MSR Asia MSM at ActivityNet Challenge 2016

Zhaofan Qiu, Dong Li, Chuang Gan, Ting Yao*, Tao Mei, and Yong Rui
Microsoft Research, Beijing, China
{tiyao, tmei, yongrui}@microsoft.com

Abstract

This notebook paper presents overview and comparative analysis of our system designed for untrimmed video classification task in ActivityNet Challenge 2016. We investigate and exploit multiple spatio-temporal clues, i.e., frames, motion (optical flow), and short video clips, using 2D or 3D convolutional neural networks (CNNs). The mechanism of different quantization methods are studied as well. Furthermore, improved dense trajectory with fisher vector encoding on long video clips and MFCC audio features are utilized. All activities are classified by late fusing the predictions of one-versus-rest linear SVMs learnt on each clue. Finally, OCR is employed to refine the prediction scores.

1. Introduction

Recognizing activities in videos is a challenging task as video is an information-intensive media with complex variations. In particular, an activity may be represented by different clues including frames, motion (optical flow), short video clips, long video clips, audio and OCR. In this work, we aim at investigating these multiple clues to activity classification in videos.

The remaining sections are organized as follows. Section 2 describes our activity recognition system. Section 3 presents all the features, while Section 4 details feature quantization strategies. In Section 5, we provide empirical evaluations, followed by the conclusions in Section 6.

2. Recognition Framework

Our activity recognition framework is shown in Figure 1. In general, the untrimmed video classification process is composed of three stages, i.e., multi-stream feature extraction, feature quantization and prediction generation. For deep feature extraction, we follow the multi-stream approaches in [4, 6], which represent the input video by a hierarchical structure including individual frames, consecutive frames and short clips. In addition to deep features, two most complementary hand-crafted features, i.e., iDT and audio MFCC, are exploited to further enrich the video representations. After extraction of raw features, different quantization and pooling methods are utilized on different features to produce representations of each video. A linear SVM is trained on each kind of video representations and the predictions from multiple SVMs are combined by linearly fusion. When training SVM, we fix $C = 100$ for all the experiments. Finally, OCR is employed to refine the list of recognized videos for each activity.

*Principal Designer.

3. Multi-Stream Features

In our framework, we extract the features from multiple clues including frames, motion, short clips, long clips, audio and OCR.

3.1. Frame

To extract frame-level representations from video, we first uniformly sample 50 frames from each video, and then use pre-trained/finetuned 2D CNNs as frame-level feature extractors. We choose three popular 2D CNNs in image classification: VGG [7], GoogLeNet [5, 9] and ResNet [1]. The performances between features extracted from different layers of different architectures will be discussed later.

3.2. Motion

To model the change of consecutive frames, we apply another CNNs to optical flow "image," which can extract motion features between consecutive frames. When extracting motion features, we follow the setting of [12], which fed 20 optical flow images, consisting of two-direction optical flow from 10 consecutive frames, into VGG_16 network in each iteration. We use VGG_16 model and sample rate is 50 per video, which means 50×20 optical flow "images" are considered for each video.

3.3. Short Clip

In addition to frames and motion between consecutive frames, we further exploit popular 3D CNN architecture, C3D [10], to construct video clip features from both spatial and temporal dimensions. The C3D model is pre-trained on

