Deep Local Video Feature for Action Recognition

Zhenzhong Lan[1] Yi Zhu[2] Alexander G. Hauptmann[1] Shawn Newsam[2]
[1]Carnegie Mellon University [2]University of California, Merced
{lanzhzh,alex}@cs.cmu.edu {yzhu25,snewsam}@ucmerced.edu

Abstract

We investigate the problem of representing an entire video using CNN features for human action recognition. End-to-end learning of CNN/RNNs is currently not possible for whole videos due to GPU memory limitations and so a common practice is to use sampled frames as inputs along with the video labels as supervision. However, the global video labels might not be suitable for all of the temporally local samples as the videos often contain content besides the action of interest. We therefore propose to instead treat the deep networks trained on local inputs as local feature extractors. The local features are then aggregated to form global features which are used to assign video-level labels through a second classification stage. This framework is more robust to the noisy local labels that result from propagating video-level labels. We investigate a number of design choices for this local feature approach such as the optimal sampling and aggregation methods. Experimental results on the HMDB51 and UCF101 datasets show that a simple maximum pooling on the sparsely sampled local features leads to significant performance improvement.

1. Introduction

Despite much effort and progress, deep convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have yet to achieve the same success on video classification as they have on image classification. This can in large part be attributed to the following two differences between image and videos, differences which are key to deep-learning based approaches. First, videos are much larger in size and thus it becomes memory prohibitive to train and apply CNNs/RNNs at the video level. Second, it is very difficult to construct the large labeled video datasets required for training deep networks. Recent approaches [14, 22, 23] circumvent these problems by learning on sampled frames or very short video clips (temporally local inputs[1]) with video-level (global) labels.

[1]local from here on will mean temporally local

However, video-level label information can be incomplete or even missing at frame/clip-level. This information mismatch leads to the problem of false label assignment. In other words, the imprecise frame/clip-level labels populated from video labels are too noisy to guide precise mapping from local video snippets to labels. To deal with this problem, a common practice is to sample multiple frames/clips from a video at testing time and aggregate the prediction scores of these sampled frames/clips to get the final results for that video. However, simply averaging the prediction scores, without another level of mapping, is not enough to recover the damages brought by false label assignment. *Especially for long untrimmed videos [7, 4] in open domain, the problem become even more significant.*

We instead compensate for the noisy labels by treating the deep networks trained on local inputs as feature extractors as shown in Figure 1. Local features extracted using the pre-trained networks are aggregated into global video-level features and another mapping function (*e.g.*, a shallow network) is learned using the same dataset to assign video-level labels.

Our method is therefore related to the fine-tuning practices that are popular in image classification. The major difference is that we train our feature extraction networks with local data and with very noisy labels due to the false label assignment. We thus rely heavily on the shallow network to compensate for the suboptimal local feature learning.

Our method is also similar to the common practice of using networks pre-trained on the ImageNet image classification task to extract frame-level (local) features for video classification [26, 9]. The main difference is that our local feature extractors (deep networks) are trained on the target dataset. Therefore, the features extracted from deep networks are in-domain. We don't have the domain gap problem as we have in using the ImageNet trained deep networks.

We name our new class of local video features Deep lOcal Video Feature (DOVF).

In summary, DOVF is a class of local video features that are extracted from deep neural networks trained on local video clips using global video labels. In this paper, we in-

