Adaptive Mutual Supervision for Weakly-Supervised Temporal Action Localization

Chen Ju, Peisen Zhao, Siheng Chen, Ya Zhang, Xiaoyn Zhang, Member, IEEE, Yanfeng Wang, and Qi Tian, Fellow, IEEE

Abstract—Weakly-supervised temporal action localization aims to localize actions from untrimmed long videos with only video-level category labels. Most previous methods ignore the incompleteness issue of Class Activation Sequences (CAS), suffering from trivial detection results. To tackle this issue, we propose a novel Adaptive Mutual Supervision (AMS) framework with two branches, where the base branch detects the most discriminative action regions, while the supplementary branch localizes the less discriminative action regions through an adaptive sampler. The sampler dynamically updates the inputs for the supplementary branch using a sampling weight sequence negatively correlated with the CAS from the base branch, thus encouraging the supplementary branch to localize the action regions underestimated by the base branch. To promote mutual enhancement between two branches, we further construct mutual location supervision. Each branch adopts the location pseudo-labels generated from the other branch as the localization supervision. By alternately optimizing two branches for multiple iterations, we progressively complete action regions. Extensive experiments on THUMOS14 and ActivityNet1.2 demonstrate that the proposed AMS method significantly outperforms state-of-the-art methods.

Index Terms—Temporal action localization, weak supervision, adaptive sampling strategy, mutual location supervision.

I. INTRODUCTION

TEMPORAL Action Localization (TAL) aims to localize actions from untrimmed videos, and plays an important role in video understanding. Although several researches [1], [2], [3], [4], [5], [6], [7] have shown promising results on strongly-supervised temporal action localization, the annotations in form of precise action boundaries are both time-consuming and noisy. Weakly-supervised Temporal Action Localization (WTAL), which handles the same task but only requires video-level action category labels, has recently received increasing attention [8], [9], [10], [11], [12], [13], [14], [15].

To date in the literature, there exist two main frameworks for WTAL. The classification-based framework [8], [9], [10], [12] follows multiple instance learning (MIL), i.e. first training an action classifier with video-level category labels, then thresholding Class Activation Sequence (CAS) from the classifier to obtain action proposals; see Fig. 1(a). In this framework, only the classification objective is optimized. On the other hand, regarding the CAS as a noisy location cue, the self-training-based framework [16], [17], [18] iteratively thresholds the CAS of the current step to generate location pseudo-labels for the next step, and progressively refines results; see Fig. 1(b).

The CAS generated from the classifier, indicates the frame-level class-specific action probabilities, and becomes the key to the performance of above two frameworks. However, CAS has a well-known incompleteness issue, i.e. it only highlights the most discriminative action regions [13], [14], [15] which contribute significantly to action classification, e.g. pole passing in high jump. The reason for this issue is a fundamental difference in optimization objectives between classification and localization. More specifically, classification mainly relies on a few frames with top scores, namely, the most discriminative regions for optimization; while localization aims to mine complete action regions. As a result, the CAS output by the

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
classifier is usually sparse and incomplete. The location pseudo-labels and action proposals produced from this naïve CAS are both low-quality, causing trivial results in these two frameworks.

To tackle this incompleteness issue, this paper considers a novel adaptive mutual supervision (AMS) framework with two branches that are collaborative and complementary. The base branch uses the naïve CAS to localize the most discriminative action regions, which is similar to the above two frameworks. While the supplementary branch localizes the less discriminative action regions to complete localization results of the whole framework. Here, we regard the rest action regions except the most discriminative regions, as the less discriminative action regions, e.g. run-up in high jump. To achieve collaboration and complementarity, we introduce two core designs: an adaptive sampler and mutual location supervision; see Fig. 1(c).

The design rationale of the adaptive sampler is to scale up the input proportion corresponding to the less discriminative regions for the supplementary branch, such that this branch could focus more on these challenging action regions. Concretely, we feed the original video into the base branch, while utilizing the adaptive sampler to probabilistically select video snippets for the supplementary branch. Since the CAS from the base branch mainly detects the most discriminative action regions, the sampling probability sequence is designed to be dynamic and negatively correlated with the CAS of the base branch. That is, we over-sample the input snippets from low-CAS regions and under-sample those from high-CAS regions. As a result, the inputs for the supplementary branch mainly consist of the snippets corresponding to the low-CAS regions of the base branch, which prompts the supplementary branch to purposefully detect less discriminative action regions, thus completing the localized action locations.

To further promote mutual enhancement between the base and supplementary branches, we are also motivated to design mutual location supervision, which encourages each branch to explicitly optimize the localization objective with location pseudo-labels from the other branch. More specifically, both branches use video-level category labels as classification supervision; meanwhile, each branch leverages location pseudo-labels generated from the CAS of the other branch as localization supervision. For optimization, we here alternately freeze one branch, then train the other branch. In this process, the localization results of each branch are treated as the localization objective of the other branch, so that the complementary action regions of the two branches are combined to make the localization supervision more complete and precise.

To optimize the whole framework, we apply multiple iterations, since one single iteration brings limited improvements to detect less discriminative action regions. In each iteration, the adaptive sampler differentiates the inputs from these two branches, so that they could purposefully highlight different action regions. Then, mutual location supervision obtains more complete location supervision by pushing the CASs from two branches to be consistent. In the next iteration, the consistent CASs in turn force the adaptive sampler to further update the inputs for the supplementary branch, thereby exploring more missing action regions. Consequently, the proposed adaptive sampler and mutual location supervision jointly contribute to more complete results during progressive iterations. In summary, our contributions are as follows.

1) We introduce an adaptive mutual supervision framework (AMS) for weakly-supervised temporal action localization. AMS contains a base branch and a supplementary branch, both of which generate the location pseudo-labels for progressive iterative refinements.

2) We design a novel adaptive sampler, which encourages the supplementary branch to further detect the underestimated action regions of the base branch, thus making the localization results more complete.

3) We propose a novel mutual location supervision, which forces each branch to use location pseudo-labels obtained from the other branch, promoting mutual enhancement.

4) We validate the effectiveness of AMS on two widely-used benchmarks, THUMOS14 [19] and ActivityNet1.2 [20]. Our AMS framework outperforms previous state-of-the-art methods, both quantitatively and qualitatively.

We organize the rest of this research as follows. To better understand our motivation on adaptive mutual supervision, we first review the related work in Section II. In Section III, we propose the novel strategies about adaptive sampler and mutual location supervision. Section IV validates the proposed method by making comparisons with the existing methods. We further perform extensive ablations to reveal the effectiveness of each component. Finally, the conclusion is drawn in Section V.

II. RELATED WORK

This section reviews previous works that motivate the proposed method. We could divide those works into three groups: strongly-supervised temporal action localization, and weakly-supervised temporal action localization, the adaptive sampling strategy. We next review these works respectively.

A. Strongly-Supervised Temporal Action Localization (STAL)

Strongly-supervised temporal action localization relies on precise boundary labels and action category labels to localize action instances. The popular pipelines could be summarized as the top-down and bottom-up frameworks. The top-down framework [1], [4], [5], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30] first pre-defines abundant anchors following the prior knowledge of action distribution; then utilizes fixed-length sliding windows to generate initial proposals; finally adopts the boundary adjustment module to refine results. Typically, TURN [5] and TAL-Net [1] explored the effects of the contextual information and dilated temporal convolution on the localization performance respectively. The bottom-up framework [2], [3], [6], [7], [31], [32], [33], [34], [35], [36], [37], [38] first predicts frame-level actionness or boundary probabilities, then groups the localized start points and end points as action proposals. Typically, BMN [3] listed all possible proposal groups, then ranked each proposal via evaluating the IoU with all ground truths. BUMR [31] constructed constraints between the action, start, and end curves to reduce invalid proposal groups. BC-GNN [32] also introduced graph convolution to group the most matching start and end points. G-TAD [33] modeled videos as the graph
based on temporal and semantic relationships to enhance the feature continuity. In general, the top-down framework can discover most action instances with few omissions, while the bottom-up framework can flexibly adjust boundaries and produce more precise predictions.