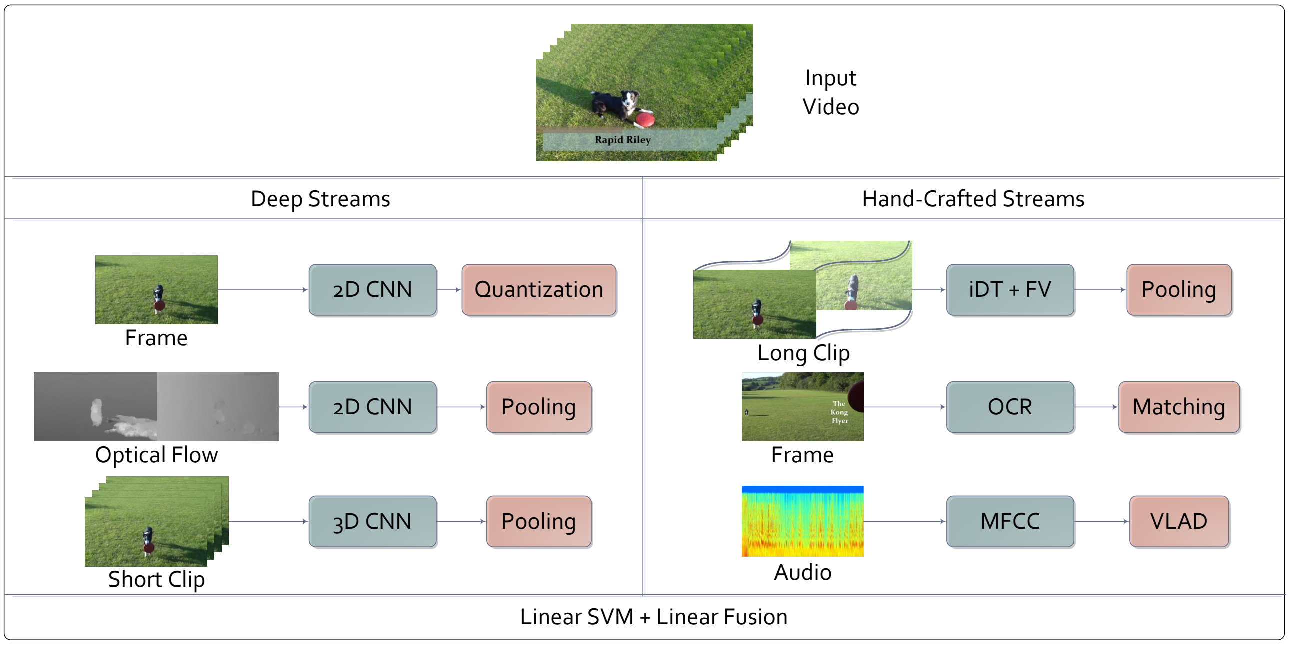

Figure 1. Framework of our proposed system.

Sports-1M dataset [3]. We fix the length of short clip to 16 frames, and sample rate is also 50 per video.

3.4. Long Clip

For long clips, we choose the state-of-the-art handcrafted features - improved dense trajectory (iDT) [11] on each sampled clip. Specifically, trajectory feature, histogram of oriented gradients (HOG), histogram of flow (HOF), and motion boundary histogram (MBH) are computed for each trajectory obtained by tracking points in video clips. Furthermore, fisher vector encoding is used to quantize the features and create high dimensional representations for each clip. Considering that the extraction of iDT is very time consuming, we split each video into a set of five-second clips evenly without any overlap.

3.5. Audio

For audio features, MFCC are extracted and exploited. As the duration of different videos are different, the counts of MFCC are also different.

3.6. OCR

Tesseract OCR [8] is used to extract text from video frames. We apply the detector on each frame from the whole video, followed by string matching with activity name. Before matching, we simplify the activity name by removing meaningless word, e.g. "doing the," and remove some misleading categories, e.g. "polo." Finally, we simply take the videos as positive samples if the activity name appears in the text of their frames.

4. Feature Quantization

In this section, we describe three quantization methods to generate video-level representations from frame-level or clip-level features.

4.1. Average Pooling

As shown in the Figure 1, we use average pooling upon the extracted features from consecutive frames, short clips and long clips. For a set of frame-level or clip-level features $F = \{f_1, f_2, ..., f_N\}$, the video-level representations are produced by simply averaging all the features in the set:

$$R_{pooling} = \frac{1}{N} \sum_{i:f_i \in F} f_i \quad , \tag{1}$$

where $R_{pooling}$ denotes the final representations.