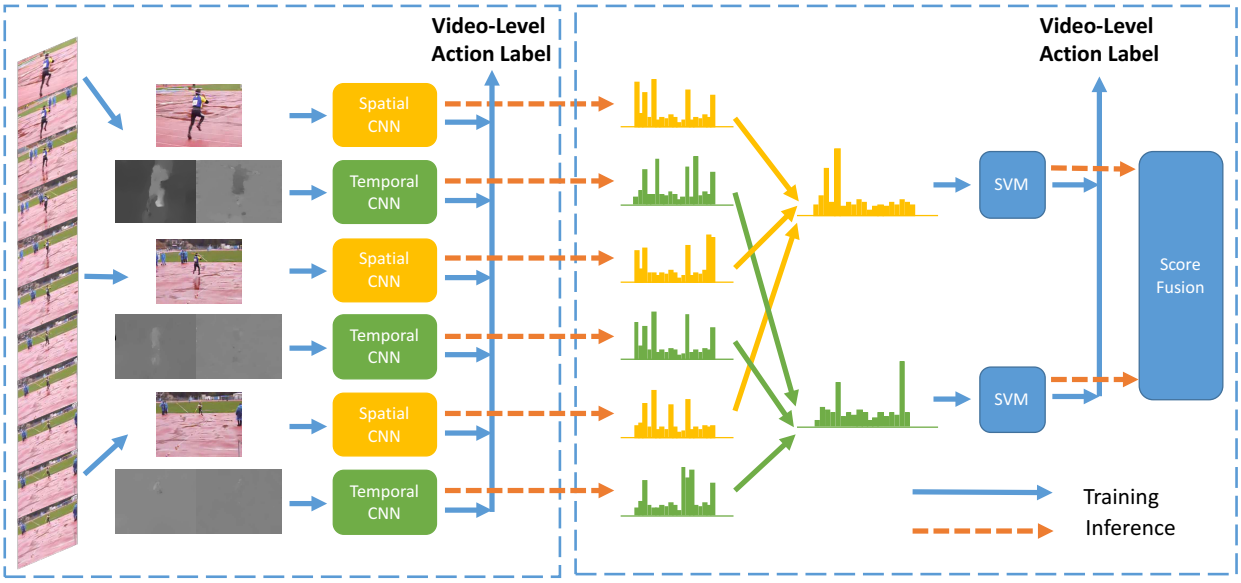

Figure 1. Overview of our proposed framework which consists of two stages. First (Left): A temporal segment network [23] is trained using local video snippets with the video-level labels. These networks are used as local feature extractors. Second (Right): The local features are aggregated to form a global feature which is then mapped to the video-level labels. Our results show that this two stage process compensates for the noisy snippet labels that result from propagating the video-level labels.

vestigate the following design choices related to DOVF:

- From which layer(s) of the CNNs should the local features be extracted? Without further investigation, the only guidance we have is that we should avoid the probability layer as it is likely to severely overfit the noisy training data and thus result in a distribution that varies greatly between the training and test sets.
- What is the best way to aggregate the local features into video-level global features? We consider a number of feature aggregation methods such as mean pooling, max pooling, Fisher Vectors (FV) encoding, *etc*.
- How densely should the local features be extracted? Sparse temporal sampling would be preferred from a efficiency standpoint.
- How complementary is DOVF to traditional local features such as IDT [20]? The more complementary they are, the more opportunity there is for improvement by applying techniques that have been developed for traditional local features.

The remainder of this paper is organized as follows. We first provide some background on video features with an emphasis on recent work on learning with deep neural networks. We then describe the experimental settings we use to evaluate our framework on the HMDB51 and UCF101 datasets. We conclude with a discussion including potential improvements.

2. Related works

New video representations have typically been the main source of breakthroughs for video classification.

In traditional video representations, trajectory based approaches [20, 6], especially the Dense Trajectory (DT) and its improved form, IDT [19, 20], are the basis of current state-of-the-art hand-crafted algorithms. These trajectory-based methods are designed to address the shortcomings of image-extended video features. Their superior performance validates the need for motion feature representations. Many studies have tried to improve upon IDT due to its success and popularity. Peng *et al*. [11] enhanced the performance of IDT by increasing the codebook sizes and fusing multiple coding methods. Sapienza *et al*. [13] explored ways to sub-sample and generate vocabularies for DT features. Hoai and Zisserman [5] achieved superior performance on several action recognition datasets by applying data augmentation, modeling the score distributions over video subsequences, and capturing the relationships among action classes. Fernando *et al*. [3] modeled the evolution of appearance in video and achieved state-of-the-art results on the Hollywood2 dataset. [10] proposed to extract features from videos at multiple playback speeds to achieve speed invariance. However, these traditional, hand-crafted methods have recently started to become overshadowed by the rise of deep learning using neural networks.

