On Efficient Variants of Segment Anything Model: A Survey

Xiaorui Sun[1] Jun Liu[2] Heng Tao Shen[1,3] Xiaofeng Zhu[1] Ping Hu[1*]
[1]School of Computer Science and Engineering, UESTC
[2]School of Computing and Communications, Lancaster University
[3]School of Computer Science and Technology, Tongji University

arXiv:2410.04960v1 [cs.CV] 7 Oct 2024

Abstract

The Segment Anything Model (SAM) is a foundational model for image segmentation tasks, known for its strong generalization across diverse applications. However, its impressive performance comes with significant computational and resource demands, making it challenging to deploy in resource-limited environments such as mobile devices. To address this, a variety of SAM variants have been proposed to enhance efficiency without sacrificing accuracy. This survey provides the first comprehensive review of these efficient SAM variants. We begin by exploring the motivations driving this research. We then present core techniques used in SAM and model acceleration. This is followed by an in-depth analysis of various acceleration strategies, categorized by approach. Finally, we offer a unified and extensive evaluation of these methods, assessing their efficiency and accuracy on representative benchmarks, and providing a clear comparison of their overall performance.

1. Introduction

The emergence of foundation models [8] has led to a thorough revolution in the field of artificial intelligence (AI). Foundation models are large-scale neural networks pretrained on massive data, which have powerful representing ability and strong generalization to perform on various tasks. In the field of natural language processing (NLP), the recently popular research trend is about the large language models (LLMs) [116, 160], which has been witnessed great development with prominent works like OpenAI's GPT series [1, 9], Google's PaLM series [3, 21] and Meta's LLaMA series [111, 112]. Meanwhile, the success of Vision Transformer (ViT) [29] which firstly introduces the Transformer [115] architecture into the field of computer vision (CV), has brought up a new era for vision foundation models (VFMs). Vision-Language foundation models like CLIP [89], LLaVA [67], and Video-ChatGPT [79] etc., which aim at aligning the vision and language modalities, have demonstrated promising performance on plenty of downstream vision tasks[34, 58, 110, 167].

Recently, a novel foundation model for general image segmentation, the Segment Anything Model (SAM), was proposed by Meta [55]. Being fully trained on their proposed SA-1B dataset, which consists of more than one billion masks and eleven million images, with the task of achieving valid segmentation given any prompt (e.g. points, boxes, masks and text), SAM is able to well generalize to a wide range of downstream tasks (e.g. edge detection, object proposal and instance segmentation) with proper prompts. Shortly after its emergence, SAM has garnered significant attention from the research community and results in a surge of related works exploring SAM's generalization ability in various scenarios [14, 15, 78], including different image segmentation tasks [2, 57, 95, 130], video analysis tasks [42, 76, 137, 154] and 3D vision tasks [28, 100, 134]. With the huge success of SAM, the upgraded Segment Anything Model 2 (SAM 2) [93] is further proposed, aiming for efficient segmentation in both images and videos. SAM 2 introduces the streaming memory mechanism to extend SAM's ability to videos. It is trained on both the SA-1B dataset and SA-V dataset which is their newly collected video segmentation dataset. As a result, SAM 2 naturally holds the powerful generalization and surpasses SAM when handling segmentation tasks [70, 99, 113, 135, 140].

Despite its success in a wide range of applications, the original Segment Anything Model (SAM), particularly SAM-H, faces significant limitations due to its slow runtime and high computational cost. These challenges become even more pronounced when SAM is deployed in resource-constrained or real-time environments, such as edge devices and mobile applications. As the demand for deploying machine learning models in practical, resource-constrained scenarios continues to grow, SAM's current design proves inefficient for widespread use. This has led to a pressing need for more lightweight and efficient variants that can maintain the model's powerful segmentation capabilities while addressing these constraints. The challenge

*email:chinahuping@gmail.com

of optimizing SAM for efficiency is further amplified by the increasing emphasis on real-time applications, mobile platforms and embedded systems, where computational resources are limited. As the community strives to overcome these obstacles [18, 77, 103, 114, 118, 147, 161, 164], a comprehensive understanding of the latest advancements in making SAM more efficient is crucial. Consequently, it is both timely and necessary to conduct a detailed survey of the ongoing efforts aimed at improving SAM's efficiency and extending its applicability across diverse environments.

