Graph Convolutional Module for Temporal Action Localization in Videos

Runhao Zeng*, Wenbing Huang*, Mingkui Tan†, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan

Abstract—Temporal action localization, which requires a machine to recognize the location as well as the category of action instances in videos, has long been researched in computer vision. The main challenge of temporal action localization lies in that videos are usually long and untrimmed with diverse action contents involved. Existing state-of-the-art action localization methods divide each video into multiple action units (*i.e.*, proposals in two-stage methods and segments in one-stage methods) and then perform action recognition/regression on each of them individually, without explicitly exploiting their relations during learning. In this paper, we claim that the relations between action units play an important role in action localization, and a more powerful action detector should not only capture the local content of each action unit but also allow a wider field of view on the context related to it. To this end, we propose a general graph convolutional module (GCM) that can be easily plugged into existing action localization methods, including two-stage and one-stage paradigms. To be specific, we first construct a graph, where each action unit is represented as a node and their relations between two action units as an edge. Here, we use two types of relations, one for capturing the temporal connections between different action units, and the other one for characterizing their semantic relationship. Particularly for the temporal connections in two-stage methods, we further explore two different kinds of edges, one connecting the overlapping action units and the other one connecting surrounding but disjointed units. Upon the graph we built, we then apply graph convolutional networks (GCNs) to model the relations among different action units, which is able to learn more informative representations to enhance action localization. Experimental results show that our GCM consistently improves the performance of existing action localization methods, including two-stage methods (*e.g.*, CBR [15] and R-C3D [47]) and one-stage methods (*e.g.*, D-SSAD [22]), verifying the generality and effectiveness of our GCM. Moreover, with the aid of GCM, our approach significantly outperforms the state-of-the-art on THUMOS14 (50.9% versus 42.8%). Augmentation experiments on ActivityNet also verify the efficacy of modeling the relationships between action units. The source code and the pre-trained models are available at https://github.com/Alvin-Zeng/GCM.

Index Terms—Temporal Action Localization, Graph Convolutional Networks, Video Analysis.

1 Introduction

Understanding human actions from raw videos is a long-standing research goal of computer vision, owing to its various applications in security surveillance, human behavior analysis and many other areas [12], [36], [38], [42]. Joining the success of deep learning, video-based action classification [6], [38], [42] has exhibited fruitful progress in recent years. Nevertheless, this task assumes a tacit approval of addressing videos that are trimmed and short, which limits its practical potential. Temporal action localization, in contrast, targets on untrimmed and long videos to localize the start and end times of every action instance of interest as well as to predict the corresponding label. Taking the sports video in Figure 1 as an example, the detector should determine where the action event is occurring and identify which class the event belongs to. The lower restriction in video collection and preprocessing makes temporal action localization a more compelling yet challenging task in video analytics.

A variety of studies have been performed on temporal action localization over the last few years [1], [2], [7], [15], [16], [22], [26], [34], [35], [56]. In general, existing methods are categorized into two types: the two-stage paradigm [7], [15], [35], [56] and the one-stage paradigm [2], [22], [26]. For the two-stage methods, they first generate a set of action proposals and then individually perform classification and temporal boundary regression on each proposal. In terms of one-stage methods, they divide each video into segments of equal number and then predict the labels and boundary offsets of the anchors mounted to each segment. Despite their difference in whether they use external proposals or not, these two paradigms share the similar spirit of independently conducting classification/regression on each action unit—a general concept corresponds to the proposal in two-stage methods and the segment in one-stage methods. Processing each action unit separately, however, will inevitably neglect the relations in-between and potentially lose critical cues for action localization. For example, the adjacent action units around the target unit can provide temporal context for localizing its temporal boundary. Meanwhile, two distant action units might also provide indicative hints of action recognition to each other if they are semantically similar.

Based upon the intuition above, this paper investigates the relationships between action units from two perspectives, namely the *temporal relationship* and the *semantic relationship*. To

- *R. Zeng is with the School of Software Engineering, South China University of Technology and also with the Pazhou Laboratory, Guangzhou, China. E-mail: runhaozeng.cs@gmail.com*
- *W. Huang is with the Department of Computer Science and Technology, Tsinghua University, State Key Lab. of Intelligent Technology and Systems, Tsinghua National Lab. for Information Science and Technology (TNList) E-mail: hwenbing@126.com*
- *M. Tan is with the School of Software Engineering, South China University of Technology and also with the Key Laboratory of Big Data and Intelligent Robot, South China University of Technology, Ministry of Education. E-mail: mingkuitan@gmail.com*
- *Y. Rong, P. Zhao and J. Huang are with the Tencent AI Laboratory, Shenzhen, China E-mail: yu.rong@hotmail.com; peilinzhao@hotmail.com; jzhuang@uta.edu*
- *C. Gan is with the MIT-IBM Watson AI Lab, Cambridge, MA 02142 USA E-mail: ganchuang1990@gmail.com*
- **Authors contributed equally. †Corresponding author.*

Manuscript received April 19, 2005; revised August 26, 2015.

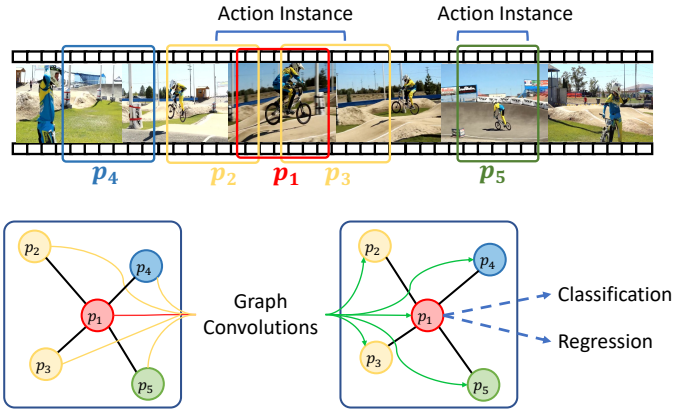

Fig. 1. Schematic depiction of our approach. We apply graph convolutional networks to model the interactions between action units and boost the temporal action localization performance.

illustrate this, we revisit the example in Figure 1, where we have generated five action units. **1)** *Temporal relationship*: the action units $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ overlapping with each other describe different parts of the same action instance (*i.e.*, the start period, main body and end period). Conventional action localization methods perform prediction on $\boldsymbol{p}_1$ by using its feature alone, which we think is insufficient to deliver complete knowledge. If we additionally consider the features of $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$, we will obtain more contextual information around $\boldsymbol{p}_1$, which is advantageous especially for the temporal boundary regression of $\boldsymbol{p}_1$. On the other hand, $\boldsymbol{p}_4$ describes the background (*i.e.*, the sport field), and its content is also helpful in identifying the action label of $\boldsymbol{p}_1$, since what is happening on the sports field is likely to be sports action (*e.g.*, "riding bicycle") but not the action that occurs elsewhere (*e.g.*, "kissing"). In other words, the classification of $\boldsymbol{p}_1$ can be partly guided by the content of $\boldsymbol{p}_4$ since they are temporally related even disjointed. **2)** *Semantic relationship:* $\boldsymbol{p}_5$ is distant from $\boldsymbol{p}_1$, but it describes the same action type as $\boldsymbol{p}_1$ ("riding bicycle") in a different view. We can acquire more complete information for predicting the action category of $\boldsymbol{p}_1$ if we additionally leverage the content of $\boldsymbol{p}_5$.

To model the interactions between action units, one possible way is to employ the self-attention mechanism [39], as what has been conducted previously in language translation [39] and object detection [19], to capture the pair-wise similarity between action units. A self-attention module can affect an individual action unit by aggregating information from all other action units with the automatically learned aggregation weights. However, this method is computationally expensive as querying all action unit pairs has a quadratic complexity of the node number (note that each video can contain more than thousands of action units). In contrast, 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 interest in the machine learning domain [24], [50]. GCNs can affect each node by aggregating information from the adjacent nodes, and thus are very suitable for leveraging the relations between action units. More importantly, unlike the self-attention strategy, applying GCNs enables us to aggregate information from only the local neighborhoods for each action unit, and thus can help remarkably decrease the computational complexity.

In this paper, we propose a general graph convolutional module (GCM) that can be easily plugged into existing action localization methods to exploit the relations between action units. In this module, we first regard the action units as the nodes of a specific graph and represent their relations as edges. To construct the graph, we investigate three kinds of edges between action units, including: **1)** the *contextual edges* to incorporate the contextual information for each proposal instance (*e.g.*, detecting $\boldsymbol{p}_1$ by accessing $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ in Figure 1); **2)** the *surrounding edges* to query knowledge from nearby but distinct action units (*e.g.*, querying $\boldsymbol{p}_4$ for $\boldsymbol{p}_1$); **3)** the *semantic edges* to involve the content of the semantically similar units for enhanced action recognition (*e.g.*, recognizing $\boldsymbol{p}_1$ by considering $\boldsymbol{p}_5$). Then, we 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 the GCNs increases. Moreover, to avoid the overwhelming computational cost, we further devise a sampling strategy to train the GCNs efficiently while still preserving the desired detection performance. We evaluate our proposed method by incorporating GCM with existing action localization methods on two popular benchmarks for temporal action detection, *i.e.*, THUMOS14 [23] and AcitivityNet1.3 [5].

In summary, our contributions are as follows:

- To the best of our knowledge, we are the first to exploit the relationships between action units for temporal action localization in videos.
- To model the interactions between action units, we propose a general graph convolutional module (GCM) to construct a graph of action units by establishing the edges based on our valuable observations and then apply GCNs for message aggregation among action units. Our GCM can be plugged into existing two-stage and one-stage methods.
- Experimental results show that GCM consistently improves the performance of SSN [56], R-C3D [47], CBR [15] and D-SSAD [22] on two benchmarks, demonstrating the generality and effectiveness of our proposed GCM. On THUMOS14 especially, our method obtains a mAP of 50.9% when $tIoU = 0.5$, which significantly outperforms the state-of-the-art, *i.e.*, 42.8% by [7]. Augmentation experiments on ActivityNet also verify the efficacy of modeling action proposal relationships.

