FastSAM3: Accelerating SAM3 with Lazy Detection and Tracking

Anonymous authors
Paper under double-blind review

Abstract

SAM3 achieves robust performance in video referring segmentation, benefiting from the collaboration and mutual disambiguation between its detector and tracker. However, we observe that the dense per-frame computation of this dual-branch design introduces substantial temporal redundancy, resulting in high computational cost and inference latency: (1) Redundant detection execution. The detector is activated at every frame by default. However, in scenarios such as stable scenes, absence of newly appearing objects, or minimal visual changes, the semantic information becomes largely redundant. (2) Redundant tracking memory. The tracker establishes object associations via cross-attention over a large memory sequence. However, during phases with limited dynamics and distractions, the contribution of most memory frames is marginal, thus involving all of them in computation leads to unnecessary resource consumption. To address these issues, we propose FastSAM3, which introduces lazy detection and tracking to address temporal redundancy and accelerate inference. Specifically, for detector, we propose Entrant-Predictive Gating, which leverages tracking signal and early-layer responses to assess the likelihood of new object emergence, thereby selectively trigger detection on demand. For tracker, we propose Divergence-Aware Gating, which switches between sparse memory and full memory based on tracking divergence, performing full computation only when necessary. Without any training, FastSAM3 achieves adaptive computational budget allocation, delivering 1.77× end-to-end speedup with only 0.6 performance drop on the SA-V test set, where lazy detection and lazy tracking provide 2.17× and 1.57× speedup, respectively.

1 Introduction

Segment Anything Models (SAMs) (Kirillov et al., 2023) have emerged as widely adopted vision foundation models owing to their strong interactive segmentation capabilities. By introducing a memory mechanism, SAM2 (Ravi et al., 2024) extends promptable segmentation from images to videos, enabling robust tracking of geometrically referred objects. Building on this advancement, SAM3 (Carion et al., 2025) further generalizes prompts from object-specific cues to open-ended concepts, supporting arbitrary noun-phrase prompts for object identification and tracking. This capability relies on a dual-branch detection–tracking framework, where the detector grounds input concept and corrects semantic drift, while the tracker maintains consistent object identities across frames through memory-based mask propagation. Although this design delivers strong performance, it incurs substantial computational cost, as both branches rely on Transformer architectures with dense frame-by-frame processing (Kamath et al., 2021), resulting in significant overhead and latency that limit its applicability in real-world streaming scenarios.

To accelerate SAM1 and SAM2, many works have made progress. For example, (Zhou et al., 2025; Xiong et al., 2024) address structural redundancy through distillation, (Lv et al., 2024; Zhang et al., 2026a) alleviate precision redundancy through quantization, and (Zhang et al., 2026b) exploit temporal continuity to suppress SAM2's computational redundancy. However, SAM3, with its computation-intensive detection–tracking dual branches, suffers from more significant computational overhead and inference latency, and also exhibits personalized redundancy patterns: **redundant detection execution** and **redundant tracking memory**. As shown in Figure 1, detection contributes only at a few moments when capturing new objects and correcting tracking drift, while most of the time it provide unnecessary semantic information. Moreover, full memory provides

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.

negligible gains in scenarios with limited dynamics and interference, indicating that most memory frames are dispensable most of the time. Although previous methods have explored temporally efficient designs for traditional tracking systems (He et al., 2022; Luo et al., 2019), *the dual-branch of SAM3 poses new challenges for suppressing temporal redundancy*. On the one hand, the outputs exhibit strong correlation and coupling between detection and tracking due to their mutual association and disambiguation, where detector's false positives and tracker's drift can all lead to inconsistent object identities (IDs). On the other hand, the tracker exhibits strong temporal continuity and consistency, meaning that an error in a single frame can lead to cumulative error propagation over time.

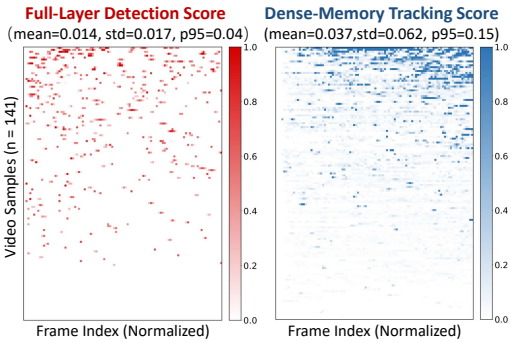

Figure 1: Temporal redundancy of detection execution and tracking memory. Detection score measures the contribution of detection, while Tracking score measures the contribution of dense memory. High scores exhibits temporal sparsity, revealing unnecessary computation. Video samples from the SA-V val set.

To this end, we conduct an in-depth analysis of SAM3's behavior and obtain the following insightful observations: ❶ **For the detector**, as illustrated in Figure 2 (left), the early layers already exhibit attention concentrated around relevant objects, closely aligning with and largely covering the final outputs. Moreover, by leveraging tracker signals such as tracking masks and confidence scores, it is possible to predict the emergence of newly appeared objects or potential tracking failures (referred to as entrants), thereby determining whether detection is necessary for the current frame. ❷ **For the tracker**, as illustrated in Figure 2 (right), compact memory is generally sufficient to produce reliable tracking results that remain consistent with those obtained using full memory, except in scenes with severe interference or highly dynamic motion. This suggests that dense full memory is often redundant, and its necessity can be adaptively determined according to the tracking difficulty.

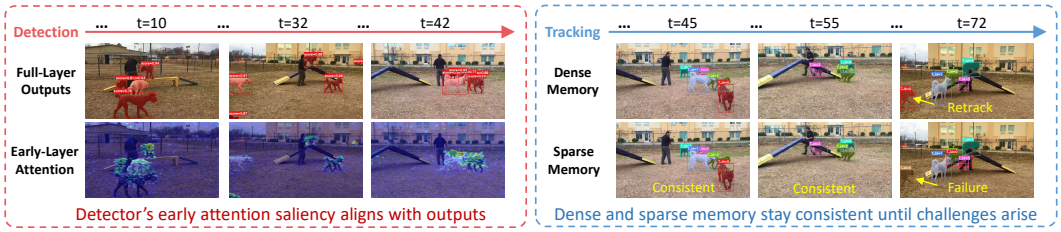

Figure 2: Detector's early attention can coarsely localize relevant objects (left). Dense memory is redundant except in challenging scenarios such as reappearance, occlusion, or fast motion (right).

With above insights, we propose FastSAM3, which accelerates SAM3 through lazy, on-demand execution of detection and tracking, achieving adaptive computational budget allocation. For the detector, we propose an Entrant-Predictive Gating (EPG) mechanism that leverage tracking signals to adaptively determines whether to skip detection or proceed with it. It uses tracking results to anchor already registered objects, and utilizes the detector's early responses to assess the presence of newly identified objects. Specifically, attention weights are used to extract salient regions as entrants proposals, and their similarity to the foreground is used to filter noise and measure semantic consistency. For the tracker, we propose a Divergence-Aware Gating (DAG) mechanism, which adaptively switches between sparse and dense memory modes according to tracking divergency. The tracker operates in a sparse memory mode by default and switches to a dense memory only when divergence occurs. Specifically, divergence is derived from three perspectives: temporal consistency, prediction discrepancy, and ID inconsistency. Our contributions are summarized as follows:

- We propose FastSAM3, an online adaptive inference acceleration framework for SAM3. It identifies temporal redundancy in detection execution and tracking memory, and accordingly proposes an adaptive, demand-driven computation allocation during inference, which is tailored to SAM3's dual-branch nature, thus preserving model performance.

- Building on these insights, we introduce dedicated lazy gating mechanisms for both detection and tracking. For detector, we use tracking signals and early-layer response to access the necessity of executing detection. For tracker, we adaptively switch between sparse memory and full memory based on tracking divergence and confidence.
- FastSAM3 achieves measured acceleration with minimal performance degradation across multiple benchmarks. By adaptively allocating computational overhead based on demand, it delivers an average 1.77× end-to-end speedup, with lazy detection and tracking contributing speedups of 2.17× and 1.57×, respectively.

2 Related Works

Segment Anything Model. SAMs (Kirillov et al., 2023) have emerged as prominent vision foundation models, owing to their exceptional general-purpose segmentation capability. SAM2 (Ravi et al., 2024; Yang et al., 2024) introduces a streaming memory mechanism to enable robust video object tracking through long-range temporal association. Building upon this, SAM3 (Carion et al., 2025) substantially advances semantic interaction to support the more challenging task of promptable concept segmentation. By adopting a collaborative detection-and-tracking dual-branch architecture, SAM3 demonstrates a higher performance ceiling in video referring segmentation and object tracking (Li et al., 2025; Zheng et al., 2025). Recently, SAM3 has attracted increasing attention from both academia and industry, with broad applications in video data annotation, multi-object tracking, and tool-augmented video understanding agents (Ye et al., 2025). However, both its detection and tracking branches rely on computation-intensive Transformer architectures, leading to substantial computational overhead and inference latency, which hinder its deployment in real-world scenarios.

