EdgeTAM: On-Device Track Anything Model

Chong Zhou[1,2]*, **Chenchen Zhu**[1], **Yunyang Xiong**[1], **Saksham Suri**[1], **Fanyi Xiao**[1], **Lemeng Wu**[1]
Raghuraman Krishnamoorthi[1], **Bo Dai**[3,4], **Chen Change Loy**[2], **Vikas Chandra**[1], **Bilge Soran**[1]
[1]Meta AI, [2]Nanyang Technological University, [3]The University of Hong Kong, [4] Feeling AI

Abstract

On top of Segment Anything Model (SAM), SAM 2 further extends its capability from image to video inputs through a memory bank mechanism and obtains a remarkable performance compared with previous methods, making it a foundation model for video segmentation task. In this paper, we aim at making SAM 2 much more efficient so that it even runs on mobile devices while maintaining a comparable performance. Despite several works optimizing SAM for better efficiency, we find they are not sufficient for SAM 2 because they all focus on compressing the image encoder, while our benchmark shows that the newly introduced memory attention blocks are also the latency bottleneck. Given this observation, we propose EdgeTAM, which leverages a novel 2D Spatial Perceiver to reduce the computational cost. In particular, the proposed 2D Spatial Perceiver encodes the densely stored frame-level memories with a lightweight Transformer that contains a fixed set of learnable queries. Given that video segmentation is a dense prediction task, we find preserving the spatial structure of the memories is essential so that the queries are split into global-level and patch-level groups. We also propose a distillation pipeline that further improves the performance without inference overhead. As a result, EdgeTAM achieves 87.7, 70.0, 72.3, and 71.7 $\mathcal{J}\&\mathcal{F}$ on DAVIS 2017, MOSE, SA-V val, and SA-V test, while running at 16 FPS on iPhone 15 Pro Max. The code and models are available here.

1. Introduction

Segment Anything Model (SAM) [20] is the first foundation model for promptable image segmentation. Various studies show its magnificent capabilities on zero-shot generalization and transfer learning [5, 27, 42, 53]. On top of SAM, recently, SAM 2 [35] extends the original SAM to handle both image and video inputs, with a memory bank mechanism, and is trained with a new large-scale multi-grained video tracking dataset (SA-V).

*Work done during the internship at Meta Reality Labs.

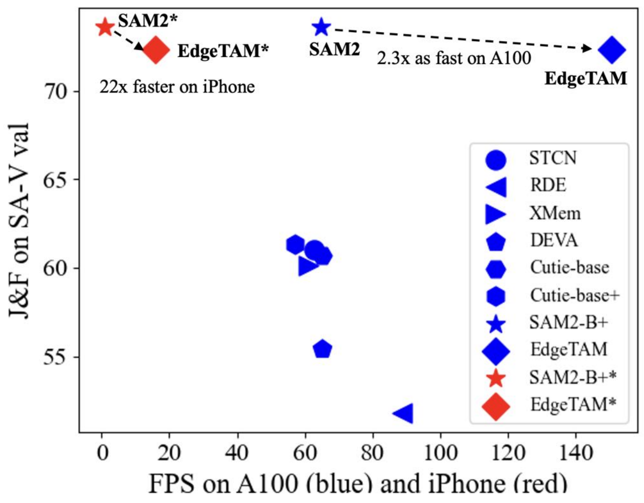


Figure 1. **Speed-performance trade-offs on iPhone 15 Pro Max and NVIDIA A100.** EdgeTAM is significantly faster than SAM 2 on edge devices and compare to other VOS methods, it is also more accurate on the challenging SA-V val dataset. Note that, EdgeTAM can run at 16 FPS on iPhone 15 Pro Max.

Despite achieving an astonishing performance compared to previous video object segmentation (VOS) models and allowing more diverse user prompts, SAM 2, as a server-side foundation model, is not efficient for on-device inference. For instance, the smallest SAM 2 variant runs at only around 1 FPS on an iPhone 15 Pro Max [1]. Furthermore, existing methods [54, 65, 68] that optimize SAM for better efficiency only consider squeezing its image encoder since the mask decoder is extremely lightweight. But as shown in Fig. 2, this is not sufficient for SAM 2 because even when the image encoder is replaced with much more compact visual backbones, such as ViT-Tiny [43] and RepViT [49], the latency does not improve by much due to the computationally demanding memory attention blocks that are newly

[1]We convert to CoreML model with coremltools [1] and benchmark with CPU and NPU. Throughout the paper, we interchangeably use iPhone and iPhone 15 Pro Max for simplicity.

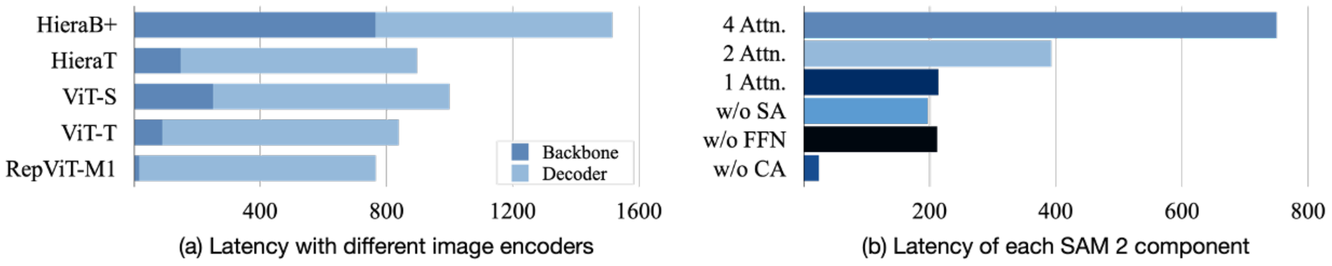


Figure 2. **Single frame latency (ms) on iPhone.** In (a), we show that only replacing image encoder with more compact backbones is not enough for further speed-up since decoder is also a bottleneck. In (b), through reducing the number of memory attention blocks and removing certain modules, we find that the cross attention (CA) is the root cause.

introduced in SAM 2. Specifically, SAM 2 encodes past frames with a memory encoder, and these frame-level memories together with object-level pointers (obtained from the mask decoder) serve as the memory bank. These are then fused with the features of current frame via memory attention blocks. As these memories are densely encoded, this leads to a huge matrix multiplication during the cross-attention between current frame features and memory features. Therefore, despite containing relatively fewer parameters than the image encoder, the computational complexity of the memory attention is not affordable for on-device inference. The hypothesis is further proved by Fig. 2, where reducing the number of memory attention blocks almost linearly cuts down the overall decoding latency and within each memory attention block, removing the cross attention gives the most significant speed-up.

