FLASHTAM: TARGET-CONDITIONED MEMORY TOKENIZATION FOR ON-DEVICE VIDEO OBJECT SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-device promptable Video Object Segmentation (VOS) aims to track a user-specified target on device-side, imposing even stricter requirements on real-time responsiveness and model compactness. Although on-device VOS has been studied for years, existing methods typically face a trade-off among model size, inference latency, and segmentation quality, and therefore may not fully preserve the segmentation quality of SAM 2. SAM 2 provides stronger target propagation through its streaming memory, but this memory mechanism also introduces substantial computational overhead, limiting its direct deployment on mobile devices. This creates a central challenge: retaining target-related temporal information for accurate propagation while organizing it into a compact memory representation suitable for efficient computation on mobile hardware. To overcome this challenge, we present a Flash Target Conditioned Memory (FlashTAM) design. FlashTAM comprises two main components: (1) Mask-guided ROI Localization (MRL). MRL reuses the predicted mask to localize a target-centric ROI on the memory grid, avoiding extra parameters or attention matrices. (2) Window-wise Adaptive Pooling (WAP). WAP adaptively pools the target-centric ROI into a near-16×16 regular token budget for efficient mobile attention. In summary, MRL decides *"Where to Sample"* and WAP decides *"How to Compress,"* jointly preserving target-relevant temporal information while reducing memory redundancy. Experiments on several benchmarks (SA-V, DAVIS-2017, MOSE, and YouTube-VOS 2019) validate that FlashTAM achieves a favorable accuracy-efficiency trade-off, notably reaching 72.9 J&F on SA-V test and 22.3 FPS on an iPhone 16 Pro, exceeding the second-best on-device method (EdgeTAM) by at least 1.2 J&F and 2.3 FPS.

1 INTRODUCTION

Promptable video object segmentation aims to segment and track a user-specified object throughout a video given point, box, or mask prompts (Ravi et al., 2024). On-device deployment involves processing successive frames under latency and computational constraints on smartphones and smart cameras. Applications such as subject-aware autofocus, real-time visual effects, and augmented reality combine continuous target tracking with low-latency processing. For real-time on-device segmentation, both segmentation quality and inference speed are essential to the effectiveness of the system.

Video object segmentation involves not only delineating an object in an individual frame, but also maintaining its identity as its appearance changes over time. Target information from earlier frames provides a basis for this cross-frame association. To address this temporal requirement, SAM 2 combines promptable segmentation with a streaming memory mechanism: a memory encoder integrates past-frame visual features with target-mask information, while memory attention incorporates this history into current-frame features. This architecture carries information about the prompted target forward, supporting its continued segmentation throughout the video. However, accessing this history requires each incoming frame to interact with a dense set of spatial memory tokens, adding per-frame computation.

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.

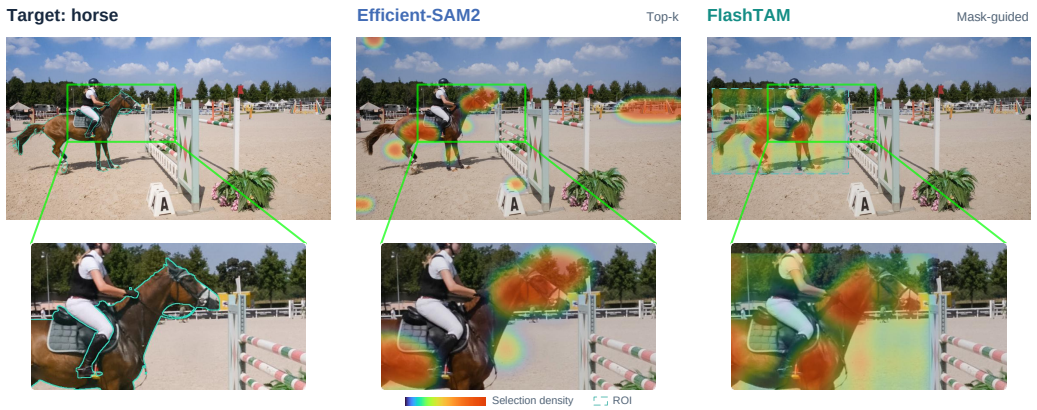


Figure 1: FlashTAM guides memory tokenization toward a target-conditioned and spatially organized representation around the predicted target region. This organization follows from three complementary design principles: (1) target-conditioned spatial selection guided by decoder predictions, (2) localized memory sampling over target-related spatial support, and (3) budgeted, spatially regular tokenization that maintains coverage within the selected region. Efficient-SAM2 instead applies global top-k selection based on attention mass. The comparison uses the same target, video frame, and token budget.

Recent work has explored lightweight SAM 2 models by reducing computation in both image encoding and memory processing. EfficientTAM combines a lightweight image encoder with local spatial pooling for memory cross-attention (Xiong et al., 2025). EdgeTAM uses a mobile backbone and a learned 2D Spatial Perceiver to compress frame-level memory features, and demonstrates on-device inference on an iPhone (Zhou et al., 2025). Efficient-SAM2 takes a different approach by sparsifying memory attention through attention-mass based token selection (Zhang et al., 2026). Together, these methods reduce video segmentation computation through lightweight encoders and compact memory representations, with EdgeTAM further demonstrating the feasibility of mobile deployment. They also reveal a broader challenge beyond reducing the number of memory tokens: retaining temporal information for propagation of the user-specified target while organizing it in a form that matches the computation patterns of mobile hardware. The central difficulty is therefore to balance target-relevant information, memory interaction cost, and hardware-friendly execution.