SAM Acceleration Methods. A growing body of work has investigated model compression and inference acceleration techniques for SAM-series models. EdgeTAM (Zhou et al., 2025) and Efficient-TAM (Xiong et al., 2024) employ knowledge distillation to derive compact SAM2 variants with fewer parameters and lower computational cost. PTQ4SAM (Lv et al., 2024), SAQ-SAM (Zhang et al., 2026a), and Mix-QSAM (Ranjan & Savakis, 2025) instead adopt model quantization to compress the numerical representations of SAM2 parameters, reducing memory consumption and operator-level computational overhead. Despite notable compression and speedup, these methods typically incur segmentation performance degradation, limiting their ability to achieve a favorable efficiency-accuracy trade-off. Efficient-SAM2 (Zhang et al., 2026b) improves efficiency through object-aware dynamic computation scheduling, but it mainly exploits spatial and memory redundancy, leaving significant temporal redundancy in inference largely ignored.

Efficient Video Object Detection and Tracking. Given the stringent real-time requirements of visual tracking, extensive efforts have been devoted to improving the efficiency of conventional tracking systems. A representative direction follows the detect-on-key-frames and propagate-on-non-key-frames paradigm, reducing inference overhead by avoiding per-frame dense feature extraction and target matching, such as Deep Feature Flow (Zhu et al., 2017) and QueryProp (He et al., 2022). DorT (Luo et al., 2019) and Scale-Time Lattice (Chen et al., 2018) further introduce adaptive key-frame scheduling by considering dynamic cues such as object size, motion velocity, and tracking stability. These methods provide useful insights for reducing temporal redundancy in SAM3 inference. However, their designs are tailored to conventional object tracking, making them insufficient for the complex semantic referring demands of concept segmentation and less responsive to newly appearing objects under dynamic and occluded conditions.

3 Method

3.1 Preliminaries

Detection-Tracking Dual Branch in SAM3. The detector and tracker collaboratively achieve consistent object segmentation and identity association across video frames. Detector $\mathcal{F}_{\text{det}}$ performs concept-conditioned localization and segmentation on each frame, as follows:

$$\mathcal{D}_t = \mathcal{F}_{\text{det}}(F_{\text{img},t},\ P_{\text{text}}) = \left\{D_t^i\right\}_{i=1}^{N_t}, \tag{1}$$

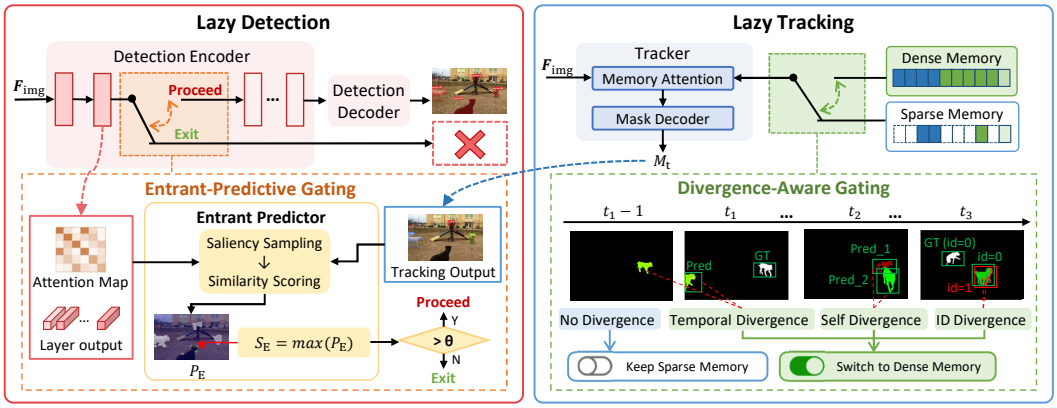

Figure 3: Overview of FastSAM3. For Lazy Detection, we propose Entrant-Predictive Gating, which leverage early responses and tracking signals to assess the likelihood of new object emergence, and then decide whether to proceed or exit inference, thereby reducing redundant detection execution. For Lazy Tracking, we propose Divergence-Aware Gating, which adaptively switches between sparse and dense memory modes by evaluating tracking reliability through temporal, prediction, and identity discrepancies, thereby reducing unnecessary dense memory attention computation.

where $F_{\text{img},t}$ denotes the visual feature extracted by the backbone, P_{text} denotes text prompt, and $D_t^i \in \mathcal{D}_t$ denotes the mask of the i-th detected object. Tracker $\mathcal{F}_{\text{track}}$ establishes inter-frame identity associations leveraging an attention-based memory mechanism:

$$\mathcal{M}_t = \mathcal{F}_{\text{track}}(F_{\text{img},t},\ \mathcal{H}_{t-1}) = \left\{M_t^i\right\}_{i=1}^{K_t}, \tag{2}$$

where $M_t^i \in \mathcal{M}_t$ denotes the mask of the i-th tracked object, and $\mathcal{H}_{t-1}$ is the memory feature from the previous frame. Next, the outputs of the tracker are associated with the detector's: $\mathcal{M}_t \rightarrow \mathcal{D}_t \cup \{\varnothing\}$, where matched tracking masks $\mathcal{M}_t^{\text{match}}$ are retained. The unmatched detections $\mathcal{D}_t^{\text{new}}$ are registered as new tracking identities, with final output masks $\mathcal{O}_t$ and memory update given by:

$$\mathcal{O}_t = \mathcal{M}_t^{\text{match}} \cup \mathcal{F}_{\text{track}}(F_{\text{img},t}, \mathcal{D}_t^{\text{new}}), \quad \mathcal{H}_t = \text{update}(\mathcal{H}_{t-1}, \mathcal{O}_t). \tag{3}$$

Overall, the detector serves as an auxiliary verifier, while the tracker provides the final outputs.

Response in Detector. The detector is composed of a fusion encoder and a DETR-like decoder. Fusion encoder $\mathcal{F}_{\text{fusion}}$ modulates visual features conditioned on the text prompt as follows:

$$F_{\text{fusion}} = \mathcal{F}_{\text{fusion}}(F_{\text{img},t}, F_{\text{text}}), \quad \mathcal{D}_t = \mathcal{F}_{\text{det}}(F_{\text{fusion}}, \{q_i\}), \tag{4}$$

where F_{fusion} is the text-conditioned image features, $\{q_i\}_{i=1}^{N_q}$ is a set of learnable queries that interacts with F_{fusion} to produce detection outputs. The Fusion Encoder consists of $L = 6$ cross-attention-based Transformer layers. At the l-th layer, the feature is updated as:

$$X_{\text{img}}^l = \text{CA}(\text{SA}(X_{\text{img}}^{l-1}), F_{\text{text}}), \tag{5}$$

where $\text{SA}(\cdot)$ denotes self-attention over visual tokens, and $\text{CA}(\cdot, F_{\text{text}})$ denotes image-to-text cross-attention. Interestingly, we observe that the self-attention in 2-th layer already signals object cues, which exhibits spatial saliency highly consistent with the final detection outputs.

Memory Bank in Tracker. The tracker in SAM3 employs a frame-level memory bank, where memory feature fusing historical image features with the corresponding tracking mask of each object. At moment t, $\mathcal{M}_t$ consists of conditioned frames $\mathcal{M}_t^c$, intermediate frames $\mathcal{M}_t^m$ and last frame. M_{t-1} (Zhou et al., 2024). Among them, $\mathcal{M}_t^c$ contain N_c periodic prompts from the detector, which serve to anchor the tracking targets; $\mathcal{M}_t^m$ correspond to the N_m most recent frames with high tracking confidence, providing spatiotemporal cues about object motion; M_{t-1} is included to suppress background interference. The memory representation at timestep t is formulated as follows:

$$\mathcal{M}_t = \mathcal{M}_t^c \oplus \mathcal{M}_t^m \oplus \{M_{t-1}\}, \tag{6}$$

where $\oplus$ denotes concatenation between sets. Under the default setting, N_c=4 and N_m=5, yielding a maximum memory length of 10 frames. Following memory modulation, the mask decoder generates the tracking mask, along with IoU score s_{iou} and occlusion score s_{obj}, which indicate the output quality and guide the selection of intermediate memory frames (Yang et al., 2024).

3.2 ENTRANTS-PREDICTIVE GATING FOR LAZY DETECTION

Based on the observed sparsity, the detector is required only when new objects emerge or tracking drift occurs. Therefore, the tracking output $\mathcal{M}_t$ naturally serves as a reliable indicator to focus on unexplored areas. Moreover, the attention saliency in early layer is highly consistent with the final detection output, providing cues for potential objects. With the above motivation, we propose **Entrants-Predictive Gating**, which stops at an early layer of detector to predict the likelihood of new object emergence leveraging the layer response and tracking signal, and then proceed with full inference only when the entrant score is high. Denote average self-attention weight $A^l \in \mathbb{R}^N$ from l-th fusion encoder layer, where N denotes the length of image tokens. The tracking mask $M_t \in \{0,1\}^N$ can partitions A^l into foreground and background regions:

$$A^l_{\text{fg}} = A^l \odot M_t, \qquad A^l_{\text{bg}} = A^l \odot (1 - M_t), \tag{7}$$

where $\odot$ denotes element-wise multiplication. Then,We respectively sample salient regions from the foreground and background, which is formulated as follows:

$$\mathcal{R}^l_{\text{fg}} = \left\{ \mathcal{N}_1(j) \,\middle|\, j \in \text{TopK}(A^l_{\text{fg}}, k_1) \right\}, \quad \mathcal{R}^l_{\text{bg}} = \left\{ \mathcal{N}_1(j) \,\middle|\, j \in \text{TopK}(A^l_{\text{bg}}, k_2) \right\}, \tag{8}$$

where $\text{TopK}(\cdot, k)$ selects the k largest elements, $\mathcal{N}_1(\cdot)$ denotes the one-hop spatial neighborhood.

