OpenTAD: A Unified Framework and Comprehensive Study of Temporal Action Detection

Supplementary Material

A. Related Works

A.1. TAD Frameworks

The absence of a standardized framework for temporal action detection has historically led researchers to either develop methods from scratch or build upon specific open-source implementations. While some methods have provided partial foundations, they remain limited in scope. For example, GTAD [63] was implemented on top of BMN [31] and was later adopted by subsequent works such as TSI [34] and BC-GNN [2]. However, these methods are tailored to specific designs and lack the flexibility to accommodate broader innovations. Additionally, adapting a feature-based method (e.g., [70]) to an end-to-end training paradigm (e.g., [71]) often requires extensive and tedious code modifications, such as restructuring feature extraction and training components. This lack of interoperability hinders efficient experimentation and fair benchmarking.

Although several works have proposed distinct "frameworks" for TAD, they primarily focus on specific paradigms rather than providing a truly unified solution. Examples include the *one-stage framework* in TriDet [48], the *multi-level cross-scale framework* in VSGN [70], the *end-to-end detection framework* in TadTR [39], and the *efficient end-to-end framework* in AdaTAD [37]. These frameworks are designed independently, making cross-method comparison and integration challenging. Some methods, such as BMN [31] and TSI [34], describe their approaches as "unified frameworks," but they primarily refer to the joint training of multiple components within a single model rather than a framework that accommodates diverse methodologies.

To address these limitations, OpenTAD introduces the first truly unified framework for TAD, integrating a broad range of approaches—including one-stage, two-stage, DETR-based, and end-to-end methods—within a single, cohesive implementation. By supporting both feature-based and end-to-end learning paradigms, OpenTAD simplifies the development, adaptation, and evaluation of new methods, ensuring a standardized and extensible platform for future research.

A.2. TAD Surveys

Several survey papers on temporal action detection provide comprehensive overviews of various methods [18, 55, 56]. These reviews categorize TAD approaches based on the original modularization described in each paper and directly compare reported performance across different methods. However, they do not account for inconsistencies in experimental configurations, such as data resolution, preprocessing, and post-processing, which can lead to unfair comparisons and obscure the true effectiveness of individual techniques.

In contrast, OpenTAD offers a unified framework that systematically re-modularizes and reimplements existing methods to ensure consistency across different architectures and training pipelines. This standardization enables faithful comparison and rigorous analysis, providing deeper insights into the impact of specific design choices. Unlike previous survey papers, which evaluate methods based on heterogeneous configurations, OpenTAD facilitates direct, controlled comparisons under a unified setting. Furthermore, while prior reviews independently classify TAD methods into categories, OpenTAD integrates diverse approaches—including one-stage, two-stage, DETR-based, and end-to-end methods—into a single, cohesive implementation, enabling a more comprehensive and adaptable benchmarking platform.

B. Evaluation Protocol and Implementation Details

Evaluation Protocol. Although mean Average Precision (mAP) is the standard evaluation metric for TAD, inconsistencies in evaluation code, ground-truth annotations, and tIoU thresholds across previous methods have led to difficulties in fair comparisons. In the OpenTAD framework, we standardize the evaluation protocol across datasets and methods, ensuring consistency. We report the mean and standard deviation of performance metrics over five different random seeds to account for variability in training.

Implementation Details. OpenTAD is implemented using PyTorch 2.0 and runs on a single NVIDIA A100 GPU. We adhere to each method's original hyperparameter settings, including learning rate, number of training epochs, and other relevant parameters. Unless otherwise stated, we use pre-extracted TSP features on ActivityNet and I3D features on THUMOS for our ablation studies. Batch sizes and optimizer configurations are kept consistent with each method's original implementation.