Motivated by the success of CNNs, researchers have invested significant effort towards developing CNN equiva-

lents for learning video features. Several accomplishments have been reported from using CNNs for action recognition in videos [27, 25, 16, 29]. Karpathy *et al*. [8] trained deep CNNs using one million weakly labeled YouTube videos and reported moderate success using the network as a feature extractor. Simonyan and Zisserman [14] demonstrated a result competitive with IDT [20] by training deep CNNs using both sampled frames and stacked optical flow. Tran *et al*. [15] explored 3D CNNs to simultaneously learn spatiotemporal features without pre-computing optical flow. This allows them to achieve competitive performance at much faster rates. Wang *et al*. [21, 22, 23] provide insightful analyses on improving two-stream frameworks such as pre-training two-stream CNNs, using smaller learning rates, using deeper networks, etc. These improvements result in a CNN-based approach that finally outperforms IDT [20] by a large margin on the UCF101 dataset. These approaches, however, all rely on shot-clip predictions to determine the final video labels and do not use global features.

Two concurrent works [1, 12] on global features for action recognition have recently been posted on arXiv. Both propose new feature aggregation methods to pool the local neural network features to form global video features. Diba *et al*. [1] propose a bilinear model to pool the outputs of the last convolutional layers of pre-trained networks and achieve state-of-the-art results on both the HMDB51 and UCF101 datasets. Qiu *et al*. [12] propose a new quantization method similar to FV and achieve comparable performance to [1]. However, neither work provides detailed analysis of the local neural network features that are used. In this paper, we perform an extensive analysis and show that a simple max pooling can achieve similar or better results compared to much more complex feature aggregation methods such as those in [1, 12].

3. Methodology

In this section, we first review temporal segment networks [23], the architecture upon which our approach is built. We next describe our Deep lOcal Video Features (DOVF), methods for aggregating them to form global features, and the mapping of the global features to video-level labels. Finally, we provide our experimental settings.

3.1. Temporal Segment Networks

With the goal of capturing long-range temporal structure for improved action recognition, Wang *et al*. propose temporal segment networks (TSN) [23] with a sparse sampling strategy. This allows an entire video to be analyzed with reasonable computational costs. TSN first divides a video evenly into three segments and one short snippet is randomly selected from each segment. Two-stream networks are then applied to the short snippets to obtain the initial action class prediction scores. TSN finally uses a segmental consensus function to combine the outputs from multiple short snippets to predict the action class probabilities for the video as a whole.

Wang et al. [23] show TSN achieves state-of-the-art results on the popular action recognition benchmarks UCF101 and HMDB51. These results demonstrate the importance of capturing long-range temporal information for video analysis. However, the training of the local snippet classifiers is performed using the video-level labels. As noted earlier, these are likely to be noisy labels and will thus limit the accuracy of the snippet-level classification.

We there propose instead to use the snippet-level analysis for local feature extraction and add a second stage which maps the aggregated features to the video-level labels. The combination of DOVF and a second classification stage compensates for the suboptimal snippet-level classification that results from the noisy training dataset.

3.2. Deep local video feature (DOVF)

Instead of performing action recognition in a single step like [23, 1], our framework consists of two stages. In the first stage, deep networks (*e.g*., TSN) that have been trained with video-level labels to perform snipped-level classification are used as local feature extractors. In the second stage, the local features are aggregated to form global features and another classifier which has also been trained using the video-level labels performs the video-level classification.

The training of our classification framework proceeds as follows where each video V in the training set has ground truth action label p. In the first stage, V is evenly divided into N segments, $v_1, v_2, \cdots, v_N$ and one short snippet is randomly selected from each segment, $s_1, s_2, \cdots, s_N$. These snippets are assigned the video-level labels and the snippets from all the training videos are used to train a two-stream CNNs (single RGB video frame and stack of consecutive optical flow images). The details on training the two-stream CNNs can be found in [22, 23]. Once trained, the network is used to extract local features, $f_1, f_2, \cdots, f_N$ from a video.

