# Video Action Segmentation via Contextually Refined Temporal Keypoints

Borui Jiang, Yang Jin, Zhentao Tan, Yadong Mu*
Peking University, Beijing, China
jiangbr@pku.edu.cn, {jiny, tanzhentao}@stu.pku.edu.cn, myd@pku.edu.cn

## Abstract

*Video action segmentation involves categorizing each frame or short snippet of an untrimmed video into predefined action categories. Despite notable advancements in recent years, a considerable number of current approaches still rely on frame-wise segmentation that tends to render fragmentary results. To address it, we present an innovative approach for video action segmentation, centered around contextually refined temporal keypoints. Initially, our method identifies a set of sparse, over-complete temporal keypoints through non-local visual cues, with each keypoint representing a potential action segment candidate. Subsequent enhancements to these initial keypoints are achieved through iterative refining and re-assembling operations. Driven by the notion that optimal temporal keypoints should collectively resemble the true ground-truth structurally, we introduce a module that conducts graph matching between the keypoint-derived graph and the reference graph constructed from accurate annotations. This module effectively learns structural features used to further refine the initial keypoints. Moreover, a set of predefined rules is applied to re-assemble all temporal keypoints. The unfiltered temporal keypoints, resulting from these operations, are harnessed to generate the final action segments. We extensively evaluate our method across three video benchmarks: 50salads, GTEA, and Breakfast. Our proposed approach consistently demonstrates substantial improvements over existing methods, establishing its superiority in video action segmentation. It achieves $F1@50$ scores (one of the key performance metrics for this task) of* 79.5%, 83.4%, *and* 60.5%, *respectively, v.s. previous state-of-the-art* 78.5%, 79.8% *and* 57.4%.

## 1. Introduction

Understanding human actions from visual sensors has been regarded as a long-standing crucial task in a variety of real-world applications, such as surveillance video analysis [4, 10, 11, 55], human-robot interaction [11, 43], autonomous driving, sports Analysis [50], etc. In the past few years, researchers have made great efforts on segmenting human actions in a frame-wise classification manner. Specifically, each frame of the video is densely labeled as a pre-defined human action category, based on the complete video stream, which is shown in Figure 1. However, existing methods [1,22,48,51] under this manner are still suffering from several mis-classification problems, such as over-segmentation or boundary-misalignment. We illustrate the above problems in two aspects:

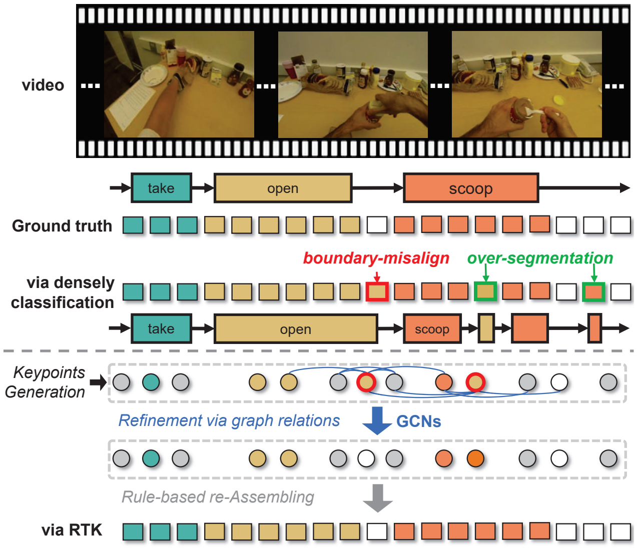


Figure 1: Comparison of frame-wise classification methods and keypoints based segmentation methods. The untrimmed video clip of *making peanut butter bread* contains three actions. Specifically, the colors indicate different categories of actions, while the white indicates *background* and grey is for *boundary point*. Frame-wise classification methods are prone to over-segmentation and boundary misalignment, while keypoints based method effectively alleviates these problems by refining keypoints using contextual information.

Frame-wise classification does not globally consider the semantics of each temporal action segment. For the frame-

[*] Corresponding author

wise classification loss, the goal of the function is always to assign the pre-defined categories to each video frame independently, regardless of whether the arrangement or duration of temporal actions are semantically appropriate. For example, in Figure 1, there is one frame mis-classified in *scoop* since the effects of video motion blur or other noises, which does not produce a large loss in frame-wise classification. Moreover, no matter where the mis-classification occurs, the loss is always the same. However, the overall meaning of the video changed considerably when the central mis-classified frame occurs, because the long *scoop* is split into three short actions (*scoop*, *open*, *scoop*). Although the smoothing-based methods [22,48] make great efforts on dynamically voting the predictions on a local region to alleviate such errors (*i.e.*, over-segmentation). However, due to the varying duration of temporal actions, the range of smoothness is often difficult to determine.

On the other hand, capturing long-term dependencies frame-wisely is computationally expensive. As the common practice, existing methods use dilated convolution [22] with a large dilation rate, or a local sliding-window based self-attention module [51], or casting multiple frames into several high ordered embeddings for further refinement [1]. However, the above methods either suffer from the size and shape of the convolution kernel or sliding window [22,51], or the inaccurate representation of video actions caused by over-segmentation in frame-level classification.

In this paper, we propose a video action segmentation framework via contextually Refined Temporal Keypoints (RTK), which treats actions as keypoints. Temporal keypoint-based methods, as first explored and advocated in this work, use sparse candidate points to represent actions, which is a high-order semantic representation. In addition, by constructing graph relationships between temporal keypoints, it is also easier to capture the long-term dependencies between actions. RTK has three main benefits:

**(i)** Detecting temporal keypoints implies localization of temporal actions, which implies that the model focuses on discriminating the global-to-local location of each action, rather than independently classifying each frames.

**(ii)** Detecting keypoints facilitates further exploration of contextual information between actions. Compared to analyze video frame-level relations, analyzing keypoint-level relations can provide higher-order semantic information, which can also help save the cost of capturing long-term dependencies. For specific, we construct the graph structure on sparse temporal keypoints while adopting a graph matching module for assistance, in modeling fine-grained action-level relationships in videos.

**(iii)** For the reason that boundary points are directly localized and optimized in the pipeline, RTK leads to precise action boundaries and avoids boundary-misalignment problems, compared to frame-wise classification.