In on-device promptable video object segmentation, how can target-related temporal information be selected from streaming video memory and organized into a compact memory representation that supports both accurate target propagation and efficient computation on mobile hardware?

To make this question concrete, we first consider it at the level of spatial support. Constructing a compact memory representation involves selecting spatial positions from the memory feature grid and organizing the corresponding features. Figure 1 presents a schematic comparison under the same target, video frame, and token budget. Efficient-SAM2 selects positions globally according to attention mass, whereas FlashTAM uses decoder-predicted target information to define target-related spatial support before organizing the retained positions within the budget. The comparison highlights the distinction between global saliency-based selection and target-conditioned spatial memory construction.

Motivated by this distinction, we hypothesize that the decoder's predicted mask can provide a target-specific signal for selecting which spatial information to retain, while the retained features can be organized into a compact and regular representation for mobile computation. Based on this principle, we propose FlashTAM with two complementary components. Mask-guided ROI localization (MRL) first resizes the predicted mask to the memory-feature grid, extracts the predicted foreground support, and expands its bounding region to retain nearby spatial context. Window-wise adaptive pooling (WAP) then forms an aspect-ratio-aware pooling grid within this region, selects the maximum-logit position in each window, and gathers the corresponding memory features into a compact token sequence near a prescribed budget. MRL therefore defines the candidate region, while WAP determines the representative positions and organizes their features for subsequent memory attention.

This separation also distinguishes where memory is sampled from what content is stored: mask logits determine the sampling locations, whereas memory features provide the stored representation. We further use feature distillation from a larger teacher to improve the accuracy of the mobile-scale model.

Our contributions are:

- **Target-conditioned memory tokenization.** We formulate on-device temporal memory construction as target-conditioned selection and representation. Decoder-predicted masks determine where to sample, while memory features determine what to store, providing a target-relevant signal without an additional selection network or an explicitly constructed attention matrix.
- **Complementary localization and adaptive tokenization.** We introduce mask-guided ROI localization (MRL) to identify target-related support and window-wise adaptive pooling (WAP) to convert it into a compact token sequence near a target budget. This design balances the retention of target-relevant information with reduced memory interaction and hardware-friendly computation.
- **End-to-end mobile evaluation.** We implement and evaluate FlashTAM on an iPhone 16 Pro. It achieves 72.9 J&F on the SA-V test set and 22.3 FPS, compared with 71.7 J&F and 16.8 FPS for EdgeTAM under the same pipeline.

2 Related Work

Promptable image and video segmentation. Segment Anything (Kirillov et al., 2023) established a general promptable segmentation framework with transferability across image domains. Subsequent studies improved the quality or efficiency of promptable image segmentation (Ke et al., 2023; Zhang et al., 2023a; Xiong et al., 2024; Shu et al., 2025). SAM 2 (Ravi et al., 2024) extended this paradigm to videos by introducing a streaming memory mechanism. By storing information from previously processed frames, SAM 2 can track and segment a user-specified object throughout a video. This memory-based design supports temporal propagation and adds memory computation to each frame.

Efficient SAM models and on-device deployment. The high computational cost of promptable segmentation has motivated a range of lightweight SAM variants. MobileSAM and EfficientSAM reduce image segmentation cost through lightweight models and distillation, while FastSAM (Zhao et al., 2023) and EdgeSAM (Zhou et al., 2023) explore alternative real-time or on-device segmentation pipelines. EfficientViT-SAM (Zhang et al., 2024) further replaces the heavy image encoder with a hardware-efficient backbone. These methods primarily target image segmentation and do not address the additional memory computation associated with video propagation. EfficientTAM studies efficient video segmentation by combining a lightweight non-hierarchical Vision Transformer with an efficient memory cross-attention module based on local spatial pooling. EdgeTAM (Zhou et al., 2025) focuses explicitly on on-device deployment. It combines a mobile backbone with a learned 2D Spatial Perceiver for compressing frame-level memory features and demonstrates execution on an iPhone. These studies address both image encoding and memory computation when adapting SAM-based models to mobile devices.

Efficient memory representations. The memory module of SAM 2 presents a distinct efficiency challenge because it repeatedly processes features accumulated from previous frames. Recent methods have explored both compressing memory representations and selecting a subset of memory tokens. EfficientTAM exploits spatial locality through regular local pooling, while EdgeTAM uses a learned spatial module to compress frame-level memories. Efficient-SAM2 (Zhang et al., 2026) takes a different approach and retrieves salient memory tokens according to their attention mass. Its top-k strategy demonstrates that many memory tokens contribute little to the current prediction and can be excluded from memory attention. These methods reduce memory computation through different representation and selection mechanisms. The remaining challenge is to construct a memory representation that preserves target-relevant information, has a fixed computational budget, and can be processed efficiently by mobile accelerators.

3 METHOD

3.1 PRELIMINARIES: SAM 2

SAM 2 performs promptable video object segmentation by augmenting image segmentation with a streaming memory (Ravi et al., 2024). For each incoming frame, the image encoder first extracts visual features. Memory attention then updates these features using information stored from previously processed frames, and the mask decoder predicts the object mask for the current frame. The frame representation and its predicted mask are subsequently written to the memory bank and reused by later frames. The memory read operation is therefore repeated for every incoming frame.

Let $\mathbf{m}_{t_i} \in \mathbb{R}^{H_m W_m \times C}$ denote the spatial memory feature stored for frame t_i, and let $\mathbf{p}_t$ denote the pointer tokens associated with the current memory state. The current frame attends to the sequence

$$\big[\, \mathbf{m}_{t_1}; \ldots; \mathbf{m}_{t_T}; \mathbf{p}_t \,\big], \tag{1}$$

whose length is $TH_mW_m + N_p$. The memory bank has a fixed capacity, so this cost does not grow with video length. With full-resolution spatial memory, the corresponding memory attention complexity is $\mathcal{O}(N_q(TH_mW_m + N_p))$, where N_q is the number of query tokens. This memory interaction contributes to per-frame latency on mobile hardware.