However, we observe that the salient regions in the background not only cover semantically relevant objects but also include noisy responses, which may lead to false alarm in entrant prediction. To mitigate this issue, we leverage the layer output features to filter out regions with low similarity to the foreground, and take the maximum similarity score as the entrant score. Specifically, we first aggregate features from the foreground regions to form a prototype as follows:

$$f^l_{\text{fg}} = \frac{1}{\left|\mathcal{R}^l_{\text{fg}}\right|} \sum_{i \in \mathcal{R}^l_{\text{fg}}} X^l_{\text{img}}(i,:) \;\in\; \mathbb{R}^{1\times C}. \tag{9}$$

Denote the feature sets of background candidates as:

$$\mathcal{F}_{\text{bg}} = \left\{ X^l_{\text{img}}[r^{\text{bg}}_i, :] \;\middle|\; r^{\text{bg}}_i \in \mathcal{R}^l_{\text{bg}} \right\} = \left\{ f^i_{\text{bg}} \in \mathbb{R}^{N_i \times C} \right\}_{i=1}^{k_2}, \tag{10}$$

where f^i_{bg} denotes the feature of the i-th entrant candidate, containing N_i tokens. Then we compute the similarity between background candidates and foreground prototype, and take the maximum similarity as entrant score S_{E}, which is formulated as follows:

$$\mathcal{P}_{\text{E}} = \left\{ \text{sim}\big(f^i_{\text{bg}}, f_{\text{fg}}\big) \right\}_{i=1}^{k_2}, \quad S_{\text{E}} = \max \mathcal{P}_{\text{E}}, \tag{11}$$

where $\text{sim}(\cdot,\cdot)$ computes cumulative cosine similarity over tokens. Based on the entrant score and tracking stability, the condition of proceeding detector is as follows:

$$\text{Proceed} = \begin{cases} \text{False}, & \text{if } S_{\text{E}} < \theta \wedge s_{\text{obj}} > \tau, \\ \text{True}, & \text{otherwise.} \end{cases} \tag{12}$$

To enhance robustness across diverse scenarios and prompts, we also incorporate a base-frequency execution scheme with a dual-threshold setting, achieving a more favorable balance between efficiency and performance. Details are provided in Appendix A.1.

3.3 DIVERGENCE-AWARE GATING FOR LAZY TRACKING

In lazy tracking, we introduce **Divergence-Aware Gating**, which adopt sparse memory mode by default and switch to dense mode when tracking is unreliable. The reliability of tracking is evaluated based on divergences from temporal consistency, prediction confidence, and identity association.

Prediction divergence leverages the tracker's inherent distractor awareness, as identified by DAM4SAM (Videnovic et al., 2025). Specifically, the tracker predicts three candidate masks $\{C^k_t\}_{k=1}^3$ along with confidence scores $\{s^k_{\text{iou}} \in [0,1]\}_{k=1}^3$, and outputs the highest-confidence mask:

$$M_t = C^{k^*}_t, \quad k^* = \arg\max_{k \in \{1,2,3\}} s^k_{\text{iou}}. \tag{13}$$

It is observed that unselected masks tend to capture distractors or tracking drift, thus reflecting tracking consistency and scene interference. We therefore define the prediction divergence score as the discrepancy between the selected mask and the remaining candidates:

$$S_{\text{div}}^{\text{pred}} = \max_{k \in \{1,2,3\} \setminus \{k^*\}} \left(1 - \text{IoU}(M_t, C_t^k)\right), \tag{14}$$

where $\text{IoU}(\cdot,\cdot)$ denotes the intersection-over-union between masks. A higher $S_{\text{div}}^{\text{pred}}$ indicates stronger disagreement between the selected mask and alternative hypotheses, suggesting greater prediction ambiguity and the need for richer memory information to make reliable decisions.

Temporal divergence measures the spatio-temporal discontinuity of object motion. Given the three most recent tracking masks $\{M_{t-2}, M_{t-1}, M_t\}$, temporal divergence score is defined as:

$$S_{\text{div}}^{\text{temp}} = \phi_{\text{temp}}(M_{t-2}, M_{t-1}, M_t), \tag{15}$$

where ϕ_{temp} evaluates temporal inconsistency by jointly modeling motion continuity, area variation, and mask overlap across consecutive frames, which is formulated as follows:

$$\phi_{\text{temp}} = 1 - \left(w_{\text{mot}}\, \phi_{\text{mot}} + w_{\text{area}}\, \phi_{\text{area}} + w_{\text{overleap}}\, \phi_{\text{overlap}}\right), \tag{16}$$

$$\phi_{\text{mot}} = \exp\left(-\frac{\|c_t - \hat{c}_t\|_2^2}{2(\sigma^2 + \lambda v_t^2)}\right), \phi_{\text{area}} = \frac{\min(a_t, \hat{a}_t)}{\max(a_t, \hat{a}_t)}, \phi_{\text{overlap}} = \text{IoU}(M_{t-1}, M_t), \tag{17}$$

$$\hat{c}_t = 2c_{t-1} - c_{t-2}, \quad \hat{a}_t = 2a_{t-1} - a_{t-2}, \quad v_t = \|c_{t-1} - c_{t-2}\|_2, \tag{18}$$

where c_t and a_t denote centroid and area of M_t, respectively; σ and λ control motion tolerance; and $w_{\text{mot}}, w_{\text{area}}, w_{\text{overlap}}$ are weighting coefficients. Thereby, this reflects temporal inconsistency.

Identity divergence is defined as the mask overlap among different identities, indicating that at least one object is incorrectly tracked, which is formulated as:

$$S_{\text{div}}^{\text{id}} = \max_{i \neq j} \text{IoU}\left(M_t^i, M_t^j\right), \tag{19}$$

where i and j are different object identities. A higher $S_{\text{div}}^{\text{id}}$ indicates stronger identity conflict.

The memory remains in sparse mode and switches to dense mode only when any of the three divergence scores exceeds its threshold or the tracking confidence score s_{iou} become low, as follows:

$$\text{MemoryMode} = \begin{cases} \text{Dense}, & \text{if } \exists\, s \in \mathcal{S}_{\text{div}},\ s > \theta_s \ \vee\ s_{\text{iou}} < \theta_{\text{iou}}, \\ \text{Sparse}, & \text{otherwise}, \end{cases} \tag{20}$$

where $\mathcal{S}_{\text{div}} = \left\{S_{\text{div}}^{\text{pred}}, S_{\text{div}}^{\text{temp}}, S_{\text{div}}^{\text{id}}\right\}$. For the default sparse mode, we adopt a simple frame-level sparsification strategy as a baseline, referred to as SMF. Specifically, it slim memory by retaining several key frames: the two most recent conditioning frames, a highest-confidence intermediate frame, and the latest frame. Compared to the default full memory in SAM3, this achieves a compression ratio of 4/10. To avoid cumulative error propagation caused by tracking failures, a dense-mode fallback is applied to extremely unreliable frames. Notably, our method is compatible with other sparse memory strategies, such as MemPool (Xiong et al., 2024) and SMR (Zhang et al., 2026b), as demonstrated in the ablation studies.

4 Experiments

4.1 Experimental Settings

Datasets. The experiments are conducted on the SA-Co/VEval benchmark for Promptable Concept Segmentation (PCS). SA-Co/VEval consists of six subsets collected from three sources: SA-V, YT-Temporal-1B, and SmartGlasses. This benchmark provides a comprehensive evaluation of a model's ability to detect, segment, and track all relevant instances in videos based on textual prompts. Among them, SA-V focuses on challenging video segmentation scenarios with complex motion, occlusion, and multiple instances; YT-Temporal-1B mainly contains open-domain dynamic scenes from large-scale YouTube videos; and SmartGlasses is an egocentric video dataset featuring rapid camera motion, viewpoint changes, and fine-grained local observations.

Table 1: Performance on SA-Co/VEval benchmark validation split. We mainly report cgF1, pmF1, pHOTA and pDetA, more metric are included in Appendix A.3. In 'Det.' and 'Tr.' columns, '✓' indicates efficiency optimization for Detector (Det.) and Tracker (Tr.) components, while '✗' indicates original setting. Speedup as component-level end-to-end acceleration. Our methods achieve significant speedup while maintaining competitive performance with the original SAM3 models.