With the growing research related to SAM, several surveys have been presented to provide an overview from different perspectives [149–152, 155, 157]. However, these existing surveys primarily focus on SAM's downstream applications and exhibit several limitations: 1) None of them address the emerging field of improving SAM's efficiency, which is gaining significant attention and is crucial for practical deployment in real-world applications. 2) With the exception of [151], these surveys lack a structured taxonomy that would allow for clearer organization and reference. 3) Most prior surveys focus on collecting and describing SAM-based methods but lack a systematic evaluation or comparison of these approaches. To address these gaps, we conduct this survey, not only to comprehensively review the development of efficient Segment Anything Models but also to fairly evaluate and compare them. The main contributions of this work can be summarized as follows:

- We provide a systematic review of the efficient SAM variants designed to accelerate segmentation tasks. We introduce a well-structured taxonomy for the methods, categorizing them based on the acceleration strategies they employ. To the best of our knowledge, this is the first survey focused specifically on this area.
- We offer comprehensive evaluations and comparisons of the efficiency and accuracy of these variants, aiming to assist researchers in selecting models that best meet their performance and application needs.
- We propose several potential directions for future research, offering insights that may inspire readers to contribute to the continued advancement of this field.

The rest of this survey is organized as follows. In Section 2, we begin by introducing the background of the original SAM, followed by a review of efficient backbones for visual representation and model compression techniques that can be applied to enhance SAM's efficiency. In Section 3, we categorize the existing methodologies based on their objectives and techniques, providing a detailed review of each category. We also discuss several potential research directions for further accelerating SAM. In Section 4, we conduct a fair evaluation of these models in terms of efficiency, accuracy, and the resulting trade-offs. Finally, in Section 5, we provide a brief summary of this survey.

2. Preliminaries

In this section, we will first introduce the details and applications of the Segment Anything Model in Section 2.1. In Section 2.2, we review efficient backbones that could potentially replace SAM's image encoder, and in Section 2.3, we examine model compression techniques that could be applied to SAM. For a more comprehensive understanding of efficient backbones and model compression methods, we recommend referring to surveys [86, 132].

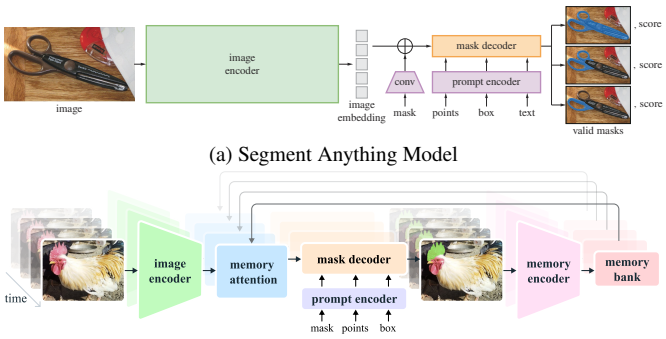
2.1. Segment Anything Model

Segment Anything Model [55] is a powerful foundation model in the image segmentation field, which can unify most image segmentation tasks by a fundamental segmentation task, i.e. the promptable segmentation task, with prompt engineering [68]. Another significant contribution of this project is the SA-1B dataset, which has over 1B masks from 11M licensed and privacy-preserving images. Trained with such abundant and high-quality data, SAM is expected to have strong robustness and generalization. The huge potential of SAM has quickly sparked researcher's interest in both exploring its ability in a wide range of real-world applications and improving its architecture to segment more efficiently or accurately.

Recently, Segment Anything Model 2 (SAM 2) [93] is proposed as a successor with a focus on efficient promptable visual segmentation (PVS) for both images and videos. To enable SAM 2 to segment anything in videos, researchers introduce the streaming memory mechanism into SAM's original architecture. SAM 2 is trained from scratch by two stages: 1) Pre-training on SA-1B dataset with the promptable segmentation task for images; 2) Training on mixed data with the promptable segmentation task for images and videos. Similar to SAM, researchers build up a data engine to generate a large-scale dataset for video segmentation, named the SA-V dataset. With both manually annotated and automatically generated masklets (object segmentation in videos), SA-V finally collects 642.6K masklets across 50.9K videos. In this survey, we consider SAM 2 as an efficient SAM variant and include it in the evaluation and comparison.

2.1.1 Model

SAM consists of three components: an image encoder, a prompt decoder and a mask decoder, as shown in Fig. 1 (a). The image encoder is a MAE [44] pre-trained Vision Transformer(ViT) with minimal modification. It takes the preprocessed images as input and outputs an image embedding for each image. The prompt decoder is to embed prompts: points, boxes, masks and text. The two embeddings are then fed into the light-weight mask decoder, which is built upon two modified Transformer decoder blocks [115] and



(a) Segment Anything Model

(b) Segment Anything Model 2

Figure 1. The architectures of (a) SAM [55] and (b) the recent proposed SAM 2 [93].

a few prediction heads, to generate valid masks. Based on SAM's architecture, SAM 2 additionally introduces a streaming memory mechanism. Specifically, a memory encoder, a memory bank and a memory attention module. The structure of SAM 2 is illustrated in Fig. 1 (b). With the memory mechanism, SAM 2 can handling videos frame by frame. The memory encoder is to generate memory of the prediction of the current frame and send it to the memory bank. The memory bank stores memories of recent prediction, feature maps of prompted frames and high-level semantic information of the target objects (i.e. object pointers). The memory attention mechanism is to make the image embedding from image encoder fully interact with the information from memory bank, resulting in the refined embedding. Apart from the memory mechanism, SAM 2 also adopts the MAE pre-trained Hiera [96] as image encoder which is more efficient than ViT-H, with expectation for faster speed.