For better performance, CTAP [39], MGG [40], AFNet [41], and A2Net [42] further introduced various fusion strategies to complement these two frameworks. PBRNet [43] proposed a coarse-to-fine strategy to progressively refine proposal boundaries. To model the relationship between action proposals, P-GCN [44], [45] and RAM [46] utilized graph convolution and self-attention mechanisms to construct the non-local networks for better feature embedding, respectively.

However, all strongly-supervised methods demand precise action location annotations, which is time-consuming and less practical in real-world scenarios. To reduce the annotation cost, this work studies the same task in weakly-supervised settings.

B. Weakly-Supervised Temporal Action Localization (WTAL)

Weakly-supervised temporal action localization only needs video category labels for training. Inspired by Class Activation Map [47] in object detection, early WTAL methods [8], [10], [48] usually adopted the classification-based framework. That is, first utilize the video-level category labels to train an action classifier; and then calculate Class Activation Sequence (CAS) based on the parameters of the classifier; finally post-process the CAS to generate the final action proposals.

Due to the objective difference between classification and localization, the CAS obtained from the classifier has a serious incompleteness issue: it only detects trivial and sparse action regions. To solve this issue, SSE [49], CPMN [14], WO [13], and A2CL-PT [50] proposed the erasing strategy, which cascades multiple video classifiers, erases discriminative regions detected from the previous classifier, then inputs the remaining regions for the latter classifier, thus gradually localizing more complete action regions. CMCS [15] optimized multiple classifiers in parallel, which are used to detect different action regions. As CAS also suffers from false-positive activation or action-context confusion, BaSNet [9] and UM [51] proposed background modeling. DGAM [11] explored the separation of context and action with conditional variational auto-encoders. CO2-Net [52] tried to fuse the main and auxiliary modalities, and enhance modal features using consensus from global-local context information. In terms of post-processing, Autoloc [12], CleanNet [53], and KT-MGFN [54] introduced the outer-inner contrastive loss to replace simple threshold operations.

C. Adaptive Sampling Strategy

Adaptive sampling aims to force the model to focus more on some specific details, through scaling up the local area of image or video. It is widely used in fine-grained recognition, small object detection, and image retargeting. Concretely, for fine-grained recognition, SSampler [61] adopted the saliency maps to sample for data augmentation. S3N [62] used class response maps as guidelines for sampling, and achieved considerable improvements. For the image retargeting, NCV [63] and EBID [64] formulated this task as the finite element and energy minimization problems. For faster action recognition, SCSampler [65] selected a small subset of salient snippets to replace the whole video through a lightweight sampler.

The dynamic sampling strategy introduced to action detection in GridPool [66], is the closest to our method. However, our method differs from GridPool in the following two aspects. (1) Different motivations of the adaptive sampler: GridPool follows a “coarse-to-fine” idea and attempts to dynamically process the video with multiple temporal resolutions for better video representations, which essentially shares a similar idea to Slow-Fast [67]; while in this paper, we focus on promoting complementary input between two “mutual learning” branches in order to find more complete action regions. As a result, the adaptive sampler is designed to select the less discriminative action regions underestimated by the base branch as inputs for the supplementary branch. In essence, our method shares a similar idea to the erasing framework [13], [14], [49], [50]. (2) Different task settings and sampling principles: GridPool is under the fully-supervised setting where precise location annotations are provided, while our AMS targets on the weakly-supervised setting where only category labels are available. GridPool samples frames using a higher frame rate where the informativeness...
is high to locate precise action locations. We sample at a higher frame rate where the informativeness (CAS) is low to find more complete action regions.

III. ADAPTIVE MUTUAL SUPERVISION

In this section, we propose the adaptive mutual supervision (AMS) framework; see the framework pipeline in Fig. 2. We first formulate weakly-supervised temporal action localization; then present the adaptive sampler and mutual location supervision; and finally, we introduce the training and testing details.

A. Problem Formulation

Suppose that we are given $N$ untrimmed videos $\{y^{i}_{j}\}_{j=1}^{N}$ and their corresponding video-level category labels $\{Y^{i}_{j}\}_{j=1}^{N}$, where $y^{i}_{j}$ is a $C$-dimensional binary vector ($C$ is the total number of action categories), with $y^{i}_{j,k} = 1$ if the $i$-th video contains the $k$-th action category, and $y^{i}_{j,k} = 0$ otherwise. Note that each video may cover multiple action categories and multiple action instances. Weakly-supervised temporal action localization aims to predict the temporal locations of these action instances, in terms of a set of quadruples $\{(s, e, c, p)\}$, where $s$, $e$, $c$, $p$ represent the start time, the end time, the action category and the localization score of the action proposal, respectively.

Following [9], [10], [11], [15], to make sure that all videos have the same length, we sample $T$ consecutive snippets from each video, then calculate the original feature $F^{\text{orig}} \in \mathbb{R}^{T \times D}$ by a pre-trained feature extractor, where $D$ is the feature dimension.

B. Overall Framework

The proposed AMS framework consists of the base branch and the supplementary branch with identical backbones. Each backbone predicts the video category probability and respective CAS. The base branch is fed the original video feature to localize the most discriminative action regions from CAS. To strengthen collaboration and complementarity between the two branches so that the whole framework covers more complete actions, we encourage the supplementary branch to localize the less discriminative action regions using a novel adaptive sampler. The sampler adaptively updates the inputs for the supplementary branch using a sampling weight sequence negatively correlated with the CAS from the base branch, that is, under-sample the inputs from high-CAS regions, while over-sample the inputs from the low-CAS regions. As a result, the supplementary branch is encouraged to excavate the action regions underestimated by the base branch, making localization more complete. To further promote mutual enhancement and explicitly optimize the localization objective, we also construct mutual location supervision between the two branches. Each branch uses pseudo-labels generated from the CAS of the other branch as localization supervision. We alternately freeze one branch and optimize the other branch, such that the complementary action information of these two branches can be fused to make the location supervision more complete and precise. During multiple iterations, our AMS framework progressively refines the localization results.

In either the base or supplementary branch, we input video features into a backbone network $h(\cdot)$ to predict the category probability and CAS. The backbone network is implemented by a multi-layer perceptron. For the base branch, its backbone network can be formalized as follows:

$$M^\text{base}, S^\text{base} = h^\text{base}(F^{\text{orig}}, \phi^\text{base}),$$

where $M^\text{base} \in \mathbb{R}^{T \times C}$ refers to the CAS, which indicates the probability distribution of each video snippet belonging to all action categories. And $S^\text{base} \in \mathbb{R}^{C}$ means the predicted video category probability. $\phi^\text{base}$ is the trainable parameters of the backbone network. The input of the base branch is the original feature $F^{\text{orig}}$. While the input of the supplementary branch is...
generated by attaching an adaptive sampler \( S(\cdot, \cdot) \) on \( F_{\text{orig}} \), and the sampling weight is negatively correlated with \( M_{\text{base}} \):

\[
F_{\text{supp}} = S(F_{\text{orig}}, M_{\text{base}}),
\]

where \( F_{\text{supp}} \in \mathbb{R}^{T \times D} \). The design details of the sampler are introduced in Section III-C. Formally, the backbone network of the supplementary branch is formalized as:

\[
M_{\text{supp}}, \quad \tilde{y}_{\text{supp}} = h_{\text{supp}}(F_{\text{supp}}, \phi_{\text{supp}}),
\]

where the output \( C \times C \times C \), \( \phi_{\text{supp}} \) is the trainable parameters. We also perform temporal alignment between \( M_{\text{base}} \) and \( M_{\text{supp}} \), and then obtain \( M_{\text{supp}} \); see details in Section III-C3.

