PointTAD: Multi-Label Temporal Action Detection with Learnable Query Points

Anonymous Author(s)
Affiliation
Address
email

Abstract

Traditional temporal action detection (TAD) usually handles untrimmed videos with small number of action instances from a single label (e.g., ActivityNet, THUMOS). However, this setting might be unrealistic as different classes of actions often co-occur in practice. In this paper, we focus on the complex multi-label temporal action detection that aims to localize all action instances from a multi-label untrimmed video. Multi-label TAD is more challenging as it requires for fine-grained class discrimination within a single video and dedicated module to precisely localize the co-occurring instances. Specifically, we extend the sparse query-based detection paradigm from the traditional TAD and propose the multi-label TAD framework of PointTAD. Existing query-based action detectors employs a segment to represent an action instance, which is insufficient to handle the concurrent instances and their richer relations. To mitigate this issue, our PointTAD introduces a small set of learnable query points to represent important frames of each action instance. Our PointTAD provides a flexible mechanism to localize the discriminative frames at boundaries and as well the important frames inside the action. Moreover, we improve action decoding with the Multi-level Interactive Module to integrate action semantics at point-level and instance-level. Finally, our PointTAD employs an end-to-end trainable framework based on RGB input for easy deployment. We evaluate our proposed method on two popular benchmarks for multi-label TAD. Our model outperforms all previous methods by a large margin under the detection-mAP metric and achieves promising results under the segmentation-mAP metric.

1 Introduction

With the increasing amount of video resources on the Internet, video understanding is becoming one of the most important topics in computer vision. Temporal action detection (TAD) [42, 19, 17, 4, 9, 15, 31] has been formally explored in traditional benchmarks such as THUMOS [13], ActivityNet [12], and HACS [41]. However, the task seems impractical because their videos almost contain non-overlapping actions from a single category: 85% videos in THUMOS are annotated with single action category. As a result, most TAD methods [19, 17, 36, 5, 29] simply cast this TAD problem into sub-problems of action proposal generation and global video classification. In this paper, we shift our playground to the more complex setup of multi-label temporal action detection, which aims to detect all action instances from multi-labeled, untrimmed videos. Existing works [8, 14, 32, 7] in this field formulate the problem as a dense segmentation task and perform multi-label classification in a frame-wise manner. Consequently, these methods are weak in localization and fails to provide the instance-level detection results (i.e., the starting time and ending time of each instance). In analogy to image instance segmentation [18], we argue that it is necessary to redefine multi-label TAD as a instance-level detection problem rather than a dense segmentation task. In this sense, multi-label TAD results not only provide the action labels, but also the exact temporal extent of each instance.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

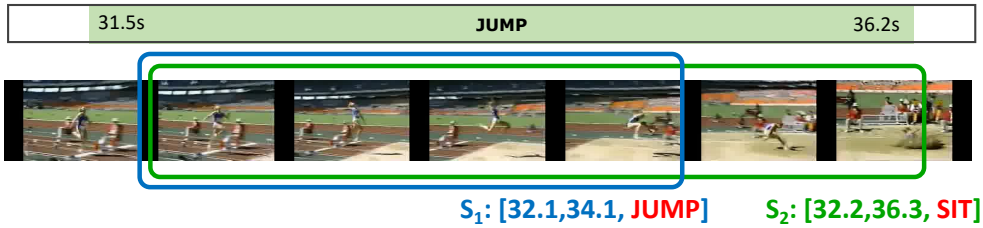

Figure 1: Illustration of action predictions by segment-based action detectors in multi-label TAD.

Direct adaptation of action detectors is insufficient to deal with the challenges of concurrent instances and complex action relations in multi-label TAD. The convention of extracting action features from **action segments** [4, 26, 40, 34] lacks the flexibility of handling both important semantic frames inside the instance as well as discriminative boundary frames. Consider the groundtruth action of *"Jump"* in Fig. 1, segment-based action detectors mainly produce two kinds of predictions, as in type $\mathbf{S_1}$ and type $\mathbf{S_2}$. $\mathbf{S_1}$ successfully predicts the correct action category with an **incomplete** segment of action highlights, whereas $\mathbf{S_2}$ does a better job in locating the boundaries yet get **misclassified** as *"Sit"* due to the distracting frames of landing. In addition, most action detectors [40, 26, 21] are inadequate in processing sampled frames and classifying fine-grained action classes. They often exploit temporal context modeling at single level and ignore the exploration of channel semantics.

To address the above issues, we present PointTAD, a sparse query-based action detector that leverages learnable query points to flexibly attend important frames for action instances. Inspired by RepPoints [38], query points are directly supervised by regression loss. Given specific regression targets, the query points learn to localize at discriminative frames at action boundaries as well as semantic key frames within actions. Hence, concurrent actions of different categories can achieve distinctive features through the specific arrangement of query points. Moreover, we improve the action location and semantics decoding by proposing the Multi-level Interactive Module with dynamic kernels based on query vector. Direct interpolation or pooling at each query point lacks temporal reasoning over consecutive frames. We extract point-level features with deformable convolution [6, 43] from a local snippet to capture the temporal cues of action change or important movement. At instance-level, both temporal context and channel semantics are captured with frame-wise and channel-wise dynamic mixing [30, 10] to further describe the distinctive features of simultaneous actions.

