

Localizing the Common Action Among a Few Videos

Pengwan Yang^{1,2*}, Vincent Tao Hu^{2*}, Pascal Mettes², and Cees G. M. Snoek²

¹ Peking University, China

² University of Amsterdam, the Netherlands

Abstract. This paper strives to localize the temporal extent of an action in a long untrimmed video. Where existing work leverages many examples with their start, their ending, and/or the class of the action during training time, we propose few-shot common action localization. The start and end of an action in a long untrimmed video is determined based on just a hand-full of trimmed video examples containing the same action, without knowing their common class label. To address this task, we introduce a new 3D convolutional network architecture able to align representations from the support videos with the relevant query video segments. The network contains: (i) a mutual enhancement module to simultaneously complement the representation of the few trimmed support videos and the untrimmed query video; (ii) a progressive alignment module that iteratively fuses the support videos into the query branch; and (iii) a pairwise matching module to weigh the importance of different support videos. Evaluation of few-shot common action localization in untrimmed videos containing a single or multiple action instances demonstrates the effectiveness and general applicability of our proposal. **Code:** <https://github.com/PengWan-Yang/commonLocalization>

Keywords: common action localization, few-shot learning

1 Introduction

The goal of this paper is to localize the temporal extent of an action in a long untrimmed video. This challenging problem [8, 32] has witnessed considerable progress thanks to deep learning solutions, *e.g.* [12, 26, 37], fueled by the availability of large-scale video datasets containing the start, the end, and the class of the action [3, 6, 17]. Recently, weakly-supervised alternatives have appeared, *e.g.* [1, 18, 24, 25, 31, 34, 43, 47]. They avoid the need for hard to obtain start and end time annotations, but still require hundreds of videos labeled with their action class. In this paper, we also aim for a weakly-supervised setup, but we avoid the need for any action class labels. We propose few-shot *common* action localization, which determines the start and end of an action in a long untrimmed video based on just a hand-full of trimmed videos containing the same action, without knowing their common class label.

* Equal contribution. {yangpengwan2016, taohu620}@gmail.com

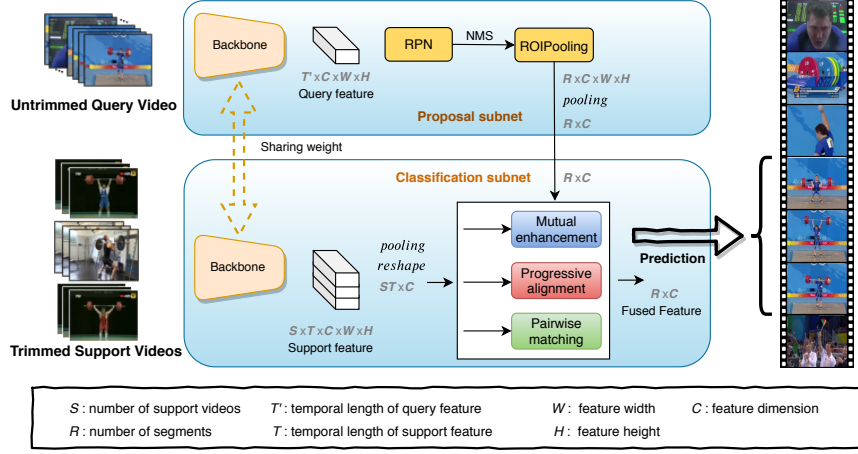


Fig. 1: **Common action localization** in an untrimmed query video from three trimmed support videos during inference. The action is localized in the query video based on the common action in the support videos.

We are inspired by recent works on few-shot object detection [7, 16, 35, 36]. Dong *et al.* [7] start from a few labeled boxes per object and a large pool of unlabeled images. Pseudo-labels for the unlabeled images are utilized to iteratively refine the object detection result. Both Shaban *et al.* [36] and Hu *et al.* [16] further relax the labeling constraint by only requiring a few examples to contain a common object, without the strict need to know their class name. Hu *et al.* [16] introduce two modules to reweigh the influence of each example and to leverage spatial similarity between support and query images. We also require that our few examples contain a common class and we adopt a reweighting module. Different from Hu *et al.*, we have no module to focus on masking objects spatially in images. Instead, we introduce three alternative modules optimized for localizing actions temporally in long untrimmed videos, as illustrated in Figure 1.

We make three contributions in this work. First, we consider common action localization from the few-shot perspective. All we require is that the few trimmed video examples share a common action, which may be obtained from social tags, hash tags or off-the-shelf action classifiers. Second, we propose a network architecture for few-shot common action localization, along with three modules able to align representations from the support videos with the relevant query video segments. The mutual enhancement module strengthens the representations of the query and support representations simultaneously by building upon non-local blocks [44]. The progressive alignment module iteratively integrates the support branch into the query branch. Lastly, the pairwise matching module learns to weigh the importance of different support videos. As a third contribution, we reorganize the videos in ActivityNet1.3 [3] and Thumos14 [17] to allow for experimental evaluation of few-shot common action localization in long untrimmed videos containing a single or multiple action instances.

