Graph Convolutional Networks for Temporal Action Localization

Runhao Zeng1* Wenbing Huang2,5* Mingkui Tan1,4† Yu Rong2
Peilin Zhao2 Junzhou Huang2 Chuang Gan3
1School of Software Engineering, South China University of Technology, China
2Tencent AI Lab 3MIT-IBM Watson AI Lab 4Peng Cheng Laboratory, Shenzhen
5Department of Computer Science and Technology, Tsinghua University, State Key Lab. of Intelligent Technology and Systems, Tsinghua National Lab. for Information Science and Technology (TNList)
{runhaozeng.cs, ganchuang1990}@gmail.com, hwenbing@126.com, mingkuitan@scut.edu.cn

Abstract

Most state-of-the-art action localization systems process each action proposal individually, without explicitly exploiting their relations during learning. However, the relations between proposals actually play an important role in action localization, since a meaningful action always consists of multiple proposals in a video. In this paper, we propose to exploit the proposal-proposal relations using Graph Convolutional Networks (GCNs). First, we construct an action proposal graph, where each proposal is represented as a node and their relations between two proposals as an edge. Here, we use two types of relations, one for capturing the context information for each proposal and the other one for characterizing the correlations between distinct actions. Then we apply the GCNs over the graph to model the relations among different proposals and learn powerful representations for the action classification and localization. Experimental results show that our approach significantly outperforms the state-of-the-art on THUMOS14 (49.1% versus 42.8%). Moreover, augmentation experiments on ActivityNet also verify the efficacy of modeling action proposal relationships.

1. Introduction

Understanding human actions in videos has been becoming a prominent research topic in computer vision, owing to its various applications in security surveillance, human behavior analysis and many other areas [10, 35, 38, 12, 13, 14, 15, 16, 42]. Despite the fruitful progress in this vein, there are still some challenging tasks demanding further exploitation — temporal action localization is such an example. To deal with real videos that are untrimmed and usually contain the background of irrelevant activities, 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. Consider a sport video as illustrated in Figure 1, the detector should find out the frames where the action event is happening and identify the category of the event.

Temporal Action localization has attracted increasing attention in the last several years [6, 18, 26, 33, 34]. Inspired by the success of object detection, most current action detection methods resort to the two-stage pipeline: they first generate a set of 1D temporal proposals and then perform classification and temporal boundary regression on each proposal individually. However, processing each proposal separately in the prediction stage will inevitably neglect the semantic relations between proposals.

We contend that exploiting the proposal-proposal relations in the video domain provides more cues to facilitate the recognition of each proposal instance. To illustrate this,
we revisit the example in Figure 1, where we have generated four proposals. On the one hand, the proposals \( p_1, p_2 \) and \( p_3 \) overlapping with each other describe different parts of the same action instance (i.e., the start period, main body and end period). Conventional methods perform prediction on \( p_1 \) by using its feature alone, which we think is insufficient to deliver complete knowledge for the detection. If we additionally take the features of \( p_2 \) and \( p_3 \) into account, we will obtain more contextual information around \( p_1 \), which is advantageous especially for the temporal boundary regression of \( p_1 \). On the other hand, \( p_4 \) describes the background (i.e., the sport field), and its content is also helpful in identifying the action label of \( p_1 \), since what is happening on the sport field is likely to be sport action (e.g., “discus throwing”) but not the one happens elsewhere (e.g., “kissing”). In other words, the classification of \( p_1 \) can be partly guided by the content of \( p_4 \) even they are temporally disjointed.

To model the proposal-proposal interactions, one may employ the self-attention mechanism [39] — as what has been conducted previously in language translation [39] and object detection [22] — to capture the pair-wise similarity between proposals. A self-attention module can affect an individual proposal by aggregating information from all other proposals with the automatically learned aggregation weights. However, this method is computationally expensive as querying all proposal pairs has a quadratic complexity of the proposal number (note that each video could contain more than thousands of proposals). On the contrary, Graph Convolutional Networks (GCNs), which generalize convolutions from grid-like data (e.g., images) to non-grid structures (e.g., social networks), have received increasing interests in the machine learning domain [25, 47]. GCNs can affect each node by aggregating information from the adjacent nodes, and thus are very suitable for leveraging the relations between proposals. More importantly, unlike the self-attention strategy, applying GCNs enables us to aggregate information from only the local neighbourhoods for each proposal, and thus can help decrease the computational complexity remarkably.

In this paper, we regard the proposals as nodes of a specific graph and take advantage of GCNs for modeling the proposal relations. Motivated by the discussions above, we construct the graph by investigating two kinds of edges between proposals, including the contextual edges to incorporate the contextual information for each proposal instance (e.g., detecting \( p_1 \) by accessing \( p_2 \) and \( p_3 \) in Figure 1) and the surrounding edges to query knowledge from nearby but distinct proposals (e.g., querying \( p_4 \) for \( p_1 \) in Figure 1).

