Temporal Action Detection with improved R-C3D

Jiayi Huang
Shanghai Jiao Tong University
Shanghai
sycamore2017@sjtu.edu.cn

Xincheng Yao
Shanghai Jiao Tong University
Shanghai
i-Dover@sjtu.edu.cn

Jintao Fan
Shanghai Jiao Tong University
Shanghai
fjt0324@sjtu.edu.cn

Abstract

We research the problem of temporal action detection in continuous,untrimmed video streams. This is a difficult task that requires extracting meaningful spatial-temporal features to capture actions, accurately localizing the start and end times of each action. We use a popular model, Region Convolutional 3D Network (R-C3D), which encodes the video streams using a three-dimensional fully convolutional network, and then generates candidate temporal regions containing actions, and finally classifies selected regions into specific actions. In many tasks, We typically assumes that training and test data are drawn from an identical distribution, however, the assumption does not always hold in practice. Such a distribution mismatch will lead to a significant performance drop. In the temporal action detection task, this distribution mismatch problem also exists. To address this problem, we propose to use domain adaptation technology and build it on the R-C3D model. Experiments on two popular action detection benchmarks (THUMOS 2014, ActivityNet v1.3) demonstrate the significant performance improvement achieved by our method. Particularly, R-C3D embedded with our domain adaptation module achieves average mAP 1.8% improvement on THUMOS 2014 dataset compared to the original model.

1. Introduction

In recent years, with the tremendous increase of video data, understanding human actions in videos is becoming fundamental issue in computer vision, owing to its various applications in security surveillance, human behavior analysis and many other areas. Specifically, temporal action detection is one of the challenging tasks. Temporal action detection, like object detection, belongs to video detection family. Temporal action localization requires the machine to not only classify the actions of interest but also localize the start and end time of every action instance, which means the detector must be able to find out the frames where the action event is happening and identify the category of the event. Many early studies developed hand-craft features to achieve action detection, such as DLSBP [1], ASM [2]and SDPM [13]. In the past few years, because of the development of convolutional neural networks (CNN), more studies have focused on deep features based action detection.

Most of existing CNN based temporal action detection methods can be categorized into two groups. The first group of methods use two-stage architectures consisting of proposal and classification [15] [18] [5], which first propose some video frames that may contain actions and then classify these frames into different action categories. The second group of methods pay more attention on one-stage architectures [8] [7] [6], motivated by the development of one-stage object detectors such as SSD, YOLO, etc. Two-stage models generally have higher detection performance, while one-stage models have better computation efficiency. In our work, the backbone model we use (R-C3D) belongs to two-stage detector family, which first generates candidate temporal regions containing actions, and finally classifies selected regions into special actions.

In many computer vision tasks, we always assumes that training and test data are drawn from an identical distribution, which, however, does not always hold in practice. Such a distribution mismatch will lead to a significant performance drop. This issue also exists in the temporal action detection task, to address this problem, we propose to add domain adaptation module like in the object detection task to the R-C3D model. we do full supervision in the source domain while unsupervision in the target domain. The domain shift could occur on image-level and instance-level, to address the domain shift, we incorporate two domain adaptation components on image level and instance level into the R-C3D model to minimize the H-divergence between two domains. Each domain adaptation is actually a domain classifier and a consistency regularization between the domain classifiers is implemented to minimize the domain discrepancy on both levels.

To summarize, the main work we do are as follows:

• We build R-C3D model for temporal action detection, we train and test the model on two popular action detection datasets (THUMOS 2014 and ActivityNet).

• We integrate the domain adaptation module into the R-C3D architecture, which includes image-level adaptation, instance-level adaptation and consistency regularization.

• Experiments on ActivityNet and THUMOS show that R-C3D with domain adaptation module can achieve significantly better action detection performance than the original R-C3D, we believe the domain adaptation method can also be expanded to other temporal action detectors such as PGCN [18].

2. Related Work

Temporal Action Detection. There is a long history of action recognition, or classifying trimmed video clips into fixed set of categories. Action detection is based on the action recognition and classification, but differently it needs to predict the start and end time of the actions within untrimmed and long videos. Early works apply hand-crafted features and sliding windows for temporal action localization. E.g., Yuan *et al.* [17] used SVM to encode the iDT features at each position and each resolution, followed by sliding window to localize the actions. Recently, frame-level deep features such as two-stream CNN [11], C3D [14] and I3D [14] have achieved both better performance and higher efficiency than hand-crafted features.

