A Simple and Efficient Pipeline to Build an End-to-End Spatial-Temporal Action Detector

Lin Sui\textsuperscript{2*}, Chen-Lin Zhang\textsuperscript{1†}, Lixin Gu\textsuperscript{3}, Feng Han\textsuperscript{3}

\textsuperscript{1} 4Paradigm Inc., Beijing, China
\textsuperscript{2} State Key Laboratory for Novel Software Technology, Nanjing University, China
\textsuperscript{3} DataElem Inc., Beijing, China

\{sulin0432, zclnjucs\}@gmail.com \{gulixin, hanfeng\}@dataelem.com

Abstract

Spatial-temporal action detection is a vital part of video understanding. Current spatial-temporal action detection methods mostly use an object detector to obtain person candidates and classify these person candidates into different action categories. So-called two-stage methods are heavy and hard to apply in real-world applications. Some existing methods build one-stage pipelines, but a large performance drop exists with the vanilla one-stage pipeline and extra classification modules are needed to achieve comparable performance. In this paper, we explore a simple and effective pipeline to build a strong one-stage spatial-temporal action detector. The pipeline is composed by two parts: one is a simple end-to-end spatial-temporal action detector. The proposed end-to-end detector has minor architecture changes to current proposal-based detectors and does not add extra action classification modules. The other part is a novel labeling strategy to utilize unlabeled frames in sparse annotated data. We named our model as SE-STAD. The proposed SE-STAD achieves around 2% mAP boost and around 80% FLOPs reduction. Our code will be released at \url{https://github.com/4paradigm-CV/SE-STAD}.

1. Introduction

Spatial-temporal action detection (STAD), which aims to classify multiple persons' actions in videos, is a vital part of video understanding. The computer vision community has drawn much attention in the field of STAD [36, 7, 42].

In previous methods, STAD is often divided into two sub-tasks: actor localization and action classification. Previous methods mostly utilize a pre-trained object detector [33, 45] and finetune it on the target dataset to obtain person candidates. Then, proposals are fed into the action classifier network to obtain the final action prediction. However, those two-stage methods are heavy and often need extra data (such as MS-COCO [24]). They need separate models and heavy computational resources. This prevents the current methods from real-world applications. A recent work [5] has shown that there is a dilemma between actor localization and action classification. Actor localization only needs a single image while action detection needs the whole input sequence. Thus, [5] proposes an end-to-end method WOO, which uses a unified backbone to perform actor localization and action detection. However, they still have a significant performance drop with the vanilla structure and need to introduce an extra attention module into the classification head to enhance the performance.

In this paper, we propose a new method named Simple and Effective Spatial-Temporal Action Detection, in short for SE-STAD. The general pipeline of SE-STAD is in Fig. 1. SE-STAD is composed by two parts. One is a strong one-stage spatial-temporal action

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Comparison between previous two-stage methods and our SE-STAD. (a). Previous two-stage STAD methods use a heavy offline person detector, which also relies on additional data, to perform actor localization and they only use annotations of keyframes to train the action classifier. (b). Our SE-STAD trains an end-to-end spatial-temporal action detector in which actor localization part only occupies a small part of the computation. We also propose temporal label assignment (TLA) to utilize unlabeled frames in large-scale sparsely annotated datasets like AVA [13].}
\end{figure}
detector architecture. Focusing on the localization ability of detector, the proposed architecture has minor architect modifications to existing two-stage methods. Thus, our methods can be applied to many existing STAD methods with comparable performance and less computational burden. With minor added components and effective training strategies, we empower the ability to conduct actor localization and action classification simultaneously without losing accuracy. Compared to existing works, our work is light-weighted and jointly optimized, which avoids the separate learning dilemma. Besides, we are the first to explore the strategy about building an end-to-end STAD from the perspective of localization ability. SE-STAD can also get benefits from other methods, such as adopting attention-based classification heads [5, 29].

The second part is a new paradigm to utilize every possible information in sparsely annotated spatial-temporal datasets. Sparse annotation is an efficient and effective way to build a large-scale STAD dataset, only the keyframes will be annotated (e.g. 1fps in AVA [13]). A huge amount of frames do not have annotations. Hence, we propose to utilize these unlabeled data to provide more clear temporal action boundaries and help the detector learn fine-grained information. Considering the distinctiveness of unlabeled data in sparse-annotated STAD datasets, we propose a novelty pseudo labeling strategy: temporal label assignment (TLA) to generate pseudo labels. With the help of TLA, end-to-end spatial-temporal action detectors successfully enjoy performance gains from the neglected unlabeled data of sparse-annotated datasets.