2.1.2 Task

The promptable segmentation task is proposed as the fundamental task of SAM, whose goal is to return a valid mask with any given prompt (e.g. a point, a box, a mask, and text). This task is not only the objective during SAM's training, but also the base that enables SAM to solve various downstream tasks. Another important task is the all-mask generation which segments all objects in a picture. It is achieved by prompting SAM with a grid of points and then predicting masks based on these dense prompts. It is also a key procedure in the last stage of the data engine, which aims at enhancing the diversity of masks in SA-1B. As illustrated in Fig. 2, the promptable segmentation task is called **Segment Anything (SegAny)**, and the all-masks generation task is termed as **Segment Everything (SegEvery)**. These two tasks have summarized SAM's segmentation ability and have led to two research directions for enhancing SAM's efficiency. In this survey, We follow the two definitions, exploring SAM-based efficient variants' performance in both SegAny and SegEvery task.

2.1.3 Application

As SAM and its successor SAM 2 have demonstrated strong generalization in plenty of zero-shot downstream tasks [55, 93], the community is diving into exploring their application in more scenarios.

One of the major applications of SAM is medical image segmentation. According to [157], works in this area can be classified into two types. Some aim at testing SAM's segmentation performance in CT images [46, 95], MRI images [82], pathological images [25], and so on. Others focus on improving the adaptation of SAM to these tasks by fine-tuning [64, 78], auto-prompting [48, 98] or framework modification [156]. Furthermore, works like [35, 57] attempt to improve medical SAM methods' efficiency. SAM is also applied to object detection across a diverse range of real-world scenes [51, 149], including crack detection [2] and crater detection [36] in civil infrastructure defect assessment, crop disease and pest detection [63] in agriculture, anomaly detection [11] and remote sensing [85, 119]. Moreover, Segment Anything has been adapted to Edit Everything [130], Inpaint Anything [141] and Caption Anything [121] to handle image editing tasks.

Apart from image segmentation tasks, SAM has been widely applied to various video tasks [151]. A large portion of the works concentrate on two basic tasks: video object segmentation (VOS) [19, 154, 158] and video object tracking (VOT) [20, 92, 137]. Researchers also ex-

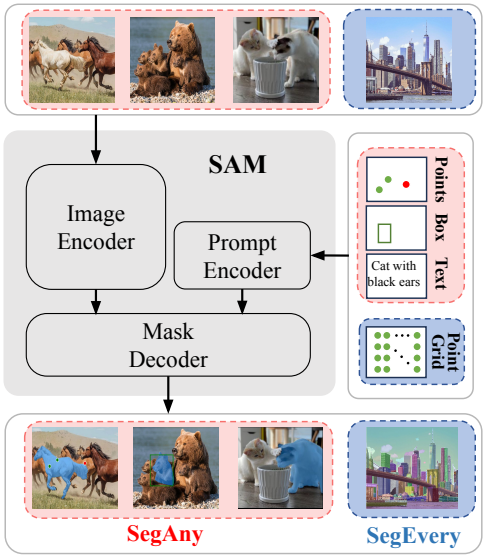


Figure 2. Illustration of the Segment Anything task (SegAny) and the Segment Everything task (SegEvery).

plore SAM's application in tasks related to generation, for example, video super-resolution [76] and video dataset annotation generation [24, 42]. Besides these, SAM is further utilized as an intermediate tool in video editing tasks [126, 163]. Beyond 2D tasks, SAM is also extended to 3D vision filed. [100] applies SAM to 3D reconstruction and [28] applies it to 3D point cloud segmentation. The recent work [134] aims at achieving real-time segmentation for any 3D thing in an online setting. For the recent proposed SAM 2, there are already some studies exploring its application in both image and video tasks. A popular trend is to apply SAM 2 to medical images and video tasks [155]. Works like [27, 99, 135] evaluate SAM 2's performance in medical images in both 2D and 3D modalities, while some others [75, 140] test it on surgical video segmentation tasks. Researchers are also seeking for strategies to adapt SAM 2 to medical tasks better [136, 166]. Besides these, SAM 2 has been applied to some specific image segmentation tasks like digital pathology semantic segmentation [153], mesh part segmentation [109] and solar panels segmentation [91]. Moreover, several works [70, 113] have utilized SAM 2 on the challenging Large-scale Video Object Segmentation (LSVOS) task and achieved good results.