The goal of FlashTAM is to reduce the spatial redundancy in each stored frame while preserving target-related information for propagation. The method uses the decoder prediction rather than a separate selection network or attention map to construct a regular memory representation for fused attention on mobile accelerators.

3.2 TARGET-CONDITIONED MEMORY TOKENIZATION

Dense frame-level memory contains spatial redundancy, while memory attention repeatedly processes these tokens for every incoming frame. Simply reducing the number of tokens, however, may discard information used to propagate the tracked target. FlashTAM addresses this problem by making memory tokenization target-conditioned. The decoder's predicted mask provides a spatial signal for identifying the region associated with the tracked object, and the memory features in this region are then converted into a compact and regular representation near a prescribed token budget $h^\star w^\star$. The tokenization explicitly separates two roles: mask logits determine where memory is sampled, while memory features determine what content is stored for later attention.

The overall architecture and data flow are illustrated in Figure 2. The procedure consists of two connected stages: mask-guided ROI localization (MRL) first identifies and expands the target-related candidate region, and window-wise adaptive pooling (WAP) then selects representative positions within this region to construct the memory representation used by subsequent memory attention. The input to this procedure is the dense memory feature map together with the mask logits predicted for the current frame. MRL resizes the logits to the memory feature grid, extracts the predicted foreground support, and returns an expanded ROI as the candidate region for WAP. WAP takes this ROI together with the corresponding mask logits and memory features, and returns a frame-wise keep set and the corresponding target-conditioned memory features. Thus, the output of MRL directly defines the input region processed by WAP.

MRL: mask-guided ROI localization. Let M_t denote the mask logits predicted for frame t. We resize these logits to the spatial resolution of the memory feature map, obtaining $L_t \in \mathbb{R}^{H_m \times W_m}$. The predicted foreground support is

$$\Omega_t = \{(i, j) : L_t[i, j] > 0\}. \tag{2}$$

We compute the smallest axis-aligned box enclosing Ω_t, expand it by a margin ρ, and clip it to the memory grid:

$$b_t = \big(\ \max(0, x_{\min} - \rho),\ \max(0, y_{\min} - \rho),\ \min(W_m - 1, x_{\max} + \rho),\ \min(H_m - 1, y_{\max} + \rho)\ \big). \tag{3}$$

The box is derived entirely from the model's own predicted mask. The expansion margin retains a small amount of surrounding context and reduces sensitivity to under-segmentation or small object motion between memory insertion and later memory reads. Because the selection signal is

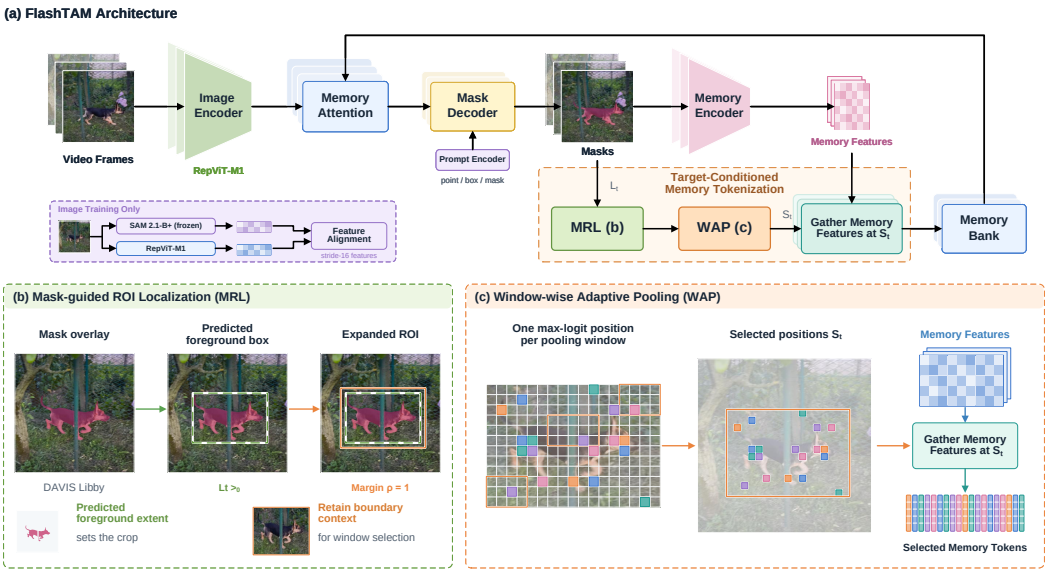


Figure 2: Overview of FlashTAM. Given an input video and an initial prompt, the RepViT-M1 image encoder extracts frame features. The mask decoder predicts the target mask, which is resized to the memory-feature grid to enable mask-guided ROI localization (MRL). The region is expanded by a margin ρ and passed to window-wise adaptive pooling (WAP). Within each pooling window, the location with the maximum mask logit is selected, and the corresponding memory feature is gathered into the target-budget memory representation. The resulting tokens are used by memory attention for subsequent frames. MRL and WAP denote the two stages of target-conditioned memory tokenization. The frozen SAM 2.1-B+ teacher provides feature-alignment targets for the mobile image encoder.

derived from the decoder prediction, severe mask underestimation or localization errors can reduce candidate-region coverage and propagate incomplete target context into memory. The expansion margin partially mitigates this sensitivity by retaining local boundary context, but it does not replace re-detection when the prediction is severely misplaced. This design trades prediction-independent global coverage for target-focused memory construction.