This paper extends our preliminary version [55] that was published in ICCV 2019 in the following several aspects. **1)** We integrate graph construction and graph convolution into a general graph convolutional module (GCM) so that the proposed module can be plugged into any of the two-stage temporal action localization methods (*e.g.*, SSN, R-C3D and CBR) and the one-stage methods (*e.g.*, D-SSAD). **2)** In addition to the temporal relationships leveraged in our ICCV paper, we further explore semantic relationships to learn more discriminative representations. Experimental results reveal that the semantic relationships provide more valuable information for action recognition. **3)** We conduct more ablation studies (*e.g.*, analysis of semantic edges, runtime comparison with the baseline methods, and comparisons for one-stage methods) to verify the effectiveness and efficiency of the proposed method. **4)** We achieve clearly better action localization results over our ICCV version on THUMOS14 (50.9% vs. 49.1%) and ActivityNet 1.3 (31.45% vs. 31.11%).

2 Related Work

Temporal action localization. Recently, great progress has been achieved in deep learning [6], [38], [42], 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, which requires a post-processing step (*e.g.*, smoothing and merging) to obtain the temporal boundaries of the action instances [29], [31], [34]; (2) approaches employing a two-stage framework similar to the two-stage object detection methods in images. They often involve proposal generation, proposal classification and boundary refinement [35], [47], [56]; (3) methods that integrate proposal generation and classification (and/or boundary regression) into end-to-end architectures, which are often called one-stage action localization methods [2], [26], [52].

Our work can be used to help both two-stage and one-stage action localization paradigms, where each video is divided into multiple action units and each action unit is processed individually. Following the two-stage paradigm, Shou *et al.* [35] proposed generating a set of proposal candidates from sliding windows and classifying them by using deep neural networks. Xu *et al.* [47] exploited the 3D convolutional networks and proposed a framework inspired by Faster R-CNN [32]. Following the one-stage paradigm, Lin *et al.* [26] divided the video into segments and used convolutional layers to obtain video features, which were further processed by an anchor layer for temporal action localization. Huang *et al.* [22] decoupled the localization and classification in a one-stage scheme. However, the above methods neglect the contextual information of action units. To address this issue, some attempts have been developed to incorporate the context to enhance the proposal feature [7], [10], [14], [15], [56]. 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 action unit individually. In contrast, our method considered the relations between action units.

Graph-based relation modeling. Relation modeling has proven to be very helpful in many computer vision tasks like object detection [19], visual reasoning [9] and image classification [44]. For instance, the performance of object detection can be improved by considering the object relations since objects in an image are often highly correlated [19]. Recently, Kipf *et al.* [24] proposed graph convolutional network (GCN) to define convolutions on non-grid structures. Due to its effectiveness in relation modeling, GCN has been widely applied to several research areas in computer vision, such as skeleton-based action recognition [50], object detection [48] and video classification [45]. Wang *et al.* [45] used a graph to represent the spatiotemporal relations between objects for the action classification task. Xu *et al.* [48] constructed an object graph relying on the spatial configurations between objects for object detection. Our work considers both the temporal and semantic relations between action units for a more challenging temporal action localization task, where both action classification and localization are required. Recently, Xu *et al.* [49] proposed a one-stage action localization method with a graph to exploit the relations between video segments. Our work is able to model the relations between action units (*i.e.*, video segments or proposals) and is more general since it can be easily plugged into existing action localization methods, including two-stage and one-stage paradigms.

Graph sampling strategy. For real-world applications, the graph can be large and directly using GCNs is inefficient. Therefore, several attempts have been made for efficient training by virtue of the sampling strategy, such as the node-wise method SAGE [17], layer-wise model FastGCN [8] and its layer-dependent variant AS-GCN [21]. In this paper, considering the flexibility and implementability, we adopt the 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 $\mathcal{P} = \{\boldsymbol{p}_i \mid \boldsymbol{p}_i = (\mathbf{x}_i, (t_{i,s}, t_{i,e}))\}_{i=1}^{N}$ be the action units of interest, where the action unit can be a proposal in two-stage action localization methods (*e.g.*, SSN [56]) or a video segment in one-stage methods (*e.g.*, SSAD [26]). Let $t_{i,s}$ and $t_{i,e}$ be the start and end times of an action unit, respectively. In addition, given action unit $\boldsymbol{p}_i$, let $\mathbf{x}_i \in \mathcal{R}^d$ be the feature extracted by a certain feature extractor (*e.g.*, the I3D network [6]) from frames between $I_{t_{i,s}}$ and $I_{t_{i,e}}$.

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

3.2 General scheme of our approach

We focus on solving the problem that existing temporal action localization methods neglect the relation between action units, which, however, is able to significantly improve the localization accuracy. Thus, we propose a general graph convolutional module (GCM) that can be inserted into existing action localization methods in a plug-and-play manner. In particular, GCM uses a graph $\mathcal{G}(\mathcal{P}, \mathcal{E})$ to present the relations between action units and then applies GCN on the graph to exploit the relations and learn powerful representations for action units. The intuition is that when performing graph convolution, each node aggregates information from its neighborhoods. In this way, the feature of each action unit is enhanced by other action units, which helps eventually improve the detection performance. The schematic of our approach is shown in Figure 2.

Without loss of generality, we assume the action units have been obtained beforehand by some methods (*e.g.*, the TAG method in [56]). Given the features of the action units $\{\mathbf{x}_i\}_{i=1}^{N}$ and their initial temporal boundaries $\{(t_{i,s}, t_{i,e})\}_{i=1}^{N}$, our GCM constructs a graph $\mathcal{G}$ according to the temporal and semantic relations between action units. Then, we apply a K-layer GCN in the GCM to exploit the relations and obtain the relation-aware features $\mathbf{Y}$ of action units. For the k-th layer ($1 \le k \le K$), the graph convolution is implemented by

$$\mathbf{X}^{(k)} = \mathbf{A}\mathbf{X}^{(k-1)}\mathbf{W}^{(k)}. \tag{1}$$

Here, $\mathbf{A}$ is the adjacency matrix, $\mathbf{W}^{(k)} \in \mathbb{R}^{d_k\times d_k}$ is the parameter matrix to be learned, $\mathbf{X}^{(k)} \in \mathbb{R}^{N\times d_k}$ are the hidden features for all action units at layer k, and $\mathbf{X}^{(0)} \in \mathbb{R}^{N\times d}$ 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,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3090167, IEEE Transactions on Pattern Analysis and Machine Intelligence

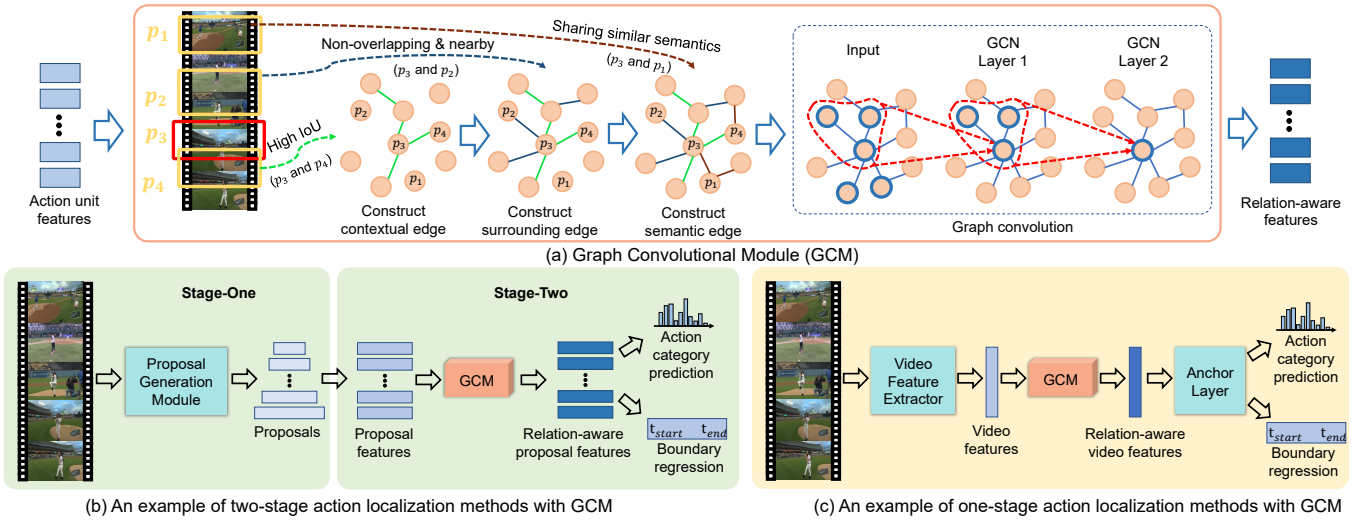

Fig. 2. Schematic of our method. (a) Given a set of action units (*e.g.*, proposals in two-stage methods and segments in one-stage methods), our graph convolutional module (GCM) instantiates the nodes in the graph by each action unit. Then, we establish three kinds of edges among nodes to model the relations between action units and employ GCNs on the constructed graph. Lastly, our GCM module outputs relation-aware features. (b) For two-stage action localization methods, our GCM can be used in the second stage to enhance the proposal features, which are used for action classification and boundary regression. (c) For one-stage action localization methods, our GCM can be exploited to enhance the video features before the anchor layer.

our experiments find it more effective by further combining the hidden features with the input features in the last layer, namely,

$$\mathbf{Y} = \mathbf{X}^{(K)} + \mathbf{X}^{(0)}, \tag{2}$$

where the summation is performed in an element-wise manner. The relation-aware action unit features $\mathbf{Y}$ are then used to jointly predict the action category $\hat{y}_i$ and temporal position $(\hat{t}_{i,s}, \hat{t}_{i,e})$ for each action unit $\boldsymbol{p}_i$ by calculating

$$\{(\hat{y}_i, (\hat{t}_{i,s}, \hat{t}_{i,e}))\}_{i=1}^{N} = F(\mathbf{Y}), \tag{3}$$

where F denotes any action localization methods, such as SSN [56], R-C3D [47], CBR [15] and D-SSAD [22].