Our contributions are listed as follows:

- We propose a simple and effective pipeline to build end-to-end spatial-temporal action detection methods. The proposed pipeline can be applied to many existing spatial-temporal action detection methods.

- We build a simple architecture for end-to-end action detection with effective training methods, which avoids extra offline person detector and achieves comparable performance with two-stage methods.

- A novel semi-supervised learning strategy with a pseudo temporal labeling strategy is proposed to utilize every possible information in sparsely annotated data. With the proposed pipeline, we achieve a 2.2%/1.5% mAP boost and around 80%/20% FLOPs reduction compared to both proposal-based methods and one stage methods with extra proposals.

2. Related Works

In this section, we will introduce works related to our SE-STAD, including spatial-temporal action detection, object detection and semi-supervised learning.

Spatial-Temporal Action Detection Spatial-temporal action detection (STAD) aims to detect the action of different persons in the input video clips. Thus, STAD models needs to be aware of spatial and temporal information. After large-scale datasets are annotated and introduced [13, 19], researchers have paid much attention to STAD.

Most existing works often follow a traditional Fast-RCNN [10] pipeline with pre-extracted proposals to perform STAD [13, 20, 42, 29, 6]. A previous work [43] shows the original R-CNN [11] pipeline works better for spatial-temporal action detection. However, those works are heavy and inefficient.

Besides those two-stage methods, researchers also proposed some single-stage methods for action detection. Some works [13, 8] also employ the Faster-RCNN [33] pipeline but with low performance. Early works including YOWO [16], ACRN [36] and Point3D [28] combine pre-trained 2D and 3D backbones to build pseudo end-to-end detectors. Recently, WOO [5] proposes a single-stage, unified network for end-to-end action detection. WOO first utilizes Sparse-RCNN [37] along with the key frame to generate action candidates. Then action candidates will be fed into the classifier to obtain final results. With the vanilla structure, WOO has a major performance gap with the proposal-based methods. Thus, WOO utilizes an extra attention module to boost performance. In contrast, SE-STAD has a simple modification to the current proposal-based architecture. We only add a simple object detector and utilize better training strategies, and we achieve better results than WOO and proposal-based methods.

Apart from the detection structure side, Many researchers also propose new modules to enhance the performance including feature banks modules [42, 29], attention modules [29, 5] and graph-based methods [9, 36, 48]. However, we want to build a simple and strong model for end-to-end spatial-temporal action detection. Thus, we do not add any extra modules to our SE-STAD.

Object Detection Actor localization needs to detect the location of persons in the input image. Thus, object detection is needed. Object detection has been a popular area in the computer vision community. Early works often use a two-stage pipeline with pre-defined anchors [11, 10, 33]. One-stage methods, especially anchor-free detectors are proposed to reduce the computational burden for object detection [23, 32, 25, 39, 50]. Anchor-free detectors are easy to use in real-world applications. More recent methods want to train object detectors in an end-to-end manner [3, 37].

In this paper, we adopt the one-stage anchor-free detector FCOS [39], as the person detector in SE-STAD. FCOS is a simple and effective choice for the person detector.

Semi-Supervised Learning Semi-supervised learning (SSL) aims to achieve better performance with the help of additional unlabeled data. In brief, recent semi-supervised learning follows two main ways: introducing consistency regularization [31, 2, 44] or performing pseudo-labeling [18, 38, 40]. Some other works, such as [35] also...
3. Methods

In this section, we will give a detailed description of our SE-STAD.

3.1. Notations

We will first define the notations used in this paper.