WAP: window-wise adaptive pooling. The MRL box provides a compact candidate region, but retaining every token inside it would make the token count depend on the object scale. WAP therefore pools the candidate region toward a target budget $h^\star w^\star$. Let $h_b \times w_b$ be the height and width of the dilated box. We choose an output grid whose aspect ratio follows the box:

$$\tilde{w} = \left\lceil \sqrt{h^\star w^\star \cdot \frac{w_b}{h_b}} \right\rceil, \qquad \tilde{h} = \left\lceil \sqrt{h^\star w^\star \cdot \frac{h_b}{w_b}} \right\rceil. \tag{4}$$

The grid is clipped to the valid memory resolution. We apply adaptive max pooling to the mask logits within the box. Each pooling cell contributes the location of its maximum-logit element, and these locations define the retained set $\mathcal{S}_t$. The corresponding key and value features at these locations are gathered from the memory feature map.

For normal-sized regions, the number of retained tokens is controlled by the target budget rather than by the object area. The spatial coverage of each token changes with the object extent: larger regions are sampled more coarsely, while smaller regions are sampled more finely. If the candidate box is smaller than the requested pooling grid, WAP retains the available box positions instead of duplicating positions.

Formally, the retained spatial positions are obtained by applying adaptive max pooling to the mask logits within the dilated box:

$$\mathcal{S}_t = \mathrm{ArgMaxPool}_{\tilde{h}\times\tilde{w}}\left(L_t\big|_{b_t}\right). \tag{5}$$

Algorithms 1 and 2 in the supplementary material summarize the procedure.

3.3 Feature Distillation

The mobile image encoder has lower capacity than the SAM 2.1-B+ teacher, which can limit the quality of the visual features available for subsequent mask prediction. We therefore use feature distillation as an auxiliary training objective: the teacher provides feature targets for the mobile image encoder, while the student is trained with both the segmentation objective and the feature-alignment objective. This supervision is applied to the image encoder only, independently of the target-conditioned memory tokenization described in Section 3.2.

Let F^t and F^s denote the stride-16 image-encoder feature maps produced by the teacher and student for the same input frame. We optimize

$$\mathcal{L} = \mathcal{L}_{\text{task}}(O, \text{GT}) + \gamma \, \text{MSE}(F^t, F^s), \qquad \gamma = 1, \tag{6}$$

where O is the student mask prediction and MSE denotes the mean squared error averaged over all elements of the aligned feature maps. The alignment is defined between the image-encoder features before memory conditioning. The teacher uses dense memory, whereas FlashTAM uses a target-dependent sparse memory with a different token layout; their memory-conditioned features are therefore not directly aligned.

The distillation term is used in the image training stages. During long-sequence fine-tuning, the image encoder is frozen and the alignment term is removed.

4 Experiments

4.1 Experimental Setup

Training and model configuration. FlashTAM uses RepViT-M1 (Wang et al., 2024) as the image encoder and follows the SAM 2.1 video segmentation architecture. We use SAM 2.1-B+ as the teacher for feature distillation. Training proceeds in three stages. We first pretrain the image encoder on 5% of SA-1B. We then train the video model on SA-V train (Ravi et al., 2024), DAVIS 2017 (Pont-Tuset et al., 2017), MOSE (Ding et al., 2023), and YouTube-VOS 2019 (Xu et al., 2018; Yang et al., 2019). Finally, we fine-tune the model on 16-frame long-sequence video clips using the same video training data, with the image encoder frozen. Training uses 640×640 inputs, while inference uses 1024×1024 inputs. The memory bank contains seven frames, and the default memory representation uses a target budget of 256 tokens with an expansion margin of $\rho = 1$. We use four memory attention layers by default and apply feature distillation with loss weight $\gamma = 1$. Unless otherwise specified, FlashTAM refers to the four-layer memory-attention configuration. Detailed training configurations are provided in the supplementary material.

Datasets and evaluation protocol. We evaluate the same checkpoint on SA-V test, SA-V val, DAVIS-2017, MOSE, and YouTube-VOS 2019 val. We use mask prompts provided on the first frame and follow the standard evaluation protocol of each benchmark. The prompt is not updated after the first frame. All annotated target objects in the evaluation videos are included in the evaluation. We report $\mathcal{J}\&\mathcal{F}$ on MOSE, DAVIS-2017, and SA-V, and the official $\mathcal{G}$ metric on YouTube-VOS 2019. The ablations are evaluated on SA-V test, with feature distillation disabled for the localization and token-selection comparisons in Tables 4 and 5.

Baselines and implementation. We compare FlashTAM primarily with EdgeTAM as the on-device baseline and with Efficient-SAM2 as the memory-selection baseline. EfficientTAM is included as a related SAM-based efficiency method where its corresponding results are available. Device measurements use an iPhone 16 Pro with the Apple Neural Engine (ANE). Unless otherwise specified, latency refers to end-to-end inference under the same evaluation pipeline.

4.2 Main Results

We compare segmentation quality across five video benchmarks in Table 1, covering representative VOS methods and efficient SAM-based models. Each baseline is cited by its original paper; the caption identifies the sources of the reported scores.

Table 1: Video object segmentation benchmarks. Baseline scores are reproduced from EdgeTAM (Zhou et al., 2025) and the corresponding papers; FlashTAM scores are from our evaluation.