In the second stage, the local features are aggregated into a global feature f_G,

$$f_G = G(f_1, f_2, \cdots, f_N) \tag{1}$$

where G denotes the aggregation function. We explore different aggregation functions in Section 4.2. We then learn a classifier that maps the global feature f_G to the video label p:

$$p = M(f_G). \tag{2}$$

Once trained, the framework can be used to predict the label of a video. Figure 1 contains an overview of the framework.

ID	VGG16			Inception-BN		
	Name	Dimension	Type	Name	Dimension	Type
L-1	fc8	101	FC	fc-action	101	FC
L-2	fc7	4096	FC	global_pool	1024	Conv
L-3	fc6	4096	FC	inception_5b	50176	Conv
L-4	pool5	25088	Conv	inception_5a	50176	Conv
L-5	conv5_3	100352	Conv	inception_4e	51744	Conv

Table 1. Names, dimensions and types of the layers that we consider in the VGG16 and Inception-BN networks for local feature extraction.

Layers	Spatial CNNs (%)		Temporal CNNs (%)		Two-stream (%)	
	VGG-16	Inception-BN	VGG16	Inception-BN	VGG-16	Inception-BN
L-1	77.8	83.9	82.6	83.7	89.6	91.7
L-2	79.5	88.3	85.1	88.8	91.4	94.2
L-3	80.1	88.3	86.6	88.7	91.8	93.9
L-4	83.7	85.6	86.5	85.3	92.4	91.4
L-5	83.5	83.6	87.0	83.6	92.3	89.8
TSN [23]	79.8	85.7	85.7	87.9	90.9	93.5

Table 2. Layer-wise comparison of VGG-16 and Inception-BN networks on the split 1 of UCF101. The values are the overall video-level classification accuracy of our complete framework.

3.3. Experimental settings

We compare two networks, VGG16 and Inception-BN, for the local feature extraction. (We use networks trained by Wang *et al*. [22, 23].) We further compare the outputs from the last five layers from each network as our features. Table 1 shows the layer names of each network and the correspondent feature dimensions. We divide the layers into two categories: fully-connected (FC) layers and convolution (Conv) layers (pooling layers are treated as Conv layers). FC layers have significantly more parameters and are thus more likely to overfit the training data than the Conv layers. As shown, VGG16 has three FC layers while Inception-BN only has one.

Following the scheme of [14, 23], we evenly sample 25 frames and flow clips for each video. For each frame/clip, we perform 10x data augmentation by cropping the 4 corners and center along with horizontal flipping. A single feature is computed for each frame/clip by averaging over the augmented data. This results in a set of 25 local features for each video. The dimensions of local features extracted from different network/layer combinations are shown in Table 1.

We compare a number of local feature aggregation methods ranging from simple mean and maximum pooling to more complex feature encoding methods such as Bag of words (BoW), Vector of Locally Aggregated Descriptors ($VLAD$) and Fisher Vector (FV) encoding. In order to incorporate global temporal information, we divide each video into three parts and perform the aggregation separately. That is, the first eight, middle nine and final eight of the 25 local features are separately aggregated and then concatenated to form the final global feature. This increases the final feature dimension by three. After concatenation, we perform a square root normalization and L2 normalization as in [10] on the global feature.

We use support vector machines (SVMs) to map (classify) the global features to video-level labels. We use a chi-square kernel and $C = 100$ as in [10] except for the FV and VLAD aggregated features where we use a linear kernel as suggested in [18]. Note that while we use SVMs to predict the video action labels, other mappings/classifiers, such as a shallow neural network, could also be used.

The spatial-net and temporal-net prediction scores of the two-stream network are fused with weights 1 and 1.5, respectively, as in [23].

4. Evaluation

In this section, we experimentally explore the design choices posed in the Introduction using the UCF101 and HMDB51 datasets.

UCF101 is composed of realistic action videos from YouTube. It contains $13,320$ video clips distributed among 101 action classes. HMDB51 includes $6,766$ video clips of 51 actions extracted from a wide range of sources, such as online videos and movies. UCF101 and HMDB51 both have a standard three split evaluation protocol. We report the average recognition accuracies over the three splits.