Given a spatial-temporal action detection dataset \( \mathcal{D} \), which is composed of a total number of \( m \) videos: \( \mathcal{D} = \{V_1, \ldots, V_m\} \). For simplicity, we suppose all videos in \( \mathcal{D} \) have same height \( h \), width \( w \) and number of frames \( n \). Thus, \( V_i \in \mathbb{R}^{n \times h \times w \times 3} \). Spatial temporal action detection needs to detect the action category of the person in the specific input frame. For a frame \( F_j \) in \( V_i \), we need to detect a tuple \((x_1, y_1, x_2, y_2, \text{cls})\) for each person in \( F_j \). \((x_1, y_1, x_2, y_2)\) is the spatial location of the person, and \( \text{cls} \in [0, 1]^C \) is the action category where \( C \) is the pre-defined set of action classes. In widely used sparse-annotated datasets such as AVA [13], ground-truth annotations are annotated at one frame per second. For such a dataset \( \mathcal{D} \), we denote the labeled part as \( \mathcal{D}^l = \{V^l_1, \ldots, V^l_m\} \) with annotations \( Y^l = \{A^l_1, \ldots, A^l_m\} \) and the unlabeled part as \( \mathcal{D}^u = \{V^u_1, \ldots, V^u_m\} \).

3.2. Motivation

As shown in Sec. 2, previous STAD methods often follow a two-stage pipeline and utilize two networks: First conducting person detection with an offline object detector, then detected area of interests (RoIs) will be fed into a traditional Fast-RCNN style network to obtain the final action predictions. The two-stage networks are not efficient. Besides, they always need extra data (such as MS-COCO [24]) to train the additional person detector. WOO [5] uses a unified backbone to perform actor localization and action classification simultaneously. However, their unified models result in a large performance drop and WOO [5] proposes an extra embedding interaction head to boost the performance.

In contrast to those extra modules, we want to build a unified, end-to-end and simple method for spatial-temporal action detection. Thus, we want to make minimal modifications to current proposal-based methods, to perform spatial-temporal action detection effectively and efficiently. We name our proposed model as Simple and Effective Spatial Temporal Action Detection (SE-STAD).

3.3. SE-STAD

Our proposed SE-STAD is composed of three parts: feature extraction part, actor localization part and action classification part. We unified the three parts into a single network. We will introduce these three parts step by step.

3.3.1 Feature Extraction

In this part, we directly use existing action classification backbones, i.e., SlowFast [7] for feature extraction. Moreover, SE-STAD can utilize any modern backbones to boost performance, including the recent Transformer-based models, i.e., Video-Swin [27], ViViT [1] and MViT [6].

3.3.2 Actor Localization Part

We need to perform actor localization in our SE-STAD. Previously, separate pre-trained object detectors are adopted on actor localization, the most commonly used are Faster-RCNN [33] with a ResNeXt-101 [45] backbone. However, the separate object detector has an extra heavy computational burden, which is inefficient. A recent work WOO [5] proposes to integrate an existing object detection head, i.e., Sparse R-CNN [37] into the current action classification backbone. In this paper, we follow the suggestions in WOO [5], performing actor localization with the spatial feature of keyframes. Regarding the low input resolution and efficiency issue, we choose a popular one-stage anchor-free object detector FCOS [39]. Thus, the loss for actor localization is:

\[
\mathcal{L}_{al} = \mathcal{L}_{cls}(c_i, b_i) + \mathcal{L}_{iou}(c_i, b_i) + \mathcal{L}_{centerness}(c_i, b_i)
\]

where \( c_i \) indicates the video clip, \( b_i \) means ground-truth bounding boxes of the keyframe of \( c_i \). \( \mathcal{L}_{cls} \), \( \mathcal{L}_{iou} \) and \( \mathcal{L}_{centerness} \) are the Focal loss [23] for binary classification (existing of actor), GIoU [34] loss for bounding box regression and centerness prediction, respectively. Compared to Sparse-RCNN [37], FCOS has dense output proposals (before post-processing), and later we show that dense outputs (without post-processing) matters in STAD performance. In Sec. 4, We ablate different heads for actor localization including anchor-based heads and anchor-free heads to verify the effectiveness of different heads, and the training strategy. Our models use a simple actor localization head and perform better than vanilla WOO [5], even comparable to or better than WOO with extra attention modules.
3.3.3 Action Classification Part

For action classification, since we want to build an end-to-end spatial-temporal action detection network with minimum efforts, we follow the common practice to build an action classification head: we use the traditional ROIAlign [23] layer with temporal pooling to get the feature for each actor proposal, then a simple linear layer is attached to get the final action predictions, we use the binary cross entropy loss to train the action classification head. Thus, the loss of the action classification head becomes:

\[ L_{ac} = L_{bce}(c_i, b_i, l_i) \]  

(2)

where \( l_i \) denotes the classification annotation and \( L_{bce} \) is binary cross entropy loss. In order to balance the scale of localization loss \( L_{al} \) and classification loss \( L_{ac} \), we introduce loss weight \( \lambda_{cls} \) for action classification which is set to 10 as default. Experiments in Sec. 4 show the model is robust to different \( \lambda_{cls} \).