We then perform graph convolutions on the constructed graph. Although the information is aggregated from local neighbors in each layer, message passing between distant nodes is still possible if the depth of GCNs increases. Besides, we conduct two different GCNs to perform classification and regression separately, which is demonstrated to be effective by our experiments. Moreover, to avoid the overwhelming computation cost, we further devise a sampling strategy to train the GCNs efficiently while still preserving desired detection performance. We evaluate our proposed method on two popular benchmarks for temporal action detection, i.e., THUMOS14 [24] and ActivityNet1.3 [4].

To sum up, our contributions are as follow:

- To the best of our knowledge, we are the first to exploit the proposal-proposal relations for temporal action localization in videos.
- To model the interactions between proposals, we construct a graph of proposals by establishing the edges based on our valuable observations and then apply GCNs to do message aggregation among proposals.
- We have verified the effectiveness of our proposed method on two benchmarks. On THUMOS14 especially, our method obtains the mAP of 49.1% when \( tIoU = 0.5 \), which significantly outperforms the state-of-the-art, i.e., 42.8% by [6]. Augmentation experiments on ActivityNet also verify the efficacy of modeling action proposal relationships.

2. Related work

Temporal action localization. Recently, great progress has been achieved in deep learning [5, 9, 19, 53], which facilitates the development of temporal action localization. Approaches on this task can be grouped into three categories: (1) methods performing frame or segment-level classification where the smoothing and merging steps are required to obtain the temporal boundaries [33, 28, 51]; (2) approaches employing a two-stage framework involving proposal generation, classification and boundary refinement [34, 46, 52]; (3) methods developing end-to-end architectures integrating the proposal generation and classification [48, 1, 26].

Our work is built upon the second category where the action proposals are first generated and then used to perform classification and boundary regression. Following this paradigm, Shou et al. [34] propose to generate proposals from sliding windows and classify them. Xu et al. [46] exploit the 3D ConvNet and propose a framework inspired by Faster R-CNN [30]. The above methods neglect the context information of proposals, and hence some attempts have been developed to incorporate the context to enhance the proposal feature [8, 17, 18, 52, 6]. They show encouraging improvements by extracting features on the extended receptive field (i.e., boundary) of the proposal. Despite their success, they all process each proposal individually. In contrast, our method has considered the proposal-proposal interactions and leveraged the relations between proposals.
Graph Convolutional Networks. Kipf et al. [25] propose the Graph Convolutional Networks (GCNs) to define convolutions on the non-grid structures [37]. Thanks to its effectiveness, GCNs have been successfully applied to several research areas in computer vision, such as skeleton-based action recognition [47], person re-identification [32], and video classification [45]. For real-world applications, the graph can be large and directly using GCNs is inefficient. Therefore, several attempts are posed for efficient training by virtue of the sampling strategy, such as the node-wise method SAGE [20], layer-wise model FastGCN [7] and its layer-dependent variant AS-GCN [23]. In this paper, considering the flexibility and implementability, we adopt SAGE method as the sampling strategy in our framework.

3. Our Approach

3.1. Notation and Preliminaries

We denote an untrimmed video as \( V = \{I_t \in \mathbb{R}^{H \times W \times 3}\}_{t=1}^T \), where \( I_t \) denotes the frame at the time slot \( t \) with height \( H \) and width \( W \). Within each video \( V \), let \( P = \{p_i \mid p_i = (x_i, (t_{i,s}, t_{i,e}))\}_{i=1}^N \) be the action proposals of interest, with \( t_{i,s} \) and \( t_{i,e} \) being the start and end time of a proposal, respectively. In addition, given proposal \( p_i \), let \( x_i \in \mathbb{R}^d \) be the feature vector extracted by certain feature extractor (e.g., the I3D network [5]) from frames between \( t_{i,s} \) and \( t_{i,e} \).

Let \( \mathcal{G}(V, E) \) be a graph of \( N \) nodes with nodes \( v_i \in V \) and edge \( e_{ij} = (v_i, v_j) \in E \). Furthermore, let \( A \in \mathbb{R}^{N \times N} \) be the adjacency matrix associated with \( \mathcal{G} \). In this paper, we seek to exploit graphs \( \mathcal{G}(P, E) \) on action proposals in \( P \) to better model the proposal-proposal interactions in videos. Here, each action proposal is treated as a node and the edges in \( E \) are used to represent the relations between proposals.

