SPATAD: PRESERVING SPATIAL STRUCTURE IN TEMPORAL ACTION DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Temporal action detection (TAD) requires modeling how visual evidence evolves over time. Most detectors, however, pool each frame or snippet into a single feature vector before temporal modeling. This is effective when action evidence can be summarized globally, but can become restrictive when an action is defined by where evidence occurs and how it moves, such as the path a vehicle takes through an intersection, or when concurrent actions rely on different regions of the frame. We therefore ask how far into a temporal action detector explicit spatial structure should be kept. We introduce **SpaTAD**, a TAD model that keeps spatial tokens separate until action predictions are computed. Rather than pooling each frame into a single vector, SpaTAD maintains a causal, location-bound recurrent state for each spatial token, predicts actions at individual tokens, and only then aggregates them across space into frame-level action predictions. On MultiTHUMOS, Charades, ATARS, and ROAD-Waymo, SpaTAD improves over the strongest matched baseline by an average of 11.6 Seg-mAP and 6.5 Det-mAP. Pooling before class prediction hurts performance on all four benchmarks, while collapsing spatial structure before temporal modeling causes an additional 5.07 Det-mAP and 10.41 Seg-mAP drop on ATARS. These results suggest that dense temporal modeling alone is not always sufficient: preserving spatial organization through class prediction can be beneficial, and location-bound temporal memory appears especially useful when actions depend on how evidence moves over time.

1 INTRODUCTION

Temporal action detection (TAD) recognizes actions in untrimmed videos and localizes their temporal extent. A central challenge is to capture how visual evidence evolves over time, and much of the progress in TAD has come from better temporal modeling. Most pipelines, however, apply temporal modeling to a sequence of compact features: the spatial representation of each frame or snippet is compressed into a single feature vector before temporal processing (Dai et al., 2022; Zhang et al., 2022; Tan et al., 2022; Sinha et al., 2026). This design is effective when action evidence can be summarized globally, but can become restrictive when the discriminative signal is spatially localized. Recent state-space adapters keep a coarse spatial grid while modeling time inside the video backbone (Qiu & Yanai, 2026a;b), but their detection heads still operate on spatially pooled features. In both designs, spatial structure is removed before class prediction, the step that maps features to per-class scores, which raises a representation question: *how far into a temporal action detector should explicit spatial structure be kept?*

The question matters most when an action is defined by spatial-temporal changes rather than by appearance, as in the ATARS example shown in Fig. 1(a). Two vehicles enter an intersection from opposite directions and both leave toward the west. After the turn, they can look alike in a single frame: they drive in the same direction on the same road. They nevertheless belong to different classes, north-to-west and south-to-west, because their motion histories differ. The scene also contains multiple vehicles on different paths, illustrating a broader challenge in dense multi-label video: concurrent actions may depend on different frame regions. Spatial pooling can create two different bottlenecks. If frames are pooled before temporal modeling (Fig. 1b), vehicle locations are mixed into one feature, making it hard to preserve which vehicle came from where. If temporally updated tokens are pooled before class prediction, evidence that occupies only a few tokens, such as a small vehicle in an aerial view, is combined with the rest of the frame into one summary, so every class

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

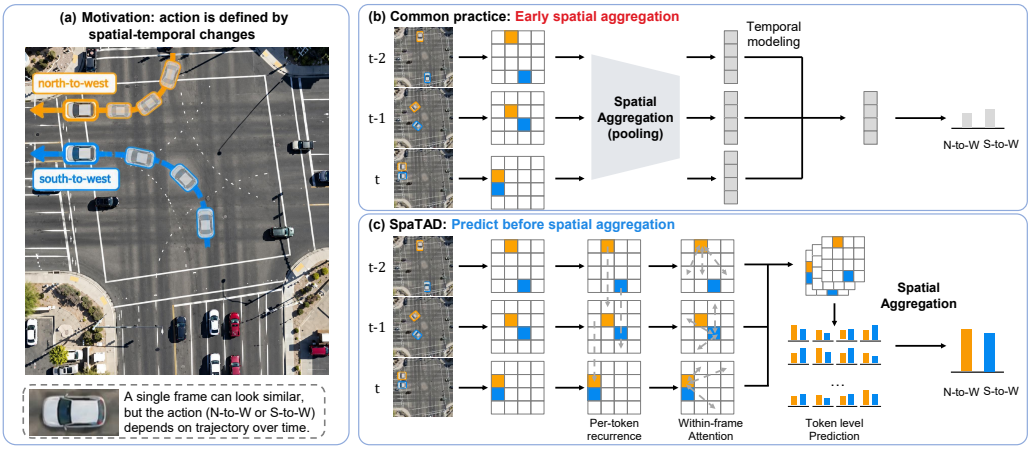

Figure 1: **How far should spatial structure be kept?** (a) Illustration on an ATARS intersection: two vehicles from opposite directions both turn west. After the turn they look alike, but they belong to different classes (north-to-west, south-to-west), and only their paths tell them apart. (b) A common TAD design compresses each frame before temporal modeling, so temporal modeling and class prediction operate on a shared representation. (c) SpaTAD keeps spatial tokens through temporal modeling, computes class scores at individual tokens, and aggregates across space only afterward.

is predicted from the same spatial mixture. These two pooling stages therefore constrain spatial evidence in different ways.

We investigate this question with **SpaTAD**, a TAD model that keeps spatial tokens separate until action predictions are computed (Fig. 1c). SpaTAD produces frame-level action predictions causally: the prediction at frame t is computed from the current and past frames. During temporal modeling, SpaTAD maintains a recurrent state for each spatial token instead of a single state for the whole frame. Each state carries information across frames while remaining associated with its own token location, forming a location-bound memory; within-frame attention exchanges information across locations while staying within a frame. Alternating the two over several layers lets evidence move across both space and time: the token at a vehicle's current position can read information propagated from locations the vehicle has passed through. In the example above, motion history can remain spatially organized rather than collapsed into one frame-level feature.

Second, SpaTAD keeps spatial tokens separate through class prediction. Instead of pooling tokens into one shared feature and then predicting classes, SpaTAD predicts action at individual tokens and aggregates them across space only afterward, using soft multiple-instance pooling (Dietterich et al., 1997; Ilse et al., 2018). In the example above, north-to-west and south-to-west can each be scored at tokens along the corresponding vehicle's path before the frame-level decision is made. The same token-level predictions also provide coarse class-specific spatial evidence without spatial annotation. Together, these two choices follow a simple principle: keep tokens separate through temporal modeling and class prediction, and aggregate across space only after action-specific evidence has been extracted.

We evaluate SpaTAD on MultiTHUMOS (Yeung et al., 2018), Charades (Sigurdsson et al., 2016), ATARS (Chen et al., 2025), and ROAD-Waymo (Khan et al., 2024), where it outperforms the strongest matched baseline on all four benchmarks (§4.2). To test our central question directly, we move spatial aggregation earlier while keeping the remaining configuration fixed. *Pooled prediction* preserves per-token temporal states but aggregates them before class prediction, whereas *global temporal state* aggregates before temporal modeling and therefore also removes location-bound memory. Pooling before class prediction hurts on every benchmark, showing that retaining spatial structure through temporal modeling alone is insufficient. The additional cost of removing location-bound memory is dataset-dependent and largest on ATARS, where the global-state control loses a further 5.07 Det-mAP and 10.41 Seg-mAP. Together, these controls show that spatial structure should remain explicit through class prediction, while spatially indexed temporal mem-

ory matters most when actions depend on where evidence moves over time. We further examine bidirectional modeling, token-level spatial evidence, and persistent-state inference.

Our contributions are:

- We study **how far explicit spatial structure should be kept** in temporal action detection, comparing spatial aggregation before temporal modeling, before class prediction, and after class prediction.
- We introduce **SpaTAD**, a TAD model that produces causal frame-level action predictions while preserving spatial tokens until prediction. SpaTAD maintains a recurrent state for each spatial token, predicts actions at individual tokens before spatial aggregation, and predicts temporal segments from the resulting dense predictions.
- Through controlled comparisons on four benchmarks, we show that pooling before class prediction consistently hurts performance, while preserving spatial structure during temporal modeling provides additional benefits when actions depend on location-bound motion history. SpaTAD outperforms the strongest baseline on all four benchmarks, with Det-mAP gains ranging from $+0.39$ to $+17.32$ and Seg-mAP gains from $+4.18$ to $+28.54$.

2 Related Work

Temporal action detection. Dense multi-label detectors predict class scores for every frame. They capture temporal structure with learned temporal filters (Piergiovanni & Ryoo, 2018; 2019), class dependencies (Tirupattur et al., 2021), multiple temporal resolutions (Kahatapitiya & Ryoo, 2021), dilated attention (Dai et al., 2021), convolution–transformer hybrids (Dai et al., 2022), and other dense designs (Sardari et al., 2026). Instance-level detectors predict segments directly, using anchor-free heads (Zhang et al., 2022; Shi et al., 2023b), learnable query points (Tan et al., 2022), or DETR-style decoders (Zhu et al., 2024), and can be trained end to end with large backbones (Liu et al., 2024). State-space layers have also been used for long-range temporal modeling in TAD (Lu et al., 2026; Sinha et al., 2026). Across these designs, many detectors operate on one compact feature vector per frame or snippet, so explicit spatial structure has already been compressed before detector-level temporal modeling. SpaTAD instead keeps this structure explicit through temporal modeling and until class scores are computed, and uses this design to study how far into a detector it should be kept.