2 Related work

Action localization from many examples. Standard action localization is concerned with finding the start and end times of actions in videos from many training videos with labeled temporal boundaries [2, 9, 37]. A common approach is to employ sliding windows to generate segments and subsequently classify them with action classifiers [5, 11, 37, 42, 46]. Due to the computational cost of sliding windows, several approaches model the temporal evolution of actions and predict an action label at each time step [9, 27, 38, 48]. The R-C3D action localization pipeline [45] encodes the frames with fully-convolutional 3D filters, generates action proposals, then classifies and refines them. In this paper, we adopt the proposal subnet of R-C3D to obtain class-agnostic action proposals. In weakly-supervised localization, the models are learned from training videos without temporal annotations. They only rely on the global action class labels [30, 31, 43]. Different from both standard and weakly-supervised action localization, our *common* action localization focuses on finding the common action in a long untrimmed query video given a few (or just one) trimmed support videos without knowing the common action class label, making our task class-agnostic. Furthermore, the videos used to train our approach contain actions that are not seen during testing.

Action localization from few examples. Yang *et al.* [46] pioneered few-shot *labeled* action localization, where a few (or at least one) positive labeled and several negative labeled videos steer the localization via an end-to-end meta-learning strategy. It relies on sliding windows to swipe over the untrimmed query video to generate fixed boundary proposals. Rather than relying on a few positive and many negative action class labels, our approach does not require any predefined positive nor negative action labels, all we require is that the few support videos have the same action in *common*. Moreover, we propose a network architecture with three modules that predicts proposals of arbitrary length from commonality only.

Action localization from one example. Closest to our work is video re-localization by Feng *et al.* [10], which introduces localization in an untrimmed query video from a *single* unlabeled support video. They propose a bilinear matching module with gating functions for the localization. Compared to video relocalization, we consider a more general and realistic setting, where more than one support video can be used. Furthermore, we consider untrimmed videos of longer temporal extent and we consider action localization from a single frame. To enable action localization under these challenging settings, we introduce modules that learn to enhance and align the query video with one or more support videos, while furthermore learning to weigh individual support videos. We find that our proposed common action localization formulation obtains better results, both in existing and in our new settings.

Action localization from no examples. Localization has also been investigated from a zero-shot perspective by linking actions to relevant objects [19, 20, 29]. Soomro *et al.* [39] tackle action localization in an unsupervised setting, where no annotations are provided overall. While zero-shot and unsupervised

action localization show promise, current approaches are not competitive with (weakly-)supervised alternatives, hence we focus on the few-shot setting.

3 Method

3.1 Problem description

For the task of few-shot common action localization, we are given a set of trimmed support videos S_c^N , where N is small, and an untrimmed query video Q_c . Both the support and query videos contain activity class c , although its label is not provided. The goal is to learn a function $f(S_c^N, Q_c)$ that outputs the temporal segments for activity class c in the query video. The function $f(\cdot, \cdot)$ is parametrized by a deep network consisting of a support and query branch. During training, we have access to a set of support-query tuples $T = \{(S_l^N, Q_l)\}_{l \in \mathcal{L}_{\text{train}}}$. During both validation and testing, we are only given a few trimmed support videos with corresponding long untrimmed query video. The data is divided such that $\mathcal{L}_{\text{train}} \cup \mathcal{L}_{\text{val}} \cup \mathcal{L}_{\text{test}} = \emptyset$.

3.2 Architecture

We propose an end-to-end network to solve the few-shot common action localization problem. A single query video and a few support videos are fed into the backbone, a C3D network [40], to obtain video representations. The weights of the backbone network are shared between the support and query videos. For the query video, a proposal subnet predicts temporal segments of variable length containing potential activities [45]. Let $\mathcal{F}_Q \in \mathbb{R}^{R \times C}$ denote the feature representation of the query video for R temporal proposal segments, each of dimensionality C . Let $\mathcal{F}_S \in \mathbb{R}^{ST \times C}$ denote the representations of the S support videos, where we split each support video into T fixed temporal parts. The main goal of the network is to align the support representations with the relevant query segment representation:

$$\mathcal{F} = \phi(\mathcal{F}_Q, \mathcal{F}_S). \quad (1)$$

In Equation 1, $\mathcal{F} \in \mathbb{R}^{R \times C}$ denotes the temporal segment representations after alignment with the support representations through ϕ . In our common localization network, representations \mathcal{F} are fed to fully-connected layers that perform a binary classification to obtain the likelihood that each proposal segment matches with the support actions, which is followed by a temporal regression to refine the activity start- and end-times for all segments.

In our network, we consider the following: *i*) the representations of the support videos need to be aligned with the representations of the activity in the query video, *ii*) not all support videos are equally informative, and *iii*) common action localization is a support-conditioned localization task, where the activity-tytness of different query segments should be guided by the support videos. We propose three modules, namely mutual enhancement module, progressive alignment module, and pairwise matching module to deal with these considerations.

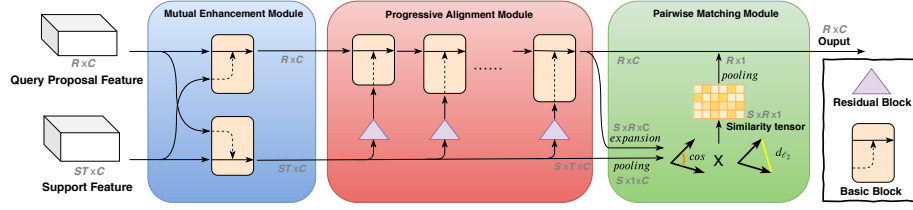


Fig. 2: **Modules for aligning representations** from the support videos with the relevant query video segments. The mutual enhancement module augments the support and query representations simultaneously through message passing. Then, the progressive alignment module fuses the support into the query branch through recursive use of the basic block. Finally, the pairwise matching module reweighs the fused features according to the similarity between the enhanced query segments and the enhanced support videos.

