Relational Prototypical Network for Weakly Supervised Temporal Action Localization

Linjiang Huang,[1,3] **Yan Huang,**[1,3] **Wanli Ouyang,**[4] **Liang Wang**[1,2,3*]
[1]Center for Research on Intelligent Perception and Computing (CRIPAC)
National Laboratory of Pattern Recognition (NLPR)
[2]Center for Excellence in Brain Science and Intelligence Technology (CEBSIT)
Institute of Automation, Chinese Academy of Sciences (CASIA)
[3]University of Chinese Academy of Sciences (UCAS)
[4]University of Sydney
SenseTime Computer Vision Research Group, Australia
linjiang.huang@cripac.ia.ac.cn, {yhuang, wangliang}@nlpr.ia.ac.cn, wanli.ouyang@sydney.edu.au

Abstract

In this paper, we propose a weakly supervised temporal action localization method on untrimmed videos based on prototypical networks. We observe two challenges posed by weakly supervision, namely action-background separation and action relation construction. Unlike the previous method, we propose to achieve action-background separation only by the original videos. To achieve this, a clustering loss is adopted to separate actions from backgrounds and learn intra-compact features, which helps in detecting complete action instances. Besides, a similarity weighting module is devised to further separate actions from backgrounds. To effectively identify actions, we propose to construct relations among actions for prototype learning. A GCN-based prototype embedding module is introduced to generate relational prototypes. Experiments on THUMOS14 and ActivityNet1.2 datasets show that our method outperforms the state-of-the-art methods.

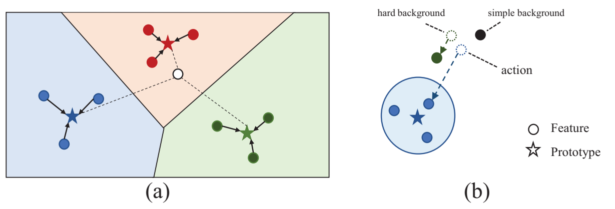

Figure 1: (a) Illustration of the prototypical network. It represents each class as a prototype, distance is employed as the measure of similarity between feature and prototype. (b) We employ a clustering loss for pushing the features of actions to the corresponding prototypes. Although the hard backgrounds, which are similar to actions, would be also pushed to the prototype, it would not be as close as actions to the prototype because of the dissimilarity.

Introduction

Temporal action localization in videos has been applied in various fields (Sun et al. 2015; Sultani, Chen, and Shah 2018). This task aims to localize action instances from untrimmed videos in the temporal dimension. Most existing methods are trained in a fully supervised way where frame-level annotations are provided. However, such a requirement of frame-level annotations does not well suit real-world applications since densely annotating large-scale videos is expensive and time-consuming. Moreover, accurate frame-level annotation is challenging even for human beings, further increases the difficulty of annotation.

To address these difficulties, weakly supervised methods (Wang et al. 2017a; Nguyen et al. 2018; Paul, Roy, and Roy-Chowdhury 2018) have been developed with only video-level labels, which are much easier to manually annotate. Because of the lack of frame-level annotations, weakly supervised methods may face two challenges. The first challenge is how to distinguish actions from backgrounds, *i.e.*, action-background separation. To alleviate this problem, many existing methods leverage auxiliary information such as temporal attention weights (Nguyen et al. 2018; Yuan et al. 2019) and hope the temporal attention weights could focus on the foreground of actions.

However, since weakly supervised methods are usually built upon classifiers, there is a contradiction between classification and detection, that is classifiers always focus on discriminative snippets but detectors should discover the whole action instance without omission (Zhong et al. 2018). To tackle this problem, some methods force the models to focus on different parts of videos. Nevertheless, it is difficult to accurately distinguish actions from backgrounds for both classifier and attention module because of the complex backgrounds, and thus degenerates the performance. Recently, Liu *et al.* (Liu, Jiang, and Wang 2019) propose to use pseudo videos for further addressing this problem. They generate pseudo videos by applying thresholding on the optical flow intensity and label them with a background class. However, this process is time-consuming and may over-segment some frames of actions into the pseudo videos.

Actually, we argue that it's unnecessary to force the model to focus on different parts of the video. If we can make the

[*]Corresponding author.

snippet-level features of the same action as similar as possible while separating actions from backgrounds, it's natural that the snippets of the same action would have similar scores of classification and thus detect complete action instances. This motivates us to find a way that can learn intra-compact representations and separate actions from backgrounds only by the original videos.

Even we can achieve this well, one more problem we have to consider, *i.e.*, action-action separation. Most existing methods just use a simple classifier to deal with it. However, different from action recognition methods (Simonyan and Zisserman 2014; Carreira and Zisserman 2017), which deal with the trimmed videos with a single action, temporal action localization is for videos where various actions may occur. So how to explicitly capture the co-occurrence of actions and accurately distinguish them is another challenge. Considering visual relationship has been well used for various tasks (Wang et al. 2017b; Yang et al. 2018), a solution is to consider the relationships between different actions.

It's a good way to achieve our target by introducing prototypical networks (Snell, Swersky, and Zemel 2017). As shown in Fig.1(a), this model is design for classification, which can be used for action-background and action-action separation. Besides, it represents each action class as a prototype, makes it possible to globally capture relations among actions. Nevertheless, the original prototypical network only focuses on the separateness between categories without considering relations, so it still faces the two challenges.

We thus propose a novel relational prototypical network. To construct relations among actions for action-action separation, we first design a co-occurrence GCN with a co-occurrence matrix, which is generated in a data-driven way. Based on the co-occurrence GCN, a prototype embedding module is devised for generating relational prototypes. Instead of regarding the prototypes as independent individuals, the learned prototypes are inter-dependent, thus can effectively assist our method in identifying actions. Following the prototypical networks, the distance between feature and prototype is employed as the measure of similarity. The snippet-wise similarities are then pooled over time with temporal attention into a video-level category similarity.

