Video You Only Look Once: Overall Temporal Convolutions for Action Recognition

Longlong Jing[a], Xiaodong Yang[b], Yingli Tian[a,c,*]

[a] *The Graduate Center, City University of New York*
[b] *NVIDIA RESEARCH*
[c] *The City College of New York, City University of New York*

Abstract

In this paper, we propose an efficient and straightforward approach, video you only look once (VideoYOLO), to capture the overall temporal dynamics from an entire video in a single process for action recognition. It remains an open question for action recognition on how to deal with the temporal dimension in videos. Existing methods subdivide a whole video into either individual frames or short clips and consequently have to process these fractions multiple times. A post process is then used to aggregate the partial dynamic cues to implicitly infer the whole temporal information. On the contrary, in VideoYOLO, we first generate a proxy video by selecting a subset of frames to roughly reserve the overall temporal dynamics presented in the original video. A 3D convolutional neural network (3D-CNN) is employed to learn the overall temporal characteristics from the proxy video and predict action category in a single process. Our proposed method is extremely fast. VideoYOLO-32 is able to process 36 videos per second that is 10 times and 7 times faster than prior 2D-CNN (Two-stream [1]) and 3D-CNN (C3D [2]) based models, respectively, while still achieves superior or comparable classification accuracies on the benchmark datasets, UCF101 and HMDB51.

Keywords: Video Understanding, Video Classification, Action Recognition, Convolutional Neural Network

1. Introduction

With more videos flourishing on the Internet, video-based applications such as automatic categorization, searching, indexing, and retrieval of videos have drawn significant attention from the multimedia community. As one of the fundamental tasks for video analytics, action recognition provides the crucial visual cues and is a field of increasing importance field in vision research. Unlike the image-based applications such as image classification, object

*Corresponding author
Email addresses: ljing@gradcenter.cuny.edu (Longlong Jing), xiaodongy@nvidia.com (Xiaodong Yang), ytian@ccny.cuny.edu (Yingli Tian)

detection, and semantic segmentation which depend only on the appearance cues, both static appearance, and dynamic motion information are required for understanding videos [3, 4, 5]. The static appearance is the color and texture of each individual frame while the dynamic motion is the temporal transformation of appearance between frames.

Inspired by the successful application of CNNs in images [6, 7, 8, 9], a number of frameworks [1, 10, 11, 12, 2, 13, 14, 15] have been proposed to model the appearance and temporal information for videos. Karparthy *et al.* proposed to use a buffer of video frames as the input to 2D-CNNs [16]. Simonyan and Zisserman proposed the two-stream networks to explicitly fuse temporal information from optical flow images and appearance information from RGB frames [1]. Xu *et al.* proposed the 3D-CNN to learn the spatial and temporal information by performing 3D convolutions on multiple consecutive frames [11].

Although significant progress has been made in learning temporal information from videos by CNNs, it has to hinge on a post-process to implicitly capture the overall temporal information by aggregating the partial temporal cues from multiple segments of a video [1, 16, 2, 11, 15]. CNN-based algorithms typically predict action category of the videos using the softmax scores [13, 14, 1, 15], or, alternatively, the fully connected layer as a feature representation [12, 2]. These methods focus on short-term temporal information as the features are learned from individual frames or short video clips. This is suboptimal for action recognition since the overall temporal information that span in a whole video is essential to recognize long-lasting activities.

To capture the long-term temporal cues in videos, recurrent neural networks (RNN) were applied to model videos as an ordered sequence of frames. Compared to the vanilla RNN [17], the long short-term memory (LSTM) uses memory cells to store, modify, and access internal state, allowing it to better utilize the long-term temporal context [18]. Based on this property, LSTM has been widely used to model the frame sequences [14, 13, 19]. For example, Donahue *et al.* divided each video into short clips and applied LSTM to learn the temporal dependencies from short clips [14]. Ng *et al.* employed LSTM to model the temporal information from frame sequences, and the final result was obtained from the softmax score of every frame [13]. While LSTM-based methods can learn the overall temporal information, they need to sequentially apply CNN and LSTM on every single frame or short clip, which largely deteriorates the computational efficiency.

To efficiently learn the overall temporal information of a video, the model should be able to process all the frames in one pass. However, one major challenge is to handle the varied video lengths which may range from seconds to minutes. In most existing methods, every video is divided into short clips (usually 16 frames), then the model learns temporal information from these short clips [2, 14]. 3D and LRCN divided each video into 16-frame clips. However, in a video at 30 frames per second (FPS), 16 frames only last for about 0.5 seconds. Gül *et al.* demonstrated that this kind of short clips is too short to identify actions such as *Play Yoyo*, *Floor Gymnastics*, *Javelin Throw*, etc. [12]. Therefore, they divided a video into longer clips, up to 100 consecutive frames, and obtained improved results.

Video frames are highly redundant and some consecutive frames can be almost identical, as shown in the first 8 frames of a *Playing Golf* video in Fig. 1(a). Existing methods feed their networks in this way so that have to employ a post-processing through feature

(a) The first 8 frames of a playing golf video

(b) Randomly sampled 8 frames from the whole video

Figure 1: (a) The first 8 consecutive images from a playing golf video. There is no observable difference among these images. (b) The 8 randomly sampled frames from the whole video. With the same number of images, frames sampled from the whole video provide more temporal dynamics than the short-term video clips composed with consecutive frames.

or score fusion to aggregate the partial temporal information from individual frames or short clips. On the other hand, the 8 evenly sampled frames of this video in Fig. 1(b) can reasonably well capture the overall temporal dynamics. Compared to the original video frames, the temporally downsampled frames provide more dynamic cues with the same amount of images. Based on this observation, we propose an efficient and straightforward approach—VideoYOLO—to model the overall temporal information from a whole video at one process. In VideoYOLO, we first generate a proxy video by selecting a subset of frames and then employ 3D-CNN to learn the appearance and temporal information from the proxy video in only one pass. This approach is generic to a variety of video applications such as action similarity labeling, scene and object recognition in videos, and real-time video classification where the overall temporal information is needed [20, 21, 22, 23, 24, 25].