In the following sections, we aim to answer two questions: (1) how to construct a graph to represent the relations between action units, and (2) how to insert our GCM into the existing action localization methods, including the two-stage paradigm and one-stage paradigm.

3.3 Action unit graph construction

For the graph $\mathcal{G}(\mathcal{P}, \mathcal{E})$ of each video, the nodes are instantiated as the action units, while the edges $\mathcal{E}$ between action units are demanded to be characterized specifically to better model the relations. One way for constructing edges is linking all action units with each other, which yet leads to overwhelming computations for going through all action unit pairs. It also incurs redundant or noisy information for action localization, as some unrelated action units should not be connected. In this paper, we devise a smarter approach by exploiting the temporal relevance/distance and the semantic relationships between action units instead. Specifically, we introduce three types of edges, the contextual edges, the surrounding edges and the semantic edges, respectively.

3.3.1 *Contextual edges*

We establish an edge between action units $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ if $r(\boldsymbol{p}_i, \boldsymbol{p}_j) > \theta_{ctx}$, where θ_{ctx} is a certain threshold. Here, $r(\boldsymbol{p}_i, \boldsymbol{p}_j)$ represents the relevance between action units and is defined by the tIoU metric, *i.e.*,

$$r(\boldsymbol{p}_i, \boldsymbol{p}_j) = tIoU(\boldsymbol{p}_i, \boldsymbol{p}_j) = \frac{I(\boldsymbol{p}_i, \boldsymbol{p}_j)}{U(\boldsymbol{p}_i, \boldsymbol{p}_j)}, \tag{4}$$

where $I(\boldsymbol{p}_i, \boldsymbol{p}_j)$ and $U(\boldsymbol{p}_i, \boldsymbol{p}_j)$ compute the temporal intersection and union of the two action units, respectively. If we focus on the proposal $\boldsymbol{p}_i$, establishing the edges by computing $r(\boldsymbol{p}_i, \boldsymbol{p}_j) > \theta_{ctx}$ will select its neighborhoods as those that have high overlaps with it. Obviously, the non-overlapping portions of the highly-overlapping neighborhoods can provide rich contextual information for $\boldsymbol{p}_i$. As already demonstrated in [7], [10], exploring the contextual information is of great help in refining the detection boundary and eventually increasing the detection accuracy. Here, by our contextual edges, all overlapping action units automatically share the contextual information with each other, and this information is further processed by the graph convolution.

3.3.2 *Surrounding edges*

The contextual edges connect the overlapping action units that usually correspond to the same action instance. Actually, surrounding but disjointed action units (including the background items) can also be correlated, and the message passing among them will facilitate the detection of each other. For example, in Figure 1, the background $\boldsymbol{p}_4$ provides guidance on identifying the action class of action unit $\boldsymbol{p}_1$ (*e.g.*, more likely to be sports actions). To handle such kind of correlations, we first utilize $r(\boldsymbol{p}_i, \boldsymbol{p}_j) = 0$ to query the disjointed action units, and then compute the following distance

$$d(\boldsymbol{p}_i, \boldsymbol{p}_j) = \frac{|c_i - c_j|}{U(\boldsymbol{p}_i, \boldsymbol{p}_j)}, \tag{5}$$

to add the edges between nearby action units if $d(\boldsymbol{p}_i, \boldsymbol{p}_j) < \theta_{sur}$, where θ_{sur} is a certain threshold. In Eq. (5), c_i (or c_j) represents the center coordinate of $\boldsymbol{p}_i$ (or $\boldsymbol{p}_j$). As a complement to the contextual edges, the surrounding edges enable the message to pass across distinct action instances and thereby provide more temporal cues for the detection.

3.3.3 Semantic edges

The above contextual and surrounding edges aim to exploit the temporal context for each action unit, which, however, still neglects the semantic information between action units. It is worth noting that one untrimmed video often contains multiple action instances (*e.g.*, each video on THUMOS14 dataset [23] contains more than 15 action instances on average), and the instances in one video often belong to the same or semantically similar action category. For example, the actions *CricketBowling* and *CricketShot* often occur in the same video on THUMOS14. Although their categories are different when performing action localization, it is intuitive that the semantics of *CricketBowling* are helpful for recognizing *CricketShot* from other actions (*e.g.*, *CliffDiving*). Therefore, the proposal that locates at a distance from an action but containing similar semantic content might provide indicative hints for detecting the action.

To exploit such semantic information for action localization, we add a semantic edge between the action units that share similar semantics. In particular, we first define an action unit set $\mathcal{S}_i$ for the i-th action unit as

$$\mathcal{S}_i = \{\boldsymbol{p}_j | r(\boldsymbol{p}_i, \boldsymbol{p}_j) = 0, j \in \mathcal{N}_l(i)\}, \tag{6}$$

where $\mathcal{N}_l(i)$ is the index set of the l nearest neighborhoods of proposal $\boldsymbol{p}_i$ and $\mathcal{N}_l(i)$ is constructed in the feature space relying on the cosine similarity between action unit features $\mathbf{x}_i$ and $\mathbf{x}_j$. Then, we establish a semantic edge between $\boldsymbol{p}_i$ and the action units in $\mathcal{S}_i$. Note that the action unit feature $\mathbf{x}_i$ can be the high-level appearance or motion feature containing rich semantic information. In other words, the action units sharing similar appearance (*e.g.*, some similar places) or motions (*e.g.*, the same action performed by different actors) can be used to help the recognition of action units. To summarize, the edge e_{ij} between nodes $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ can be formulated as

$$e_{ij} = \begin{cases} 1, & \text{if} \quad r(\boldsymbol{p}_i, \boldsymbol{p}_j) > \theta_{ctx}; \\ 1, & \text{if} \quad r(\boldsymbol{p}_i, \boldsymbol{p}_j) = 0, d(\boldsymbol{p}_i, \boldsymbol{p}_j) < \theta_{sur}; \\ 1, & \text{if} \quad r(\boldsymbol{p}_i, \boldsymbol{p}_j) = 0, j \in \mathcal{N}_l(i); \\ 0, & else. \end{cases} \tag{7}$$

3.3.4 Adjacency matrix

In Eq. (1), we need to compute the adjacency matrix $\mathbf{A}$. 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

$$\mathbf{A}_{ij} = \begin{cases} \frac{\mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \cdot \|\mathbf{x}_j\|_2}, & e_{ij} = 1; \\ 0, & e_{ij} = 0. \end{cases} \tag{8}$$

In the above computation, we compute $\mathbf{A}_{ij}$ relying on the feature vector $\mathbf{x}_i$. 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.

3.4 GCM for two-stage action localization methods

Due to the residual nature of GCM (see Eq. (2)), the proposed GCM can be easily plugged into existing two-stage action localization methods, which typically involve the following steps: **Step 1:** generates a set of proposal candidates, which may contain action instances; **Step 2:** uses some certain feature extractors, which can be off-the-shelf [15] or trained in an end-to-end manner [47], to obtain the proposal features; **Step 3:** processes the proposal features using an action classifier and a boundary regressor, which are often implemented as fully-connected layers; **Step 4:** performs duplicate removal, which is usually achieved by using non-maximum suppression (NMS).

In this paper, our proposed GCM is used between Step 2 and Step 3. Given a set of proposals, our GCM first constructs a proposal graph according to Equation (7). Then, the relation-aware proposal features are obtained by performing graph convolution on the constructed graph via Equations (1) and (2). Joining the previous work SSN [56], we find that it is beneficial to predict the action label and temporal boundary separately by virtue of two GCMs—one conducted on the original proposal features $\mathbf{x}_i$ and the other one on the extended proposal features $\mathbf{x}'_i$. The first GCM is formulated as

$$\{\hat{y}_i\}_{i=1}^N = \mathrm{softmax}(\mathrm{FC}_1(\mathrm{GCM}_1(\{\mathbf{x}_i\}_{i=1}^N))), \tag{9}$$

where we apply a fully-connected (FC) layer with soft-max operation on top of GCM_1 to predict the action label $\hat{y}_i$. The second GCM can be formulated as

$$\{(\hat{t}_{i,s}, \hat{t}_{i,e})\}_{i=1}^N = \mathrm{FC}_2(\mathrm{GCM}_2(\{\mathbf{x}'_i\}_{i=1}^N)), \tag{10}$$

$$\{\hat{e}_i\}_{i=1}^N = \mathrm{FC}_3(\mathrm{GCM}_2(\{\mathbf{x}'_i\}_{i=1}^N)), \tag{11}$$

where the graph structure $\mathcal{G}(\mathcal{P}, \mathcal{E})$ is the same as that in Eq. (9) but the input proposal feature is different. The extended feature $\mathbf{x}'_i$ is attained by first extending the temporal boundary of $\boldsymbol{p}_i$ 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 GCM_2, one for predicting the boundary $(\hat{t}_{i,s}, \hat{t}_{i,e})$ and the other one for predicting the completeness score $\hat{c}_i$, which indicates whether the proposal is complete or not. It has been demonstrated by [56] that, incomplete action units that have low tIoU with the ground-truths can have high classification scores, 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.

For other two-stage action localization methods (*e.g.*, CBR [15], R-C3D [47]) that do not rely on the two-stream pipeline such as SSN, we only insert one GCM into them. Specifically, GCM takes the original proposal features $\mathbf{x}_i$ as input and outputs the relation-aware features, which are further processed by two individual FC layers for predicting the action classification and boundary regression, respectively. Formally, the action localization process can be formulated as

$$\begin{aligned} \{(\hat{t}_{i,s}, \hat{t}_{i,e})\}_{i=1}^N &= \mathrm{FC}_4(\mathrm{GCM}_3(\{\mathbf{x}_i\}_{i=1}^N)), \\ \{\hat{y}_i\}_{i=1}^N &= \mathrm{softmax}(\mathrm{FC}_5(\mathrm{GCM}_3(\{\mathbf{x}_i\}_{i=1}^N))). \end{aligned} \tag{12}$$

where FC_* denotes the fully-connected (FC) layers, whose inputs are the same relation-aware features produced by GCM.