To separate actions from backgrounds and learn intra-compact features, a clustering loss is adopted as shown in Fig.1(b). The clustering loss can push the features of actions to their corresponding prototypes and thus generates clustered features, which can help in detecting complete action instances. Meanwhile, the features of actions would be separated from those of backgrounds because of the dissimilarity. Besides, we develop a post-processing module, namely similarity weighting module for further filtering out backgrounds. A prototype updating strategy is devised to enforce the learned prototypes closer to the true cluster centers of a video, ensuring the weights for similarity meaningful.

Our method outperforms the state-of-the-art methods on two benchmark datasets THUMOS14 (Idrees et al. 2017) and ActivityNet1.2 (Caba Heilbron et al. 2015), demonstrating the effectiveness of our method. In summary, our contributions are three-fold: 1) A prototypical network is utilized with a clustering loss for separating actions from backgrounds. 2) A co-occurrence GCN based prototype embedding module is proposed to explicitly capture relations among actions. 3) Our method obtains the state-of-the-art results on two important benchmarks of action localization.

Related Work

Fully supervised temporal action localization. Many recent fully supervised methods (Yuan et al. 2016; Chao et al. 2018) adopt a two-stage pipeline including proposal generation and classification. Namely, they first produce class-agnostic proposals from videos and then classify each proposal individually. For these methods, it is natural to improve the quality of proposals (Lin et al. 2018) and learning more robust and accurate classifiers (Shou et al. 2017; Zhao et al. 2017). Besides, some other methods (Lea et al. 2017; Yuan et al. 2017) focus on generating snippet-wise action labels, and then use these labels to predict the temporal action boundaries. But all these methods require frame-level annotations for both proposal generation and classifier training.

Weakly supervised temporal action localization. To alleviate the requirement for frame-level annotations, Wang *et al.* (Wang et al. 2017a) first propose to only use video-level category labels for temporal action localization.

To address the challenge of action-background separation, some methods leverage attention mechanism (Nguyen et al. 2018) or marginalized average aggregation (Yuan et al. 2019) to focus on discriminative snippets of actions and fuse salient snippet-level features into a video-level feature which is fed into a following classifier. However, there is a contradiction between classification and detection, which makes these methods unable to capture complete action instances. To deal with this problem, some methods force the models to focus on different parts of videos by using masks (Singh and Lee 2017), step-by-step erasion (Zhong et al. 2018) and diversity loss (Liu, Jiang, and Wang 2019). Recently, Liu *et al.* (Liu, Jiang, and Wang 2019) propose to utilize pseudo videos that are generated using static clips and labeled with a new background class. Nevertheless, generating pseudo videos is time-consuming and may over-segment some frames of actions into the pseudo videos.

As for the second challenge of action-action separation, most existing methods only employ a simple classifier to deal with it. The most relevant work is W-TALC (Paul, Roy, and Roy-Chowdhury 2018), it uses a co-activity similarity loss to enforce the feature similarity between instances of the same class. However, it only considers the correlations between videos of the same action, ignoring the relationships among actions. Moreover, the co-activity similarity loss only imposes a pair-wise constraint, which may be insufficient in learning better representations.

Proposed Method

In this section, we elaborate on the proposed method as shown in Fig.2. Before going into the details of our method, let us define the notations and problem statement formally.

Problem definition. Let $V = \{v_t\}_{t=1}^{L}$ be a video with variable temporal durations, where L denotes the temporal length. Assume that we have a set of N training videos