The overall structure of our SE-STAD is quite simple. We only add a simple FCOS head to perform actor localization. However, a simple model achieves comparable results to proposal-based methods [7] and the recently unified backbone method [5] which introduces additional attention modules.

Besides the model structure, we propose a novel semi-supervised training strategy to better utilize every possible piece of information in the training video. With the semi-supervised training stage, our model can achieve better results than originally trained models.

3.4. Semi-Supervised Action Detection for SE-STAD

It’s well known that sparse annotation is an efficient and effective way to build large-scale spatial-temporal action detection datasets. However, as large parts of data are unlabeled, sparse annotations fail to provide clear temporal action boundaries. This phenomenon has been shown by previous literature [22]. Utilizing the unlabeled part is a natural way to help the detector to learn fine-grained information. Hence, we propose a new semi-supervised training method for sparsely annotated datasets in spatial-temporal action detection.

When performing semi-supervised training in SE-STAD, we adopt the online updating paradigm. Besides, in order to avoid the inductive bias introduced in semi-supervised training, following the widely used Mean Teacher [38] pipeline, we also build a teacher-student mutual-learning paradigm. Firstly, to get a good initialization for the detector, we do not perform semi-supervised training directly at first. That means we only use data with annotations to warm up the detector \( D \) by Eq. 3.

\[ L_{sup} = \frac{1}{N} \sum_i L_{ac}(c_i, b_i, l_i^f) + \lambda_{cls} L_{al}(c_i^f, b_i^f) \]  

(3)

After warming up the spatial-temporal action detector \( D \), weights of \( D \) will be copied to the teacher model \( D_{teacher} \) and student model \( D_{student} \) as the initialization weights. Then we use both the labeled data and unlabeled data to further train the detector. The student model is updated via gradient back-propagation, but the gradient back-propagating to the teacher model is stopped. The teacher model is maintained by exponential moving average so as to eliminate the influence of inductive bias and provide more accurate person proposals for the student at the beginning. Loss function...
Algorithm 1 Temporal Label Assignment (TLA)

**Input:** video clip \( c^n \), boxes and labels of nearest former keyframe \( b_{left}^{i,ft} \), \( l_{left}^{i,ft} \) and nearest later keyframe \( b_{right}^{i,ft} \), \( l_{right}^{i,ft} \), detector \( D \)

**Output:** pseudo bounding boxes \( b^u \), pseudo labels \( l^u \)

1. \( b^u, s = D(c^n) \)
2. \( b^{gt}_i = b_{left}^{i,ft} \cup b_{right}^{i,ft}, l^{gt}_i = l_{left}^{i,ft} \cup l_{right}^{i,ft} \)
3. for \( i = 1, \ldots, N \)
4.  for \( j = 1, \ldots, M \)
5.     \( \text{Cost}_{ij} = L_{bce}(s_i, l^{gt}_j) + L_{L1}(b_i^u, b^{gt}_j) + L_{iou}(b_i^u, b^{gt}_j) \)
6.     Assignment \( \hat{\pi} = \arg\min_{\pi \in \Pi_M} \sum_i \text{Cost}_{i,\pi(i)} \)
7.     \( \text{inds} = [\hat{\pi}(1), \cdots, \hat{\pi}(N)] \)
8.     \( l^u = l_{gt}[\text{inds}] \)

where \( L_{bce}, L_{L1} \) and \( L_{iou} \) are binary cross entropy loss, smooth-L1 loss and GIoU loss. Weights of loss functions are set to 1. Then we use Hungarian algorithm [17] to calculate the optimal label assignment policy \( \hat{\pi} \) to minimize Eq. 7.

\[
\hat{\pi} = \arg\min_{\pi \in \Pi_M} \sum_i \text{Cost}_{i,\pi(i)}
\] (7)

Finally, we can use \( \hat{\pi} \) to assign pseudo classification label \( l^u_{\pi(i)} \) to \( i \)-th person boxes \( b^u_i \). Each ground-truth bounding boxes could only be assigned to one person proposal. If the number of proposals \( N \) is larger than the number ground-truth bounding boxes \( M \), additional background objects will be added. The cost between one prediction and one background object only contains the classification part (i.e. the binary cross entropy loss).