### C. Adaptive Sampler

To tackle the incompleteness issue of CAS, we here desire to strengthen the complementarity between the two branches. To achieve this, the novel adaptive sampler is designed to differentiate the inputs of two branches. Specifically, we input the original video features into the base branch, while dynamically selecting snippet features for the supplementary branch using the adaptive sampler. The whole sampling process is divided into two parts as demonstrated in Fig. 3.

1) Sampling Weight Sequence: Similar to previous studies [13], [14], [49], [50], the CAS of the base branch tends to focus on the most discriminative action regions. To make the supplementary branch purposefully localize the less discriminative action regions without repeated detection, we first need to completely aggregate all action information in CAS from the base branch, then generate the sampling weight sequence. Here, we calculate the class-agnostic CAS to achieve aggregation and prevent omissions. Specifically, given the CAS from the base branch \( M_{\text{base}} \in \mathbb{R}^{T \times C} \), we only keep the channels of ground truth categories, then perform the maximum operation on these channels along the temporal dimension. Formally, the aggregated CAS is denoted as \( m = \{m_t\} \in \mathbb{R}^T \).

According to the meaning of CAS [10], [12], \( m \) represents the action confidences of all video snippets along the temporal dimension. That is, the higher value of \( m_t \) indicates a higher confidence of existing an action at the \( t \)-th snippet. Considering the incompleteness issue of CAS, there might exist some missing actions (less discriminative actions) in the low-value regions of the aggregated CAS \( m \). Therefore, we aim to make the supplementary branch focus more on the low-value regions while less on the high-value regions. And the sampling weight sequence \( w \) is hence designed to be negatively correlated with the aggregated CAS \( m \):

\[
w = \{w_t\} = \max (m) - m + \eta \in \mathbb{R}^T,
\]

where \( \eta \) denotes a sampling adjustment value, and \( \max (\cdot) \) is the maximum operation. Here, each element in \( w \) represents the probability that the corresponding snippet will be selected by the sampling operation. For example, a lower value of the aggregated CAS \( m_t \) corresponds to a higher probability of \( w_t \), which indicates that the \( t \)-th video snippet with lower action confidence determined by the base branch is more likely to be sampled. Using such a sampling weight sequence, we naturally over-sample the snippets in the low-value regions of \( m_t \), while under-sample in the corresponding high-value regions.

2) Sampling Operation: In this section, we perform the sampling operation to generate the inputs for the supplementary branch. Since the original features only cover \( T \) snippets, to achieve more fine-grained sampling in the temporal dimension, we first up-sample \( F_{\text{orig}} \) using linear interpolation, then based on the sampling weight sequence, adaptively select \( T \) snippet features from the interpolated original features to form the inputs for the supplementary branch.

Formally, the interpolated original features are denoted as \( F_{\text{orig}} \in \mathbb{R}^{HT \times D} \), where \( H \) means the interpolation factor. To match the temporal length, we also calculate the interpolated sampling weight sequence, and denote it as \( \tilde{w} \in \mathbb{R}^{HT} \). Next, we introduce the sampling timestamps and sampling features in detail, as demonstrated in Fig. 3(b).

#### Sampling Timestamps. Following the inverse transformation theory [68], we utilize a cumulation-mapping manner to adaptively select \( T \) snippet features. To be specific, regarding the interpolated weight sequence \( \tilde{w} \) as the probability mass function, we first cumulate it along the temporal dimension to obtain the cumulative distribution function \( G(t) \):

\[
G(t) = \sum_{\tau=1}^{t} \tilde{w}_\tau d\tau.
\]

Intuitively, \( G(t) \) corresponds to the black curve in Fig. 3(b). Its role is to map the sampling probability of \( HT \) snippets uniformly distributed in the temporal dimension, in proportion to the interval length of the cumulation axis. A larger sampling probability \( \tilde{w}_t \) indicates a longer interval on the cumulation axis. As a result, sampling \( T \) timestamps from the time axis based on the interpolated weight sequence \( \tilde{w} \) is equivalent to uniformly sample \( T \) points on the cumulation axis.
To achieve the sampling goal, we uniformly sample $T$ points from the cumulation axis; map these points to the cumulative distribution function $G(t)$; map the points on $G(t)$ to the time axis; and finally, the corresponding $T$ timestamps on the time axis constitute a candidate sampling timestamp set $\mathcal{K}$.

**Sampling Features.** Afterward, based on the sampling timestamp set $\mathcal{K}$, we obtain the input features of the supplementary branch from the interpolated original features $\mathbf{F}^{\text{orig}}$: \[
\mathbf{F}^{\text{supp}} = \mathcal{K} \times \mathbf{F}^{\text{orig}} \in \mathbb{R}^{T \times D},
\]
where $\times$ denotes the index operation, that is, chronologically take out the features corresponding to $T$ sampling timestamps. Accordingly, $\mathbf{F}^{\text{supp}}$ is mainly composed of the features corresponding to the low-CAS value regions of the base branch. Intuitively, the efficacy of adaptive sampler can be interpreted as slowing down the video in the low-CAS regions of the base branch while speeding up the video in the high-CAS regions. The supplementary branch is thus encouraged to focus more on the less discriminative regions, completing detection results.

3) Temporal Alignment: Under-sampling or over-sampling scales up the background regions in the input features of the supplementary branch, and makes the action regions smaller. Accordingly, the CAS $\mathbf{M}^{\text{supp}} \in \mathbb{R}^{T \times C}$ generated by the supplementary branch highlights the less discriminative regions, but ignores the most discriminative action regions. In other words, compared with the uniform temporal distribution in the original input features, adaptive sampling results in the non-uniform temporal distributions for the inputs and CAS from the supplementary branch. To eliminate such side effects, we aim to make temporal alignment between the CASs from two branches. More specifically, we here re-sample the CAS from the supplementary branch for a uniform temporal distribution. For the regions with under-sampled input features, we over-sample their CAS through linearly interpolating these regions along the temporal dimension. While for these regions with over-sampled features, we under-sample on their CAS regions. Formally, denoting the temporal alignment module as $A$, the above process can be formulated as: \[
\mathbf{M}^{\text{supp}} = A(\mathbf{M}^{\text{supp}}) \in \mathbb{R}^{T \times C},
\]
where $\mathbf{M}^{\text{supp}}$ is the CAS output from the backbone network of the supplementary branch, and $\mathbf{M}^{\text{supp}}$ refers to the aligned CAS, which is employed to compensate for sampling. Note that, since the localization results are directly determined by CAS rather than inputs, the performance of the supplementary branch is now almost free from adaptive sampling.

4) Discussion: Note that, the proposed adaptive sampling could be regarded as the general version of the erasing operation [13], [14], [49], [50]. And when the sampling weights of high-CAS regions are fixed at 0%, and the others are 100%, our sampling strategy will be similar to erasing. But in most cases, adaptive sampling owns two main advantages. (1) Our sampling probability is soft, while the erasing probability is binary. This means that we always retain some action regions determined from the base branch, which act as action anchors for the supplementary branch to avoid paying attention to the background; while erasing removes all action regions found previously, easily misleading the model into the background. (2) Our sampling strategy slows down the less discriminative action regions and speeds up the most discriminative regions; while erasing only removes the most discriminative regions. This illustrates that the inputs produced by our strategy are more temporally fine-grained and more purposeful for less discriminative actions, which can bring better performance.