We evaluate the framework on three popular datasets for temporal action segmentation: 50salads [38], GTEA [8], and Breakfast [14] dataset. RTK utilizes a modified ASFormer [51] architecture with several extra modules(keypoints generatation heads, graph matching modules) and achieves $83.4\%$ $F1@0.50$ scores on GTEA, $79.5\%$ on 50salads, and $60.5\%$ on Breakfast, which improves the segmentation baselines by about $2 \sim 4\%$ and significantly alleviating the over-segmentation problems.

## 2. Related Work

**Keypoints detectors.** In past few years, keypoints based methods have achieved impressive success in a plethora of computer vision tasks, including human pose estimation [30, 41], visual object detection [6, 15, 40], human action localization [26,27,39], video action detection [24,44], skeleton-based action recognition [28, 35], or unsupervised boundary detection [5]. However, the role of keypoint detectors in dense labelling tasks including temporal human action segmentation has not been fully explored. In our proposed solution, we utilize temporal keypoints to represent actions, which also construct stronger prior constraints. These temporal keypoints with global contextual relations effectively remove the out-of-context actions and significantly relief over-segmentation errors.

**Fine-grained relations between temporal actions.** Studying the relationship between actions has always been an important part of action segmentation tasks. Earlier works have studied the hidden Markov model [19, 36] to model the statistical dependences of actions. A few existing studies [1, 21] have explored non-local correlation among actions. However, fine-grained structural relationships of temporal action keypoints, represented by sophisticated structures such as keypoints graphs, still remains inadequately studied.

**Deep graph matching on computer vision.** Early deep graph matching methods have been proved effective on extracting dedicated node / edge features or affinity models. Researchers make great efforts [32,54] to explore advanced pipelines, where graph embedding [9, 45, 52], graph connectivity learning [53], geometric learning [9,56]. Most of them uses a common technique refers to the so-called graph convolutional networks (GCN) [49]. In GCN, node features are aggregated from adjacent neighbors and different nodes with learnable parameters. GCN based graph matching models [45,46] are developed for deep node / edge embeddings by exploiting high-order proximity jointly. Therefore, deep embeddings in graph matching models are very suitable for extracting contextual representations between keypoints. For this reason, we adopt the graph matching module behind the keypoints generator to refine the over-complete keypoints candidates with structured embeddings.

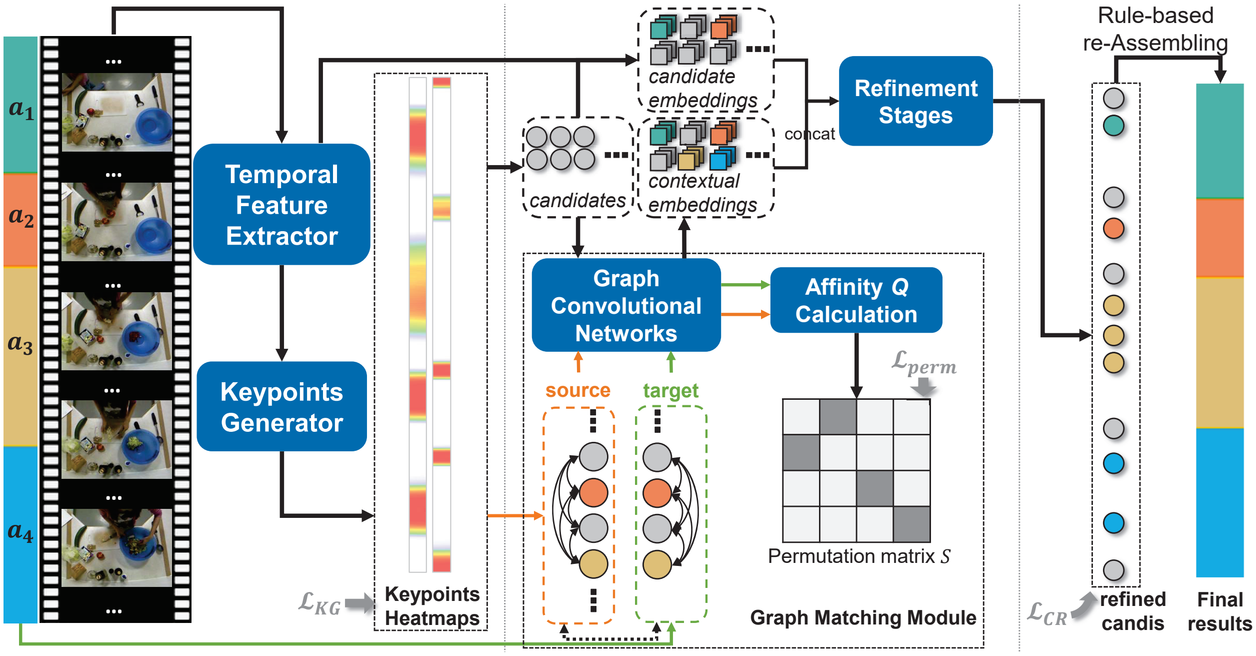


Figure 2: Overview of action segmentation by Refined Temporal Keypoints (RTK) pipeline. There are three main parts: ***left***: Temporal feature extraction and initial temporal keypoint generation. ***middle***: Action candidates refinement via graph matching module. ***right***: Assembling refined candidates to final segmentation results guided by human-designed rules.

## 3. Method

This section elaborates on the technical details of the proposed Refined Temporal Keypoints (RTK). An overview of the whole model can be found in Figure 2.

### 3.1. Initial Temporal Keypoint Generation (KG)

Rather than separately predicting the semantic category of each video frame, we develop a temporal keypoint based approach for detecting action segments. Suppose that $A = \{a_1, a_2, \ldots, a_N\}$ is a series of actions of an untrimmed video $\mathcal{V}$ with $T$ video frames (or snippets). It is crucial to ensure the temporal keypoints are well-defined, both visually distinguishable to other points and informative for the interested task. In our formulation, two kinds of temporal points are defined: *boundary points* that correspond to the starting or ending of an action, and *action points* that are temporally middle points of a ground-truth action.