We evaluate the proposed method on two benchmark datasets for action recognition: UCF101 [26] and HMDB51 [27]. In the experiments, the VideoYOLO trained with a few frames per video achieves comparable accuracy as the state-of-the-art methods that use all frames while our approach is much more efficient.

2. Related Work

With the great progress in image recognition and a variety of video datasets such as UCF101 [26], HMDBD51 [27], ActivityNet [28], Sport-1M [16], and Youtube-8M [29], there has been an increasing interest in applying deep learning methods in video understanding. Inspired by the deep networks in image domain [6, 7, 8, 9], various deep learning based methods for video classification have been proposed [11, 1, 14, 13, 12, 2, 19, 16]. According to the network architectures, these methods fall into three categories: 2D-CNN, 3D-CNN, and RNN.

Karpathy *et al.* employed 2D-CNN to classify videos based on the stacked raw RGB frames [16]. In order to fuse motion and appearance information, Simonyan and Zisserman incorporated two CNNs to learn appearance information from raw RGB frames and temporal information from optical flow [1]. Wang *et al.* employed 2D-CNN to capture the temporal information from different segments of a video, then late fusion is applied to the features extracted from these segments [15]. Ji *et al.* took 3D-CNN to learn both the appearance and temporal information simultaneously from multiple consecutive frames [11]. This method was later extended by C3D [2]. LTC proposed in [12] fed 3D-CNN with longer frame sequence and achieved improved performance. RNN was employed in [14, 13, 19] to learn the long-term temporal cues from long sequences. Donahue *et al.* [14] employed VGG [8] to extract appearance features from continuous frames which are then used to train RNN to learn the temporal connections between frames. Ng *et al.* [13] coupled LSTM with AlexNet and GoogLeNet for action recognition. Instead of trying to learn spatiotemporal features over short time periods, they can learn how to integrate temporal dynamics over time. Jeff *et al.* also employed LSTM to learn temporal information from videos but had to divide videos into short clips, and the final result was the average of softmax score from every clip [14].

Among these methods, C3D [2], LTC [12] and TSN [15] are most close to our work. C3D and LTC are 3D-CNN based networks, while TSN is a 2D-CNN based network. In C3D, each video is subdivided into a set of 16-frame clips and the network is trained on these short clips. Schmid *et al.* argued that the video representations typically learned at the level of a few video frames failing to model actions at their full temporal extent. Thus they trained the network to learn video representations using neural networks with long-term temporal convolutions (LTC), and demonstrate that LTC models with increased temporal extents can improve the accuracy of action recognition. Wang *et al.* proposed to combine a sparse temporal sampling strategy and video-level supervision to select the effective frames from a video, then late fusion is applied on the features extracted by 2D-CNN from these segments. However, all of the methods need to look the video for multiple times, once for each clip, and apply CNN on every clip and then aggregate the scores from them. This kind of pipeline, however, largely decreases the processing efficiency.

Joseph *et al.* proposed YOLO for object detection in images by using a single network to directly predict object bounding boxes and class probabilities directly from full images in one evaluation [30]. It was then extended to YOLO9000 which is a real-time object detection system that can detect over 9000 object categories [31]. Motivated by the name of YOLO, we propose our approach, VideoYOLO, for action recognition from videos which can evaluate every video in one process. By avoiding to apply CNN multiple times and post-process to aggregate partial temporal information, our method is an order of magnitude faster compared to the state-of-the-art 2D-CNN [16, 1] and 3D-CNN [2, 12] based models.

3. The Proposed Method

We compare the pipelines of prior methods and our approach in Fig. 2. As shown in Fig. 2(a), the computing process of most existing methods [17, 1, 11, 2, 13, 14] typically divide a video into a set of short clips and obtain prediction score for each clip by CNN or