Online and streaming video models. Online action detection labels each incoming frame using current and past observations (Xu et al., 2019; 2021; Zhao & Krähenbühl, 2022; An et al., 2023), while streaming video models learn causal representations under the same constraint (Yan et al., 2025). Long histories are commonly summarized using recurrent states (Xu et al., 2019; An et al., 2023), compressed memories (Xu et al., 2021), constant-cost attention updates (Zhao & Krähenbühl, 2022), or a small set of memory tokens (Ryoo et al., 2023). In many such models, temporal memory is represented globally or by tokens not explicitly tied to image locations. SpaTAD likewise produces frame-level action prediction causally, while maintaining one recurrent state per spatial token, so the history keeps its spatial organization while the total state size stays fixed as the stream grows.

Spatial structure in temporal modeling. Recurrent models have long preserved spatial structure in their state: ConvLSTM and ConvGRU maintain spatial feature maps across recurrent updates (Shi et al., 2015; Ballas et al., 2016), and TrajGRU learns location-dependent recurrent connections to better accommodate motion (Shi et al., 2017). Video transformers and state-space backbones likewise model spatio-temporal tokens jointly, for example through motion-aware attention (Patrick et al., 2021) or state-space processing of video tokens (Li et al., 2024), typically on clips processed offline. Within TAD, ESTF-SSM (Qiu & Yanai, 2026b) and its extension DST-Mamba (Qiu & Yanai, 2026a) insert state-space adapters into a frozen video backbone. DST-Mamba's temporal branch pools each frame to a coarse spatial grid (4×4 by default) and applies a bidirectional temporal scan at every grid cell; collapsing this grid to a single cell noticeably lowers detection accuracy. However, its detection head still operates on spatially pooled temporal features. SpaTAD builds on this observation by keeping spatial structure one step further—through class prediction as well as temporal modeling—and doing so causally on the dense token grid produced by a frozen image encoder and multi-granularity tokenizer.

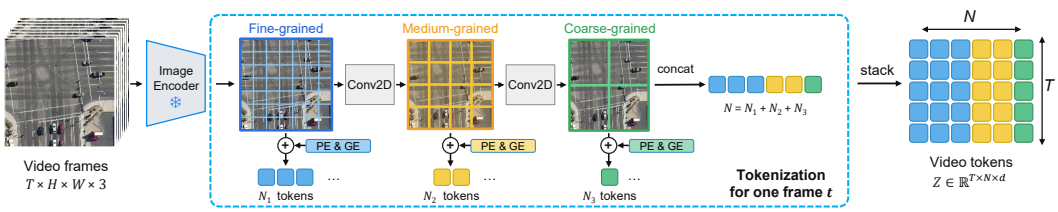

Figure 2: **Multi-granularity tokenizer.** A frozen encoder and two Conv2D stages produce fine-, medium-, and coarse-grained tokens, each with PE and GE added, concatenated into Z.

Object-centric video models preserve structured information over time by indexing representations with slots rather than image locations (Locatello et al., 2020; Kipf et al., 2022; Zadaianchuk et al., 2023). Such slots can follow moving objects, although maintaining consistent slot–object assignments over time remains an active research problem (Manasyan et al., 2025). In traffic video, Action-slot uses slot attention to focus on regions associated with atomic activities and recognizes multiple activities from short clips (Kung et al., 2024). SpaTAD instead indexes recurrent states by spatial location, avoiding explicit object or slot discovery; within-frame attention lets information move between locations as scene content evolves.

Multiple-instance learning and spatial aggregation. Multiple-instance learning (MIL) learns from bag-level labels when the responsible instances are unknown (Dietterich et al., 1997; Maron & Lozano-Pérez, 1998). Computing class scores at spatial locations and pooling them afterward is widely used with image-level supervision, including max pooling (Oquab et al., 2015), log-sum-exp pooling (Ramon & De Raedt, 2000; Pinheiro & Collobert, 2015; Wang et al., 2017), class activation map aggregation (Zhou et al., 2016), positive–negative evidence pooling (Durand et al., 2017), and attention-based pooling (Ilse et al., 2018; Zhu & Wu, 2021). In weakly supervised temporal action localization, MIL is instead applied over time: snippets are treated as instances and pooled with attention (Nguyen et al., 2018) or top-k selection (Paul et al., 2018; Lee et al., 2020; Zhang et al., 2021). SpaTAD applies MIL across *space* inside a temporal detector: each frame forms a bag of spatial tokens, class scores are computed at the token level, and spatial aggregation is performed only afterward. These token-wise predictions are learned without spatial annotations, letting us study whether spatial structure should remain explicit through class prediction.

3 Method

SpaTAD keeps spatial tokens separate through temporal modeling and class prediction, aggregating across space only afterward. It consists of a multi-granularity tokenizer (§3.1), a spatial-temporal encoder with per-token temporal memory (§3.2), and token-wise action prediction (§3.3).

3.1 Multi-granularity Tokenizer

To preserve spatial structure at different granularities, SpaTAD constructs multiple spatial token grids for each frame. As shown in Fig. 2, a frozen image encoder produces the fine-grained tokens, and two Conv2D stages further generate medium- and coarse-grained tokens. A 2-D positional embedding (PE) and a learned granularity embedding (GE) are added at each level. The three levels are then flattened and concatenated to form

$$Z = [z_{t,n}] \in \mathbb{R}^{T \times N \times d}, \qquad N = N_1 + N_2 + N_3. \tag{1}$$

3.2 Spatial-Temporal Encoder

To preserve spatial structure during temporal modeling, SpaTAD maintains a recurrent state for each spatial token, while within-frame attention exchanges information across locations.

Temporal State Space Module. Each spatial token $z_{t,n} \in \mathbb{R}^d$ keeps its own recurrent memory matrix $H_{t,n} \in \mathbb{R}^{d \times d_{\text{state}}}$. A shared Mamba block (Gu & Dao, 2024) projects the current token to

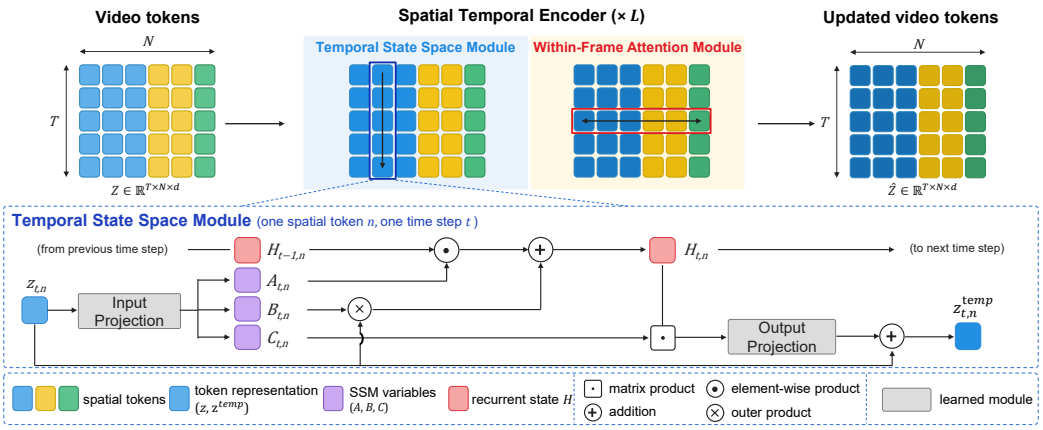

Figure 3: **Spatial-temporal encoder.** Each spatial token carries its own SSM state over time, and within-frame attention mixes tokens within a frame; the two alternate for L layers without pooling. Bottom: the SSM update for one token.

a scalar retention factor $A_{t,n} \in (0,1)$ and write/read vectors $B_{t,n}, C_{t,n} \in \mathbb{R}^{d_{\text{state}}}$. The temporal memory and token representation are updated as

$$H_{t,n} = A_{t,n} H_{t-1,n} + z_{t,n} B_{t,n}^{\top}, \tag{2}$$

$$z_{t,n}^{\text{temp}} = z_{t,n} + W_{\text{out}} \left(H_{t,n} C_{t,n} \right), \tag{3}$$

where $W_{\text{out}} \in \mathbb{R}^{d \times d}$. Here, $A_{t,n}$ controls memory retention, $B_{t,n}$ writes the current token into memory, and $C_{t,n}$ reads from the updated memory. The Mamba parameters are shared across token positions and pyramid levels, while each spatial token maintains its own recurrent memory.

Within-Frame Attention Module. Temporal modeling updates each token using its recurrent memory, while spatial modeling allows tokens within the current frame to exchange information. Let $Z_t^{\text{temp}} \in \mathbb{R}^{N \times d}$ denote the temporally updated tokens of frame t. We apply within-frame attention:

$$Z_t^{\text{spatial}} = Z_t^{\text{temp}} + \text{MHA}\big(\text{LN}\big(Z_t^{\text{temp}}\big)\big), \tag{4}$$

followed by a feed-forward network. Stacking the temporal and spatial modules for L layers produces the final encoded tokens $\widehat{Z}_t$ as in Fig. 3.

3.3 Token-wise Action Prediction

The encoder preserves spatial structure through temporal modeling. SpaTAD further preserves this structure through class prediction by predicting directly from each encoded spatial token before aggregating across space. For an action spanning $I = [f_s, f_e]$ at frame t, we define the start and end offsets as

$$a = t - f_s, \quad b = f_e - t,$$

which specify how far the action start lies in the past and the end lies in the future. The ground-truth end offset is a training target; presence and offset-bin predictions at frame t are computed causally. As shown in Fig. 4, action prediction proceeds in the following three steps.