				YT-Temporal-1B val				SA-V val				SmartGlasses val			
Method	Det.	Tr.	Speedup	cgF_1	pmF_1	pHOTA	pDetA	cgF_1	pmF_1	pHOTA	pDetA	cgF_1	pmF_1	pHOTA	pDetA
SAM3	-	-	-	49.8	56.6	70.0	59.7	29.1	44.3	60.6	44.5	32.8	43.0	59.2	45.0
LD-Uniform	✓	✗	2.17x	48.1	54.9	68.8	57.7	26.8	43.6	58.8	41.6	30.9	40.5	57.7	42.7
LD-Random	✓	✗	2.17x	46.6	54.0	68.4	57.1	25.6	43.1	58.4	41.2	29.2	39.2	56.8	41.9
LD-EPG (ours)	✓	✗	2.17x	**49.7**	**55.7**	**69.7**	**59.1**	**28.6**	**43.6**	**60.4**	**44.1**	**32.0**	**40.9**	**58.9**	**44.8**
MemPool	✗	✓	1.35x	44.6	50.8	67.1	57.4	25.9	40.4	59.1	43.4	26.3	34.6	56.2	43.6
SMR	✗	✓	1.05x	48.0	54.8	69.6	59.3	27.6	42.4	60.0	44.2	31.5	41.4	59.1	45.3
SMF	✗	✓	1.64x	48.4	55.1	69.3	59.1	28.2	42.9	59.9	44.0	31.3	41.1	58.5	44.5
LT-Random	✗	✓	1.57x	48.7	55.6	69.6	59.4	28.2	43.0	60.0	44.0	31.6	41.5	58.5	44.9
LT-Uniform	✗	✓	1.57x	48.5	55.4	69.7	59.4	28.0	43.0	59.9	43.9	31.6	41.4	58.7	44.8
LT-DAG(ours)	✗	✓	1.57x	**49.4**	**56.3**	**70.1**	**59.9**	**28.9**	**43.9**	**60.6**	**44.6**	**32.3**	**42.3**	**59.3**	**45.4**
FastSAM3(ours)	✓	✓	1.77x	49.5	55.7	69.9	59.6	28.0	43.1	60.1	43.9	31.2	40.3	58.5	44.4

Table 2: Performance on SA-Co/VEval test split. More metric are reported in Appendix A.3.

				YT-Temporal-1B test				SA-V test				SmartGlasses test			
Method	Det.	Tr.	Speedup	cgF_1	pmF_1	pHOTA	pDetA	cgF_1	pmF_1	pHOTA	pDetA	cgF_1	pmF_1	pHOTA	pDetA
SAM3	-	-	-	50.0	56.4	69.4	59.6	30.3	43.7	57.8	40.6	35.8	45.9	62.9	49.1
LD-Uniform	✓	✗	2.17x	48.0	55.1	68.3	57.9	27.9	42.2	56.2	38.2	34.6	44.5	61.9	47.3
LD-Random	✓	✗	2.17x	46.0	53.8	67.4	56.7	27.8	41.7	56.2	38.3	33.4	43.6	61.5	47.0
LD-EPG (ours)	✓	✗	2.17x	**49.9**	**55.2**	**69.1**	**59.3**	**30.1**	**42.9**	**57.7**	**40.4**	**35.3**	**44.5**	**63.2**	**49.6**
MemPool	✗	✓	1.35x	45.6	51.6	66.6	57.7	26.7	38.9	56.2	39.5	29.1	37.3	60.0	47.4
SMR	✗	✓	1.05x	48.8	55.1	68.9	59.5	29.4	42.6	**57.7**	40.5	34.0	43.6	62.3	48.9
SMF	✗	✓	1.64x	48.5	54.8	68.5	59.1	29.2	42.1	57.3	40.3	33.9	43.4	61.7	48.4
LT-Random	✗	✓	1.57x	49.1	55.4	68.6	59.2	29.5	42.5	57.4	40.3	33.8	43.2	61.5	48.5
LT-Uniform	✗	✓	1.57x	48.8	55.1	68.7	59.1	29.6	42.6	57.4	40.4	34.1	43.7	61.9	48.6
LT-DAG(ours)	✗	✓	1.57x	**49.8**	**56.2**	**69.3**	**59.5**	**29.7**	**42.9**	57.6	**40.5**	**35.3**	**45.1**	**62.6**	**49.1**
FastSAM3(ours)	✓	✓	1.77x	49.7	54.9	69.1	59.4	29.7	42.3	57.5	40.5	34.2	43.1	62.5	49.0

Baselines. For lazy detection, we compare our method with two baselines: uniform detection (LD-Uniform), which performs detection at fixed intervals, and random detection (LD-Random), which triggers detection randomly with a predefined probability. For fairness, both baselines are configured to have detection frequencies aligned with our method. For lazy tracking, we evaluate three memory sparsification strategies, including token-level sparsification (SMR (Zhang et al., 2026b)), frame-level sparsification (SMF), and memory pooling (MemPool (Xiong et al., 2024)). Specifically, SMR sparsifies intermediate memory frames while keeping conditioning frames and the latest frame dense, resulting in a sparsity ratio of 0.48. SMF prunes memory frames, achieving a sparsity rate of 0.6. MemPool compresses memory through pooling operations, achieving a sparsity ratio of 0.75. We further compare different memory mode gating strategies, including random switching (LT-Random), which switches between dense and sparse memory modes with a predefined probability, and uniform switching (LT-Uniform), which periodically switches to dense memory mode at fixed intervals. The switching frequency is also aligned with our method to ensure a fair comparison.

Implementation Details. We largely adopt the default setting of original SAM3 and follow standard inference pipeline of PCS, with only several performance-irrelevant engineering modifications introduced to accommodate our method. For example, to leverage tracking signals to guide the execution of detection, we reorder the two decoupled branches. FastSAM3 also adopts the same heuristic disambiguation strategy as SAM3 during post-processing. The main difference is that the disambiguation parameters are dynamically adjusted according to the detection frequency. In addition, for frames where no object has been identified, FastSAM3 performs detection at a base frequency until an object is detected and the tracker is activated, after which the detection gating mechanism takes effect. The experiments are conducted on a single NVIDIA A6000 GPU and an NVIDIA A100 GPU. The speedup are tested on NVIDIA A6000 GPU, with a consistent and conservative sparsity ratio (0.7 in Lazy Detection and 0.9 in Lazy Tracking). The reported speedups are evaluated at the component level, measured by the end-to-end wall-clock time of the detector and tracker branches,

with the backbone computation excluded. For lazy detection, we place the exit gate after the second layer of the fusion encoder. The attention weights are derived from its self-attention computation, while the features used for similarity computation are taken from the output of the first layer.

4.2 Main Results

Quantitative Results. Table 1 and Table 2 present Results on SA-Co/VEval benchmark, on the validation and test splits, respectively. Our methods achieve substantial efficiency gains with minimal accuracy degradation. Applying lazy optimization only to the detector (LD-EPG) yields a 2.17× speedup while retaining performance close to the original model. On the validation sets, LD-EPG achieves cgF1 scores of 49.7, 28.6, and 32.0 on YT-Temporal-1B, SA-V, and SmartGlasses, respectively, compared with 49.8, 29.1, and 32.8 for SAM3. Moreover, LD-EPG consistently outperforms uniform and random sampling (e.g., pHOTA: 69.7 vs. 68.8/68.4), demonstrating that policy-based keyframe selection effectively reduces detection frequency while preserving semantic accuracy.

When applying only spatial memory sparsification (MemPool, SMR, SMF), although the speedup is considerable (e.g., SMF reaches 1.64×), the performance degradation is more pronounced. For instance, on YT-Temporal-1B val, cgF1 drops from 49.8 to 44.6 with MemPool, and to 48.4 with SMF. In contrast, with lazy tracking, LT-DAG (ours) achieves cgF1 = 49.3 and pHOTA = 70.1 under a 1.57× speedup, nearly matching the baseline (70) while clearly outperforming both random and uniform strategies. These results suggest that naive sparsification or random/uniform switching disrupts temporal modeling, whereas DAG effectively preserves long-term temporal consistency while reducing computation. Combining Lazy Detection and Lazy Tracking (FastSAM3) further achieves a 1.7× speedup while maintaining stable performance close to the baseline.

Qualitative Results. In Figure 4, we present several qualitative results. It can be observed that FastSAM3 keeps the model in a low-computation lazy mode and activates full inference only when necessary, achieving substantial acceleration. For most frames, our method incurs significantly lower latency than the original model. For lazy detection, the model promptly captures newly entering objects (entrants) and triggers full inference when needed. It can also react in time to tracking failures, effectively correcting drift. For lazy tracking, sparse memory is sufficient during stable tracking phases; when divergence occurs, the system switches to full memory to ensure comprehensive spatiotemporal interactions. Admittedly, newly appeared objects are often only partially visible upon entry, leading to weak features. As a result, lazy detection may miss them for one frame (e.g., t=32), but typically recovers in the next frame (t=33) as more informative features become available.