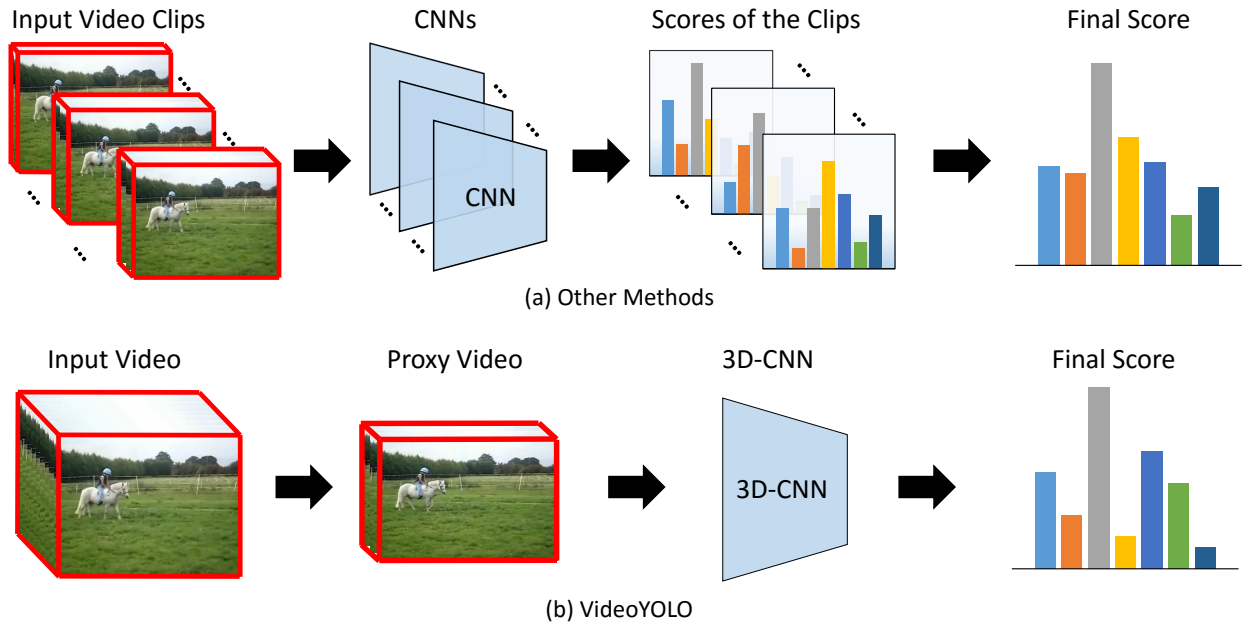

Figure 2: The comparison of other frameworks and ours. (a) The pipeline of other frameworks. Usually, every video is divided into multiple short clips and then CNN or RNN is applied on these short clips one by one. Finally, the prediction scores of these clips are weighted to form the final prediction score. (b) The pipeline of our proposed VideoYOLO framework. Instead of dividing every video into short clips, every video is sampled into a proxy video which is a constant length of frames, then 3D-CNN is employed to process every proxy video at once. Therefore, by discarding the majority frames, VideoYOLO is able to process every video in one pass.

RNN. The final classification score for the entire video is obtained by weighting the scores of all the clips. Therefore, these methods have to apply CNN or RNN in multiple clips which significantly degrade the efficiency. In contrast, as shown in Fig. 2(b), through selecting a subset of frames as a proxy video to roughly preserve the overall temporal information, VideoYOLO employs 3D-CNN to capture the overall temporal information from this proxy video at once. With only one look at the proxy video, VideoYOLO can identify the action in the video and greatly speeds up the classification process.

3.1. Generating Proxy Videos

Videos are dynamic and with varied sequences. The speed of actions varies in the temporal dimension. Even for the same action, the speed may vary when performed by different subjects or at a different time. As illustrated in Fig. 1(b), with only 8 frames sampled from a playing golf video, we can still recognize this action. Thus, we propose to sample a subset of frames as the proxy video from the original whole video to represent the overall temporal dynamics for action recognition.

We aim to build a framework to identify the action from the proxy video in one process without prior knowledge of the video. Based on the various action dynamics in videos, we propose to generate a proxy video in two ways: evenly sampling and randomly sampling as

shown in Eqs. (1) and (2). Assume the length of a video is N and we sample T frames from the video:

$$S_i = 1 + \lfloor N/T \rfloor * i, \tag{1}$$

$$S_i = random(\lfloor N/T \rfloor) + \lfloor N/T \rfloor * i, \tag{2}$$

where N is the length of the video, T is the number of the sampled frames from the video, S_i is the ith sampled frame, $random(N/T)$ generates one random number ranging between 0 and $\lfloor N/T \rfloor$ for every i. For randomly sampling, every video is uniformly divided into T intervals and one frame is randomly sampled from every interval. If a video is too short, we pad the video with the last frame to the length of T. The proxy video generated by the two sampling schemes makes it possible to process every video in one process without any prior knowledge about the video.

After a video is sampled into a constant length, we employ 3D-CNN to process the proxy video in one process. In our experiments, we evaluate the effects of different lengths of proxy videos by designing different types of networks. For example, our 32-frame network processes a video in one pass when sampling the video into a 32-frame proxy video.

3.2. 3D Convolutional Neural Network

3D-CNN was first proposed in [11] for action recognition and was later on improved by the 11-layer C3D network in [2]. 3D-CNN-based models have been widely applied in a number of video-based applications. Compared to 2D-CNN, 3D-CNN simultaneously extracts both spatial and temporal features from multiple frames. The input of C3D is 16 consecutive raw RGB frames where the appearance and temporal cues from the 16-frame clips are extracted. However, for most actions, 16 consecutive frames are insufficient to represent the whole action. So based on C3D, LTC [12] proposed to train 3D-CNN with longer raw RGB frame sequences and corresponding optical flow images, ranging from 20 to 100 frames. The classification accuracy of LTC outperforms C3D by a large margin by using longer clips. The performance of these two networks demonstrates the advantages of 3D-CNN over 2D-CNN for video classification.

Based on the two networks, we design an 11-layer 3D-CNN model for VideoYOLO which includes 8 convolution layers, 2 fully connected layers, and 1 softmax layer. We train multiple networks to evaluate the effect of the different lengths of the input frames. In other methods such as LTC, a network handling longer input frames usually involves more parameters than a network operating on shorter input sequences. However, in our model, all networks have the same number of parameters which makes a fair comparison of the performance among different lengths of the proxy videos.

3.3. Architecture of VideoYOLO

To investigate the relation of the performance and the temporal resolution of proxy videos, we train 8 types of 3D-CNN. Each network is able to process the proxy video in one process. These networks are referred as 'VideoYOLO-T' where T is the length of the proxy video (the input frames for that network). Specifically, 'VideoYOLO-96' represents

Table 1: The configurations of 8 VideoYOLO networks are listed in every column. For space economy, we use 'VYOLO' to represent 'VideoYOLO' in this table. 'VideoYOLO-T' can handle T frames at once. After sampling each video into T-frame proxy video, VideoYOLO-T network can process these T-frame proxy video at once. These networks have the same convolutional layers, fully connected layers, and the only difference is the max pooling layer.