**Boundary points.** The category of a boundary point is susceptible to the inherent ambiguity of action boundaries. Thus, we opt for a category-agnostic definition for all boundary points. Probably the single most important difficulty to localize such points is the inherent variability of a specific action's transition into another. It is always challenging to localize the boundary points at a frame-wise level of accuracy, even for a human annotator. Instead we set a small neighborhood around a true boundary points to be all true, and penalize a false prediction accordingly. Formally, splating all ground truth points $p$ onto a category-agnostic heatmap $Y^b$ using a 1D Gaussian kernel $e^{-(x-p)^2/2\sigma_b^2}$, where $\sigma_b$ is an object size-adaptive standard deviation [6]. It is proportional to a hand-designed region length $r_b$. If two Gaussians of the same class overlap, the value will depend on the element-wise maximum [2]. We adopt a focal loss style objective [25], formalized as following:

$$\mathcal{L}_f = \frac{1}{K}\sum_t \begin{cases} -(1-\gamma)\,(1-\hat{y}_t)^\alpha \log\,\hat{y}_t & \text{if } y_t = 1,\\ -\gamma\,(1-y_t)^\beta \hat{y}_t^\alpha \log(1-\hat{y}_t) & \text{otherwise,}\end{cases} \qquad (1)$$

where $\hat{y}_t$ is the predicted probability at time $t$ on the localization heatmaps, and $y_t$ is the ground-truth heatmap augmented with the un-normalize Gaussians. $K$ is the number of positive keypoints. $\alpha$, $\beta$, $\gamma$ are hyper-parameters that control or balance the contribution of samples.

**Action points.** Similarly, sparsely annotating the mid-point of an action segment tends to lead severe imbalance among data. The mid-point is always a true action point. Beyond that, in our implementation the temporal neighborhood (say 30% of the action segment) around an action point is also treated as positive. In addition, in order to alleviate the overfitting problem caused by insufficient training data, the heatmap $\hat{Y}^c$ of action points is devised to be decoupled into the product of two heatmaps: a category-agnostic localization heatmap $\hat{Y}^{cl}$ and a category-probability heatmap $\hat{Y}^{cc}$. Formally we have:

$$\hat{Y}^c_{t,i} = \hat{Y}^{cl}_t * \hat{Y}^{cc}_{t,i}, \quad t = 1 \ldots T,\ i = 1 \ldots C. \qquad (2)$$

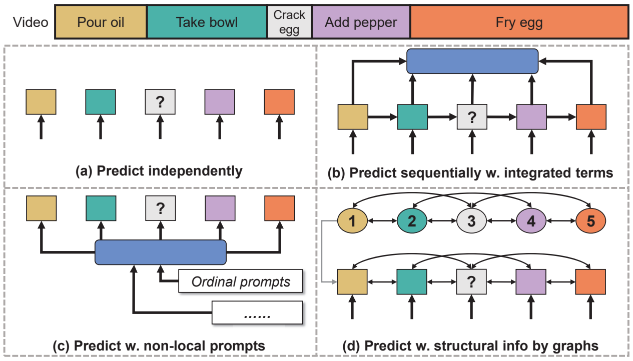

Video
Pour oil
Take bowl
Crack egg
Add pepper
Fry egg
?
(a) Predict independently
(b) Predict sequentially w. integrated terms
Ordinal prompts
......
(c) Predict w. non-local prompts
1
2
3
4
5
(d) Predict w. structural info by graphs

Figure 3: Comparison of existing methods (a-c) and our proposed RTK (in (d)). The illustrated example of activity *making fried egg* is composed of five actions. Conventional segmentation approaches [22, 51] in (a) classify each frame independently; HASR [1] and Br-prompt [21] shown in (b) and (c) either recurrently harnesses historical context or enforces the model to be non-local (such as the ordinal prediction as in (c)). In contrast, our RTK in (d) exploits even more fine-grained high-order relationship between action segments represented by graphs.

The generating procedure of $\hat{Y}^{cl}$ is similar to $\hat{Y}^{b}$, while the penalty of 1D Gaussian kernel is $e^{-(x-p)^2/2\sigma_c^2}$ for each $p$ that $Y_p^{cl} = 1$, where $\sigma_c$ is another object size-adaptive standard deviation which is proportional to corresponding length $L_i$ of action $a_i$. Following the definition, $\hat{Y}^{cc}$ the same as the logit-probabilities in frame-wise classification.

Putting all together, the loss function is a combination of action points and boundary points:

$$\mathcal{L}_{KG} = (\lambda_1 \mathcal{L}_{loc}^{c} + \lambda_2 \mathcal{L}_{cat}^{c}) + \lambda_3 \mathcal{L}_{loc}^{b}, \tag{3}$$

where $\mathcal{L}_{loc}^{c}$ is the Focal loss [25] defined in the style Equation (1) for $\hat{Y}^{cl}$, while $\mathcal{L}_{cat}^{c}$ is the Cross Entropy loss for $\hat{Y}^{cc}$ to determine the categories of action points. Similarly, $\mathcal{L}_{loc}^{b}$ is another Focal loss [25] for $\hat{Y}^{b}$. $\lambda_1, \lambda_2$ and $\lambda_3$ are model hyper-parameters to determine the contribution of different losses.

## 3.2. Refinement via Graph Matching (GM)

During inference, the estimated heatmaps $\{\hat{Y}^{b}, \hat{Y}^{c}\}$ generate boundary or action keypoints via simple thresholding. In specific, the heatmaps undergo a watershed algorithm [42]. All the locally maximal peaks that exceed a certain threshold are marked as candidates of boundary or action keypoints. However, these candidate points are independently predicted from each other. This implies that the inter-point contextual information among the candidate points is not adequately exploited. This motivates the additional graph matching procedure as described in this section that further forges these initial temporal points.

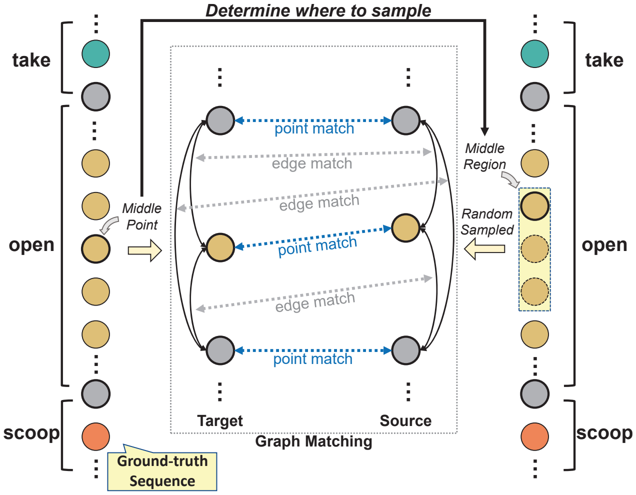


Figure 4: Example of graph matching during training. Points colored in *grey* indicate boundary points. Target graph is determined by the ground-truth segmentation sequence, while the nodes from source graph are sampled from detected keypoints. The graph matching between the source graph and the target graph means that all homologous points are clustered together and tightly connected by the structural properties of the graphs.