4. Experiments

In this section, we will provide the experiments settings, results, and ablations.

4.1. Experimental Setup

4.1.1 Datasets

We mainly use AVA [13] and JHMDB [15] to conduct all our experiments.

AVA [13] is a major dataset for benchmarking the performance of spatial-temporal action detection. It contains about 211k training clips and 57k validating video clips. The labels are annotated at 1FPS. Following standard evaluation protocol [13, 8, 7], we evaluate 60 classes among the total 80 classes. We evaluate on both versions (v2.1 and v2.2) of annotations on AVA.

JHMDB [15] consists 21 action classes and 928 clips. JHMDB is a densely annotated dataset with per-frame annotations. Following previous works [43, 5], we report the frame-level mean average precision (frame-mAP) with IoU threshold of 0.5.

4.1.2 Training Details

We use a server with eight 3090 GPUs to conduct all our experiments. We use PyTorch [30] to implement our SE-STAD. To conduct a fair comparison, we adopt the commonly used backbone, SlowOnly and SlowFast [7] network as our backbone. We use SlowOnly ResNet50, SlowFast ResNet50 and SlowFast ResNet101 with non-local [41] modules to perform experiments. For the actor localization head, we use an improved version of FCOS [39], i.e., FCOS with center sampling as our actor localization head.

Following previous works [8, 7], we use SGD with momentum as our optimizer. The hyperparameters are listed as follows: The batch size is 48 with 8 GPUs (6 clips per
Table 1. Results on AVA dataset. We report the FLOPs of action classification network plus the FLOPs of person detector for proposal-based methods. We calculate the FLOPs of person detector according to the official configure file provided by [42]. ∗ means the method reports the performance by testing with 320 resolution.

For the actor localization head, we will use a post-processing step with 0.3 scoring threshold and maximum number of 100 proposals during training. We do not perform non-maximum suppression (NMS) for actor localization head in the training stage. Then those proposals will be fed into the action classification head. The loss is showed in Sec. 3. The loss weight for $L_{al}$ is 1 and 10 for $L_{ac}$. Generated proposals which have at least 50% intersection-over-union (IoU) with the ground-truth boxes will be treated as positive proposals in the action classification stage, otherwise those proposals will be ignored.

4.2. Results on AVA

In this section, we will provide results and analyses of results on AVA. The results are listed in Table 1. From the table, we can have the following observations:

- The extra person detector will bring a huge computational burden to the spatial-temporal action detection model.

model. The FLOPs of the person detector is 406.5G FLOPs, which is around 7 times larger than the FLOPs of SlowOnly R50 (4 × 16), and more than 2 times larger than the heaviest backbone SlowFast R101-NL (8 × 8). The large FLOPs come from the high input resolution in the person detector. The high input resolution along with the high FLOPs makes proposal-based methods hard to apply in real-world scenarios.

- With a simply added component, i.e., FCOS head, our model can have roughly comparable or better performance than proposal-based methods, even than the recent WOO [5] and we do not use extra SSAD techniques. This is quite encouraging because we have around 70–90% FLOPs drop with proposal-based SlowFast, and we have around 20–35% FLOPs drop with WOO [5]. This shows the effectiveness of our simple models. We will dive into the model details part to figure out what makes the simple model work so well.

- With the extra SSAD techniques (the semi-supervised learning stage and the temporal labeling assignment), our model can have an extra performance boost with no extra modules and no computational cost. For example, SlowFast R50 can have an extra 1.4% mAP performance boost onAVA v2.2 and 1.5% mAP boost on AVA v2.1. Similar performance gaps are observed on SlowFast R101. However, SSAD stage can only have a 0.5% performance gain on SlowOnly R50. We conjecture that it may be due to the input capacity. SlowOnly R50 only has 4 frames as input. The low number of input frames prevents SlowOnly R50 to have better performance. We can achieve 31.8 mAP onAVA v2.1, which is 0.2 mAP higher than TubeR [49], and our model has 20% fewer FLOPs than TubeR. TubeR uses extra encoder-decoder structure with Transformers to perform end-to-end STAD. Our SE-STAD, have comparable or better performance and simple design.