Networks	VYOLO-4	VYOLO-8	VYOLO-16	VYOLO-32	VYOLO-48	VYOLO-64	VYOLO-80	VYOLO-96
Convolution	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64
Maxpool	$1\times2\times2$	$1\times2\times2$	$1\times2\times2$	$1\times2\times2$	$1\times2\times2$	$1\times2\times2$	$1\times2\times2$	$1\times2\times2$
Convolution	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128
Maxpool	**1×2×2**	**1×2×2**	$2\times2\times2$	$2\times2\times2$	$2\times2\times2$	$2\times2\times2$	**3×2×2**	**3×2×2**
Convolution	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
Convolution	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
Maxpool	**1×2×2**	$2\times2\times2$	$2\times2\times2$	$2\times2\times2$	$2\times2\times2$	$2\times2\times2$	**3×2×2**	**4×2×2**
Convolution	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
Convolution	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
Maxpool	$2\times2\times2$	$2\times2\times2$	$2\times2\times2$	$2\times2\times2$	**3×2×2**	**4×2×2**	**3×2×2**	**4×2×2**
Convolution	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
Convolution	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
Maxpool	$2\times2\times2$	$2\times2\times2$	$2\times2\times2$	**4×2×2**	**4×2×2**	**4×2×2**	**3×2×2**	$2\times2\times2$
fc6-4096								
fc7-4096								
Softmax								

the network whose input is 96-frame proxy video. The architectures of these networks are shown in Table 1 with one network per column.

All networks contain 8 convolutional layers, 2 fully connected layers and 1 softmax layer. All the kernels have the size of $3\times3\times3$, the convolution stride is fixed to 1 pixel. In order to preserve the resolution of feature maps after convolution, the paddings for both spatial and temporal dimensions are set to 1 pixel. In C3D, apart from the first layer, the size of all the max pooling is $2\times2\times2$ and with a stride of 2. However, if we keep the size of max pooling constant as C3D [2] in our networks, the number of parameters will grow tremendously as the number of frames increase. In LTC [12] the number of network parameters increases as more frames are utilized. To compensate for the growth in network parameters, low spatial resolution images are used to train the networks that have longer video clip in LTC. However, the low spatial resolution may result a decrease on the classification performance. In VideoYOLO we take the advantage of the max pooling layer to make the number of network parameters constant among all networks.

The pooling function replaces the output of the network at a certain location with a statistic summary of the nearby output [32]. It is widely used in CNN for two main reasons: (1) to make the representation approximately invariant to small translations of the input; (2) to reduce the dimensions of intermediate feature maps. For a $T\times X\times Y$ feature map, after passing a 3D-Pooling layer with size of $2\times2\times2$ and stride of $2\times2\times2$, the size of the feature map becomes $\lfloor T/2\rfloor\times\lfloor X/2\rfloor\times\lfloor Y/2\rfloor$. The feature map after this max pooling layer can be smaller if we use max pooling with a larger stride. We, therefore, use smaller max pooling stride for networks with shorter sequence inputs while larger max pooling stride for the networks handling longer sequence input. Because pooling in the temporal dimension can destroy the subtle temporal difference between frames, we follow C3D [2] and LTC

[12] to preserve the temporal resolution at the shallow layer in order to keep the temporal information.

Under the criteria, a total of 8 types of networks are designed to handle different lengths of proxy videos. The dimensions of a proxy video are $T \times 112 \times 112$ where T represents the length of the proxy video ranging from 4 to 96. After the frames in the proxy video forward pass through all the convolutional layers, the size of the feature map is $1 \times 3 \times 3$, while '1' is the temporal dimension, and '3' is the spatial dimension. This feature map then goes through two fully connected layers and one softmax layer to generate the probability distribution. Since all networks have the same number of convolutional layers and fully connected layers, and the features before all the fully connected layer are with the same dimension, the number of parameters remains constant for all these networks.

4. Experiments

In this section, we extensively evaluate the proposed VideoYOLO on the two public benchmark datasets for video classification: UCF101 [26] and HMDB51 [27].

4.1. Datasets

The UCF101 is a widely used benchmark for action recognition with over 13K videos in about 27 hours. Videos have the spatial resolution of 320×240 pixels and 25 FPS frame rate. This dataset contains 101 action classes with large variations in scale, viewpoint, illumination, camera motion, and cluttered background. We follow the standard experimental setting as [26] to use three training and testing splits and report the mean accuracy over the three splits.

The HMDB51 dataset is a large collection of realistic videos from various sources, such as movies and web videos. The dataset consists of 7K videos belonging to 51 actions. The videos have 320×240 pixels spatial resolution and 30 FPS frame rate. Our experiments follow the original evaluation scheme using three training and testing splits and report average accuracy over these splits [27].

Following LTC [12], our evaluation of performance at different lengths of input proxy videos are only conducted on the first split of UCF101.

4.2. Training and Evaluation

For both UCF101 and HMDB51 datasets, every video is sampled into proxy videos with different length of $\{4, 8, 16, 32, 48, 64, 80, 96\}$ frames. Following [2, 12], all the frames in these two datasets are resized to 128×171. Moreover, to center the input data, the mean value of the training data is subtracted from every frame. During training, a 112×112 patch is randomly cropped from proxy videos and horizontally flipped with 50% probability. Training is performed by stochastic gradient descent (SGD) with 30 epochs. The initial learning rate is 0.0001 and is multiplied by 0.1 after every 10 epochs. The optimization is stopped at 30 epochs. During the testing phase, only a single center crop is passed through the network to make the action prediction.

Experiments in [2, 12] show that it is difficult to train deep 3D networks from scratch with UCF101. In [2, 12], the performance gap between the model trained from scratch and the pre-trained model is more than 20%. Especially, with our method, UCF101 is converted into 13K short proxy videos and each proxy video is only one training data sample. It is very difficult to train a network with 70 million parameters from scratch with 9K training samples. Following [2, 12], the pre-trained model in Sport-1M is employed to initialize all our networks. A randomly initialized fully connected layer of size N (number of classes) is added at the end of the network. Then all the layers are fine-tuned at the same time.

4.3. Raw RGB Frame Networks

To examine the effectiveness of sampling a video into one proxy video, we test the performance of the VideoYOLO-16 and VideoYOLO-32 networks with different sampling methods (evenly and randomly) and compare with the C3D and LTC as baseline performances.

Table 2: The comparison of the performance of different sampling methods. With only 16 frames per video, VideoYOLO-16 achieves comparable result with the C3D and LTC models that are trained with all the frames.