C. Introduction and Categorization of Implemented Methods

OpenTAD implements 16 representative temporal action detection methods. To align each paper's original design with OpenTAD's modular framework, we categorize the methods based on their sub-components, such as neck, dense

Table 7. **Re-implemented results of different methods in OpenTAD** in terms of mAP (%). N/A means not provided in the paper or released code. For THUMOS-14, previous papers usually reported 5 numbers from mAP at tIoU={0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7}, and some reported average mAP values, which mean differently across papers. Here, we standardize average mAP as the average of the mAP values at tIoU={0.3, 0.4, 0.5, 0.6, 0.7}, compute this number based on the reported mAP at these 5 tIoUs. For ActivityNet, we compute the average mAP at tIoU={0.5:0.95:0.05}.

Method	Backbone	THUMOS-14				ActivityNet-v1.3	
		Original tIoU=0.5	OpenTAD tIoU=0.5	Original Average	OpenTAD Average	Original Average	OpenTAD Average
ActionFormer [68]	I3D | TSP	71.00	73.16	66.84	68.44	36.60	37.07
TemporalMaxer [52]	I3D | TSP	71.80	71.66	67.70	68.33	N/A	N/A
TriDet [48]	I3D | TSP	72.90	73.41	69.28	69.60	36.80	36.51
CausalTAD [36]	I3D | TSP	73.57	73.57	69.75	69.75	37.46	37.46
DyFADet [64]	VideoMAEv2-G | TSP	73.70	76.32	70.50	71.70	38.50	38.62
VideoMambaSuite [9]	InternVideo2	76.90	77.17	72.72	73.24	42.02	42.80
BMN [31]	TSN	38.80	47.56	38.48	46.19	33.85	34.21
GTAD [63]	TSN	43.04	48.50	41.41	46.49	34.09	34.18
TSI [34]	TSN | TSP	42.60	46.14	42.26	44.75	35.24	35.36
VSGN [70]	TSN | TSP	45.52	49.37	43.37	47.25	35.94	36.89
TadTR [39]	I3D | TSP	60.10	59.00	56.68	56.23	36.75	N/A
AFSD [27]	I3D | TSP	55.50	60.16	52.00	55.96	34.40	36.10
E2E-TAD [38]	I3D	47.00	59.00	45.08	56.23	N/A	N/A
ETAD [35]	R(2+1)D	56.17	58.23	54.66	55.56	38.25	38.76
Re^{2}TAL [71]	Re^{2}SlowFast-101	64.90	74.27	61.52	70.19	37.01	37.55
AdaTAD [37]	VideoMAEv2-G	80.90	81.24	76.88	77.07	41.93	41.85

Table 8. **Detection performance on all 9 supported datasetsusing ActionFormer [68],** measured by average mAP(%). N/A. means not implemented in ActionFormer's original codebase.

Method	THUMOS-14	ActivityNet-v1.3	EPIC-Kitchens	Ego4D-MQ	HACS
Original	66.83	36.56	21.88 | 23.51	23.29	N/A
OpenTAD	68.44	37.07	22.33 | 24.93	25.57	37.71
Method	**Multi-THUMOS**	**Charades**	**FineAction**	**EPIC-Sounds**	
Original	N/A	N/A	N/A	N/A	
OpenTAD	39.18	19.39	19.62	13.89	

head, RoI, loss functions, and more. Table 14 provides a detailed mapping of these components. We classify the methods into four categories: one-stage, two-stage, DETR-based, and end-to-end methods. From this table, we can clearly see that the primary difference between one-stage and two-stage methods is that the latter includes an additional step involving RoI extraction to further refine the candidate actions.

For reproducibility, we re-implemented all 16 methods on the ActivityNet and THUMOS datasets, with results reported in Table 7. Our results closely match the original papers, and in some cases, such as BMN, we achieve significantly better performance. Additionally, OpenTAD also supports 9 widely used temporal action detection datasets: ActivityNet-v1.3 [4], THUMOS-14 [20], EPIC-Kitchens 100 [10], Ego4D-MQ [16], HACS [72], Multi-THUMOS [66], Charades [49], FineAction [40], and EPIC-Sounds [19]. We benchmarked ActionFormer as the base model across all datasets, with results presented in Table 8.