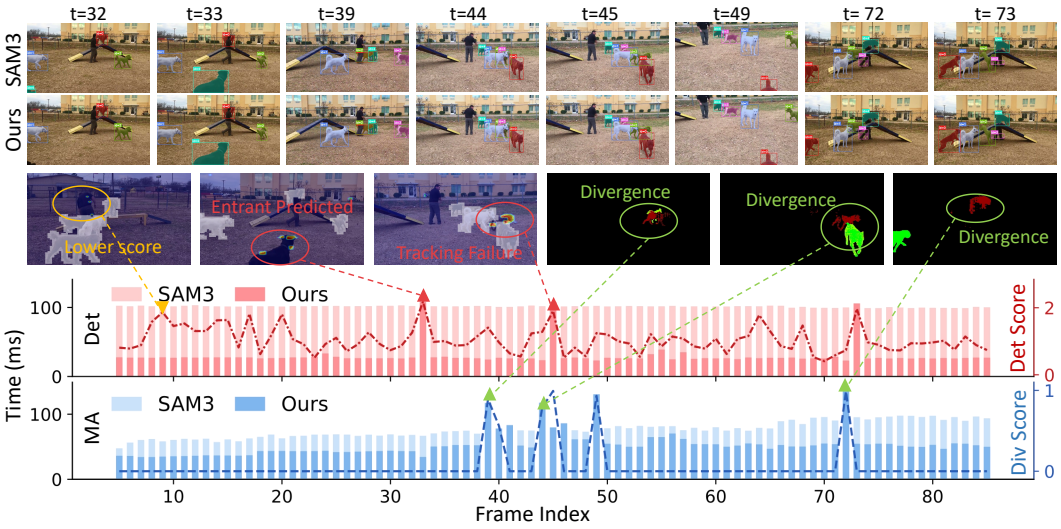

Figure 4: Qualitative Results. Compared with the original SAM3, our method significantly reduced inference latency with nearly lossless performance. the detector is fully executed only on several frames to capture new objects and recalibrate tracking, while the tracker switches to dense memory when divergence is detected to get richer historical information.

4.3 Ablation Study

Sparsity Analysis. FastSAM3, as an adaptive computation budget allocation framework for address temporal redundancy, achieves acceleration gains that depend on the complexity of the task. The results reported in this paper achieve measurable speedups with only marginal performance degradation, corresponding to a detection sparsity ratio of 0.7 and a tracking sparsity ratio of 0.9. To better show the efficiency gain of our method across different datasets, we report the average sparsity ratios in Table 3. Despite the high difficulty of these datasets, our method is still able to achieve high sparsity ratios while introducing only minimal impact on performance.

Table 3: Cross-dataset sparsity analysis. Our method achieves favorable sparsity while maintaining consistently strong performances.

Datasets	# Prompt	LD sparsity	LT sparsity
SA-V val	1454	0.61	0.91
SA-V test	1504	0.62	0.91
YT-Temporal-1B val	1422	0.63	0.98
YT-Temporal-1B test	1571	0.63	0.82
SmartGlasses val	2327	0.63	0.92
SmartGlasses test	2264	0.61	0.92

Ablation for Lazy Detection. Figure 5 compares the performance of LD-EPG (ours) with uniform and random detection strategies under different sparsity ratios on the YT-Temporal-1B validation set. We control the sparsity by adjusting the threshold settings (Single- or Dual-Threshold) and base frequency in EPG; however, the resulting sparsity is still inherently influenced by the characteristics of the dataset. Overall, our method consistently outperforms uniform and random strategies across different sparsity ratios. Notably, it maintains performance close to the full SAM3 model under moderate sparsity (0.4–0.6), demonstrating a superior trade-off between efficiency and accuracy.

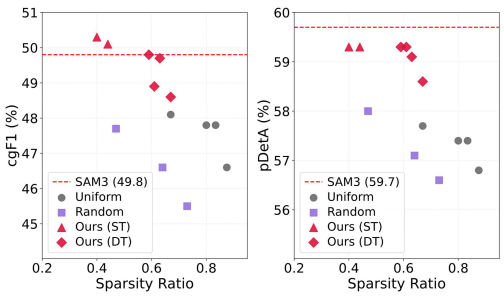

Figure 5: Performance of lazy detection with different settings. Our LD-EPG achieve excellent performance-speedup trade-offs.

Ablation for Lazy Tracking. Table 4 presents the ablation study of the three divergence components in DAG, including temporal divergence (Temp Div), prediction divergence (Pred Div), and identity divergence (ID Div), on the YT-Temporal-1B test set. All three components contribute positively to performance, with temporal and identity divergence providing consistent gains and prediction divergence delivering the most significant improvement. When all components are combined, the model obtains the best overall results. Table 5 shows that our Lazy Tracking is compatible with other memory comparison methods (e.g., SMR, MemPool) and consistently improves performance.

Table 4: DAG ablation. All three divergence scores contribute to performance.

Temp Div	ID Div	Pred Div	cgF1	pmF1	pHOTA	pDetA
✗	✗	✗	48.5	54.8	68.5	59.1
✓	✗	✗	49.1	55.4	68.8	59.3
✗	✓	✗	48.9	55.2	68.8	59.3
✗	✗	✓	49.8	56.2	69.2	59.5
✓	✓	✓	**49.8**	**56.2**	**69.3**	**59.5**

Table 5: Compatibility test for DAG. It supports other memory compression methods.

Method	cgF1	pmF1	pHOTA	pDetA
SMR	48.8	55.1	68.9	59.5
+DAG(ours)	49.6 ↑0.8	56 ↑0.9	69.3 ↑0.4	59.5 ↑0.0
MemPool	45.6	51.6	66.6	57.7
+DAG(ours)	48.3 ↑2.7	54.6 ↑3.0	68.7 ↑2.1	59.2 ↑1.5

5 Conclusion

In this work, we revisit the efficiency bottlenecks of SAM3 for video referring segmentation and demonstrate that a substantial portion of computation in both detection and tracking becomes unnecessary under temporally stable conditions. Motivated by this observation, we propose FastSAM3, a training-free adaptive inference framework that allocates computation according to scene dynamics and tracking uncertainty. FastSAM3 employs Entrant-Predictive Gating for selective detector activation and Divergence-Aware Gating for adaptive memory utilization. Extensive experiments demonstrate a favorable efficiency–accuracy trade-off without architectural modifications. Although the achievable acceleration is data- and difficulty-dependent, our results highlight the potential of computation-aware temporal modeling for efficient video segmentation.

AI USE STATEMENT

We have not used generative AI tools for any tasks requiring or recommending disclosure. All other disclosure categories are not applicable to this work. The authors take full responsibility for the final content of this work.

REFERENCES

Hexin Bai, Wensheng Cheng, Peng Chu, Juehuan Liu, Kai Zhang, and Haibin Ling. Gmot-40: A benchmark for generic multiple object tracking. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6715–6724. IEEE, 2021.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

Kai Chen, Jiaqi Wang, Shuo Yang, Xingcheng Zhang, Yuanjun Xiong, Chen Change Loy, and Dahua Lin. Optimizing video object detection via a scale-time lattice. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7814–7823, 2018.

Yuzhe Duan, Xuanxuan Ren, Guizhe Dong, Xu Yang, and Yanhua Yang. Mix-qsam2: Mixed-precision quantization for high fidelity segmentation in resource constrained scenarios. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 3741–3749, 2026.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642, 2024.

Fei He, Naiyu Gao, Jian Jia, Xin Zhao, and Kaiqi Huang. Queryprop: Object query propagation for high-performance video object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 834–842, 2022.

Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1780–1790, 2021.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4015–4026, 2023.

Jingjing Li, Yue Feng, Yuchen Guo, Jincai Huang, Wei Ji, Qi Bi, Yongri Piao, Miao Zhang, Xiaoqi Zhao, Qiang Chen, et al. Sam3-i: Segment anything with instructions. *arXiv preprint arXiv:2512.04585*, 2025.

Hao Luo, Wenxuan Xie, Xinggang Wang, and Wenjun Zeng. Detect or track: Towards cost-effective video object detection/tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8803–8810, 2019.

Chengtao Lv, Hong Chen, Jinyang Guo, Yifu Ding, and Xianglong Liu. Ptq4sam: Post-training quantization for segment anything. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 15941–15951, 2024.

Patrick Poggi, Divake Kumar, Theja Tulabandhula, and Amit Ranjan Trivedi. Uncertainty-guided inference-time depth adaptation for transformer-based visual tracking. *arXiv preprint arXiv:2602.16160*, 2026.

Navin Ranjan and Andreas Savakis. Mix-qsam: Mixed-precision quantization of the segment anything model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3305–3315, 2025.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

Jovana Videnovic, Alan Lukezic, and Matej Kristan. A distractor-aware memory for visual object tracking with sam2. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 24255–24264, 2025.

Yunyang Xiong, Chong Zhou, Xiaoyu Xiang, Lemeng Wu, Chenchen Zhu, Zechun Liu, Saksham Suri, Balakrishnan Varadarajan, Ramya Akula, Forrest Iandola, et al. Efficient track anything. *arXiv preprint arXiv:2411.18933*, 2024.

Cheng-Yen Yang, Hsiang-Wei Huang, Wenhao Chai, Zhongyu Jiang, and Jenq-Neng Hwang. Samurai: Adapting segment anything model for zero-shot visual tracking with motion-aware memory. *arXiv preprint arXiv:2411.11922*, 2024.