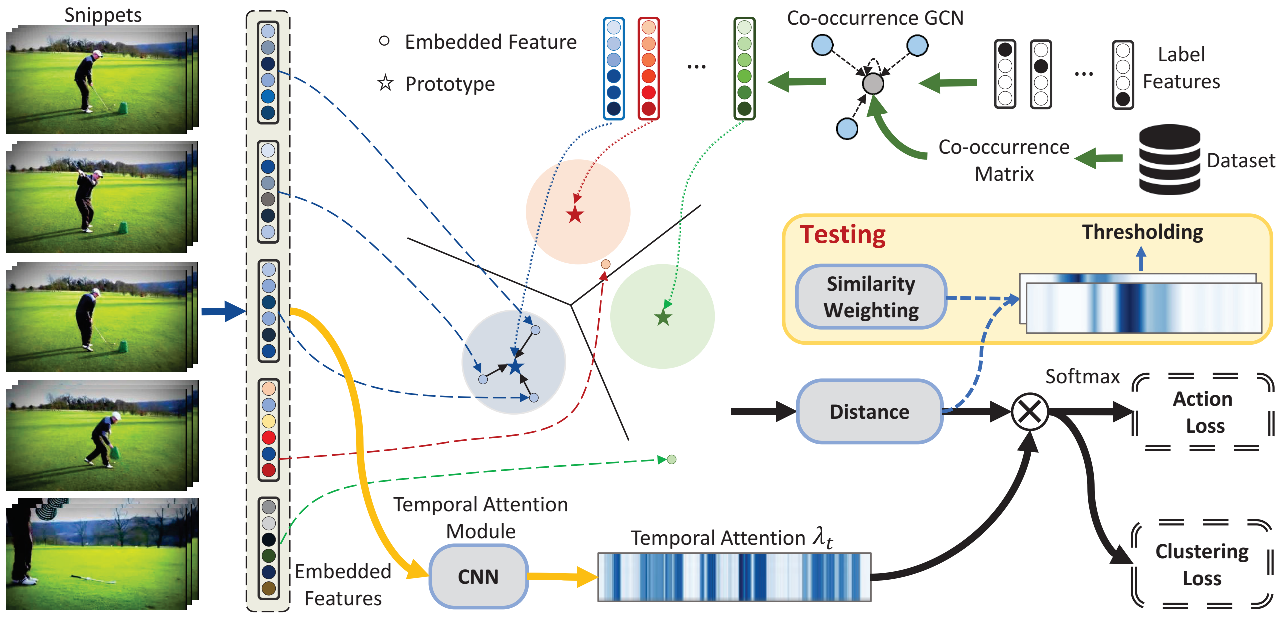

Figure 2: The overview of our method. During training, the snippets are fed to the feature extraction module followed by the feature embedding module for generating embedding features. Meanwhile, the prototype embedding module takes label features as input and outputs relational prototypes. The distances between features and prototypes are employed as the measure of similarity. The snippet-wise similarities are pooled over time with temporal attention into a video-level category similarity. We impose two losses, *i.e.*, action loss and clustering loss on it for learning separated and clustered representations. During testing, we employ the similarity weighting module for further filtering out backgrounds.

$\{V_i\}_{i=1}^N$ which are denoted by the corresponding activity labels $\{\boldsymbol{y}_i\}_{i=1}^N$, where $\boldsymbol{y} \in \mathcal{Y} = \{0,1\}^C$ is a C-dimensional binary vector indicates the presence/absence of actions. During test time, we wish to predict a set of action instances $\{c_j, s_j, e_j, q_j\}$, where c_j denotes the predicted action class, s_j and e_j represents the start time and end time of the instance, and q_j shows the confidence score of the instance.

Relational Prototypical Network

Our Relational Prototypical Network (RPN) mainly consists of six parts, namely feature extraction module, feature embedding module, prototype embedding module, prototype matching module, temporal attention module and similarity weighting module, which are detailed as follows.

Feature extraction module. Following previous methods (Nguyen et al. 2018; Liu, Jiang, and Wang 2019), we mainly focus on two state-of-the-art frameworks, *i.e.*, UntrimmedNet (Wang et al. 2017a) and I3D (Carreira and Zisserman 2017), for extracting high-level representations of motion and appearance of the input video. Given an input video V, two snippet-wise features $\boldsymbol{X}^r \in \mathbb{R}^{T\times D}$ and $\boldsymbol{X}^o \in \mathbb{R}^{T\times D}$ are extracted by the pre-trained feature extraction module, where T denotes the number of snippets and D denotes the dimension of features. $\boldsymbol{X}^r$ and $\boldsymbol{X}^o$ represent the RGB and optical flow features respectively. For simplicity, we use $\boldsymbol{X}$ to represent them in the rest of this paper. Note that, we do not fine tune the feature extraction module during training.

Feature embedding module. Since our target is to separate one action from other action classes and complex backgrounds, it is desired to utilize a task-specific module, *i.e.*, feature embedding module, for learning a new set of features. We adopt a multi-layer temporal convolutional network, which is interleaved with activation function and dropout operation, to achieve this. The network can be formalized for the video feature $\boldsymbol{X}$ as follows:

$$\boldsymbol{X}_e = \mathcal{F}(\boldsymbol{X}, \boldsymbol{W}_{emb}) \tag{1}$$

where $\mathcal{F}(\cdot,\cdot)$ represents the embedding network, $\boldsymbol{W}_{emb}$ is the corresponding parameters, $\boldsymbol{X}_e \in \mathbb{R}^{T\times E}$ is the learned embedding, and E is the dimension of embedded features.

Prototype embedding module. Prototypes are very important in our framework. They can be learned from the whole dataset, and thus allows us to globally capture relationships between action categories, rather than the pairwise fashion in W-TALC (Paul, Roy, and Roy-Chowdhury 2018). When only considering action-action separation, prototypes should be separated as far as possible. However, considering the relations among actions, it is desired that the distances of related prototypes are smaller than the ones of unrelated prototypes, such that when one action occurs the related action is likely to occur. In other words, the co-occurrence of two related classes can be measured by the distance between their prototypes in the embedded space. This motivates us to find a way for capturing the relationships among actions.

In this paper, we propose a co-occurrence graph convolutional network to achieve this. Our motivations are mainly two-fold. First, graphs have been proven effective in representing relations. GCN propagates information along edges of connected vertices, makes the actions within connected components inter-dependent. Second, graph convolution operation is actually a process of Laplacian smoothing (Li, Han, and Wu 2018), we can benefit from its nature to push the semantic-related prototypes closer. Now the problem is how to construct an adjacency matrix $\boldsymbol{A}$ for graph convolution because no ground truth is given. Actually, the element a_{ij} of matrix $\boldsymbol{A}$ can be viewed as the relation between the *i*-th action and the *j*-th action. We thus denote the adjacency matrix $\boldsymbol{A}$ as co-occurrence matrix in the rest of this paper.

Following the previous methods (Xue et al. 2011; Chen et al. 2019), the co-occurrence matrix $\boldsymbol{A} \in \mathbb{R}^{C \times C}$ can be obtained as:

$$a_{ij} = \frac{N_{i \cap j}}{N_i} \tag{2}$$

where element a_{ij} is the weight assigned to the edge (i, j). $N_{i \cap j}$ represents the number of co-occurrence of action i and j, and N_i denotes the occurrence times of action i.