Mutual enhancement module. Building on the recent success of the transformer structure [41] and the non-local block [44], which are forms of self-attention, we propose a module which can simultaneously enhance the representations of the support and query videos from each other. The basic block for this module is given as:

$$m(I_1, I_2) = c_1(\text{soft}(c_2(I_1) \times c_3(I_2^T)) \times c_4(I_2)) + I_1, \quad (2)$$

where c_1, c_2, c_3, c_4 are fully-connected layers, soft denotes the softmax activation, and \times denotes matrix multiplication. I_1 and I_2 denote the two inputs. A detailed overview and illustration of the basic block is provided in the supplementary materials. Based on the basic block, we design a mutual enhancement module to learn mutually-enforced representations, both for query proposals and support videos, as shown in Figure 2. The mutual enhancement module has two streams $m_{s \rightarrow q}$, $m_{q \rightarrow s}$ that are responsible for enhancing query proposals and support videos respectively. The inputs to the mutual enhancement module, \mathcal{F}_Q and \mathcal{F}_S , will be enhanced by each other:

$$m_{s \rightarrow q} = m(\mathcal{F}_Q, \mathcal{F}_S), \quad (3)$$

$$m_{q \rightarrow s} = m(\mathcal{F}_S, \mathcal{F}_Q). \quad (4)$$

Progressive alignment module. We also propose a progressive alignment module to achieve a better fusion of representations from query proposals and support videos. The idea behind this module is to reuse the basic block from the mutual enhancement module to integrate the support branch into the query branch. Inspired by the successful application of residual learning [14, 15], we employ a residual block to make the progressive alignment effective:

$$r(I) = c_1(\text{relu}(c_2(I))) + I, \quad (5)$$

where c_1 , c_2 are fully-connected layers, $relu$ denotes the ReLU activation. A detailed overview and illustration of the residual block is provided in the supplementary materials. We first take query proposal representations from the first module $m_{s \rightarrow q}$ as 0-depth outcome \mathcal{P}_0 . On top, we adopt our basic block m to integrate this outcome with $m_{q \rightarrow s}$ which has been recalibrated by our residual block r . We perform this operation multiple times in a recursive manner, *i.e.*:

$$\mathcal{P}_0 = m_{s \rightarrow q}, \quad (6)$$

$$\mathcal{P}_k = m(\mathcal{P}_{k-1}, r(m_{q \rightarrow s})), \quad k = 1, 2, \dots, n. \quad (7)$$

Where we set $n = 3$ in practice. The advantage of a progressive design is that it strengthens the integration of the support branch into the query branch as we increase the number of basic block iterations. By using the same efficient basic blocks as our first module, the computational overhead is small. An illustration of the progressive alignment module is shown in Figure 2.

Pairwise matching module. In common action localization, a small number of support videos is used. Intuitively, not every support video is equally informative for the query segments. In addition, different query segments should not be treated equally either. To address these intuitions, we add a means to weigh the influence between each support video and each query segment, by introducing a pairwise matching module.

The input for the matching module are all segments of the query video and all support videos. The pair-wise matching is a mapping $\text{PMM} : (\mathbb{R}^{R \times C}, \mathbb{R}^{S \times T \times C}) \mapsto \mathbb{R}^{S \times R \times 1}$. To align the two components, we first perform an expansion operation e on the query segments, denoted as $e(\mathcal{P}_n) \in \mathbb{R}^{S \times R \times C}$. Then a pooling p is applied over the support videos along the temporal dimension, denoted as $p(m_{q \rightarrow s}) \in \mathbb{R}^{S \times 1 \times C}$. Afterwards, we perform an auto broadcasting operation b on $p(m_{q \rightarrow s})$, which can broadcast the dimension of $p(m_{q \rightarrow s})$ from $\mathbb{R}^{S \times 1 \times C}$ to $\mathbb{R}^{S \times R \times C}$ to align with the dimension of $e(\mathcal{P}_n)$. For query segments \mathcal{P}_n and for support videos $m_{q \rightarrow s}$, their match is given by the cosine similarity (\cos) and ℓ_2 Euclidean distance (d_{ℓ_2}) along the segment axis:

$$M = \cos(\mathcal{P}_n, m_{q \rightarrow s}) = \frac{\langle e(\mathcal{P}_n), b(p(m_{q \rightarrow s})) \rangle}{\|e(\mathcal{P}_n)\| \cdot \|b(p(m_{q \rightarrow s}))\|}, \quad (8)$$

$$N = d_{\ell_2}(\mathcal{P}_n, m_{q \rightarrow s}) = \|e(\mathcal{P}_n) - b(p(m_{q \rightarrow s}))\|. \quad (9)$$

We combine both distance measures:

$$\mathcal{W} = \text{PMM}(\mathcal{P}_n, m_{q \rightarrow s}) = M \odot \sigma(-N), \quad (10)$$

where σ denotes the Sigmoid operation. Tensor $\mathcal{W} \in \mathbb{R}^{S \times R \times 1}$ can be interpreted as a weight tensor to achieve attention over the R and S dimensions. $\mathcal{W}[i, j]$ is a scalar depicting the similarity between the j -th query segment representation and the i -th support representation. For the j -th query segment representation,