Kai Ye, Xiaotong You, Jianghang Lin, Jiayi Ji, Pingyang Dai, and Liujuan Cao. Evolving, not training: Zero-shot reasoning segmentation via evolutionary prompting. *arXiv preprint arXiv:2512.24702*, 2025.

Jing Zhang, Zhikai Li, Chengzhi Hu, Xuewen Liu, and Qingyi Gu. Saq-sam: Semantically-aligned quantization for segment anything model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 12547–12555, 2026a.

Jing Zhang, Zhikai Li, Xuewen Liu, and Qingyi Gu. Efficient-sam2: Accelerating sam2 with object-aware visual encoding and memory retrieval. *arXiv preprint arXiv:2602.08224*, 2026b.

Libo Zhang, Junyuan Gao, Zhen Xiao, and Heng Fan. Animaltrack: A benchmark for multi-animal tracking in the wild. *International Journal of Computer Vision*, 131(2):496–513, 2023.

Xiangqing Zheng, Chengyue Wu, Kehai Chen, and Min Zhang. Locot2v-bench: Benchmarking long-form and complex text-to-video generation. *arXiv preprint arXiv:2510.26412*, 2025.

Chong Zhou, Chenchen Zhu, Yunyang Xiong, Saksham Suri, Fanyi Xiao, Lemeng Wu, Raghuraman Krishnamoorthi, Bo Dai, Chen Change Loy, Vikas Chandra, et al. Edgetam: On-device track anything model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 13832–13842, 2025.

Junbao Zhou, Ziqi Pang, and Yu-Xiong Wang. Rmem: Restricted memory banks improve video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18602–18611, 2024.

Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2349–2358, 2017.

A FastSAM3: Supplementary Material

A.1 Base Frequency Design of Lazy Detection

We observe that a single-threshold design cannot effectively balance efficiency and accuracy under a unified threshold. To address this, we introduce a baseline-frequency gating mechanism with dual thresholds. Specifically, we define a base operating frequency for the detector. When the current frame is in the True state, a lower threshold θ_{low} is applied: if S_{E} is below the threshold and tracking is stable, the state switches to False; otherwise, it remains True. Conversely, when the current state is False, a higher threshold θ_{high} is used: if S_{E} exceeds the threshold, the state switches to True; otherwise, it remains False. By default, we set the base frequency to 3. For datasets dominated by large targets (YT-Temporal-1B, SmartGlasses), we set [θ_{low},θ_{high}]=[1.8,2.5], while for datasets dominated by small targets (SA-V), we set them to [2,3].

A.2 Speed Test Details

Our implementation is based on the initial version of SAM3 rather than the optimized SAM3.1 pipeline. The speedups reported in the main paper are evaluated on a moderately dynamic 40-frame video sample containing six targets. Each experiment is run for three consecutive inference passes, and the runtime is measured using the final run. The reported runtime corresponds to component-level end-to-end latency, excluding backbone computation; specifically, it measures the complete execution time of the detector and tracker branches with the backbone separated out. Under this setting, FastSAM3 achieves a detector sparsity rate of 0.7 and a tracker sparsity rate of 0.9. Specifically, LD-EPG reduces the Detector latency by 2.17×, while LT-DAG accelerates the memory interaction module by 1.57×. We provide a detailed latency breakdown including the backbone computation below. For FastSAM3, the component-level speedup (Detector + Tracker) is 1.77×, while the full end-to-end pipeline achieves a 1.40× speedup when including feature extraction. Note that these statistics are computed only on samples containing tracking targets; negative samples, where no target appears in the video and the model is evaluated for false positives, are excluded from the analysis.

Table 6: Detailed latency breakdown including the backbone computation

Time (ms)	Backbone	Detector	Tracker-Memory	Tracker-Decoder	Total	Detector+Tracker
SAM3	98	100	77	23	317	200
LD-EPG	95	46 (**2.17**×)	72	17	249	135
LT-DAG	95	101	49 (**1.57**×)	27	291	177
FastSAM3	94	42	49	22	226 (**1.40**×)	113 (**1.77**×)

A.3 More Metrics in Main Results

Tables 7, 8, and 9 report additional evaluation metrics across different datasets, where our method consistently demonstrates superior performance.

Table 7: Results on YT-Temporal-1B dataset.

	YT-Temporal-1B val							YT-Temporal-1B test						
Method	cgF_1	VL_MCC	pmF_1	pHOTA	pDetA	pAssA	TETA	cgF_1	VL_MCC	pmF_1	pHOTA	pDetA	pAssA	TETA
SAM3	49.8	0.88	56.6	70	59.7	82.8	70.4	50	0.89	56.4	69.4	59.6	81.4	70.2
LD-Uniform	48.1	0.87	54.9	68.8	57.7	82.7	69.6	48.0	0.87	55.1	68.3	57.9	81.1	**70.5**
LD-Random	46.6	0.86	54	68.4	57.1	82.7	69.1	46.0	0.85	53.8	67.4	56.7	80.6	69.6
LD-EPG (ours)	**49.7**	**0.89**	**55.7**	**69.7**	**59.1**	**82.9**	**70.2**	**49.9**	**0.9**	**55.2**	**69.1**	**59.3**	**81.1**	70.3
MemPool	44.6	0.88	50.8	67.1	57.4	79.2	67.7	45.6	0.88	51.6	66.6	57.7	77.5	67.3
SMR	48.0	0.88	54.8	69.6	59.3	82.2	69.7	48.8	0.89	55.1	68.9	59.5	80.6	69.6
SMF	48.4	0.88	55.1	69.3	59.1	81.9	69.6	48.5	0.89	54.8	68.5	59.1	79.9	69.6
LT-Random	48.7	0.88	55.6	69.6	59.4	82.3	69.7	49.1	0.89	55.4	68.6	59.2	80.1	69.7
LT-Uniform	48.5	0.88	55.4	69.7	59.4	82.4	69.9	48.8	0.89	55.1	68.7	59.1	80.4	69.6
LT-DAG(ours)	**49.4**	**0.88**	**56.3**	**70.1**	**59.9**	**82.7**	**70.2**	**49.8**	**0.89**	**56.1**	**69.3**	**59.5**	**81.3**	**70.1**
FastSAM3(ours)	49.5	0.89	55.7	69.9	59.6	82.5	70.5	49.7	0.91	54.9	69.1	59.4	81	70.4

Table 8: Results on SA-V dataset.

	SA-V val							SA-V test						
Method	cgF_1	VL_MCC	pmF_1	pHOTA	pDetA	pAssA	TETA	cgF_1	VL_MCC	pmF_1	pHOTA	pDetA	pAssA	TETA
SAM3	29.1	0.66	44.3	60.6	44.5	83.2	57.1	30.3	0.69	43.7	57.8	40.6	83.3	56.5
LD-Uniform	26.8	0.62	43.6	58.8	41.6	**83.9**	**57.5**	27.9	0.66	42.2	56.2	38.2	**83.9**	56.6
LD-Random	25.6	0.59	43.1	58.4	41.2	83.8	57.4	27.8	0.67	41.7	56.2	38.3	83.7	**56.7**
LD-EPG (ours)	**28.6**	**0.66**	**43.6**	**60.4**	**44.1**	83.4	56.1	**30.1**	**0.70**	**42.9**	**57.7**	**40.4**	83.4	56.5
MemPool	25.9	0.64	40.4	59.1	43.4	81.4	56.3	26.7	0.69	38.9	56.2	39.5	81	55.5
SMR	27.6	0.65	42.4	60.0	44.2	82.3	56.6	29.4	0.69	42.6	**57.7**	40.5	**83.4**	56.7
SMF	28.2	0.66	42.9	59.9	44.0	82.4	56.9	29.2	0.69	42.1	57.3	40.3	82.4	56.8
LT-Random	28.2	0.66	43.0	60.0	44.0	82.5	56.7	29.5	0.69	42.5	57.4	40.3	82.7	56.9
LT-Uniform	28.0	0.65	43.0	59.9	43.9	82.6	56.7	29.6	0.69	42.6	57.4	40.4	82.8	56.8
LT-DAG(ours)	**28.9**	**0.66**	**43.9**	**60.6**	**44.6**	**83.0**	**57.1**	**29.7**	**0.69**	**42.9**	57.6	**40.5**	82.9	**57.0**
FastSAM3(ours)	28.0	0.65	43.1	60.1	43.9	83.1	56.4	29.7	0.70	42.3	57.5	40.5	82.7	56.8

Table 9: Results on SmartGlasses dataset.