Nowadays, the main research of temporal action detection can be divided into: temporal action proposal and temporal action classification. The former focuses on investigating how to precisely localize video frames which contain actions, while the latter further classifies these actions into known classes. Once the localization of action proposals completes, the natural way for temporal action detection is to further classify the proposals into known action classes, making the process in two-stage manner. In two-stage manner, some work mainly focus on generating high-quality proposals, and the other explore how to better classify the proposals. However, most two-stage networks are trained separately, recently there are some attempts to jointly train the two-stage networks in an unified network, which is easier to train and also get higher performance. To further speed up temporal action detection, there have been several one-stage techniques being proposed recently. Our backbone model is an unified two-stage network, so the entire model is trained end-to-end from video frames.

Convolutional 3D Network. 3D ConvNets are more suitable for learning of spatial-time features compared to 2D ConvNets. The features learned by 3D ConvNet encapsulate information related to the targets, scenes, and actions in a video, making these features useful for different tasks without the need to fine-tune the model for each task. Du Tran *et al.* in [14] proposed deep C3D network and exploited 3D ConvNets in the context of large-scale supervised training datasets and modern deep architectures to achieve the best performance on different types of video analysis tasks. C3D is a good descriptor: it is generic, compact, simple, and efficient, so it can be used in many video analysis tasks. In the R-C3D model, the C3D is used as a shared feature extractor, then a proposal subnet and a classification subnet follow the C3D extractor to use the rich spatial-time features extracted by the 3D ConvNet.

Domain Adaptation. Domain Adaptation has been widely studied for image classification in computer vision. Conventional methods include domain transfer multiple kernel learning, subspace interpolation, convariance matrix alignment, *etc*. Recent works aim to improve the domain adaptability of deep neural networks. Different from those works, we focus on the temporal action detection problem, which is more challenging as both temporal location and category need to be predicted.Compared to the research in domain adaptation for classification, much less attention has been paid to domain adaptation for other computer vision tasks. Many works have been done in image recognition field such as semantic segmentation, fine-grained recognition, *etc*. [16] proposed to mitigate the domain shift problem of the deformable part-based model (DPM) by introducing an adaptive SVM, in [10] the authors use R-CNN model as feature extractor, then the features are aligned with the subspace alignment method. Recently, there are some works to focus on using domain adaptation in the video field, such as from images to videos [12], from 3D models [9], or from synthetic models [4].

3. Approach

In this section, we will first introduce the R-C3D model and the domain adaptation technology, and then elaborate the corresponding technical details of R-C3D framework equipped with the proposed domain adaption module.

3.1. R-C3D Framework

The R-C3D model mainly consists of three stages: (i) a shared 3D ConvNet to extract rich spatial-time feature hierarchies from a input video; (ii) a temporal proposal stage (iii) an action classification and refinement stage. The framework of R-C3D is motivated by the Faster R-CNN in objection detection, a key innovation in R-C3D is to extend 2D RoI pooling to 3D RoI pooling. The input to the extractor is a sequence of RGB video frames with dimension $\mathbb{R}^{3\times L\times H\times W}$. The input to the 3D ConvNet extractor is of variable length, a feature map $C_{conv5b} \in \mathbb{R}^{512\times \frac{L}{8}\times \frac{H}{6}\times \frac{W}{16}}$ is the final feature map as the output of the extractor. We use C_{conv5b} activations as the shared input to the proposal and classification subnets. The H and W of the frames are 112, and the number of frames L can be arbitrary. The R-C3D model is showed in the Figure 1.

The next stage following the extractor is temporal proposal subnet. The subnet predicts potential proposal segments with respect to anchor segments and a binary label indicating whether the predicted proposal contains an action or not. The anchor segments are pre-defined multi-scale windows, and each temporal location specifies K anchors, thus the total number of anchor segments is $\frac{L}{8} \times K$. The anchors can be seen as reference action segments for proposals at each temporal location, the number of anchors K is depended on the dataset. The shared feature map will first be filtered by a 3D convolutional filter with kernel size $3 \times 3 \times 3$, and then downsampled by a 3D max-pooling feature map with kernel size $1 \times \frac{H}{16} \times \frac{W}{16}$. The obtained feature map $C_{tpn} \in \mathbb{R}^{512\times \frac{L}{8}\times 1\times 1}$ is for predicting proposals with respect to these anchor segments. The 512-dimensional feature vector at each temporal location in C_{tpn} is used to predict a relative offset $\{\delta_{ci}, \delta_{li}\}$ to center location and the length of each anchor segment $\{c_i, l_i\}$,$i \in 1, ..., K$. (c_i means the center frame, l_i means the length of frames) and it also predicts the binary scores for each proposal to indicate an action or background. The proposal offsets and scores are predicted by using two $1 \times 1 \times 1$ convolutional layers to filter C_{tpn}.