$\mathcal{W}[:, j] \in \mathbb{R}^{S \times 1}$ corresponds to the weight for different support videos, while for the i -th support representation, $\mathcal{W}[i, :] \in \mathbb{R}^{R \times 1}$ resembles the weight for different query segments. In the end, we enforce the pairwise matching weight \mathcal{W} :

$$\phi(\mathcal{F}_Q, \mathcal{F}_S) = \mathcal{P}_n \odot \text{AP}(\mathcal{W}), \quad (11)$$

where AP denotes an average pooling operation along the support dimension, in other words, $\text{AP} : \mathbb{R}^{S \times R \times 1} \mapsto \mathbb{R}^{R \times 1}$.

3.3 Optimization

To optimize our network on the training set, we employ both a classification loss and a temporal regression loss. Different than *e.g.*, R-C3D [45], our classification task is specifically dependent on the few support videos. Accordingly, the loss function is given as:

$$L = \frac{1}{N_{cls}} \sum_i L_{cls}(a_i, a_i^*) + \frac{1}{N_{reg}} \sum_i a_i^* L_{reg}(t_i, t_i^*), \quad (12)$$

where N_{cls} and N_{reg} stand for batch size and the number of proposal segments, while i denotes the proposal segment index in a batch, a_i is the predicted probability of the proposal segment, a_i^* is the ground truth label, and t_i represents predicted relative offset to proposals. In the context of this work, the ground truth label is class-agnostic and hence binary (foreground/background), indicating the presence of an action or not. Lastly, t_i^* represents the coordinate transformation of ground truth segments to proposals.

The above loss function is applied on two parts: the support-agnostic part and the support-conditioned part. All losses for the two parts are optimized jointly. In the support-agnostic part, the foreground/background classification loss L_{cls} predicts whether the proposal contains an activity, or not, and the regression loss L_{reg} optimizes the relative displacement between proposals and ground truths. For the support-conditioned part, the loss L_{cls} predicts whether the proposal has the same common action as the one among the few support videos. The regression loss L_{reg} optimizes the relative displacement between activities and ground truths. We note explicitly that this is done for the training set only.

During inference, the proposal subnet generates proposals for the query video. The proposals are refined by Non-Maximum Suppression (NMS) with a threshold of 0.7. Then the selected proposals are fused with the support videos through the mutual enhancement, progressive alignment, and pairwise matching modules. The obtained representation is fed to the classification subnet to again perform binary classification and the boundaries of the predicted proposals are further refined by the regression layer. Finally, we conduct NMS based on the confidence scores of the refined proposals to remove redundant ones, and the threshold in NMS is set a little bit smaller than the overlap threshold θ in evaluation ($\theta = 0.1$ in this paper).

Table 1: **Overview of the common (multi-)instance datasets.** The common instance datasets contain a single target action per video, while the common multi-instance datasets contain more frames and more actions per video, adding to the challenge of few-shot common action localization.

| | Common instance | | Common multi-instance | |
|----------------------------|-----------------|--------|-----------------------|--------|
| | ActivityNet | Thumos | ActivityNet | Thumos |
| Video statistics | | | | |
| number of instances | 1 | 1 | 1.6 | 14.3 |
| number of frames | 266.9 | 284.6 | 444.5 | 5764.2 |
| length (sec) | 89.0 | 11.4 | 148.2 | 230.6 |
| number of train videos | 10035 | 3580 | 6747 | 1665 |
| number of val+test videos | 2483 | 775 | 1545 | 323 |
| Class statistics | | | | |
| number of train actions | 160 | 16 | 160 | 16 |
| number of val+test actions | 40 | 4 | 40 | 4 |

Optimizing for long videos. The longer the untrimmed query video, the larger the need for common localization, as manual searching for the activity becomes problematic. In our setup, the length of the input video is set to 768 frames to fit the GPU memory. When the query video is longer than 768 frames, we employ multi-scale segment generation [37]. We apply temporal sliding windows of 256, 512, and 768 frames with 75% overlap. Consequently, we generate a set of candidates $\Phi = \{(s_h, \psi_h, \psi'_h)\}_{h=1}^H$ as input for the proposal subnet, where H is the total number of sliding windows, and ψ_h and ψ'_h are the starting time and ending time of the h -th segment s_h . All refined proposals of all candidate segments together go through the NMS to remove redundant proposals.

4 Experimental setup

4.1 Datasets

Existing video datasets are usually created for classification [17, 22], temporal localization [3], captioning [4], or summarization [13]. To evaluate few-shot common action localization, we have revised two existing datasets, namely ActivityNet1.3 [3] and Thumos14 [17]. Both datasets come with temporal annotations suitable for our evaluation. We consider both common instance and common multi-instance, where the latter deals with query videos containing multiple instances of the same action.