3.2. General Scheme of Our Approach

In this paper, we use a proposal graph \( \mathcal{G}(P, E) \) to present the relations between proposals and then apply GCN on the graph to exploit the relations and learn powerful representations for proposals. The intuition behind applying GCN is that when performing graph convolution, each node aggregates information from its neighborhoods. In this way, the feature of each proposal is enhanced by other proposals, which helps boost the detection performance eventually.

Without loss of generality, we assume the action proposals have been obtained beforehand by some methods (e.g., the TAG method in [52]). In this paper, given an input video \( V \), we seek to predict the action category \( y_i \) and temporal position \( (t_{i,s}, t_{i,e}) \) for each proposal \( p_i \) by exploiting proposal relations. Formally, we compute

\[
\{\{\hat{y}_i, (\hat{t}_{i,s}, \hat{t}_{i,e})\}\}_{i=1}^N = F(\text{GCN}(\{x_i\}_{i=1}^N, \mathcal{G}(P, E))),
\]

where \( F \) denotes any mapping functions to be learned. To exploit GCN for action localization, our paradigm takes both the proposal graph and the proposal features as input and perform graph convolution on the graph to leverage proposal relations. The enhanced proposal features (i.e., the outputs of GCN) are then used to jointly predict the category label and temporal bounding box. The schematic of our approach is shown in Figure 2. For simplicity, we denote our model as P-GCN henceforth.

In the following sections, we aim to answer two questions: (1) how to construct a graph to represent the relations between proposals; (2) how to use GCN to learn representations of proposals based on the graph and facilitate the action localization.
3.3. Proposal Graph Construction

For the graph $\mathcal{G}(\mathcal{P}, \mathcal{E})$ of each video, the nodes are instantiated as the action proposals, while the edges $\mathcal{E}$ between proposals are demanded to be characterized specifically to better model the proposal relations.

One way to construct edges is linking all proposals with each other, which yet will bring in overwhelming computations for going through all proposal pairs. It also incurs redundant or noisy information for action localization, as some unrelated proposals should not be connected. In this paper, we devise a smarter approach by exploiting the temporal relevance/distance between proposals instead. Specifically, we introduce two types of edges, the contextual edges and surrounding edges, respectively.

**Contextual Edges.** We establish an edge between proposal $p_i$ and $p_j$ if $r(p_i, p_j) > \theta_{ctx}$, where $\theta_{ctx}$ is a certain threshold. Here, $r(p_i, p_j)$ represents the relevance between proposals and is defined by the IoU metric, i.e.,

$$r(p_i, p_j) = tIoU(p_i, p_j) = \frac{I(p_i, p_j)}{U(p_i, p_j)},$$

where $I(p_i, p_j)$ and $U(p_i, p_j)$ compute the temporal intersection and union of the two proposals, respectively. If we focus on the proposal $p_i$, establishing the edges by computing $r(p_i, p_j) > \theta_{ctx}$ will select its neighbourhoods as those have high overlaps with it. Obviously, the non-overlapping portions of the highly-overlapping neighbourhoods are able to provide rich contextual information for $p_i$. As already demonstrated in [8, 6], exploring the contextual information is of great help in refining the detection boundary and increasing the detection accuracy eventually. Here, by our contextual edges, all overlapping proposals automatically share the contextual information with each other, and these information are further processed by the graph convolution.

**Surrounding Edges.** The contextual edges connect the overlapping proposals that usually correspond to the same action instance. Actually, distinct but nearby actions (including the background items) could also be correlated, and the message passing among them will facilitate the detection of each other. For example in Figure 1, the background proposal $p_i$ will provide a guidance on identifying the action class of proposal $p_j$ (e.g., more likely to be sport action). To handle such kind of correlations, we first utilize $r(p_i, p_j) = 0$ to query the distinct proposals, and then compute the following distance

$$d(p_i, p_j) = \frac{|c_i - c_j|}{U(p_i, p_j)},$$

where $U(p_i, p_j)$ represents the center coordinate of $p_i$ (or $p_j$). As a complement of the contextual edges, the surrounding edges enable the message to pass across distinct action instances and thereby provides more temporal cues for the detection.

3.4. Graph Convolution for Action Localization

Given the constructed graph, we apply the GCN to do action localization. We build $K$-layer graph convolutions in our implementation. Specifically for the $k$-th layer ($1 \leq k \leq K$), the graph convolution is implemented by

$$X^{(k)} = AX^{(k-1)}W^{(k)},$$

where $A$ is the adjacency matrix; $W^{(k)} \in \mathbb{R}^{d_k \times d_k}$ is the parameter matrix to be learned. $X^{(k)} \in \mathbb{R}^{N \times d_k}$ are the hidden features for all proposals at layer $k$; $X^{(0)} \in \mathbb{R}^{N \times d}$ are the input features.