To make such a video-based tracking model run on device, in EdgeTAM, we look at exploiting the redundancy in videos. To do this in practice, we propose to compress the raw frame-level memories before performing memory attention. We start with naïve spatial pooling and observe a significant performance degradation, especially when using low-capacity backbones. To mitigate this issue, we turn to learning-based compressors such as Perceiver [18, 19], which summarizes the dense feature map with a small fixed set of learned queries. However, naïvely incorporating a Perceiver also leads to a severe drop in performance. We hypothesize that as a dense prediction task, the video segmentation requires preserving the spatial structure of the memory bank, which a naïve Perceiver discards.

Given these observations, we propose a novel lightweight module that compresses frame-level memory feature maps while preserving the 2D spatial structure, named 2D Spatial Perceiver. Specifically, we split the learnable queries into two groups, where one group functions similarly to the original Perceiver, where each query performs global attention on the input features and outputs a single vector as the frame-level summarization. In the other group, the queries have 2D priors, *i.e.*, each query is only responsible for compressing a non-overlapping local patch, thus the output maintains the spatial structure while reducing the total number of tokens. As a plug-in module, 2D Spatial Perceiver can be integrated with any variants of SAM 2 and speed up the memory attention by $8\times$ with comparable performance. For instance, when using RepViT-M1 [49] as the backbone and two memory attention blocks, leveraging the 2D Spatial Perceiver yields 16 FPS on iPhone, which is $6.4\times$ faster than the baseline and even surpasses it on the challenging SA-V val set [35] by 0.9 $\mathcal{J}\&\mathcal{F}$.

In addition to the architecture improvement, we further propose a distillation pipeline that transfers the knowledge of the powerful teacher SAM 2 to our student model, which improves the accuracy at no cost of inference overhead. Specifically, the training procedure of SAM 2 has two stages, where firstly the model is trained with the promptable image segmentation task on SA-1B [20] with memory-related module detached, then in the second stage, it is trained with all modules included for the promptable video segmentation task on both SA-1B and SA-V [35] datasets. We find that in both stages, aligning the features from image encoders of the original SAM 2 and our efficient variant benefits the performance. Besides, we further align the feature output from the memory attention between the teacher SAM 2 and our student model in the second stage so that in addition to the image encoder, memory-related modules can also receive supervision signals from the SAM 2 teacher. As a result, with the proposed distillation pipeline, we improve the $\mathcal{J}\&\mathcal{F}$ on SA-V val and test by 1.3 and 3.3, respectively.

Putting together, we propose **EdgeTAM** (Track Anything Model for Edge devices), that adopts a 2D Spatial Perceiver for efficiency and knowledge distillation for accuracy. Our contributions can be summarized in the following:

- Through comprehensive benchmark, we reveal that the latency bottleneck lies in the memory attention module.
- Given the latency analysis, we propose a 2D Spatial Perceiver that significantly cuts down the memory attention

computational cost with comparable performance, which can be integrated with any SAM 2 variants.

- We experiment with a distillation pipeline that performs feature-wise alignment with the original SAM 2 in both the image and video segmentation stages and observe performance improvements without any additional cost during inference.
- The resulting EdgeTAM can run at **16 FPS** on an iPhone, which is notably faster than existing video object segmentation models and surpasses or is on par with the previous state-of-the-art methods. To our knowledge, it is the first model running on device for the task of unified segmentation and tracking.

2. Related Work

Video Object Segmentation (VOS). The objective of the VOS task is, given the ground-truth (GT) object segmentation mask on the first frame, tracking and predicting the object mask throughout the following frames in the video. Online learning approaches [3, 4, 16, 26, 28, 29, 32, 33, 36, 39, 46, 52] formulate the task as a semi-supervised learning problem, where during test time, the model is fine-tuned with the GT mask on the first frame. However, this line of work usually suffers from inference inefficiency, being input sensitive and hard to scale up with large amounts of training data. To avoid test-time training, offline-trained models propose to leverage template matching [7, 17, 30, 47, 57, 58, 60, 62], or memory bank [23, 31] to keep track of the identity information in the annotated and predicted frames. In terms of the network architecture, some works adopt recurrent networks for spatial-temporal encoding [21, 22, 45, 55], while recently, Transformer-based models [2, 8, 9, 11, 14, 21, 38, 50, 51, 59, 61, 63, 66] demonstrate better performance.

Segment Anything Model (SAM). SAM [20] defines a new prompt-based segmentation task where the user prompts can be points, boxes, and masks. SAM 2 [35] further extends the task to the video inputs, namely promptable video segmentation (PVS). Different from VOS, users can provide annotations at any frame and at multiple time steps with any combination of SAM prompts, making VOS a special case of PVS. Both SAM and SAM 2 follow the same meta architecture of image encoder and prompt-based mask decoder, but to capture temporal information, SAM 2 supplements a memory banking mechanism. Thanks to training on diverse and large-scale datasets, SA-1B [20] and SA-V [35], SAM excels in both general perception and downstream tasks [5, 6, 27, 42, 53, 64]. To make SAM more efficient and more friendly to low-capacity devices, several works [48, 54, 65, 67, 68] propose to squeeze its image encoder to more compact visual backbones with knowledge distillation and/or masked image pre-training. However, through our benchmark, we find that apart from the image encoder, the newly introduced memory-related modules in SAM 2 are also the speed bottleneck; thus, replacing the image encoder is no longer sufficient. Therefore, we propose a novel plug-in module to accelerate memory fusion to address the problem, together with a distillation pipeline adapted for video inputs.

3. Methodology

In this section, we first briefly introduce the Segment Anything Model 2 (SAM 2), which our model is based on. Then, we propose our architecture-level improvements and knowledge distillation pipeline, respectively.