PointTAD streamlines end-to-end trainable temporal action detection with joint optimization of the backbone network and action decoder. We validate our model on two popular benchmarks of the challenging multi-label TAD. Our model achieves the state-of-the-art detection-mAP performance and competitive segmentation-mAP performance to previous methods with RGB input only.

2 Related work

Multi-label temporal action detection. Multi-label temporal action detection has been studied as a multi-label frame classification problem in the previous literature. Early methods [24, 25] paid a lot of attention to model the temporal relations between frames with the help of Gaussian filters in temporal dimension. Other works deployed dilated temporal kernels [8] or iterative temporal convolution and self attention blocks [7] to integrate features at different temporal scales. Recently, attention has shifted beyond temporal modeling. Coarse-Fine[14] employed X3D backbone to handle different temporal resolutions in the slow-fast fashion and perform spatial-temporal attention during fusion. MLAD [32] used multi-head self-attention blocks at both spatial and class dimension to model class relations at each timestamp. In our proposed method, we study the temporal context at different semantic levels and model the channel dynamics within instances. In addition, we take into account the paired relations between action starts and ends, and employ query-based action detection framework for more accurate and precise action localization.

Segment representation in TAD. Following the prevailing practice of bounding boxes [16, 11, 27, 1] in object detection, existing temporal action detectors incorporated action segments heavily in their frameworks with three kinds of usage: as anchors to estimate action location, as proposals for subsequent recognition and as final predictions. We divide temporal action detectors mainly into three kinds to explain their usage of action segments: anchor-based, anchor-free and query-based methods. Segments as anchors are explored mainly in anchor-based frameworks. Early methods [23, 22]

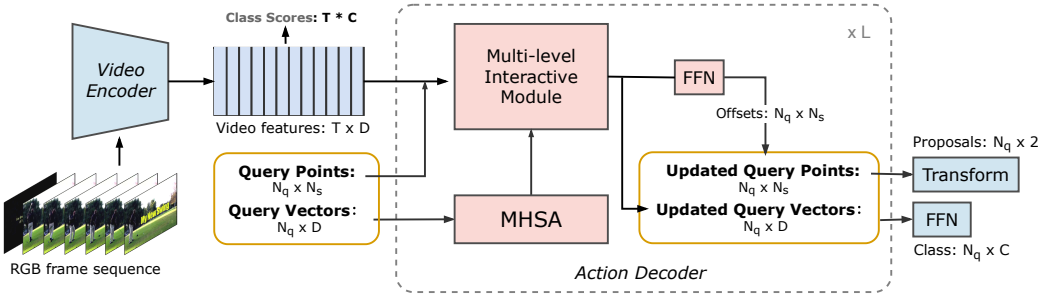

Figure 2: **Pipeline of PointTAD**. It consists of a backbone network that extracts video features from consecutive RGB frames and an action decoder of L layers that directly decodes actions from video features. Each decoder layer contains a Multi-Head Self-Attention (MHSA) block for pair-wise query communication and a Multi-level Interactive Module with dynamic modeling based on the query vector. PointTAD enables end-to-end training of backbone and action decoder without any post-processing of predictions.

used sliding windows to generate pre-defined anchors and methods to date [40, 26] took in proposal generation results as anchors. TAL-Net [4] followed Faster R-CNN [27] to learn its own anchor segments. Segments as intermediate proposals are exploited in both anchor-based and anchor-free methods. Anchor-based methods commonly adopt uniform sampling within proposal segments to generate action features. P-GCN [40] applied max-pooling over all frames within local segments for proposal features. G-TAD [36] uniformly divided segments into bins and average the frame features in each bin to achieve proposal features. Anchor-free methods, such as AFSD[15], converted center points and boundary offsets to segment proposals and proposed novel boundary pooling module to boost boundary information in refined action features. Segments as final predictions are employed among all three types of TAD frameworks, because segments generally facilitate the computation of action overlaps and loss calculation. In this paper, we replace temporal segments in the first and second usage with learnable query points. Compared with anchor-free method, we liberate the extraction of action features from uniform bins and handcrafted boundary regions with flexible frame sampling strategy that characterize the action semantics as well as boundaries. In contrast to query-based methods that followed DETR and its variant style [2, 44], PointTAD replaces the global, dense attention with sparse learnable query points under direct regression supervision to better encode action locations and semantics

Temporal context in videos. Context aggregation at different levels of semantics is crucial for temporal action reasoning and has been discussed in previous TAD methods. G-TAD [36] treated each snippet input as graph node and applied graph convolution networks to enhance snippet-level features with global context. ContextLoc [45] handled action semantics in hierarchy: it updated snippet features with global context, obtained proposal features with frame-wise dynamic modeling within each proposal and modeled the inter-proposal relations with GCNs. Although we considered the same levels of semantic modeling, our method is different from ContextLoc. On the contrary, PointTAD focuses on aggregating *local* temporal cues at snippet-level, attaining informative query feature by frame and *channel* attention and applying multi-head self-attention for inter-query relation modeling.

3 PointTAD

3.1 Overview

We formulate the task of multi-label TAD as a set prediction problem. Formally, given a video segment with T consecutive frames, we predict a set of action instances $\Psi = \{\psi_n = (t_n^s, t_n^e, c_n)\}_{n=1}^{N_q}$, N_q is the number of learnable queries. The groundtruth action set to detect are denoted $\hat{\Psi} = \{\hat{\psi} = (\hat{t_n^s}, \hat{t_n^e}, \hat{c_n})\}_{n=1}^{N_g}$, where $\hat{t_n^s}, \hat{t_n^e}$ are the starting and ending timestamp of the n-th action, $\hat{c_n}$ is the groundtruth action category, N_g is the number of groundtruth actions.