D. Mutual Location Supervision

To promote mutual enhancement and explicitly optimize the localization objective, in this section, we further construct mutual location supervision between the base and supplementary branches. Different from the self-training-based strategy [16], [17], [18], we force two branches to provide location pseudo-labels for each other, and progressively refine the localization results.

1) Location Pseudo-Labels: After making temporal alignment for the two branches, we generate location pseudo-labels from CAS. Given that CAS stands for the action confidence probability, i.e. high-CAS regions indicate high confidences of containing actions, to reduce uncertainty and label noise, we threshold CAS through a hyperparameter $\alpha$ to produce binary (action or background) pseudo-labels. As claimed in [9], [51], in addition to the incompleteness issue, the localization results of WTAL also suffer from the false-positive issue. Hence, we adopt binary pseudo-labels to tackle the above issues respectively, following the observation that even noisy pseudo-labels are effective [16], [17]. To be specific, we regard the regions with CAS values greater than $\alpha$ as the action pseudo-labels for the most and less discriminative regions, and they could act as positive action examples to alleviate the incompleteness issue. While the remaining regions are considered as the background pseudo-labels, i.e. negative action examples to tackle the false-positive issue. The location pseudo-labels are formulated as: \[
\overline{m}^k_t = \begin{cases} 
1, & \text{if } m^k_t > \alpha \text{ and } y^k = 1, \\
0, & \text{otherwise},
\end{cases}
\]
where $m^k_t$ is the CAS value from the $t$-th snippet and the $k$-th category channel, $\overline{m}^k_t$ is the corresponding pseudo-label.

2) Optimization Process: To promote mutual enhancement between two branches, we here force each branch to leverage location pseudo-labels from the other branch as the localization objective. And the optimization process is to alternately freeze one branch, while train the other branch. Specifically, in Phase zero, we train the base branch with only video-level category labels, to obtain the initial location pseudo-labels. In Phase one, we freeze the base branch, and produce the inputs for the supplementary branch through the adaptive sampler; then optimize the supplementary branch with action category labels and location pseudo-labels from the base branch; finally, update pseudo-labels using the CAS from the supplementary branch. In Phase two, we optimize the base branch with action category labels and the updated location pseudo-labels.

Functionally, mutual supervision can play a similar role as multi-view co-training [69]. The inputs of two branches, which
are sampled to be of somewhat complementary information, can be deemed as two distinctive views of the video. Mutual location supervision forces these two branches to learn from each other with regard to the localization results. Take the supplementary branch as an example, its localization results are determined by both input features and mutual supervision labels. On one hand, sampled features encourage it to localize the less discriminative action regions, and the temporal alignment module is adopted to compensate for adaptive sampling. On the other hand, the location pseudo-labels from the base branch, supervise the supplementary branch to highlight most discriminative regions. Hence, under the joint constraints of input features and mutual supervision, the localization results of the supplementary branch can be intuitively understood as a voting ensemble for the most and less discriminative regions.

Given that one single iteration may only bring limited gains in mining less discriminative action regions, we also utilize multiple iterations to optimize the proposed AMS framework. During each iteration, the adaptive sampler differentiates the inputs of these two branches, such that they could focus on different action regions purposefully. Then mutual location supervision encourages the two branches to remind each other of missing action regions, thus resulting in more complete localization. Note that, since we have made temporal alignment for the CAS from the sampled features, adaptive sampling does not affect mutual supervision or location pseudo-labels from two branches. During the progressive iterations, mutual location supervision and adaptive sampling work together and jointly contribute to more complete detection results.

3) Loss Function: For localization, following [2], [3], [31], we calculate the weighted cross-entropy loss between location pseudo-labels and the output CAS:

$$L_{local} = \frac{1}{C} \sum_{k=1}^{C} \left( \frac{1}{T^+} \sum_{t \in T^+} \mathcal{H}(m^k_t, \hat{m}^k_t) + \frac{1}{T^-} \sum_{t \in T^-} \mathcal{H}(m^k_t, \hat{m}^k_t) \right),$$

(9)

where $C$ refers to the number of action categories, $m^k_t \in \{0, 1\}$ is the output CAS of the $t$-th snippet and the $k$-th category channel, $\hat{m}^k_t \in \{0, 1\}$ means the location pseudo-label from the other branch, $\mathcal{H}$ is the regular cross-entropy loss, $T^+$ and $T^-$ are the number of positive and negative samples.

For video classification, we calculate the cross-entropy loss between the predicted action category probability $\hat{y} \in \mathbb{R}^C$ and the category label $y = [y^1, \ldots, y^C]^T$:

$$L_{class} = \frac{1}{C} \sum_{k=1}^{C} \mathcal{H}(\hat{y}^k, y^k),$$

(10)

where $\hat{y}$ is calculated by aggregating CAS with top-$k$ mean technique [9], [10], [76]. For better classification, we combine the classification loss $L_{class}$ with the Co-Activity Similarity loss in WTALC [10], as the basic loss $L_{basic}$ of each branch.

And finally, during the training of the AMS framework, we combine the basic loss and the localization loss to optimize the base branch or the supplementary branch:

$$L_{total} = L_{basic} + \lambda L_{local},$$

(11)

where $\lambda$ denotes a balance hyperparameter.

E. Inference

The AMS framework is separately optimized with RGB and flow features, and the final localization results are generated in a late-fusion fashion. During inference, for an input video, we fuse the two CASs from RGB and flow modalities, then average the CASs of the two branches as the final predicted CAS $M^{final} \in \mathbb{R}^{T \times C}$. The process is given by:

$$M^{final} = \frac{1}{2} \left( M^{base \_flow} + M^{supp \_flow} + \beta M^{base \_rgb} + \beta M^{supp \_rgb} \right),$$

(12)

where $\beta$ is a fusion hyperparameter. And here, the localization results from two branches are further fused together, resulting in a more complete overall performance. In this way, even if adaptive sampling has a slight side effect on the supplementary branch, it has little effect on the overall performance.

Then, we aggregate $M^{final}$ to derive the video-level category probability. For video classification, we only retain the classes whose probabilities are above the classification threshold $\theta_{cls}$. And for the remaining categories, we directly threshold $M^{final}$ by the localization threshold $\theta_{loc}$, then concatenate consecutive candidate snippets as the action proposals $\{(s_j, e_j, c_j, p_j)\}_{j=1}^{o}$, where $o$ is the number of proposals, $s_j, e_j, c_j, p_j$ respectively represent the start time, the end time, the action category, and the localization score of the $j$-th action proposal. The action category $c_j$ of the $j$-th proposal is set to the video category. And the localization score $p_j$ is calculated by the maximum value of $M^{final}$ within the proposal interval $[s_j, e_j]$.

IV. EXPERIMENTAL RESULTS

In this section, we conduct extensive experiments to validate the framework effectiveness and ablate each component. On two popular benchmarks, our framework outperforms existing state-of-the-art methods both quantitatively and qualitatively.

A. Dataset and Evaluation

THUMOS14 [19] covers total 413 untrimmed videos belonging to 20 action categories, and each video contains an average of 15 instances. The model is trained on 200 validation videos, and evaluated on 213 testing videos. The dataset is challenging and widely-used, for the reason that video lengths vary widely and the action instances occur frequently. ActivityNet1.2 [20] contains 9682 videos from 100 categories. The standard data split is 4619 training videos, 2383 validation videos, and 2480 testing videos. Most videos contain one action category, and the action regions account for more than half of the duration. We train on the training set, and evaluate the validation set.