3.1. Preliminary: SAM 2

Overall, SAM 2 consists of four components, namely image encoder E_{img}, mask decoder D, memory encoder E_{mem}, and memory attention A, with the former two almost identical to the original SAM except for the skip connection between the two. In particular, E_{img} is a hierarchical backbone called Hiera [37], which outputs feature maps in three different strides, 4, 8, and 16 denoted by F_4, F_8, F_{16}, respectively:

$$\{F_4, F_8, F_{16}\} = E_{\text{img}}(I), \tag{1}$$

where I is the current frame input. Then, F_{16} is fused with memory features $\{M_1, M_2, \dots, M_T\}$ [2] from previous T frames with the memory attention A. The memory attention is essentially a stack of Transformer [44] blocks. In this setup, F_{16} serves as the queries, while memory features, concatenated along the temporal dimension, provide the keys and values:

$$F_M = A(F_{16}, M_1, M_2, \dots, M_T), \tag{2}$$

where F_M is the image feature conditioned on memories. Next, mask decoder D encodes the user prompt and decodes the mask prediction O given the prompt embedding P and image features F_M, F_4, F_8:

$$O = D(F_M, F_4, F_8, P). \tag{3}$$

Finally, F_{16} and O are fused and encoded with the memory encoder E_{mem} and enqueued the memory bank in a first-in-first-out manner:

$$M_{T+1} = E_{\text{mem}}(F_{16}, O). \tag{4}$$

3.2. EdgeTAM

Naïve Adaptations. As shown in Fig. 3, the meta architecture of SAM 2 follows closely with SAM, whose image encoder is the heaviest component in terms of parameters and

[2]For simplicity, M denotes the frame-level memory feature map and we omit the object pointers (vectors from the mask decoder), which add negligible computational cost.

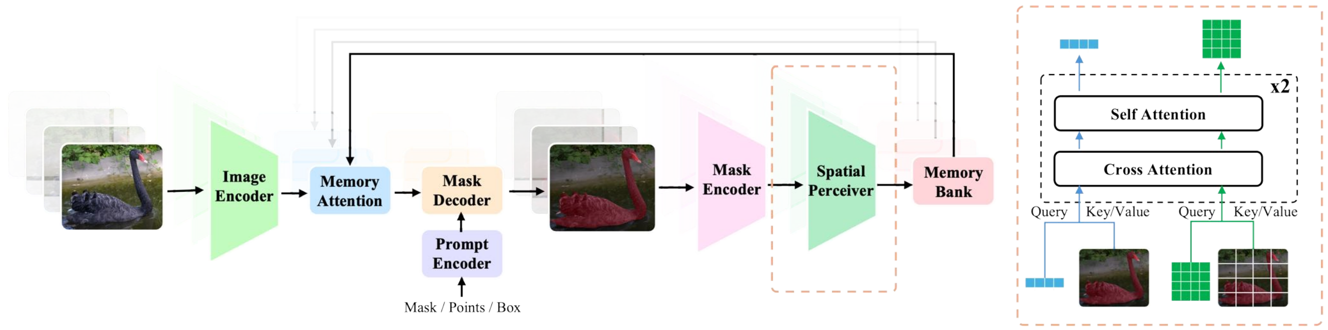


Figure 3. **Overall architecture of EdgeTAM.** The meta architecture of EdgeTAM follow SAM 2 and the main difference is the proposed plug-in module, 2D Spatial Perceiver, which is marked with orange dotted box.

computation. While the newly introduced memory-related module takes up only a small proportion of the total parameters, our benchmark (Fig. 2) shows that memory attention is also a latency bottleneck. Therefore, a naïve technique to push for improved efficiency is to substitute the image encoder with compact backbones and to reduce the number of memory attention blocks. To this end, following EdgeSAM [68], we opt for RepViT-M1 [49] as the backbone and decrease the memory attention from 4 to 2 blocks. However, the inference throughput is still far from being satisfactory when deployed on mobile devices (merely 2.5 FPS on iPhone 15 Pro Max).

Taking a closer look, we observe that each memory feature M_t has the same size as the image feature $F_M \in \mathcal{R}^{C\times H\times W}$, where $C = 64$, $H = W = 64$ denote channels, height and width respectively. With T frames in the memory bank, the computational complexity of memory attention becomes $\mathcal{O}(TCH^2W^2)$, which translates to a huge matrix multiplication that mobile devices with limited scale of parallelism perform inefficiently. While T is already relatively small compared to other VOS methods, reducing it will lead to the degradation of temporal consistency and occlusion handling. On the other hand, videos are known to be information redundant. Thus, we propose to summarize the memory spatially before performing memory attention.

Global Perceiver. Inspired by Perceiver [18, 19], we encode each memory feature M_t with a stack of attention modules to compress the densely stored memories $M_t \in \mathcal{R}^{C\times H\times W}$ into a small set of vectors $G_t \in \mathcal{R}^{C\times N_g}$, where N_g is the number of learnable latents and $N_g \ll H \times W$. Specifically, we denote the latents as $Z_g \in \mathcal{R}^{C\times N_g}$ and perform single-head cross attention (CA) between Z_g and M_t, followed by self attention (SA) as follows:

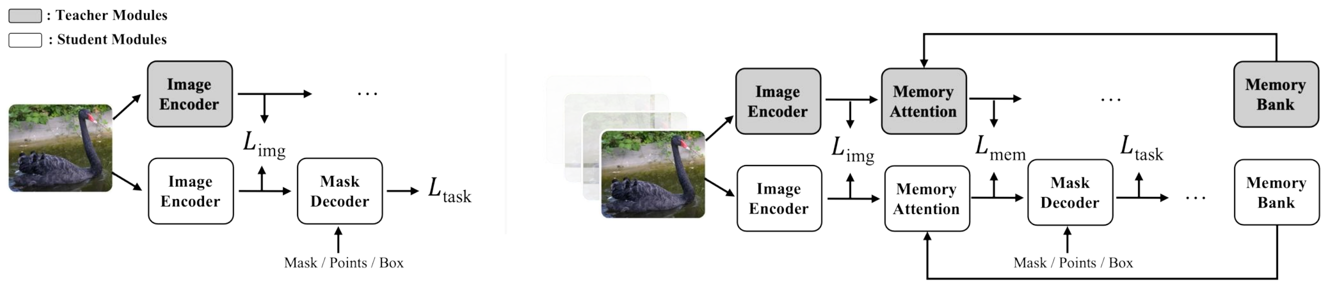
$$\begin{aligned} Z'_g &= \text{CA}(Q(Z_g), K(M_t + p), V(M_t + p)), \\ G_t &= \text{SA}(Z'_g), \end{aligned} \tag{5}$$