Method	MOSE val	DAVIS 2017 val	SA-V val	SA-V test	YTVOS 2019 val
STCN (Cheng et al., 2021)	52.5	85.4	61.0	62.5	82.7
RDE (Li et al., 2022)	46.8	84.2	51.8	53.9	81.9
SimVOS-B (Wu et al., 2023)	–	88.0	44.2	44.1	84.2
JointFormer (Zhang et al., 2023b)	–	90.1	–	–	87.4
ISVOS (Wang et al., 2023)	–	88.2	–	–	86.3
DEVA (Cheng et al., 2023)	66.0	87.0	55.4	56.2	85.4
Cutie-base (Cheng et al., 2024)	69.9	87.9	60.7	62.7	87.0
Cutie-base+ (Cheng et al., 2024)	71.7	88.1	61.3	62.8	87.5
SAM 2.1-B+ (Ravi et al., 2024)	76.6	90.2	76.8	77.0	88.6
EfficientTAM (Xiong et al., 2025)	69.3	89.1	–	70.7	86.7
EdgeTAM (Zhou et al., 2025)	70.0	87.7	72.3	71.7	86.2
Efficient-SAM2 (Zhang et al., 2026)	70.9	89.3	74.5	75.5	–
FlashTAM	70.0	88.9	73.4	72.9	86.0

Segmentation quality. On SA-V test, FlashTAM obtains 72.9 J&F, compared with 71.7 for EdgeTAM. This result indicates that the proposed memory representation can maintain the segmentation quality of an on-device SAM 2 variant while changing the memory construction strategy. The results on SA-V val, DAVIS 2017, MOSE, and YouTube-VOS 2019 provide complementary evidence under different levels of object motion, appearance variation, and occlusion.

On-device efficiency. We further evaluate the complete inference pipeline on an iPhone 16 Pro using the Apple Neural Engine. Under the same deployment pipeline, EdgeTAM reaches 16.8 FPS, while FlashTAM-2L and FlashTAM-4L reach 24.6 and 22.3 FPS, respectively (Table 2). These correspond to $1.46\times$ and $1.33\times$ the throughput of EdgeTAM. Despite using four memory-attention layers rather than two, FlashTAM-4L runs 32.7% faster than EdgeTAM while providing higher segmentation quality (72.9 versus 71.7 J&F). We retain the four-layer configuration as the default quality–efficiency operating point. All three throughput measurements use the same device and deployment pipeline. Time per frame is computed as $1000/\mathrm{FPS}$ and rounded to one decimal place.

Table 2: On-device end-to-end performance on iPhone 16 Pro.

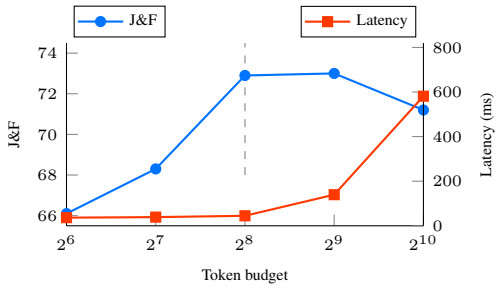
Method	Mem. Attn. Blocks	Backend	FPS	Time/frame (ms)
EdgeTAM	2	ANE	16.8	59.5
FlashTAM-2L	2	ANE	24.6	40.7
FlashTAM-4L	4	ANE	22.3	44.8

Memory compression. The dense memory feature grid contains $64 \times 64 = 4096$ spatial tokens. FlashTAM uses a target budget of 256 tokens, with the retained count typically near this value rather than identical for every frame. With this budget, a sparse frame discards 93.75% of its spatial tokens. With two dense and five sparse frames, the seven-frame bank contains approximately $9{,}472$ spatial tokens instead of $28{,}672$, a reduction of approximately 67.0% (Table 3). These budget-based estimates exclude pointer tokens. The bank has fixed capacity, so its maximum size does not grow with video length. The image encoder and mask decoder are unaffected by this token reduction.

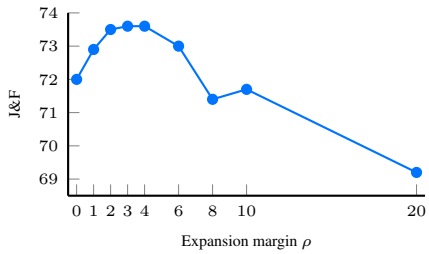
Table 3: Spatial-token compression with a seven-frame memory bank.

Memory setting	Tokens	Discard
Dense frame	4096	–
Sparse frame	256	$\approx 93.75\%$
Seven-frame bank	≈ 9472	$\approx 67.0\%$

Budget sensitivity. We examine the quality–latency trade-off across token budgets. Figure 3 defines this comparison using J&F on the left axis and end-to-end latency on the right axis, with the target budget on the horizontal axis. The sweep reports five measured budget settings from 64 to 1024 tokens. Latency is measured using the same device and deployment pipeline as in Table 2.



(a) Token-budget trade-off



(b) Margin sensitivity

Figure 3: Budget and margin sensitivity of target-conditioned memory tokenization. The left panel reports J&F and end-to-end iPhone 16 Pro latency against the target token budget; the right panel reports J&F against the expansion margin ρ at a token budget of 256. The right panel reports the measured values at $\rho \in \{0, 1, 2, 3, 4, 6, 8, 10, 20\}$. The selection rule in Eq. 7 is applied to the budget sweep before final test evaluation.

We select the smallest budget that remains within a quality tolerance of the largest evaluated budget:

$$B^\star = \min \{B : \ \mathcal{J}\&\mathcal{F}(B) \geq \mathcal{J}\&\mathcal{F}(B_{\max}) - \delta\}, \tag{7}$$

where δ is fixed before evaluating the test set. This criterion separates budget selection from the subsequent final test evaluation.