D. Benchmark Results

In this section, we present the benchmark results for all implemented methods within the OpenTAD framework. For ActivityNet, we use TSP features with standardized evaluation annotations across all methods. For THUMOS, we adopt two-stream I3D features while ensuring consistency in evaluation annotations. The results are reported in Table 9 and Table 10.

From our benchmarks, we observe that one-stage detection methods have emerged as the preferred choice for both datasets. On THUMOS, two-stage methods such as GTAD and BMN, which rely on external classifiers for action classification, generally underperform compared to recent one-stage methods. However, on ActivityNet, where both one-stage and two-stage methods leverage external video-level classification results, two-stage methods with cascaded proposal refinements still achieve slightly better performance than their one-stage counterparts.

Table 9. **Benchmarking results on ActivityNet-v1.3 with TSP feature** in terms of mAP (%).

Method	#Param. (M)	0.5	0.75	0.95	Avg. mAP
TSI [34]	4.54	52.44	35.57	9.80	35.36
TemporalMaxer [52]	1.38	54.59	37.13	7.11	36.03
AFSD [27]	13.41	54.44	36.72	8.69	36.10
GTAD [63]	5.58	52.33	37.58	8.42	36.20
BMN [31]	2.80	52.90	37.30	9.67	36.40
TriDet [48]	12.81	54.84	37.46	7.98	36.51
VSGN [70]	6.50	54.80	37.35	9.80	36.89
ActionFormer [68]	6.94	55.08	38.27	8.91	37.07
VideoMambaSuite [9]	4.30	55.61	38.49	9.18	37.45
CausalTAD [36]	12.75	55.62	38.51	9.40	37.46
ETAD [35]	8.52	54.91	38.98	9.09	37.73

Table 10. **Benchmarking results on THUMOS14 with I3D feature** in terms of mAP (%).

Method	#Param.(M)	0.3	0.4	0.5	0.6	0.7	Avg. mAP
TSI [34]	4.84	62.56	57.00	50.22	40.18	30.17	48.03
GTAD [63]	6.14	64.35	59.07	51.76	42.65	31.66	49.70
BMN [31]	3.10	64.99	60.70	54.54	44.11	34.16	51.80
VSGN [70]	8.37	68.25	62.46	54.99	44.07	32.36	52.43
ETAD [35]	5.37	67.74	64.22	58.23	49.19	38.41	55.56
AFSD [27]	14.24	73.20	68.45	60.16	46.74	31.24	55.96
TadTR [39]	8.66	71.90	67.29	59.00	48.34	34.61	56.23
TemporalMaxer [52]	7.12	83.17	79.09	71.66	61.72	46.00	68.33
ActionFormer [68]	29.25	83.78	80.06	73.16	60.46	44.72	68.44
VideoMambaSuite [9]	18.57	84.33	80.60	74.19	61.99	46.71	69.57
TriDet [48]	15.99	84.46	81.05	73.41	62.58	46.51	69.60
CausalTAD [36]	52.11	84.43	80.75	73.57	62.70	47.33	69.75

E. Supplementary Experiments

E.1. Ablation Study on Neck Design in the THUMOS Dataset

To further examine the impact of neck design, we conduct an ablation study on the ActivityNet dataset, with results presented in Table 11. These findings align with those reported in Table 2 of the main paper. Our experiments show that LSTM achieves performance comparable to or even better than the SSM module in BMN and GTAD. Furthermore, by integrating both designs, we obtain the best overall performance across all four evaluated methods.

E.2. Ablation Study on Loss Functions

In this section, we analyze commonly used loss functions in TAD methods and conduct an ablation study on the effectiveness of the actionness loss proposed for the one-stage method, ActionFormer.

Action Category Losses. These losses are typically classification-based, as action categories are discrete. TAD methods can employ either binary or multi-class classification losses, depending on the objective—distinguishing action from non-action segments or classifying specific action categories. Methods that incorporate external class labels during post-processing (e.g., G-TAD [63] and ActionFormer [68] on ActivityNet) typically use binary classification for all category losses within the network. Conversely, methods without such external annotations use multi-class classification for final category predictions while still employing binary classification for intermediate stages. Common classification losses include focal loss [32] for binary classification, as used in VSGN [70], and cross-entropy loss for multi-class classification, as seen in ActionFormer [68]. Some approaches, such as BSN [30] and BMN [31], instead regress action confidence based on the IoU between proposals and ground-truth actions, treating the problem as a regression task rather than a strict binary classification.