The last stage is action classification and regression subnet. The three main things will be done in this subnet are: (i) selecting proposals from the previous stage; (ii) using 3D pooling to extract size fixed features from C_{conv5b}; (iii) action classification and boundary regression. Some action proposals obtained from the proposal subnet overlap with each other and some have a low action score, so The Non-Maximum Suppression (NMS) is first used in this subnet to eliminate some negative proposals. Next, The 3D RoI pooling is used to extract fixed-size features in order to use fully connected layers for further action classification and regression. Specifically, in 3D RoI pooling, an input feature map of size($l \times h \times w$) is divided into ($l_s \times h_s \times w_s$) sub-volumes each with approximate size ($\frac{l}{l_s} \times \frac{h}{h}_s \times \frac{w}{w_s}$), and then max pooling is performed inside each sub-volume. The output of the 3D RoI pooling is fed to a series of two fully connected layers. Here, the one is classification head for predicting binary scores, and the other is regression head for generating temporal offsets to refine the proposals.

The loss function of R-C3D model is combined with classification loss and regression loss.The softmax loss is used for classification and smooth $L1$ loss is used for regression loss. Specifically, the loss function is formulated by:

$$Loss = \frac{1}{N_{cls}} \sum_i L_{cls}(a_i, a_i^*) + \lambda \frac{1}{N_{reg}} \sum_i a_i^* L_{reg}(t_i, t_i^*) \tag{1}$$

where N_{cls} and N_{reg} is the number of proposals, λ is the loss trade-off parameter and is set to 1. a_i is the predicted probability of the proposal, a_i^* is the ground truth, $t_i = \{\delta\hat{c_i}, \delta\hat{l_i}\}$ is the predicted relative offset to proposals, $t_i^* = \{\delta c_i, \delta l_i\}$ is the coordinate transformation of ground truth segments to proposals.

3.2. Domain Adaptation

In the many computer vision tasks, we typically assumes that training and test data are drawn from an identical distribution, however, the assumpation does not always hold in practice. Such a distribution mismatch will lead to to significant performance drop. So if we can address the domain shift, the performance in test dataset will be greatly improved. We address the domain shift on two levels like the tackling process in object detection: (i) the image-level shift; (ii) the instance-level shift. The two domain adaptation components are based on H-divergence theory, and are implemented by learning a domain classifier in adversarial training manner. We will consider the unsupervised domain adaptation scenario: full supervision is given in the source domain while no supervision is available in the target domain. Thus the performance of detector will be improved in target domain. The domain adaptation module is showed in the Figure 2.

The H-divergence is designed to measure the divergence between two sets of samples with different distributions. We denote x a feature vector. The source domain sample is denoted as x_s and the target domain sample as x_t. we also denote by $h : x \to 0, 1$ a domain classifier, which aims to predict the source samples x_s to be 0, and target domain sample x_t to be 1. Suppose the H is the set of possible domain classifiers, the H-divergence defined the distance between two domains as follows:

$$d_H(S,T) = 2(1 - \min_{h\in H}(err_S(h(x)) + err_T(h(x)))) \tag{2}$$

where err_S and err_T are the prediction errors of $h(x)$ on source and target domain samples. In deep neural networks, we denote by f the network that produces x. To align the two domains, we need the networks f to output features that minimize the domain distance, formally:

$$\min_f d_H(S,T) = \max_f \min_{h\in H} err_S(h(x)) + err_T(h(x)) \tag{3}$$

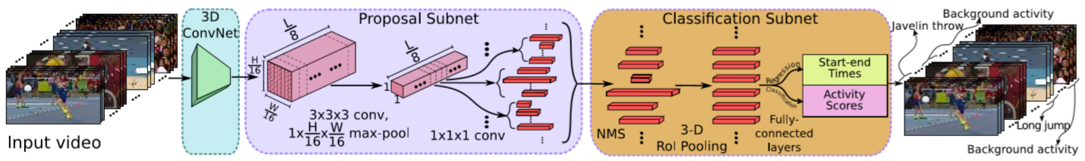

Figure 1. The architecture of R-C3D model defined in [15].