A few existing studies [1, 21] have explored non-local correlation among actions. For example, Hierarchical Action Segmentation Refiner (HASR) [1] sequentially improves initial actions with recurrent neural networks (RNNs). The work in [21] crafts multiple text prompts (particularly, statistical prompt that counts the entire actions, ordinal prompt that estimates the temporal position of each action, and integrated prompt that demands holistic matching between video / all text queries), thereby enforcing the model to be context-aware. However, fine-grained structural relationships, represented by sophisticated structures such as graphs, still remains inadequately studied. Figure 3 graphically contrasts the pipelines for different methods.

**Target graphs.** Figure 4 illustrates the proposed idea of harnessing an auxiliary task (*i.e.*, graph matching among temporal points) for learning enhanced contextualized representations. As seen, for each video a target graph is built from manually-annotated points (*i.e.*, boundary-points and mid-points), while directed edges are defined among all points. This way generates an effective representation of fine-grained information upon action level.

**Source graphs.** During predicting boundary / action points, we can always obtain an over-complete candidate set by choosing proper threshold of confidences. Suppose there are totally $M$ points in the ground-truth annotation for a video, we practically find that randomly sampling exactly $M$ points from the over-complete candidates generated from previous step can make the optimization more tractable, without much performance sacrifice. Specifically, we follow the definition of keypoints localization in Section 3.1, points on the source graph should match their corresponding regions. The failure that mis-detecting points in the region is regarded as the bottleneck of over-completed Keypoint Generation, rather than that to be solved by graph matching module. When suffering this special case, the source graph is constructed by a compromise strategy which to sample the extra points to match the target points.

The idea underlying the proposed graph matching is to treat the ground-truth "target" as a *matching template*, and force the detected points to learn the formation of target graphs. During the training, the graph matching module makes efforts on finding the second-order similar graph structure with ground-truth at the action level. At the same time, Refinement Stages (RS) (shown in Figure 2) exploits the graph matching module as high-order contextual embeddings extractor, further for refining action candidates. The graph matching module is supervised by the permutation loss [45] $\mathcal{L}_{perm}$, which is described with details in the supplemental material. However, due to the limitation of computation and filling the quantitative gap caused by randomly sampling, it is necessary to limit the number of action candidates. In Table 4, we further analyze the influence of the number of selected action candidates on performance.

Now we can describe the whole pipeline of Refinement. Suppose there are $N$ candidates selected from the keypoint generator. The candidate embeddings of action candidates are denoted by $\mathcal{I} \in \mathbb{R}^{d \times N}$ and the higher-order contextual embeddings from GCNs are denoted by $\mathcal{G} \in \mathbb{R}^{d \times N}$, $d$ is the dimension of embeddings. We adopt several stacked convolutional layers $\mathcal{N}'$ to aggregate embeddings:

$$\mathcal{G}' = \mathcal{N}'(cat(\mathcal{G}, \mathcal{I})), \tag{4}$$

where $\mathcal{G}'$ is the refined embeddings, while $cat$ indicates concatenation operation. Finally, $\mathcal{G}'$ are then used to reclassify for further predictions by a cross-entropy loss $\mathcal{L}_{CR}$. The final loss function is a combination:

$$\mathcal{L} = \mathcal{L}_{KG} + \mathcal{L}_{perm} + \mathcal{L}_{CR}. \tag{5}$$

### 3.3. Refinement via Rule-based re-Assembling (RA)

This section describes a post-processing procedure of detected points for precisely representing action segmentation results. The idea is to alternately arrange boundary / action points in chronological order. A single action point represents the category of the action, while the boundary point represents the position where action changes. To satisfy this, a simple and effective approach is to merge or remove over-complete points according to some hand-designed rules, called Rule-based re-Assembling (RA). Specifically, non-alternating points are merged into one point by weighted average, while keeping the category with the highest confidence score.

Formally, supposed that the sorted set of detected action points is $S_0 = \{(d_1, c_1), \dots, (d_{N_0}, c_{N_0})\}$, $d_i < d_{i+1}$, where $d_i$ and $c_i$ indicate coordinate and category respectively. Similarly, the set of boundary points is $S_1 = \{b_1, \dots, b_{N_1}\}$, $b_i < b_{i+1}$, where $b_i$ indicates coordinate. The confidence scores of points are omitted for convenience. The objective can be formally written as below:

$$N_0 + 1 = N_1, \tag{6a}$$

$$b_1 = 1, b_{N_1} = T + 1, \tag{6b}$$

$$b_i \le d_i < b_{i+1} \le d_{i+1} < b_{i+2}, \quad i = 1 \dots N_0 - 1, \tag{6c}$$

$$c_i \ne c_{i+1}, \quad i = 1 \dots N_0 - 1, \tag{6d}$$

We simply decompose the RA into four steps:**(i)** Merge action points between every two adjacent boundary points. **(ii)** Merge boundary points between every two adjacent action points. **(iii)** Iteratively merging adjacent actions to satisfy Equation 6d. (*i.e.*, removing boundary points between two adjacent action points with same category, and remaining the action point with higher confidence). **(iv)** Casting the alternating action and boundary points to construct the segmentation results. Since RA is non-differentiable and parameters-free, it can be directly applied to the generated keypoints to construct the final segmentation result.

## 4. Experiments

### 4.1. Implementation Details

**RTK architecture.** The experiments adopt a fixed I3D [3] network as the pre-trained feature extractor $\Phi$. For better generalizing keypoints heatmap, we adopted a modified ASFormer [51] backbone with its encoder layers and light-weighted decoders. The classification and keypoints generation are integrated into one shared head for all encoder and decoders. All the hyper-parameters not mentioned in this paper are following the settings of their original paper (*i.e.*, ASFormer [51], and PCA Graph Matching models [47]) while the trainable parameters are randomly initialized and trained from scratch under the default setting of Pytorch [33] without pre-training any external dataset. More details are found in the supplemental material.