where Q, K, and V represent the projections for query, key, and value in CA, respectively. Z'_g is the intermediate feature and p denotes the positional embeddings [40]. Here, each latent can attend globally to the memory feature and summarize it into a single vector. While the Global Perceiver introduces negligible inference cost, it cuts down the complexity of the memory attention to $\mathcal{O}(TCHWN_g)$. However, despite adding positional embeddings to the input of Global Perceiver, the resulting compressed memories contain only implicit positional information as the output does not maintain its spatial structure. Meanwhile, as a dense prediction task, video object segmentation requires more explicit positional information [35] and local features [38]. We thus further propose a 2D Spatial Perceiver for this purpose.

2D Spatial Perceiver. Similar to the Global Perceiver, 2D Spatial Perceiver shares the same network architecture and parameters. However, we assign spatial prior to the learnable latents $Z_l \in \mathcal{R}^{C\times N_l}$ and restrict each latent to only attend to a local window. Specifically, we perform the window partition [25] to split the memory feature map into N_l non-overlapping patches, and move the positional embedding p' from input to output L_t:

$$\begin{aligned} M'_t &= \text{window\_partition}(M_t), \\ Z'_l &= \text{CA}(Q(Z_l), K(M'_t), V(M'_t)), \\ L'_t &= \text{SA}(Z'_l), \\ L_t &= \text{window\_unpartition}(L_t) + p'. \end{aligned} \tag{6}$$

The different designs of Global and 2D Spatial Perceiver encourage different behaviors, where global latents Z_g have certain redundancy (multiple latents attend to the same input) and can dynamically distribute all over the image whereas 2D latents Z_l are forced to deal with local patches. And both possess desirable merits for feature summarization. Therefore, we combine them by flattening along the spatial dimension and concatenating along the flattened dimension. Note that, our implementation stacks the blocks in Eq. 5 and Eq. 6 twice. Overall, when applying the proposed modules, the complexity of memory attention decreases from $\mathcal{O}(TCH^2W^2)$ to $\mathcal{O}(TCHW(N_g + N_l))$. In



(a) Image segmentation pre-training stage.

(b) Video segmentation training stage.

Figure 4. **The distillation pipeline in EdgeTAM.** In the image pre-training stage, we align the features from teacher's and student's image encoder. And in the video training stage, we additionally align the features output from memory attention between teacher and student. For both stages, task-specific losses are used.

practice, we control the speed-up ratio to around T times, *i.e.*, $(HW)/(N_g+N_l) \approx T$, so that the self and cross attention blocks in memory attention have similar complexity.

3.3. Distillation Pipeline

As shown in Fig. 4, the training pipeline of SAM 2 can be divided into image segmentation pre-training S_{img} and video segmentation training S_{vid} stages. Previous methods [54, 65, 68] demonstrate that knowledge distillation on S_{img} helps improve performance on images. Here, we extend this idea to the video domain and treat the distillation loss as an auxiliary loss, meaning task-specific losses are also implemented during training.

Particularly, during S_{img}, we adopt the same task-specific losses $\mathcal{L}_{\text{task}}$ as SAM (dice loss [41] and focal loss [24] for mask prediction and L1 loss for mask confidence prediction) and meanwhile, align the image encoder feature map (F_{16} in Eq. 1) between the teacher and student models with MSE loss $\mathcal{L}_{\text{img}}$. The pre-training loss $\mathcal{L}_{\text{sam}}$ can be formulated with:

$$\mathcal{L}_{\text{sam}} = \mathcal{L}_{\text{task}}(O, \text{GT}) + \gamma \cdot \mathcal{L}_{\text{img}}(F^t_{16}, F^s_{16}), \tag{7}$$

where O is the mask prediction obtained from Eq. 1 and Eq. 3. Here Eq. 2 is skipped due to the lack of memory bank and $F_M = I$. Here, GT, γ, F^t_{16} and F^s_{16} denote the ground-truth labels, loss weight, teacher and student image encoder features respectively.

Finally, in stage S_{vid}, the task-specific losses include an additional BCE loss for occlusion prediction. Besides, in order to let student's memory-related modules receive supervision from the teacher, apart from $\mathcal{L}_{\text{img}}$, we add another MSE loss $\mathcal{L}_{\text{mem}}$ to align the F^t_M and F^s_M from teacher and student (Eq. 2). The resulting total loss becomes:

$$\begin{aligned}\mathcal{L}_{\text{sam2}} = &\mathcal{L}_{\text{task}}(O, \text{GT}) + \alpha \cdot \mathcal{L}_{\text{img}}(F^t_{16}, F^s_{16}) \\ &+ \beta \cdot \mathcal{L}_{\text{mem}}(F^t_M, F^s_M),\end{aligned} \tag{8}$$

with α and β serving as the loss weights.