Token-wise Presence and Offset Prediction. To predict a temporal action segment from each frame, SpaTAD predicts action presence together with the start and end offsets from the current frame. Because multiple instances of the same action class may be active at frame t, a single regressed offset can be ambiguous during training. We therefore discretize the start offset a into K_s bins and the end offset b into K_e bins. A start bin indicates how many frames before t the action starts, while an end bin indicates how many frames after t the action ends. The bins are predicted independently, allowing multiple offset bins to be supervised as positive when multiple matching

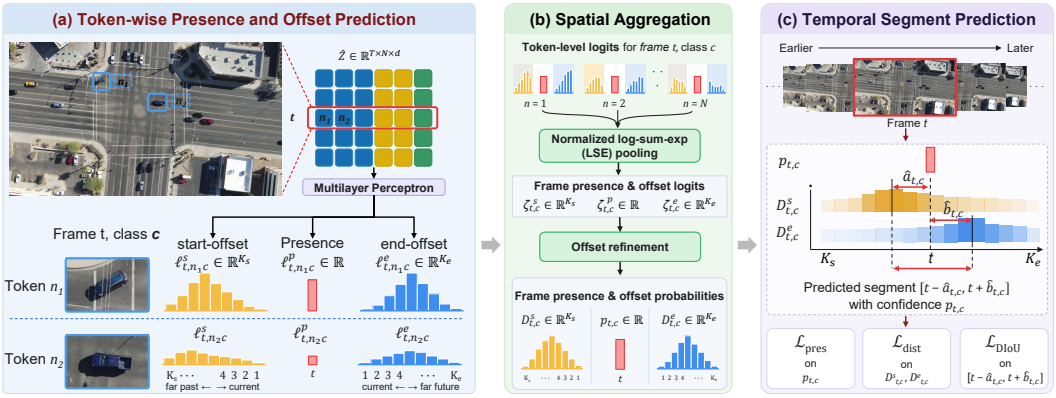

Figure 4: **Token-wise action prediction.** Each encoded token predicts presence and start/end offset logits. Token-level predictions are aggregated with LSE and refined for segment prediction.

instances are present. As shown in Fig. 4(a), a shared MLP is applied to each encoded token $\hat{z}_{t,n}$ and, for class c, outputs

$$\ell^p_{t,n,c} \in \mathbb{R}, \qquad \boldsymbol{\ell}^s_{t,n,c} \in \mathbb{R}^{K_s}, \qquad \boldsymbol{\ell}^e_{t,n,c} \in \mathbb{R}^{K_e}, \tag{5}$$

for presence, start-offset bins, and end-offset bins, respectively.

Spatial Aggregation. After token-level class prediction, SpaTAD aggregates predictions across the N spatial tokens to obtain frame-level outputs. Following MIL (Dietterich et al., 1997), a frame is treated as a bag of spatial tokens, where one or more tokens may provide evidence for an action. We therefore use normalized log-sum-exp (LSE) pooling to emphasize high-scoring tokens while allowing multiple locations to contribute (Fig. 4b). Pooling is applied across spatial tokens separately for presence and for each start- and end-offset bin:

$$\begin{aligned} \zeta^p_{t,c} &= \frac{1}{\tau}\log\left(\frac{1}{N}\sum_{n=1}^{N} e^{\tau \ell^p_{t,n,c}}\right), \\ \zeta^{s,k}_{t,c} = \frac{1}{\tau}\log\left(\frac{1}{N}\sum_{n=1}^{N} e^{\tau \ell^{s,k}_{t,n,c}}\right), &\qquad \zeta^{e,j}_{t,c} = \frac{1}{\tau}\log\left(\frac{1}{N}\sum_{n=1}^{N} e^{\tau \ell^{e,j}_{t,n,c}}\right). \end{aligned} \tag{6}$$

Here, $k = 1, \ldots, K_s$, $j = 1, \ldots, K_e$, and τ controls how strongly LSE favors high-scoring tokens.

We further refine every offset bin using the frame-level logits within its offset range. For each start bin k, we add the maximum $\zeta^{s,1}$ among the frames covered by that bin; each end bin j similarly uses the maximum $\zeta^{e,1}$. After sigmoid, $p_{t,c} = \sigma(\zeta^p_{t,c})$, and the offset-bin probabilities are

$$D^s_{t,c,k} = \sigma\left(\zeta^{s,k}_{t,c} + \max_{r \in k} \zeta^{s,1}_{t-r,c}\right), \qquad D^e_{t,c,j} = \sigma\left(\zeta^{e,j}_{t,c} + \max_{r \in j} \zeta^{e,1}_{t+r,c}\right), \tag{7}$$

where r denotes an integer offset covered by the corresponding bin. This refinement is used only for temporal segment prediction; frame-level action predictions at t remain causal. Collecting the probabilities over all bins gives $\mathbf{D}^s_{t,c} \in [0,1]^{K_s}$ and $\mathbf{D}^e_{t,c} \in [0,1]^{K_e}$, together with the frame-level presence probability $p_{t,c}$.

Temporal Segment Prediction. The frame-level offset probabilities $\mathbf{D}^s_{t,c}$ and $\mathbf{D}^e_{t,c}$ are decoded into the continuous start and end offsets $\hat{a}_{t,c}$ and $\hat{b}_{t,c}$ shown in Fig. 4(c). Each offset bin has a representative value ρ, defined as the median training offset assigned to that bin. For the start offset, we locate the highest-probability bin $\arg\max_k D^s_{t,c,k}$ and compute $\hat{a}_{t,c}$ by probability-weighted averaging its representative value with those of its immediate neighboring bins. The end offset $\hat{b}_{t,c}$ is obtained analogously from $\arg\max_j D^e_{t,c,j}$. The resulting temporal segment is

$$\widehat{I}_{t,c} = [t - \hat{a}_{t,c},\ t + \hat{b}_{t,c}], \tag{8}$$

with confidence $p_{t,c}$.

Training minimizes

$$\mathcal{L} = \mathcal{L}_{\text{pres}} + \mathcal{L}_{\text{dist}} + \mathcal{L}_{\text{DIoU}}. \tag{9}$$

$\mathcal{L}_{\text{pres}}$ supervises action presence with BCE. $\mathcal{L}_{\text{dist}}$ applies BCE-with-logits to the pre-sigmoid logits corresponding to $\mathbf{D}^s_{t,c}$ and $\mathbf{D}^e_{t,c}$, using bin targets derived from the ground-truth offsets a and b. Finally, $\mathcal{L}_{\text{DIoU}}$ directly supervises $\widehat{I}_{t,c}$ against I using DIoU (Zheng et al., 2020).

4 Experiments

4.1 Experimental Setup

Datasets and Features. We evaluate on MultiTHUMOS (Yeung et al., 2018), Charades (Sigurdsson et al., 2016), ATARS (Chen et al., 2025), and ROAD-Waymo (Khan et al., 2024). Our primary comparison uses the same frozen SigLIP 2 encoder (Tschannen et al., 2025) to provide a common source of dense patch-level spatial features across methods. Applicable baselines are additionally evaluated with RGB I3D features (Carreira & Zisserman, 2017), which encode spatial and temporal information jointly through 3D convolutions.

Metrics and Evaluation. Prior methods use different output formats: segment-based detectors output scored temporal segments, whereas dense detectors output per-frame class predictions. SpaTAD follows the dense TAD formulation, producing causal frame-level action predictions and predicting temporal segments from these dense outputs for Det-mAP evaluation. We report both *Det-mAP* and *Seg-mAP*. Det-mAP evaluates temporal segment localization over tIoU thresholds $\{0.1, 0.2, \ldots, 0.9\}$, while Seg-mAP evaluates frame-level multi-label action presence.

To compare both families under both metrics, predicted segments are converted to frame-level predictions by taking the maximum confidence over overlapping segments. Conversely, dense baseline predictions are converted to segments by thresholding and grouping consecutive positive frames, while SpaTAD uses its offset decoder (§3.3). All models are trained on the training split and evaluated once on the test split, with checkpoints and decoding settings selected on validation.

4.2 Comparison with Prior Methods

Quantitative results. As shown in Table 1, under the matched SigLIP 2 protocol, SpaTAD achieves the highest Det-mAP and Seg-mAP on all four benchmarks. The largest gains occur on ATARS ($+17.32$ Det / $+28.54$ Seg). As illustrated in Fig. 1, ATARS contains trajectory-defined actions and multiple concurrent vehicle instances, making it especially dependent on preserving where evidence occurs and how it moves over time; §4.3 separates this spatial contribution from the rest of the gain. SpaTAD also improves MultiTHUMOS by $+3.99$/ $+9.20$ and Charades by $+4.44$/ $+4.39$. Both datasets are highly multi-label, with 65 classes in MultiTHUMOS and 157 in Charades, and multiple actions can be active in the same frame. Keeping spatial evidence separate helps distinguish these concurrent actions (see pooled prediction, Table 2). On ROAD-Waymo, SpaTAD improves Seg-mAP by $+4.18$ but Det-mAP by only $+0.39$. Actions are annotated per agent, so several vehicles often share the same label at once (e.g., *Stop* at a red light), producing overlapping same-class segments. Seg-mAP is unaffected by this, but Det-mAP requires each instance to be recovered: SpaTAD decodes one segment per class per frame, so heavily overlapping instances tend to merge, whereas set-prediction detectors can output several overlapping segments.

Because SigLIP 2 encodes each frame independently, the matched protocol may disadvantage baselines designed for snippet-level spatiotemporal features. We therefore also evaluate them with I3D, the feature setting of their originally reported results. I3D gives no consistent gain on MultiTHUMOS, Charades, or ROAD-Waymo, but substantially improves most baselines on ATARS, where motion cues are essential (e.g., TriDetPlus: 6.39→11.22 Det-mAP). All of them nonetheless remain below SpaTAD, which uses only per-frame SigLIP 2 features. Spatiotemporal input features alone do not close the gap to SpaTAD under this setting.