	SmartGlasses val							SmartGlasses test						
Method	cgF_1	VL_MCC	pmF_1	pHOTA	pDetA	pAssA	TETA	cgF_1	VL_MCC	pmF_1	pHOTA	pDetA	pAssA	TETA
SAM3	32.8	0.76	43.0	59.2	45.0	78.6	64.3	35.8	0.78	45.9	62.9	49.1	81.1	65.4
LD-Uniform	30.9	0.76	40.5	57.7	42.7	**78.6**	**63.0**	34.6	0.78	44.5	61.9	47.3	**81.6**	64.6
LD-Random	29.2	0.74	39.2	56.8	41.9	77.7	62.3	33.4	0.77	43.6	61.5	47.0	81.0	**65.1**
LD-EPG (ours)	**32.0**	**0.78**	**40.9**	**58.9**	**44.8**	78.1	62.7	**35.3**	**0.79**	**44.5**	**63.2**	**49.6**	81.3	64.5
MemPool	26.3	0.76	34.6	56.2	43.6	73.0	62.4	29.1	0.78	37.3	60.0	47.4	76.4	63.2
SMR	31.5	0.76	41.4	59.1	45.3	77.9	64.1	34.0	0.78	43.6	62.3	48.9	80.0	64.8
SMF	31.3	0.76	41.1	58.5	44.5	77.6	64.2	33.9	0.78	43.4	61.7	48.4	79.2	64.7
LT-Random	31.6	0.76	41.5	58.5	44.9	77.0	64.1	33.8	0.78	43.2	61.5	48.5	78.6	64.8
LT-Uniform	31.6	0.76	41.4	58.7	44.8	77.6	64.1	34.1	0.78	43.7	61.9	48.6	79.6	65.0
LT-DAG(ours)	**32.3**	**0.76**	**42.3**	**59.3**	**45.4**	**78.2**	**64.4**	**35.3**	**0.78**	**45.1**	**62.6**	**49.1**	**80.5**	**65.5**
FastSAM3(ours)	31.2	0.78	40.3	58.5	44.4	77.5	63.0	34.2	0.79	43.1	62.5	49.0	80.4	64.2

A.4 Comparison with more baseline methods

Efficiency optimization for SAM3 remains relatively under-explored, and directly comparable methods are still limited. In addition to the scheduling and memory sparsification baselines considered in the main paper, we further compare FastSAM3 with recent adaptive inference and memory compression strategies.

Adaptive inference for detector. We introduce three additional baselines for adaptive detector computation. **DG-AU** performs detection gating based on attention uncertainty (Poggi et al., 2026). **DLS-AU** applies uncertainty-guided adaptive layer skipping following the layer-skipping paradigm (Elhoushi et al., 2024). Specifically, **DLS-AU**$_{\mathrm{E}}$ skips layers only in the encoder, while **DLS-AU**$_{\mathrm{ED}}$ applies adaptive skipping to both the encoder and decoder.

Memory sparsification for tracker. For tracker-side memory optimization, we additionally reproduce the frame-level memory merging strategy in Mix-QSAM2 (Duan et al., 2026), denoted as **MemMerge**.

As shown in Table 10, LD-EPG consistently preserves higher segmentation and tracking accuracy than DG-AU and DLS-AU on both YT-1B and SA-V. In particular, uncertainty-based detector gating and aggressive layer skipping can lead to noticeable performance degradation, whereas LD-EPG maintains performance close to the original SAM3. On the tracker side, LT-DAG also outperforms MemMerge across the reported metrics, demonstrating the effectiveness of divergence-aware memory adaptation compared with direct frame-level memory merging.

Table 10: Comparison with adaptive inference and memory sparsification baselines.

Dataset	Method	cgF1	pmF1	pHOTA	pDetA
YT-1B val	SAM3	49.8	56.6	70.0	59.7
	DG-AU	42.9	49.4	64.8	50.4
	DLS-AU$_{\text{E}}$	49.5	52.8	66.7	53.2
	DLS-AU$_{\text{ED}}$	36.1	38.3	62.9	48.3
	LD-EPG (ours)	**49.7**	**55.7**	**69.7**	**59.1**
	MemMerge	44.5	50.7	66.7	58.0
	LT-DAG (ours)	**49.4**	**56.3**	**70.1**	**59.9**
SA-V val	SAM3	29.1	44.3	60.6	44.5
	DG-AU	25.4	42.2	58.3	40.8
	DLS-AU$_{\text{E}}$	23.5	38.8	57.2	37.7
	DLS-AU$_{\text{ED}}$	21.3	35.2	56.1	36.6
	LD-EPG (ours)	**28.2**	**43.5**	**60.3**	**44.0**
	MemMerge	27.4	41.8	59.6	44.1
	LT-DAG (ours)	**28.9**	**43.9**	**60.6**	**44.6**

A.5 More Temporal Redundancy Visualizations

Figures 6, 7, and 8 illustrate the sparsity of detector activation and dense-memory tracking contribution scores across different datasets.

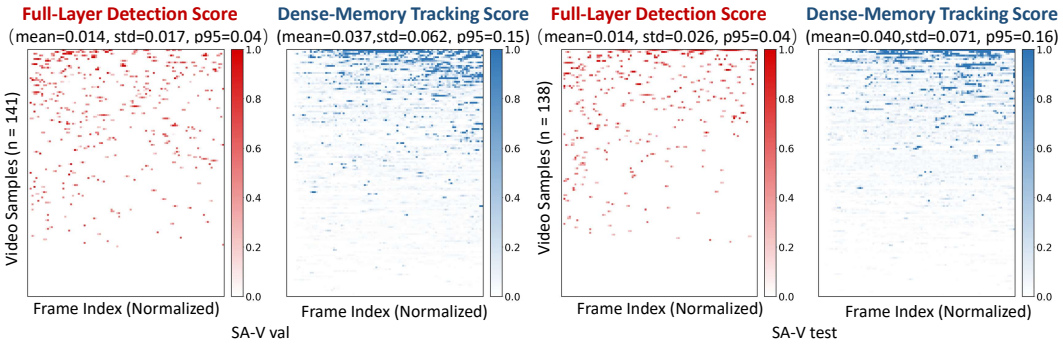

Figure 6: Temporal Sparsity of SA-V datasets.

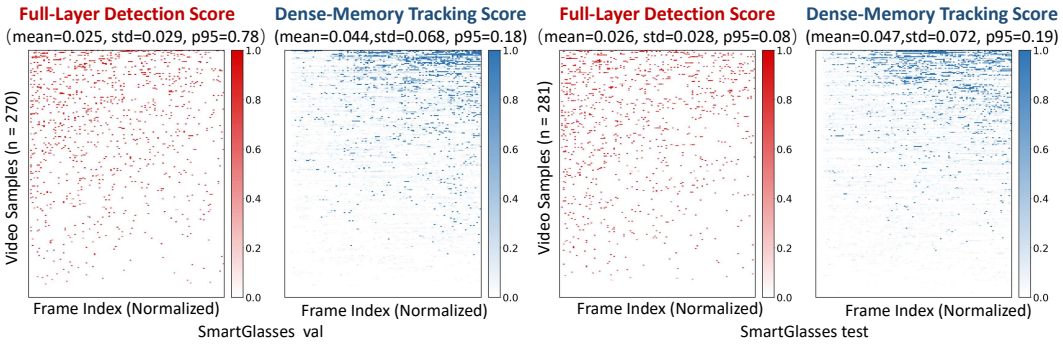

Figure 7: Temporal Sparsity of SA-V datasets.

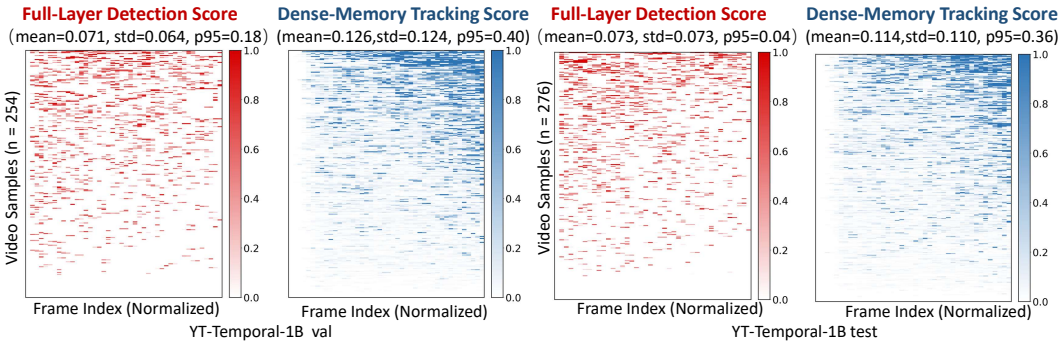

Figure 8: Temporal Sparsity of SA-V datasets.

A.6 The Visualization Early-Layer Attention

Figure 9 shows the attention heatmaps of different layers in the fusion encoder.



Figure 9: Self-attention heatmaps of different layers in the fusion encoder. Starting from the second layer, the attention begins to concentrate on relevant targets.

A.7 Performance on Videos with Multiple Similar Objects

To further evaluate the robustness of FastSAM3 in challenging scenarios containing multiple densely distributed and visually similar objects, we conduct additional experiments on the GMOT (Bai et al., 2021) and AnimalTrack(Zhang et al., 2023) datasets. These benchmarks provide complementary settings for assessing whether the proposed dynamic computation mechanisms can preserve tracking performance under increased object ambiguity and interaction.