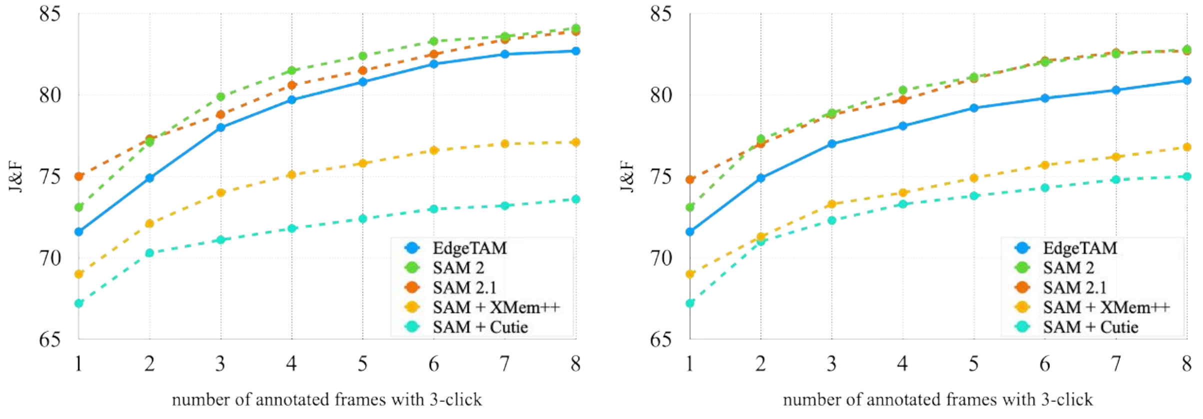
4. Experiments

4.1. Implementation Details

Training. In general, the training procedure of EdgeTAM follows SAM 2. We set the input resolution to 1024×1024. During the image segmentation pre-training stage, we train on the SA-1B dataset for 2 epochs with a batch size of 128. The loss weights for dice, focal, IoU, and $\mathcal{L}_{\text{img}}$ are 20, 1, 1, and 1, respectively. For each training sample, we allow a maximum of 64 objects and add 7 correction points iteratively. Random horizontal flip is the only data augmentation in this stage. For video segmentation training, we train on SA-V, a 10% randomly sampled subset of SA-1B, DAVIS, MOSE, and YTVOS for 130K iterations with a 256 batch size. The loss balancing factor for dice is 20 and 1 for focal, IoU, occlusion, $\mathcal{L}_{\text{img}}$, and $\mathcal{L}_{\text{mem}}$. Each video sample contains 8 frames with almost 3 objects and is augmented with horizontal flip, color jitter, affine, and grayscale transformations. More details can be found in the supplementary materials.

Progressive fine-tuning with longer training samples. Following SAM 2.1, we fine-tune the trained EdgeTAM model on 16-frame sequences. During the fine-tuning, we freeze the image encoder and do not apply distillation. The training set is the same as the video segmentation training stage but the total iterations are reduced to 1/3 of the original schedule. Furthermore, given that EdgeTAM consumes much less VRAM than SAM 2, we are able to further fine-tune the 16-frame model with 32-frame training samples with the same schedule. Note that the memory bank size stays the same and only the training samples become longer, so the inference cost remains the same.

Model. By default, we use RepViT-M1 [49] pre-trained on ImageNet [12] classification as the image encoder. We also experiment with ViT-Tiny [43] pre-trained with MAE [15] on ImageNet. The number of memory attention blocks is 2 and we allocate 256 learnable latents for both Global



(a) *Offline* average $\mathcal{J}\&\mathcal{F}$ across 9 datasets (3-click).

(b) *Online* average $\mathcal{J}\&\mathcal{F}$ across 9 datasets (3-click).

Figure 5. **Zero-shot PVS accuracy across 9 datasets in offline and online settings.**

Table 1. **Zero-shot accuracy on the SA task across 23 datasets.** We report 1 (5) click mIoU results. FPS is measured on iPhone. Our mix does not contain the internal datasets that SAM 2 uses.

Model	**Data**	**SA-23 All**	**SA-23 Image**	**SA-23 Video**	**FPS**
SAM	SA-1B	58.1 (81.3)	60.8 (82.1)	54.5 (80.3)	-
SAM 2	SA-1B	58.9 (81.7)	60.8 (82.1)	56.4 (81.2)	1.3
SAM 2	SAM2's mix	61.4 (83.7)	63.1 (83.9)	59.1 (83.3)	1.3
SAM 2.1	SAM2's mix	**61.9 (83.5)**	**63.3 (83.8)**	**60.1 (83.2)**	1.3
EdgeTAM	Our mix	55.5 (81.7)	56.0 (81.9)	54.8 (81.5)	**40.4**

Perceiver and 2D Spatial Perceiver. The memory bank sizes for frame-level memories and object pointers are 7 and 16 following SAM 2. The positional embeddings of Global Perceiver and 2D Spatial Perceiver are sinusoidal, and 2D-RoPE [40], respectively. We use the SAM2-HieraB+ as the teacher with the publicly available checkpoint[3].

4.2. Datasets

Training. We train on SA-1B [20], SA-V [35], DAVIS [34], MOSE [13], and YTVOS [56] datasets. SA-1B contains 11M images with 1.1B mask annotations in diverse granularities (in both part-level and object-level). The average resolution of images in SA-1B is 3300 $\times$ 4950 pixels. So far, it is the largest dataset available for image segmentation tasks. SA-V follows the criteria of SA-1B and collects 190.9K masklet annotations across 50.9K videos, which have an average duration of 14 seconds with 54%/46% indoor/outdoor scenes and are resampled to 24 FPS. Note that, the annotation frame rate is 6 FPS. Besides, 293/278 masklets from 155/150 videos are reserved as the SA-V val/test splits, which are manually picked to focus on challenging cases with fast-moving, complex occlusions, and disappearance.

Evaluation. Our evaluation can be split into three settings: (1) Promptable Video Segmentation (PVS), where the user can click on any frames in the video to indicate an object of interest; (2) Segment Anything (SA), which is same as PVS but works with images; (3) Semi-supervised Video Object Segmentation (VOS), where ground-truth masks on the first frame are available during inference. For the video task, we report $\mathcal{J}\&\mathcal{F}$ [34] and $\mathcal{G}$ [56] as the metric and for images, we use mIoU.

For PVS, we evaluate with the zero-shot protocol across 9 datasets with both online and offline modes. For SA, we evaluate on SA-23 [20], which consists of 23 open-source datasets in both video (each frame is considered as an image) and image domains. Finally, for VOS, we provide performance on the popular DAVIS 2017 [34], MOSE [13], and YouTubeVOS [56] val sets and the challenging SA-V val/test set [35].

4.3. Promptable Video Segmentation (PVS)

One of the key features of EdgeTAM is that it follows the same meta architecture of SAM 2, which enables it to perform promptable video segmentation with various user inputs on any frames. As shown in Fig. 5, we follow the same online and offline PVS settings as SAM 2, which simulate user interaction in the real world. The offline mode allows multiple times of playbacks to only add correction points on the frames with large errors, while the online mode only

[3]https://github.com/facebookresearch/sam2

Table 2. **Performance on the VOS task.** We report the $\mathcal{G}$ for YTVOS and $\mathcal{J}\&\mathcal{F}$ for other datasets. The FPS on A100 is obtained with torch compile. Nota that, for SAM 2, SAM 2.1, and EdgeTAM, we evaluate all the datasets with the same model.