Common instance. For the revision of ActivityNet1.3, we follow the organization of Feng *et al.* [10]. We divide videos that contain multiple actions into independent videos, with every newly generated video consisting of just one action and background. Next we discard videos longer than 768 frames. We split the remaining videos into three subsets, divided by action classes. We randomly select 80% of the classes for training, 10% of the classes for validation, and the remaining 10% of the classes for testing. Besides ActivityNet, we also revise the Thumos dataset using the same protocol.

| | | | ActivityNet | | Thumos | |
|-----|-----|-----|-------------|-------------|-------------|-------------|
| MEM | PAM | PMM | one-shot | five-shot | one-shot | five-shot |
| | | | 42.4 | 42.5 | 37.5 | 38.4 |
| | ✓ | | 49.7 | 52.0 | 42.3 | 44.5 |
| | ✓ | ✓ | 51.3 | 53.6 | 44.8 | 46.0 |
| ✓ | ✓ | | 52.5 | 55.3 | 47.6 | 49.6 |
| ✓ | ✓ | ✓ | 53.1 | 56.5 | 48.7 | 51.9 |

Table 2: **Module evaluation** on ActivityNet and Thumos in the common instance setting. All three modules have a positive mAP effect on the localization performance with only a slight increase in parameters.

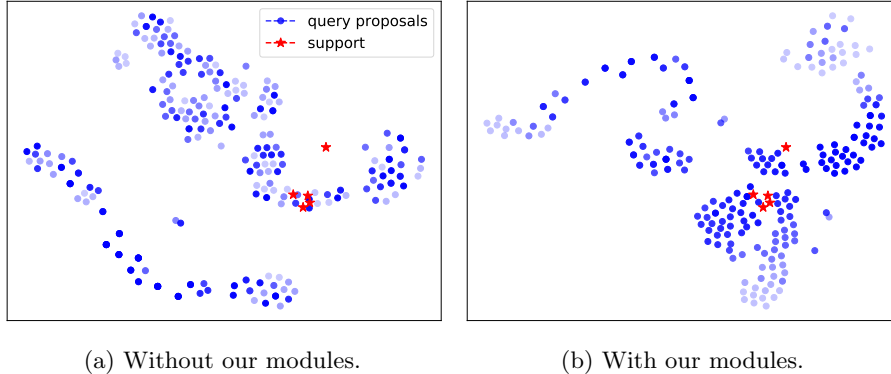


Fig. 3: **Module evaluation** by t-SNE visualization of support and query representations. Colors of query proposals indicate their overlap with the ground truth action, the darker the better. Without our modules (left), both relevant and irrelevant query proposals are near the support videos. Afterwards (right), only relevant proposals remain close to the support videos, highlighting the effectiveness of our modules for localizing common actions among a few videos.

Common multi-instance. Query videos in real applications are usually unconstrained and contain multiple action segments. Therefore, we also split the original videos of ActivityNet1.3 and Thumos14 into three subsets according to their action classes without any other video preprocessing. As a result, we obtain long query videos with multiple action instances. The support videos are still trimmed action videos.

During training, the support videos and query video are randomly paired, while the pairs are fixed for validation and testing. The differences between the common instance and common multi-instance video datasets are highlighted in Table 1.

4.2 Experimental details

We use PyTorch [33] for implementation. Our network is trained with Adam [23] with a learning rate of $1e-5$ on one Nvidia GTX 1080TI. We use 40k training iterations and learning rate is decayed to $1e-6$ after 25k iterations. To be consistent with the training process of our baselines [10, 49], we use the same C3D

backbone [40]. The backbone is pre-trained on Sports-1M [21] and is fine-tuned with a class-agnostic proposal loss on the training videos for each dataset. The batch size is set to 1. The proposal score threshold is set as 0.7. The proposal number after NMS is 128 in training and 300 in validation and testing.

4.3 Evaluation

Following [10,37], we measure the localization performance using (mean) Average Precision. A prediction is correct when it has the correct foreground/background prediction and has a ground truth overlap larger than the overlap threshold. The overlap is set to 0.5 unless specified otherwise.

5 Experimental results

5.1 Ablation study

Module evaluation. We evaluate the effect of the mutual enhancement module (MEM), the progressive alignment module (PAM), and the pairwise matching module (PMM) for our task on the common instance datasets. We report results using one and five support videos in Table 2. To validate the effectiveness of our modules, we compare to our baseline system without any modules. Here the support representations are averaged and added to the query representations. We observe that the progressive alignment module increases over the baseline considerably, showing its efficacy. Adding the pairwise matching on top of the progressive alignment or using the mutual enhancement before the progressive alignment further benefits few-shot common action localization. Combining all three modules works best.

To get insight into the workings of our modules for common action localization, we have analysed the feature distribution before and after the use of our modules. In Figure 3, we show the t-SNE embedding [28] before and after we align the five support videos with the 300 proposals in one query video. We observe that after the use of our modules, the proposals with high overlap are closer to the support videos, indicating our ability to properly distill the correct action locations using only a few support videos. Irrelevant proposals are pushed away from the support videos, which results in a more relevant selection of action locations.