**Hyper-parameters settings.** Referring to the practice of focal loss [25] on other tasks such as object detection [6, 15, 40], we choose $\alpha = 2$, $\beta = 4$, $\gamma = 0.75$ in $\mathcal{L}^c_{loc}$, while specially in $\mathcal{L}^b_{loc}$ the hyper-parameter $\gamma = 0.8$ is for balancing positive and negative boundary points. For better performance, we set $\lambda_1 = \lambda_3 = 100, \lambda_2 = 1$. As for keypoints generating, $r_b = 33, \sigma_b = 0.5 r_b, r_c = 0.8l, \sigma_c = 0.25 r_c$, where $l$ is the length of the corresponding action segment. Moreover, the length of positive region of action points is set to $0.32l$. During the selection of candidates, the threshold is $0.4$ for action points and $0.2$ for boundary points in order to further reduce the over-segmentation points due to ambiguity.

| | **50salads** | | | | | **GTEA** | | | | | **Breakfast** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1@{10,25,50} | | | Edit | Acc | F1@{10,25,50} | | | Edit | Acc | F1@{10,25,50} | | | Edit | Acc |
| SCNN [18] | 32.3 | 27.1 | 18.9 | 24.8 | 54.9 | - | | | | | - | | | | |
| IDT+LM [34] | 44.4 | 38.9 | 27.8 | 45.8 | 48.7 | - | | | | | - | | | | |
| Bi-LSTM [37] | 62.6 | 58.3 | 47.0 | 55.6 | 55.7 | 66.5 | 59.0 | 43.6 | - | 55.5 | - | | | | |
| DTCN [17] | 52.2 | 47.6 | 37.4 | 43.1 | 59.3 | - | | | | | - | | | | |
| ST-CNN [18] | 55.9 | 49.6 | 37.1 | 45.9 | 59.4 | 58.7 | 54.4 | 41.9 | - | 60.6 | - | | | | |
| TResNet [12] | 69.2 | 65.0 | 54.4 | 60.5 | 66.0 | 74.1 | 69.9 | 57.6 | 64.4 | 65.8 | - | | | | |
| TRN [20] | 70.2 | 65.4 | 56.3 | 63.7 | 66.9 | 77.3 | 71.3 | 59.1 | 72.2 | 67.8 | - | | | | |
| TDRN [20] | 72.9 | 68.5 | 57.2 | 66.0 | 68.1 | 79.2 | 74.4 | 62.7 | 74.1 | 70.1 | - | | | | |
| ED-TCN [17] | 68.0 | 63.9 | 52.6 | 59.8 | 64.7 | 72.2 | 69.3 | 56.0 | - | 64.0 | - | | | | 43.3 |
| BCN [48] | 82.3 | 81.3 | 74.0 | 74.3 | 84.4 | 88.5 | 87.1 | 77.3 | 84.4 | 79.8 | 68.7 | 65.5 | 55.0 | 66.2 | 70.4 |
| ASRF [13] | 84.9 | 83.5 | 77.3 | 79.3 | 84.5 | 89.4 | 87.8 | 79.8 | 83.7 | 77.3 | 74.3 | 68.9 | 56.1 | 72.4 | 67.6 |
| HASR [1] | 86.6 | 85.7 | 78.5 | 81.0 | 83.9 | 89.2 | 87.2 | 74.8 | 84.5 | 76.9 | 74.7 | 69.5 | 57.0 | 71.9 | 69.4 |
| ETSN [23] | 85.2 | 83.9 | 75.4 | 78.8 | 82.0 | 91.1 | 90.0 | 77.9 | 86.2 | 78.2 | 74.0 | 69.0 | 56.2 | 70.3 | 67.8 |
| G2L [10] | 80.3 | 78.0 | 69.8 | 73.4 | 82.2 | 89.9 | 87.3 | 75.8 | 84.6 | 78.5 | 76.3 | 69.9 | 54.6 | 74.5 | 70.8 |
| MS-TCN [7] | 76.3 | 74.0 | 64.5 | 67.9 | 80.7 | 85.8 | 83.4 | 69.8 | 79.0 | 76.3 | 52.6 | 48.1 | 37.9 | 61.7 | 66.3 |
| MS-TCN++ [22] | 80.7 | 78.5 | 70.1 | 74.3 | 83.7 | 88.8 | 85.7 | 76.0 | 83.5 | 80.1 | 64.1 | 58.6 | 45.9 | 65.6 | 67.6 |
| ASFormer [51] | 85.1 | 83.4 | 76.0 | 79.6 | 85.6 | 90.1 | 88.8 | 79.2 | 84.6 | 79.7 | 76.0 | 70.6 | 57.4 | 75.0 | **73.5** |
| RTK | **87.4** | **86.1** | **79.5** | **81.4** | **85.9** | **91.2** | **90.6** | **83.4** | **87.9** | **80.3** | **76.9** | **72.4** | **60.5** | **76.1** | 73.3 |

Table 1: Evaluation results of action segmentation methods on three datasets with I3D [3] features. Score in bold indicates the best performance.

**Training process.** The training process of RTK is step-by-step, since the hierarchical dependencies between modules in whole pipeline. There are two main dependencies: $(a)$ The Graph Matching (GM) depends on semantic features to further learn structural information in keypoints graphs. $(b)$ The Refinement Stages (RS) depends on well-trained GM module to further improve the keypoints quality. Therefore, for total $120$ training epochs, the first $80$ only trains Keypoints Generator (KG), and then joins the GM module with $16\times$ batch size for $20$ epochs, and the last $20$ epochs jointly trains all modules from end to end.

**Datasets.** The evaluation results are based on 3 popular datasets: Georgia Tech Egocentric Activities (GTEA) [8], 50salads [38] and the Breakfast dataset [14].

**Evaluation Metrics.** For all the dataset, the following evaluation metrics are reported as in [7, 16, 48]: frame-wise **accuracy**, the segmental **edit score**, and the segmental **F1 scores** at temporal intersection over union (tIoU) thresholds of $0.10$, $0.25$ and $0.50$, denoted by $F1@\{0.10, 0.25, 0.50\}$. In this paper, segmental F1 scores and segmental edit scores are more important than frame accuracy. Because over-segmentation errors are always reflected on these segmental metrics, which is similar to the effect of mean average precision (mAP) in object detection. Furthermore, the **mean IoU scores** (MIoU) is discussed to demonstrate that RTK enables more accurate localization. Unlike the pixel-wise MIoU in semantic segmentation [29, 31], one calculates a segment-wise mean IoU score of a threshold(similar to thresholds in $F1$ scores) equals $0.0$. This metric can intuitively show the overall IoU value of all actions, thus reflecting the localization accuracy of all the actions.