Evaluation Metrics. Following the convention, we evaluate with the standard mean Average Precision (mAP) at different thresholds of temporal intersection over union (T-IoU). Note that a proposal is regarded as positive only if both the predicted action category is correct and T-IoU exceeds the set threshold. Each ground-truth instance can only match one proposal.
TABLE I

| Supervision | Training | Method | Modality | Feature | mAP| AVG (0.1-0.5)| AVG (0.3-0.7) |
|-------------|----------|--------|----------|---------|----|-------------|-------------|
|             |          |        |          | C3D     |    |             |             |
| Strong      | -        | SCNN   | [4]      | RGB+Flow| 47.7| 0.1 to 0.5  | 19.0 to 5.3  |
|             |          | TURN   | [5]      | RGB+Flow| 54.0| 0.3 to 0.5  | 25.6 to 7.7  |
|             |          | SNN    | [7]      | RGB+Flow| 66.0| 0.4 to 0.5  | 29.8 to 10.7 |
|             |          | A2Net  | [40]     | RGB+Flow| 61.1| 0.5 to 0.7  | 32.5 to 17.2 |
|             |          | TAL-Net| [1]      | RGB+Flow| 59.8| 0.6 to 0.7  | 28.0 to 20.8 |
|             |          | BUMR   | [31]     | RGB+Flow| 58.2| 0.7 to 0.8  | 25.0 to 19.6 |
| Weak        | Count-level | - | STARN | [53] | RGB+Flow | 68.8 | 0.1 to 0.5 | 23.0 to 6.2  |
|             |          | 3C-Net | [54]     | RGB+Flow| 60.0| 0.3 to 0.5  | 18.4 to 9.6  |
| Weak        | Point-level | - | SF-Net | [55] | RGB+Flow | 71.0 | 0.4 to 0.5  | 36.3 to 14.8 |
|             |          | BackTAL | [56]     | RGB+Flow| 64.9| 0.5 to 0.7  | 32.3 to 14.8 |
|             |          | DC-Net | [57]     | RGB+Flow| 72.6| 0.6 to 0.7  | 34.6 to 11.9 |
|             |          | LACP   | [58]     | RGB+Flow| 75.1| 0.7 to 0.8  | 39.3 to 20.8 |
| Weak Video-level | Single | - | STPN   | [8] | RGB+Flow | 45.3 | 0.3 to 0.5  | 16.2 to 9.1  |
|             |          | WO     | [13]     | RGB+Flow| 57.6| 0.4 to 0.5  | 23.9 to 7.1  |
|             |          | WTALC  | [10]     | RGB+Flow| 55.2| 0.5 to 0.7  | 31.1 to 11.9 |
|             |          | Cleanet| [51]     | RGB+Flow| 37.0| 0.6 to 0.8  | 19.6 to 5.4  |
|             |          | RbSNet | [9]      | RGB+Flow| 58.2| 0.7 to 0.8  | 31.2 to 11.9 |
|             |          | CMCS   | [15]     | RGB+Flow| 57.4| 0.8 to 0.9  | 21.3 to 7.6  |
|             |          | BM     | [72]     | RGB+Flow| 64.2| 0.9 to 1.0  | 27.5 to 13.6 |
|             |          | ASSG   | [73]     | RGB+Flow| 65.6| 0.9 to 1.0  | 25.6 to 12.9 |
|             |          | A2CL-PT| [48]     | RGB+Flow| 61.2| 1.0 to 1.2  | 28.8 to 11.4 |
|             |          | DGAM   | [11]     | RGB+Flow| 60.0| 1.1 to 1.3  | 29.8 to 11.4 |
|             |          | UM     | [49]     | RGB+Flow| 67.5| 1.2 to 1.3  | 33.3 to 11.4 |
| Multiple    | -        | RefineLoc | [16]     | RGB+Flow| 40.8| 0.2 to 0.4  | 23.1 to 5.3  |
|             |          | TSCN   | [17]     | RGB+Flow| 63.4| 0.3 to 0.5  | 28.7 to 11.4 |
|             |          | HM-ML  | [18]     | RGB+Flow| 59.1| 0.3 to 0.5  | 30.5 to 10.2 |
|             |          | AMS    | (Ours)   | RGB+Flow| 62.3| 0.4 to 0.6  | 33.1 to 13.0 |

Avg(0.1-0.5) and avg(0.3-0.7) are the average map from IOU 0.1 to 0.5 and from IOU 0.3 to 0.7, respectively. ‘Single’ denotes training with one single iteration. ‘Multiple’ denotes training with multiple iterations. All methods pre-extract video features from RGB and flow modalities for fair comparisons. C3D [70], TSN [71], and I3D [72] denote three different feature extractors. Our AMS method outperforms most state-of-the-art methods of video-level weakly-supervised settings, while performing comparably with several strongly-supervised methods.

B. Implementation Details

Feature Extraction. Following previous methods [9], [10], [11], [13], [14], [15], to reduce the computation overhead, we pre-extract the high-level features of original videos, then train the AMS framework from these features. For each video, we first split it into non-overlapping 16-frame snippets, then randomly sample $T$ consecutive snippets to deal with variable video lengths. $T$ is set to 1000 on THUMOS14, and 400 on ActivityNet1.2. We adopt the TV-L1 algorithm [77] to calculate optical flow from RGB data; utilize the two-stream I3D architecture [72] pre-trained on Kinetics-400 [72] to extract RGB and flow features; and finally, for each video snippet, we could obtain the 1024-dimensional feature from RGB or flow data.

Backbone Network. In the base branch and the supplementary branch, the backbone network maps the video features into CAS and video category probabilities. Structurally, it cascades a feature transformation module and a mapping module. The former consists of a fully connected layer, followed by ReLU activation and Dropout to fine-tune the pre-extracted video features. The latter consists of two parallel fully connected layers, followed by the softmax function respectively, to generate the CAS and video category probabilities.

Implementation. The proposed framework is implemented with Pytorch [78], using Adam optimizer [79] with a learning rate of $10^{-4}$. For fair comparisons, we freeze the pre-trained parameters from the feature extractor without fine-tuning. We optimize the framework for 20 epochs in Phase zero, and then alternately train these two branches every 5 epochs in Phase one and Phase two. All the hyperparameters are determined by grid search: the balance hyperparameter $\lambda = 1.0$, the fusion hyperparameter $\beta = 0.15$, the sampling adjustment value $\eta = 0.75$, the interpolation factor $H = 20$, and the classification threshold $\theta_{cls} = 0.25$. The threshold $\alpha$ for generating location pseudo-labels is set equal to the localization threshold $\theta_{loc}$, which is adaptively calculated by $0.7 \times \text{avg}(M)$.

C. Comparison With State-of-The-Art Methods

1) Localization Performance: We make full comparisons with existing methods from various levels of supervision settings, e.g. strong supervision and point-level weak supervision. As introduced in Section I, existing weakly-supervised methods could be divided into two categories: classification-based framework and self-training-based framework. The main difference is that the latter requires multiple iterations. We here abbreviate these two frameworks as ‘single’ and ‘multiple’ groups, and put our method into the ‘multiple’ group.

Table I reports comparisons on THUMOS14. We note that all the methods pre-extract video features from RGB and flow modalities, and most extractors are the I3D [72] pre-trained on Kinetics-400, thus the comparison is fair. Under the video-level weakly-supervised setting, our method obtains the state-of-the-art performance in terms of the average mAP (0.1 to 0.5), and performs better than most methods in terms of the average mAP (0.3 to 0.7). This well proves the effectiveness of our framework,
and reveals that our localization results are more precise and complete. Moreover, despite using a lower level of supervision (only video category labels), our method performs comparably with several count-level or point-level weakly-supervised methods [55], [56], [57] and even early strongly-supervised methods [4], [5], [7] at some IoU thresholds.