3.5 GCM for one-stage action localization methods

Our proposed GCM is a general module for exploiting the relationships between action units, which can be the segments in one-stage action localization methods, as discussed in Section 1.

Existing one-stage methods [22], [26] are inspired by the single-shot object detection methods in images [28]. A three-step pipeline is used in these methods, as summarized below. **Step 1:** evenly divides the input video into T segments and extracts a C-dim feature vector for each segment, thus leading to a 1D feature map $\mathbf{F} \in \mathbb{R}^{T \times C}$; **Step 2:** obtain 1D feature maps with multiple temporal scales (*i.e.*, different temporal granularity) relying on $\mathbf{F}$; **Step 3:** predict the action category and boundary offsets of the anchors mounted to each location on the 1D feature maps. For better readability, we call the feature vector at each location as a feature unit.

Our proposed GCM is used between Step 2 and Step 3. Although the boundaries of feature units are non-overlapping, we can incorporate our GCM to exploit the relations between feature units with a minor modification. In particular, we only consider the surrounding and semantic edges to link the feature units and perform graph convolution to aggregate messages. The intuition is that the feature units can be regarded as a special case of proposals. Specifically, each feature unit corresponds to a segment in the videos with a certain duration, and these segments are non-overlapping. By adding the GCM to the 1D feature maps, we are able to exploit the relationship between the feature units in a 1D feature map. It is worth mentioning that our module can be inserted one or multiple times throughout the network to model the feature relationships at different scales.

3.6 Training details

3.6.1 Loss functions

Our proposed method not only predicts the action category and the completeness score (when inserting our GCM into SSN [56]) of each proposal but also refines the temporal boundary of action units by location regression. To train our model, we define the following loss functions:

Classification Loss. We define the training loss function for the action classifier as follows:

$$L_{cls} = \frac{1}{N} \sum_{i=1}^{N} L_1(y_i, \hat{y}_i), \tag{13}$$

where y_i and $\hat{y}_i$ are the ground truth and the prediction of the i-th action unit, respectively. We use the cross-entropy loss as L_1, and N is the number of action units in a mini-batch.

Completeness Loss. Given the predicted completeness score $\hat{e}_i$ and the ground truth e_i of the i-th action unit, we use the following loss function to train the completeness predictor:

$$L_{com} = \frac{1}{N_{com}} \sum_{i=1}^{N} \mathbb{1}^i_{com} L_2(e_i, \hat{e}_i), \tag{14}$$

where we use hinge loss as L_2 and N_{com} is the number of completeness training samples. $\mathbb{1}^i_{com}$ is the indicator function, being 1 if $y_i \geq 1$ (*i.e.*, the action unit is not considered as part of the background) and 0 otherwise.

Regression Loss. We devise a set of location regressors $\{R_m\}_{m=1}^{N_{class}}$, each for an action category. For an action unit, we regress the boundary using the closest ground-truth instance as the target. Our method predicts the offset $\hat{o}_i = (\hat{o}_{i,c}, \hat{o}_{i,l})$ relative to the action unit , where $\hat{o}_{i,c}$ and $\hat{o}_{i,l}$ are the offset of center coordinate and length, respectively. The ground-truth offset is denoted as $o_i = (o_{i,c}, o_{i,l})$ and parameterized by:

$$\begin{aligned} o_{i,c} &= (c_i - c_{gt})/l_i, \\ o_{i,l} &= log(l_i/l_{gt}), \end{aligned} \tag{15}$$

Algorithm 1 Training details of our method.

Input: Action unit set $\mathcal{P} = \{\boldsymbol{p}_i \mid \boldsymbol{p}_i = (\mathbf{x}_i, (t_{i,s}, t_{i,e}))\}_{i=1}^{N}$; original action unit features $\{\mathbf{x}_i^{(0)}\}_{i=1}^{N}$; extended action unit features $\{\mathbf{x}_i'^{(0)}\}_{i=1}^{N}$; graph depth K; sampling size N_s

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

```
instantiate the nodes by the action units p_i, ∀p_i ∈ P
establish edges between nodes using Eq. (7)
obtain an action unit graph G(P, E)
calculate adjacent matrix using Eq. (8)
while not converges do
  for k = 1 ... K do
    for p ∈ P do
      sample N_s neighborhoods of p
      aggregate information using Eq. (18)
    end for
  end for
  predict action categories {ŷ_i}_{i=1}^N using Eq. (9)
  perform boundary regression using Eq. (10)
  predict completeness score {ĉ_i}_{i=1}^N using Eq. (11)
  compute L_total using Eq. (17)
  update parameters via stochastic gradient descent
end while
```

where c_i and l_i denote the original center coordinate and length of the action unit, respectively. c_{gt} and l_{gt} are the center coordinate and length of the closest ground truth, respectively. To train the regressor, we define the following loss function:

$$L_{reg} = \frac{1}{N_{reg}} \sum_{i=1}^{N} \mathbb{1}^i_{reg} L_3(o_i, \hat{o}_i), \tag{16}$$

where N_{reg} is the number of regression training samples. $\mathbb{1}^i_{reg}$ is the indicator function, being 1 if $y_i \geq 1$ and $e_i = 1$ (*i.e.*, the proposal is a foreground sample) and 0 otherwise. We use the smooth-L1 loss as L_3 because it is less sensitive to outliers.

Multi-task Loss. We train the whole model by using the following multi-task loss function:

$$L_{total} = L_{cls} + \lambda_1 L_{com} + \lambda_2 L_{reg}, \tag{17}$$

where λ_1 and λ_2 are hyper-parameters to trade-off these losses. We set $\lambda_1 = \lambda_2 = 0.5$ in all the experiments and find that it works well across all of them. It is worth mentioning that we consider the completeness loss only when we plug our GCM into the SSN method [56].

3.6.2 Efficient training by sampling

Typical action unit generation methods usually produce thousands of action units for each video. Applying the aforementioned graph convolution (Eq. (1)) on all action units demands many computations and large memory footprints. To accelerate the training of GCNs, several approaches [8], [17], [21] have been proposed based on neighborhood sampling. Here, we adopt the SAGE method [17] in our method for its flexibility.

The SAGE method uniformly samples the fixed-size neighborhoods 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 neighborhoods 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 manner. Here, we specify the aggregation function to be a sampling form of Eq. (1), namely,

$$\mathbf{x}_i^{(k)} = \left(\frac{1}{N_s} \sum_{j=1}^{N_s} \mathbf{A}_{ij} \mathbf{x}_j^{(k-1)} + \mathbf{x}_i^{(k-1)} \right) \mathbf{W}^{(k)}, \tag{18}$$

where node j is sampled from the neighborhoods of node i, *i.e.*, $j \in \mathcal{N}(i)$, and N_s is the sampling size and is much less than the total number N. The summation in Eq. (18) is further normalized by N_s, which empirically makes the training more stable. In addition, we also enforce the self-addition of its feature for node i in Eq. (18). 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 [23] 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 sets contain 1010, 1574 and 2500 untrimmed videos, respectively. The temporal action localization task of THUMOS14, which contains videos over 20 hours from 20 sports classes, is very challenging since each video has more than 15 action instances and its 71% frames are occupied by background items. Following the common setting in [23], we apply 200 videos in the validation set for training and conduct evaluation on the 213 annotated videos from the testing set.

ActivityNet [5] is another popular benchmark for action localization on untrimmed videos. We evaluate our method on ActivityNet v1.3, which contains approximately 10K training videos and 5K validation videos corresponding 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 the 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.

Graph construction. We construct the graph by fixing the values of θ_{ctx} as 0.7 and θ_{sur} as 1 for both streams, which are selected by grid search. We adopt a 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. (18) for neighborhood 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 are divided by 10 every 15 epochs. The dropout ratio is 0.8.

Testing. We do not perform neighborhood sampling (*i.e.*, Eq. (18)) 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 action units for each action class separately. We use 800 and 100 action units per video for computing mAPs on THUMOS14 and ActivityNet v1.3, respectively.

Action units and features for two-stage methods. The action units in two-stage methods refer to the action proposals. Our model is implemented under the two-stream strategy [36]: RGB frames and optical-flow fields. **1)** For SSN [56], we first uniformly divide each input video into 64-frame RGB/optical-flow segments and adopt a two-stream I3D model pre-trained on Kinetics [6] to obtain a 1024-dimensional feature vector for each segment. Upon the I3D 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 finetune the parameters of the I3D model in our training phase. In addition to the I3D features and BSN proposals, our ablation studies in Section 5.4 also explore other types of features (*e.g.*, 2D features [27]) and proposals (*e.g.*, TAG action units [56]). **2)** For CBR [15], we use the two-stream model [46] pre-trained on the ActivityNet v1.3 training set as the feature extractor. We use the proposals obtained from the proposal stage in [15] to perform action localization. **3)** For R-C3D [47], we use a 3D ConvNet modified from C3D [38] to extract proposal features. We adopt the proposals generated by the proposal subnet in [47] for a fair comparison.

Action units and features for one-stage methods. The action units in one-stage methods refer to the video segments. We follow [22] to use two-stream networks [36] pre-trained on Kinetics [6] to extract spatial and temporal feature representations for each video clip with length 512. We keep other settings (*e.g.*, the learning rate, anchor settings) the same as those are used in [22] for fair comparisons.

4.3 Comparison with state-of-the-art results

THUMOS14. Our method is compared with the state-of-the-art methods in Table 1. GCM consistently boosts the performance of both two-stage methods (*e.g.*, SSN [56], R-C3D [47], CBR [15]) and one-stage methods (*e.g.*, D-SSAD [22]) on THUMOS14, demonstrating the generality and effectiveness of our proposed GCM. With the aid of GCM, our method (*i.e.*, SSN+GCM) reaches the highest mAP over all thresholds, implying that our method can recognize and localize actions much more accurately than any other method. In particular, our method outperforms the previously best method (*i.e.*, TAL-Net [7]) by 8.1% absolute improvement and the second-best result [16] by more than 13.5%, when $tIoU = 0.5$. When using the proposals of higher quality (*i.e.*, BMN proposals [25]), our method (*i.e.*, SSN+GCM†) lifts the mAP to 51.9% when $tIoU = 0.5$.