Method	MOSE val	DAVIS 2017 val	SA-V val	SA-V test	YTVOS 2019 val	A100	V100	iPhone
STCN [9]	52.5	85.4	61.0	62.5	82.7	62.8	13.2	-
SwinB-AOT [61]	59.4	85.4	51.1	50.3	84.5	-	-	-
SwinB-DeAOT [59]	59.9	86.2	61.4	61.8	86.1	-	-	-
RDE [21]	46.8	84.2	51.8	53.9	81.9	88.8	24.4	-
XMem [8]	59.6	86.0	60.1	62.3	85.6	61.2	22.6	-
SimVOS-B [51]	-	88.0	44.2	44.1	84.2	-	3.3	-
JointFormer [66]	-	90.1	-	-	87.4	-	3.0	-
ISVOS [50]	-	88.2	-	-	86.3	-	5.8	-
DEVA [10]	66.0	87.0	55.4	56.2	85.4	65.2	25.3	-
Cutie-base [11]	69.9	87.9	60.7	62.7	87.0	65.0	36.4	-
Cutie-base+ [11]	71.7	88.1	61.3	62.8	87.5	57.2	17.9	-
SAM 2-B+ [35]	75.8	**90.9**	73.6	74.1	88.4	64.8	-	0.7
SAM 2.1-B+ [35]	**76.6**	90.2	**76.8**	**77.0**	**88.6**	64.1	-	0.7
EdgeTAM	70.0	87.7	72.3	71.7	86.2	**150.9**	-	**15.7**

Table 3. **Ablation Studies.**

(a) Effectiveness of each proposed component.

Memory Efficiency	Distill	SA-V val	SA-V test	FPS
-		63.5	62.1	2.5
Average Pooling		61.8	59.8	**15.7**
2D Perceiver		64.4	62.5	**15.7**
2D Perceiver	✓	**65.7**	**65.8**	**15.7**

(b) Latents allocation for 2D Perceiver.

Global Latents	2D Latents	SA-V val	SA-V test	FPS
0	0	63.5	62.1	2.5
256	0	62.0	60.6	**15.7**
0	256	63.1	62.4	**15.7**
256	256	**64.4**	**62.5**	**15.7**

(c) EdgeTAM with different backbones and # of memory attention blocks.

Image Encoder	Mem. Attn. Blocks	SA-V val	SA-V test	FPS
ViT-Tiny	1	65.1	64.1	8.5
ViT-Tiny	2	**67.9**	**66.0**	7.4
RepViT-M1	1	64.3	61.6	**22.2**
RepViT-M1	2	65.7	65.8	15.7
RepViT-M1	4	65.0	65.6	10.0

(d) Ablation on using self attention in 2D Perceiver.

Self-Attn in Perceiver	SA-V val	SA-V test	FPS
No	62.6	**62.7**	15.7
Yes	**64.4**	62.5	15.7

annotates the frames in a single forward pass. Compared to SAM + XMem++ and SAM + Cuite, EdgeTAM outperforms both across all settings with considerable margins. Besides, thanks to being trained in an end-to-end manner and distilled with the SAM 2 teacher the gap becomes larger as the number of annotated frames increases. Besides, even compared with the original SAM 2, EdgeTAM achieves comparable results despite being significantly smaller and faster.

4.4. Segment Anything (SA)

Both SAM 2 and EdgeTAM can function as image segmentation models with the memory module detached. As shown in Tab. 1, EdgeTAM achieves comparable mIoU performance with SAM and SAM 2, especially with more input points. For example, with five input points, on average, EdgeTAM even surpasses SAM-H (81.7 *v.s.* 81.3), which is dedicated to image segmentation. Note that, our EdgeTAM is not trained with the internal datasets that both SAM 2 and SAM 2.1 use. Given its real-time speed, EdgeTAM can be used as a unified on-device segmentation model for both images and videos.

4.5. Video Object Segmentation (VOS)

While EdgeTAM is trained only with the SA-V and SA-1B dataset, as shown in Tab. 2, on MOSE, DAVIS, and YTVOS, it is on par or surpasses previous state-of-the-art VOS models that are trained on these datasets. This demonstrates the robustness of EdgeTAM under the zero-shot setting. More importantly, it is impractical to deploy multiple models on device with one for certain types of data. Also, thanks to training on SA-V, EdgeTAM surpasses all its counterparts except for SAM 2 and SAM 2.1 on SA-V val and test. Note that, the masks in SA-V val/test have different granularities, while those of other datasets are at object-level. This shows the flexibility of EdgeTAM. In ad-