Table 3: **Influence of noisy support videos** on common-instance ActivityNet for the five-shot setting. The result shows that our approach is robust to the inclusion of noisy support videos, whether they come from the same or different classes.

| | |
|---|------|
| No noise | 56.5 |
| 1 noisy support video | 53.5 |
| 2 noisy support videos of different class | 51.9 |
| 2 noisy support videos of same class | 50.6 |

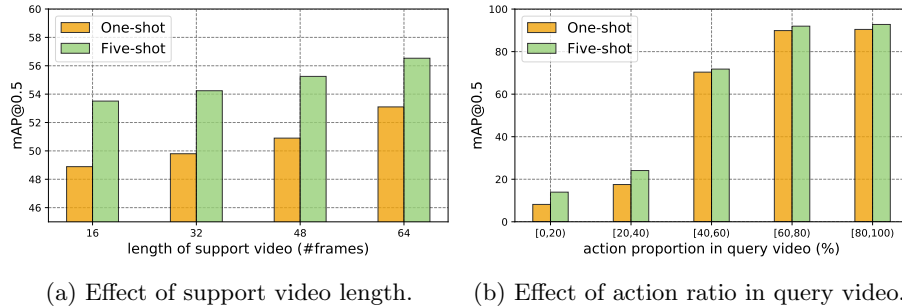


Fig. 4: **Ablation studies** on the length of the support videos and the action proportion in the query video. Both studies are on common-instance ActivityNet. Left: The longer the support videos, the better we perform, as we can distill more knowledge from the limited provided supervision. Right: High scores can be obtained when the common action is dominant, localization of short actions in long videos remains challenging.

Few-shot evaluation. Our common action localization is optimized to work with multiple examples as support. To show this capability, we have measured the effect of gradually increasing the number of support videos, we found that the mAP gradually increases as we enlarge the number of support videos from one to six on common-instance ActivityNet. We obtain an mAP of 53.1 (one shot), 53.8 (two shots), 54.9 (three shots), 55.4 (four shots), 56.5 (five shots), 56.8 (six shots). The results show that our approach obtains high accuracy with only a few support videos. Using more than one support video is beneficial for common action localization in our approach, showing that we indeed learn from using more than one support video. Results stagnate when using more than six examples.

Effect of support video length. We ablate the effect of the length of the support videos on the localization performance in Figure 4a. We sample 16, 32, 48 and 64 frames for each support video respectively. We find that the result gradually increases with longer support videos, which indicates that temporal information in the support videos is beneficial to our modules for common action localization.

Influence of action proportion in query video. Figure 4b shows that for query videos with a dominant action, we can obtain high scores. An open challenge remains localizing very short actions in very long videos.

Influence of noisy support videos. To test the robustness of our approach, we have investigated the effect of including noisy support videos in the five-shot setting. The results are shown in Table 3. When one out of five support videos contains the wrong action, the performance drops only 3% from 56.5 to 53.5. The performance drop remains marginal when replacing two of the five support videos with noisy videos. When two noisy support videos are from the same class, the drop is larger, which is to be expected, as this creates a stronger

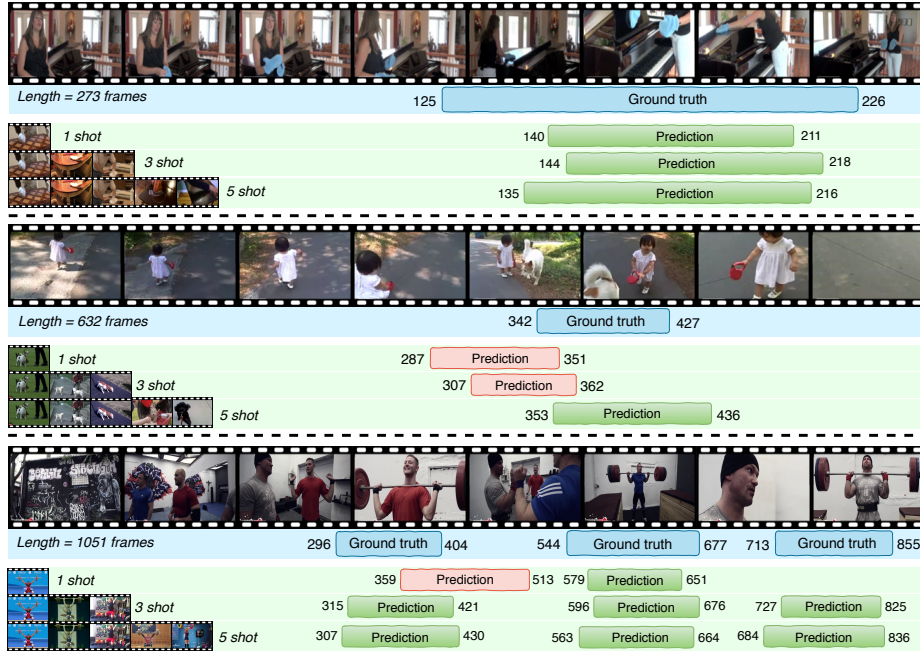


Fig. 5: **Qualitative result** of predictions by our approach under 1-shot, 3-shot and 5-shot settings. Correct predictions with an overlap larger than 0.5 are marked in **green**, and incorrect predictions are marked in **red**. The length and start-end boundary of segment are indicated in frame numbers.

bias towards a distractor class. Overall, we find that our approach is robust to noise for common action localization.

Qualitative results. To visualize the result of our method, we show three cases in Figure 5. For the first example, we can find the common action location from one support video. Adding more support videos provides further context, resulting in a better fit. For the second one, our method can recover the correct prediction only when five support videos are used. As shown in the third case, our method can also handle the multi-instance scenario. We show a query video with three instances. With only one support video, we miss one instance and have low overlap with another. When more support videos are added, we can recover both misses.

5.2 Comparisons with others

To evaluate the effectiveness of our proposed approach for common action localization, we perform three comparative evaluations.