4.3 Ablation Studies

Spatial localization. Table 4 compares target-agnostic selection with mask-conditioned localization under a target budget of 256 tokens. *Random* and *Pooling* sample or pool positions from the full memory grid, respectively. The mask-conditioned variants start from the predicted tight rectangle: *Tight bbox* uses $\rho = 0$, whereas *MRL (ours)* expands the rectangle by $\rho = 1$ on all four sides. The target-conditioned variants improve over the global baselines, and MRL further improves over the tight bbox from 70.6 to 71.3 J&F. These results support target-conditioned localization with a small boundary margin under the evaluated token budget.

Within-window position selection. Table 5 compares token-selection rules within the same MRL region and under the same token budget. The window-based variants use the same partition and select the top-left position, a random position, or the maximum-logit position (WAP) in each window. Global top-k instead selects across the whole ROI without a per-window constraint. Top-left, random, and global top-k selection reach 68.9, 69.1, and 70.0 J&F, respectively, whereas *WAP (ours)* reaches 71.3 J&F. WAP therefore improves over global top-k by 1.3 points and over random in-window selection by 2.2 points. The result supports the second design choice in FlashTAM: after MRL identifies the candidate region, window-wise selection preserves spatial coverage while choosing the maximum-logit position in each local region.

Table 4: Spatial localization.

Variant	Region	J&F ↑
Random	Full grid	63.5
Pooling	Full grid	68.7
Tight bbox	Rectangle, $\rho = 0$	70.6
MRL (ours)	Rectangle, $\rho = 1$	**71.3**

Table 5: Token selection.

Selection rule	Scope	J&F ↑
Top-left	Per window	68.9
Random	Per window	69.1
Global top-k	Whole ROI	70.0
WAP (ours)	Per window	**71.3**

Complementary contributions. Table 6 summarizes how the two memory components and feature distillation contribute to the final model. The first two rows provide reference constructions based on full-grid pooling and retaining all tokens inside the mask ROI; the last two rows use the same MRL+WAP construction and toggle distillation. Target-conditioned localization improves J&F from 68.7 to 70.4, and replacing all ROI tokens with the compact MRL+WAP representation further improves it to 71.3 with a target budget of 256 tokens. This comparison supports combining target localization with selective spatial coverage rather than retaining all ROI tokens. With the memory construction fixed, feature distillation provides an additional 1.6-point gain, yielding 72.9 J&F for the complete FlashTAM configuration.

Table 6: Reference component and distillation ablation.

Memory construction	**Distill.**	**Token budget**	**J&F**
Full-grid fixed-window max pooling	×	1024	68.7
Mask ROI + full tokens	×	ROI-dependent	70.4
MRL + WAP	×	256	71.3
MRL + WAP	✓	256	72.9

4.4 Memory Selection Analysis

Qualitative results. Figure 4 compares SAM 2.1 and FlashTAM under the same first-frame mask prompt. FlashTAM generally preserves the target's silhouette and location under appearance changes and clutter.

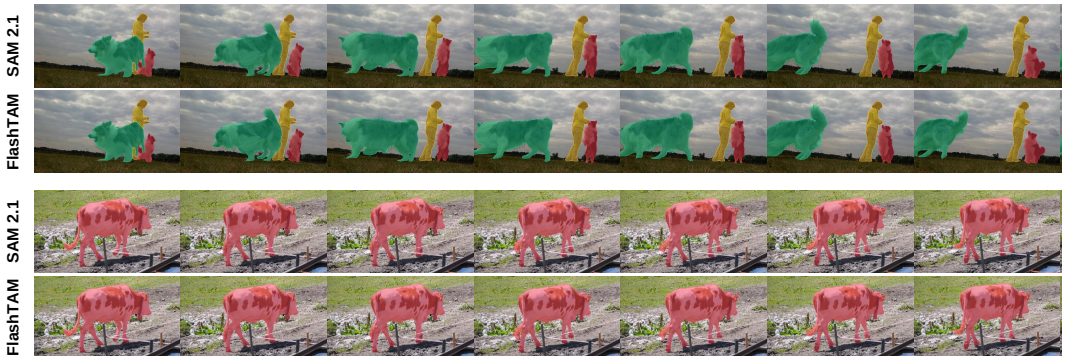


Figure 4: Qualitative comparison between SAM 2.1 and FlashTAM on representative video frames. The two methods are evaluated on the same sequences and initialized with the same first-frame mask prompt.

5 Conclusion

We introduce FlashTAM, a target-conditioned memory tokenization method for on-device promptable video object segmentation. Mask-guided ROI localization uses decoder-predicted masks to define target-related spatial support, while window-wise adaptive pooling selects representative positions within this support to form a compact and regular memory representation near a prescribed token budget. This design separates the signal used to select memory positions from the features stored for subsequent attention, while retaining the spatial correspondence of the selected features and reducing memory interaction cost.

With feature distillation from SAM 2.1-B+, FlashTAM reaches 72.9 J&F on SA-V test and 22.3 FPS on an iPhone 16 Pro. These results indicate that target-conditioned memory tokenization provides a way to organize target-related temporal information into a compact representation that maintains segmentation quality while supporting real-time inference on mobile devices.

AI USE STATEMENT

We used generative AI tools to aid or polish writing. Generative AI tools were not used to generate experimental data or run experiments. All AI-assisted text was reviewed and edited by the authors. We take responsibility for the final content of this work.