2.1.4 Limitation

Although SAM has demonstrated promising performance across various tasks, it still faces two key challenges in practical applications: 1) it often struggles to predict complete masks for fine structures, leading to imprecise boundaries; and 2) it is not real-time and remains resource-intensive, particularly when using heavy image encoders like ViT-H. To address these issues, works such as [49, 54] aim to improve mask quality by utilizing high-resolution images, while others [131, 147, 161, 164] focus on creating more efficient architectures to reduce SAM's time and resource consumption. Previous surveys [78, 100, 158] have explored recent advances in enhancing SAM for higher-quality results. In this survey, we focus specifically on efforts to improve SAM's efficiency.

2.2. Efficient Backbone

SAM's inefficiency primarily stems from its heavy-weight image encoder. The sizes of SAM's image encoder are detailed in Tab. 1, with further estimates of SAM's total parameters provided in Section 4.1. As shown, in SAM-H, the ViT-H image encoder contains approximately 632M parameters, while the total model size is 641M, meaning the image encoder accounts for most of the model's parameters. Even in the smallest variant, SAM-B, the image encoder still represents over 90% of the total parameters. Therefore, a straightforward yet effective method to accelerate SAM is to replace the large image encoder with more efficient backbones. These efficient backbones can include pure convolutional neural networks (CNNs) such as [43, 94, 117], efficient vision Transformer architectures like [10, 61, 127], and recent Transformer-alternative models like [87].

Parameters	SAM-H	SAM-L	SAM-B
Image Encoder	632M	307M	86M
Prompt Encoder		0.006M	

Table 1. Parameters of SAM's encoders.

2.2.1 Efficient Vision Transformer

Efforts to improve the efficiency of Vision Transformers can generally be categorized into two approaches: 1) Designing more efficient architectures; 2) Refactoring the attention mechanism.

To reduce computational cost from a structural perspective, MobileViT [80], a pioneering hybrid architecture, creatively integrates CNN blocks (MobileNetV2 blocks [97]) with Transformer blocks into a single model. Subsequent works such as [10, 62, 127] basically follow this idea to build up efficient ViTs with hybrid structures, which have been widely used to substitute SAM's heavy image encoder. In [103, 161], TinyViT [127] serves as the efficient backbone, while in [162] and [159], EfficientFormerV2 [62] and EfficientViT [10] supplant SAM's original image encoder, respectively. Another influential ViT design, MetaFormer [142], which abstracts the attention mechanism into a broader concept called the token mixer, can deliver performance on par with Transformers using various token mixers. The simplest variant, PoolFormer, which uses pooling operations as the token mixer without introducing additional learnable parameters, has been leveraged as the base architecture to develop the light-weight image encoder for Lite-SAM [33].

Researchers have also made significant progress in optimizing the attention mechanism. It has been observed that the softmax operation within the attention mechanism contributes significantly to the overall computational cost. In EfficientViT [10], a novel ReLU linear attention is proposed to achieve global receptive field with higher efficiency. This efficient backbone is further adopted in [159] to accelerate SAM. Improvements to the attention mechanism have also been made at the hardware level. FlashAttention [23] significantly reduces computation costs through techniques such as tiling, kernel fusion, and recomputation. And it is utilized in SAM acceleration works [101, 104] to reduce memory need and enhance computation efficiency.

2.2.2 Transformer-alternative Models

While Transformers currently dominate both the language and vision fields, several newly proposed models have shown the potential to surpass them in terms of efficiency and performance.

The Receptance Weighted Key Value (RWKV) model [87], which combines the strengths of both recurrent neural networks (RNNs) and Transformers, achieves linear time complexity as sequence length increases. RWKV is well-suited for handling long sequence processing challenges. To facilitate global information interaction, RWKV replaces the traditional attention mechanism, which has quadratic complexity, with the more efficient WKV Operator and output gating mechanism. These are formulated as follows,

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} \odot v_i + e^{u+k_t} \odot v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}} \quad (1)$$

$$o_t = W_o \cdot (\sigma(r_t) \odot wkv_t) \quad (2)$$

where r, k, v represents the shifted tokens of receptance, key and value respectively and W refers to weight.

RWKV has also been extended to the vision tasks. The Vision-RWKV (VRWKV) model [30] demonstrates performance comparable to Vision Transformers (ViT) but with greater efficiency. To adapt RWKV from 1D sequences to 2D images, Q-shift tokens is introduced to fuse neighborhood information in four directions. In [145], a RWKV-based variant of SAM has achieved outstanding performance in efficiency, by adopting the efficient backbone mixed of MobileNetV2 blocks [97] and VRWKV blocks.

2.3. Model Compression

Model compression encompasses a range of techniques aimed at reducing the size and computational complexity of models, making it essential for deploying large models in real-world applications where computational resources are limited. The four primary methods for model compression and acceleration are knowledge distillation, quantization, pruning, and low-rank factorization.