## 4.2. Comparison of State-of-the-art Methods

Table 1 shows the comparison of the RTK to the state-of-the-art methods on three challenging benchmarks: 50Salads, GTEA and Breakfast datasets with I3D features. RTK performs better on all these metrics than modern methods, especially the $F1@50$ scores (RTK achieves the state-of-the-art $F1@50$ scores of $83.4\%$, $79.5\%$, and $60.5\%$ on three datasets, respectively). Higher $F1@50$ scores means that RTK predicts more precise action segment boundaries and categories, which results in a larger area of intersection with the ground-truth actions.

## 4.3. Ablations on proposed modules

We perform ablation experiments to test the effectiveness among all the proposed modules between the frame-wise classification based methods and the proposed method. Keypoints based methods have natural advantages when dealing with over-segmentation and boundary misalignment issues — by directly estimating the precise keypoints and utilizing contextual information of the keypoints. To understand the contribution of multiple proposed modules, we specify the following baselines and RTK variants:

**Keypoints based models.** In this series of experiments, the base models are replaced the frame-wise linear classification layers to the keypoints generators, along with post-processing described in Section 3.3 to assemble the keypoints to the final segmentation results.

**Refinement Stages.** In original frame-wise classification, RS effectively alleviates over-segmentation errors, while in keypoints generators, RS acts as a cascade module to further improve the performance of keypoints (seen in Table 6).

| KP | RS | GCN | GM | F1@{10,25,50} | | | Edit | Acc |
|---|---|---|---|---|---|---|---|---|
| | | | | 83.4 | 81.7 | 71.9 | 77.1 | 78.7 |
| ✓ | | | | 88.2 | 85.7 | 76.5 | 82.4 | 79.4 |
| ✓ | ✓ | | | 90.8 | 88.7 | 82.9 | 86.6 | 80.1 |
| ✓ | ✓ | ✓ | | 90.7 | 88.7 | 82.9 | 86.5 | 80.2 |
| | ✓ | | | 90.1 | 88.8 | 79.2 | 84.6 | 79.7 |
| | ✓ | ✓ | | 89.7 | 87.4 | 78.6 | 82.5 | 79.3 |
| | ✓ | ✓ | ✓ | 90.9 | 88.1 | 79.3 | 83.6 | 79.5 |
| ✓ | ✓ | ✓ | ✓ | **91.2** | **90.6** | **83.4** | **87.9** | **80.3** |

Table 2: Ablation study on GTEA dataset. "KP" means that whether adopts keypoint-based architecture; "RS" indicates the Refinement Stages, while "GCN" means that whether adds GCN features to Refinement Stages; "GM" indicates whether GCN features are supervised under the graph matching permutation loss.

| E.D. | type | F1@{10,25,50} | | | Edit | Acc |
|---|---|---|---|---|---|---|
| | CIE [52] | 80.9 | 80.1 | 72.1 | 76.4 | 73.5 |
| ✓ | CIE [52] | 81.3 | 80.8 | 74.5 | 79.6 | 75.7 |
| | PCA [45] | 90.7 | 88.4 | 78.6 | 82.9 | 78.5 |
| ✓ | PCA [45] | **91.2** | **90.6** | **83.4** | **87.9** | **80.3** |

Table 3: Ablation studies of the specific graph matching module for RTK on the GTEA dataset. *E.D.* indicates the embeddings used in GM is whether detached during the backward.

**Graph matching modules.** We study the influence of GCN layers and permutation loss in RTK. Experiments have proved that the structural information between keypoints is helpful for better performance of keypoints.

In Table 2 we show the main ablation results, which indicate that keypoints based methods and the extra modules make positive contributions to the final performance in all the settings, which is consistent to our claims.

## 4.4. Ablations in Graph Matching modules

In our design, we tried two graph matching methods: CIE [52] model (which use proposed Hungarian Attention mechanism to consistent with the strategy used in the testing stage) and PCA [45] model (an embedding-based method). Table 3 shows the quantitative results, which indicates that PCA model make positive contributions to the final performance. It is reasonable because CIE focuses on the cross information fusion of source and target graphs rather than extracting embedded information independently. However, the simple GCN framework in PCA is more suitable for the settings of extracting contextual information without target graph. Furthermore, we also study the influence of the gradients of graph matching module to the previous keypoints generator and feature extractor. As shown in Table 3, the gradients of graph matching module have adverse influence on the features for generating keypoints. This may be attributed to the fact that the graph matching module is more biased towards the global structural information and the clustering information of each action, rather than the patterns and localizations that distinguish different kinds of actions.

| $\alpha$ | F1@{10,25,50} | | | Edit | Acc |
|---|---|---|---|---|---|
| 0.001 | 83.5 | 80.7 | 74.6 | 78.5 | 81.3 |
| 0.01 | 88.9 | 86.3 | 81.7 | 83.2 | 79.4 |
| 0.1 | 91.2 | **90.6** | **83.4** | **87.9** | 80.3 |
| 1.0 | **91.4** | 90.2 | 82.7 | 86.5 | **80.4** |

Table 4: Impact of the number of selected over-complete candidates($\propto \alpha$) to Refinement Stages for RTK on GTEA.

| | #params | FLOPs | GPU Mem. |
|---|---|---|---|
| ASFormer [51] | 1.134M | 6.80G | $\sim$ 3.5G |
| RTK | 0.779M | 7.41G | $\sim$ 2.9G |

Table 5: Comparison of RTK and ASFormer with respect to the number of parameters, FLOPs and GPU memory cost of a video input length $= 3000$.

| method | RS | action Acc/Rec | | boundary Acc/Rec | |
|---|---|---|---|---|---|
| ASFormer [51] | ✓ | 81.5 | 88.1 | 39.5 | 48.8 |
| +ASRF [13] | ✓ | 83.3 | 88.5 | 53.9 | 67.6 |
| RTK | | 88.4 | **92.6** | 86.3 | **80.7** |
| RTK | ✓ | **90.5** | 90.7 | **92.1** | 71.1 |