The overall architecture of PointTAD is depicted in Fig. 2. PointTAD consists of a **backbone video encoder** and an **action decoder**. The model takes three inputs: RGB frame sequence of length T, a set of learnable query points $\mathbf{p} \in \mathbb{R}^{N_q \times N_s}$ and one query vector $\mathbf{q} \in \mathbb{R}^{N_q \times D}$. Learnable query points describe the action locations by positioning themselves on action transits as well as semantic key frames, and the query vector encodes action semantics. The backbone video encoder extracts video features $X \in \mathbb{R}^{T \times D}$ from RGB frames. The action decoder contains L stacked decoder layers and takes query points, query vector $\mathbf{q}$ and video features X as input. Each decoder layer contains two parts: 1) the multi-head self-attention [33] block models the pair-wise relationship of query vectors; 2) the **Multi-level Interactive Module** models the point-level and instance-level semantics with dynamic weights based on query vector. From local to global, the action decoder accomplishes context modeling at **point-wise**, **intra-proposal** and **inter-proposal** levels. Finally, we decode action labels from query vectors and transform query points to pseudo segments for loss computation and prediction output.

3.2 Learnable Query Points

Segment representation (i.e., representing each instance simply with a starting and ending time) is limited in describing boundary and action characteristics at the same time without redundancy. We present a novel query point representation to automatically learn the positions of action boundary as well as semantic key frames. The query point representation is denoted $\mathcal{P} = \{t_i\}_{i=1}^{N_s}$ for each query, where t_i is the temporal location of ith query point, and the point quantity per query N_s is set to 21 empirically. We explain the updating strategy and the learning of query points below.

Iterative Point Refinement. The query points are refined by a query vector through iterations of decoder layers to reach final positions. We design a self-paced updating strategy with adaptive scaling for each query at each layer to stabilize the training process. At decoder layer l, the query points are represented by $\mathcal{P}^l = \{t_i^l\}_{i=1}^{N_s}$. The query vector predicts N_s offsets, denoted $\{\Delta t_i^l\}_{i=1}^{N_s}$, for each query point. The refinement can be summarized as:

$$\mathcal{P}^{l+1} = \{(t_i^l + \Delta t_i^l \cdot s^l \cdot 0.5)\}_{i=1}^{N_s}, \tag{1}$$

where $s^l = max(\{t_i^l\}) - min(\{t_i^l\})$ is the scaling parameter and describes the span of query points at layer l. As a result, the updated step size gets smaller for shorter action, which helps with the localization of short actions. Updated query points from previous layer are inputs to the next layer.

Learning Query Points. The training of query points is directly supervised by regression loss. We follow [38] to transform query points to pseudo segments for regression loss calculation. The converted pseudo segments participate in the calculation of L1-loss and tIoU loss with groundtruth action segments in both label assignment and loss computation.

The transformation function is denoted by $\mathcal{T} : \mathcal{P} \rightarrow \mathcal{S} = (t^s, t^e)$. We experiment with two kinds of functions: Min-max $\mathcal{T}_1$ and Partial min-max $\mathcal{T}_2$. *Min-max* is to take the minimum and maximum location from the set of query points as the starting and ending timestamp of the pseudo segment, $\mathcal{T}_1 : \mathcal{P} \rightarrow min(\{t_i\}_{i \in \mathcal{P}}), max(\{t_i\}_{i \in \mathcal{P}})$. With Min-max transformation, the query points are strictly bounded within the local segment of the target action instance. *Partial min-max* function is to select a subset of query points $\mathcal{P}_{local}$ and perform the min-max function on them to determine a pseudo segment, $\mathcal{T}_2 : \mathcal{P} \rightarrow min(\{t_i\}_{i \in \mathcal{P}_{local}}), max(\{t_i\}_{i \in \mathcal{P}_{local}})$. It allows several query points to aggregate information from outside the action region. Empirically, we choose partial min-max by default.

3.3 Multi-level Interactive Module

Apart from the limited segment representation, previous temporal action detectors are also insufficient in decoding sampled frames. These methods seldom consider semantic aggregation from different levels and at multiple aspects. Accordingly, we present a multi-level interactive module that consider both **local temporal cues at point-level** and **intra-proposal relation modeling at instance-level**, depicted in Fig. 3. To achieve distinct representations for each query, these parameters at both levels are dynamically generated based on the query vector.

Point-level Locality Preservation. Paired with the refined point representation, we employ deformable convolutions to extract point features within the local neighborhood. For each query point,

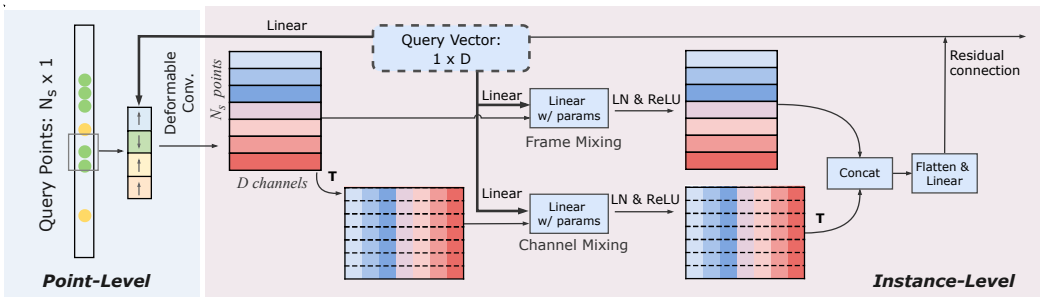

Figure 3: **Multi-level Interactive Module** aggregate action semantics at point-level and instance-level with dynamic parameters. The model takes in the query vector to interact with local snippets centered at query points and intra-proposal point features at channel and temporal aspects, then outputs informative features to update query vector with residual connection.