REFERENCES

Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. Tracking anything with decoupled video segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. Putting the object back into video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip H.S. Torr, and Song Bai. MOSE: A new dataset for video object segmentation in complex scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20224–20234, 2023.

Lei Ke, Mingqiao Ye, Martin Danelljan, Yifan Liu, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Segment anything in high quality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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. *arXiv preprint arXiv:2304.02643*, 2023.

Mingxing Li, Li Hu, Zhiwei Xiong, Bang Zhang, Pan Pan, and Dong Liu. Recurrent dynamic embedding for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1332–1341, 2022.

Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

Han Shu, Wenshuo Li, Yehui Tang, Yiman Zhang, Yihao Chen, Houqiang Li, Yunhe Wang, and Xinghao Chen. TinySAM: Pushing the envelope for efficient segment anything model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 20470–20478, 2025.

Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. RepViT: Revisiting mobile CNN from ViT perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Junke Wang, Dongdong Chen, Zuxuan Wu, Chong Luo, Chuanxin Tang, Xiyang Dai, Yucheng Zhao, Yujia Xie, Lu Yuan, and Yu-Gang Jiang. Look before you match: Instance understanding matters in video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Qiangqiang Wu, Tianyu Yang, Wei Wu, and Antoni B. Chan. Scalable video object segmentation with simplified framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Yunyang Xiong, Balakrishnan Varadarajan, Lemeng Wu, et al. EfficientSAM: Leveraged masked image pretraining for efficient segment anything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas S. Huang. YouTube-VOS: A large-scale video object segmentation benchmark. *CoRR*, abs/1809.03327, 2018.

Linjie Yang, Yuchen Fan, and Ning Xu. The 2nd large-scale video object segmentation challenge — video object segmentation track, October 2019.

Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight SAM for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023a.

Jiaming Zhang, Yutao Cui, Gangshan Wu, and Limin Wang. Joint modeling of feature, correspondence, and a compressed memory for video object segmentation. *arXiv preprint arXiv:2308.13505*, 2023b.

Jing Zhang, Zhikai Li, Xuewen Liu, and Qingyi Gu. Efficient-SAM2: Accelerating SAM2 with object-aware visual encoding and memory retrieval. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2602.08224.

Zhuoyang Zhang, Han Cai, and Song Han. EfficientViT-SAM: Accelerated segment anything model without performance loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 7859–7863, 2024.

Xu Zhao, Wenchao Ding, Yao An, et al. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023.

Chong Zhou, Xiangtai Li, Chen Change Loy, and Bo Dai. EdgeSAM: Prompt-in-the-loop distillation for on-device deployment of SAM. *arXiv preprint arXiv:2312.06660*, 2023.

Chong Zhou, Chenchen Zhu, Yunyang Xiong, Saksham Suri, Fanyi Xiao, Lemeng Wu, Raghuraman Krishnamoorthi, Bo Dai, Chen Change Loy, Vikas Chandra, and Bilge Soran. EdgeTAM: On-device track anything model. *arXiv preprint arXiv:2501.07256*, 2025.

FLASHTAM: SUPPLEMENTARY MATERIAL

Anonymous authors
Paper under double-blind review

1 MEMORY TOKENIZATION ALGORITHMS

Algorithm 1 summarizes memory reads and frame insertion. Algorithm 2 constructs the keep set using MRL and WAP. Together, they specify the procedure illustrated in Figure 2 of the main paper.

Algorithm 1 Target-conditioned memory tokenization and memory read

Input: current frame x_t; memory bank $\mathcal{B}$; target budget $h^\star w^\star$; expansion margin ρ; mode $m \in \{\text{train}, \text{inference}\}$.

Output: memory-conditioned features Y_t and the updated memory bank $\mathcal{B}$.

$F_t \leftarrow \mathrm{ImageEncoder}(x_t)$
Select the memory entries from $\mathcal{B}$ for the current frame.
for each selected memory frame i **do**
  if i is a conditioning/prompt frame or the newest frame **then**
    $\mathcal{S}_i \leftarrow$ full keep set
  else
    $\mathcal{S}_i \leftarrow$ keep set stored with frame i
  end if
end for
Assemble memory features and positional encodings in the selected order.
Project F_t and the assembled memory to Q, K, and V.
Apply positional encoding to spatial keys using their original grid positions.
$\mathcal{S} \leftarrow$ concatenate the keep sets in the same order; enable all pointer tokens.
if $m = \text{train}$ **then**
  $Y_t \leftarrow \mathrm{SDPA}(Q, K, V; \text{key_mask} = \mathcal{S})$
else
  $(\widetilde{K}, \widetilde{V}) \leftarrow \mathrm{Gather}(K, V, \mathcal{S})$
  $Y_t \leftarrow \mathrm{SDPA}(Q, \widetilde{K}, \widetilde{V})$
end if
$M_t \leftarrow \mathrm{MaskDecoder}(Y_t)$
$D_t \leftarrow \mathrm{MemoryEncoder}(F_t, \sigma(M_t))$
$P_t \leftarrow$ positional encoding associated with D_t
$(H_m, W_m) \leftarrow$ spatial shape of D_t
$\mathcal{S}_t \leftarrow \mathrm{BuildKeepSet}(M_t, H_m, W_m, h^\star, w^\star, \rho)$ *// Algorithm 2*
Append $(D_t, P_t, \mathcal{S}_t)$ to $\mathcal{B}$.
return $Y_t, \mathcal{B}$

Algorithm 2 Mask-guided ROI localization and window-wise adaptive pooling

Input: mask logits M_t; memory grid size $H_m \times W_m$; target budget $h^\star w^\star$; expansion margin ρ.
Output: Boolean keep set $\mathcal{S}_t \in \{0,1\}^{H_m W_m}$.