Network	Sampling Method	Acc.(%)
C3D [2]	All Frames	82.3
LTC [12]	All Frames	82.4
VideoYOLO-16	Evenly	81.6
VideoYOLO-16	Randomly	82.0
VideoYOLO-32	Evenly	85.1
VideoYOLO-32	Randomly	85.0

The performances of our VideoYOLO-16 and VideoYOLO-32 are listed in Table 2. The performance of the two baselines is directly from the paper C3D [2] and LTC [12]. C3D extracts the activations of the fully connected layer of all the frames, then SVM is trained to classify the action of each video, while LTC obtains the video score by average the per-clip softmax scores and then takes the maximum value of the average as class label. However, with only 16 frames per video, and without SVM classifier, our VideoYOLO-16 achieves comparable performance with C3D and LTC. VideoYOLO-32 which is trained with 32 frames per video outperforms C3D baseline by 2.8%. This demonstrates the effectiveness of our method. The experiments show that there are only minor differences in performance between evenly and randomly sampling methods. In the following section, unless explicitly illustrated, we use evenly sampling as default in order to easily reproduce and compare the performance.

We then investigate the effects of different lengths of the proxy videos at frames of {4, 8, 16, 32, 48, 64, 80, 96} with raw RGB input. As shown in Table 3, VideoYOLO-8, even discarding 95.7% of the training frames and trained with the rest 4.3% of the training frames,

achieves comparable result with C3D and LTC which are trained with all the training frames. This validates our idea that 3D-CNN can learn the ability to identify the action with *only* a few frames. When the length of the proxy videos is increased to 32, the performance is 85.1% which outperforms VideoYOLO-16 by 3.2%. The performance of raw RGB frame networks keeps increasing as adding more frames and saturates at VideoYOLO-80.

Table 3: The performance of different lengths of proxy videos from RGB frames. The second column is the amount of the UCF101 frames used to train the corresponding networks. Even discarding 95.7% of the training frames in the dataset, VideoYOLO-8 achieves comparable performance as C3D and LTC which are trained using all the training data.

Network	Frame Used (%)	RGB Acc (%)
C3D [2]	100.0	82.3
LTC [12]	100.0	82.4
VideoYOLO-4	2.1	79.8
VideoYOLO-8	4.3	81.6
VideoYOLO-16	8.5	81.9
VideoYOLO-32	17.1	85.1
VideoYOLO-48	25.7	85.8
VideoYOLO-64	34.2	85.6
VideoYOLO-80	42.8	85.9
VideoYOLO-96	51.3	85.1

4.4. Optical Flow Networks

Optical flow encodes the pattern of apparent motion of objects between adjacent frames, which makes the recognition easier, as the network does not need to estimate motion implicitly [33]. Moreover, optical flow also conveys rough shape cues of moving objects. 3D-CNN models the high-order motion cues such as spatial and temporal derivatives of optical flow which have been successfully applied to hand engineered features. We train models on raw RGB frames and optical flow images and perform late fusion similar to the two-stream [1] to combine features from these two kinds of inputs. The approach of Brox is a sophisticated optical flow estimator and is known to perform well in various flow estimation benchmarks [33]. To make a fair comparison with others, we employ this algorithm to compute optical flow.

The original optical flow only has two channels with the raw values of horizontal and vertical displacements. In order to train optical flow and raw RGB frame with the same network, we follow [13] to add an extra dimension to make it have the same number of channels as raw RGB frames. Specifically, we threshold the optical flow pixels within $[-20, 20]$ and then linearly rescale it to [0, 255] range. The third channel is set to the magnitude of the first two channels. The colorized optical flow images enable us to reuse the RGB pre-trained models which help to reduce over-fitting.

We systematically study the performance at different temporal resolutions of {4, 8, 16, 32, 48, 64, 80, 96} for optical flow and fusions of optical flow and RGB frame networks. When fusing the optical flow and raw RGB frame networks, geometric mean is computed of the two networks, and then the index with the largest value is assigned as the action label of a video.

Table 4: The performance of fusion of raw RGB frame networks and optical flow networks.

Network	RGB (%)	Flow (%)	Fusion (%)
VideoYOLO-4	79.8	49.9	83.2
VideoYOLO-8	81.6	59.3	85.6
VideoYOLO-16	81.9	64.9	86.4
VideoYOLO-32	85.1	70.8	88.8
VideoYOLO-48	85.8	72.7	89.0
VideoYOLO-64	85.6	72.7	89.8
VideoYOLO-80	85.9	73.7	89.5
VideoYOLO-96	85.1	74.2	89.2

Table 4 illustrates the performance of different lengths of proxy video. For the RGB frame network, the performance keeps increasing as the number of input frame increases and saturates at 80-frame networks. There is slight difference between the performance of {64, 80, 96} networks. For the optical flow network, the performance keeps increasing as we add more optical flow images in the sequence. The performance of fusion networks keeps growing when the temporal resolution is under 64. Even though the performance of optical flow network still increases, the fusion result varies slightly.

4.5. Performance of Temporal Pyramid Fusion

In addition to fuse the optical flow and RGB frame networks, we further study the fusion of networks of different temporal resolutions. The networks with more input frames provide fine-grained information, while networks with fewer input frames provide coarse information.

Table 5: Recognition performance for fusion of different networks. The temporal pyramid fusion achieves about 2% improvement.

Network	Combination(%)				
VideoYOLO-48	✓				✓
VideoYOLO-64	✓	✓		✓	✓
VideoYOLO-80		✓	✓	✓	✓
VideoYOLO-96			✓	✓	✓
Flow Fusion	74.5	76.3	76.0	77.8	79.0
RGB Fusion	86.8	86.2	86.2	86.7	87.0
RGB + Flow	90.3	90.8	90.5	90.9	90.9

Table 5 illustrates the performance of the temporal pyramid fusion. Experiments show that the temporal pyramid fusion usually improves the accuracy by 3% for optical flow networks and 2% for raw RGB frame networks. The experiments demonstrate that these features are complementary to each other. There are minor differences in performance among the networks of {48, 64, 80, 96} frames, and their fusions. Therefore, to balance the training time and the accuracy, we train VideoYOLO-48 and VideoYOLO-64 networks in all the three splits of these two datasets. The performance over three splits is averaged as the final performance on these two datasets.