We apply an activation function (i.e., ReLU) after each convolution layer before the features are forwarded to the next layer. In addition, our experiments find it more effective by further concatenating the hidden features with the input features in the last layer, namely,

$$X^{(K)} = X^{(K)} \| X^{(0)},$$

where $\|$ denotes the concatenation operation.

Joining the previous work [52], we find that it is beneficial to predict the action label and temporal boundary separately by virtue of two GCNs—one conducted on the original proposal features $x_i$ and the other one on the extended proposal features $x'_i$. The first GCN is formulated as

$$\{\hat{y}_i\}_{i=1}^N = \text{softmax}(FC_1(GCN_1(\{x_i\}_{i=1}^N, \mathcal{G}(\mathcal{P}, \mathcal{E})))),$$

where we apply a Fully-Connected (FC) layer with softmax operation on top of $GCN_1$ to predict the action label $\hat{y}_i$. The second GCN can be formulated as

$$\{\{\hat{t}_{i,s}, \hat{t}_{i,e}\}\}_{i=1}^N = FC_2(GCN_2(\{x'_i\}_{i=1}^N, \mathcal{G}(\mathcal{P}, \mathcal{E}))),$$

$$\{\hat{c}_i\}_{i=1}^N = FC_3(GCN_2(\{x'_i\}_{i=1}^N, \mathcal{G}(\mathcal{P}, \mathcal{E}))),$$

where the graph structure $\mathcal{G}(\mathcal{P}, \mathcal{E})$ is the same as that in Eq. (6) but the input proposal feature is different. The extended feature $x'_i$ is attained by first extending the temporal boundary of $p_i$ with $\frac{1}{2}$ of its length on both the left and right sides and then extracting the feature within the extended boundary. Here, we adopt two FC layers on top of $GCN_2$, one for predicting the boundary $(\hat{t}_{i,s}, \hat{t}_{i,e})$ and the other one for predicting the completeness label $\hat{c}_i$, which indicates whether the proposal is complete or not. It has been demonstrated by [52] that, incomplete proposals that have low IoU with the ground-truths could have high classification score, and thus it will make mistakes when using the classification score alone to rank the proposal for the mAP test; further applying the completeness score enables us to avoid this issue.
Adjacency Matrix. In Eq. (4), we need to compute the adjacency matrix $A$. Here, we design the adjacency matrix by assigning specific weights to edges. For example, we can apply the cosine similarity to estimate the weights of edge $e_{ij}$ by

$$
A_{ij} = \frac{x^T_i x_j}{\|x_i\|_2 \cdot \|x_j\|_2}.
$$

In the above computation, we compute $A_{ij}$ relying on the feature vector $x_i$. We can also map the feature vectors into an embedding space using a learnable linear mapping function as in [44] before the cosine computation. We leave the discussion in our experiments.

3.5. Efficient Training by Sampling

Typical proposal generation methods usually produce thousands of proposals for each video. Applying the aforementioned graph convolution (Eq. (4)) on all proposals demands hefty computation and memory footprints. To accelerate the training of GCNs, several approaches [7, 23, 20] have been proposed based on neighbourhood sampling. Here, we adopt the SAGE method [20] in our method for its flexibility.

The SAGE method uniformly samples the fixed-size neighbourhoods of each node layer-by-layer in a top-down passway. In other words, the nodes of the $(k-1)$-th layer are formulated as the sampled neighbourhoods of the nodes in the $k$-th layer. After all nodes of all layers are sampled, SAGE performs the information aggregation in a bottom-up fashion. Here we specify the aggregation function to be a sampling form of Eq. (4), namely,

$$
x_i^{(k)} = \left( \frac{1}{N_s} \sum_{j=1}^{N_s} A_{ij} x_j^{(k-1)} + x_i^{(k-1)} \right) W^{(k)},
$$

where node $j$ is sampled from the neighbourhoods of node $i$, i.e., $j \in \mathcal{N}(i)$; $N_s$ is the sampling size and is much smaller than the total number $N$. The summation in Eq. (10) is further normalized by $N_s$, which empirically makes the training more stable. Besides, we also enforce the self addition of its feature for node $i$ in Eq. (10). We do not perform any sampling when testing. For better readability, Algorithm 1 depicts the algorithmic Flow of our method.

4. Experiments

4.1. Datasets

THUMOS14 [24] is a standard benchmark for action localization. Its training set known as the UCF-101 dataset consists of 13320 videos. The validation, testing and background set contain 1010, 1574 and 2500 untrimmed videos, respectively. Performing action localization on this dataset is challenging since each video has more than 15 action instances and its 71% frames are occupied by background items. Following the common setting in [24], we apply 200 videos in the validation set for training and conduct evaluation on the 213 annotated videos from the testing set.