We add a self connection to the co-occurrence matrix, that is, $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_C$. To represent the differences between actions, we utilize different weights for neighbor connection and self connection as follows:

$$\boldsymbol{H}^{s+1} = \sigma(\hat{\boldsymbol{D}}^{-1}(\boldsymbol{A}\boldsymbol{H}^{s}\boldsymbol{W}_1 + \boldsymbol{H}^{s}\boldsymbol{W}_2)) \tag{3}$$

where $\hat{\boldsymbol{D}} \in \mathbb{R}^{C \times C}$ is the degree matrix defined as $\hat{d}_{ii} = \sum_j \hat{a}_{ij}$, and $\{\boldsymbol{W}_1, \boldsymbol{W}_2\}$ are trainable parameters. $\sigma(\cdot)$ is the activation function. Based on the co-occurrence matrix, a co-occurrence GCN is employed to project the features from label space ($\in \mathbb{R}^F$) to feature space ($\in \mathbb{R}^E$) as follows:

$$\boldsymbol{P} = \mathcal{G}(\boldsymbol{L}, \boldsymbol{A}, \boldsymbol{W}_g) \tag{4}$$

where $\boldsymbol{W}_g$ are the trainable parameters, $\boldsymbol{L} = \{\boldsymbol{l}_i\}_{i=1}^{C}$ are the input features, and $\boldsymbol{P} = \{\boldsymbol{p}_i\}_{i=1}^{C}$ are the learned prototypes, C is the number of categories.

Prototype matching module. Following the prototypical network (Snell, Swersky, and Zemel 2017), distance is used to measure the similarity between feature and prototype, the classification thus turns into a process of prototype matching. The class-wise similarity for each snippet is:

$$s_{tj} = -\|\boldsymbol{x}_e^t - \boldsymbol{p}_j\|_2^2 \tag{5}$$

where $\boldsymbol{x}_e^t$ represents the embedding feature of the *t*-th snippet, $\boldsymbol{p}_j$ denotes the prototype corresponding to the *j*-th action, and $\|\cdot\|_2$ is L_2-norm of vector. Then $\boldsymbol{s}_t$ is forwarded to a softmax along the category dimension, yielding probability distribution at each time location.

$$\widetilde{s_{tj}} = p(\boldsymbol{x}_e^t \in c_j | \boldsymbol{x}_e^t) = \frac{e^{\gamma_c s_{tj}}}{\sum_{k=1}^{C} e^{\gamma_c s_{tk}}} \tag{6}$$

where γ_c is a hyper-parameter that control the hardness of probability assignment. Similar with other methods (Nguyen et al. 2018; Paul, Roy, and Roy-Chowdhury 2018; Yuan et al. 2019), $\widetilde{\boldsymbol{s}_t}$ can be viewed as the Class Activation Sequence (CAS) for localizing action instances.

Temporal attention module. Attention module is essential for addressing the weakly supervised task. The attention module $\Phi(\cdot, \cdot)$ produces class-agnostic attention weight λ_t for each embedding feature $\boldsymbol{x}_e^t$ as:

$$\lambda_t = \frac{e^{\Phi(\boldsymbol{x}_e^t, \boldsymbol{W}_{att})}}{\sum_{k=1}^{T} e^{\Phi(\boldsymbol{x}_e^k, \boldsymbol{W}_{att})}} \tag{7}$$

where $\boldsymbol{W}_{att}$ are trainable parameters. According to the attention weights, all the snippet-wise unnormalized scores $\boldsymbol{s}_t$ are fused into a video-level score $\bar{\boldsymbol{s}}$ by a global weighted average pooling. Then we perform a softmax along the category dimension on it:

$$\boldsymbol{p} = softmax(\bar{\boldsymbol{s}}) \qquad \bar{\boldsymbol{s}} = \sum_{t=1}^{T} \lambda_t \boldsymbol{s}_t \tag{8}$$

where $\boldsymbol{p} \in \mathbb{R}^C$ is the class distribution for the whole video. The classification loss is cross-entropy between the predicted $\boldsymbol{p}$ and ground-truth, which can be represented as:

$$\mathcal{L}_{cls} = -\sum_{i=1}^{C} \hat{y}_i log p_i \tag{9}$$

where $\hat{y}_i$ is the element of the normalized ground-truth vector $\hat{\boldsymbol{y}} = \boldsymbol{y} / \sum_c y_c$.

The prototype matching module can be viewed as a classifier where the hyperplanes for classification are generated by prototypes. However, it only focuses on the inter-class separateness, some hard backgrounds such as context snippets (Liu, Jiang, and Wang 2019) may also have high classification scores. Moreover, the temporal attention weights may only focus on discriminative snippets, this makes the classifier unable to accurately classify the whole features of the action and thus degenerates the detection performance. In light of this, a clustering loss is adopted:

$$\mathcal{L}_{clu} = -\delta\left(\sum_{i=1}^{C} y_i = 1\right) \sum_{i=1}^{C} y_i \bar{s}_i \tag{10}$$

where $\delta(\cdot)$ denotes a conditional expression which equals to 1 when condition is true, otherwise 0. To avoid breaking the inter-class separateness, it works when only one class of action occurs. The clustering loss pushes the features of actions to their corresponding prototypes, which can result in intra-compact features. Although the hard backgrounds, such as contexts would be also pushed to the prototype, it would not be as close as actions to the prototype because of the dissimilarity. Therefore, the features of actions would be separated from those of backgrounds. Finally, we combine the classification loss with the clustering loss as:

$$\mathcal{L} = \mathcal{L}_{cls} + \alpha \mathcal{L}_{clu} \tag{11}$$

where α is coefficient to control the extent of compactness.