Our default configuration uses the outputs of the global_pool layer in the Inception-BN network as the local features due to this layer's low dimension (3072 dimensions with global information encoding). It also uses maximum pooling to aggregate the local features to form the global features.

Layers	Spatial CNNs (%)		Temporal CNNs (%)		Two-stream (%)	
	HMDB51	UCF101	HMDB51	UCF101	HMDB51	UCF101
$Mean$	56.0	87.5	63.7	88.3	71.1	93.8
$Mean\_Std$	58.1	88.1	65.2	88.5	72.0	94.2
Max	57.7	88.3	64.8	88.8	72.5	94.2
BoW	36.9	71.9	47.9	80.0	53.4	85.3
FV	39.1	69.8	55.6	81.3	58.5	83.8
$VLAD$	45.3	77.3	57.4	84.7	64.7	89.2

Table 3. Comparison of different local feature aggregation methods on split 1 of UCF101 and HMDB51.

# of samples	Spatial CNNs (%)		Temporal CNNs (%)		Two-stream (%)	
	HMDB51	UCF101	HMDB51	UCF101	HMDB51	UCF101
3	52.5	85.6	54.9	82.4	64.6	91.6
9	56.1	87.4	62.2	87.7	70.9	93.5
15	56.9	88.2	64.4	88.5	72.3	93.8
21	57.1	88.1	64.8	88.6	71.8	94.1
25	57.7	88.3	64.8	88.8	72.5	94.2
Max	57.6	88.4	65.3	88.9	72.4	94.3

Table 4. Number of samples per video versus accuracy on split 1 of UCF101 and HMDB51.

4.1. From which layer(s) should the local features be extracted?

We conduct experiments using both VGG16 and Inception-BN to explore which layers are optimal for extracting the local features. The video-level action classification accuracies on split 1 of UCF101 using different layers are shown in Table 2.

Layer L-2 from Inception-BN and layer L-4 from VGG16 give the best performance. These are the final convolution layers in each network which suggests the following three reasons for their superior performance. First, the convolution layers have far fewer parameters compared to the fully connected layers and thus are less likely to overfit the training data that has false label assignment problem. Second, the fully connected layers do not preserve spatial information while the convolution layers do. Third, the later convolution layers encode more global (spatial) information than the earlier ones. We conclude that extracting the local features from the final convolution layers is the optimal choice. We believe this finding helps explain why several recent works [26, 1, 12] also choose the output of the final convolution layer for further processing.

Compared to the results of Wang et al. [23], from which we get the pre-trained networks, we can see that our approach do improve the performance on both spatial-net and temporal-net. However, the improvements from spatial networks are much larger. This larger improvement may be because that, in training local feature extractors, the inputs for spatial net are single frames while the input for temporal net are video clips with 10 stacked frames. Smaller inputs lead to larger chance of false label assignment hence larger performance gap compared to our global feature approach.

Previous work [27, 9, 26] on using local features from networks pre-trained using the ImageNet dataset show that combining features from multiple layers can improve the overall performance significantly. We investigated combining features from multiple layers but found no improvement. This difference shows that fine-tuning brings some new characteristics to the local features.

In the remaining experiments, we use the output of the global_pool layer from the Inception-BN network as it achieves the best performance.

4.2. What is the optimal aggregation strategy?

We consider six aggregation methods on split 1 of the the UCF101 and HMDB51 datasets.

Given n local features, each of which has a dimension of d, the six different aggregation methods are as follows:

- $Mean$ computes the mean value of the n local features along each dimension.
- Max selects the maximum value along each dimension.
- $Mean\_Std$, inspired by Fisher Vector encoding, computes the mean and standard deviation along each dimension.
- BoW quantizes each of the n local features as one of k codewords using a codebook generated through k-means clustering.
- $VLAD$ is similar to BoW but encodes the distance between each of the n local features and the assigned codewords.

- FV models the distribution of the local features using a Gaussian mixture model (GMM) with k components and computes the mean and standard deviation of the weighted difference between the n local features and these k components.