ActivityNet [4] is another popular benchmark for action localization on untrimmed videos. We evaluate our method on ActivityNet v1.3, which contains around 10K training videos and 5K validation videos corresponded to 200 different activities. Each video has an average of 1.65 action instances. Following the standard practice, we train our method on the training videos and test it on the validation videos. In our experiments, we contrast our method with the state-of-the-art methods on both THUMOS14 and ActivityNet v1.3, and perform ablation studies on THUMOS14.

4.2. Implementation details

Evaluation Metrics. We use mean Average Precision (mAP) as the evaluation metric. A proposal is considered to be correct if its temporal IoU with the ground-truth instance is larger than a certain threshold and the predicted category is the same as this ground-truth instance. On THUMOS14, the tIoU thresholds are chosen from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$; on ActivityNet v1.3, the IoU thresholds are from $\{0.5, 0.75, 0.95\}$, and we also report the average mAP of the IoU thresholds between 0.5 and 0.95 with the step of 0.05.

Features and Proposals. Our model is implemented under the two-stream strategy [35]: RGB frames and optical flow.

Algorithm 1 The training process of P-GCN model.

Input: Proposal set $P = \{p_i | p_i = (x_i, (t_{i,s}, t_{i,e}))\}_{i=1}^{N}$; original proposal features $\{x^{(0)}_i\}_{i=1}^{N}$; extended proposal features $\{x^{(t)}_i\}_{i=1}^{N}$; graph depth $K$; sampling size $N_s$

Parameter: Weight matrices $W^{(k)}$, $\forall k \in \{1, \ldots, K\}$

1: instantiate the nodes by the proposals $p_i$, $\forall p_i \in P$
2: establish edges between nodes
3: obtain a proposal graph $G(P, E)$
4: calculate adjacent matrix using Eq. (9)
5: while not converges do
6: for $k = 1 \ldots K$ do
7: for $p \in P$ do
8: sample $N_s$ neighborhoods of $p$
9: aggregate information using Eq. (10)
10: end for
11: end for
12: predict action categories $\{\hat{y}_i\}_{i=1}^{N}$ using Eq. (6)
13: perform boundary regression using Eq. (7)
14: predict completeness $\{\hat{c}_i\}_{i=1}^{N}$ using Eq. (8)
15: end while

Output: Trained P-GCN model
fields. We first uniformly divide each input video into 64-frame segments. We then use a two-stream Inflated 3D ConvNet (13D) model pre-trained on Kinetics [5] to extract the segment features. In detail, the 13D model takes as input the RGB/optical-flow segment and outputs a 1024-dimensional feature vector for each segment. Upon the 13D features, we further apply max pooling across segments to obtain one 1024-dimensional feature vector for each proposal that is obtained by the BSN method [27]. Note that we do not fine-tune the parameters of the 13D model in our training phase. Besides the 13D features and BSN proposals, our ablation studies in § 5 also explore other types of features (e.g. 2-D features [27]) and proposals (e.g. TAG proposals [52]).

**Proposal Graph Construction.** We construct the proposal graph by fixing the values of \( \theta_{ctx} \) as 0.7 and \( \theta_{sur} \) as 1 for both streams. More discussions on choosing the values of \( \theta_{ctx} \) and \( \theta_{sur} \) could be found in the supplementary material. We adopt 2-layer GCN since we observed no clear improvement with more than 2 layers but the model complexity is increased. For more efficiency, we choose \( N_s = 4 \) in Eq. (10) for neighbourhood sampling unless otherwise specified.

**Training.** The initial learning rate is 0.001 for the RGB stream and 0.01 for the Flow stream. During training, the learning rates will be divided by 10 every 15 epochs. The dropout ratio is 0.8. The classification \( \hat{y}_i \) and completeness \( \hat{c}_i \) are trained with the cross-entropy loss and the hinge loss, respectively. The regression term \( (\hat{t}_{i,s}, \hat{t}_{i,c}) \) is trained with the smooth \( L_1 \) loss. More training details can be found in the supplementary material.

**Testing.** We do not perform neighbourhood sampling (i.e. Eq. (10)) for testing. The predictions of the RGB and Flow streams are fused using a ratio of 2:3. We multiply the classification score with the completeness score as the final score for calculating mAP. We then use Non-Maximum Suppression (NMS) to obtain the final predicted temporal proposals for each action class separately. We use 600 and 100 proposals per video for computing mAPs on THUMOS14 and ActivityNet v1.3, respectively.