2.3.1 Knowledge Distillation

Knowledge distillation (KD) [45] was first introduced as a solution for deploying large, complex neural networks in resource-constrained environments. The core concept is to transfer the knowledge and representation capability from a larger, well-trained model (the teacher model) to a smaller, more efficient model (the student model).

When applying KD to accelerate SAM, the goal is to distill the knowledge from the original, larger SAM and impart it to more efficient SAM-like models. Given SAM's encoder-decoder architecture, KD can generally be categorized into two approaches: distilling the entire SAM model or distilling the image encoder alone. Most works, such as [84, 103, 117, 147, 159], focus on distilling only the efficient backbone, while retaining the original SAM's prompt encoder and mask decoder. However, other approaches, like [162, 164], aim to distill the entire model by supervising the outputs of both the encoder and decoder.

2.3.2 Quantization

Quantization is the process to convert model's high precision weights/activation values X (e.g. 32-bit floating point) to low precision formats X_q (e.g. 16-bit floating point, 8-bit integer). One of the widely used quantization function, uniform symmetric quantization, is formulated as follows,

$$X_q = Clamp(round(\frac{X}{s}), -2^b - 1, 2^{b-1} - 1) \quad (3)$$

$$X \approx \hat{X} = X_q * s \quad (4)$$

where b, s refers to the bit number in low precision and the scaling factor respectively, $Clamp(x, a1, a2)$ clips integer x to $a1/a2$ when $x \leq a1/x \geq a2$. There are two primary types of quantization for neural networks: Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). PTQ applies quantization to fully trained models using a small calibration dataset to mitigate accuracy loss, while QAT adapts models to low precision during training, requiring the full training dataset. As a result, QAT is generally more costly and inefficient compared to PTQ. Numerous PTQ methods have been developed for CNNs [6, 83, 122] and Transformer-based architectures [32, 72, 139, 146]. In the context of accelerating SAM, works like [77, 103] utilize PTQ techniques to compress SAM with targeted strategies for improved efficiency.

2.3.3 Pruning

Model pruning reduces a model's size and complexity by eliminating redundant weights or connections, while aiming to maintain accuracy as much as possible. Pruning methods are typically categorized into two types: structured pruning [4, 123] and unstructured pruning [16, 56].

Structured pruning removes parameters in groups based on specific criteria, targeting substructures such as channels, layers, or blocks in a systematic manner. In contrast, unstructured pruning focuses on individual weights, often resulting in a sparse and fragmented network. However, unstructured pruning may not lead to effective acceleration on general hardware due to the irregularity of the remaining network structure. In [18], structured pruning is applied to lighten SAM, significantly reducing the model's size while