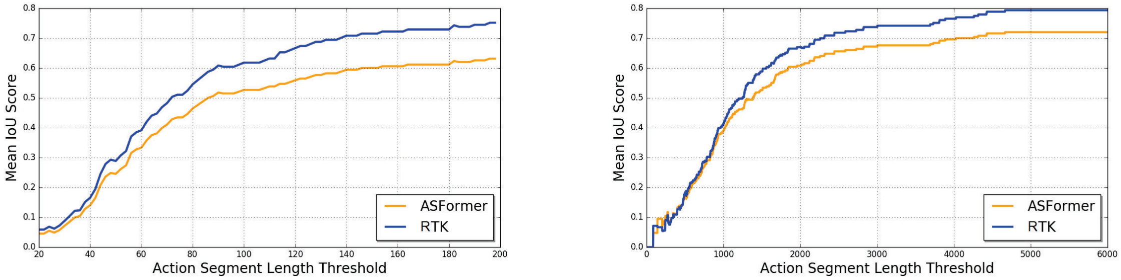
Table 6: Keypoints quality evaluation between the baselines and keypoints detection methods on 50salads [38] dataset. *RS.* indicates that whether uses Refinement Stages. *Acc* indicates accuracy while *Rec* indicates recall.

## 4.5. Ablations on maximum number of candidates in RS

In our experiments, the maximum number of candidates $N_{max} = \alpha T$ is proportional to the number of video frames $T$. Table 4 shows the impact of different number of candidates on GTEA. When the number of candidates drop down, the performance drops sharply, which means the absence of candidate points is fatal for action segment prediction. However, when the number of candidate points increases to a certain extent, most of the candidate points will be merged or deleted in post-processing, so it does not have much impact on the final prediction result. But the increase of candidates brings a huge computational cost on graph matching module. Therefore, we choose $\alpha = 0.1$ as the optimal choice.

## 4.6. Quantitative results of keypoints

To reveal the reason behind RTK's improvement, we design to evaluate the MIoU gain over baseline about the length of actions. Similar to $F1$ scores, the calculation of MIoU is based on averaging the IoU of all matched actions less than a certain length. Specially, in Figure 5a, RTK performs well on the actions around 30 to 100 frames on GTEA dataset. While in Figure 5b, the main gains of RTK



(a) GTEA dataset test split 1.

(b) 50salads dataset test split 2.

Figure 5: The curve of Mean IoU Scores v.s. an Action Segment Lengths Threshold on GTEA and 50salads dataset. Each point on the curve represents the average intersection-over-union (IoU) with the ground-truth actions of all actions below the action segment length threshold. Compared with baseline ASFormer methods, RTK performs better on actions at all scales.

| | F1@{10,25,50} | | | Edit | Acc |
|---|---|---|---|---|---|
| ASFormer | 94.1 | 92.0 | 83.0 | 91.6 | 81.2 |
| RTK | **94.9** | **93.8** | **87.8** | **92.8** | **84.4** |

Table 7: Complementarity of RTK using Br-prompt [21] features on the GTEA dataset.

| | F1@{10,25,50} | | | Edit | Acc |
|---|---|---|---|---|---|
| HASR | 90.4 | 88.8 | 79.3 | 85.3 | 78.5 |
| RTK | **91.2** | **90.6** | **83.4** | **87.9** | **80.3** |

Table 8: Complementarity of RTK and HASR [1] using ASFormer [51] on GTEA dataset.

come from the actions larger than 1000 frames on 50salads dataset. These results reveal that RTK improves MIoU on all scales of actions, which strongly support our claims.

Moreover, we evaluate the quality of keypoints on 50salads [38] dataset. The evaluation metrics of the action points and boundary points are defined as following: **(a)** Action points are regarded to be correct only if the category of the action segment is correct. **(b)** Boundary points are considered correct only if the points are covered over a radius of 5 frames by any ground-truth boundary points. The metric **(b)** is reasonable under 50salads [38] because there are few actions with a total duration less than 10 frames. There are several baseline methods to be considered: **(1)** Vanilla frame-wise classification based methods ASFormer [51] extract keypoints from frame-wise predictions. **(2)** ASFormer+ASRF [13] directly uses cross-entropy loss to train the classifier of boundary points. **(3)** Keypoints Generator (KG), with or without the Refinement Stage (RS) via graph matching module. As in Table 6, KG achieves better performance than cross-entropy based ASRF [13] and vanilla baselines. The significantly improved quality of keypoints means that RTK predicts and refines precise boundaries of temporal actions, consistent to the analysis in Section 3.3. Some of the over-complete keypoints will be deleted by RS, which slightly damages the recall of keypoints. However, the keypoint accuracy gains significant improvement in the overall result, which is also under an accuracy-recall tradeoff.

### 4.7. Comparision with existing methods

As shown in Figure 3, RTK is better than HASR [1] and Br-prompt [21] in modeling the structural relationships between temporal actions. In Table 8, HASR [1] and RTK (both using ASFormer [51] backbone) both model the action-level relationships and RTK performs better with all metrics. Moreover, RTK can be further improved with Br-prompt [21] high-level features, which is shown in Table 7.

### 4.8. Learnable Parameters and Computational Cost

Due to the simplified Decoders in ASFormer [51] backbone network and shared keypoints generation heads, the parameters and computational complexity of RTK are comparable to the baseline method, shown in Table 5. As a result, the memory cost of RTK is also smaller than ASFormer [51].

## 5. Conclusion

This paper describes a temporal segmentation framework via refined temporal keypoints to address a suite of research challenges in human action segmentation. Our technical solutions intrinsically segment actions as multiple keypoints and leverage the contextual embedding of the sparse keypoints from graph matching models to effectively guide the action segmentation process, which improve the accuracy and increase fine-grained structural awareness. Extensive experiments on several challenging datasets, as well as comprehensive quantitative evaluations, have demonstrated the superior performance of our method.

**Acknowledgement**: This work is supported by National Key R&D Program of China (2022ZD0160305) and Beijing Natural Science Foundation (Z190001).

# References

[1] Hyemin Ahn and Dongheui Lee. Refining action segmentation with hierarchical video representations. In *IEEE International Conference on Computer Vision*, pages 16282–16290, 2021.

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(1):172–186, 2021.

[3] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4733, 2017.

[4] Robert T. Collins, Christophe Biernacki, Gilles Celeux, Alan J. Lipton, Gérard Govaert, and Takeo Kanade. Introduction to the special section on video surveillance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):745–746, 2000.