4.6. Efficiency Analysis

One major advantage of VideoYOLO is the efficiency. Previous methods have to employ a post-processing step to weighting the scores of all the clips belonging to a video to obtain the final score. For example, in the testing phase of Two-Stream [1], 25 frames are selected from a video and 10 crops are obtained from each frame are passed into CNN, meaning the CNN has to be applied at least 250 times in a video. However, since every video is sampled into a constant length of a proxy video, the time consumption of VideoYOLO is fixed for every video. Therefore, we report the speed of processing every video instead of every frame.

Current models for action recognition fall into 3 categories: 2D-CNN, 3D-CNN, and RNN. RNN-based methods actually are the combination of RNN and CNN. Activations of fully connected layers are extracted as feature representation which then is fed to RNN to learn temporal information. Therefore, RNN-based methods usually are slower than 2D-CNN [13]. We compare the runtime of VideoYOLO with C3D [2], Two-Stream [1], and Slow-Fusion [16]. We measure the runtime of the four above-mentioned methods in testing data of the first split of UCF101 using a single NVIDIA TITAN X GPU. The comparison of the efficiency of these networks is illustrated in Table 6. The performances of these methods are directly from the corresponding paper and the time consumption is measured by us.

As shown in Table 6, VideoYOLO is the most efficient method among them. Slow-Fusion [16] and Two-Stream [1] belong to 2D-CNN-based model which is the most inefficient among the four methods because CNN is applied on every frame. C3D [2] and LTC [12]

Table 6: The comparison of VideoYOLO with other frameworks in efficiency and performance. The experiments are conducted on the first split of UCF101. The average length of the video is 7 seconds. The 'VPS' means videos per second. VideoYOLO-32 is able to process 36 videos per second that is 10 times and 7 times faster than prior 2D-CNN and 3D-CNN based models, respectively, while still achieves better performance.

Network	Speed (VPS)	RGB Acc (%)
Slow-Fusion [16]	3.2	65.4
Two-Stream [1]	1.0	72.8
C3D [2]	5.7	82.3
VideoYOLO-8	94.0	81.6
VideoYOLO-16	72.0	81.9
VideoYOLO-32	36.0	85.1

is 3D-CNN-based model which is more efficient than 2D-CNN since the prediction is based on 16-frame clips. Because LTC directly employs C3D model and achieves the nearly same (82.3%/82.4%) performance with RGB frames, we do not test its runtime. VideoYOLO is the most efficient method for real-time processing while obtaining the best performance among these networks.

4.7. Comparison with the State-of-the-Arts

Table 7 demonstrates the comparison of our methods with others including the hand-designed feature-based methods such as Fisher Vector (FV) [34], Hybrid Super Vector (HSV) [35], Multi-Skip Feature Stacking (MIFS) [36], and Stacked Fisher Vectors (SFV) [37], and deep learning methods. These methods are grouped together according to being hand-crafted, using only RGB input to CNNs, or fusion of optical flow and RGB features.

LTC and C3D belong to 3D-CNN methods which are most close to ours. As shown in Table 3, trained with only about 17% of the training frames, VideoYOLO-32 outperforms the C3D by 2.8% which is trained with all the frames. VideoYOLO outperforms C3D in both single and fusion networks by 3.0% and 1.1%, respectively. Compared to LTC, VideoYOLO achieves comparable performance in UCF101 and outperforms LTC in HMDB51 by 1.4%. Moreover, LTC employs heavy data augmentation during training, such as dividing the video with a step of 4. During testing, LTC averages 10 crops from every clip as the classification score for that clip. VideoYOLO does not have such kind of data augmentation, and the classification score is only based on a single center crop of every video.

Spatial-stream [1] and Slow-Fusion [16] belong to 2D-CNN-based methods. Based on a subset frames of the original video, VideoYOLO obtains better performance than both of them. LRCN [14] and LSTM [13] belong to RNN-based methods. Ng *et al.* [13] employed RNN to process up to 120 frames of a video to model the temporal information. However, VideoYOLO obtains better performance in both datasets. Moreover, because they are based on the features extracted by 2D-CNN, it is inefficient compared with VideoYOLO.

Table 7: Comparison with the state-of-the-arts on UCF101 and HMDB51 (mean accuracy across 3 splits). VideoYOLO achieves 86.3% in UCF101 and 58.2% in HMDB51, which are better than any deep approaches trained from RGB frames. For the fusion networks, VideoYOLO achieves comparable performance as the state-of-the-art in these two datasets.

	Method		UCF101 Acc (%)	HMDB51 Acc (%)
Hand-crafted	IDT+FV	[34]	85.9	57.2
	IDT+HSV	[35]	87.9	61.1
	IDT+MIFS	[36]	89.1	65.1
	IDT+SFV	[37]	-	66.8
	MoFVP	[38]	61.7	88.3
CNN (RGB)	Slow fusion (from scratch)	[16]	41.3	-
	C3D (from scratch)	[2]	44	-
	Slow fusion	[16]	65.4	-
	Spatial stream	[1]	73.0	40.5
	C3D (1 net)	[2]	82.3	-
	LTC	[12]	82.4	-
	C3D (3 nets)	[2]	85.2	-
	VideoYOLO$_{RGB}$	Ours	**85.3**	**56.5**
	VideoYOLO$_{RGB}$ (2 nets)	Ours	**86.3**	**58.2**
Fusion (Flow+RGB)	LRCN	[14]	82.9	-
	Two-stream	[1]	88.0	59.4
	Convolutional pooling	[13]	88.2	-
	LSTM	[13]	88.6	-
	Multi Source CNN	[39]	89.1	54.9
	LTC	[12]	91.7	64.8
	Transformations	[40]	92.4	62.0
	VideoYOLO	Ours	**90.6**	**66.6**

For the single RGB networks, VideoYOLO is much faster than both 2D-CNN methods [1, 16] and 3D-CNN methods [2, 12] while achieves better performance than them. For the fusion networks, by discarding the majority of the frames, VideoYOLO learns the overall temporal information from proxy videos and the fusion of the networks is much simpler than other methods, while achieves comparable performance to state-of-the-arts in these two datasets.