### 4.3. Comparison with state-of-the-art results

**THUMOS14.** Our P-GCN model is compared with the state-of-the-art methods in Table 1. The P-GCN model reaches the highest mAP over all thresholds, implying that our method can recognize and localize actions much more accurately than any other method. Particularly, our P-GCN model outperforms the previously best method (i.e. TAL-Net [6]) by 6.3% absolute improvement and the second-best result [27] by more than 12.2%, when \( tIoU = 0.5 \).

**ActivityNet v1.3.** Table 2 reports the action localization results of various methods. Regarding the average mAP, P-GCN outperforms SSN [52], CDC [33], and TAL-Net [6] by 3.01%, 3.19%, and 6.77%, respectively. We observe that the method by Lin et al. [27] (called LIN below) performs promisingly on this dataset. Note that LIN is originally designed for generating class-agnostic proposals, and thus relies on external video-level action labels (from UntrimmedNet [41]) for action localization. In contrast, our method is self-contained and is able to perform action localization without any external label. Actually, P-GCN can still be modified to take external labels into account. To achieve this, we assign the top-2 video-level classes predicted by UntrimmedNet to all the proposals in that video. We provide more details about how to involve external labels in P-GCN in the supplementary material. As summarized in

<table>
<thead>
<tr>
<th>tIoU</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oneata et al. [29]</td>
<td>36.6</td>
<td>33.6</td>
<td>27.0</td>
<td>20.8</td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td>Wang et al. [40]</td>
<td>18.2</td>
<td>17.0</td>
<td>14.0</td>
<td>11.7</td>
<td>8.3</td>
<td></td>
</tr>
<tr>
<td>Caba et al. [3]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>13.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Richard et al. [31]</td>
<td>39.7</td>
<td>35.7</td>
<td>30.0</td>
<td>23.2</td>
<td>15.2</td>
<td></td>
</tr>
<tr>
<td>Shou et al. [34]</td>
<td>47.7</td>
<td>43.5</td>
<td>36.3</td>
<td>28.7</td>
<td>19.0</td>
<td></td>
</tr>
<tr>
<td>Yeung et al. [38]</td>
<td>48.9</td>
<td>44.0</td>
<td>36.0</td>
<td>26.4</td>
<td>17.1</td>
<td></td>
</tr>
<tr>
<td>Yuan et al. [49]</td>
<td>51.4</td>
<td>42.6</td>
<td>33.6</td>
<td>26.1</td>
<td>18.8</td>
<td></td>
</tr>
<tr>
<td>Escorcia et al. [11]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>13.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Buch et al. [2]</td>
<td>-</td>
<td>-</td>
<td>37.8</td>
<td>23.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shou et al. [33]</td>
<td>-</td>
<td>-</td>
<td>40.1</td>
<td>29.4</td>
<td>23.3</td>
<td></td>
</tr>
<tr>
<td>Yuan et al. [50]</td>
<td>51.0</td>
<td>45.2</td>
<td>36.5</td>
<td>27.8</td>
<td>17.8</td>
<td></td>
</tr>
<tr>
<td>Buch et al. [1]</td>
<td>-</td>
<td>45.7</td>
<td>29.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gao et al. [18]</td>
<td>60.1</td>
<td>56.7</td>
<td>50.1</td>
<td>41.3</td>
<td>31.0</td>
<td></td>
</tr>
<tr>
<td>Hou et al. [21]</td>
<td>51.3</td>
<td>-</td>
<td>43.7</td>
<td>22.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dai et al. [8]</td>
<td>-</td>
<td>-</td>
<td>33.3</td>
<td>25.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gao et al. [17]</td>
<td>54.0</td>
<td>50.9</td>
<td>44.1</td>
<td>34.9</td>
<td>25.6</td>
<td></td>
</tr>
<tr>
<td>Xu et al. [46]</td>
<td>54.5</td>
<td>51.5</td>
<td>44.8</td>
<td>35.6</td>
<td>28.9</td>
<td></td>
</tr>
<tr>
<td>Zhao et al. [52]</td>
<td>66.0</td>
<td>59.4</td>
<td>51.9</td>
<td>41.0</td>
<td>29.8</td>
<td></td>
</tr>
<tr>
<td>Lin et al. [27]</td>
<td>-</td>
<td>-</td>
<td>53.5</td>
<td>45.0</td>
<td>36.9</td>
<td></td>
</tr>
<tr>
<td>Chao et al. [6]</td>
<td>59.8</td>
<td>57.1</td>
<td>53.2</td>
<td>48.5</td>
<td>42.8</td>
<td></td>
</tr>
<tr>
<td><strong>P-GCN</strong></td>
<td><strong>69.5</strong></td>
<td><strong>67.8</strong></td>
<td><strong>63.6</strong></td>
<td><strong>57.8</strong></td>
<td><strong>49.1</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tIoU</th>
<th>0.5</th>
<th>0.75</th>
<th>0.95</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Singh et al. [36]</td>
<td>34.47</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Wang et al. [43]</td>
<td>43.65</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Shou et al. [33]</td>
<td>45.30</td>
<td>26.00</td>
<td>0.20</td>
<td>23.80</td>
</tr>
<tr>
<td>Dai et al. [8]</td>
<td>36.44</td>
<td>21.15</td>
<td>3.90</td>
<td>-</td>
</tr>
<tr>
<td>Xu et al. [46]</td>
<td>26.80</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Zhao et al. [52]</td>
<td>39.12</td>
<td>23.48</td>
<td>5.49</td>
<td>23.98</td>
</tr>
<tr>
<td>Chao et al. [6]</td>
<td>38.23</td>
<td>18.30</td>
<td>1.30</td>
<td>20.22</td>
</tr>
<tr>
<td><strong>P-GCN</strong></td>
<td><strong>42.90</strong></td>
<td><strong>28.14</strong></td>
<td><strong>2.47</strong></td>
<td><strong>26.99</strong></td>
</tr>
<tr>
<td>Lin et al. [27]</td>
<td>46.45</td>
<td>29.96</td>
<td>8.02</td>
<td>30.03</td>
</tr>
<tr>
<td><strong>P-GCN</strong></td>
<td><strong>48.26</strong></td>
<td><strong>33.16</strong></td>
<td><strong>3.27</strong></td>
<td><strong>31.11</strong></td>
</tr>
</tbody>
</table>