ActivityNet v1.3. We report the action localization results of various methods in Table 2. Regarding the average mAP, our method (*i.e.*, SSN+GCM) outperforms SSN [56], and CDC [34] by 2.83% and 3.40%, respectively. We observe that BSN [27] and BMN [25] perform promisingly on this dataset. Note that these two methods were originally designed for generating class-agnostic proposals, and thus rely 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, our method can be modified to take external labels into account. To achieve this, we replace the predicted action classes in Eq. (9) with the external action labels. Specifically, given an input video, we use UntrimmedNet to predict the top-2 video-level classes and assign these classes to all the proposals in this

TABLE 1
Action localization results on THUMOS14, measured by mAP (%) at different tIoU thresholds α. (†) indicates the method that uses BMN proposals [25].

Paradigm	tIoU	0.1	0.2	0.3	0.4	0.5
One-Stage	Yeung *et al.* [52]	-	-	36.0	26.4	17.1
	Lin *et al.* [26]	-	-	43.0	35.0	24.6
	Buch *et al.* [2]	-	-	45.7	-	29.2
	Huang *et al.* [22]	66.4	64.7	59.8	53.4	43.2
	Huang *et al.* + GCM	66.4	65.2	61.4	54.7	44.8
Two-Stage	Wang *et al.* [40]	18.2	17.0	14.0	11.7	8.3
	Caba *et al.* [4]	-	-	-	-	13.5
	Escorcia *et al.* [11]	-	-	-	-	13.9
	Oneata *et al.* [30]	36.6	33.6	27.0	20.8	14.4
	Richard *et al.* [33]	39.7	35.7	30.0	23.2	15.2
	Yeung *et al.* [52]	48.9	44.0	36.0	26.4	17.1
	Yuan *et al.* [54]	51.0	45.2	36.5	27.8	17.8
	Yuan *et al.* [53]	51.4	42.6	33.6	26.1	18.8
	Shou *et al.* [35]	47.7	43.5	36.3	28.7	19.0
	Hou *et al.* [18]	51.3	-	43.7	-	22.0
	Buch *et al.* [3]	-	-	37.8	-	23.0
	Shou *et al.* [34]	-	-	40.1	29.4	23.3
	Dai *et al.* [10]	-	-	-	33.3	25.6
	Gao *et al.* [14]	54.0	50.9	44.1	34.9	25.6
	Huang *et al.* [20]	-	-	-	-	27.7
	Yang *et al.* [51]	-	-	44.1	37.1	28.2
	Zhao *et al.* [56]	66.0	59.4	51.9	41.0	29.8
	Gao *et al.* [13]	-	-	-	-	29.9
	Alwassel *et al.* [1]	-	-	51.8	42.4	30.8
	Lin *et al.* [27]	-	-	53.5	45.0	36.9
	Gleason *et al.* [16]	52.1	51.4	49.7	46.1	37.4
	Chao *et al.* [7]	59.8	57.1	53.2	48.5	42.8
	Xu *et al.* [47]	54.5	51.5	44.8	35.6	28.9
	Xu *et al.* + GCM	56.3	53.5	47.0	37.9	30.9
	Gao *et al.* [15]	60.1	56.7	50.1	41.3	31.0
	Gao *et al.* + GCM	61.2	57.8	50.3	42.4	32.2
	Zhao *et al.* [56] (I3D)	69.7	67.5	64.6	58.3	49.3
	Zhao *et al.* + GCM	70.5	68.6	65.2	59.8	50.9
	Zhao *et al.* + GCM†	**72.5**	**70.9**	**66.5**	**60.8**	**51.9**

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

tIoU	0.5	0.75	0.95	Average
Singh *et al.* [37]	34.47	-	-	-
Wang *et al.* [43]	43.65	-	-	-
Shou *et al.* [34]	45.30	26.00	0.20	23.80
Dai *et al.* [10]	36.44	21.15	3.90	-
Xu *et al.* [47]	26.80	-	-	-
Zhao *et al.* [56]	39.12	23.48	5.49	23.98
Chao *et al.* [7]	38.23	18.30	1.30	20.22
Lin *et al.* [27] (BSN*)	46.45	29.96	8.02	30.03
Xu *et al.* [49] (GTAD*)	50.36	34.60	9.02	34.09
Lin *et al.* [25] (BMN*)	50.07	34.78	8.29	33.85
SSN (BSN prop [27])	38.59	24.53	4.57	24.37
SSN + GCM (BSN prop [27])	42.55	28.27	2.84	27.20
SSN* + GCM (BSN prop [27])	47.92	32.91	4.16	31.45
SSN* + GCM (BMN prop [25])	51.03	35.17	7.44	34.24

TABLE 3
Ablation study of GCM on CBR and R-C3D, measured by mAP (%) when tIoU=0.5 on THUMOS14.

Setting	mAP@IoU=0.5	Gain
CBR [15]	31.00	-
CBR + GCM	**32.24**	1.24
R-C3D [47]	28.90	-
R-C3D + GCM	**30.85**	1.95

video. Thus, each proposal has two predicted action classes. To compute mAP, we follow [27] to obtain the score of each proposal by calculating $s_{prop} = s_{gcm} * s_{bsn/bmn} * s_{unet}$, where s_{gcm} is the proposal score predicted by our model (*i.e.*, SSN+GCM), $s_{bsn/bmn}$ is the confidence score produced by BSN (or BMN) and s_{unet} denotes the action score predicted by UntrimmedNet. As summarized in Table 2, our enhanced version (*i.e.*, SSN*+GCM) consistently outperforms BSN and BMN when using the same proposals. Moreover, SSN*+GCM outperforms GTAD [49] even though GTAD uses additional video classification scores from [41]. These results further demonstrate the effectiveness of our method.

5 Ablation Results on Two-Stage Methods

In this section, we will perform complete and in-depth ablation studies to evaluate the impact of each component of our model.

5.1 Effectiveness and generality of GCM

In this section, we incorporate our GCM into two popular two-stage action localization methods (*i.e.*, CBR [15] and R-C3D [47]) to validate the effectiveness and generality of GCM. In the following, we present the implementation details and the results.

Cascaded Boundary Regression (CBR) [15]. The CBR method adopts a cascaded framework to iteratively regress the boundary of the action units. In the proposal stage, CBR uses a deep model to obtain the initial action units by refining the boundary of the sliding windows. In the detection stage, CBR uses another deep model to learn better representations of the action units. Last, these action units are forwarded to fully-connected layers for action classification and boundary regression. In our experiments, we insert our GCM in the detection stage. The outputs of the GCM are forwarded to the action classifier and regressor. For a fair comparison with CBR, we use two-stream features and unit-level offsets. As shown in Table 3, our GCM helps to lift the action localization results over all IoU thresholds, demonstrating its effectiveness.

Region Convolutional 3D Network (R-C3D) [47]. Inspired by the faster-RCNN [32] approach in object detection, Xu *et al.* proposed an end-to-end R-C3D network for activity detection. The network encodes the frames with fully-convolutional 3D layers and then uses a proposal subnet to generate activity segments (*i.e.*, action units). Last, they use a classification subnet to classify and refine the action units based on the RoI-pooled features. Our GCM takes the pooled features as input and enhances the features by constructing a graph and performing graph convolution. The outputs of the GCM are forwarded to the action classifier and regressor. We follow the same settings in [47] for a fair comparison. From Table 3, the action localization performance of R-C3D is significantly improved with the help of our GCM. More critically, the performance gain on two action localization methods demonstrates the generality of our module.

5.2 How do proposal-proposal relations help?

As illustrated in Section 3.4, we apply two GCMs 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 GCM except that we remove the adjacent matrix $\mathbf{A}$ in Eq. (1). Specifically, for the k-th layer, the propagation in Eq. (1) becomes $\mathbf{X}^k = \mathbf{X}^{k-1}\mathbf{W}^k$, where $\mathbf{W}^k$ are the trainable parameters. Without using $\mathbf{A}$, MLP processes each

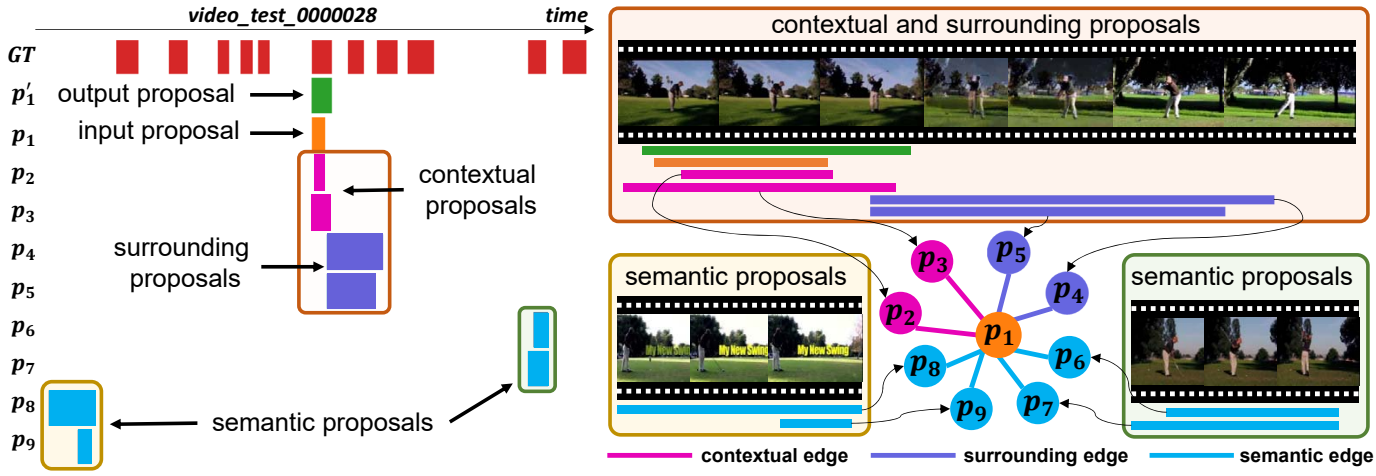

Fig. 3. Visualization results of the graph constructed by our GCM on THUMOS14. The temporal boundary of the input proposal is not precise (*i.e.*, some portions of the corresponding ground truth have not been detected). Our proposed GCM helps to aggregates contextual information from other proposals and lastly predicts the action category correctly and refines the temporal boundary of the input proposal precisely.