we predict four temporal offsets $\{o_j\}$ from the point location and corresponding weights $\{w_j\}$. The query point t_i act as the center point and is added with temporal offsets to achieve four deformable sub-points. These sub-points characterize the local neighborhood of center points. The features at sub-points are extracted by a bilinear interpolation, then multiplied with weights and combined together to get point-level feature $x(t_i)$. This process can be expressed as:

$$x(t_i) = \sum_j x(t_i + o_j) \cdot w_j. \tag{2}$$

Both the offsets and weights are generated from query vector $\mathbf{q}$ by linear projection,

$$\{o_j\} = \mathrm{Linear}(\mathbf{q}), \;\; \{w_j\} = \mathrm{Softmax}(\mathrm{Linear}(\mathbf{q})), \tag{3}$$

where the weights are additionally normalized by softmax.

Instance-level Adaptive Mixing. Faced with the challenge of simultaneous actions, with only temporal modeling, actions with large overlap may result in similar representation and harms the classification. To tackle this problem, we propose adaptive mixing at both frames and channels with dynamic temporal convolution of kernel size 1. The stacked features of query points is denoted by $x \in \mathbb{R}^{N_s \times D}$. Given the query vector $\mathbf{q}$, we generate dynamic parameters for frame mixing and channel mixing. Frame mixing is carried out with dynamic projection followed by LayerNorm and ReLU activation on N_s points to explore intra-proposal frame relations:

$$M_f = \mathrm{Linear}(\mathbf{q}) \in \mathbb{R}^{N_s \times N_s}, \tag{4}$$

$$\mathrm{FrameMix}(x) = \mathrm{ReLU}(\mathrm{LayerNorm}(x^T M_f)). \tag{5}$$

Similar to frame mixing, channel mixing uses two bottle-necked layers of dynamic projection on the channel dimension to enhance action semantics:

$$M_{c,1} = \mathrm{Linear}(\mathbf{q}) \in \mathbb{R}^{D \times D'}, \tag{6}$$

$$M_{c,2} = \mathrm{Linear}(\mathbf{q}) \in \mathbb{R}^{D' \times D}, \tag{7}$$

$$\mathrm{ChannelMix}(x) = \mathrm{ReLU}(\mathrm{LayerNorm}(\mathrm{ReLU}(\mathrm{LayerNorm}(xM_{c,1}))M_{c,2})). \tag{8}$$

These Two mixed features are concatenated along channel and get squeezed by flatten and linear operations to the size of query vector. The query vector is then updated with residual connection.

3.4 Video Encoder

We exploit the commonly used I3D backbone [3] as the video encoder in our framework. The video encoder is trained end-to-end with the action decoder and optimized by action detection loss to bridge the domain gap between action recognition and detection. For easy deployment of our framework on online platforms, we avoid the usage of optical flow due to its dreadful preparation procedure. In order to achieve good performance on par with two-stream features using only RGB input, we follow [37] to remove the temporal pooling at Mixed_5c and fuse the features from Mixed_5c with features from Mixed_4f as in [20]. As a result, the temporal stride of encoded video features is 4. Average pooling is performed to squeeze the spatiotemporal representations from backbone to temporal features.

3.5 Training and Inference

Label Assignment. Similar to all query-based detectors, we apply hungarian matcher on pseudo segments to search for the optimal permutation $\sigma(\cdot)$ for label assignment. The groundtruth set $\hat{\Psi}$ on each video segment is extended with no action $\varnothing$ to the size of N_q. The matching cost is formulated as:

$$\mathcal{C} = \sum_{n:\sigma(n)\neq\varnothing} \alpha_{L1} \cdot \mathcal{L}_{L1}(\mathcal{T}(\mathcal{P}_n), \hat{\psi}_{\sigma(n)}) - \alpha_{iou} \cdot \text{tIoU}(\mathcal{T}(\mathcal{P}_n), \hat{\psi}_{\sigma(n)}) - \alpha_{cls} \cdot c_n. \tag{9}$$

By minimizing the matching cost, the bipartite matching algorithm finds the desired permutation $\sigma(\cdot)$ that assigns each prediction with a target. $\alpha_{L1}, \alpha_{iou}, \alpha_{cls}$ is set to $1, 5, 5$ respectively.