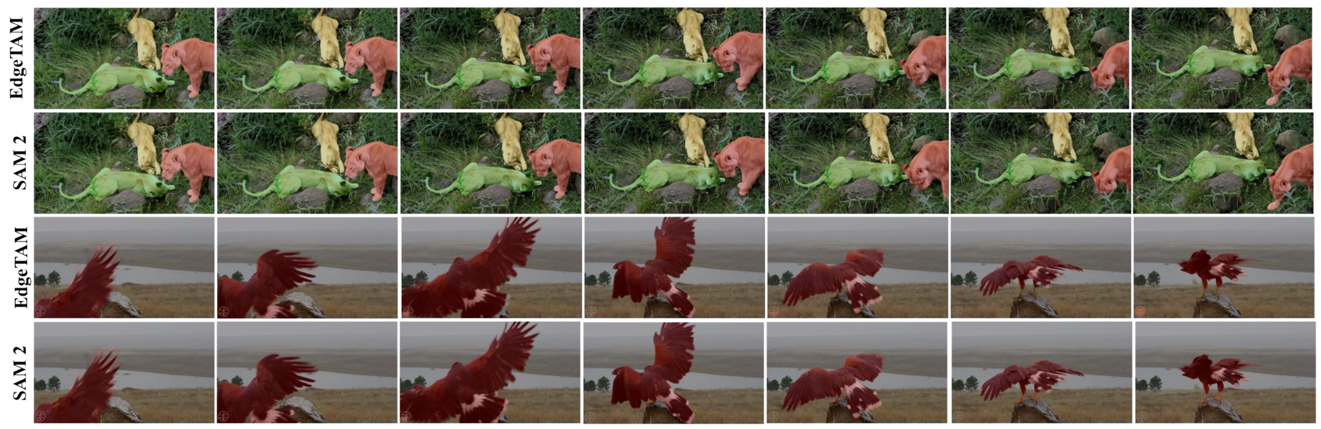


Figure 6. **Qualitative results of EdgeTAM compared with SAM 2.** In the upper example, we show tracking multiple instances from the same class, which also stay closely to each other. Our EdgeTAM delivers similar mask quality as SAM 2. In the lower example, we demonstrate a fast moving object with large distortion. While in general, EgdeTAM yields results that the boundary well, it outputs different granularities as SAM 2, not tracking the bird feet.

dition, for speed benchmarking, our main goal is inference on edge devices and we observe even with torch compile, the streaming multiprocessor utilization of EdgeTAM is still relatively low. Through the Torch profile, we find that on high-end GPU, the CPU (CUDA kernel launching) becomes the bottleneck for EdgeTAM. Thus, we encourage focusing on edge device latency, which EdgeTAM is designed for.

4.6. Ablations

For all the ablation studies, we train with one-third of the original training schedule (43k steps). As shown in Tab. 3(a), we first ablate the effectiveness of each proposed component. In the table, we set the baseline as RepViT-M1 with two memory attention blocks and we also compare with simply downsampling the spatial memories instead of using the 2D Perceiver. Experiments show that 2D Spatial Perceiver is both faster and more accurate than the baseline and 4×4 average pooling (0.4 to 2.7 better). Besides, the proposed distillation pipeline further improves the $\mathcal{J}\&\mathcal{F}$ on SA-V val and test by 1.3 and 3.3. Then, in Tab. 3(b), we vary the number of global and 2D latents and find that using both yields the best performance and speed-up. Note that, using 2D latents speed up the baseline by 6.3 × with better performance. Tab. 3(c) shows using 2D Perceiver on different combinations of image encoders and the number of memory attention blocks. And we opt for RepViT-M1 with two memory attentions for the best trade-off. Finally, in Tab. 3(d), we study whether to use self attention in the 2D Perceiver network. The motivation here is that as each 2D latent attends to a local patch that has no overlap with each other, incorporating self attention blocks will encourage the communication between 2D latents to yield better features. Our results verify this hypothesis.

4.7. Qualitative Results

In Fig. 6, we compare the visualization results of EdgeTAM and SAM 2 on the YouTubeVOS val dataset. We pick two representative examples, one with multiple instances from the same class gathering together, and the other with a fast-moving object with a large distortion. For the first example, EdgeTAM yields similar results as SAM 2 and keeps the identity of each instance throughout the clip. However, in the second example, we observe that EdgeTAM falls into a typical failure case that the tracking granularity might always follow SAM 2. In the example, EdgeTAM does not include the bird feet in the mask predictions given that in previous frames, the feet are not visible.

5. Conclusion

In this paper, we identify that the latency bottleneck of SAM 2 lies in the memory attention module and propose EdgeTAM to reduce the heavy overhead of cross attention with minimal performance degradation. Specifically, we propose 2D Spatial Perceiver to encode the densely stored frame-level memories into much smaller token sets while preserving their 2D spatial structure, which is essential for dense prediction tasks. As a plug-in module, 2D Spatial Perceiver can be applied to any SAM 2 variants. Besides, we also extend the knowledge distillation pipeline used in SAM for image segmentation to the video domain, which further improves the performance of EdgeTAM without inference-time cost. Our experiments show EdgeTAM nicely preserves the capability of SAM 2 across PVS, VOS, and SA tasks. More importantly, it runs 22× faster than SAM 2 and achieves 16 FPS on iPhone 15 Pro Max.

Acknowledgement. In this work, Bo Dai is supported in part by HKU Startup Fund.

References

[1] Core ml tools. https://github.com/apple/coremltools, 2021. 1

[2] Maksym Bekuzarov, Ariana Bermudez, Joon-Young Lee, and Hao Li. Xmem++: Production-level video segmentation from few annotated frames. In *ICCV*, 2023. 3

[3] Goutam Bhat, Felix Järemo Lawin, Martin Danelljan, Andreas Robinson, Michael Felsberg, Luc Van Gool, and Radu Timofte. Learning what to learn for video object segmentation. In *ECCV*, 2020. 3

[4] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *CVPR*, 2017. 3

[5] Keyan Chen, Chenyang Liu, Hao Chen, Haotian Zhang, Wenyuan Li, Zhengxia Zou, and Zhenwei Shi. Rsprompter: Learning to prompt for remote sensing instance segmentation based on visual foundation model. *IEEE Transactions on Geoscience and Remote Sensing*, 2024. 1, 3

[6] Tianrun Chen, Ankang Lu, Lanyun Zhu, Chaotao Ding, Chunan Yu, Deyi Ji, Zejian Li, Lingyun Sun, Papa Mao, and Ying Zang. Sam2-adapter: Evaluating & adapting segment anything 2 in downstream tasks: Camouflage, shadow, medical image segmentation, and more. *arXiv preprint arXiv:2408.04579*, 2024. 3

[7] Yuhua Chen, Jordi Pont-Tuset, Alberto Montes, and Luc Van Gool. Blazingly fast video object segmentation with pixel-wise metric learning. In *CVPR*, 2018. 3

[8] Ho Kei Cheng and Alexander G Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *ECCV*, 2022. 3, 7

[9] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. In *NeurIPS*, 2021. 3, 7

[10] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. Tracking anything with decoupled video segmentation. In *ICCV*, 2023. 7

[11] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. Putting the object back into video object segmentation. In *CVPR*, 2024. 3, 7

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[13] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip HS Torr, and Song Bai. Mose: A new dataset for video object segmentation in complex scenes. In *ICCV*, 2023. 6