4.8. Discussion

We analyze the statistics of UCF101 and HMDB51. In these two datasets, the length of the original videos actually is the length of the actions. Fig. 3 shows that nearly 90% of videos in these two datasets have less than 300 frames. To investigate the effects of the length of original videos to VideoYOLO performance, we analyze the details of the relationship between the accuracy and the length of the original videos. All the analysis is

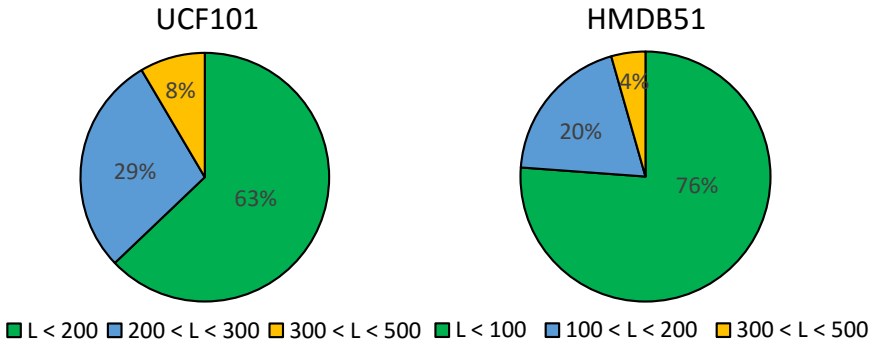

Figure 3: The statistical analysis of the length of the videos in UCF101 and HMDB51. L represents the length of the action lasting in the original videos.

based on the performance of VideoYOLO networks in the first split of UCF101.

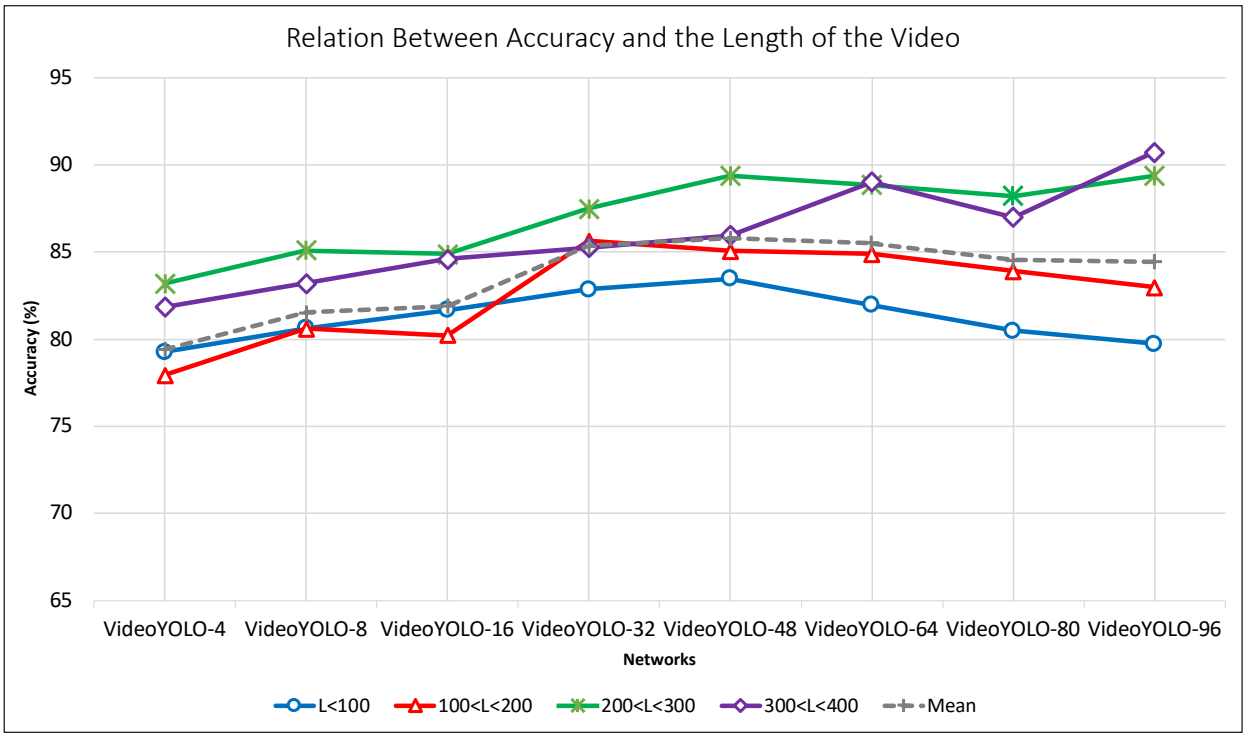

Figure 4: The statistical analysis of the relation between the recognition accuracy and length of actions in UCF101. L represents the number of the frames of each original video. The length of the original videos nearly equals to the length of the action in this dataset. The accuracy of L<100 is the average accuracy of all the videos with length less than 100 frames. With the proxy video, VideoYOLO performs well in both short-lasting and long-lasting actions.