The results are reported in Tables 11 and 12. On GMOT, FastSAM3 achieves a HOTA of 61.51, slightly higher than SAM3 (61.20), while on AnimalTrack, it achieves 59.44 compared with 59.89 for SAM3. The individual LD-EPG and LT-DAG modules also consistently preserve competitive performance across both datasets, demonstrating the robustness of the proposed computation-saving strategies in scenes with multiple similar objects.

Table 11: Performance on the GMOT.

Method	Component	HOTA	DetA	AssA
SAM3	-	61.20	53.00	72.61
LD-Random	Detector	61.13	52.38	73.28
LD-Uniform	Detector	60.77	52.32	72.55
LD-EPG (ours)	Detector	**61.39**	**53.12**	**72.85**
MemPool	Tracker	53.15	45.46	63.65
SMR	Tracker	60.85	52.92	71.90
SMF	Tracker	60.40	52.30	71.62
LT-Uniform	Tracker	60.65	52.45	72.02
LT-DAG (ours)	Tracker	**61.18**	**53.09**	**72.40**
FastSAM3 (ours)	Detector+Tracker	61.51	53.43	72.72

Table 12: Performance on the AnimalTrack.

Method	Component	HOTA	DetA	AssA
SAM3	-	59.89	54.19	67.91
LD-Random	Detector	58.93	52.72	67.67
LD-Uniform	Detector	58.96	52.77	67.63
LD-EPG (ours)	Detector	**59.42**	**53.32**	**68.00**
MemPool	Tracker	56.20	50.60	64.07
SMR	Tracker	57.86	52.40	65.59
SMF	Tracker	58.71	53.07	66.52
LT-Uniform	Tracker	59.00	53.27	67.00
LT-DAG (ours)	Tracker	**59.87**	**54.28**	**67.74**
FastSAM3 (ours)	Detector+Tracker	59.44	53.61	67.66

A.8 Hyperparameter Sensitivity Analysis

We further investigate the sensitivity of the hyperparameters introduced in LD-EPG and LT-DAG. Specifically, we evaluate a wider range of threshold settings and analyze their effects on both model performance and computation sparsity.

Table 13: Sensitivity analysis of the dual thresholds $(\theta_{\text{low}}, \theta_{\text{high}})$ in LD-EPG. The underlined entries denote the configurations adopted in the corresponding experiments.

Dataset	Method	$(\theta_{\text{low}}, \theta_{\text{high}})$	Sparsity	cgF1	pmF1	pHOTA	pDetA
YT-1B val	SAM3	-	0.00	49.8	56.6	70.0	59.7
	LD-EPG (ours)	$\underline{(\mathbf{1.8}, \mathbf{2.5})}$	0.63	49.7	55.7	69.7	59.1
		$(1.8, 2.2)$	0.59	49.9	56.1	69.8	59.1
		$(1.8, 2.8)$	0.68	48.6	54.9	69.2	58.4
		$(1.8, 3.0)$	0.68	48.7	55.1	69.1	58.2
		$(1.2, 2.5)$	0.63	50.2	56.1	70.1	59.8
		$(1.5, 2.5)$	0.63	49.5	55.7	69.6	59.1
YT-1B test	SAM3	-	0.00	50.0	56.4	69.4	59.6
	LD-EPG (ours)	$\underline{(\mathbf{1.8}, \mathbf{2.5})}$	0.63	49.9	55.2	69.1	59.3
		$(1.8, 3.0)$	0.64	49.5	55.2	69.1	59.2
		$(1.8, 2.0)$	0.57	49.9	55.4	69.2	59.5
SA-V val	SAM3	-	0.00	29.1	44.3	60.6	44.5
	LD-EPG (ours)	$(\mathbf{1.8}, \mathbf{2.5})$	0.50	29.2	43.3	60.5	44.4
		$\underline{(2.0, 3.0)}$	0.61	28.6	43.6	60.4	44.1
		$(2.0, 3.5)$	0.63	28.2	43.5	60.3	44.0
		$(1.5, 3.5)$	0.48	29.4	44.1	60.9	44.9
		$(2.5, 3.5)$	0.65	27.5	43.6	60.0	43.4
SA-V test	SAM3	-	0.00	30.3	43.7	57.8	40.6
	LD-EPG (ours)	$(\mathbf{1.8}, \mathbf{2.5})$	0.49	30.3	42.7	57.8	40.5
		$\underline{(2.0, 3.5)}$	0.62	30.1	42.9	57.7	40.4
		$(2.0, 3.0)$	0.58	30.4	42.8	57.8	40.6
		$(2.0, 4.0)$	0.53	30.5	42.6	57.7	40.6
		$(1.5, 3.5)$	0.48	30.8	42.8	57.9	40.7
		$(1.8, 3.0)$	0.47	31.0	43.0	57.7	40.5
SmartGlasses val	SAM3	-	0.00	32.8	43.0	59.2	45.0
	LD-EPG (ours)	$(\mathbf{1.8}, \mathbf{2.5})$	0.58	32.1	40.9	59.2	45.1
		$\underline{(2.0, 3.0)}$	0.63	32.0	40.9	58.9	44.8
		$(2.0, 3.5)$	0.64	31.2	40.4	58.5	44.2
		$(1.6, 2.2)$	0.59	31.9	40.0	59.1	45.1
		$(1.6, 2.0)$	0.58	31.7	39.9	58.8	44.7
SmartGlasses test	SAM3	-	0.00	35.8	45.9	62.9	49.1
	LD-EPG (ours)	$(\mathbf{1.8}, \mathbf{2.5})$	0.59	35.9	45.2	63.0	49.3
		$\underline{(1.8, 3.0)}$	0.61	35.3	44.5	63.2	49.6
		$(2.0, 2.8)$	0.61	35.0	44.3	62.7	48.6
		$(2.0, 3.0)$	0.63	34.7	44.0	62.4	48.3
		$(1.5, 3.5)$	0.76	33.0	42.9	61.6	47.1

Sensitivity of LD-EPG. LD-EPG employs a dual-threshold mechanism, parameterized by $(\theta_{\text{low}}, \theta_{\text{high}})$, to control the aggressiveness of detector gating. Different threshold configurations provide different trade-offs between computation sparsity and prediction performance.

Table 13 reports the results under a wide range of threshold settings on YT-1B, SA-V, and SmartGlasses. Overall, LD-EPG exhibits stable performance across a broad range of configurations. Moderate changes in either θ_{low} or θ_{high} generally result in only limited performance variation, while allowing the sparsity level to be adjusted according to the desired efficiency–accuracy trade-off.

The results also demonstrate that a shared threshold configuration can generalize well across datasets with different motion patterns and object densities. For example, $(1.8, 2.5)$ consistently preserves competitive performance on YT-1B, SA-V, and SmartGlasses without dataset-specific tuning. Dataset-specific threshold configurations can further adjust the sparsity–performance trade-off, but are not required for maintaining stable performance.

Sensitivity of LT-DAG LT-DAG determines whether dense memory computation is required according to three divergence thresholds, corresponding to prediction divergence, identity duplication, and temporal inconsistency, denoted by $(\text{div}, \text{dup}, \text{tmp})$, respectively. We vary each threshold individually around the default configuration $(0.5, 0.7, 0.3)$ while keeping the remaining thresholds fixed.

As shown in Table 14, the performance remains highly stable across different threshold configurations. For example, pHOTA varies only within a narrow range of 70.01–70.14 when changing the prediction-divergence threshold, and similar stability is observed when varying the duplication and temporal-divergence thresholds. These results suggest that LT-DAG is not sensitive to fine-grained threshold tuning and that the default configuration provides a robust operating point.

Table 14: Sensitivity analysis of the divergence thresholds in LT-DAG.

Method	$(\mathrm{div}, \mathrm{dup}, \mathrm{tmp})$	cgF1	pmF1	pHOTA	pDetA
SAM3	-	49.80	56.60	70.00	59.70
LT-DAG (ours)	$(\mathbf{0.5}, \mathbf{0.7}, \mathbf{0.3})$	49.42	56.29	70.10	59.90
	$(\mathbf{0.3}, 0.7, 0.3)$	49.34	56.19	70.01	59.82
	$(\mathbf{0.4}, 0.7, 0.3)$	49.38	56.24	70.02	59.84
	$(\mathbf{0.6}, 0.7, 0.3)$	49.41	56.27	70.14	59.94
	$(\mathbf{0.7}, 0.7, 0.3)$	49.37	56.23	70.06	59.85
	$(0.5, \mathbf{0.5}, 0.3)$	49.48	56.35	70.11	59.89
	$(0.5, \mathbf{0.6}, 0.3)$	49.43	56.30	70.05	59.86
	$(0.5, \mathbf{0.8}, 0.3)$	49.43	56.30	70.06	59.87
	$(0.5, \mathbf{0.9}, 0.3)$	49.46	56.32	70.10	59.90
	$(0.5, 0.7, \mathbf{0.1})$	49.28	56.12	70.04	59.90
	$(0.5, 0.7, \mathbf{0.2})$	49.43	56.29	70.08	59.91
	$(0.5, 0.7, \mathbf{0.4})$	49.23	56.06	70.02	59.83
	$(0.5, 0.7, \mathbf{0.5})$	49.23	56.07	70.05	59.86