One-shot comparison. For the one-shot evaluation, we compare to the one-shot video re-localization of Feng *et al.* [10] and to Hu *et al.* [16], which focuses on few-shot common object detection. We evaluate on the same setting as Feng

| | Overlap threshold | | | | | |
|------------------------------|-------------------|-------------|-------------|-------------|------------|-------------|
| | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 0.5:0.9 |
| Common instance | | | | | | |
| Hu <i>et al.</i> [16] * | 41.0 | 33.0 | 27.1 | 15.9 | 6.8 | 24.8 |
| Feng <i>et al.</i> [10] | 43.5 | 35.1 | 27.3 | 16.2 | 6.5 | 25.7 |
| <i>This paper</i> | 53.1 | 40.9 | 29.8 | 18.2 | 8.4 | 29.5 |
| Common multi-instance | | | | | | |
| Hu <i>et al.</i> [16] * | 29.6 | 23.2 | 12.7 | 7.4 | 3.1 | 15.2 |
| Feng <i>et al.</i> [10] * | 31.4 | 25.5 | 16.1 | 8.9 | 3.2 | 17.0 |
| <i>This paper</i> | 42.1 | 36.0 | 18.5 | 11.1 | 7.0 | 22.9 |

Table 4: **One-shot comparison on common instance ActivityNet.** Results marked with * obtained with author provided code. In both settings, our approach is preferred across all overlaps, highlighting its effectiveness.

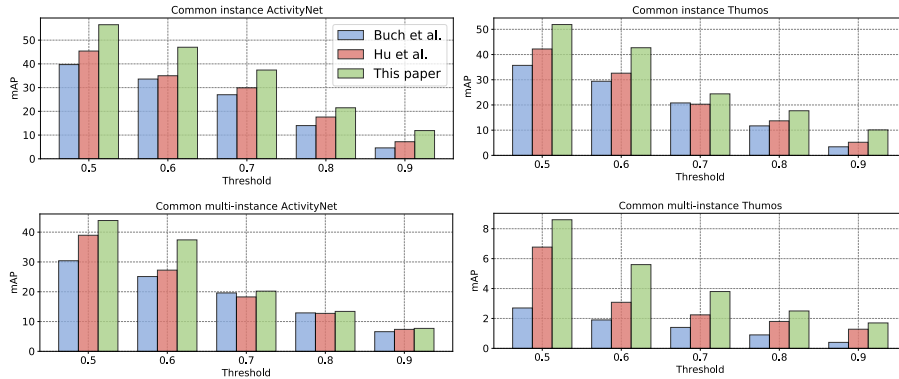


Fig. 6: **Five-shot comparison.** We evaluate our method as well as modified versions of Hu *et al.* [16] and Buch *et al.* [2] on all common instance and multi-instance datasets, we obtain favourable results. Detailed numerical results are provided in supplementary file to facilitate the comparison for the follow-up works. Best viewed in color.

et al. [10], namely the revised ActivityNet dataset using the one-shot setting (common instance). Note that we both use the C3D base network. To evaluate the image-based approach of Hu *et al.* [16], we use their proposed similarity module on the temporal video proposals, rather than spatial proposals based on author provided code [16]. The results in Table 4 show that across all overlap thresholds, our approach is preferred. At an overlap threshold of 0.5, we obtain an mAP of 53.1 compared to 41.0 for [16] and 43.5 for [10]. It is of interest to note that without our three modules, we obtain only 42.4 (Table 2). This demonstrates that a different training setup or a different model architecture by itself does not benefit common action localization. We attribute our improvement to the better alignment between the support and query representations as a result of our three modules. Next to a comparison on the common instance dataset, we also perform the same experiment on the longer multi-instance ActivityNet variant. In this more challenging setting, our approach again outperforms the baselines. We note that we are not restricted to the one-shot setting, where the baseline by Feng *et al.* [10] is.

Table 5: **Localization from images** on the common instance datasets. Our method generalizes beyond videos as support input and outperforms Zhang *et al.* [49]

| | ActivityNet | | Thumos | |
|---------------------|-------------|-------------|-------------|-------------|
| | one-shot | five-shot | one-shot | five-shot |
| Zhang <i>et al.</i> | 45.2 | 48.5 | 36.9 | 38.9 |
| <i>This paper</i> | 49.2 | 52.8 | 43.0 | 45.6 |

Five-shot comparison. Second, we evaluate the performance of our approach on all datasets in the five-shot setting. We compare to a modified version of SST by Buch *et al.* [2]. We add a fusion layer on top of the original GRU networks in SST to incorporate the support feature, and then choose the proposal with the largest confidence score. SST is used as baseline, because the approach of Feng *et al.* [10] cannot handle more than one support video. We also include another comparison to Hu *et al.* [16]. This time also using their feature reweighting module. The results are shown in Figure 6. We observe that our method performs favorably compared to the two baselines on all datasets, reaffirming the effectiveness of our method. Also note that even when our support videos are noisy (Table 3), we are still better than the baselines without any noise based on Buch *et al.* [2] and Hu *et al.* [16] (39.7 and 45.4 for a threshold of 0.5 on common instance ActivityNet). The large amount of distractor actions in the long videos of common multi-instance Thumos results in lower overall scores, indicating that common action localization is far from a solved problem.