proposal feature independently. By comparing the performance of MLP with GCN, we can justify the importance of message passing along action units. To do so, we replace each GCM with an MLP and have the following variants of our model including: (1) $\mathbf{MLP}_1$ + $\mathbf{GCM}_2$ where GCN_1 is replaced; (2) $\mathbf{GCM}_1$ + $\mathbf{MLP}_2$ where GCM_2 is replaced; and (3) $\mathbf{MLP}_1$ + $\mathbf{MLP}_2$ where both GCMs are replaced. Table 4 shows that all these variants decrease the performance of our model, thus verifying the effectiveness of GCNs for both action classification and boundary regression. Overall, our method significantly outperforms the MLP protocol (*i.e.*$\mathbf{MLP}_1$ + $\mathbf{MLP}_2$), validating the importance of considering the relations between action units in temporal action localization. The MLP baseline is indeed a particular implementation of SSN [56]. We compare the runtime between GCM and MLP baseline in Table 5. In detail, we train each model with 200 iterations on a Titan X GPU and report the average processing time per video per iteration (note that proposal generation and feature extraction are excluded for each model). It reads that GCM only incurs a relatively small additional runtime compared with the MLP baseline but is able to improve the performance significantly.

Visualization of the constructed graph. To understand how proposal-proposal relations help improve the action localization performance, we visualize an example of the graph constructed by our proposed method in Figure 3. Specifically, given an input proposal $\boldsymbol{p}_1$, we choose $K = 8$ proposals with the largest weights among all connected proposals. The temporal boundary of the input proposal $\boldsymbol{p}_1$ is not precise (*i.e.*, the ending period of the corresponding ground truth action instance has not been detected in $\boldsymbol{p}_1$). The contextual and surrounding edges connect four proposals ($\boldsymbol{p}_2$, $\boldsymbol{p}_3$, $\boldsymbol{p}_4$ and $\boldsymbol{p}_5$) that can provide a wider receptive field for $\boldsymbol{p}_1$ to detect the ending period of actions. Interestingly, the semantic edges connect not only two proposals ($\boldsymbol{p}_6$ and $\boldsymbol{p}_7$) that provide action information from other action instances in the same video but also two proposals ($\boldsymbol{p}_8$ and $\boldsymbol{p}_9$) with background scenes related to the action instance. Lastly, the temporal boundary of $\boldsymbol{p}_1$ is refined to match the corresponding ground truth and the action category is correctly predicted by our method. Clearly, our proposed GCM is able to exploit contextual information to improve the action localization performance.

TABLE 4
Comparison between our model and the MLP baseline on THUMOS14, measured by mAP (%) when tIoU=0.5..

mAP@tIoU=0.5	RGB	Gain	Flow	Gain
MLP_1 + MLP_2	36.82	-	46.74	-
MLP_1 + GCM_2	38.11	1.29	47.39	0.65
GCM_1 + MLP_2	37.87	1.05	48.14	1.40
GCM_1 + GCM_2	**39.38**	**2.56**	**48.76**	**2.02**

TABLE 5
Comparison with MLP baseline in terms of runtime, computation complexity in FLOPs, and action localization mAP on THUMOS14.

Method	Runtime	FLOPs	mAP@tIoU=0.5	
			RGB	Flow
MLP_1 + MLP_2	0.376s	16.57M	36.82	46.74
GCM_1 + GCM_2	0.404s	17.70M	39.38	48.76

TABLE 6
Comparison between our model and mean-pooling (MP) on THUMOS14, measured by mAP (%) when tIoU=0.5.

mAP@tIoU=0.5	RGB	Gain	Flow	Gain
MP_1 + MP_2	37.12	-	46.96	-
MP_1 + GCM_2	38.32	1.20	47.66	0.80
GCM_1 + MP_2	38.38	1.26	47.93	1.07
GCM_1 + GCM_2	**39.38**	**2.26**	**48.76**	**1.80**

5.3 How does the graph convolution help?

In addition to graph convolutions, performing mean pooling among proposal features is another way to enable information dissemination between action units. We thus conduct another baseline by first adopting MLP on the action unit features and then conducting mean pooling on the output of MLP over adjacent action units. The adjacent connections are formulated by using the same graph as GCN. We term this baseline as MP below. Similar to the setting in Section 5.2, we have three variants of our model including: (1) $\mathbf{MP}_1$ + $\mathbf{MP}_2$; (2) $\mathbf{MP}_1$ + $\mathbf{GCM}_2$; and (3) $\mathbf{GCM}_1$ + $\mathbf{MP}_2$. We report the results in Table 6. The models with two GCMs outperform all MP variants, demonstrating the superiority of graph

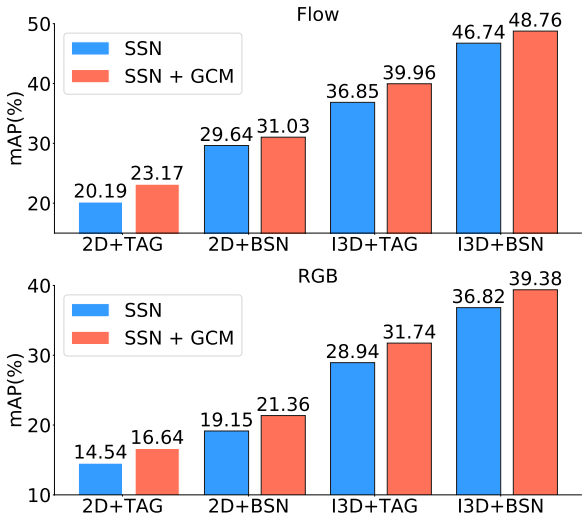

Fig. 4. Action localization results on THUMOS14 with different backbones, measured by mAP@tIoU=0.5.

convolution over mean pooling in capturing between-proposal connections. The protocol $\mathbf{MP}_1$ + $\mathbf{MP}_2$ in Table 6 performs better than $\mathbf{MLP}_1$ + $\mathbf{MLP}_2$ in Table 4, which again reveals the benefit of modeling the relations between action units, even though we pursue it using the naive mean pooling.

5.4 Influences of different backbones

Our framework is general and compatible with different backbones (*i.e.*, action units and features). In addition to the backbones applied above, we further perform experiments on TAG action units [56] and 2D features [27]. We try different combinations: (1) BSN+I3D, (2) BSN+2D, (3) TAG+I3D, and (4) TAG+2D, and report the results of SSN and SSN+GCM in Figure 4. In comparison with MLP, our method leads to significant and consistent improvements in all types of features and action units. These results conclude that our method is generally effective and is not limited to the specific feature or proposal type.

5.5 The weights of edge and self-addition

We have defined the weights of edges in Eq. (8), 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 7. No obvious improvement is attained by replacing cos-sim with embed-cos-sim (the mAP difference between them is less than 0.3%). Eq. (18) has considered the self-addition of the node feature. We also investigate the importance of this term in Table 7. It suggests that the self-addition leads to at least 1.06% absolute improvements on both RGB and flow streams.

Comparisons with learned weights. To further verify the effectiveness of our graph construction strategy, we conduct an experiment by using learned weights of edge. Specifically, we first construct a fully-connected graph and then follow the "scaled dot-product attention" mechanism in [39] to obtain the adjacent matrix by computing $\mathbf{A}_{ij} = \frac{e^{(\mathbf{W}_1\mathbf{x}_i)^T(\mathbf{W}_2\mathbf{x}_j)}}{\sum_{n=1}^{N} e^{(\mathbf{W}_1\mathbf{x}_i)^T(\mathbf{W}_2\mathbf{x}_n)}}$, where $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable parameters and N is the number of proposals

TABLE 7
Comparison of different types of edge functions on THUMOS14, measured by mAP (%) when tIoU=0.5.

mAP@tIoU=0.5	RGB	Flow
cos-sim	38.32	47.62
cos-sim + self-add	39.38	48.76
embed-cos-sim + self-add	39.27	48.92

TABLE 8
Comparisons between our GCM and the baseline using learned weights on THUMOS14.

Method	mAP at different tIoUs				
	0.1	0.2	0.3	0.4	0.5
SSN [56]	69.7	67.5	64.6	58.3	49.3
SSN + Learned weights	70.3	68.3	64.5	58.2	49.7 (↑0.4)
SSN + GCM (ours)	**70.5**	**68.6**	**65.2**	**59.8**	**50.9 (↑1.6)**

TABLE 9
Comparison of three types of edge on THUMOS14, measured by mAP (%) when tIoU=0.5.

mAP@tIoU=0.5	RGB	Gain	Flow	Gain
w/ all edges	39.38	-	48.76	-
w/o surrounding edges	38.80	-0.58	47.69	-0.56
w/o contextual edges	38.28	-1.10	47.57	-0.68
w/o semantic edges	39.02	-0.36	47.38	-0.87
w/o edges (MLP)	36.82	-2.56	46.74	-2.02

TABLE 10
Comparison of different sampling sizes and training time for each iteration on THUMOS14, measured by mAP@tIoU=0.5.