Similarity weighting module. Based on the well learned representations, which are clustered and separated, an effective way can be employed to weight the similarity $\widetilde{s_{tj}}$ (refers to Eq.6) for further filtering out backgrounds. Given the unnormalized similarity s_{tj} (refers to Eq.5) of prototype $\boldsymbol{p}_j$ and embedded feature $\boldsymbol{x}_e^t$, we feed it to a softmax along the temporal dimension:

$$\widehat{s_{tj}} = \frac{e^{\gamma_t s_{tj}}}{\sum_{t=1}^{T} e^{\gamma_t s_{tj}}} \tag{12}$$

Algorithm 1 Prototype updating.

```
Input: $\boldsymbol{p}_j$: $j$-th prototype, $\{\boldsymbol{x}^t_e\}_{t=1}^{T}$: embedded features,
    $n$: iteration, $k$: number of selection, $\lambda$: moving rate
Output: $\widehat{s_{tj}}^n$: updated scores
 1: initialize $\boldsymbol{p}^0_j = \boldsymbol{p}_j$
 2: for $m \leftarrow 1, 2, ..., n$ do
 3:     calculate similarity: $s^m_{tj} = -\|\boldsymbol{x}^t_e - \boldsymbol{p}^{m-1}_j\|^2_2$
 4:     select top $k$: $\bar{\boldsymbol{s}}^m_j, idx = topk(\boldsymbol{s}^m_j)$
 5:     normalization: $\widehat{s_{tj}}^{norm} = e^{\gamma_t \bar{s}^m_{tj}} / \sum_{t=1}^{k} e^{\gamma_t \bar{s}^m_{tj}}$
 6:     initialize new score: $\widehat{\boldsymbol{s}_j}^m = zeros(\boldsymbol{s}^m_j)$
 7:     scatter: $\widehat{\boldsymbol{s}_j}^m = scatter(\widehat{\boldsymbol{s}_j}^{norm}, idx)$
 8:     update: $\boldsymbol{p}^m_j = \boldsymbol{p}^{m-1}_j + \lambda \sum_t \widehat{s_{tj}}^m (\boldsymbol{x}^t_e - \boldsymbol{p}^{m-1}_j)$
 9: end for
10: calculate similarity: $s_{tj} = -\|\boldsymbol{x}^t_e - \boldsymbol{p}^n_j\|^2_2$
11: normalization: $\widehat{s_{tj}}^n = e^{\gamma_t s_{tj}} / \sum_{t=1}^{T} e^{\gamma_t s_{tj}}$
12: return $\widehat{s_{tj}}^n$
```

The final score for localization is $\varphi_{tj} = \widehat{s_{tj}}\widetilde{s_{tj}}$.

Nevertheless, the prototypes are learned based on the whole dataset, which may not fit in a specific video. Especially, for the case that multiple actions occur in a snippet, the prototypes are not consistent with the cluster centers of features, this operation may degenerate the performance. To address this problem, we propose a **prototype updating strategy** to make the prototypes closer to the cluster centers of a video, which is shown in Algorithm.1. The prototype updating strategy based on a precondition that the prototypes are close to the cluster centers. We thus use the neighbor features of the prototype to update itself. After updating, the prototypes are closer to the cluster centers, results in meaningful weighting scores. The final score can be turned into $\varphi_{tj} = \widehat{s_{tj}}^n\widetilde{s_{tj}}$. Note that, the above operations should be based on well-learned representations, so we only adopt it during testing. We will analyze this in the experiment.

Experiments

Datasets

THUMOS14. We use the subset from THUMOS14 that offers frame-level annotations for 20 classes. We train the models on 200 untrimmed videos of the validation set and evaluate it on 212 untrimmed videos from the test set.

ActivityNet1.2. This dataset covers 100 classes and has 4,819 videos for training, 2,383 for validation and 2,480 for test. We use the training set to train our model and the validation set to evaluate.

Implementation Details

Our method is implemented with PyTorch (Paszke et al. 2017). During training, we loop through each video in the current training batch and accumulate the gradient to deal with variable video length. We use Adam (Kingma and Ba 2014) to optimize the model. The learning rates are set as 0.0001 for both datasets. The training procedure stops at 1000 epochs.

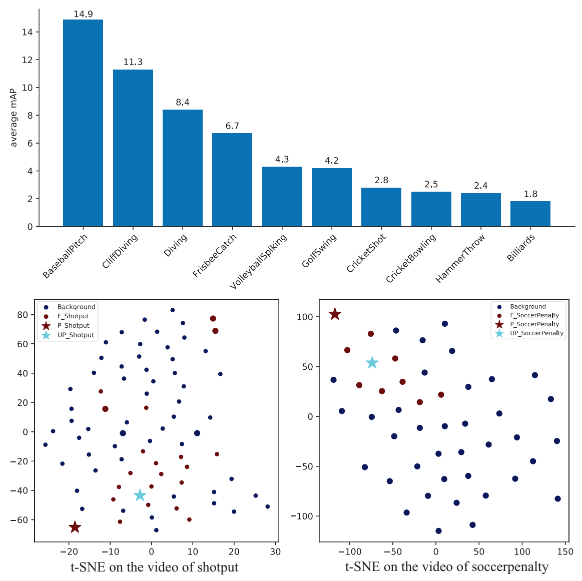

Figure 3: **Top:** Class-specific gain resulting from building relations of actions. Performance differences between our full model and the one with random initialization are shown. **Bottom:** The learned embedding features (F_*), prototypes (P_*) and updated prototypes (UP_*).

During testing, we utilize the learned model to localize action instances. First, we reject the category whose class probability p_c is lower than 0.1. And we employ the prototype updating strategy with $n = 2$, $k = 0.1 * T$ and $\lambda = 0.1$. Then we localize action instances of category c on the final CAS, *i.e.*, φ_{tc} by a predefined threshold. Finally, we employ non-maximum suppression among two-stream proposals to remove highly overlapped proposals. More details are provided in the supplementary material.

Comparison with The State-of-the-art

We compare our method with state-of-the-art weakly supervised methods and several fully supervised ones, the results are shown in Tab.1, Tab.2. On THUMOS14, our method outperforms previous weakly supervised methods by a large margin at all IoU thresholds and improves the average mAP by 6.0%. Besides, our method even surpasses some fully supervised methods at IoU 0.1 and 0.2. The gain is mainly from high IoUs, indicating our method can produce more complete instances. On ActivityNet1.2, our method also outperforms existing weakly supervised methods by 0.9%. These experiments verify the effectiveness of our method.