Table II also reports the results on ActivityNet1.2. Generally speaking, our method surpasses most previous methods at the same level of supervision, and achieves the best performance of the ‘multiple’ group. Note that, comparing to THUMOS14, ActivityNet only has one-tenth of action instances per video on average, and almost all the videos contain only one action category. In other words, this dataset has lower localization requirements, and almost all the videos contain only one action category.

In Table IV, comparing (B) to (A), we can find that there is no significant difference between the localization performance of the dual-branch model and the one-branch model. The only 0.2% average mAP improvement suggests that simply doubling model parameters cannot bring a significant increase in performance. Moreover, (A) and (B) perform worst among six setups. This is because relying only on the classification supervision easily misleads methods to focus on the most discriminative regions, resulting in sparse and incomplete localization.

2) Computational Cost: In addition to localization performance, Table III also compares the computational cost, i.e., the processing time and complexity analysis, with some state-of-the-art methods. Since none of previous methods provide the quantitative results on computational cost, we have to choose several popular methods that provide the official open-source implementations, for evaluation respectively.

In general, since all methods are trained from pre-extracted video features, the training times are all within one hour, and their testing times are all within two minutes on THUMOS14. Moreover, we find that approaches with stronger performance generally require longer processing time. Although our method is optimized using progressive iterations, its training time is still close to that of UM [51], which performs similarly to ours but without multiple iterations. In terms of the testing time, our method is almost identical to the compared methods.

In terms of the model efficiency, we here adopt parameters and GFLOPs to evaluate space complexity and time complexity, respectively. In our framework, the video features are pre-extracted and the only trainable modules are the backbone networks of the two branches. Therefore, comparing to UM [51] or BaSNet [9], our framework requires fewer parameters and less GFLOPs. All these indicate that the complexity of our AMS method is within the acceptable limits.

D. Ablation Studies

On THUMOS14, we analyze the effect of each component, i.e., the branch number, mutual location supervision, and adaptive sampler. Recently, the self-training-based strategy [16], [17], [18] is proposed to explicitly optimize localization objective. It adopts the pseudo-labels generated by CAS in the current step as location supervision for the next step, and relies on self-training to iteratively refine results. We also reproduce it under the same settings, for a detailed comparison.

Specifically, we experiment with the following six setups. The baseline is set as a vanilla classification-based method [8], [10], which is trained using only the basic loss $L_{\text{basic}}$, then thresholds the CAS for localization results. (A): The baseline with only one branch; (B): The baseline with two identical branches; (C): Provide the adaptive sampler to (B); (D): Add the self-training-based strategy on (B); (E): Add the mutual location supervision strategy on (B); (F): Add the adaptive sampler and mutual location supervision strategy on (B). For the above setups, as these two branches could generate quite different CASs, we average two CASs as the final output CAS, and report all the localization results in Table IV.

1) One Branch Vs. Dual Branch: In the AMS framework, the dual-branch setting causes the training parameters to be doubled. We here explore the efficacy of the branch number. Comparing (B) to (A), we can find that there is no significant difference between the localization performance of the dual-branch model and the one-branch model. The only 0.2% average mAP improvement suggests that simply doubling model parameters cannot bring a significant increase in performance. Moreover, (A) and (B) perform worst among six setups. This is because relying only on the classification supervision easily misleads methods to focus on the most discriminative regions, resulting in sparse and incomplete localization.

2) Self-Training-Based Strategy Vs. Mutual Supervision: To evaluate the efficacy of location pseudo-labels, we add mutual supervision or self-training-based strategy to (B), and perform multiple iterations. Comparing (D) or (E) to (B), we can see that multiple iterations with location supervision bring 3.4%
average mAP gains. This reflects that explicitly optimizing the localization objective can reconcile the contradiction between classification and localization in the WTAL task.

Moreover, comparing (E) to (D), mutual supervision alone only outperforms self-training-based strategy by 0.5% average mAP. We conjecture this is because, without adaptive sampler, the inputs to these two branches are the same, leading to very small differences in the two networks and their respective localization pseudo-labels. In this case, mutual location supervision is actually close to self-training, and the slight improvements mainly come from dual-branch ensemble learning [80]. While with adaptive sampler, mutual supervision (F) brings a gain about 4.3% average mAP over the self-training-based strategy (D). Here, the adaptive sampler ensures two branches to take somewhat complementary inputs, which could be deemed as two distinctive views of the video. Hence, mutual supervision plays a similar role as multi-view co-training [69], forcing the two branches to learn from each other with regarding to the localization results, so as to give full play to its great potential.

3) Effectiveness of the Adaptive Sampler: We also add the adaptive sampler on (B) and (E), to analyze the effectiveness of the designed sampling strategy. Comparing (C) to (B), the adaptive sampler differentiates the inputs of the two branches, contributing the gain of 3.1% average mAP. By purposefully selecting the less discriminative snippets for the supplementary branch, the sampler promotes it to explore the action regions underestimated by the base branch, thus completing the localization results of the whole framework.

Moreover, comparing (F) to (E), the adaptive sampler further boosts the effectiveness of mutual supervision, bringing a gain of 3.7% average mAP. By integrating adaptive sampler and mutual supervision, our AMS framework achieves the best performance, indicating that both components play essential roles and jointly contribute to more complete results.

E. Validation and Analysis Experiments

1) Effect of the Sampling Weight Strategies: As described in Section III-C1, we design the sampling weight sequence to be negatively correlated with the CAS from the base branch. To verify the effectiveness of this strategy, we here implement three experiments in Table V. (A): uniformly generate sample weights; (B): randomly generate sample weights; (C): adaptively generate sample weights by our proposed strategy. The results show that the adaptive sampling strategy significantly surpasses the other two, boosting the performance to more than 3.4% average mAP. As is evident, the adaptive sampling strategy effectively differentiates the two branches, prompting the supplementary branch to complement the detection results of the base branch. Moreover, (B) outperforms (A) by 0.8% average mAP. We speculate this is because random sampling acts as a role of data augmentation to scale up the amount of training data, thus bringing a certain gain.

2) Effect of Progressive Iterative Refinement: Fig. 4 quantifies the results of progressive refinement under mutual supervision for the base branch and supplementary branch. It can be seen that the performance of both branches increases during training, showing more complete coverage of the ground-truth localization. And the best performance is obtained in the third iteration, gaining 8.2% mAP compared to the initial iteration.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Moreover, the results of either branch inevitably have noise. For each branch, mutual supervision essentially encourages a voting ensemble of the most and less discriminative regions. During ensemble, the noise that only exists in one branch is ruled out, helping the overall framework to denoise slightly. In progressive iterations, with the growth of framework performance, the quality of pseudo-labels is gradually improved, i.e., the noise is continuously reduced. And better pseudo-labels in turn boost performance, thus creating a virtuous cycle.

Furthermore, we notice that the gain of one single iteration is decreasing until it becomes zero. For the initial iteration, the location pseudo-labels are all low-quality, or even none. While after three iterations, the quality of pseudo-labels tends to be high and stable, the performance reaches the main bound.

3) Effect of Class-Agnostic and Class-Specific Weights: As described in Section III-C1, by aggregating all action information from the CAS channels, we obtain class-agnostic weights for sampling, rather than using class-specific ones. Table VI summarizes the comparison. For the class-agnostic weights, we also compare the maximum and average aggregation operations, while for class-specific weights, we randomly select a ground-truth CAS channel. In general, there is no significant performance difference between average and maximum. Class-agnostic weights are slightly superior to class-specific weights. The main reason is that class-specific weights still enable the supplementary branch to repeatedly detect some action regions already found by the base branch. This increases the detection overlap of two branches, thus damaging the performance.