N_s	1	2	3	4	5	10
mAP	48.28	48.47	48.54	**48.76**	48.34	48.30
Time(s)	0.10	0.23	0.33	0.41	0.48	1.72

in one video. Note that one video often contains thousands of proposals, and thus using a fully-connected graph will inevitably incur large computation cost when aggregating information from all other proposals. From Table 8, our GCM outperforms the baseline using learned weights. This is probably because the fully-connected graph may introduce noise from irrelevant proposals, which may even make the training unstable. In contrast, our GCM passes messages only from the temporally adjacent and semantically correlated action units, and thus may eliminate noisy information from irrelevant action units and yield better performance. Moreover, using learned weights is able to lift the action localization performance of the baseline (49.7% vs 49.3%). These results reveal that exploiting action unit relations helps localize actions more precisely, and they also justify our motivation for considering the relations between action units.

5.6 Is it necessary to consider three types of edges?

To evaluate the necessity of formulating three types of edges, we perform experiments on three variants of our method, each of which removes one type of edge in the graph construction stage. From Table 9, the result drops remarkably when any kind of edge is removed. Another crucial point is that our method still improves the MLP baseline when only the surrounding edges remain. The rationale behind this could be that actions in the same video are correlated and exploiting the surrounding relation enables more accurate action classification.

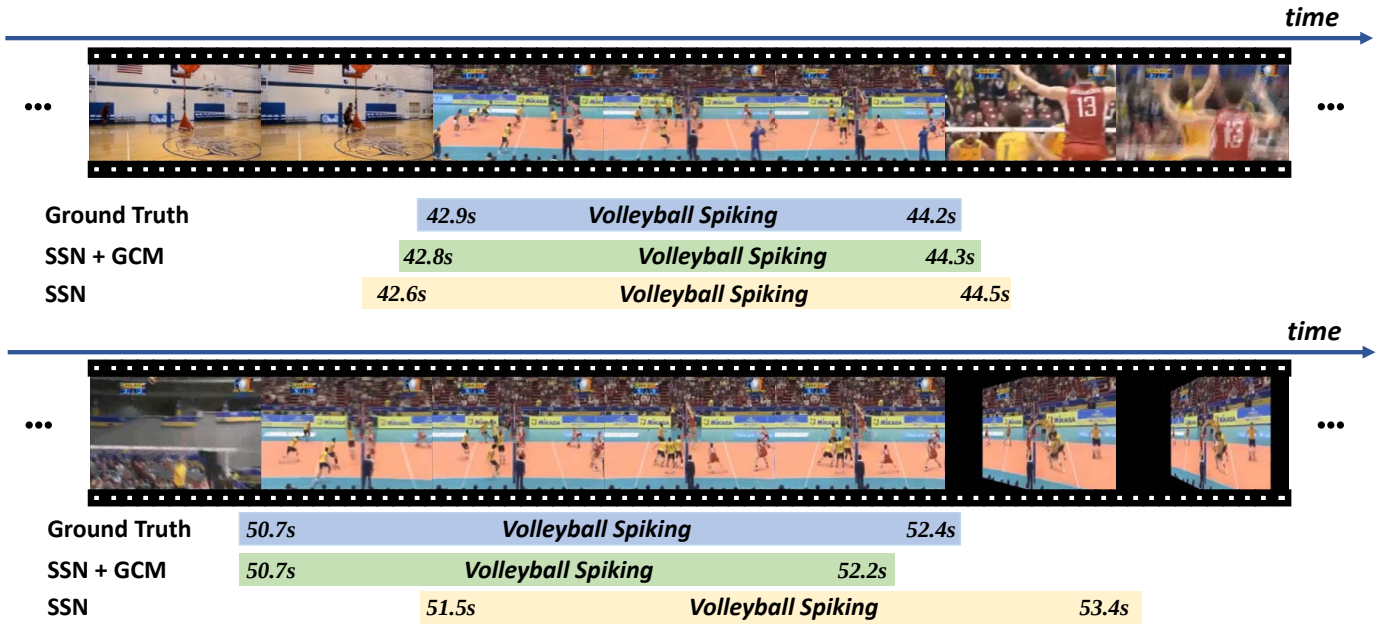

Fig. 5. Qualitative results on THUMOS14 dataset. Our proposed GCM helps SSN to predict a more precise temporal boundary.

TABLE 11
Ablation study of our GCM on D-SSAD, measured by mAP (%) when tIoU=0.5 on THUMOS14.

Setting	mAP@IoU=0.5	Gain
D-SSAD [22] (our impl.)	43.21	-
D-SSAD + GCM × 1	43.47	0.26
D-SSAD + GCM × 2	44.29	1.08
D-SSAD + GCM × 3	**44.77**	1.56

TABLE 12
Comparison of two types of edge on THUMOS14, conducted on D-SSAD [22] with GCM.

Settings		mAP@tIoU=0.5	Gain
surrounding edges	semantic edges		
×	×	43.21	-
✓	×	43.82	0.61
×	✓	44.19	0.98
✓	✓	**44.77**	1.56

5.7 The efficiency of our sampling strategy

We train our model efficiently based on the neighborhood sampling in Eq. (18). Here, we are interested in how the sampling size N_s affects the final performance. Table 10 reports the testing mAPs corresponding to different N_s values varying from 1 to 5 (and also 10). The training time per iteration is also added in Table 10. We observe that when $N_s = 4$, the model achieves higher mAP than the full model (*i.e.*, $N_s = 10$) while reducing the training time by 76% for each iteration. This is interesting, as sampling fewer nodes yields even better results. We conjecture that the neighborhood sampling can bring in more stochasticity and guide our model to escape from the local minimal during training, thus delivering better results.

6 Ablation Results on One-Stage Methods

6.1 Effectiveness of GCM

Decoupled single-shot temporal action detection (D-SSAD) [22]. Huang *et al.* decoupled the localization and classification in a one-stage scheme. In particular, D-SSAD consists of three main components: a base feature network, an anchor network, and a classification/regression module. The base feature network extracts representations of each video segment to form feature maps. Then, a multi-branch anchor network takes the feature maps as input and produces multiple anchors at each location on the feature maps. Last, the anchors are processed by the classification and regression module. In our experiments, we add our GCM to the feature maps before generating anchors. As discussed in Section 3.5, the GCM can be inserted one or multiple times throughout the network to model the feature relationships at different scales. Therefore, we add the GCM to feature maps with multiple scales (from 1 to 3). From Table 11, the performance of D-SSAD is improved by using our GCM to enhance the features. As more GCMs are inserted, the action localization results increase, which demonstrates that our GCM is general and compatible with the one-stage action localization methods.

6.2 How much does each type of edge help?

To evaluate the effectiveness of surrounding and semantic edges, we perform experiments by gradually adding one type of edge to our GCM. From Table 12, adding the surrounding edge and semantic edge to the baseline (*i.e.*, without both types of edges) results in at least 0.61% improvements in terms of action localization mAP. When considering both surrounding and semantic edges simultaneously, the performance is further improved to 44.77%, which strongly supports the necessity of constructing two types of edges in our proposed GCM.

7 Qualitative Results

Given the significant improvements, we also attempt to find out in what cases our method improves over the baseline method. We

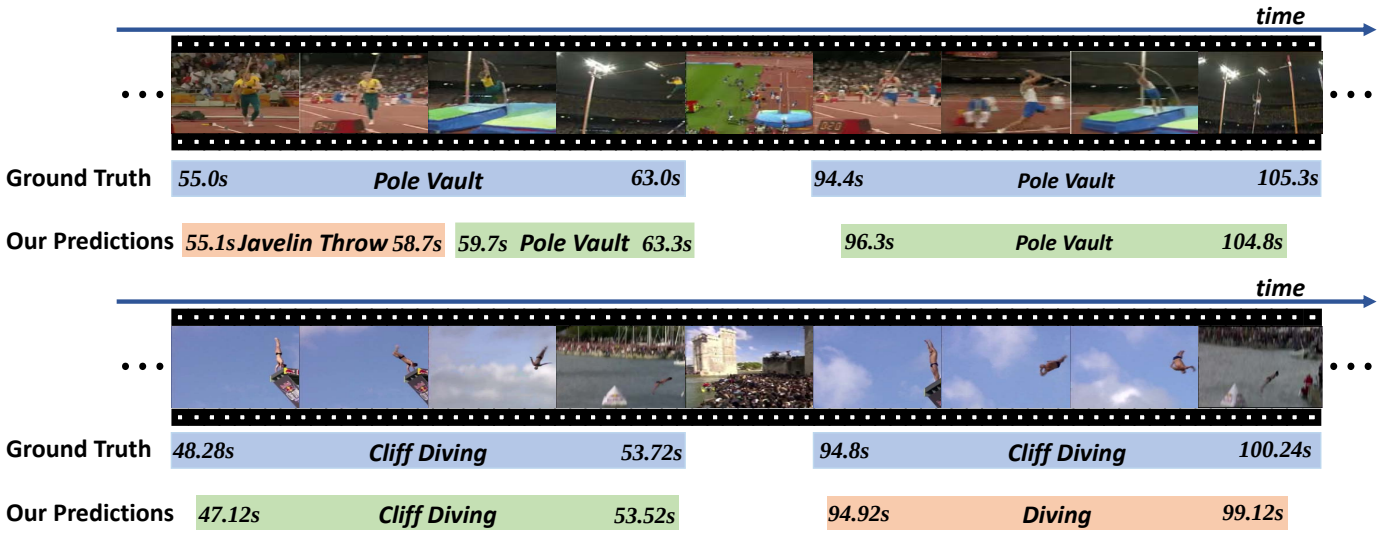

Fig. 6. Examples of failure cases. **Top:** Our method predicts the beginning portion of *Pole Vault* as *Javelin Throw* since these two actions have similar contents (*i.e.*, an athlete running with a pole). **Bottom:** Our method mis-classifies the action *Cliff Diving* into the action *Diving* without recognizing the background *cliff*.

visualize the qualitative results on THUMOS14 in Figure 5. In these examples, the baseline method (*i.e.*, SSN [56]) is able to predict the action category correctly, while failing to precisely predict the location of actions. With the help of our proposed GCM, we predict a more precise temporal boundary, which demonstrates the effectiveness of GCM for temporal action localization.