Loss Functions. PointTAD is jointly optimized by localization loss and classification loss. We use L1 loss and iou loss as localization loss:

$$\mathcal{L}_{loc} = \sum_{n:\sigma(n)\neq\varnothing} \mathcal{L}_{L1}(\mathcal{T}(\mathcal{P}_n), \hat{\psi}_{\sigma(n)}) + (1 - \text{tIoU}(\mathcal{T}(\mathcal{P}_n), \hat{\psi}_{\sigma(n)})). \tag{10}$$

The cross-entropy loss between query labels and target labels is used as classification loss. In addition, to facilitate the performance under segmentation-mAP, we generate dense classification scores $S \in \mathbb{R}^{T\times C}$ by linear projection from video features X. Cross-entropy loss is enforced on S with dense groundtruth $\hat{S} \in \mathbb{R}^{T\times C}$. Therefore, the classification loss is composed of the set prediction loss and the dense per-frame loss:

$$\mathcal{L}_{cls} = [\sum_{n} \mathcal{L}_{ce}(c_n, \hat{c}_n)] + \lambda_{seg}\mathcal{L}_{ce}(S, \hat{S}). \tag{11}$$

The overall loss function is formulated as follows:

$$\mathcal{L} = \lambda_{loc} \cdot \mathcal{L}_{loc} + \lambda_{cls} \cdot \mathcal{L}_{loc}, \tag{12}$$

where λ_{cls}, λ_{loc} and λ_{seg} are hyper-parameters that are set to $5, 10, 0.5$ respectively.

Inference. During inference, the network uses a single linear projection layer followed by LayerNorm and ReLU activation to predict class labels from query vectors. As for localization, we use pseudo segments transformed from query points as the final predictions. These sparse predictions are then tested under the detection-mAP metric. Additional dense scores S could be generated at video features for the segmentation-mAP calculation. The sparse predictions are filtered with threshold γ and processed with Gaussian kernels to approximate dense scores at each frame. Then, the approximated scores are added with weight β to the predicted dense scores for segmentation-mAP calculation. The final dense scores are generated as:

$$S_{final} = \beta \cdot \sum_{n=1}^{N_q} \mathbb{1}_{c_n>\gamma} \cdot \text{Gaussian}(\psi_n) + (1-\beta) \cdot S. \tag{13}$$

Note that the sparse predictions are adjusted by dense scores only for segmentation-mAP.

4 Experiments

4.1 Dataset and Setup

Dataset. We conduct experiments on two popular multi-label action detection benchmarks: MultiTHUMOS [39] and Charades [28]. **MultiTHUMOS** is a densely labeled dataset extended from THUMOS'14. It includes 413 sports videos of 65 classes. The dataset has an average number of distinctive action categories per video of 10.5, compared with 1.1 in THUMOS14. **Charades** is a large Multi-label TAD dataset that contains 9848 videos of daily in-door activities. The annotations are spread over 157 action classes, with an average of 6.8 instances per video.

Implementation Details. With I3D backbone network, we encode videos at 10 fps for MultiTHUMOS and 12 fps for Charades. The spatial resolution is set to 192×192 for both datasets. The video

Table 1: **Comparison with the state-of-the-art** on MultiTHUMOS test set and Charades test set, under detection-mAP (%) and segmentation-mAP(%).

Methods	MultiTHUMOS		Charades	
	detection-mAP	segmentation-mAP	detection-mAP	segmentation-mAP
R-C3D [35]	-	-	-	12.7
Super-event [24]	-	36.4	-	18.6
TGM [25]	-	37.2	-	20.6
PDAN [8]	17.1	40.2	5.0	23.7
Coarse-Fine [14]	-	-	3.2	25.1
MLAD [32]	13.2	42.2	-	18.4
MS-TCT [7]	15.3	**43.1**	4.1	**25.4**
Ours	**21.5**	39.8	**11.1**	21.0

sequence is processed with the widely-used sliding window mechanism. To accommodate most of the actions, the window size is set to 256 frames for MultiTHUMOS (99.1% actions included), and 400 frames for Charades (97.3% actions included). Short videos in Charades are padded with the last input frame. The overlap ratio is 0.75 at training, and 0 at inference.

Appropriate **initialization** is required for backbone, query points and point-level deformable convolutions for stable training. Following convention, the I3D backbone are initialized with K400 pre-trained weights for MultiTHUMOS and Charades pre-trained weights for Charades. Query points are initialized with constant 0.5. Other possible initialization are explored in ablations. The linear layer to produce deformable offsets are initialized as follows: zeroing for weights and $[1, 2, 3, 4]$ for biases. The weights and biases to generate deformable weights are initialized as zero.

We adopt AdamW as optimizer with 1e-4 weight decay. The network is trained on 8-card V100 GPU. The batch size is 3 per GPU. The learning rate is set to 2e-4 and drops by half at every 12 epochs. Backbone learning rate is additionally multiplied with 0.1 for stable training.

Evaluation Metrics. The default evaluation metric for multi-label TAD is segmentation-mAP, which is the per-frame mean average precision. In addition, we extend the detection-mAP metric from traditional TAD to further evaluate the completeness of predicted action instances. The detection-mAP is the mean average precision of action predictions under different tIoU thresholds. We report the average mAP as well as mAP with tIoU threshold set $\{0.1, 0.2, 0.3\}$ for both datasets. The average detection-mAP is calculated with tIoU thresholds set $[0.1 : 0.1 : 0.9]$. We argue that detection-mAP is more reasonable for an instance detection algorithm.

