approach on the two public benchmark datasets for action recognition: UCF101 (Soomro, Zamir, and Shah 2012) and HMDB51 (Kuehne et al. 2011). The UCF101 dataset is a widely used benchmark which consists of 101 action categories with 13,320 videos in about 27 hours. The majority of video clips in UCF101 have the \( 320 \times 240 \) pixels spatial resolution and 25 frames per second (FPS) frame rate. Each action class has at least 100 video samples. HMDB51 dataset is a large collection of realistic videos from various sources, such as web videos and movies. This dataset is composed of 51 action categories with 6,766 video clips in all. A split in HMDB51 includes 3,570 training and 1,530 test instances, while each split in UCF101 contains around 9,500 training and 1,500 test video samples. For both datasets, we adopt the three standard training/testing splits provided in original works as the evaluation scheme and report the mean accuracy over these three splits. Following (Li et al. 2016), the exploration study for training T-C3D is only conducted on the first split of UCF101. As for speed evaluation, we adopt FPS as metric and conduct experiments on a CPU (E5-2640 v3) and a K40 GPU.

4.2 Implementation Details

For both datasets, each video is sampled to generate the clips to feed the network. Following (Tran et al. 2017), every clip contains eight frames and each frame in the clip is resized to \( 128 \times 171 \) from the original spatial resolution. Then every frame subtract the mean value of the training data to center the input data. Moreover, to reduce the effect of severe over-fitting and learn powerful features from T-C3D, we adopt two types of data augmentation techniques. Firstly, we horizontally flip frames with 50 probability. Secondly, we extend the random crop with scale jittering and aspect ratio jittering techniques that are commonly used in still image classification. Specially, we randomly select the width and height of the cropped region on three scales 1, 0.875, and 0.75, generating more training instances. Then all the cropped regions
are resized into $112 \times 112$. Namely, the network employs an $8 \times 112 \times 112$ input, the largest that can fit within GPU memory limits and maintain a large enough mini-batch.

The network parameters are learned in an end-to-end fashion with the mini-batch stochastic gradient descent algorithm, where the momentum is set to 0.9 and the batch size is set to 8. The pre-trained models on Sport-1M and Kinetics are utilized to initialize network weights. We randomly initialize the last fully connected layer and add a dropout layer after the global pooling layer with high dropout ratio (set to 0.8 in experiments) to prevent over-fitting. On UCF101, the initial learning rate is 0.005 and decreased to its 1/10 every 8,000 iterations. The whole optimization procedure is stopped at 20,000 iterations. For HMDB51, the training scheme is the same as that of UCF101, except that the iteration numbers are adjusted according to the number of training instances.

During the testing phase, to balance the speed and classification accuracy, we investigate both multi-scale testing strategy and only a single center crop to predict the action category. More specifically, the feed forward process of CNN is performed by GPUs when evaluating the speed.

4.3 Comparison with The-State-of-The-Art Methods

Table 1 shows the comparison of our architecture with current state-of-the-art methods including:

1) **Hand-engineered feature methods**: IDT encoded with Fisher Vector (Wang and Schmid 2013), DT encoded with Multi-view super vector (MVSV) (Cai et al. 2014), and Motion Vector (MV) encoded with Fisher Vector;

2) **One-stream methods**: C3D (Tran et al. 2015), Res3D (Tran et al. 2017), Slow Fusion (Karpathy et al. 2014), and Temporal Segment Network (TSN) with input RGB (Wang et al. 2016);

3) **Two-stream methods**: original two-stream method (Simonyan and Zisserman 2014) with shallow and deep CNN models, Two-stream+LSTM (Yue-Hei Ng et al. 2015), LTC (Varol, Laptev, and Schmid 2017), AdaScan (Kar et al. 2017), ActionVLAD (Girdhar et al. 2017), TDD+MV (Wang, Qiao, and Tang 2015) and TSN (Wang et al. 2016), and Enhanced MV (Zhang et al. 2016).