4.2. VLAD

Recently, Vectors of Locally Aggregated Descriptors (VLAD) [2] shows good ability on feature quantization. With K-means centers $C = \{c_1, c_2, ..., c_K\}$, video-level representations from VLAD can be described as:

$$\begin{aligned} u_k &= \sum_{i:NN(f_i)=c_k} (f_i - c_k) \\ R_{vlad} &= normalize(u) \end{aligned} \quad , \tag{2}$$

where $NN(f_i)$ denotes f_i's nearest neighbor in C. We choose the variant of VLAD called VLAD-k, which replaces the nearest neighbor with k-nearest neighbors, and fix $k = 5$. For feature normalization, we choose power, l_2 and intra-normalization by default.

Table 1. Top1-accuracy of different 2D CNN architectures on ActivityNet validation set. The video feature are extracted on 50 sampled frames followed by average pooling.

Network	Fintuned	Layer	Top1
VGG_19		fc6	66.59%
GoogLeNet		pool5	68.76%
ResNet_152		pool5	**71.43**%
VGG_16	√	fc6	67.03%
GoogLeNet	√	pool5	68.57%
ResNet_50	√	pool5	68.43%
ResNet_152	√	pool5	**74.82**%

4.3. Deep Quantization

VLAD has two obvious weaknesses: (1) high computation and storage cost; (2) label information is ignored. Therefore, we present a novel network-based quantization method called Deep Quantization (DQ).

First, we train a generative neural network with parameters θ on the top of feature extraction network. Following the fisher kernel method, the video-level representations are defined as

$$\begin{aligned} &L_{Generative}(\theta) = \sum_{f \in TrainingSet} -log\, p(f, \theta) \\ &\widehat{\theta} = \arg\max_{\theta} L_{Generative}(\theta) \qquad , \\ &R_{DQ} = normalize(\sum_{i: f_i \in F} \tfrac{\partial(-log\, p(f_i, \widehat{\theta}))}{\partial \widehat{\theta}}) \end{aligned} \tag{3}$$

where $p(f, \theta)$ is the generative network output. After optimizing parameters θ, the gradient calculation and accumulation can be processed in an end-to-end manner during backpropagation, and no extra storage is required. To further improve the ability of representations, we propose a semi-supervised optimizing function as:

$$\begin{aligned} &L(\theta) = L_{Generative}(\theta) + \lambda L_{Classification}(\theta) \\ &\widehat{\theta} = \arg\max_{\theta} L(\theta) \qquad . \\ &R_{DQ} = normalize(\sum_{i: f_i \in F} \tfrac{\partial(-log\, p(f_i, \widehat{\theta}))}{\partial \widehat{\theta}}) \end{aligned} \tag{4}$$

The detailed description of our deep quantization network and more experimental analysis will be published on arXiv.org soon.

5. Experiment

5.1. 2D CNNs Comparison

Here we compare three popular 2D CNN architectures: VGG, GoogLeNet and ResNet. The comparison results on validation set are shown in Table 1.

The settings of 2D CNN are generally divided into two parts, i.e., "*pre-trained model + average pooling*" and "*finetuned model + average pooling*." All the four finetuned networks are initialized by pre-trained models, and finetuned on ActivityNet training set. We can observe that ResNet_152 achieves the highest accuracy among the three architectures and it will be further improved by finetuning.

Table 2. Top1-accuracy of different quantization methods on ActivityNet validation set. All the local feature are extracted from ResNet_152 architecture.

Method	Fintuned	Layer	Top1
Average Pooling	√	pool5	74.82%
Average Pooling		pool5	71.43%
VLAD		rec5c	76.70%
Deep Quantization		rec5c	**78.55**%

5.2. Quantization Comparison

Table 2 shows the results of different quantization methods on ResNet_152. We exploit VLAD and our Deep Quantization on the outputs of Res5c layer which is the last convolutional layer. It is worth noting that we only apply these two quantization methods on default ResNet_152 model. For VLAD, we first reduce the feature dimension to 1024 by PCA, and then apply k-means with $k = 256$, which means the dimension of representations for each video is 1024×256. For Deep Quantization, we set the number of hidden state to 128, making the feature dimension of 2048×128 in total.