Action Boundary Losses. These losses aim to refine the boundaries between predicted and ground-truth action segments, as precise boundary localization is crucial for TAD performance. Various methods improve boundary regression accuracy by directly predicting the distances to start and end locations (e.g., ActionFormer [68], VSGN [70]) or by regressing offsets relative to predefined anchors, as in PGCN [33].

Effect of Actionness Loss. Beyond classification and boundary regression losses, the two-stage method BMN introduces a Temporal Evaluation Module, which classifies each timestep based on actionness/startness/endness to enhance boundary learning. However, this design has been

Table 11. **Analysis of the neck design choices**, measured by average mAP(%) on ActivityNet-v1.3. The **4 macro-block regions** mean the following respectively. **Top region**: macro blocks with their original sequential modules are adopted as a whole; **Transformer Block**: self-attention modules in Transformer blocks are replaced with different sequential modules; **Mamba Block**: SSM modules in Mamba blocks are replaced with different sequential modules; **Bottom region**: a combination of two blocks. Note that in Row 1, BMN and GTAD directly use identity mapping since they don't downscale temporally. TSP [1] features are used.

Neck		Method			
Macro Block	**Sequential Module**	**ActionFormer**	**TriDet**	**BMN**	**GTAD**
Convolution Block	Convolution	36.88±0.03	36.87±0.02	36.44±0.05	36.36±0.09
GCN Block	Graph convolution	37.03±0.04	37.00±0.05	36.41±0.02	36.24±0.04
Transformer Block	Self-attention	37.00±0.05	36.93±0.09	**36.50**±0.03	36.31±0.06
Mamba Block	SSM	**37.40**±0.03	**37.33**±0.07	36.36±0.05	**36.36**±0.06
Transformer Block	Convolution	36.99±0.02	36.93±0.09	36.40±0.07	36.31±0.07
	Graph convolution	37.15±0.03	36.98±0.07	36.41±0.02	36.13±0.08
	Self-attention	37.00±0.05	36.93±0.09	36.50±0.03	36.31±0.06
	LSTM	37.18±0.04	36.95±0.07	**36.87**±0.09	**36.47**±0.12
	SSM	**37.40**±0.04	**37.33**±0.07	36.58±0.23	36.21±0.08
Mamba Block	Convolution	37.07±0.03	36.97±0.07	36.40±0.05	36.17±0.08
	Graph convolution	37.13±0.07	37.06±0.04	36.33±0.05	36.16±0.09
	Self-attention	36.21±0.22	36.33±0.14	36.35±0.07	36.12±0.04
	LSTM	36.95±0.11	36.57±0.07	**36.42**±0.08	**36.23**±0.05
	SSM	**37.40**±0.03	**37.33**±0.07	36.36±0.05	36.36±0.06
Mamba + Transf.	SSM + Self-attention	37.48±0.03	37.35±0.08	36.74±0.27	36.25±0.02
Mamba + Transf.	**SSM + LSTM**	**37.50**±0.07	**37.41**±0.05	**36.94**±0.05	**36.52**±0.08

Table 12. **Analysis of the RoI extraction design choices**, measured by average mAP (%) on two datasets. N/A means not applicable to the model.

	ActivityNet-v1.3			THUMOS-14		
RoI Extraction	**BMN**	**GTAD**	**VSGN**	**BMN**	**GTAD**	**VSGN**
Keypoint Sample [70]	34.29±0.07	32.68±0.05	36.71±0.13	43.36±0.24	45.34±0.59	52.42±0.31
RoI Align [17]	36.06±0.05	35.99±0.13	**36.74**±0.02	49.48±0.67	49.40±0.46	**52.53**±0.41
SGAlign [63]	36.28±0.12	36.24±0.04	36.66±0.14	49.00±0.45	50.36±0.29	52.38±0.64
Boundary Match [31]	**36.44**±0.07	**36.34**±0.07	N/A	**51.40**±0.58	**50.48**±0.35	N/A

Table 13. **Analysis of the loss design choices** on ActionFormer.