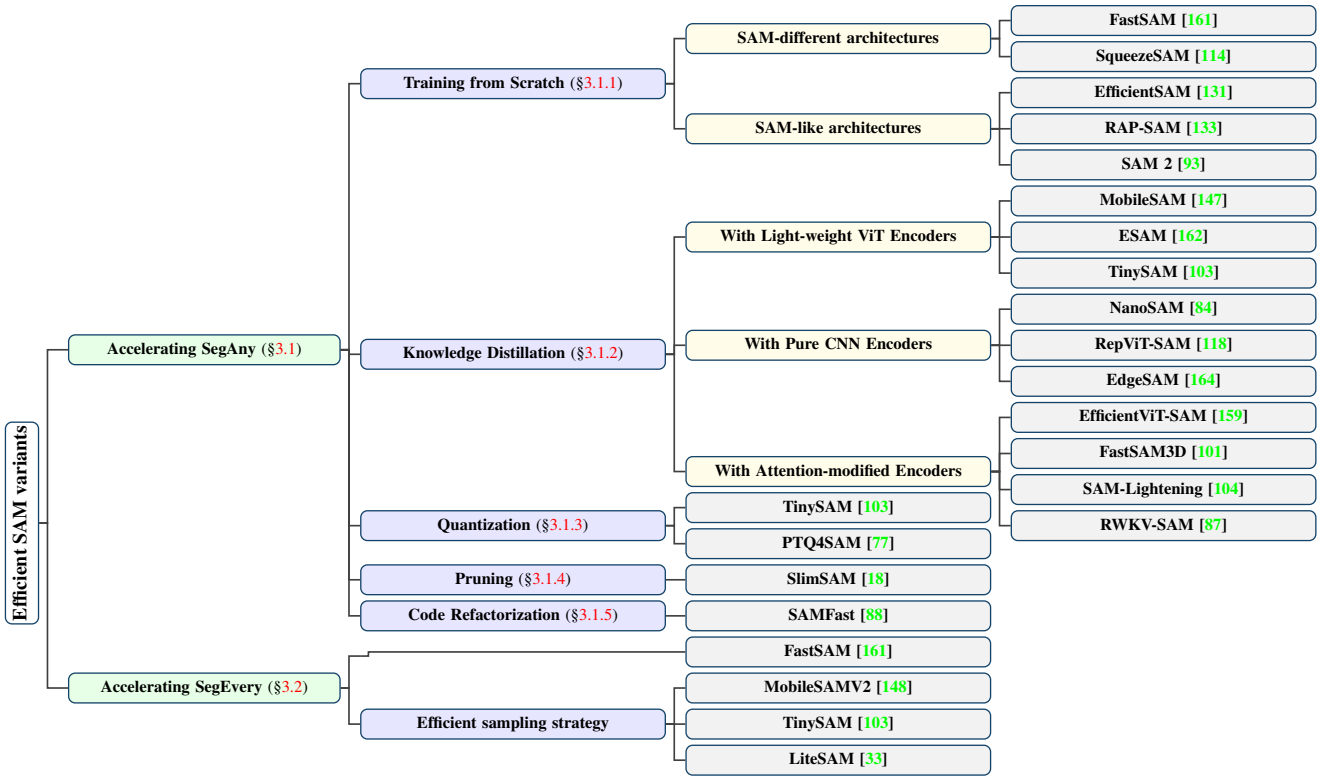


Figure 3. Taxonomy of Efficient Variants of Segment Anything Model (SAM).

preserving most of SAM's capabilities by removing large amounts of redundant weights.

2.3.4 Low-Rank Factorization

Low-rank factorization is the technique that decomposes the high dimension matrix into the product of low dimension matrices, aiming at reducing parameters of models for less space occupation and higher computational efficiency. A commonly used method of low-rank factorization is the singular value decomposition (SVD). It factorizes the complex matrix $A \in \mathbb{R}^{m \times n}$ into three matrices and the calculation can be further rewritten as follows:

$$A = U \Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{5}$$

where $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix with the none-zero singular values of A, while $r \leq \min\{m, n\}$ refers to the rank of A. The low-rank factorization has been widely applied in deep neural networks, such as CNNs [108, 144], LSTMs [37, 124], and Transformer-based models [31, 41, 107]. However, to date, no studies have applied this technique to compress and accelerate SAM, presenting a potential future research direction.

3. Efficient Variants of SAM

This section reviews the efforts to develop lightweight, efficient SAM-like models that have emerged since SAM gained prominence. These works aim to reduce the model's high computational cost and enable efficient performance, while preserving SAM's robust segmentation capabilities and generalization strength. As outlined in Section 2.1.2, SAM addresses two primary tasks including Segment Anything (SegAny) and Segment Everything (SegEvery). Accordingly, we discuss the research aimed at improving each task separately: Section 3.1 focuses on accelerating the SegAny task, and Section 3.2 covers efforts to accelerate the SegEvery task. Notably, some methods are applicable to both tasks, and we discuss these contributions individually. Additionally, we categorize all models into different classes based on the techniques they employ, and the taxonomy of methodologies is presented in Fig. 3. Finally, in Section 3.3, we outline four potential directions for future research in this area.

3.1. Accelerating SegAny Tasks

As analyzed in Section 2.2, the primary bottleneck of the SegAny task lies in SAM's heavy architecture. One straightforward solution is to replace the encoder with a more efficient backbone. Alternatively, adopting a different

architecture that retains the same segmentation capabilities as SAM is another approach. Works following these strategies either involve training lightweight models entirely from scratch (discussed in Section 3.1.1) or training models using knowledge distillation with suitable supervision (covered in Section 3.1.2). Additionally, some research explores methods such as quantization, pruning, or localized optimizations to compress SAM directly, without replacing the encoder or constructing a new architecture. These efforts will be reviewed in Sections 3.1.3, 3.1.4, and 3.1.5, respectively.

3.1.1 Training from Scratch

This subsection focuses on works that train SAM variants entirely from scratch. Based on their architecture, these models can be categorized into two types: SAM-different architectures and SAM-like architectures. We will explore each type in detail, following this classification.

FastSAM [161] is one of the first SAM variants that does not rely on SAM's original Encoder-Decoder architecture. To achieve faster segmentation, it divides the SegAny task into two sub-tasks: all-instance segmentation and prompt-guided selection. Since instance segmentation has been effectively addressed by many CNN-based methods, FastSAM offers improved efficiency compared to the Transformer-based SAM. For instance segmentation, FastSAM employs the YOLOv8-Seg [53] model, using the YOLACT method [7] for enhanced performance. FastSAM can reliably predict objects of interest using points, boxes, or text as prompts. In addition to accelerating the SegAny task, FastSAM also excels in the SegEvery task, as this can be efficiently achieved alongside all-instance segmentation. However, as an early efficient variant of SAM, FastSAM still has some limitations, such as producing low-quality masks for smaller objects and generating masks with less smooth boundaries. Despite these shortcomings, FastSAM marks significant progress by introducing a CNN-based architecture into this domain. The architecture of FastSAM is illustrated in Fig 4.