4.2 Comparison with state-of-the-art methods

In Tab. 1, we compare the performance of PointTAD with previous multi-label TAD methods under both segmentation-mAP and detection-mAP. The sparse prediction tuples of PointTAD are converted to dense segmentation scores with Eq. (13) for segmentation-mAP. In order to compare the detection-mAP, we reproduce some of the multi-lable TAD methods and convert their dense segmentation results to sparse predictions following [39]. The prediction confidence of an action with l consecutive frames is calculated as:

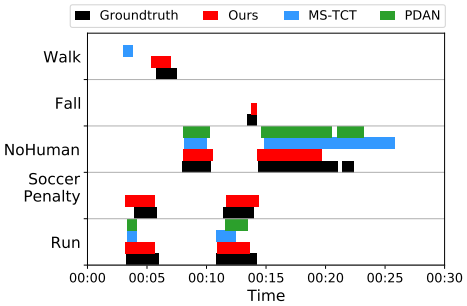

Figure 4: **Qualitative results** on the MultiTHUMOS test set.

$$\text{score}(C, p_1 ... p_L) = (\sum_{i}^{l} p_i) \times \exp(\frac{-0.01(l - \mu_C)^2}{\sigma_c^2}), \tag{14}$$

where p_i is the probability for class c at frame i, μ_c and σ_c is the mean and standard derivation of action duration of class c in the training set.

Our method surpasses all previous multi-label TAD methods by large under detection-mAP, indicating our ability to predict complete actions is beyond previous dense segmentation models. As for

Table 2: **PointTAD Ablation experiments** on MultiTHUMOS. Default setting is colored gray.

(a) **Segments** vs. **Query Points** in query-based action detectors

	0.1	0.2	0.3	Avg
Segment	35.4	33.1	30.0	19.4
Point	39.1	36.6	33.0	21.5

(b) **Initialization** of query points at training.

Init.	0.1	0.2	0.3	Avg
Uniform(0,1)	39.7	36.9	33.2	21.3
Norm(0.5,0.3)	39.9	37.0	33.1	21.2
Const.(0.5)	39.1	36.6	33.0	21.5

(c) **Point2Segment Transformation** $\mathcal{T} : \mathcal{P} \rightarrow \mathcal{S}$.

$\mathcal{T}$	0.1	0.2	0.3	Avg
Min-max	39.0	36.6	33.1	21.6
Partial Min-max	39.1	36.6	33.0	21.5

(d) **Number of Query Points** N_s per query.

N_s	0.1	0.2	0.3	Avg
9	38.6	36.2	32.4	21.0
15	38.9	36.6	33.2	21.4
21	39.1	36.6	33.0	21.5
27	39.0	36.6	32.9	21.4

(e) **Point-Level**: Deformable Convolution.

Deform. Conv.	0.1	0.2	0.3	Avg
✓	39.1	36.6	33.0	21.5
	38.1	35.7	32.2	20.8

(f) **Instance-Level**: different variants of mixing strategy.

Mixing	0.1	0.2	0.3	Avg
Frame Only	38.5	35.7	32.3	20.9
Channel Only	36.8	34.2	30.8	20.1
Frame→Channel	37.0	34.3	30.9	19.9
Channel→Frame	33.2	30.7	27.4	17.1
Parallel Mixing	39.1	36.6	33.0	21.5

segmentation-mAP, we achieve comparable results to previous methods on both benchmarks, which is encouraging for a sparse detection framework.

In Fig. 4, we show the qualitative results of PointTAD on the MultiTHUMOS test set compared to PDAN and MS-TCT. PointTAD detects more instances with finer localization especially at harder categories, such as "Fall" and "SoccerPenalty". MS-TCT and PDAN perform better at "NoHuman" category, we argue that this is because "NoHuman" class is not a well-defined action category that has paired action boundaries, whereas PointTAD leverages pair relations of boundaries for localization.

4.3 Ablation Study

Segment representation vs Point representation. In Tab. 2a, we compare the performance between segment representation and query point representation. In segment implementation, we apply temporal roi align to segments as in [34]. Temporal roi align divides the segment into uniform bins and apply average pooling on each bin to arrange frame features for query vector. The pooler resolution for RoI Align is set to N_s, the same as query point quantity. Parallel dynamic mixing is carried out in both implementations. Results show that point representation significantly outperforms segment representation, demonstrating the strength of adaptive sampling over grid-like alignment upon static segments.

Study on query point initialization. We consider three different initialization for query points at training: uniform distribution in $[0, 1]$, normal distribution with mean=0.5 and std = 0.3, and initialization with constant 0.5. Results in Tab. 2b indicates that all three initialization methods are beneficial for the training, with little performance different in avg-mAP. In addition, normal distribution and uniform distribution achieve higher detection-mAP with smaller tIoU threshold, which shows these two initialization methods are weaker at accurate localization. Empirically, we set constant initialization as default.

Study on pseudo segment transformation functions. Tab. 2c shows two alternatives of transformation function $\mathcal{T}$. Min-max and Partial min-max achieve similar performance under average detection-mAP. We choose Partial min-max as the default function because it offers a more relaxed constraint for query points.

Study on the number of query points. We ablate with different numbers of query points N_s in Tab. 2d. In general, the performance is in proportion to the point quantity, although the increase of N_s only benefits the performance by little. N_s reaches the sweet spot at 21, where the performance no longer increases with larger N_s. To balance the model complexity and performance, we set N_s to 21.