$L_t \leftarrow \mathrm{Resize}(M_t, H_m, W_m)$
$\Omega_t \leftarrow \{(i,j) : L_t[i,j] > 0\}$
$b_t \leftarrow$ smallest axis-aligned box enclosing Ω_t
Expand b_t by ρ on each side and clip it to the memory grid.
$(h_b, w_b) \leftarrow$ height and width of b_t
$\widetilde{w} \leftarrow \lceil \sqrt{h^\star w^\star \, (w_b/h_b)} \rceil$
$\widetilde{h} \leftarrow \lceil \sqrt{h^\star w^\star \, (h_b/w_b)} \rceil$
Clip $\widetilde{h}$ and $\widetilde{w}$ to the valid memory-grid dimensions.
if b_t is smaller than the requested pooling grid **then**
    $\mathcal{I}_t \leftarrow$ all spatial positions inside b_t
else
    $\mathcal{I}_t \leftarrow \mathrm{ArgMaxPool}_{\widetilde{h} \times \widetilde{w}}(L_t|_{b_t})$
    Map $\mathcal{I}_t$ to the original memory-grid coordinates.
end if
$\mathcal{S}_t \leftarrow \mathbf{0}_{H_m W_m}$
Set entries indexed by $\mathcal{I}_t$ in $\mathcal{S}_t$ to one.
return $\mathcal{S}_t$

The mask logits determine the positions selected by Algorithm 2; the corresponding memory features provide the content used by memory attention in Algorithm 1.

Training and inference paths. The training path retains rectangular key and value tensors and applies $\mathcal{S}$ as a Boolean key mask; this supports different keep sets for different samples in the same batch. The inference path can physically gather the retained key and value entries after positional encoding, thereby shortening the attention sequence. Both paths use the same frame-wise keep sets and the same original spatial positions. If a predicted mask contains no foreground, the training implementation falls back to a full keep set for that sample to avoid masking every memory key.

2 TRAINING CONFIGURATIONS

Table S1: Training configurations for the three training stages.

Configuration	Stage 1 Image encoder pretraining	Stage 2 Video training	Stage 3 Long-sequence fine-tuning
Data	5% SA-1B	SA-V, DAVIS, MOSE, YTVOS	
Training mode	Image only, single frame	Video, 8-frame clips	Video, 16-frame clips
Iterations	$\sim$80k	$\sim$120k	$\sim$45k
Batch size	256	64	32
Frozen modules	all except image encoder	None	image encoder
Learning rate (lr)	image: 3×10^{-4} base: 5×10^{-5}	image: 6×10^{-5} other: 5×10^{-5}	5×10^{-5}
Gradient clipping	None	ℓ_2, max 0.1	
Image augmentation	hflip, affine, color jitter, grayscale	–	
Video augmentation	–	hflip, affine, color jitter, grayscale, per-frame color jitter	
Occlusion loss (weight)	None	BCE (1)	
Distillation loss (weight)	1	0	
Resolution	640×640		
Precision	bfloat16 AMP		
Optimizer	AdamW		
Optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$		
Weight decay	0.1 (bias, LayerNorm: 0)		
LR schedule	Cosine decay to 10^{-6}		
Warmup	Linear from 10^{-6}; 11.5% of training		
Cooldown	None		
Mask losses (weight)	focal (20), dice (1)		
IoU loss (weight)	L1 (1)		

Only image-encoder parameters are optimized in Stage 1, so the configured base learning rate does not apply to trainable parameters. In Stage 2, no modules are frozen and both learning rates apply. In Stage 3, the image encoder is frozen and only the base learning rate of 5×10^{-5} is used. Stage 1 is image-only, so no occlusion loss is used. In Stages 2 and 3, object presence is supervised by the object-score term (`loss_class`) with weight 1, which reduces to binary cross-entropy and corresponds to the occlusion classification term in SAM 2 and EdgeTAM. Stage 1 does not configure gradient clipping. No cooldown schedule is implemented. AdamW uses the library default $\beta_1, \beta_2 = 0.9, 0.999$. Stages 2 and 3 train on SA-V, DAVIS, MOSE, and YouTube-VOS only, with at most 3 objects per clip. Stage 2 uses 8-frame clips and Stage 3 uses 16-frame clips. Stage 1 uses random horizontal flipping, affine transformation ($25°$, shear $20°$), color jitter (brightness 0.1, contrast 0.03, saturation 0.03), and grayscale with probability 0.05. Stages 2 and 3 use the same operators with temporally consistent transforms, plus a per-frame color jitter (brightness 0.1, contrast 0.05, saturation 0.05). All stages resize to 640×640 and apply ImageNet normalization.

3 LIMITATIONS AND FUTURE DIRECTIONS

Training resolution. The current model is trained primarily with 640×640 inputs because of the memory limitations of the available training hardware, while inference uses 1024×1024 inputs. Training at a higher input resolution could preserve more local structure and boundary cues, which may benefit segmentation of complex boundaries and small objects. However, increasing the training resolution also increases the memory required by intermediate features, gradients, and optimizer states. Exploring higher-resolution or multi-scale training, as well as better matching between training and inference resolutions, remains an avenue for future work.

Backbone selection. RepViT-M1 provides a lightweight image encoder for mobile deployment, but it was primarily designed for efficient image recognition rather than video object segmentation. A mobile backbone designed specifically for VOS could provide more suitable features for boundary representation, cross-frame stability, and target propagation while preserving the computational characteristics required for on-device execution. The practical benefit would also depend on the target hardware, operator implementations, and deployment backend.