[14] Brendan Duke, Abdalla Ahmed, Christian Wolf, Parham Aarabi, and Graham W Taylor. Sstvos: Sparse spatiotemporal transformers for video object segmentation. In *CVPR*, 2021. 3

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 5

[16] Yuan-Ting Hu, Jia-Bin Huang, and Alexander Schwing. Maskrnn: Instance level video object segmentation. In *NeurIPS*, 2017. 3

[17] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G Schwing. Videomatch: Matching based video object segmentation. In *ECCV*, 2018. 3

[18] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021. 2, 4

[19] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. In *ICLR*, 2022. 2, 4

[20] 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 *CVPR*, 2023. 1, 2, 3, 6

[21] Mingxing Li, Li Hu, Zhiwei Xiong, Bang Zhang, Pan Pan, and Dong Liu. Recurrent dynamic embedding for video object segmentation. In *CVPR*, 2022. 3, 7

[22] Yu Li, Zhuoran Shen, and Ying Shan. Fast video object segmentation using the global context module. In *ECCV*, 2020. 3

[23] Yongqing Liang, Xin Li, Navid Jafari, and Jim Chen. Video object segmentation with adaptive feature bank and uncertain-region refinement. In *NeurIPS*, 2020. 3

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 5

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 4

[26] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *ACCV*, 2018. 3

[27] Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 2024. 1, 3

[28] K-K Maninis, Sergi Caelles, Yuhua Chen, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. Video object segmentation without temporal information. *IEEE TPAMI*, 2018. 3

[29] Tim Meinhardt and Laura Leal-Taixé. Make one-shot video object segmentation efficient again. In *NeurIPS*, 2020. 3

[30] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *CVPR*, 2018. 3

[31] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *ICCV*, 2019. 3

[32] Hyojin Park, Jayeon Yoo, Seohyeong Jeong, Ganesh Venkatesh, and Nojun Kwak. Learning dynamic network using a reuse gate function in semi-supervised video object segmentation. In *CVPR*, 2021. 3

[33] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *CVPR*, 2017. 3

[34] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 6

[35] 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. 1, 2, 3, 4, 6, 7

[36] Andreas Robinson, Felix Jaremo Lawin, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Learning fast and robust target models for video object segmentation. In *CVPR*, 2020. 3

[37] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, et al. Hiera: A hierarchical vision transformer without the bells-and-whistles. In *ICML*, 2023. 3

[38] Abdelrahman Shaker, Syed Talal Wasim, Martin Danelljan, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Efficient video object segmentation via modulated cross-attention memory. *arXiv preprint arXiv:2403.17937*, 2024. 3, 4

[39] Jae Shin Yoon, Francois Rameau, Junsik Kim, Seokju Lee, Seunghak Shin, and In So Kweon. Pixel-level matching for video object segmentation using convolutional neural networks. In *ICCV*, 2017. 3

[40] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024. 4, 6

[41] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *MICCAIW*, 2017. 5

[42] Lv Tang, Haoke Xiao, and Bo Li. Can sam segment anything? when sam meets camouflaged object detection. *arXiv preprint arXiv:2304.04709*, 2023. 1, 3

[43] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1, 5

[44] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 3

[45] Carles Ventura, Miriam Bellver, Andreu Girbau, Amaia Salvador, Ferran Marques, and Xavier Giro-i Nieto. Rvos: End-to-end recurrent network for video object segmentation. In *CVPR*, 2019. 3

[46] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017. 3

[47] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *CVPR*, 2019. 3

[48] Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Repvit-sam: Towards real-time segmenting anything. *arXiv preprint arXiv:2312.05760*, 2023. 3

[49] Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Repvit: Revisiting mobile cnn from vit perspective. In *CVPR*, 2024. 1, 2, 4, 5

[50] 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 *CVPR*, 2023. 3, 7

[51] Qiangqiang Wu, Tianyu Yang, Wei Wu, and Antoni B Chan. Scalable video object segmentation with simplified framework. In *ICCV*, 2023. 3, 7

[52] Huaxin Xiao, Jiashi Feng, Guosheng Lin, Yu Liu, and Maojun Zhang. Monet: Deep motion exploitation for video object segmentation. In *CVPR*, 2018. 3

[53] Junyu Xie, Charig Yang, Weidi Xie, and Andrew Zisserman. Moving object segmentation: All you need is sam (and flow). *arXiv preprint arXiv:2404.12389*, 2024. 1, 3

[54] Yunyang Xiong, Bala Varadarajan, Lemeng Wu, Xiaoyu Xiang, Fanyi Xiao, Chenchen Zhu, Xiaoliang Dai, Dilin Wang, Fei Sun, Forrest Iandola, et al. Efficientsam: Leveraged masked image pretraining for efficient segment anything. In *CVPR*, 2024. 1, 3, 5

[55] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. Youtube-vos: Sequence-to-sequence video object segmentation. In *ECCV*, 2018. 3

[56] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018. 6

[57] Xiaohao Xu, Jinglu Wang, Xiao Li, and Yan Lu. Reliable propagation-correction modulation for video object segmentation. In *AAAI*, 2022. 3

[58] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. Efficient video object segmentation via network modulation. In *CVPR*, 2018. 3

[59] Zongxin Yang and Yi Yang. Decoupling features in hierarchical propagation for video object segmentation. In *NeurIPS*, 2022. 3, 7

[60] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by foreground-background integration. In *ECCV*, 2020. 3

[61] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. In *NeurIPS*, 2021. 3, 7

[62] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by multi-scale foreground-background integration. *IEEE TPAMI*, 2021. 3

[63] Zongxin Yang, Jiaxu Miao, Yunchao Wei, Wenguan Wang, Xiaohan Wang, and Yi Yang. Scalable video object segmentation with identification mechanism. *IEEE TPAMI*, 2024. 3

[64] Jieming Yu, An Wang, Wenzhen Dong, Mengya Xu, Mobarakol Islam, Jie Wang, Long Bai, and Hongliang Ren. Sam 2 in robotic surgery: An empirical evaluation for robustness and generalization in surgical video segmentation. *arXiv preprint arXiv:2408.04593*, 2024. 3

[65] 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*, 2023. 1, 3, 5

[66] 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*, 2023. 3, 7

[67] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023. 3

[68] 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. 1, 3, 4, 5