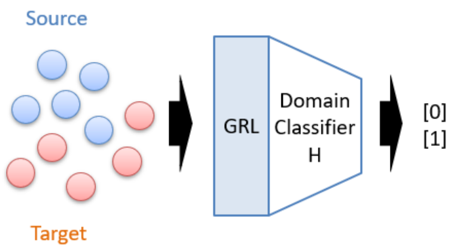

Figure 2. The domain adaptation module. The source domain sample is predicted to be 0, and the target domain sample is predicted to be 1.

It also means to maximize the prediction errors on domain classifier.To achieve this goal, we apply a gradient reversal layer (GRL)[3] to reverses the gradient signs during back-propagation.

Image-Level Adaptation: In the R-C3D model, the image level representation refers to the last feature map extracted by the C3D extractor. we denote the feature map as base_feat, and tgt_base_feat as target domain feature map. To eliminate the domain distribution mismatch, we will train a domain classifier as shown in the bottom left part of figure 3, which can be called imageDA head. Let us denote by D_i the domain label of the $i-th$ input buffer, with $D_i = 0$ for the source domain and $D_i = 1$ for the target domain. By denoting the output of the imageDA head as $p_i^{u,v}$ and using the cross entropy loss, the image-level loss can be written as:

$$L_{img} = -\sum_{i,u,v}[D_i log p_i^{(u,v)} + (1-D_i) log(1-p_i^{(u,v)})] \quad (4)$$

Instance-Level Adaptation: The instance-level representation refers to the RoI-based vectors, we denote as pooled_feat, and tgt_pooled_feat as target domain feature vector. we will train a instance level classifier which can be called instanceDA head. Let us denote the output of the instanceDA head for the $j-th$ region proposal in the $i-th$ input buffer as $p_{i,j}$. The instance-level adaptation loss can be written as:

$$L_{ins} = -\sum_{i,j}[D_i log p_{i,j} + (1-D_i) log(1-p_{i,j})] \quad (5)$$

Consistency Regularization: Enforcing consistency between the domain classifier on different levels helps to learn the cross-domain robustness. The consistency regularizer can be written as:

$$L_{cst} = \sum_{i,j} || \frac{1}{|I|} \sum_{u,v} p_i^{(u,v)} - p_{i,j} ||_2 \quad (6)$$

The loss is MSE loss, where $|I|$ denotes the total number of activations in a feature map.

3.3. Framework Overview

The overall network is show in Figure 3, consisting of baseline structure and three adaptation components. In our framework, the output of 3D convolutional filter C_{conv5b} consists of rich spatio-temporal features. We take these features as the input of image level domain classifier. Similar to image level adaptation,the input of the instance classifier is the ROI-based feature vectors,which is the output of proposal subnet.

The total loss is the combination of baseline prediction loss and three types of adaptation loss. The formula can be written as:

$$Loss = L_{RC3D} + \lambda(L_{img} + L_{ins} + L_{cst}) \quad (7)$$

where λ is the trade-off parameter for adaptation loss.

4. Experiment

To validate the performance of the proposed methods, we carry out our experiment on three datasets: THUMOS14, ActivityNet1-3v, HACS. Then, We separate our experiment into 2 parts: 1) self-domain adaptation 2)cross-domain adaptation. For self-domain adaptation, we carry out our method on THUMOS14 and ActivityNet1-3v, while for cross-domain adaptation, our experiments are based on ActivityNet1-3v and HACS. Finally, we evaluate the results by mAP at the IoU threshold k%(k=10,30,50,70)

4.1. self-domain adaptation

In this section, we separate the same datasets into source domain and target domain. With target domain unlabeled, we trained our model on both domain jointly, and compared with the model trained on source domain.