Qualitative analysis. Figure 5 compares class-specific spatial evidence: normalized token readout weights $\mathrm{softmax}_n(\tau \ell^p_{t,n,c})$ for SpaTAD, and gradient-based patch attribution for TriDetPlus. On

Table 1: **Comparison with prior methods.** Det-mAP / Seg-mAP (mean ± std; best in bold). Top: all methods use the same frozen SigLIP 2 encoder; SpaTAD uses its dense patch tokens, while baselines use its pooled frame embedding. The last three rows of this block are dense methods. Bottom: baselines with I3D features, the setting of their original results.

		MultiTHUMOS		Charades		ATARS		ROAD-Waymo	
Method	Backbone	Det	Seg	Det	Seg	Det	Seg	Det	Seg
SpaTAD (ours)	SigLIP 2	**32.44**±0.48	**50.63**±0.07	**31.90**±0.84	**38.12**±0.54	**23.71**±0.42	**40.92**±0.90	**16.29**±0.39	**49.04**±0.55
TriDetPlus (Shi et al., 2023a)	SigLIP 2	28.45±0.28	41.43±0.06	27.46±0.54	33.73±0.35	6.39±0.73	11.95±0.86	15.90±0.10	42.13±0.50
TriDet (Shi et al., 2023b)	SigLIP 2	27.11±0.12	40.29±0.34	25.68±0.08	32.41±0.09	6.25±0.57	11.77±0.80	11.32±0.17	37.29±0.58
ActionFormer (Zhang et al., 2022)	SigLIP 2	24.03±0.19	38.51±0.20	26.72±0.12	33.64±0.09	5.32±0.55	10.90±0.80	11.31±0.28	37.66±0.56
TadTR (Liu et al., 2022)	SigLIP 2	19.77±0.20	35.21±0.23	20.43±0.35	29.11±0.30	4.62±0.72	12.38±1.07	6.52±0.12	26.31±0.36
PointTAD (Tan et al., 2022)	SigLIP 2	13.77±0.21	23.98±0.36	10.02±0.50	23.97±0.22	1.57±0.66	7.16±1.17	3.84±0.05	22.78±0.07
Action-slot (Kung et al., 2024)	SigLIP 2	8.05±0.87	26.49±1.66	8.36±0.26	22.94±0.35	3.25±3.58	11.24±5.50	5.42±0.42	44.00±0.88
MS-TCT (Dai et al., 2022)	SigLIP 2	12.99±0.77	36.88±0.84	12.99±0.31	30.93±0.27	2.19±0.43	12.25±0.58	4.97±0.17	44.86±0.26
MS-Temba (Sinha et al., 2026)	SigLIP 2	12.12±0.23	38.67±0.07	8.42±0.33	28.20±0.44	2.00±0.80	9.95±1.89	4.48±0.37	43.65±0.49
TriDetPlus	I3D	26.66±0.25	41.09±0.02	20.26±0.18	25.09±0.13	11.22±0.22	16.30±0.08	13.71±0.26	41.69±0.59
TriDet	I3D	25.78±0.09	40.56±0.47	19.48±0.23	24.63±0.34	10.35±0.46	16.70±0.24	10.47±0.64	37.93±0.08
ActionFormer	I3D	24.54±0.24	40.04±0.36	19.69±0.17	25.13±0.26	10.17±0.84	17.68±0.38	11.49±0.29	38.82±0.66
TadTR	I3D	22.34±0.27	31.41±0.14	14.26±0.18	20.92±0.20	1.02±0.29	10.55±0.07	6.45±0.29	31.05±0.77
PointTAD	I3D	11.84±3.79	10.26±0.75	8.42±0.62	17.74±0.07	4.45±0.08	12.42±0.15	3.78±0.22	25.49±0.50
MS-TCT	I3D	9.68±0.09	37.45±0.91	6.44±0.36	23.79±0.14	4.92±0.45	33.16±0.56	3.76±0.40	44.79±0.71
MS-Temba	I3D	12.93±0.61	40.29±0.64	3.54±0.20	21.16±0.23	4.82±0.54	27.61±1.46	3.80±0.17	44.53±0.13

Table 2: **Ablations and streaming checks.** The first three rows report paired changes vs. SpaTAD. *Persistent vs. windowed* reports the change on MultiTHUMOS when the recurrent state is carried across windows, and p95 is the per-frame latency including online SigLIP 2 encoding. † Reported only on MultiTHUMOS, where video and action durations substantially exceed the training window.

	MultiTHUMOS		Charades		ATARS		ROAD-Waymo	
Variant	ΔDet	ΔSeg	ΔDet	ΔSeg	ΔDet	ΔSeg	ΔDet	ΔSeg
Pooled prediction	−3.72±0.70	−4.76±0.27	−5.38±0.94	−5.88±0.82	−4.73±1.57	−4.81±2.45	−1.14±0.74	−2.55±1.27
Global temporal state	−7.20±0.02	−7.82±0.50	−4.76±0.81	−5.23±0.79	−9.80±0.68	−15.22±1.09	−0.95±0.01	−2.10±0.02
Bidirectional SSM	−0.19±0.12	+0.95±0.38	−4.03±2.86	+3.61±1.27	−3.99±0.94	−3.09±0.29	−0.07±0.17	−0.04±0.04
Persistent vs. windowed	+0.19±0.04	+0.20±0.19	N/A†		N/A†		N/A†	
p95 latency (ms)	30.8		61.5		19.6		49.7	

ATARS, vehicles look alike and share the same intersection context, so the class must be inferred from which vehicle follows which path. SpaTAD follows the vehicle associated with each trajectory class, while TriDetPlus often responds to the surrounding intersection. On MultiTHUMOS, SpaTAD responds when the athlete actually squats or jumps rather than whenever the athlete is visible: its evidence appears as the motion starts and fades when the action ends, whereas TriDetPlus sometimes activates on background regions or outside the action phase. On Charades, labels overlap in both time and space, so the same local tokens can support several classes. SpaTAD still separates their evidence: *eating* peaks near the head, *holding food* shifts toward the hand and held object, and *sitting* is supported around the body–seat region. TriDetPlus is less consistent across frames and often drifts to unrelated regions. SpaTAD's maps are learned from frame-level labels alone, without any spatial supervision.

4.3 Ablations and Streaming Analysis

Where to aggregate. The two controls keep SpaTAD's setup and differ only in where spatial aggregation occurs, using mean pooling. *Pooled prediction* keeps a temporal state at each location but pools tokens before class prediction, so every class reads from one spatial mixture and class-specific spatial readout is lost. *Global temporal state* pools before temporal modeling, so neither per-token memory nor within-frame attention remains. The gap between the two therefore measures the added value of spatially indexed temporal modeling. Pooling before class prediction hurts on all four benchmarks (Table 2). The additional cost of a global state is largest on ATARS ($5.07/10.41$), where several similar-looking vehicles share one scene and differ mainly in the paths they have taken. Even this variant reaches $13.91/25.70$ on ATARS, exceeding the best matched SigLIP 2 baselines of 6.39 and 12.38, respectively; the remaining $9.80/15.22$ of SpaTAD's gain comes from preserving spatial structure. MultiTHUMOS shows a smaller additional cost ($3.48/3.06$), consistent with actions driven by the local motion of one or a few athletes. On Charades and ROAD-Waymo, the two controls yield similar mean performance. Charades videos typically contain a single actor,

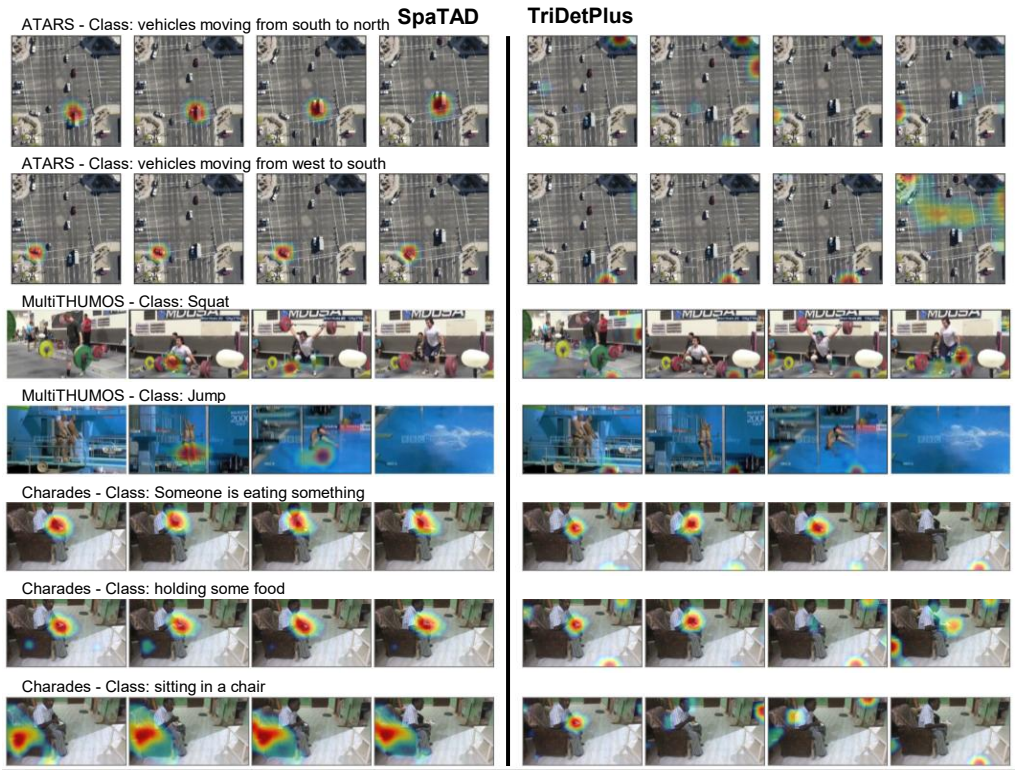

Figure 5: **Qualitative comparison of class-specific spatial evidence.** SpaTAD (left) shows normalized token readout weights, while TriDetPlus (right) shows gradient-based patch attribution through the frozen SigLIP 2 pooling head. Warmer colors indicate stronger evidence.

so a global state already captures whose motion matters. Many ROAD-Waymo classes are scene-level states (*Stop*, *MovAway*, *MovTow*, traffic lights), and its ego-centric camera moves, so memory indexed by image location does not correspond to fixed world locations. Location-bound memory thus matters most when multiple moving actors in a roughly static view must be distinguished by their motion histories.