4) Effect of the Sampling Adjustment Value: We adopt \( \eta \) to adjust the sampling weight sequence in Eq. 4, and demonstrate the efficacy in Fig. 5. It shows a clear trend to peak at 0.75. And there are only slight differences in performance when \( \eta \) varies from 0.5 to 1. However, when we continue to increase \( \eta \) to 2, the performance drops a lot. This is because in this case, the sampling weights of different video snippets tend to be the same, causing adaptive sampling to degenerate into uniform sampling. Accordingly, the inputs of the two branches become almost identical, and our AMS framework degenerates into the naive mutual location supervision model.

5) Effect of the Optimization Losses: We use three losses to optimize the proposed framework, namely, video classification loss \( L_{\text{class}} \), co-activity similarity loss \( L_{\text{casl}} \) designed in [10], and location supervision loss \( L_{\text{local}} \). Functionally, the first two provide action classification supervision, while the last one provides action localization supervision. Table VII ablates the contribution of each loss. Consistent with the conclusion in W-TALC [10], \( L_{\text{casl}} \) can effectively improve the model performance. In addition, \( L_{\text{local}} \) explicitly optimizes the localization objective, thus bringing significant gains.

6) Correlation of the Input Features: In Fig. 6, we demonstrate the correlation of the input features from two branches, under various detection performances. When we feed the same input features into these two branches, the correlation is equal to 1, but the localization performance is far from satisfactory. When using the adaptive sampler to progressively differentiate the input features, the correlation decreases continuously, and the mAP performance first increases to the peak, then drops gradually. This is as expected, because the sampled features with the proper correlation, can encourage the supplementary branch to localize action regions missed by the base branch. However, when the feature correlation is too low, the sampled features will be mainly composed of background, resulting in low-quality pseudo-labels and inferior performance.
Fig. 7 Qualitative Results on THUMOS14. The first three plots are the input video, CAS, and localization results of the base branch. The middle three plots illustrate the input video, CAS, and localization results of the supplementary branch. The last three plots are the final CAS, localization results of the whole framework, and ground truth action intervals. The base branch, which is fed with the original videos of uniformly temporal distribution, focuses on the most discriminative actions. Using the adaptive sampler, we input videos of non-uniformly temporal distribution to the supplementary branch, thus forcing it to complement the less discriminative actions. Mutual location supervision makes our final results more complete and precise.

F. Qualitative Results

Fig. 7 qualitatively demonstrates our superiority. For clear understanding, we provide the inputs, CAS, and localization results of the base and supplementary branches in turn. In general, for videos covering sparse or dense action instances, our localization results are relatively complete and precise. The base branch, which is fed with videos of uniformly temporal distribution, focuses on the most discriminative action regions. Hence, its corresponding results are sparse and trivial. Relying on the adaptive sampler, we select the uncertain video snippets of the base branch as inputs for the supplementary branch. The videos with non-uniformly temporal distribution, force the supplementary branch to purposefully complement less discriminative actions underestimated by the base branch, but also causes several false-positive background predictions. Through mutual location supervision, the localization results of the two branches are mutually enhanced, bringing more complete and precise prediction results ultimately. These qualitative results again prove the effectiveness of our AMS framework.

V. Conclusion

This work proposes an adaptive mutual supervision framework with two branches, to solve the incompleteness issue in WTAL. The base branch leverages CAS to detect the most discriminative action regions, while the supplementary branch localizes the less discriminative regions with a novel adaptive sampler. The sampler dynamically updates the inputs of the supplementary branch with sampling weights negatively correlated with the CAS from the base branch, thus prompting the supplementary branch to detect action regions underestimated by the base branch. To promote mutual enhancement between the two branches, we construct mutual location supervision. Each branch adopts the location pseudo-labels generated from the other branch as localization supervision. Through alternately optimizing two branches during multiple iterations, our method progressively localizes more complete action regions. Extensive experiments demonstrated the effectiveness and outstanding performance of our AMS framework.

REFERENCES

[1] Y.-W. Chao et al., “Rethinking the faster R-CNN architecture for temporal action localization,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 1130–1139.
[2] T. Liu, X. Zhao, H. Su, C. Wang, and M. Yang, “BSN: Boundary sensitive network for temporal action proposal generation,” in Proc. Eur. Conf. Comput. Vis., Springer, 2018, pp. 3–19.
[3] T. Lin, X. Liu, X. Li, E. Ding, and S. Wen, “BMN: Boundary-matching network for temporal action proposal generation,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 3889–3898.
[4] Z. Shou, D. Wang, and S.-F. Chang, “Temporal action localization in untrimmed videos via multi-stage CNNs,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 1049–1058.
[5] J. Gao, Z. Yang, K. Chen, C. Sun, and R. Nevatia, “Turn tap: Temporal unit regression network for temporal action proposals,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 3628–3636.
[6] C. Lin et al., “Fast learning of temporal action proposal via dense boundary generator,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 11499–11506.
[7] Y. Zhao et al., “Temporal action detection with structured segment networks,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2914–2923.
[8] P. Nguyen, T. Liu, G. Prasad, and B. Han, “Weakly supervised action localization by sparse temporal pooling network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 6752–6761.
[9] P. Lee, Y. Oh, and H. Byun, “Background suppression network for weakly-supervised temporal action localization,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 11320–11327.
[10] S. Paul, S. Roy, and A. K. Roy-Chowdhury, “W-Talc: Weakly-Supervised Temporal Activity Localization and Classification,” in Proc. Eur. Conf. Comput. Vis., Springer, 2018, pp. 563–579.
[11] B. Shi, Q. Dai, Y. Mu, and J. Wang, “Weakly-supervised action localization by generative attention modeling,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 1009–1019.
[12] Z. Shou, H. Gao, L. Zhang, K. Miyazawa, and S.-F. Chang, “Autoloc: Weakly-Supervised Temporal Action Localization in Untrimmed Videos,” in Proc. Eur. Conf. Comput. Vis., Springer, 2018, pp. 154–171.
[31] R. Zeng et al., “Breaking winner-takes-all: Iterative-winners-out networks for weakly supervised temporal action localization,” *IEEE Trans. Image Process.*, vol. 28, no. 12, pp. 5797–5808, Dec. 2019.

[32] H. Su, X. Zhao, and T. Lin, “Hierarchical pyramid mining network for weakly supervised temporal action localization,” in *Proc. Asian Conf. Comput. Vis.*, Springer, 2018, pp. 558–574.

[33] D. Liu, T. Jiang, and Y. Wang, “Completeness modeling and context separation for weakly supervised temporal action localization,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1298–1307.

[34] A. Pardo, H. Alwassel, F. Caba, A. Thabet, and B. Ghanem, “Refinement: Iterative refinement for weakly-supervised action localization,” in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2021, pp. 3319–3328.

[35] Y. Zhai et al., “Two-stream consensus network for weakly-supervised temporal action localization,” in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 37–54.

[36] Z. Luo et al., “Weakly-supervised action localization with expectation-propagation moment-instance learning,” in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 729–745.

[37] Y.-G. Jiang et al., “THUMOS challenge: Action recognition with a large number of classes,” 2014. [Online]. Available: http://crcv.ucf.edu/THUMOS14/.

[38] F. Caba Heilbron, V. Escorcia, B. Ghamam, and J. Carlos Niebles, “Activitynet: A large-scale video benchmark for human activity understanding,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 961–970.

[39] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5734–5743.

[40] T. Lin, X. Zhao, and Z. Shou, “Single shot temporal action detection,” in *Proc. the ACM Int. Conf. Multimedia.*, 2017, pp. 988–996.

[41] H. Xu, A. Das, and K. Saenko, “R-C3D: Region convolutional 3D network for temporal activity detection,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5783–5792.

[42] F. Long et al., “Gaussian temporal awareness networks for action localization,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 344–353.