Compared with hand-crafted features-based methods, T-C3D outperforms the most discriminative hand-engineered feature (IDT) encoded with the robust encoder (FV). Moreover, it achieves the best accuracy among one-stream methods that use only RGB input on both datasets. TSN (RGB) and Slow Fusion belong to 2D-CNN based approaches. They are both inferior to T-C3D since 2D-CNN might be unsuitable for extracting the spatial-temporal information of videos. Although TSN can achieve more impressive performance of 87.3% at real-time on split 1 of UCF101 when using RGB and RGB Difference, it still obtain lower accuracy and the higher computational cost compared with our method. Finally, T-C3D attains higher accuracy than the two-stream methods with very deep CNN architectures. More specially, it achieves competitive performance to the state-of-the-art methods or even outperforms some recently proposed extended works of two-stream framework when per-training on the Kinetics dataset. Despite superior performance on both datasets, TSN is computational expensive (5 fps) and far from the real-time requirements. Note that the improved algorithm of two-stream also can be applied to further enhance T-C3D. For speed evaluation, it should be noticed that T-C3D beats all the real-time models on the accuracy by a large margin while achieving the fastest speed. Our fast version can achieve the 969 FPS.

4.4 Exploration Study

In this subsection, we conduct exploration study of the T-C3D from the following four aspects: 1) sampling methods for generating snippets, 2) the number of snippets sampled from a video, 3) aggregation functions, and 4) parameter initialization schemes. In this empirical study, we conduct all experiments on the split 1 of UCF101 dataset with the proposed framework.

**Evaluation on sampling methods.** We investigate the effects of two sampling methods described in Section 3.2. Table 2 summarizes the results. We can observe that sampling consecutive frames is more suitable for learning the 3D-CNN parameters than sampling frames evenly from the whole video. The latter methods can get the whole video information with non-consecutive frames, the two adjacent

<table>
<thead>
<tr>
<th>Method</th>
<th>UCF101</th>
<th>HMDB51</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand-crafted Feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDT+FV</td>
<td>85.9</td>
<td>57.2</td>
<td>2</td>
</tr>
<tr>
<td>DT+MVSVC</td>
<td>83.5</td>
<td>55.9</td>
<td>N/A</td>
</tr>
<tr>
<td>MV+FV</td>
<td>78.5</td>
<td>N/A</td>
<td>133</td>
</tr>
<tr>
<td>One-stream (RGB)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3D</td>
<td>82.3</td>
<td>51.6</td>
<td>&lt;14</td>
</tr>
<tr>
<td>C3D(3nets)</td>
<td>85.2</td>
<td>N/A</td>
<td>314</td>
</tr>
<tr>
<td>Slow Fusion</td>
<td>65.8</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Res3D</td>
<td>85.8</td>
<td>54.9</td>
<td>N/A</td>
</tr>
<tr>
<td>TSN(RGB)</td>
<td>85.1</td>
<td>51.0</td>
<td>N/A</td>
</tr>
<tr>
<td>Two-stream (Based)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSN(VGG-M)</td>
<td>88.0</td>
<td>59.4</td>
<td>14</td>
</tr>
<tr>
<td>TSN(Resnet50)</td>
<td>91.7</td>
<td>61.2</td>
<td>&lt;14</td>
</tr>
<tr>
<td>TSN+LSTM</td>
<td>88.6</td>
<td>N/A</td>
<td>&lt;14</td>
</tr>
<tr>
<td>LTC</td>
<td>91.7</td>
<td>64.8</td>
<td>&lt;14</td>
</tr>
<tr>
<td>AdaScan</td>
<td>89.4</td>
<td>54.9</td>
<td>&lt;14</td>
</tr>
<tr>
<td>ActionVLAD</td>
<td>92.7</td>
<td>66.9</td>
<td>&lt;14</td>
</tr>
<tr>
<td>TDD+MV</td>
<td>90.3</td>
<td>63.2</td>
<td>&lt;14</td>
</tr>
<tr>
<td>TSN</td>
<td>94.2</td>
<td>69.4</td>
<td>5</td>
</tr>
<tr>
<td>Enhanced MV</td>
<td>86.4</td>
<td>N/A</td>
<td>390</td>
</tr>
<tr>
<td>T-C3D</td>
<td>Ours(Sports1M)</td>
<td>89.4</td>
<td>58.6</td>
</tr>
<tr>
<td></td>
<td>Ours(Kinetics)</td>
<td>92.5</td>
<td>62.4</td>
</tr>
<tr>
<td></td>
<td>Ours(Fast)</td>
<td>91.8</td>
<td>62.8</td>
</tr>
</tbody>
</table>