Causal and streaming inference. Making the SSM bidirectional, which gives it access to future frames, does not improve Det-mAP on any benchmark and raises Seg-mAP only on MultiTHUMOS and Charades, so the causal design gives up little. For long-running applications such as surveillance and driving, the recurrent state must also remain stable beyond the training window. On MultiTHUMOS, where videos reach 28 minutes (over 4,000 frames) and some actions last several times the 128-frame training window, carrying the state across windows instead of resetting it slightly improves performance ($+0.19/+0.20$; Table 2). Including online SigLIP 2 encoding, SpaTAD's frame-level predictions have p95 latency below 62 ms on all four benchmarks (RTX 5090), within a 10 Hz frame budget. Offline detectors such as ActionFormer, TriDet, and TriDetPlus instead process fixed windows and must wait for future frames.

5 Conclusion

This work asks how far explicit spatial structure should be kept in temporal action detection. SpaTAD keeps spatial tokens separate through causal per-token temporal modeling and token-wise action prediction, aggregating across space only afterward. Across four benchmarks, pooling before class prediction consistently hurts, while the additional value of location-bound temporal memory is dataset-dependent and largest when multiple moving actors must be distinguished by their paths. More broadly, our results suggest treating where to aggregate spatially as an explicit design choice.

AI USE STATEMENT

Generative AI tools were used to polish the writing and assist with manuscript formatting. All AI-assisted edits were reviewed by the authors, who take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

All datasets and evaluation benchmarks used in this work are publicly available. The model architecture, training objectives, experimental settings, and evaluation protocol are described in the paper. We intend to release the model code and configurations upon completion of the required internal approval.

REFERENCES

Joungbin An, Hyolim Kang, Su Ho Han, Ming-Hsuan Yang, and Seon Joo Kim. MiniROAD: Minimal RNN framework for online action detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6299–6308, 2017.

Zihao Chen, Hsuanyu Wu, Chi-Hsi Kung, Yi-Ting Chen, and Yan-Tsung Peng. ATARS: An aerial traffic atomic activity recognition and temporal segmentation dataset. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11592–11598, 2025. doi: 10.1109/IROS60139.2025.11247259.

Rui Dai, Srijan Das, Luca Minciullo, Lorenzo Garattoni, Gianpiero Francesca, and François Brémond. PDAN: Pyramid dilated attention network for action detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

Rui Dai, Srijan Das, Kumara Kahatapitiya, Michael S. Ryoo, and François Brémond. MS-TCT: Multi-scale temporal convtransformer for action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20041–20051, 2022.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 2024.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1–2):31–71, 1997. doi: 10.1016/S0004-3702(96)00034-3.

Thibaut Durand, Taylor Mordan, Nicolas Thome, and Matthieu Cord. WILDCAT: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation. In *CVPR*, 2017.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling (COLM)*, 2024.

Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *ICML*, 2018.

Kumara Kahatapitiya and Michael S. Ryoo. Coarse-fine networks for temporal activity detection in videos. In *CVPR*, 2021.

Salman Khan, Izzeddin Teeti, Reza Javanmard Alitappeh, Mihaela C. Stoian, Eleonora Giunchiglia, Gurkirt Singh, Andrew Bradley, and Fabio Cuzzolin. ROAD-Waymo: Action awareness at scale for autonomous driving. *arXiv preprint arXiv:2411.01683*, 2024.

Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *International Conference on Learning Representations (ICLR)*, 2022.

Chi-Hsi Kung, Shu-Wei Lu, Yi-Hsuan Tsai, and Yi-Ting Chen. Action-slot: Visual action-centric representations for multi-label atomic activity recognition in traffic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18451–18461, 2024.

Pilhyeon Lee, Youngjung Uh, and Hyeran Byun. Background suppression network for weakly-supervised temporal action localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 11320–11327, 2020. doi: 10.1609/aaai.v34i07.6793.

Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. VideoMamba: State space model for efficient video understanding. In *ECCV*, 2024.

Shuming Liu, Chen-Lin Zhang, Chen Zhao, and Bernard Ghanem. End-to-end temporal action detection with 1B parameters across 1000 frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18591–18601, 2024.

Xiaolong Liu, Qimeng Wang, Yao Hu, Xu Tang, Shiwei Zhang, Song Bai, and Xiang Bai. End-to-end temporal action detection with transformer. *IEEE Transactions on Image Processing*, 31: 5427–5441, 2022.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Hui Lu, Yi Yu, Shijian Lu, Deepu Rajan, Boon Poh Ng, Alex C. Kot, and Xudong Jiang. MambaTAD: When state-space models meet long-range temporal action detection. *IEEE Transactions on Multimedia*, 2026. doi: 10.1109/TMM.2026.3676839.

Anna Manasyan, Maximilian Seitzer, Filip Radovic, Georg Martius, and Andrii Zadaianchuk. Temporally consistent object-centric learning by contrasting slots. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, pp. 570–576, 1998.

Phuc Nguyen, Ting Liu, Gautam Prasad, and Bohyung Han. Weakly supervised action localization by sparse temporal pooling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6752–6761, 2018.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free? – Weakly-supervised learning with convolutional neural networks. In *CVPR*, 2015.

Mandela Patrick, Dylan Campbell, Yuki M. Asano, Ishan Misra, Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, and João F. Henriques. Keeping your eye on the ball: Trajectory attention in video transformers. In *NeurIPS*, 2021.

Sujoy Paul, Sourya Roy, and Amit K. Roy-Chowdhury. W-TALC: Weakly-supervised temporal activity localization and classification. In *Computer Vision – ECCV 2018*, volume 11208 of *Lecture Notes in Computer Science*, pp. 588–607. Springer, 2018. doi: 10.1007/978-3-030-01225-0_35.

AJ Piergiovanni and Michael S. Ryoo. Learning latent super-events to detect multiple activities in videos. In *CVPR*, 2018.

AJ Piergiovanni and Michael S. Ryoo. Temporal gaussian mixture layer for videos. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, 2019.

Pedro O. Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1713–1721, 2015.

Yicheng Qiu and Keiji Yanai. DST-Mamba: Spatial–temporal adapter with boundary-aware mamba for video temporal action detection. *Big Data and Cognitive Computing*, 10(7):231, 2026a. doi: 10.3390/bdcc10070231.

Yicheng Qiu and Keiji Yanai. Efficient spatial-temporal focal adapter with SSM for temporal action detection. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2026b. arXiv:2604.09164.

Jan Ramon and Luc De Raedt. Multi instance neural networks. In *Proceedings of the ICML-2000 Workshop on Attribute-Value and Relational Learning*, pp. 53–60, 2000.

Michael S. Ryoo, Keerthana Gopalakrishnan, Kumara Kahatapitiya, Ted Xiao, Kanishka Rao, Austin Stone, Yao Lu, Julian Ibarz, and Anurag Arnab. Token Turing machines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19070–19081, 2023.

Faegheh Sardari, Armin Mustafa, Philip J. B. Jackson, and Adrian Hilton. An effective-efficient approach for dense multi-label action detection. *International Journal of Computer Vision*, 134 (3):124, 2026. doi: 10.1007/s11263-026-02738-x.

Dingfeng Shi, Qiong Cao, Yujie Zhong, Shan An, Jian Cheng, Haogang Zhu, and Dacheng Tao. Temporal action localization with enhanced instant discriminability. *arXiv preprint arXiv:2309.05590*, 2023a.

Dingfeng Shi, Yujie Zhong, Qiong Cao, Lin Ma, Jia Li, and Dacheng Tao. TriDet: Temporal action detection with relative boundary modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18857–18866, 2023b.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-Chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, volume 28, pp. 802–810, 2015.

Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. In *NeurIPS*, 2017.

Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Computer Vision – ECCV 2016*, volume 9905 of *Lecture Notes in Computer Science*, pp. 510–526. Springer, 2016. doi: 10.1007/978-3-319-46448-0_31.

Arkaprava Sinha, Monish Soundar Raj, Pu Wang, Ahmed Helmy, Hieu Le, and Srijan Das. MS-Temba: Multi-scale temporal mamba for understanding long untrimmed videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9815–9826, 2026.

Jing Tan, Xiaotong Zhao, Xintian Shi, Bin Kang, and Limin Wang. PointTAD: Multi-label temporal action detection with learnable query points. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Praveen Tirupattur, Kevin Duarte, Yogesh S. Rawat, and Mubarak Shah. Modeling multi-label action dependencies for temporal action localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. ChestX-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S. Davis, and David J. Crandall. Temporal recurrent networks for online action detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Mingze Xu, Yuanjun Xiong, Hao Chen, Xinyu Li, Wei Xia, Zhuowen Tu, and Stefano Soatto. Long short-term transformer for online action detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Yibin Yan, Jilan Xu, Shangzhe Di, Yikun Liu, Yudi Shi, Qirui Chen, Zeqian Li, Yifei Huang, and Weidi Xie. Learning streaming video representation via multitask training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9900–9912, 2025.

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. *International Journal of Computer Vision*, 126(2):375–389, 2018. doi: 10.1007/s11263-017-1013-y.

Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Object-centric learning for real-world videos by predicting temporal feature similarities. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Can Zhang, Meng Cao, Dongming Yang, Jie Chen, and Yuexian Zou. CoLA: Weakly-supervised temporal action localization with snippet contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16010–16019, 2021.

Chen-Lin Zhang, Jianxin Wu, and Yin Li. ActionFormer: Localizing moments of actions with transformers. In *Computer Vision – ECCV 2022*. Springer, 2022.

Yue Zhao and Philipp Krähenbühl. Real-time online video detection with temporal smoothing transformers. In *Computer Vision – ECCV 2022*. Springer, 2022.

Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 12993–13000, 2020. doi: 10.1609/aaai.v34i07.6999.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

Ke Zhu and Jianxin Wu. Residual attention: A simple but effective method for multi-label recognition. In *ICCV*, 2021.

Yuhan Zhu, Guozhen Zhang, Jing Tan, Gangshan Wu, and Limin Wang. Dual DETRs for multi-label temporal action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18559–18569, 2024.

This appendix supplements the main text and follows its order. §A gives the implementation of the three components of §3 (Figs. 2–4), §B the training details behind Eq. (9), §C the inference details behind Eq. (8), §D the evaluation protocol of §4.1, §E the implementation of the ablation variants of Table 2, and §F the persistent-state and latency measurements of Table 2. Notation is that of §3; in addition, β_k denotes the lower edge of offset bin k (Table 4) and fps the frame rate of the token sequence (Table 5). Offsets a and b are counted in frames, and f_e in $I = [f_s, f_e]$ is the last frame of the action, so the first frame of an action has $a = 0$ and its last frame has $b = 0$.

A Implementation Details

A.1 Multi-granularity tokenizer (Sec. 3.1, Fig. 2)

Frozen image encoder. The image encoder of Fig. 2 is SigLIP 2 So400m NaFlex (patch size 16), used through its vision tower only and never fine-tuned. Its output tokens (1152-d, after the final LayerNorm) are extracted once per frame and cached. The image processor keeps the aspect ratio and fits at most 256 patches per frame, which gives the fine grids of Table 3. The pooled-feature baselines of Table 1 consume the encoder's own attention-pooled frame vector (1152-d) computed from the same tokens.

Conv2D stages. Each fine token is projected by $\mathrm{LN}(\mathrm{Linear}_{1152\to512}(\cdot))$, so $d = 512$. The two Conv2D stages of Fig. 2 are each a learned 3×3 convolution with stride 2, applied serially (medium from fine, coarse from medium); the resulting grids are listed in Table 3.

PE and GE. The positional embedding PE is a fixed 2-D sinusoidal embedding (256 dimensions per axis, temperature 10^4) shared by the three levels. Its input coordinate is computed on the finest grid: the cells of the finest grid that the token covers are taken, and the centre of that region is divided by the finest grid size, giving a position in $[0, 1]^2$. A fine token thus sits at the centre of its own cell, a medium token at the centre of the fine cells read by its Conv2D stage, and a coarse token at the centre of the fine cells covered through both stages. The granularity embedding GE is a learned vector per level (3×512, initialized from $\mathcal{N}(0, 1)$). Both are added after down-sampling and before the encoder, as in Fig. 2.

Table 3: Token grids per dataset. Every fine token corresponds to one 16×16 patch of the processed frame; the medium and coarse grids follow from the two stride-2 Conv2D stages.

Dataset	Fine (N_1)	Medium (N_2)	Coarse (N_3)	N	d
MultiTHUMOS	12×21 (252)	6×11 (66)	3×6 (18)	336	512
Charades	12×21 (252)	6×11 (66)	3×6 (18)	336	512
ATARS	16×16 (256)	8×8 (64)	4×4 (16)	336	512
ROAD-Waymo	13×19 (247)	7×10 (70)	4×5 (20)	337	512

A.2 Spatial-temporal encoder (Sec. 3.2, Fig. 3)

Every temporal and spatial module of Fig. 3 is a pre-LN residual block; the number and order of modules per dataset are given in Table 5. The temporal state space module is the Mamba-2 block (Dao & Gu, 2024), run along time: for each token position n, the tokens at that position over the frames, $(z_{1,n}, \ldots, z_{T,n})$, form one input sequence of the block, so the recurrence of §3.2 advances the state of position n from frame to frame. As stated in §3.2, the block's parameters are shared by all positions and levels, while every position keeps its own state. We use the official Mamba-2 implementation with token width $d = 512$ and state dimension $d_{\mathrm{state}} = 64$, and its default settings otherwise. The within-frame attention module is multi-head self-attention over all N tokens of the frame (8 heads of dimension 64, no locality mask, tokens of different granularities attend to each other), followed by a two-layer feed-forward network (widths $512 \to 2048 \to 512$, GELU). Dropout $p = 0.1$ (Charades 0.2) is applied to the temporal-module output, to the attention weights, and inside the feed-forward network and the prediction head; no stochastic depth is used.

A.3 Token-wise action prediction (Sec. 3.3, Fig. 4)

Shared MLP and LSE. The shared MLP of §3.3 is a LayerNorm followed by two linear layers (widths $512 \to 2048 \to C(1 + K_s + K_e)$) with GELU and dropout between them, applied to every encoded token $\hat{z}_{t,n}$. Its $C\,(1 + K_s + K_e)$ outputs are, for each of the C classes, one presence logit ℓ^p, K_s start-bin logits $\ell^{s,1..K_s}$ and K_e end-bin logits $\ell^{e,1..K_e}$, stored in that order class by class; $K_s = K_e = 9$. Eq. 6 uses $\tau = 3$ on every channel.

Offset bins and representatives ρ. §3.3 discretizes the start offset a into K_s bins and the end offset b into K_e bins, each with a representative ρ. A bin is an interval $[\beta_k, \beta_{k+1})$ of integer offsets, fitted once from the training-split annotations before training and then frozen, as follows. For every frame t of every training instance $[f_s, f_e]$, we collect its start offset $a = t - f_s$, the distance from that frame back to the first frame of the instance; an instance of L frames thus contributes one value per frame, $a = 0, 1, \ldots, L-1$. Its end offsets $b = f_e - t$ take the same values in reverse temporal order, the first frame having $b = L - 1$ and the last frame $b = 0$. Bin 1 is reserved for $a = 0$, i.e. the frame is the first frame of its action. The remaining values $a \geq 1$ are sorted, and their $1/8, 2/8, \ldots, 7/8$ quantiles (rounded down to integers, duplicates removed) are the lower edges $\beta_2, \ldots, \beta_K$; the last bin is open. The representative ρ_k is the median offset of the frames that fall in bin k, with $\rho_1 = 0$. The end side is fitted identically from b; since both sides see the same values $0, \ldots, L-1$, the fitted start and end bins coincide. Table 4 lists the result. On MultiTHUMOS (10 Hz), for example, the

bins are $\{0\}$, $[1, 5)$, $[5, 10)$, $[10, 18)$, $[18, 28)$, $[28, 44)$, $[44, 75)$, $[75, 199)$ and $[199, \infty)$ frames with $\rho = 0, 2, 7, 13, 22, 35, 56, 110, 522$; a frame that lies 0.8 s (8 frames) after the start of its action falls in the third bin, and one that lies a minute after it in the last bin.

Table 4: Fitted offset bins (lower edges β_k) and representatives ρ_k, in frames at the dataset's frame rate; the start and end sides are identical.

Dataset	Lower edges β_k	ρ_k
MultiTHUMOS	0,1,5,10,18,28,44,75,199	0,2,7,13,22,35,56,110,522
Charades	0,1,5,10,15,21,30,43,63	0,2.5,7,12,17,25,35,51,78
ATARS	0,1,2,3,5,7,9,14,23	0,1,2,3,5,7,11,17,34
ROAD-Waymo	0,1,5,10,16,24,35,49,72	0,2,7,12,19,29,41,58,89

Offset refinement (Eq. 7). In Eq. 7, $r \in k$ means $\beta_k \leq r < \beta_{k+1}$ (Table 4). Bin 1 receives no refinement on either side, since its own logit already is the evidence that the action starts, or ends, at this frame. The refinement is applied both in training and at inference, after spatial aggregation and before the sigmoid. As stated in §3.3, the end-side term uses frames after t; it is therefore a post-processing step applied once the causal per-frame logits of the training window or of the test video have been collected, and it does not change those logits.

Segment prediction (Eq. 8). The probability-weighted averaging of §3.3 uses the arg-max bin and its two immediate neighbours; a neighbour outside $1..K_s$ is dropped, so the first and last bin average over two bins, and $\hat{a}_{t,c} = \sum_{k\in\mathcal{N}} D^s_{t,c,k}\,\rho_k \,/\, \sum_{k\in\mathcal{N}} D^s_{t,c,k}$ with $\mathcal{N}$ the retained bins; $\hat{b}_{t,c}$ is obtained in the same way from $\mathbf{D}^e_{t,c}$. Det-mAP compares segments with the annotations in seconds, so $\widehat{I}_{t,c}$ is converted from frames to seconds: it spans from the beginning of its first frame to the end of its last frame, $[(t - \hat{a}_{t,c})/\mathrm{fps},\ (t + \hat{b}_{t,c} + 1)/\mathrm{fps})$, clipped to the video.

B Training Details

This section details the three terms of Eq. 9 and the training protocol of §4.1. Dataset-specific settings are collected in Table 5; everything not listed there is shared across datasets.

Annotations. An annotated instance $[s, e]$ in seconds is converted to frames at the dataset's frame rate by $f_s = \mathrm{round}(s \cdot \mathrm{fps})$ and $f_e = \mathrm{round}(e \cdot \mathrm{fps}) - 1$, so that f_e is the last frame inside the annotation.