It can be observed that VLAD obtains large performance improvement over Average Pooling method (76.70% vs 71.43%), which is even higher than finetuned model. Our proposed Deep Quantization model achieves better accuracy than VLAD (78.55% vs 76.70%), and it is the best setting of our 2D CNN.

5.3. Performance Comparison

Table 3 shows the performances of all the components in our submission and their fusion weights. The fusion weights are tuned using gradient descend on validation set by minimizing the classification loss. The OCR results are considered as post-processing and employed after linear fusion.

Overall, our Deep Quantization on ResNet_152 achieves the highest accuracy (78.55%) of single component, and it also obtains the highest fusion weight (24.9%). Although MFCC only gets 17.92% top1-accuracy, its fusion weight (19.1%) is the second highest due to the high complementarity between aural and visual features.

For the final submission, we train the SVMs using training and validation sets. All the components are fused using the weights tuned on validation set. Our final performance on test set is also shown on Table 3. Our top1-accuracy on test set is about 2% higher than validation set. This result basically indicates that more data used in training process may lead to higher recognition accuracy.

Table 3. Comparisons of different components in our framework on ActivityNet validation set. We also include the performance on ActivityNet test set from leader-board. Please note that there are two different settings of iDT, while the "Sample-10" means we randomly sample 10 long clips and average the predicting probabilities.

Stream	Feature	Fintuned	Layer	Quantization	Top1	Top3	MAP	Fusion Weights
Frame	VGG_19		fc6	Ave	66.59%	82.70%	70.22%	0.7%
	GoogLeNet		pool5	Ave	68.76%	84.73%	73.37%	1.2%
	ResNet_152		pool5	Ave	71.43%	86.45%	76.56%	0.6%
	VGG_16	√	fc6	Ave	67.03%	83.68%	70.12%	0.4%
	GoogLeNet	√	pool5	Ave	68.57%	85.26%	72.19%	1.4%
	ResNet_50	√	pool5	Ave	68.72%	86.13%	72.96%	8.4%
	ResNet_152	√	pool5	Ave	74.82%	87.59%	79.43%	6.3%
	ResNet_152		res5c	VLAD	76.70%	89.07%	81.52%	3.8%
	ResNet_152		res5c	DQ	78.55%	91.16%	84.09%	24.9%
Motion	VGG_16	√	fc6	Ave	49.05%	65.96%	49.06%	8.3%
Short Clip	C3D		fc6	Ave	65.80%	81.16%	67.68%	8.8%
Long Clip	iDT+FV			Ave	64.70%	77.98%	68.69%	14.3%
	iDT+FV			Sample-10	65.90%	80.15%	69.18%	1.8%
Audio	MFCC			VLAD	17.94%	26.10%	15.47%	19.1%
Fusion all					83.23%	94.24%	89.17%	
+OCR					84.26%	94.65%	90.03%	
On test set					86.68%	95.53%	91.93%	

6. Conclusion

In ActivityNet Challenge 2016, we mainly focused on multiple visual features and different strategies of feature quantization. The audio features can help classify some activities and OCR can be further employed to improve the accuracy. Our future works include the exploration of ASR and more in-depth studies of how fusion weights of different clues could be determined to boost the classification performance.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[2] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[3] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[4] Q. Li, Z. Qiu, T. Yao, T. Mei, Y. Rui, and J. Luo. Action recognition by learning deep multi-granular spatio-temporal video representation. In *ICMR*, 2016.

[5] P. Mettes, D. C. Koelma, and C. G. Snoek. The imagenet shuffle: Reorganized pre-training for video event detection. In *ICMR*, 2016.

[6] Z. Qiu, Q. Li, T. Yao, T. Mei, and Y. Rui. Msr asia msm at thumos challenge 2015. In *THUMOS'15 Action Recognition Challenge*, 2015.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[8] R. Smith. An overview of the tesseract ocr engine. In *ICDAR*, 2007.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[10] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. *ICCV*, 2015.

[11] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[12] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015.