Ablation Studies

To evaluate the effectiveness of each component, we conduct following ablation studies on the THUMOS14 dataset, the results are shown in Tab.3. and Tab.4. More experimental results like sensitivities of hyperparameters and the power of learned features are reported in the supplementary material.

Prototype embedding module. Three experiments are performed on the way of generating prototypes. In the first

Table 1: Detection performance comparisons over the THUMOS14 dataset. The column AVG indicates the average mAP at IoU thresholds 0.1:0.1:0.5. UNT and I3D represent UntrimmedNet features and I3D features respectively.

Supervision	Method	AP @ IoU							
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	AVG
Full	S-CNN (Shou, Wang, and Chang 2016)	47.7	43.5	36.4	28.7	19.0	-	5.3	35.0
Full	R-C3D (Xu, Das, and Saenko 2017)	54.5	51.5	44.8	35.6	28.9	-	-	43.1
Full	SSN (Zhao et al. 2017)	**60.3**	56.2	50.6	40.8	29.1	-	-	47.4
Full	TAL-Net (Chao et al. 2018)	59.8	**57.1**	**53.2**	**48.5**	**42.8**	**33.8**	**20.8**	**52.3**
Weak	Hide-and-Seek (Singh and Lee 2017)	36.4	27.8	19.5	12.7	6.8	-	-	20.6
Weak	UntrimmedNet (Wang et al. 2017a)	44.4	37.7	28.2	21.1	13.7	-	-	29.0
Weak	Step-by-Step Erasion (Zhong et al. 2018)	45.8	39.0	31.1	22.5	15.9	-	-	30.9
Weak	STPN (UNT) (Nguyen et al. 2018)	45.3	38.8	31.1	23.5	16.2	9.8	5.1	31.0
Weak	W-TALC (UNT) (Paul, Roy, and Roy-Chowdhury 2018)	49.0	42.8	32.0	26.0	18.8	-	6.2	33.7
Weak	AutoLoc (UNT) (Shou et al. 2018)	-	-	35.8	29.0	**21.2**	13.4	5.8	-
Weak	CMCS (UNT) (Liu, Jiang, and Wang 2019)	53.5	46.8	37.5	29.1	19.9	12.3	6.0	37.4
Weak	Ours (UNT)	**54.2**	**47.1**	**37.8**	**29.4**	**21.2**	**13.9**	**6.8**	**37.9**
Weak	STPN (I3D) (Nguyen et al. 2018)	52.0	44.7	35.5	25.8	16.9	9.9	4.3	35.0
Weak	W-TALC (I3D) (Paul, Roy, and Roy-Chowdhury 2018)	55.2	49.6	40.1	31.1	22.8	-	7.6	39.8
Weak	CMCS (I3D) (Liu, Jiang, and Wang 2019)	57.4	50.8	41.2	32.1	23.1	15.0	7.0	40.9
Weak	MAAN (I3D) (Yuan et al. 2019)	59.8	50.8	41.1	30.6	20.3	12.0	6.9	40.5
Weak	Ours (I3D)	**62.3**	**57.0**	**48.2**	**37.2**	**27.9**	**16.7**	**8.1**	**46.5**

Table 2: Results on ActivityNet1.2 validation set. The AVG indicates the average mAP at IoU thresholds 0.5:0.05:0.95.

Method	AP @ IoU			
	0.5	0.75	0.95	AVG
Step-by-Step Erasion	27.3	14.7	2.9	15.6
AutoLoc (U)	27.3	15.1	3.3	16.0
CMCS (U)	33.9	19.9	5.1	20.5
Ours (U)	**37.0**	**21.1**	**5.2**	**22.0**
W-TALC (I)	37.0	-	-	18.0
CMCS (I)	36.8	22.0	**5.6**	22.4
Ours (I)	**37.6**	**23.9**	5.4	**23.3**

Table 3: Ablation study on prototype embedding module.

Methods	Random Initialization	FC	GCN
AVG (0.1:0.5)	44.6	44.7	**46.5**

Table 4: Ablation study on THUMOS14. The column AVG indicates the average mAP at IoU thresholds 0.1:0.1:0.5. Up means the prototype updating strategy.

Method				AP @ IoU			
$\mathcal{L}_{cls}$	$\mathcal{L}_{clu}$	$\widehat{S_{tj}}$	Up	0.1	0.3	0.5	AVG
√				55.8	40.8	22.4	39.9
√	√			60.6	46.9	26.3	45.2
√		√		16.1	4.4	1.3	6.7
√	√	√		61.3	47.2	27.0	45.7
√	√	√	√	**62.3**	**48.2**	**27.9**	**46.5**

experiment, we random initialize the prototypes, and in the second experiment, we feed the label features to a fully connected network. Results are shown in Tab.3. We can see that the prototype embedding module with co-occurrence GCN exceeds the random initialization and fully connected network evidently, indicating the importance of capturing relations among actions. To attain an intuitive understanding about this module, we demonstrate the class-specific gains resulting from building relations of actions on THUMOS14 dataset. Performance differences (top-10) between our full model and the one with random initialization are shown in Fig.3(a). We can see that most of related actions, e.g., *Cliff-Diving* and *Diving*, have gains when adopting the prototype embedding module, demonstrating the effectiveness of our method in capturing relations of actions. Moreover, we also show the distances between prototypes in the supplemental material for obtaining further insights into this module.

Clustering loss. As shown in Tab.4, the clustering loss can benefit our framework a lot. By cooperating with this loss, our method achieves 5.3% gain on average mAP. This reveals the effectiveness of clustering loss in separating actions from backgrounds as well as learning intra-compact representations. Besides, we find that the performance drops significantly (from 39.9% to 6.7% on average mAP) when using similarity weighting module without this loss. It indicates that intra-compact representation is a premise that makes the weights for similarity valid. To attain further insights into this loss, we visualize the learned features, the visualization is shown in the supplemental material.