Building on the successful application of CNNs in FastSAM, Varadarajan et al. introduced SqueezeSAM [114], which further replaces SAM's Transformer-based architecture with a U-Net structure [94]. U-Net is composed of an encoder for feature extraction and a decoder for information recovery. SqueezeSAM retains the general U-Net architecture but incorporates two Transformer layers at the lowest scale of the U-Net to strike a balance between speed and accuracy. Additionally, SqueezeSAM features several micro-level optimizations, such as capping the output channels at 256, using BatchNorm [50] instead of LayerNorm [5] for efficiency, and introducing skip connections [43] between the encoder and decoder. A unique challenge for SqueezeSAM lies in handling prompts. Unlike SAM, where prompt

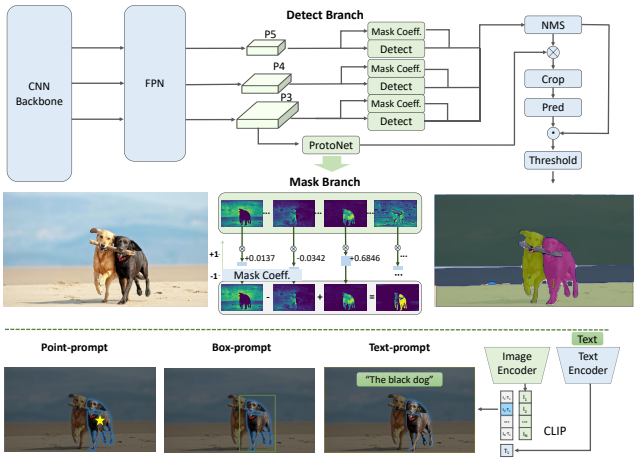


Figure 4. The architecture of FastSAM [161]. It takes two stages to achieve segmenting anything: all-instance segmentation (the upper part) and prompt-guided selection (the lower part). It is worth noting that FastSAM have introduced CLIP[89] to support text prompt.

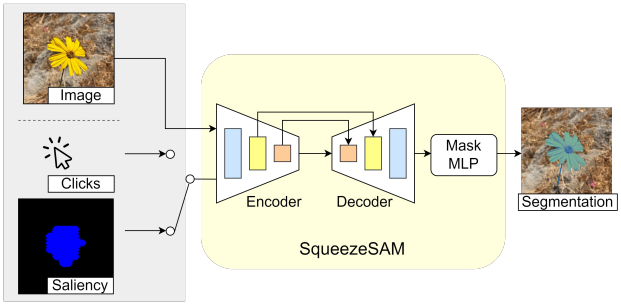


Figure 5. The architecture of SqueezeSAM [114]. It replaces the Transformer-based encoder-decoder structure with U-Net backbone[94]. Clicks from users and masks of salience objects are fed into SqueezeSAM with the input image to achieve interactive segmentation.

tokens are utilized at the decoding stage, SqueezeSAM adopts an early fusion strategy, adding encoded prompts as additional input channels before feeding them into the encoder. The model is trained from scratch using the SA-1B dataset, with data augmentation techniques addressing low-quality data issues. SqueezeSAM is primarily designed for deployment in photography applications, where efficient interactive segmentation is needed. As depicted in Fig. 5, its workflow involves generating an initial mask of the salient object, followed by fine-grained segmentation refined by user clicks.

Instead of introducing an entirely new network, EfficientSAM [131] retains SAM's original architecture but replaces the image encoder. They use ViT-tiny or ViT-small as a lightweight encoder and re-train them from scratch, leveraging the SAM-based Masked Image (SAMI) pretraining

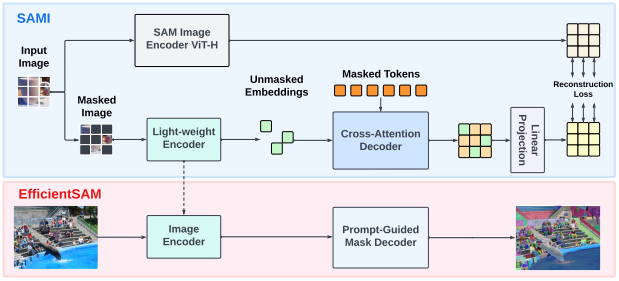


Figure 6. The framework of SAMI and efficientSAM [131]. The target of SAMI is to pretrain a light-weight encoder which holds representation ability close to SAM's image encoder. And the encoder is further extracted to build up the EfficientSAM.