For those feature aggregation methods that require clustering, we project each local feature into 256 dimensions using PCA and set the number of clusters as 256. This is similar to what suggested in [26] except we don't break the local features into multiple subfeatures.

As can be seen in Table 3, maximum pooling (Max) achieves the best overall performance (two-stream network results). This result is different from that of [9] where mean pooling ($Mean$) performs better than maximum pooling (Max). It is also interesting that $Mean\_std$ consistently performs better than $Mean$. The more complicated encoding methods, BoW, FV and $VLAD$, all perform much worse than simple pooling. We conjecture that extracting a larger number of local features for each video and dividing the features into lower dimension subfeatures as in [26] would likely improve the performance of the more complicated methods. This would, however, incur excessive computational cost and limit practical application.

We use maximum pooling in the remainder of the experiments.

4.3. How densely should the local features be extracted?

We vary the number of local features extracted from each video between 3 and 25. We also experiment with using the maximum number (Max) by extracting features for every frame/clip (for optical flow, we use a sliding window with step size equal to 1). The videos in HMDB51 and UCF101 contain 92 and 185 frames on average, respectively.

The results in Table 4 show that, after a threshold of around 15, the number of sampled frames/clips does not have much of an effect on performance. Sampling 25 frames/clips achieves similar performance to using them all. This is consistent with the observations in [9] and is likely due to the high level of redundancy between frames. However, since the videos in UCF101 and HMDB51 are quite short and the datasets are small, these results should be taken with a bit of caution.

4.4. Comparison with state-of-the-art

Table 5 compares our best performance with the state-of-the-art. We improve upon TSN [23], which forms the basis of our approach, by around 3% and 1% on HMDB51 and UCF101, respectively. Our results are also much better than both traditional IDT based methods [10] and the original Two-stream CNNs [14]. In comparison to TLE [1] and Deep Quantization [12], our maximum pooling achieves similar performance to their more complex bilinear models and FV-VAE framework. We also show the results of fusing our prediction scores with those from MIFS [2] using late fusion. For HMDB51, the improvement from fusing with MIFS is very significant and is more than 3%. The improvement for UCF101 is smaller as the accuracy is already saturated.

	HMDB51	UCF101
IDT [20]	57.2	85.9
Two-stream [14]	59.4	88.0
MIFS [10]	65.1	89.1
C3D [15]	-	90.4
TDD [21]	65.9	91.5
Action Transformations [24]	62.0	92.4
LTC [17]	67.2	92.7
Depth2Action [29]	68.2	93.0
Key Volume Mining [28]	63.3	93.1
Two-stream Fusion [2]	69.2	93.5
TSN [23]	68.5	94.0
Deep Quantization [12]	-	95.2
TLE [1]	71.1	95.6
DOVF (ours)	71.7	94.9
DOVF + MIFS (ours)	75.0	95.3

Table 5. Comparison with state-of-the-art. Mean classification accuracy on UCF101 and HMDB51 over three splits.

5. Conclusion

We propose an effective method for deriving global video features from local features extracted using CNNs. We study a set of design choices such as which layer(s) to extract the features from, how to aggregate them and how densely to sample them. Based on a set of experiments on the UCF101 and HMDB51 datasets, we conclude that 1) the local features should be extracted from the final convolution layer; 2) maximum pooling works better than other feature aggregation methods including those which need further encoding; and 3) a sparse sampling of around 15 frames/clips per video is sufficient. While we propose plausible explanations for these conclusions, further investigation into DOVF is warranted. Also, the current two-stage approach only corrects the mistakes after it happens, we believe that a better way would be directly mapping a whole video into the global label, or so called end-to-end learning. Our future work will focus on these directions.

6. Acknowledge

This work was partially supported by the National Science Foundation under Grant No. IIS-1251187. This work was also supported in part by a National Science Founda-

[2] We download the the prediction scores of MIFS from here

tion CAREER grant, No. IIS-1150115, and a seed grant from the Center for Information Technology in the Interest of Society (CITRIS). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and CITRIS. We gratefully acknowledge the support of NVIDIA Corporation through the donation of the Titan X GPU used in this work.