Similarity weighting module. Several comparative experiments are conducted to show the effectiveness of this module. As shown in Tab.4, without this module, the average mAP drops by 1.3% on THUMOS14. Moreover, the prototype updating strategy can further improve the performance by 0.8% compared with only using the learned prototypes. To attain further insight of the prototype updating strategy, we visualize the embedding features, prototypes

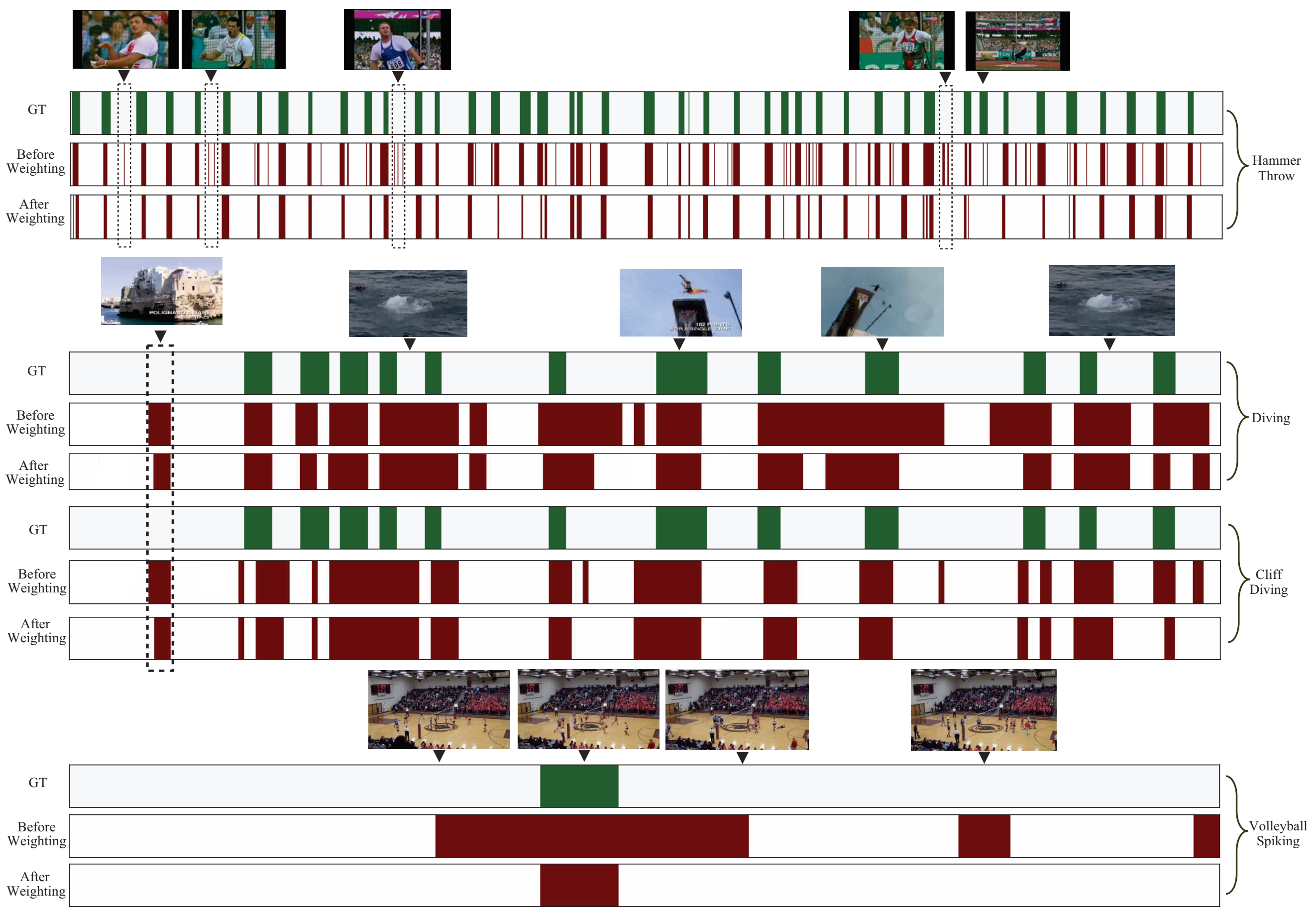

Figure 4: Three prediction examples. **Top:** *Hammer Throw*. Some hard backgrounds like celebration are removed after adopting similarity weighting module. **Middle:** *Diving and Cliff Diving*. Our method can apply to videos occurring multi-actions. There are some false positives like splash after diving. **Bottom:** *Volleyball Spiking*. Our method can deal with sparse action occurrence. False positives which are highly similar with the annotations are filtered out after adopting similarity weighting module.

and updated prototypes by the t-SNE (Maaten and Hinton 2008) in Fig.3(b). We can see that the updated prototypes are closer to the true cluster centers of occurred actions in the video, and so making the weights for similarity more meaningful.

Qualitative Results

We visualize some examples of localized actions in Fig.4. Examples are from THUMOS14 testing set using I3D features. In the first example of *Hammer Throw*, our method nearly pinpoints all the annotated actions. Moreover, the detected instances are more accurate and less redundant after adopting similarity weighing module. In the second example of *Diving and Cliff Diving*, which is a video occurring multiple actions, our method also achieves a promising result. Although there are some false positives, most of them are common patterns of diving, such as the splash. The third example is a sparse action, namely *Volleyball Spiking*. This video has no shot change and all frames are highly similar. In spite of this, our detection result is almost the same as the ground-truth, demonstrating the robustness of our method.