### 4.3. Results on JHMDB

To verify the effectiveness of SE-STAD, we further evaluate our model on JHMDB [15]. Since JHMDB is densely-annotated, we directly apply the basic SE-STAD model. The results are in Table 2. From the table, we can observe that: SE-STAD models can achieve 82.5% mAP with SlowFast R101 8x8 backbone, which is 2.0% higher than WOO [5]. Even with a weaker backbone, SE-STAD can still achieve 80.7% mAP, which is still 0.2% higher than WOO. These results show the effectiveness of SE-STAD.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>JHMDB mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context-Aware RCNN [43]</td>
<td>SlowFast R101-NL 8x8</td>
<td>80.5</td>
</tr>
<tr>
<td>WOO [5]</td>
<td>SlowFast R101-NL 8x8</td>
<td>80.5</td>
</tr>
<tr>
<td>SlowFast R50 8x8</td>
<td>SlowFast R101-NL 8x8</td>
<td>82.5</td>
</tr>
</tbody>
</table>

Table 2. Results on JHMDB dataset.

In this section, we will provide ablations of our model, including the head choice for actor localization, loss coefficients, input resolutions and the methods to train the classification head. In this section, unless specified, all experiments use SlowFast R50 (8 × 8) as the backbone network.

### 4.4. Ablation Study

#### 4.4.1 Head Choices for Actor Localization

In this section, we vary the head of actor localization for SE-STAD. We try different heads, including the popular anchor-based heads: RPN + RCNN [33], RetinaNet [23] and GFocalV2 [21]. The ablation results are in Table 3. From Table 3 we can observe that, Anchor-based heads perform significantly worse than anchor-free heads, i.e., FCOS [39]. Two-stage RPN+RCNN [33] and RetinaNet [23] have a large performance drop. Even the most recent GFocalV2 head (anchor-based version) will have a 1.8% mAP gap with the FCOS head. Besides, tricks on FCOS will improve around 0.6% mAP, and the original FCOS head will still achieve a 24.9% mAP. It may be due to the low input resolution and pre-defined anchor shape. Moreover, with the simple FCOS head, our model performs slightly better than WOO [5]. WOO has an extra attention module. In contrast, our SE-STAD keeps a simple architectural design and has a good performance.

<table>
<thead>
<tr>
<th>Actor Heads</th>
<th>Type</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPN+RCNN [33]</td>
<td>Anchor-Based</td>
<td>21.0</td>
</tr>
<tr>
<td>RetinaNet [23]</td>
<td>Anchor-Based</td>
<td>19.7</td>
</tr>
<tr>
<td>GFocalV2 [21]</td>
<td>Anchor-Based</td>
<td>23.7</td>
</tr>
<tr>
<td>FCOS- [39]</td>
<td>Anchor-Free</td>
<td>24.9</td>
</tr>
<tr>
<td>WOO* [5]</td>
<td>Anchor-Free</td>
<td>25.4</td>
</tr>
</tbody>
</table>

Table 3. Ablation study on different heads for actor localization. We try different heads with SlowFast R50 backbone. We apply the anchor-based version of GFocalV2 [21]. FCOS- is the original FCOS [39] version without tricks. We do not use self/semi-training for all methods. WOO [5] is listed only for comparison.

#### 4.4.2 Different Strategies to Train Action Classification

The action classification part is the other important part for SE-STAD. We will use different strategies to train our models to ablate our FCOS head. We vary the input of action classification head between training and testing, and verify the performance of our model.

The results are in Table 4. We can find that:

- When testing with pre-extracted proposals, our model can have better performance than FCOS generated boxes. It is not surprising because we are performing actor localization with low-resolution inputs.

- However, our model still performs better than proposal-based methods. Also, our model trained with sparse in-
To try different inputs for both training and testing of our models. For “GT Only”, we only feed the ground-truth boxes into the action classification head. For “FCOS Output (Sparse)”, we perform NMS to FCOS generated proposals in the training stage. For “FCOS Output (Dense)”, we do not perform NMS in the training stage.