Table 1. Action localization results on THUMOS14, measured by mAP (%) at different tIoU thresholds \( \omega \).

Table 2. Action localization results on ActivityNet v1.3 (val), measured by mAP (%) at different tIoU thresholds and the average mAP of tIoU thresholds from 0.5 to 0.95. (*) indicates the method that uses the external video labels from UntrimmedNet [41].
5. Ablation Studies

In this section, we will perform complete and in-depth ablation studies to evaluate the impact of each component of our model. More details about the structures of baseline methods (such as MLP and MP) can be found in the supplementary material.

5.1. How do the proposal-proposal relations help?

As illustrated in § 3.4, we apply two GCNs for action classification and boundary regression separately. Here, we implement the baseline with a 2-layer MultiLayer-Perceptron (MLP). The MLP baseline shares the same structure as GCN except that we remove the adjacent matrix $A$ in Eq. (4). To be specific, for the $k$-th layer, the propagation in Eq. (4) becomes $X^k = X^{k-1}W^k$, where $W^k$ are the trainable parameters. Without using $A$, MLP processes each proposal feature independently. By comparing the performance of MLP with GCN, we can justify the importance of message passing among proposals. To do so, we replace each GCN with an MLP and have the following variants of our model including: (1) MLP$_1$ + MLP$_2$; (2) MLP$_1$ + GCN$_2$; and (3) GCN$_1$ + MLP$_2$. We report the results in Table 4. Our P-GCN outperforms all MP variants, demonstrating the superiority of graph convolution over mean pooling on capturing between-proposal connections. The protocol MP$_1$ + MP$_2$ in Table 4 performs better than MLP$_1$ + MLP$_2$ in Table 3, which again reveals the benefit of modeling the proposal-proposal relations, even we pursue it using the naïve mean pooling.

5.2. How does the graph convolution help?

Besides graph convolutions, performing mean pooling among proposal features is another way to enable information dissemination between proposals. We thus conduct another baseline by first adopting MLP on the proposal features and then conducting mean pooling on the output of MLP over adjacent proposals. The adjacent connections are formulated by using the same graph as GCN. We term this baseline as MP below. Similar to the setting in § 5.1, we have three variants of our model including: (1) MP$_1$ + MP$_2$; (2) MP$_1$ + GCN$_2$; and (3) GCN$_1$ + MP$_2$. We report the results in Table 4. Our P-GCN outperforms all MP variants, demonstrating the superiority of graph convolution over mean pooling on capturing between-proposal connections.

5.3. Influences of different backbones

Our framework is general and compatible with different backbones (i.e., proposals and features). Beside the backbones applied above, we further perform experiments on TAG proposals [52] and 2D features [27]. We try different combinations: (1) BSN+I3D; (2) BSN+2D; (3) TAG+I3D; (4) TAG+2D, and report the results of MLP and P-GCN in Figure 3. In comparison with MLP, our P-GCN leads to significant and consistent improvements in all types of features.
Table 6. Comparison of two types of edge on THUMOS14, measured by mAP (%).