Study on point-level deformable operator. In Tab. 2e, we compare point-level deformable convolution and direct interpolation for point-level feature extraction. Using deformable operator achieves

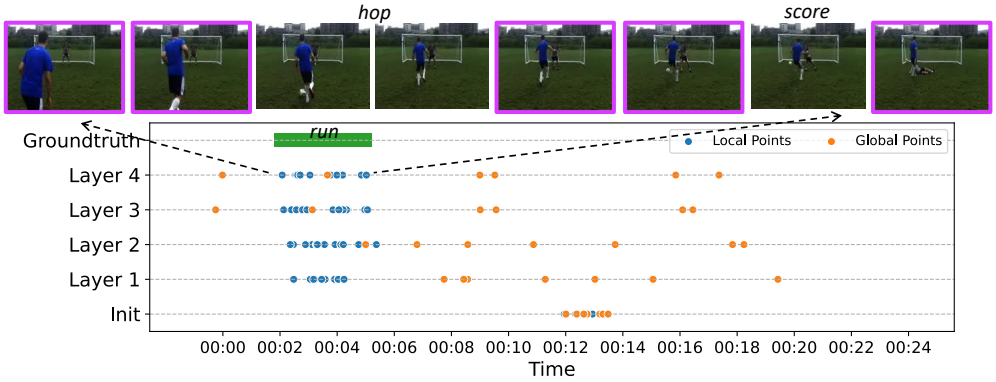

Figure 5: Visualization of **learnt query points** and the corresponding action groundtruth on a sample from MultiTHUMOS. The query points capture the action boundaries with extreme local points and recognize the distinctive action characteristic with interior local points.

higher performance than direct interpolation at query point location, proving that it's beneficial to consider local context with dynamic parameters in action detection.

Study on instance-level dynamic mixing. The effect of frame-wise and channel-wise mixing is studied in Tab. 2f. We first ablate with the single application of each mixing, i.e. frame-only and channel-only. Results show that the performance degrades more without frame mixing, demonstrating the importance of frame mixing over channel mixing in multi-label TAD. Furthermore, we explore different combinations of frame and channel mixing with two cascaded alternatives, i.e. frame mixing $\rightarrow$ channel mixing and channel mixing $\rightarrow$ frame mixing. Tab. 2f shows that compared with frame-only and channel-only performances respectively, cascaded designs backfire at the performance with a decrease of 1% avg-mAP at frame $\rightarrow$ channel and 3% avg-mAP at channel $\rightarrow$ frame. To tackle action detection in video domain, intra-proposal channel mixing does not help with subsequent frame mixing and vice versa.

4.4 Query points Visualization

Fig. 5 illustrates the learnt query points and the corresponding action target on a sample from MultiTHUMOS. Partial mix-max transformation divides the query points into confined local points (blue) and open-ended global points (orange). Local points are learned to acknowledge action boundaries with two extreme points and semantic key frames with interior points. Global points sparsely distribute all over the video for open-ended context integration. Specifically, we observe that the interior local points effectively capture the most distinguishing characteristic of target action *"Run"*, by capturing rapid forward movement of *both* legs and neglecting frames of similar yet disruptive movements: hopping and scoring. Moreover, the iterative refining process of query points are depicted from initial positions to the final positions in the last decoder layer. Query points are learnt to start around the median position in the window. After the first decoder layer, the query points are already able to coarsely locate the action target. Then, these query points automatically adjust their locations for finer localization throughout the first three layers and stabilize at the fourth layer.

5 Conclusion

In this paper, we have studied the complex multi-label TAD that aims to detect all actions from a multi-label untrimmed video. We formulate this problem as sparse detection task and extend sparse query-based action detect framework from traditional TAD. Faced with the challenge of concurrent instances and complex action relations, we introduce a set of learnable query points to effectively capture action boundaries as well as characterize action semantics for fine-grained action features. Moreover, to facilitate the decoding process, we propose the Multi-level Interactive Module that integrates action semantics at point-level and instance-level, with dynamic kernels based on query vector. Finally, PointTAD streamlines end-to-end trainable architecture using only RGB inputs for easy deployment. Our proposed method surpasses all previous methods by a large margin under detection-mAP and achieves promising results under segmentation-mAP on two popular multi-label TAD benchmarks.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In *CVPR*, pages 6154–6162. Computer Vision Foundation / IEEE Computer Society, 2018.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV (1)*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2020.

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

[4] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster R-CNN architecture for temporal action localization. In *CVPR*, pages 1130–1139. Computer Vision Foundation / IEEE Computer Society, 2018.

[5] Guo Chen, Yin-Dong Zheng, Limin Wang, and Tong Lu. DCAN: improving temporal action detection via dual context aggregation. *CoRR*, abs/2112.03612, 2021.

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773. IEEE Computer Society, 2017.

[7] Rui Dai, Srijan Das, Kumara Kahatapitiya, Michael S. Ryoo, and François Brémond. MS-TCT: multi-scale temporal convtransformer for action detection. *CoRR*, abs/2112.03902, 2021.

[8] Rui Dai, Srijan Das, Luca Minciullo, Lorenzo Garattoni, Gianpiero Francesca, and François Brémond. PDAN: pyramid dilated attention network for action detection. In *WACV*, pages 2969–2978. IEEE, 2021.

[9] Jialin Gao, Zhixiang Shi, Guanshuo Wang, Jiani Li, Yufeng Yuan, Shiming Ge, and Xi Zhou. Accurate temporal action proposal generation with relation-aware pyramid network. In *AAAI*, pages 10810–10817. AAAI Press, 2020.

[10] Ziteng Gao, Limin Wang, Bing Han, and Sheng Guo. Adamixer: A fast-converging query-based object detector. *CoRR*, abs/2203.16507, 2022.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):386–397, 2020.

[12] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970. IEEE Computer Society, 2015.