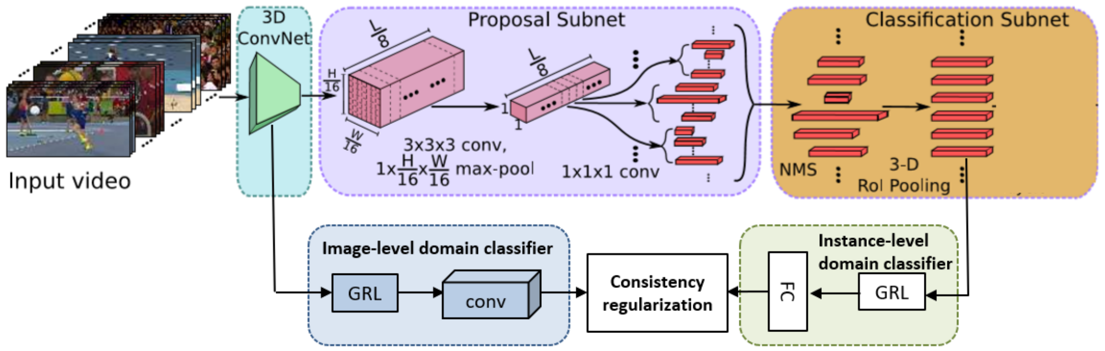

Figure 3. The overall network of our method. Based on architecture of R-C3D model, we design two domain classifiers on image-level and instance-level respectively, and further improved by consistency regularization. The input is source domain videos and unlabeled target domain videos.

THUMOS14	img	ins	cst	mAP@IoU			
				10	30	50	70
RC3D				51.7	48.8	33.9	15.3
DA-RC3D		✓		51.3	48.1	36.0	16.2
		✓	✓	53.3	50.1	36.8	16.6
	✓	✓	✓	51.5	49.2	34.6	15.3

ActivityNet	img	ins	cst	mAP@IoU			
				10	30	50	70
RC3D				10.4	8.3	6.3	4.0
DA-RC3D		✓	✓	10.7	8.8	6.6	4.5
	✓	✓	✓	10.7	8.6	6.3	3.9

Table 1. The mean average precision(mAP) for self-domain adaptation with different adaptation components. RC3D refers to the model trained on pretrained model and finetuned on source domain dataset. DA-RC3D is also trained on pretrained model and finetuned on both source and target domain.

Datasets: THUMOS'14 consists of 2,765 trimmed training videos and 200 and 213 untrimmed videos in the validation and test sets respectively. Most untrimmed videos contain a single activity covering a great deal of the video. There are also some videos containing multiple activities. In this experiment, we mix the validation set and testing set as our dataset.Then we hold out 20% videos for evaluation, and divide other videos to source domain and target domain equally. For target domain dataset, we remove activity labels as no supervision.

ActivityNet1-3v consists of 10,024 training videos, 4,926 validation videos and 5,044 testing videos, all of them are untrimmed videos from 200 different sport activities. The videos of ActivityNet is longer and more complex than THUMOS'14, which means it is more challenge to locate the activity. Since the ground truth of testing set is incomplete, we only make use of the training set and validation set. We divide the training set to source domain and target domain equally, and evaluate our model on validation set.

Experimental Setup: In the experiment on THUMOS'14, we use the pretrained model that trained on Sports-1M and finetuned on UCF101.The scale values of the anchor segments K we use for this dataset are [2,4,5,6,8,9,10,12,14,16]. The learning rate is 0.0001. Since the duration of activities is longer in ActivityNet1-3v, we choose the scale values of the anchor segments K as follows:[1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56, 64]. We also use pretrained model, which trained on activitynet after 30000 iterations, and freeze the first two convolutional layers in our model to accelerate our experiment.

To analyze the efficiency of each domain adaptation component, we gradually increase the number of components and compared with the baseline model. If the IoU between the predicted proposal and the ground truth activity is more than threshold, the proposal is considered correct.

Results: The performance of different methods is presented in Table 1. The first line is trained on baseline model, where the input is source domain dataset. DA-RC3D refers to our model and the input is the source domain dataset and the unlabeled target domain dataset. The table shows that if we combine instance-level adaptation component and consistency component, we can achieve average mAP 1.8% improvement on THUMOS'14 dataset and 0.4% improvement on ActivityNet1-3v dataset. We can see that image-level adaptation component does not perform well in self-domain experiment, which may be due to the fact that the source domain and target domain have the similar distribution.

Cross Domain	mAP@IoU				
	10	20	30	40	50
RC3D	15.826	13.417	11.309	8.980	7.241
DA-RC3D	16.151	13.936	11.692	9.408	7.248

Table 2. The mean average precision(mAP) for cross-domain adaptation with three adaptation components. RC3D is trained with training set of ActivityNet1-3v and evaluate on validation set of HACS. DA-RC3D is trained on training set of ActivityNet1-3v combined with unlabeled training set of HACS and evaluate on validation set of HACs.

4.2. cross-domain adaptation

In this section, we do a further research to investigate what will happen if the source domain and the target domain belong to different datasets, which means there must be a large distribution mismatch between source and target. The length of the activity, the form of the activity and the background can vary greatly.