Table 2: Comparison of sampling methods on split 1 of UCF101 dataset.

<table>
<thead>
<tr>
<th>Sampling Methods</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consecutive Sampling</td>
<td>89.5</td>
</tr>
<tr>
<td>Non-Consecutive Sampling</td>
<td>89.2</td>
</tr>
</tbody>
</table>
frames sampled by this method might last across a long period of time. The 3D-CNN possibly lacks the capability of capturing the large motion features.

Evaluation on aggregation functions. The results of different aggregation functions are summarized in Table 3. The attention aggregation function achieves the best performance, and average pooling obtains quite similar performance. This conclusion possibly suggests that on clean datasets with fewer scale variations and cluttered background, the simple aggregation module can result in better recognition accuracies. In this sense, we exploit average pooling for the aggregation function in later experiments.

Evaluation on snippets number. We study extensively the impact of the number of clips sampled from one video. Figure 3 obviously shows that increasing the number of clips can result in better performance. Note that the model weights are updated without the temporal encoding method when the number of clips is 1. It is easily concluded that temporal encoding method is benefit for increasing the classification accuracy. Besides, when the number of clips increases from 4 to 6 or 8, the performance saturates. Given the trade-off between training time and accuracy, we set the number of clips to 3 as the default setting.

Evaluation on parameter initialization. In Table 4, we find that pre-training the weight parameters of T-C3D on a large-scale dataset can get better performance. Sports-1M with the ground truth is not well labelled has less accuracy, although it have more training samples and more categories. Thus, the quality seems to be more important than quantity when choosing the datasets to initialize the parameter.

Evaluation on the balance between speed and accuracy. We also make efforts to find the trade-off between the performance and the speed. According to Table 5, feeding forward the network with 5 crops and mirror (multi-scale) brings a slight improvement on performance but a serious deceleration. Aggregating all clips of a video is a good choice which balances the accuracy and computational cost. Moreover, sampling $S$ clips per video also obtains an impressive performance in an extremely fast speed. $S$ is number of clips adopted by T-C3D.

5 Conclusion

We present T-C3D, an end-to-end trainable framework to take advantage of the temporal encoding method and deep 3D-CNN to learn the overall temporal information of a video. By feeding consecutive frames to network, T-C3D extracts the complementary information on spatial information from a single image and motion features between sequential frames. T-C3D captures the overall temporal dynamics of the whole video through the temporal encoding method. Owing to the capacity to model multiple granularity features of videos, our approach achieves competitive performance to the state-of-the-art methods. Furthermore, T-C3D does not require heavy computational process so that it processes videos in a speed of very faster than real-time, which makes it possible to deploy action recognition algorithms on mobile devices. In the future, we will extend T-C3D for online processing where the system performs recognition as the frames are received instead of presenting the entire video.

6 Acknowledgement

This work is partially supported by the Funds for International Cooperation and Exchange of the National Natural Science Foundation of China (No. 61720106007), the National Natural Science Foundation of China (No. 61602049), the NSFC-Guangdong Joint Fund (U1501254), and the Beijing Training Project for the Leading Talents in S&T (ljrc 201502).
References


Gan, C.; Yao, T.; Yang, K.; Yang, Y.; and Mei, T. 2016b. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. In Computer Vision and Pattern Recognition, 923–932.