Targets. For every frame t inside an instance of class c, the presence target is 1. The targets of the start bins are built from the start offset a of the frame as follows. Each bin k has a representative value ρ_k (Table 4). Find the two neighbouring representatives that enclose a, $\rho_j \leq a < \rho_{j+1}$. Give bin j the target $1 - w$ and bin $j{+}1$ the target w, with $w = (a - \rho_j)/(\rho_{j+1} - \rho_j)$; all other bins get 0. The two targets sum to one, and $(1 - w)\rho_j + w\rho_{j+1} = a$, so the targets encode the exact offset, not only the bin it falls in. If $a < \rho_1$, bin 1 gets target 1; if $a \geq \rho_{K_s}$, bin K_s gets target 1. Example on MultiTHUMOS: a frame 8 frames after the start of its action has $a = 8$, which lies between $\rho_3 = 7$ and $\rho_4 = 13$; bin 3 gets $5/6$, bin 4 gets $1/6$, and $7 \cdot 5/6 + 13 \cdot 1/6 = 8$. The end-bin targets are built in the same way from b. The bin edges play no role in the targets; they define the refinement ranges of Eq. 7 and the reference bin of $\mathcal{L}_{\mathrm{DIoU}}$ below. When several instances of the same class cover (t, c), the presence and bin targets are merged by an element-wise maximum without renormalization, so more than one bin can be positive, which is the multi-instance case motivated in §3.3. Frames outside every instance of class c have all targets equal to zero.

$\mathcal{L}_{\mathrm{pres}}$ and $\mathcal{L}_{\mathrm{dist}}$. Let $y^p_{t,c} \in \{0, 1\}$ and $\mathbf{y}^s_{t,c}, \mathbf{y}^e_{t,c}$ be the targets above and $\omega(y; w) = 1 + (w - 1)\,y$. With π the fraction of positive (t, c) pairs in the training split (over full videos, after the same-class

merge) and π_k the fraction of pairs whose start bin k is positive (likewise π^e_j),

$$\mathcal{L}_{\text{pres}} = \frac{1}{|\mathcal{V}|\,C} \sum_{(t,c)\in\mathcal{V}} \omega(y^p_{t,c}; w_p)\; q_{t,c}\,\text{BCE}\big(\zeta^p_{t,c},\, y^p_{t,c}\big), \qquad w_p = \min\big(10,\, \sqrt{(1-\pi)/\pi}\,\big), \quad (10)$$

$$\mathcal{L}_{\text{dist}} = \frac{1}{|\mathcal{V}|\,C\,K_s} \sum_{(t,c)\in\mathcal{V}} \sum_{k=1}^{K_s} \omega(y^{s,k}_{t,c}; w^s_k)\,\text{BCE}\big(\tilde{\zeta}^{s,k}_{t,c},\, y^{s,k}_{t,c}\big) \;+\; \text{(same for the end side)}, \quad (11)$$

where $\mathcal{V}$ is the set of valid (non-padded) frames of the window, $\tilde{\zeta}^{s,k}_{t,c}$ is the refined logit inside the sigmoid of Eq. 7 (so that $D^s_{t,c,k} = \sigma(\tilde{\zeta}^{s,k}_{t,c})$), and $w^s_k = \min(10, \sqrt{(1-\pi_k)/\pi_k})$. The factor $q_{t,c}$ ties the presence score to localization quality. At inference the confidence of a segment is $p_{t,c}$ (Eq. 8), so a frame whose predicted segment is badly placed should not be pushed to a high presence score as strongly as a frame whose segment is accurate. We therefore weight the presence loss of every positive pair by $q_{t,c} = Q^{0.2}_{t,c}$, where $Q_{t,c} \in [0,1]$ is the IoU between the segment predicted at (t,c) and its ground-truth instance (computed in $\mathcal{L}_{\text{DIoU}}$ below, without gradient). The exponent 0.2 follows TriDetPlus and keeps the weight close to one except for badly localized frames. Negative pairs use $q_{t,c} = 1$. $\mathcal{L}_{\text{dist}}$ is evaluated on all valid frames, including frames outside every instance, whose bin targets are zero.

$\mathcal{L}_{\text{DIoU}}$. $\mathcal{L}_{\text{DIoU}}$ supervises $\widehat{I}_{t,c}$ against I for every ground-truth instance-frame (t,c,I) of the W-frame window, so overlapping instances of the same class each contribute their own term. It uses the refined probabilities $\mathbf{D}^s_{t,c}$ and $\mathbf{D}^e_{t,c}$ of Eq. 7, but the bin around which $\hat{a}_{t,c}$ is averaged is chosen with the ground truth in view, so that each instance is compared with the prediction that is closest to it. For an instance-frame with true offsets (a,b), let k_a be the bin with $\beta_{k_a} \le a < \beta_{k_a+1}$ and k_b the bin containing b. A start bin is a candidate if its probability is no smaller than that of its existing neighbouring bins (the first and last bin compare with their single neighbour). Among the candidates, the bin nearest to k_a in bin index is selected, ties going to the smaller index; this selection is not differentiated. $\hat{a}$ is then computed by a probability-weighted average of the representatives in a three-bin neighbourhood around the selected bin. For this DIoU training term, if the selected bin is the first or last bin, the missing neighbour is replaced by the selected bin itself, so the endpoint bin is counted twice. $\hat{b}$ is obtained in the same way from the end bins and k_b. With interval lengths counted in frames, $\text{IoU} = \frac{\min(\hat{a},a)+\min(\hat{b},b)+1}{(\hat{a}+\hat{b}+1)+(a+b+1)-\min(\hat{a},a)-\min(\hat{b},b)-1}$, enclosing length $E = \max(\hat{a},a) + \max(\hat{b},b) + 1$ and centre offset $\delta = \frac{1}{2}\big((\hat{b}-\hat{a}) - (b-a)\big)$,

$$\mathcal{L}_{\text{DIoU}} = \frac{1}{M} \sum_{(t,c,I)} \Big(1 - \text{IoU} + \delta^2/E^2\Big), \quad (12)$$

averaged over the M instance-frames of the batch. $Q_{t,c}$ in $\mathcal{L}_{\text{pres}}$ is the largest of these IoU values over the instances covering (t,c), detached. The three terms are summed with unit weights, as in Eq. 9. Padded frames are masked out of $\mathcal{L}_{\text{pres}}$ and $\mathcal{L}_{\text{dist}}$; $\mathcal{L}_{\text{DIoU}}$ is defined on instance-frames and uses no mask.

Window sampling. Training windows have a fixed length of W frames (Table 5). Candidate windows are placed over every training video at a stride of $0.75\,W$, plus one window ending at the last frame; candidates that contain no instance are dropped. Each time a candidate is used, two random perturbations are applied so that the same instances are seen at different positions and lengths within the window: the window start is shifted by a uniform random offset of at most J frames in either direction (the jitter J of Table 5), and the window length is drawn uniformly between $0.9\,W$ and W frames. If the perturbed window contains no instance, the perturbation is redrawn, at most 20 times. An instance belongs to a window if it overlaps it; its presence target covers the overlap, while its offsets are measured to its true endpoints, also when these lie outside the window. Windows shorter than W, including the tail of a video, are zero-padded, and the padded frames are masked out of $\mathcal{L}_{\text{pres}}$ and $\mathcal{L}_{\text{dist}}$. No feature-level augmentation is used. ATARS ships eight geometric variants (rotations and mirror) of every source clip; they are treated as separate videos and always fall on the same side of a split.

Optimization and model selection. The optimizer is AdamW (learning rate 2×10^{-4}, weight decay 0.05, constant schedule, no warm-up) with gradient-norm clipping at 0.1. Each step accumulates 4 mini-batches of 2 windows. The forward pass runs in bf16 autocast; losses are computed in fp32. An exponential moving average of the weights (decay 0.999, warm-up $\min(0.999, (1+n)/(10+n))$ at step n) is used for every evaluation and for the reported model. Training runs for at most 60 epochs; the EMA model is evaluated on the validation split every 5 epochs with the default segment prediction of §C, training stops early when the validation Det-mAP no longer improves, and the checkpoint with the best validation Det-mAP is evaluated on the test split once, as stated in §4.1. Reported numbers are the mean $\pm$ standard deviation over training runs with different random seeds.

Table 5: Dataset-specific settings. W is the window length in frames at the dataset's frame rate; J is the largest random shift of the window start during training; T and S denote the temporal and spatial modules of Fig. 3 in order; parameters are the trainable parameters of SpaTAD, excluding the frozen image encoder (the head width scales with C).

	MultiTHUMOS	Charades	ATARS	ROAD-Waymo
Classes C	65	157	32	22
Frame rate (Hz)	10	3	30	6
Train window W / jitter J (frames)	128 / 96	72 / 54	128 / 12	128 / 12
Test window / stride (frames)	128 / 64	72 / 36	128 / 64	128 / 64
Encoder modules	T-S-T-S-T	T-S-T-S-T-S	T-S-T-S-T	T-S-T-S-T
Dropout	0.1	0.2	0.1	0.1
Trainable parameters (M)	15.98	22.71	14.69	14.30

C Inference Details

This section details how the frame-level probabilities of Eq. 7 are obtained on a test video and how the segments of Eq. 8 are turned into the detection output scored in §4.1.

Frame-level probabilities. A test video is processed in windows of length W with stride $W/2$ (Table 5; the last window is aligned to the video end). Every window starts from a zero recurrent state, and the pre-sigmoid frame-level logits $(\zeta^{p}, \zeta^{s,\cdot}, \zeta^{e,\cdot})$ of overlapping windows are averaged per frame. The offset refinement of Eq. 7 is then applied on the whole video and the sigmoid is taken, giving $p_{t,c}$, $\mathbf{D}^{s}_{t,c}$ and $\mathbf{D}^{e}_{t,c}$ for every frame. Seg-mAP uses $p_{t,c}$ directly.