Localization from images. Next to using videos, we can also perform common action localization using images as support. This provides a challenging setting, since any temporal information is lost. We perform localization from support images by inflating the images to create static support videos. We perform a common action localization on common instance ActivityNet and Thumos. We compare to the recent approach of Zhang *et al.* [49], which focuses on video retrieval from images. Results in Table 5 show we obtain favourable results on both datasets, even though our approach is not designed for this setting.

6 Conclusion

In this paper we consider action localization in a query video given a few trimmed support videos that contain a common action, without specifying the label of the action. To tackle this challenging problem, we introduce a new network architecture along with three modules optimized for temporal alignment. The first module focuses on enhancing the representations of the query and support representation simultaneously. The second module progressively integrates the representations of the support branch into the query branch, to distill the common action in the query video. The third module weighs the different support videos to deal with non-informative support examples. Experiments on reorganizations of ActivityNet and Thumos dataset, both with settings containing a single and multiple action instances per video, show that our approach can robustly localize the action which is common amongst support videos in both standard and long untrimmed query videos.

References

- [1] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *ECCV*, 2014.
- [2] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. Sst: Single-stream temporal action proposals. In *CVPR*, 2017.
- [3] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.
- [4] David L Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *ACL*, 2011.
- [5] Xiyang Dai, Bharat Singh, Guyue Zhang, Larry S Davis, and Yan Qiu Chen. Temporal context network for activity localization in videos. In *ICCV*, 2017.
- [6] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018.
- [7] Xuanyi Dong, Liang Zheng, Fan Ma, Yi Yang, and Deyu Meng. Few-example object detection with model communication. *PAMI*, 2018.
- [8] Olivier Duchenne, Ivan Laptev, Josef Sivic, Francis Bach, and Jean Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009.
- [9] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In *ECCV*, 2016.
- [10] Yang Feng, Lin Ma, Wei Liu, Tong Zhang, and Jiebo Luo. Video re-localization. In *ECCV*, 2018.
- [11] Jiyang Gao, Kan Chen, and Ram Nevatia. Ctap: Complementary temporal action proposal generation. In *ECCV*, 2018.
- [12] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In *ICCV*, 2017.
- [13] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *ECCV*, 2014.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.
- [16] Tao Hu, Pascal Mettes, Jia-Hong Huang, and Cees GM Snoek. Silco: Show a few images, localize the common object. In *ICCV*, 2019.
- [17] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos “in the wild”. *CVIU*, 2017.
- [18] Mihir Jain, Amir Ghodrati, and Cees G. M. Snoek. ActionBytes: Learning from trimmed videos to localize actions. In *CVPR*, 2020.
- [19] Mihir Jain, Jan C van Gemert, Thomas Mensink, and Cees G M Snoek. Objects2action: Classifying and localizing actions without any video example. In *ICCV*, 2015.
- [20] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Joint learning of object and action detectors. In *ICCV*, 2017.
- [21] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

- [22] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv*, 2017.
- [23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.
- [24] Hilde Kuehne, Alexander Richard, and Juergen Gall. A hybrid rnn-hmm approach for weakly supervised temporal action segmentation. *arXiv*, 2019.
- [25] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *ICCV*, 2017.
- [26] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. BSN: Boundary sensitive network for temporal action proposal generation. In *ECCV*, 2018.
- [27] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 2016.
- [28] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.
- [29] Pascal Mettes and Cees GM Snoek. Spatial-aware object embeddings for zero-shot localization and classification of actions. In *ICCV*, 2017.
- [30] Phuc Nguyen, Ting Liu, Gautam Prasad, and Bohyung Han. Weakly supervised action localization by sparse temporal pooling network. In *CVPR*, 2018.
- [31] Phuc Xuan Nguyen, Deva Ramanan, and Charles C. Fowlkes. Weakly-supervised action localization with background modeling. In *ICCV*, 2019.
- [32] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013.
- [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. In *NeurIPS*, 2017.
- [34] Sujoy Paul, Sourya Roy, and Amit K Roy-Chowdhury. W-talc: Weakly-supervised temporal activity localization and classification. In *ECCV*, 2018.
- [35] Johann Sawatzky, Martin Garbade, and Juergen Gall. Ex paucis plura: Learning affordance segmentation from very few examples. In *GCPR*, 2018.
- [36] Amirreza Shaban, Amir Rahimi, Stephen Gould, Byron Boots, and Richard Hartley. Learning to find common objects across image collections. In *ICCV*, 2019.
- [37] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, 2016.
- [38] Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *CVPR*, 2016.
- [39] Khurram Soomro and Mubarak Shah. Unsupervised action discovery and localization in videos. In *ICCV*, 2017.
- [40] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.
- [41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- [42] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition and detection by combining motion and appearance features. *THUMOS14 Action Recognition Challenge*, 2014.
- [43] Limin Wang, Yuanjun Xiong, Dahua Lin, and Luc Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *CVPR*, 2017.

- [44] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.
- [45] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017.
- [46] Hongtao Yang, Xuming He, and Fatih Porikli. One-shot action localization by learning sequence matching network. In *CVPR*, 2018.
- [47] Jiong Yang and Junsong Yuan. Common action discovery and localization in unconstrained videos. In *ICCV*, 2017.
- [48] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.
- [49] Zhu Zhang, Zhou Zhao, Zhijie Lin, Jingkuan Song, and Deng Cai. Localizing unseen activities in video via image query. In *IJCAI*, 2019.