Table 4. Ablation study on different training inputs for action classification. We try different inputs for both training and testing of our models. For “GT Only”, we only feed the ground-truth boxes into the action classification head. For “FCOS Output (Sparse)”, we perform NMS to FCOS generated proposals in the training stage. For “FCOS Output (Dense)”, we do not perform NMS in the training stage.

<table>
<thead>
<tr>
<th>Training input</th>
<th>Testing input</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposal</td>
<td>Proposal</td>
<td>24.7</td>
</tr>
<tr>
<td>GT Only</td>
<td>Proposal</td>
<td>24.5</td>
</tr>
<tr>
<td>GT Only</td>
<td>FCOS Output</td>
<td>23.7</td>
</tr>
<tr>
<td>FCOS Output (Sparse)</td>
<td>FCOS Output</td>
<td>24.3</td>
</tr>
<tr>
<td>FCOS Output (Dense)</td>
<td>FCOS Output</td>
<td>25.5</td>
</tr>
<tr>
<td>FCOS Output (Dense)</td>
<td>Proposal</td>
<td>26.2</td>
</tr>
</tbody>
</table>

Table 5. Ablation study on \( \lambda_{unsup} \) and \( \lambda_{cls} \). We study the effect of \( \lambda_{unsup} \) and \( \lambda_{cls} \) to verify the robustness of SE-STAD.

<table>
<thead>
<tr>
<th>Method</th>
<th>( \lambda_{cls} )</th>
<th>( \lambda_{unsup} )</th>
<th>val mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE-STAD</td>
<td>1</td>
<td>-</td>
<td>25.2</td>
</tr>
<tr>
<td>SE-STAD</td>
<td>10</td>
<td>-</td>
<td>25.5</td>
</tr>
<tr>
<td>SE-STAD</td>
<td>20</td>
<td>-</td>
<td>24.7</td>
</tr>
<tr>
<td>SE-STAD+TLA</td>
<td>10</td>
<td>0.2</td>
<td>26.8</td>
</tr>
<tr>
<td>SE-STAD+TLA</td>
<td>10</td>
<td>0.5</td>
<td>26.9</td>
</tr>
<tr>
<td>SE-STAD+TLA</td>
<td>10</td>
<td>1.0</td>
<td>26.5</td>
</tr>
</tbody>
</table>

4.4.3 Ablations on Loss coefficients

As stated in Sec. 3, we introduce \( \lambda_{cls} \) to balance the actor localization and action classification loss. We also introduce \( \lambda_{unsup} \) to balance labeled and unlabeled losses. Here, we make ablation experiments to show the robustness of each coefficient. Results in Table 4 support the robustness of \( \lambda_{cls} \) and \( \lambda_{unsup} \). \( \lambda_{cls} = 10 \) achieves best performance. Besides, when utilizing the unlabeled data, ablation experiments show that it’s better to set the ratio of loss weight between the labeled part and unlabeled part to 2:1.

4.4.4 Computational Efficiency

In this section, we will show the computational efficiency of our model under different input resolutions. We vary the input resolution for our model during testing. The results are in Table 7. We can observe that with the default 256 input resolution, we perform slightly better than WOO [5] and proposal-based SlowFast [7]. When we use a larger input resolution, i.e., 320, we can get a 0.6% performance boost and slightly higher FLOPs than WOO [5] but much lower than proposal-based SlowFast.

4.4.5 Ablations on Pseudo Label Generation

For the semi-supervised action detection part, pseudo label generation is the critical part for this part. It’s a natural way to perform pseudo label generation by predicting the classification labels directly. However, as stated before, multi-

5. Conclusion

In this paper, we presented SE-STAD, an end-to-end method for spatial-temporal action detection. SE-STAD has a simple design and small computational burden, yet achieves good results across the major spatial-temporal action detection dataset. The performance gain comes from two parts: one is the powerful anchor-free detector head. The other is the proposed novel semi-supervised training schema along with the label assignment strategy. We hope that our model, notwithstanding its simplicity, can en-light the broader problem of video understanding. We will continue to explore the multi-label and long-tailed problems that existed in spatial-temporal action detection.
References