References

[1] A. Diba, V. Sharma, and L. Van Gool. Deep Temporal Linear Encoding Networks. *arXiv preprint arXiv:1611.06678*, 2016.

[2] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *CVPR*, 2016.

[3] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, T. Tuytelaars, and L. Belgium. Modeling Video Evolution For Action Recognition. In *CVPR*, 2015.

[4] A. Gorban, H. Idrees, Y.-G. Jiang, A. Roshan Zamir, I. Laptev, M. Shah, and R. Sukthankar. THUMOS Challenge: Action Recognition with a Large Number of Classes. http://www.thumos.info/, 2015.

[5] M. Hoai and A. Zisserman. Improving Human Action Recognition using Score Distribution and Ranking. In *ACCV*, 2014.

[6] Y.-G. Jiang, Q. Dai, X. Xue, W. Liu, and C.-W. Ngo. Trajectory-Based Modeling of Human Actions with Motion Reference Points. In *ECCV*. 2012.

[7] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS Challenge: Action Recognition with a Large Number of Classes. http://crcv.ucf.edu/THUMOS14/, 2014.

[8] 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.

[9] Z. Lan, L. Jiang, S.-I. Yu, S. Rawat, Y. Cai, C. Gao, S. Xu, H. Shen, X. Li, Y. Wang, et al. CMU-Informedia at TRECVID 2013 Multimedia Event Detection. In *TRECVID 2013 Workshop*, volume 1, page 5, 2013.

[10] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj. Beyond Gaussian Pyramid: Multi-skip Feature Stacking for Action Recognition. In *CVPR*, 2015.

[11] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of Visual Words and Fusion Methods for Action Recognition: Comprehensive Study and Good Practice. *arXiv preprint arXiv:1405.4506*, 2014.

[12] Z. Qiu, T. Yao, and T. Mei. Deep Quantization: Encoding Convolutional Activations with Deep Generative Model. *arXiv preprint arXiv:1611.09502*, 2016.

[13] M. Sapienza, F. Cuzzolin, and P. H. Torr. Feature Sampling and Partitioning for Visual Vocabulary Generation on Large Action Classification Datasets. *arXiv preprint arXiv:1405.7545*, 2014.

[14] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *NIPS*, 2014.

[15] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *ICCV*, 2015.

[16] B. Varadarajan, G. Toderici, S. Vijayanarasimhan, and A. Natsev. Efficient Large Scale Video Classification. *arXiv preprint arXiv:1505.06250*, 2015.

[17] G. Varol, I. Laptev, and C. Schmid. Long-term Temporal Convolutions for Action Recognition. *arXiv preprint arXiv:1604.04494*, 2016.

[18] A. Vedaldi and A. Zisserman. Efficient Additive Kernels via Explicit Feature Maps. *PAMI*, 2012.

[19] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *CVPR*, 2011.

[20] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *ICCV*, 2013.

[21] L. Wang, Y. Qiao, and X. Tang. Action Recognition with Trajectory-Pooled Deep-Convolutional Descriptors. In *CVPR*, 2015.

[22] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards Good Practices for Very Deep Two-Stream ConvNets. *arXiv preprint arXiv:1507.02159*, 2015.

[23] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *ECCV*, 2016.

[24] X. Wang, A. Farhadi, and A. Gupta. Actions˜ Transformations. In *CVPR*, 2016.

[25] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Modeling Spatial-Temporal Clues in a Hybrid Deep Learning Framework for Video Classification. *arXiv preprint arXiv:1504.01561*, 2015.

[26] Z. Xu, Y. Yang, and A. G. Hauptmann. A Discriminative CNN Video Representation for Event Detection. In *CVPR*, 2015.

[27] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov. Exploiting Image-Trained CNN Architectures for Unconstrained Video Classification. *arXiv preprint arXiv:1503.04144*, 2015.

[28] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A Key Volume Mining Deep Framework for Action Recognition. In *CVPR*, 2016.

[29] Y. Zhu and S. Newsam. Depth2Action: Exploring Embedded Depth for Large-Scale Action Recognition. In *ECCVW*, 2016.