<table>
<thead>
<tr>
<th></th>
<th>mAP@tIoU=0.5</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RGB Gain</td>
<td>Flow Gain</td>
<td></td>
</tr>
<tr>
<td>w/ both edges (P-GCN)</td>
<td>37.27</td>
<td>-</td>
<td>46.53</td>
</tr>
<tr>
<td>w/o surrounding edges</td>
<td>35.84</td>
<td>-1.43</td>
<td>45.89</td>
</tr>
<tr>
<td>w/o contextual edges</td>
<td>36.81</td>
<td>-0.46</td>
<td>45.57</td>
</tr>
<tr>
<td>w/o both edges (MLP)</td>
<td>34.75</td>
<td>-2.52</td>
<td>43.68</td>
</tr>
</tbody>
</table>

Table 7. Comparison of different sampling size and training time for each iteration on THUMOS14, measured by mAP@tIoU=0.5.

<table>
<thead>
<tr>
<th>Ns</th>
<th>RGB</th>
<th>Flow</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>36.0</td>
<td>36.92</td>
<td>35.68</td>
</tr>
<tr>
<td>Flow</td>
<td>46.15</td>
<td>45.06</td>
<td>45.13</td>
</tr>
<tr>
<td>Time(s)</td>
<td>0.10</td>
<td>0.23</td>
<td>0.33</td>
</tr>
</tbody>
</table>

and proposals. These results conclude that, our method is generally effective and is not limited to the specific feature or proposal type.

5.4. The weights of edge and self-addition

We have defined the weights of edges in Eq. (9), where the cosine similarity (cos-sim) is applied. This similarity can be further extended by first embedding the features before the cosine computation. We call the embedded version as embed-cos-sim, and compare it with cos-sim in Table 5. No obvious improvement is attained by replacing cos-sim with embed-cos-sim (the mAP difference between them is less than 0.4%). Eq. (10) has considered the self-addition of the node feature. We also investigate the importance of this term in Table 5. It suggests that the self-addition leads to at least 1.7% absolute improvements on both RGB and Flow streams.

5.5. Is it necessary to consider two types of edges?

To evaluate the necessity of formulating two types of edges, we perform experiments on two variants of our P-GCN, each of which considers only one type of edge in the graph construction stage. As expected, the result in Table 6 drops remarkably when either kind of edge is removed. Another crucial point is that our P-GCN still boosts MLP when only the surrounding edges are remained. The rationale behind this could be that, actions in the same video are correlated and exploiting the surrounding relation will enable more accurate action classification.

5.6. The efficiency of our sampling strategy

We train P-GCN efficiently based on the neighbourhood sampling in Eq. (10). Here, we are interested in how the sampling size $N_s$ affects the final performance. Table 7 reports the testing mAPs corresponded to different $N_s$ varying from 1 to 5 (and also 10). The training time per iteration is also added in Table 7. We observe that when $N_s = 4$, the model achieves higher mAP than the full model (i.e., $N_s = 10$) while reducing 76% of training time for each iteration. This is interesting, as sampling fewer nodes even yields better results. We conjecture that, the neighbourhood sampling could bring in more stochasticity and guide our model to escape from the local minimal during training, thus delivering better results.

5.7. Qualitative Results

Given the significant improvements, we also attempt to find out in what cases our P-GCN model improves over MLP. We visualize the qualitative results on THUMOS14 in Figure 4. In the top example, both MLP and our P-GCN model are able to predict the action category correctly, while P-GCN predicts a more precise temporal boundary. In the bottom example, due to similar action characteristic and context, MLP predicts the action of “Shotput” as “Throw Discus”. Despite such challenge, P-GCN still correctly predicts the action category, demonstrating the effectiveness of our method. More qualitative results could be found in the supplementary material.

6. Conclusions

In this paper, we have exploited the proposal-proposal interaction to tackle the task of temporal action localization. By constructing a graph of proposals and applying GCNs to message passing, our P-GCN model outperforms the state-of-the-art methods by a large margin on two benchmarks, i.e., THUMOS14 and ActivithNet v1.3. It would be interesting to extend our P-GCN for object detection in image and we leave it for our future work.

Acknowledgements. This work was partially supported by National Natural Science Foundation of China (NSFC) 61602185, 61836003 (key project), Program for Guangdong Introducing Innovative and Entrepreneurial Teams 2017ZT07X183, Guangdong Provincial Scientific and Technological Funds under Grants 2018B010107001, and Tencent AI Lab Rhino-Bird Focused Research Program (No. JR201902).
References


[16] Chuang Gan, Ting Yao, Kuiyuan Yang, Yi Yang, and Tao Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. In CVPR, pages 923–932, 2016. 1