[13] Haroon Idrees, Amir Roshan Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The THUMOS challenge on action recognition for videos "in the wild". *Comput. Vis. Image Underst.*, 155:1–23, 2017.

[14] Kumara Kahatapitiya and Michael S. Ryoo. Coarse-fine networks for temporal activity detection in videos. In *CVPR*, pages 8385–8394. Computer Vision Foundation / IEEE, 2021.

[15] Chuming Lin, Chengming Xu, Donghao Luo, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu. Learning salient boundary feature for anchor-free temporal action localization. In *CVPR*, pages 3320–3329. Computer Vision Foundation / IEEE, 2021.

[16] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 936–944. IEEE Computer Society, 2017.

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

[18] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV (5)*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.

[19] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. BSN: boundary sensitive network for temporal action proposal generation. In *ECCV (4)*, volume 11208 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2018.

[20] Xiaolong Liu, Song Bai, and Xiang Bai. An empirical study of end-to-end temporal action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[21] Xiaolong Liu, Qimeng Wang, Yao Hu, Xu Tang, Song Bai, and Xiang Bai. End-to-end temporal action detection with transformer. *CoRR*, abs/2106.10271, 2021.

[22] Pascal Mettes, Jan C. van Gemert, Spencer Cappallo, Thomas Mensink, and Cees G. M. Snoek. Bag-of-fragments: Selecting and encoding video fragments for event detection and recounting. In *ICMR*, pages 427–434. ACM, 2015.

[23] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, pages 1817–1824. IEEE Computer Society, 2013.

[24] A. J. Piergiovanni and Michael S. Ryoo. Learning latent super-events to detect multiple activities in videos. In *CVPR*, pages 5304–5313. Computer Vision Foundation / IEEE Computer Society, 2018.

[25] A. J. Piergiovanni and Michael S. Ryoo. Temporal gaussian mixture layer for videos. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5152–5161. PMLR, 2019.

[26] Zhiwu Qing, Haisheng Su, Weihao Gan, Dongliang Wang, Wei Wu, Xiang Wang, Yu Qiao, Junjie Yan, Changxin Gao, and Nong Sang. Temporal context aggregation network for temporal action proposal refinement. In *CVPR*, pages 485–494. Computer Vision Foundation / IEEE, 2021.

[27] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[28] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV (1)*, volume 9905 of *Lecture Notes in Computer Science*, pages 510–526. Springer, 2016.

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

[30] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse R-CNN: end-to-end object detection with learnable proposals. In *CVPR*, pages 14454–14463. Computer Vision Foundation / IEEE, 2021.

[31] Jing Tan, Jiaqi Tang, Limin Wang, and Gangshan Wu. Relaxed transformer decoders for direct action proposal generation. In *ICCV*, pages 13506–13515. IEEE, 2021.

[32] Praveen Tirupattur, Kevin Duarte, Yogesh S. Rawat, and Mubarak Shah. Modeling multi-label action dependencies for temporal action localization. In *CVPR*, pages 1460–1470. Computer Vision Foundation / IEEE, 2021.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[34] Jiannan Wu, Peize Sun, Shoufa Chen, Jiewen Yang, Zihao Qi, Lan Ma, and Ping Luo. Towards high-quality temporal action detection with sparse proposals. *CoRR*, abs/2109.08847, 2021.

[35] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: region convolutional 3d network for temporal activity detection. In *ICCV*, pages 5794–5803. IEEE Computer Society, 2017.

[36] Mengmeng Xu, Chen Zhao, David S. Rojas, Ali K. Thabet, and Bernard Ghanem. G-TAD: sub-graph localization for temporal action detection. In *CVPR*, pages 10153–10162. Computer Vision Foundation / IEEE, 2020.

[37] Min Yang, Guo Chen, Yin-Dong Zheng, Tong Lu, and Limin Wang. BasicTAD: an astounding rgb-only baseline for temporal action detection. *CoRR*, abs/2205.02717, 2022.

[38] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *ICCV*, pages 9656–9665. IEEE, 2019.

[39] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *Int. J. Comput. Vis.*, 126(2-4):375–389, 2018.

[40] Runhao Zeng, Wenbing Huang, Chuang Gan, Mingkui Tan, Yu Rong, Peilin Zhao, and Junzhou Huang. Graph convolutional networks for temporal action localization. In *ICCV*, pages 7093–7102. IEEE, 2019.

[41] Hang Zhao, Antonio Torralba, Lorenzo Torresani, and Zhicheng Yan. HACS: human action clips and segments dataset for recognition and temporal localization. In *ICCV*, pages 8667–8677. IEEE, 2019.

[42] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. *Int. J. Comput. Vis.*, 128(1):74–95, 2020.

[43] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets V2: more deformable, better results. In *CVPR*, pages 9308–9316. Computer Vision Foundation / IEEE, 2019.

[44] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *ICLR*. OpenReview.net, 2021.

[45] Zixin Zhu, Wei Tang, Le Wang, Nanning Zheng, and Gang Hua. Enriching local and global contexts for temporal action localization. In *ICCV*, pages 13496–13505. IEEE, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the supplementary material.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] No potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] No theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A] No theoretical results.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code and data are proprietary.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Sec. 4.1 and the supplementary material
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See the supplementary material
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sec. 4.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Sec. 4.1
   (b) Did you mention the license of the assets? [Yes] See the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See the supplementary material for details.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] The datasets we use are open datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]