[43] H. Song et al., “Temporal action localization in untrimmed videos using action pattern trees,” *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 717–730, Mar. 2019.

[44] D. Guo, W. Li, and X. Fang, “Fully convolutional network for multi-scale temporal action proposals,” *IEEE Trans. Multimedia*, vol. 20, no. 12, pp. 3428–3438, Dec. 2018.

[45] C. Sun, H. Song, X. Wu, Y. Jia, and J. Luo, “Exploiting informative video segments for temporal action localization,” *IEEE Trans. Multimedia*, vol. 24, pp. 274–287, 2021.

[46] Y. Zhou, R. Wang, H. Li, and S.-Y. Kung, “Temporal action localization using short-term dependency,” *IEEE Trans. Multimedia*, vol. 23, pp. 4363–4375, 2020.

[47] H. Liu, S. Wang, W. Wang, and J. Cheng, “Multi-scale based context-aware net for action detection,” *IEEE Trans. Multimedia*, vol. 22, pp. 337–348, 2020.

[48] Z. Qing et al., “Temporal context aggregation network for temporal action proposal refinement,” in *Proc. the ACM Int. Conf. Multimedia.*, 2021, pp. 483–494.

[49] P. Zhao et al., “Bottom-up temporal action localization with mutual regularization,” in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 539–555.

[50] Y. Bai et al., “Boundary content graph neural network for temporal action proposal generation,” in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 131–137.

[51] M. Xu, C. Zhao, D. S. Rojas, A. Thabet, and B. Ghanem, “G-TAD: Subgraph localization for temporal action detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10156–10165.

[52] J. Gao et al., “Accurate temporal action proposal generation with relation-aware pyramid network,” in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 10810–10817.

[53] F. Long et al., “Coarse-to-fine localization of temporal action proposals,” *IEEE Trans. Multimedia*, vol. 22, no. 6, pp. 1577–1590, Jun. 2020.

[54] C. Ju, T. Han, K. Zheng, Y. Zhang, and X. Xie, “Promoting visual-language models for efficient video understanding,” in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 1–20.

[55] P. Zhao, L. Xie, Y. Zhang, Y. Wang, and Q. Tian, “Privileged knowledge distillation for online action detection,” *Pattern Recognit.*, vol. 129, 2022, Art. no. 108741.

[56] P. Zhao, L. Xie, Y. Zhang, and Q. Tian, “Actionness-guided transformer for anchor-free temporal action localization,” *IEEE Signal Process. Lett.*, vol. 29, pp. 194–198, 2021.
K. Kahatapitya and M. S. Ryooy, “Coarse-fine networks for temporal activity detection in videos,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 8385–8394.

C. Feichtenhofer, H. Fan, J. Malik, and K. He, “Slowfast networks for video recognition,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 6202–6211.

L. Devroye, “Sample-based non-uniform random variate generation,” in Proc. 18th Conf. Winter Simul., 1986, pp. 260–265.

F. Ma, D. Meng, X. Dong, and Y. Yang, “Self-paced multi-view co-training,” J. Mach. Learn. Res., vol. 21, no. 57, pp. 1–38, 2020.

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3D convolutional networks,” in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 4489–4497.

K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in Proc. Adv. Neural Inf. Process. Syst., 2014, vol. 27, pp. 1–9.

J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 6299–6308.

P. X. Nguyen, D. Ramanan, and C. C. Fowlkes, “Weakly-supervised action localization with background modeling,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 5502–5511.

C. Zhang et al., “Adversarial seed sequence growing for weakly-supervised temporal action localization,” in Proc. the ACM Int. Conf. Multimedia., 2019, pp. 738–746.

T. Yu et al., “Temporal structure mining for weakly supervised action detection,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 5522–5531.

P. Lee, J. Wang, Y. Lu, and H. Bynu, “Weakly-supervised temporal action localization by uncertainty modeling,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 1854–1862.

A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers, “An improved algorithm for Tv-L 1 optical flow,” in Statistical and Geometrical Approaches to Visual Motion Analysis. Berlin, Germany: Springer, 2009, pp. 23–45.

A. Paszke et al., “Automatic differentiation in pytorch,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1–4.

D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Representations, 2015, pp. 1–15.

T. G. Dietterich et al., “Ensemble learning,” The Handbook of Brain Theory and Neural Networks, vol. 2, Cambridge, MA, USA: MIT Press, 2002, pp. 110–125.

Chen Ju received the B.E. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2018. He has been working toward the Ph.D. degree with the Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai, China, since 2018. His research interests include vision-language-audio pre-training and applications, multi-modal video analysis and understanding, and music composition. He is also the reviewer of some prestigious international conferences.

Peisen Zhao received the B.E. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2016, and the Ph.D. degree from Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai, China, in 2022. He is currently a Researcher of artificial intelligence with Huawei Cloud and AI, Shenzhen, China. His research interests include video generation, action recognition, and understanding. He is also the reviewer of some prestigious international conferences and journals.

Siheng Chen received the B.E. degree from the Beijing Institute of Technology, China, and the M.S. and Ph.D. degrees from Carnegie Mellon University, Pittsburgh, PA, USA. He is currently an Associate Professor with Shanghai Jiao Tong University, Shanghai, China. He was an Autonomy Engineer with Uber Advanced Technologies Group, Pittsburgh, PA, USA, and Research Scientist with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA. His research interest include graph signal processing, graph neural networks, and autonomous driving. He was the recipient of the 2018 IEEE Signal Processing Society Young Author Best Paper Award, and also his paper was the recipient of the Best Student Paper Award at IEEE GlobalSIP 2018.

Ya Zhang is currently a Professor with Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai, China, and chief AI Scientist with the State Key Laboratory of UHD video and audio production and presentation. She has authored or coauthored more than 100 refereed papers in prestigious international conferences and journals. Her research interest mainly include machine learning with applications to multimedia and healthcare. Dr. Zhang was the recipient of several best paper awards of international journals and conferences, and directed one Outstanding Doctorate Dissertations awarded by Chinese Association for Artificial Intelligence.

Xiaoyun Zhang (Member, IEEE) received the B.S. and M.S. degrees in applied mathematics from Xi’an Jiaotong University, Xi’an, China, and the Ph.D. degree in pattern recognition from Shanghai Jiao Tong University, Shanghai, China. She is currently an Associate Professor with the CMIC, Shanghai Jiao Tong University. Her Ph.D. thesis was nominated as the National 100 Best Ph.D. Theses of China. Her research interests include computer vision, pattern recognition, image processing, video compression, and digital TV system.

Yanfeng Wang received the B.S. degree from PLA Information Engineering University, Beijing, China, and the M.S. and Ph.D. degrees in business management from Shanghai Jiao Tong University, Shanghai, China. He is currently the Vice Director of Cooperative Medianet Innovation Center and also the Vice Dean of the School of Electrical and Information Engineering, Shanghai Jiao Tong University. His research interest mainly include multimedia information retrieval, computer vision, pattern recognition, image processing, video compression, research laboratories, and commercial applications of information technology and technology transfer.

Qi Tian (Fellow, IEEE) is currently the Chief Scientist of artificial intelligence with Huawei Cloud & AI, Shenzhen, China, and is also a Full Professor with the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX, USA. He has authored or coauthored more than 700 refereed journal and conference papers, including many Best Papers and Best Paper Candidates. His research interests include multimedia information retrieval, computer vision, pattern recognition. He was recipient of the 2017 UTSA President’s Distinguished Award for 2016 UTSA Innovation Award, 2014 Research Achievement Awards from the College of Science, UTSA, 2010 Google Faculty Award, and the 2010 ACM Service Award. He is an Associate Editor for many journals and on the Editorial Board of the Journal of Multimedia, and the Journal of Machine Vision and Applications.