[5] Zexing Du, Xue Wang, Guoqing Zhou, and Qing Wang. Fast and unsupervised action boundary detection for action segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 3313–3322. IEEE, 2022.

[6] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6568–6577, 2019.

[7] Yazan Abu Farha and Jürgen Gall. MS-TCN: multi-stage temporal convolutional network for action segmentation. In *IEEE Conference on computer Vision and Pattern Recognition*, pages 3575–3584, 2019.

[8] Alireza Fathi, Xiaofeng Ren, and James M. Rehg. Learning to recognize objects in egocentric activities. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3281–3288, 2011.

[9] Matthias Fey, Jan Eric Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. Deep graph matching consensus. In *International Conference on Learning Representations*, 2020.

[10] Shang-Hua Gao, Qi Han, Zhong-Yu Li, Pai Peng, Liang Wang, and Ming-Ming Cheng. Global2local: Efficient structure search for video action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 16805–16814, 2021.

[11] Mahmudul Hasan and Amit K. Roy-Chowdhury. A continuous learning framework for activity recognition using deep hybrid feature models. *IEEE Trans. Multim.*, 17(11):1909–1922, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[13] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *IEEE Winter Conference on Applications of Computer Vision*, pages 2321–2330, 2021.

[14] Hilde Kuehne, Ali Bilgin Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 780–787, 2014.

[15] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *European Conference on Computer Vision*, volume 11218, pages 765–781, 2018.

[16] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1003–1012, 2017.

[17] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1003–1012, 2017.

[18] Colin Lea, Austin Reiter, René Vidal, and Gregory D. Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European Conference on Computer Vision*, volume 9907, pages 36–52, 2016.

[19] Jun Lei, Guohui Li, Jun Zhang, Qiang Guo, and Dan Tu. Continuous action segmentation and recognition using hybrid convolutional neural network-hidden markov model model. *IET Comput. Vis.*, 10(6):537–544, 2016.

[20] Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6742–6751, 2018.

[21] Muheng Li, Lei Chen, Yueqi Duan, Zhilan Hu, Jianjiang Feng, Jie Zhou, and Jiwen Lu. Bridge-prompt: Towards ordinal action understanding in instructional videos. *CoRR*, abs/2203.14104, 2022.

[22] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[23] Yunheng Li, Zhuben Dong, Kaiyuan Liu, Lin Feng, Lianyu Hu, Jie Zhu, Li Xu, Yuhan wang, and Shenglan Liu. Efficient two-step networks for temporal action segmentation. *Neurocomputing*, 454:373–381, 2021.

[24] Yixuan Li, Zixu Wang, Limin Wang, and Gangshan Wu. Actions as moving points. In *European Conference on Computer Vision*, volume 12361, pages 68–84, 2020.

[25] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):318–327, 2020.

[26] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. BMN: boundary-matching network for temporal action proposal generation. In *IEEE International Conference on Computer Vision*, pages 3888–3897, 2019.

[27] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. BSN: boundary sensitive network for temporal action proposal generation. In *European Conference on Computer Vision*, volume 11208, pages 3–21, 2018.

[28] Jun Liu, Gang Wang, Ling-Yu Duan, Kamila Abdiyeva, and Alex C. Kot. Skeleton-based human action recognition with global context-aware attention lstm networks. *IEEE Trans. on Image Processing*, 27(4):1586–1599, 2018.

[29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[30] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, volume 9912, pages 483–499, 2016.

[31] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[32] Alex Nowak, Soledad Villar, Afonso S. Bandeira, and Joan Bruna. Revised note on learning quadratic assignment with graph neural networks. In *IEEE Data Science Workshop*, pages 229–233, 2018.

[33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[34] Alexander Richard and Juergen Gall. Temporal action detection using a statistical language model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3131–3140, 2016.

[35] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with multi-stream adaptive graph convolutional networks. *IEEE Trans. on Image Processing*, 29:9532–9545, 2020.

[36] Qinfeng Shi, Li Cheng, Li Wang, and Alexander J. Smola. Human action segmentation and recognition using discriminative semi-markov models. *Int. J. Comput. Vis.*, 93(1):22–32, 2011.

[37] Bharat Singh, Tim K. Marks, Michael J. Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1961–1970, 2016.

[38] Sebastian Stein and Stephen J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 729–738, 2013.

[39] Haisheng Su, Weihao Gan, Wei Wu, Yu Qiao, and Junjie Yan. BSN++: complementary boundary regressor with scale-balanced relation modeling for temporal action proposal generation. In *AAAI Conference on Artificial Intelligence*, pages 2602–2610, 2021.

[40] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: fully convolutional one-stage object detection. In *IEEE International Conference on Computer Vision*, pages 9626–9635, 2019.

[41] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

[42] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(6):583–598, 1991.

[43] Nam N. Vo and Aaron F. Bobick. From stochastic grammar to bayes network: Probabilistic parsing of complex activity. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2014.

[44] Limin Wang, Yu Qiao, and Xiaoou Tang. Video action detection with relational dynamic-poselets. In *European Conference on Computer Vision*, volume 8693, pages 565–580, 2014.

[45] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *IEEE International Conference on Computer Vision*, pages 3056–3065, 2019.

[46] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Combinatorial learning of robust deep graph matching: an embedding based approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[47] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.

[48] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. Boundary-aware cascade networks for temporal action segmentation. In *European Conference on Computer Vision*, volume 12370, pages 34–51, 2020.

[49] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, volume 97, pages 6861–6871, 2019.

[50] Fei Wu, Qingzhong Wang, Jian Bian, Haoyi Xiong, Ning Ding, Feixiang Lu, Jun Cheng, and Dejing Dou. A survey on video action recognition in sports: Datasets, methods and applications. *CoRR*, abs/2206.01038, 2022.

[51] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. *CoRR*, abs/2110.08568, 2021.

[52] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Learning deep graph matching with channel-independent embedding and hungarian attention. In *International Conference on Learning Representations*, 2020.

[53] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Deep latent graph matching. In *International Conference on Machine Learning*, volume 139, pages 12187–12197, 2021.

[54] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2684–2693, 2018.

[55] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph convolutional module for temporal action localization in videos. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10):6209–6223, 2022.

[56] Zhen Zhang and Wee Sun Lee. Deep graphical feature learning for the feature matching problem. In *IEEE International Conference on Computer Vision*, pages 5086–5095, 2019.