Temporal segments. Every (t, c) with $p_{t,c} \geq 10^{-3}$ (at most the 2000 highest-p frames per class) proposes the segment $\widehat{I}_{t,c}$ of Eq. 8, converted to seconds as in §A. The segments of a class are then merged by Gaussian soft-NMS (score decay $\exp(-\mathrm{tIoU}^2/0.5)$, stop below 10^{-3}, at most 8000 kept), and the 8000 highest-scoring segments of the video are returned. The segment confidence is $p_{t,c}$ by default. As for every method in §4.1, all decoding settings are selected on the validation split and the test split is evaluated once. For SpaTAD the selected setting is the form of the confidence, chosen from $\{p,\ p\,m_s m_e,\ p\sqrt{m_s m_e},\ p\,m_s,\ p\,e^{-\lambda(h_s+h_e)}\}$ with $\lambda \in \{0.5, 1, 2\}$, where m is the maximum bin probability and h the normalized entropy of the start or end bin probabilities; for this selection the soft-NMS stop is lowered to 10^{-7}. All rows of Table 2 use the default confidence $p_{t,c}$ for every variant, so paired differences share the same segment prediction setting.

D Evaluation Protocol

This section details the protocol summarized in §4.1.

Splits. Every dataset uses its official test split unchanged. A validation split is carved out of the official training videos once, deterministically, stratified so that every class keeps a minimum number of validation videos: MultiTHUMOS 160/40/211, Charades 7143/840/1863, ATARS 168/48/48 (split by source clip, so geometric variants stay together), ROAD-Waymo 539/61/198 (train/val/test videos). All methods, including the baselines, use the same three lists.

Ground truth. MultiTHUMOS, Charades and ATARS provide segment annotations, which are used as instances directly. ROAD-Waymo provides per-frame annotations: every frame lists the agents present and the action of each agent. We derive the two ground truths from these annotations. Instances: for one agent, a run of consecutive frames with the same action is one instance. Frame labels: a class is present in a frame if any agent performs it in that frame. Because several agents can perform the same action at the same time, ROAD-Waymo has same-class instances that overlap in time.

Det-mAP. For each class, predicted segments of all test videos are ranked by confidence and matched greedily to unmatched ground-truth instances of the same video at temporal IoU $\geq \theta$ (ActivityNet-style average precision); mAP is the mean over classes with at least one instance, averaged over $\theta \in \{0.1, 0.2, \ldots, 0.9\}$ as stated in §4.1.

Seg-mAP. For each class, average precision of the frame-level score over all test frames of all videos against the frame-level label; mAP is the mean over classes with at least one positive frame.

Cross-family conversion. Every method is evaluated with both metrics. Det-mAP needs scored segments and Seg-mAP needs frame scores. A method that natively produces only one of the two is converted to the other with a single fixed rule, the same for every method. Segments to frame scores: every segment writes its confidence onto the frames it covers, a frame covered by several segments takes the maximum, and uncovered frames score zero. Frame scores to segments: the frame scores are thresholded with a threshold selected on the validation split, consecutive positive frames are grouped into one segment, and the segment takes the mean frame score inside it as its confidence. Set-prediction baselines produce segments with their official post-processing and receive the first conversion; dense baselines produce frame scores and receive the second. SpaTAD produces both frame scores and segments (§3.3, §C) and needs neither conversion.

Baselines. All baselines of Table 1 run from their official implementations with a thin adapter that reads our feature files (Table 6). Under the SigLIP 2 track they receive the encoder's pooled frame vector; under the I3D track the single-stream RGB InceptionI3d feature (Kinetics weights, 1024-d) at the snippet stride of their published configurations. Training schedules follow the published configurations, with an epoch cap of two to three times the published length, validation every 5 epochs and early stopping as for SpaTAD; the checkpoint with the best native validation metric is evaluated on the test split once.

Table 6: Baseline inputs and selection. Temporal input is the native unit of each method on MultiTHUMOS (frames at the feature rate); TadTR uses 256 frames on ATARS, TriDetPlus 1152 frames on Charades, and PointTAD 256 frames with I3D features.

Method	Family	SigLIP 2 input	I3D input	Temporal input	Selection metric
TriDetPlus	set-prediction	pooled frame vector	1024-d snippet	whole video ($\leq$4096)	Det-mAP
TriDet	set-prediction	pooled frame vector	1024-d snippet	whole video ($\leq$2304)	Det-mAP
ActionFormer	set-prediction	pooled frame vector	1024-d snippet	whole video ($\leq$2304)	Det-mAP
TadTR	set-prediction	pooled frame vector	1024-d snippet	128-frame slices	Det-mAP
PointTAD	set-prediction	pooled frame vector	1024-d snippet	64-frame windows	Det-mAP
MS-TCT	dense	pooled frame vector	1024-d snippet	1024-frame windows	Seg-mAP
MS-Temba	dense	pooled frame vector	1024-d snippet	512-frame windows	Seg-mAP
Action-slot	dense	patch tokens	–	64-frame windows	Seg-mAP

E Ablation Variants of Table 2

The three variants keep every setting of §B and §C and change one component; each is compared with the SpaTAD run trained under identical settings, so Table 2 reports paired differences.

Pooled prediction. After the spatial-temporal encoder, the N encoded tokens of a frame are pooled into one vector by a learned attention pooling (one learned query attending over the tokens with multi-head attention, pre-norm on keys and values), and the shared MLP of §3.3 is applied

once to this vector. The LSE aggregation of Eq. 6 therefore acts on a single token and becomes the identity; the per-token temporal states of the encoder are unchanged.

Global temporal state. The same attention pooling is applied to the tokenized frame before the first temporal module, so the encoder processes one token per frame: a single recurrent state per frame and a within-frame attention over a single token. The head is applied to this token as above.

Bidirectional SSM. Every temporal module runs its Mamba-2 block twice with the same parameters, once on the frame sequence and once on the time-reversed sequence, and averages the two outputs. The module therefore reads past and future frames, and the variant is evaluated with windowed inference only.

F Persistent-State and Runtime Measurements

This section details the *Persistent vs. windowed* row and the p95 latency of Table 2.

Persistent-state protocol. *Windowed* inference is the protocol of §C: every window of W frames starts from a zero recurrent state and overlapping logits are averaged. *Persistent* inference initializes the state once at the first frame of a video and advances it frame by frame with the Mamba-2 recurrence to the last frame, never resetting it; each frame updates the state exactly once, and only the current token grid and the per-token recurrent states are carried from frame to frame. Segment prediction and evaluation are identical in both modes, so the comparison isolates the effect of carrying the state beyond the training window. Training never exposes the recurrence to more than $W = 128$ frames (12.8 s), so persistent inference over a whole video is an extrapolation.

Choice of benchmark. MultiTHUMOS test videos range from 15 s to 27.9 min (median 2.7 min, 16,734 frames at most), i.e. up to 130 training windows, and 3.8% of its instances are longer than the window, so windowed inference restarts from an empty state inside long actions. ROAD-Waymo clips are 20 s ($\leq$ 120 frames, shorter than one window), Charades videos are at most 217 frames (median 92, at most three 72-frame windows), and ATARS instances are almost all shorter than the window (median 9 frames, 0.2% longer than 128), so carrying the state across windows either coincides with windowed inference or changes a few seconds of context; the stress test is therefore reported on MultiTHUMOS only, as noted in Table 2.

Elapsed-time breakdown. Table 7 evaluates frame-level mAP on the same frames under both modes, grouped by the elapsed time of the frame within its video, over all 211 test videos.

Table 7: Persistent vs. windowed inference on MultiTHUMOS (Seg-mAP per elapsed-time bucket; mean $\pm$ std over runs). Overall Det-mAP / Seg-mAP with the default segment prediction: windowed 32.19 ± 0.37 / 50.63 ± 0.07, persistent 32.39 ± 0.36 / 50.83 ± 0.15 ($\Delta = +0.19\pm0.04$ / $+0.20\pm0.19$).

Elapsed (min)	0–1	1–3	3–7	7–14	14–28
Frames (classes)	120k (65)	164k (65)	95k (59)	57k (43)	22k (29)
Windowed	53.56 ± 0.43	51.37 ± 0.63	49.85 ± 0.31	46.46 ± 0.75	48.02 ± 0.14
Persistent	53.83 ± 0.43	51.46 ± 0.54	49.68 ± 0.54	46.42 ± 0.69	47.67 ± 0.92
Δ	$+0.27 \pm 0.02$	$+0.10 \pm 0.12$	-0.17 ± 0.30	-0.05 ± 0.06	-0.35 ± 0.81

Runtime measurement. The p95 latency of Table 2 is measured on one NVIDIA RTX 5090 (PyTorch 2.9, CUDA 12.8) in the per-frame path with batch size 1 and no gradient. The raw frames of one test video are delivered by a feeder thread on a fixed schedule $t_0 + i/\mathrm{fps}$ at the dataset's frame rate, as a camera would deliver them. For every frame, the consumer runs the SigLIP 2 encoder on the raw image, advances the per-token states of SpaTAD, and outputs the frame-level scores, with a CUDA synchronization after the encoder and after the model. Latency is the wall-clock time from the moment a raw frame becomes available to the moment its scores are output. It therefore includes image encoding, the SpaTAD step, and any waiting time when the previous frame is still being processed; it excludes segment post-processing. The first 20 frames are excluded as warm-up

after 10 untimed warm-up steps. The throughput ceiling is the same loop with frames delivered as fast as they are consumed.