Category	**Boundary**	**Actionness**	**ActivityNet-v1.3**	**THUMOS-14**
✓	✓		37.00±0.05	67.93±0.19
✓	✓	✓	37.19±0.05	67.11±0.30

primarily used in two-stage methods. To evaluate its effectiveness in a one-stage setting, we integrate a temporal evaluation head into ActionFormer and apply actionness loss supervision. The results, shown in Table 13, indicate that while actionness loss provides a minor improvement on ActivityNet, it negatively impacts performance on THUMOS. Given its marginal benefit on ActivityNet and its detrimental effect on THUMOS, we exclude actionness loss from our final design.

F. Limitations

OpenTAD provides a unified framework for implementing and benchmarking various temporal action detection methods, supporting eight datasets. However, it currently focuses exclusively on fully supervised TAD and does not yet support weakly-supervised or open-vocabulary TAD.

Another limitation lies in the scale of existing TAD datasets. Current datasets are relatively small, leading to high variance in experimental results across different random seeds. This variability poses challenges for ensuring training stability and reproducibility. Addressing the scalability of TAD datasets and improving the robustness of training pipelines remains an open research direction that warrants further exploration.

Table 14. **Components mapping from each method to OpenTAD.** Cls. and Reg. denote classification loss and regression loss respectively.

	Method	Backbone	Neck	RoI extraction	Losses
One-Stage	ActionFormer [68]	I3D, TSP, SlowFast	Multi-scale Transformer		Cls. (Focal), Reg. (DIoU)
	TriDet [48]	I3D, TSP, R(2+1)D, SlowFast	Scalable-Granularity Perception		Cls. (IoU weighted Focal) Reg. (DIoU)
	TemporalMaxer [52]	I3D, SlowFast	MaxPool1d		Cls. (Focal), Reg. (DIoU) w/ SimOTA
	VideoMambaSuite [9]	InternVideo2-6B	Mamba		Cls. (Focal), Reg. (DIoU)
	DyFADet [64]	I3D, TSP	Dynamic Feature Aggregation		Cls. (Focal), Reg. (DIoU)
	CausalTAD [36]	I3D, TSP	Mamba + Transformer		Cls. (Focal), Reg. (DIoU)
Two-Stage	BMN [31]	TSN	Conv1D	Boundary Matching	Cls. (weighted BLR), Reg. (L_2), TEM loss (weighted BCE))
	GTAD [63]	TSN	GCNeXt	SGAlign	Cls. (weighted BCE), Reg. (L_2), Node Cls. (weighted BCE))
	TSI [34]	TSN, I3D, TSP	Temporal Boundary Detector	IoU Map Regressor	Cls. (Scale-Invariant loss), Reg. (L_2), TBD loss (BLR)
	VSGN [70]	TSN, I3D, R(2+1)D	xGPN	Boundary Sampling	Cls. (Focal), Reg. (GIoU), Boundary Adj. (GIoU), Supp. Scores (weighted BLR)
DETR-Based	TadTR [39]	TSN, I3D	Deformable Transformer	Action Queries, RoIAlign	Cls. (CE), Reg. (IoU, L_1), Act. (L_1), w/ bipartite matching
End-to-End	AFSD [27]	I3D	FPN	Boundary pooling	Cls. (Focal), Reg. (IoU, L_1)
	E2E-TAD [38]	TSN, TSM, I3D, SlowFast	Same as G-TAD, AFSD, TadTR		
	ETAD [35]	TSM, R(2+1)D	LSTM	SGAlign	Cls. (CE), Reg. (smooth-L_1)
	Re^{2}TAL [71]	Re^{2}Vswin, Re^{2}Slowfast	Same as ActionFormer, VSGN		
	AdaTAD [37]	VideoMAE, SlowFast	Same as ActionFormer		