As shown in Fig. 4, VideoYOLO performs well on actions with different lengths. Take the VideoYOLO-4 as the example, the recognition accuracy is 79.3% for actions lasting less

than 100 frames, 77.9% for actions lasting between 100 and 200 frames, 83.0% for actions lasting between 200 to 300 frames, and 81.9% for actions lasting between 300 and 400 frames. This demonstrates that VideoYOLO can capture temporal and appearance information from a few frames well for both long-lasting and short-lasting actions. All networks following the similar pattern demonstrate that the proxy video can preserve the dynamics of the original video well with a few frames.

5. Conclusions

We have presented VideoYOLO, a model to take advantage of the long-term temporal convolution to learn the overall temporal information of a video. By selecting a subset of frames of the original video as the proxy video, 3D-CNN is applied to process every video in one single step. Unlike other approaches which need to aggregate temporal information from short clips belonging to that video, VideoYOLO is able to capture the overall temporal information from the proxy videos in one process. Furthermore, VideoYOLO does not need any post-processing and is very efficient compared to other methods such as 2D-CNN and RNN-based methods which makes it possible for real-time applications.

6. Acknowledgment

This work was supported in part by NSF Grants EFRI-1137172 and IIS-1400802.

References

[1] K. Simonyan, A. Zisserman, Two-Stream Convolutional Networks for Action Recognition in Videos, in: NIPS, 2014.
[2] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, Learning spatiotemporal features with 3D convolutional networks, in: ICCV, 2015.
[3] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, K. Saenko, Sequence to Sequence – Video to Text, in: ICCV, 2015.
[4] S. Yeung, O. Russakovsky, G. Mori, L. Fei-Fei, End-to-end learning of action detection from frame glimpses in videos, in: CVPR, 2016.
[5] X. Yang, P. Molchanov, J. Kautz, Multilayer and Multimodal Fusion of Deep Neural Networks for Video Classification, in: ACM Multimedia, 2016.
[6] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in: NIPS, 2012.
[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: CVPR, 2015.
[8] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: ICLR, 2015.
[9] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016.
[10] L. Jing, Y. Ye, X. Yang, Y. Tian, 3D Convolutional Neural Network with Multi-Model Framework for Action Recognition.
[11] S. Ji, W. Xu, M. Yang, K. Yu, 3D convolutional neural networks for human action recognition, TPAMI 35 (1) (2013) 221–231.
[12] G. Varol, I. Laptev, C. Schmid, Long-term Temporal Convolutions for Action Recognition, arXiv:1604.04494.

[13] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, G. Toderici, Beyond Short Snippets: Deep Networks for Video Classification, in: CVPR, 2015.
[14] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, T. Darrell, Long-term recurrent convolutional networks for visual recognition and description, in: CVPR, 2015.
[15] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, L. Van Gool, Temporal segment networks: towards good practices for deep action recognition, in: ECCV, 2016.
[16] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale Video Classification with Convolutional Neural Networks, in: CVPR, 2014.
[17] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE TNN 5 (2) (1994) 157–166.
[18] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.
[19] N. Srivastava, E. Mansimov, R. Salakhutdinov, Unsupervised Learning of Video Representations using LSTMs, in: ICML, 2015.
[20] O. Kliper-Gross, T. Hassner, L. Wolf, The action similarity labeling challenge, TPAMI 34 (3) (2012) 615–621.
[21] X. Peng, Y. Qiao, Q. Peng, Q. Wang, Large margin dimensionality reduction for action similarity labeling, IEEE SPL 21 (8) (2014) 1022–1025.
[22] K. G. Derpanis, M. Lecce, K. Daniilidis, R. P. Wildes, Dynamic scene understanding: The role of orientation features in space and time in scene classification, in: CVPR, 2012.
[23] N. Shroff, P. Turaga, R. Chellappa, Moving vistas: Exploiting motion for describing scenes, in: CVPR, 2010.
[24] C. Feichtenhofer, A. Pinz, R. P. Wildes, Bags of spacetime energies for dynamic scene recognition, in: CVPR, 2014.
[25] X. Ren, M. Philipose, Egocentric recognition of handled objects: Benchmark and analysis, in: CVPRw, IEEE, 2009, pp. 1–8.
[26] K. Soomro, A. R. Zamir, M. Shah, UCF101: A dataset of 101 human actions classes from videos in the wild, CRCV-TR 12-01.
[27] H. Kuehne, H. Jhuang, R. Stiefelhagen, T. Serre, Hmdb51: A large video database for human motion recognition, in: HPCSE, Springer, 2013, pp. 571–582.
[28] B. G. Fabian Caba Heilbron, Victor Escorcia, J. C. Niebles, ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding, in: CVPR, 2015.
[29] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, S. Vijayanarasimhan, YouTube-8M: A Large-Scale Video Classification Benchmark, CoRR abs/1609.08675.
[30] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, in: CVPR, 2016.
[31] J. Redmon, A. Farhadi, YOLO9000: Better, Faster, Stronger, in: CVPR, 2017.
[32] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, `http://www.deeplearningbook.org`.
[33] T. Brox, A. Bruhn, N. Papenberg, J. Weickert, High accuracy optical flow estimation based on a theory for warping, in: ECCV, 2004.
[34] H. Wang, C. Schmid, Action recognition with improved trajectories, in: ICCV, 2013.
[35] X. Peng, L. Wang, X. Wang, Y. Qiao, Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice, CVIU 150 (2016) 109–125.
[36] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, B. Raj, Beyond gaussian pyramid: Multi-skip feature stacking for action recognition, in: CVPR, 2015.
[37] X. Peng, C. Zou, Y. Qiao, Q. Peng, Action recognition with stacked fisher vectors, in: ECCV, 2014.
[38] L. Wang, Y. Qiao, X. Tang, MoFAP: A Multi-level Representation for Action Recognition, IJCV 119 (2015) 254–271.
[39] E. Park, X. Han, T. L. Berg, A. C. Berg, Combining multiple sources of knowledge in deep CNNs for action recognition, in: WACV, 2016.
[40] Actions˜ transformations, author=Wang, Xiaolong and Farhadi, Ali and Gupta, Abhinav, in: CVPR,

2016.