Datasets: In this experiment, we use the training set of the ActivityNetv1-3 as the source domain and the training datasets of HACs as target domain. Finally, we evaluate our model on the validation datasets of HACs. HACs is a large-scale video datasets, which contains 140,000 complete segments on 50,000 videos. The classes of HACs is totally the same with ActivityNet, while it is more challenging. We remove the labels of training set and take it as unlabeled target domain.

Experimental Setup: We use pretrained model in this experiment, which is the same with ActivityNet. Since the format of videos is similar to ActivityNet, we choose the scale values of the anchor segments K for HACS as follows:[1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28,32, 40, 48, 56, 64]. The learning rate is 0.0001. Because the source domain and target domain are from different datasets, there is a huge distribution mismatch between these two domains. Thus, we use all parts of adaptation components in cross-domain experiment.

Results: The performance of cross-domain experiment is shown in Table 2. The first line represents the result trained only with source domain dataset on baseline model. The second line represents the result trained with image-level, instance-level and consistency component. The table shows that if we make use of the unlabeled datasets, we can achieve average mAP 0.3324% improvement.

5. Conclusion

In this paper, we mainly explore the application of domain adaptation technology in the field of video, specifically, we focus on the temporal action detection task. We use the R-C3D model as our base action detection framework, which is a widely used temporal action detection model. Then, we incorporate the domain adaptation module including image-level adaptation, instance-level adaptation and consistency regularization to the R-C3D framework. We validate the performance of our DA-RC3D on three datasets and we separate our experiment into 2 parts: 1) self-domain adaptation 2) cross-domain adaptation. The experiment results show that the DA-RC3D can achieve average mAP 1.8% improvement on THUMOS'14 dataset and 0.4% improvement on ActivityNet1-3v dataset compared to the base R-C3D model. In the field of temporal action detection, the R-C3D model is no longer the state-of-art model, so the further work can focus on incorporating the domain adaptation module to current state-of-art action detection models. We believe that these models can perform better with the domain adaptation module.

References

[1] Duchenne, O. Laptev, I. Siviv, J. B. F., and J. Ponce. Automatic annotation of human actions in video. *ICCV*, 2009. 1

[2] A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. *CVPR*, 2011. 1

[3] Y. Ganin and V. S. Lempitsky. Unsupervised domain adaptation by backpropagation. *ArXiv*, abs/1409.7495, 2014. 4

[4] H. Hattori, V. N. Boddeti, K. M. Kitani, and T. Kanade. Learning scene-specific pedestrian detectors without real data. *CVPR*, 2015. 2

[5] J. Li, X. Liu, Z. Zong, W. Zhao, M. Zhang, and J. Song. Graph attention based proposal 3d convnets for action detection. *AAAI*, 2020. 1

[6] X. Li, T. Lin, X. Liu, C. Gan, W. Zuo, C. Li, X. Long, D. He, F. Li, and S. Wen. Deep concept-wise temporal convolutional networks for action localization. *CVPR*, 2019. 1

[7] Q. Liu and Z. Wang. Progressive boundary refinement network for temporal action detection. *AAAI*, 2020. 1

[8] F. Longy, T. Yaoz, Z. Qiuy, X. Tiany, J. Luox, and T. Meiz. Gaussian temporal awareness networks for action localization. *CVPR*, 2019. 1

[9] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3d models. *ICCV*, 2015. 2

[10] A. Raj, V. P. Namboodiri, and T. Tuytelaars. Subspace alignment based domain adaptation for rcnn detector. *BMVC*, 2015. 2

[11] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *NIPS*, 2014. 2

[12] K. Tang, V. Ramanathan, L. Fei-Fei, and D. Koller. Shifting weights: Adapting object detectors from image to video. *NIPS*, 2012. 2

[13] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. *CVPR*, 2013. 1

[14] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. *CVPR*, 2015. 2

[15] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. *CVPR*, 2017. 1, 4

[16] J. Xu, S. Ramos, D. Vazquez, and A. M. Lopez. Domain adaptation of deformable part-based models. *TPAMI,36(12):2367–2380*, 2014. 2

[17] J. Yuan, B. Ni, X. Yang, and A. Kssim. Temporal action localization with pyramid of score distribution features. *CVPR*, 2016. 2

[18] R. Zeng, W. Huang, M. Tan, Y. Rong, P. Zhao, J. Huang, and C. Gan. Graph convolutional networks for temporal action localization. *ICCV*, 2018. 1, 2