Conclusion

In this paper, we have proposed a relation prototypical network for weakly supervised temporal action localization. We first identified two challenges posed by the weak supervision, namely action-background separation and action relation construction. To tackle the first challenge, a clustering loss is adopted to separate actions from backgrounds and learn intra-compact representations of actions. Meanwhile, a similarity weighting module is introduced for further filtering out backgrounds. To deal with the second challenge, a co-occurrence GCN-based prototype embedding module is proposed to generate relational prototypes. We have evaluated our proposed approach on two benchmark datasets and achieved state-of-the-art performance.

Acknowledgments

This work is jointly supported by National Key Research and Development Program of China (2016YFB1001000), National Natural Science Foundation of China (61525306, 61633021, 61721004, 61420106015, 61806194, U1803261), Capital Science and Technology Leading Talent Training Project (Z181100006318030), HW2019SOW01, and CASAIR. This work is also supported by grants from NVIDIA and the NVIDIA DGX-1 AI Supercomputer.

References

Caba Heilbron, F.; Escorcia, V.; Ghanem, B.; and Carlos Niebles, J. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 961–970. IEEE.

Carreira, J., and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 6299–6308. IEEE.

Chao, Y.-W.; Vijayanarasimhan, S.; Seybold, B.; Ross, D. A.; Deng, J.; and Sukthankar, R. 2018. Rethinking the faster r-cnn architecture for temporal action localization. In *CVPR*, 1130–1139. IEEE.

Chen, Z.-M.; Wei, X.-S.; Wang, P.; and Guo, Y. 2019. Multi-label image recognition with graph convolutional networks. In *CVPR*, 5177–5186. IEEE.

Idrees, H.; Zamir, A. R.; Jiang, Y.-G.; Gorban, A.; Laptev, I.; Sukthankar, R.; and Shah, M. 2017. The thumos challenge on action recognition for videos in the wild. *Computer Vision and Image Understanding* 155:1–23.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lea, C.; Flynn, M. D.; Vidal, R.; Reiter, A.; and Hager, G. D. 2017. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 156–165. IEEE.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*.

Lin, T.; Zhao, X.; Su, H.; Wang, C.; and Yang, M. 2018. Bsn: Boundary sensitive network for temporal action proposal generation. In *ECCV*, 3–19. Springer.

Liu, D.; Jiang, T.; and Wang, Y. 2019. Completeness modeling and context separation for weakly supervised temporal action localization. In *CVPR*, 1298–1307. IEEE.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Nguyen, P.; Liu, T.; Prasad, G.; and Han, B. 2018. Weakly supervised action localization by sparse temporal pooling network. In *CVPR*, 6752–6761. IEEE.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Paul, S.; Roy, S.; and Roy-Chowdhury, A. K. 2018. W-talc: Weakly-supervised temporal activity localization and classification. In *ECCV*, 588–607. Springer.

Shou, Z.; Chan, J.; Zareian, A.; Miyazawa, K.; and Chang, S.-F. 2017. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *CVPR*, 5734–5743. IEEE.

Shou, Z.; Gao, H.; Zhang, L.; Miyazawa, K.; and Chang, S.-F. 2018. Autoloc: Weakly-supervised temporal action localization in untrimmed videos. In *ECCV*, 154–171. Springer.

Shou, Z.; Wang, D.; and Chang, S.-F. 2016. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, 1049–1058. IEEE.

Simonyan, K., and Zisserman, A. 2014. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 568–576.

Singh, K. K., and Lee, Y. J. 2017. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *ICCV*, 3544–3553. IEEE.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *NIPS*, 4077–4087.

Sultani, W.; Chen, C.; and Shah, M. 2018. Real-world anomaly detection in surveillance videos. In *CVPR*, 6479–6488. IEEE.

Sun, C.; Shetty, S.; Sukthankar, R.; and Nevatia, R. 2015. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *ACM MM*, 371–380. ACM.

Wang, L.; Xiong, Y.; Lin, D.; and Van Gool, L. 2017a. Untrimmednets for weakly supervised action recognition and detection. In *CVPR*, 4325–4334. IEEE.

Wang, Z.; Chen, T.; Li, G.; Xu, R.; and Lin, L. 2017b. Multi-label image recognition by recurrently discovering attentional regions. In *ICCV*, 464–472. IEEE.

Xu, H.; Das, A.; and Saenko, K. 2017. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 5783–5792. IEEE.

Xue, X.; Zhang, W.; Zhang, J.; Wu, B.; Fan, J.; and Lu, Y. 2011. Correlative multi-label multi-instance image annotation. In *ICCV*, 651–658. IEEE.

Yang, J.; Lu, J.; Lee, S.; Batra, D.; and Parikh, D. 2018. Graph r-cnn for scene graph generation. In *ECCV*, 670–685. Springer.

Yuan, J.; Ni, B.; Yang, X.; and Kassim, A. A. 2016. Temporal action localization with pyramid of score distribution features. In *CVPR*, 3093–3102. IEEE.

Yuan, Z.; Stroud, J. C.; Lu, T.; and Deng, J. 2017. Temporal action localization by structured maximal sums. In *CVPR*, 3684–3692. IEEE.

Yuan, Y.; Lyu, Y.; Shen, X.; Tsang, I. W.; and Yeung, D.-Y. 2019. Marginalized average attentional network for weakly-supervised learning. *arXiv preprint arXiv:1905.08586*.

Zhao, Y.; Xiong, Y.; Wang, L.; Wu, Z.; Tang, X.; and Lin, D. 2017. Temporal action detection with structured segment networks. In *ICCV*, 2914–2923. IEEE.

Zhong, J.-X.; Li, N.; Kong, W.; Zhang, T.; Li, T. H.; and Li, G. 2018. Step-by-step erasion, one-by-one collection: A weakly supervised temporal action detector. In *ACM MM*, 35–44. ACM.