Failure case analysis. Our method achieves state-of-the-art performance on two benchmark action localization datasets, but like other methods, it is still not sufficiently capable of detecting actions when they share the similar contents. For example, in Figure 6, our method correctly detects the locations of actions but misclassifies the action *Pole Vault* into the action *Javelin Throw* since both these actions share similar contents (*i.e.*an athlete runs when holding a pole). Another failure case is the misclassification between *Cliff Diving* and *Diving*. While this is a common challenge in temporal action localization, exploiting more advanced feature extraction methods may solve it to some extent, which will be left for future exploration.

8 Conclusions

In this paper, we have exploited the relationships between action units to address the task of temporal action localization in videos. Specifically, we have proposed to construct a graph of action units based on the temporal context and semantic information, and apply GCNs to enable message passing among action units. In this way, we enhanced the action unit features and eventually improved the action localization performance. More critically, we have integrated the above graph construction and graph convolution processes into a general graph convolutional module (GCM), which can be easily inserted into existing action localization methods, including the one-stage paradigm and two-stage paradigm. Experimental results show that our GCM is compatible with other action localization methods and helps to consistently improve their action localization accuracy. With the aid of GCM, our method 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 method for object detection in images and we leave it for our future work.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China (NSFC) 62072190, 61836003 (key project), Key-Area Research and Development Program of Guangdong Province 2018B010107001, the Scientific Innovation 2030 Major Project for New Generation of AI under Grant 2020AAA0107300, Ministry of Science and Technology of the People's Republic of China, and National Natural Science Foundation of China (NSFC) 62006137.

References

[1] H. Alwassel, F. Caba Heilbron, and B. Ghanem. Action search: Spotting actions in videos and its application to temporal action localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 251–266, 2018.

[2] S. Buch, V. Escorcia, B. Ghanem, L. Fei-Fei, and J. C. Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. In *Proceedings of the British Machine Vision Conference*, 2017.

[3] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles. Sst: Single-stream temporal action proposals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6373–6382. IEEE, 2017.

[4] F. Caba Heilbron, J. Carlos Niebles, and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1914–1923, 2016.

[5] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[6] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[7] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[8] J. Chen, T. Ma, and C. Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *International Conference on Learning Representations*, 2018.

[9] X. Chen, L.-J. Li, L. Fei-Fei, and A. Gupta. Iterative visual reasoning beyond convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7239–7248, 2018.

[10] X. Dai, B. Singh, G. Zhang, L. S. Davis, and Y. Qiu Chen. Temporal context network for activity localization in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, Oct 2017.

[11] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In *Proceedings of the European Conference on Computer Vision*, pages 768–784, 2016.

[12] L. Fan, W. Huang, C. Gan, S. Ermon, B. Gong, and J. Huang. End-to-end learning of motion representation for video understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[13] J. Gao, K. Chen, and R. Nevatia. Ctap: Complementary temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 68–83, 2018.

[14] J. Gao, Z. Yang, K. Chen, C. Sun, and R. Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3628–3636, 2017.

[15] J. Gao, Z. Yang, and R. Nevatia. Cascaded boundary regression for temporal action detection. In *BMVC*, 2017.

[16] J. Gleason, R. Ranjan, S. Schwarcz, C. Castillo, J.-C. Chen, and R. Chellappa. A proposal-based solution to spatio-temporal action detection in untrimmed videos. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 141–150. IEEE, 2019.

[17] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[18] R. Hou, R. Sukthankar, and M. Shah. Real-time temporal action localization in untrimmed videos by sub-action discovery. In *BMVC*, 2017.

[19] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[20] J. Huang, N. Li, T. Zhang, G. Li, T. Huang, and W. Gao. Sap: Self-adaptive proposal model for temporal action detection based on reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] W. Huang, T. Zhang, Y. Rong, and J. Huang. Adaptive sampling towards fast graph representation learning. In *Advances in Neural Information Processing Systems*, pages 4558–4567, 2018.

[22] Y. Huang, Q. Dai, and Y. Lu. Decoupling localization and classification in single shot temporal action detection. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1288–1293. IEEE, 2019.

[23] Y. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2014.

[24] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[25] T. Lin, X. Liu, X. Li, E. Ding, and S. Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3889–3898, 2019.

[26] T. Lin, X. Zhao, and Z. Shou. Single shot temporal action detection. In *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017*, pages 988–996, 2017.

[27] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *The European Conference on Computer Vision*, September 2018.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[29] A. Montes, A. Salvador, and X. Giro-i Nieto. Temporal activity detection in untrimmed videos with recurrent neural networks. *1st NIPS Workshop on Large Scale Computer Vision Systems*, 2016.

[30] D. Oneata, J. Verbeek, and C. Schmid. The lear submission at thumos 2014. 2014.

[31] A. Piergiovanni and M. Ryoo. Temporal gaussian mixture layer for videos. In *International Conference on Machine learning*, pages 5152–5161. PMLR, 2019.

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[33] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3131–3140, 2016.

[34] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

[35] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1049–1058, 2016.

[36] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[37] G. Singh and F. Cuzzolin. Untrimmed video classification for activity detection: submission to activitynet challenge. *ActivityNet Large Scale Activity Recognition Challenge*, 2016.

[38] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[40] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. *THUMOS14 Action Recognition Challenge*, 1(2):2, 2014.

[41] L. Wang, Y. Xiong, D. Lin, and L. Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

[42] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[43] R. Wang and D. Tao. Uts at activitynet 2016. *ActivityNet Large Scale Activity Recognition Challenge*, 2016.

[44] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[45] X. Wang and A. Gupta. Videos as space-time region graphs. In *The European Conference on Computer Vision*, September 2018.

[46] Y. Xiong, L. Wang, Z. Wang, B. Zhang, H. Song, W. Li, D. Lin, Y. Qiao, L. Van Gool, and X. Tang. Cuhk & ethz & siat submission to activitynet challenge 2016. *arXiv preprint arXiv:1608.00797*, 2016.

[47] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the IEEE International Conference on Computer Vision*, Oct 2017.

[48] H. Xu, C. Jiang, X. Liang, and Z. Li. Spatial-aware graph relation network for large-scale object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[49] M. Xu, C. Zhao, D. S. Rojas, A. Thabet, and B. Ghanem. G-tad: Sub-graph localization for temporal action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10156–10165, 2020.

[50] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[51] K. Yang, P. Qiao, D. Li, S. Lv, and Y. Dou. Exploring temporal preservation networks for precise temporal action localization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[52] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2678–2687, 2016.

[53] J. Yuan, B. Ni, X. Yang, and A. A. Kassim. Temporal action localization with pyramid of score distribution features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3093–3102, 2016.

[54] Z. Yuan, J. C. Stroud, T. Lu, and J. Deng. Temporal action localization by structured maximal sums. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

[55] R. Zeng, W. Huang, M. Tan, Y. Rong, P. Zhao, J. Huang, and C. Gan. Graph convolutional networks for temporal action localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7094–7103, 2019.

[56] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision*, Oct 2017.

Runhao Zeng received the bachelor's degree in Automation Science and Engineering from South China University of Technology, Guangzhou, China, in 2015. He is currently pursuing the Ph.D. degree with the School of Software Engineering at South China University of Technology. His research interests include machine learning, deep learning and their applications in video understanding.

Wenbing Huang is now an assistant researcher at Department of Computer Science and Technology, Tsinghua University. He received his Ph.D. degree of computer science and technology from Tsinghua University in 2017. His current research mainly lies in the areas of machine learning, computer vision, and robotics, with particular focus on learning on irregular structures including graphs and videos. He has published about 30 peer-reviewed top-tier conference and journal papers, including the Proceedings of NeurIPS, ICLR, ICML, CVPR, etc. He served (will serve) as a Senior Program Committee of AAAI 2021, Area Chair of ACMMMM workshop HUMA 2020, and Session Chair of IJCAI 2019.

Mingkui Tan is currently a professor with the School of Software Engineering at South China University of Technology. He received his Bachelor Degree in Environmental Science and Engineering in 2006 and Master degree in Control Science and Engineering in 2009, both from Hunan University in Changsha, China. He received the Ph.D. degree in Computer Science from Nanyang Technological University, Singapore, in 2014. From 2014-2016, he worked as a Senior Research Associate on computer vision in the School of Computer Science, University of Adelaide, Australia. His research interests include machine learning, sparse analysis, deep learning and large-scale optimization.

Yu Rong is a Senior researcher of Machine Learning Center in Tencent AI Lab. He received the Ph.D. degree from The Chinese University of Hong Kong in 2016. He joined Tencent AI Lab in June 2017. His main research interests include social network analysis, graph neural networks, and large-scale graph systems. In Tencent AI Lab, he is working on building the large-scale graph learning framework and applying the deep graph learning model to various applications, such as ADMET prediction and malicious detection. He has published several papers on data mining, machine learning top conferences, including the Proceedings of KDD, WWW, NeurIPS, ICLR, CVPR, ICCV, etc.

Peilin Zhao is currently a Principal Researcher at Tencent AI Lab, China. Previously, he has worked at Rutgers University, Institute for Infocomm Research (I2R), Ant Financial Services Group. His research interests include: Online Learning, Recommendation System, Automatic Machine Learning, Deep Graph Learning, and Reinforcement Learning etc. He has published over 100 papers in top venues, including JMLR, ICML, KDD, etc. He has been invited as a PC member, reviewer or editor for many international conferences and journals, such as ICML, JMLR, etc. He received his bachelor's degree from Zhejiang University, and his Ph.D. degree from Nanyang Technological University.

Junzhou Huang received the BE degree from the Huazhong University of Science and Technology, China, the MS degree from the Chinese Academy of Sciences, China, and the PhD degree from Rutgers University. He is an associate professor with the Computer Science and Engineering Department, University of Texas at Arlington. His major research interests include machine learning, computer vision, and imaging informatics. He was selected as one of the 10 emerging leaders in multimedia and signal processing by the IBM T.J. Watson Research Center in 2010. He received the NSF CAREER Award in 2016.

Chuang Gan is currently a Researcher with the MIT-IBM Watson AI Lab. His research interests mainly include multi-modality learning for video understanding.