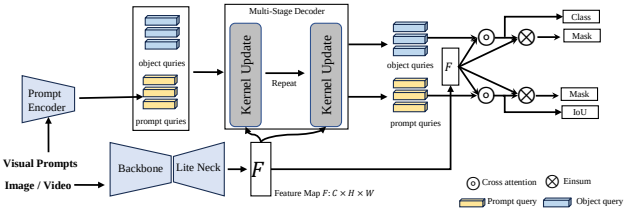


Figure 7. The architecture of RAP-SAM[133]. The encoder embeds images and videos to feature maps and encodes visual prompts as queries (tokens). The multi-stage decoder is to refine queries by interacting with feature maps. In the end, the feature maps and refined queries are fed into adapters to generate masks and classes.

strategy. SAMI is adapted from the Masked AutoEncoder (MAE) framework [44], which was initially used to pretrain SAM's original image encoder. SAMI follows an Encoder-Decoder pipeline: the encoder generates latent feature representations from unmasked patches, while the decoder reconstructs the missing embeddings of masked patches. This process is supervised by reconstruction loss, comparing the embeddings produced by SAM's ViT-H encoder with those generated by the SAMI pipeline. After pretraining, the lightweight encoder is extracted from the SAMI pipeline and integrated with the rest of SAM's components, forming EfficientSAM. The final step involves fine-tuning the entire model on the SA-1B dataset for further alignment and refinement. SAMI is a general pretraining approach that can be applied to train any backbone for SAM variants. The overall structure of SAMI and EfficientSAM is illustrated in Fig. 6.

Xu et al. propose the RAP-SAM [133], a model designed to achieve real-time, all-purpose segmentation, including panoptic segmentation (PS), video instance segmentation (VIS), and interactive segmentation (equivalent to the SegAny task). RAP-SAM retains SAM's basic Encoder-Decoder architecture but incorporates more efficient components to enhance performance. For the encoder, RAP-SAM combines Feature Pyramid Networks (FPN) [66] with deformable convolutions [22] to extract features from both images and videos, while using a prompt encoder to embed visual prompts. In the decoder, RAP-SAM employs a three-stage pipeline utilizing novel pooling-based dynamic convolutions to refine the mask tokens. The tokens generated in each stage, along with the feature maps from the encoder, serve as inputs. These inputs are first processed by dynamic convolutions [17], followed by refinement using Multi-Head Self Attention (MHSA) and a Feed Forward Network (FFN). After the decoder, two additional prompt adapters are introduced to enhance interaction between the visual prompts and segmentation tokens. The final masks are generated by computing the inner product between the updated tokens and updated prompts. The architecture of RAP-SAM is illustrated in Fig. 7.

Recently, Ravi et al. introduced the Segment Anything Model 2 (SAM 2) [93], an extension of the original SAM. The objective of SAM 2 is to deliver high-quality, real-time promptable segmentation across both images and videos. In image segmentation tasks, SAM 2 is reported to achieve higher accuracy and a 6× improvement in efficiency compared to the original SAM. This significant advancement is largely attributed to its efficient image encoder, Hiera [96], a hierarchical ViT that has been simplified from MViTv2 [60] by removing redundant components and utilizing the MAE framework for training. Hiera is a streamlined, purely transformer-based architecture that runs faster and delivers better accuracy than traditional ViTs in both image and video tasks.

3.1.2 Knowledge Distillation based Methods

From the taxonomy shown in Fig. 3, we observe that many methods utilize knowledge distillation, as this approach typically requires less time and fewer resources compared to full model training. In this section, we review SAM variants that adopt efficient backbones for the image encoder while employing knowledge distillation for training. We categorize these models into three groups based on their encoder types: models with (i) lightweight ViT encoders, (ii) pure CNN encoders, and (iii) attention-modified encoders. We will introduce each category in turn.

(i) Lightweight ViT Encoders

Zhang et al. [147] made the first attempt to replace SAM's heavy ViT encoder with the more efficient TinyViT [127], resulting in the integrated model named MobileSAM . As highlighted in [55], training SAM from scratch requires multiple days and 128 GPUs. MobileSAM attributes this complexity to the challenge of optimizing both the encoder and decoder simultaneously. To address this, they propose an encoder-only distillation strategy as illustrated in Fig. 8,

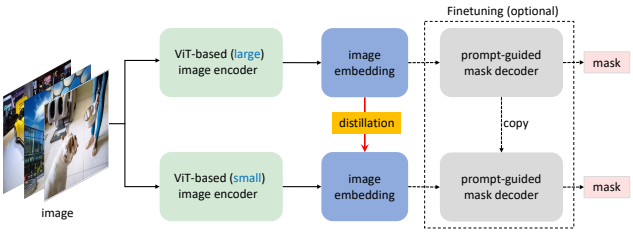


Figure 8. The framework of the encoder-only distillation for MobileSAM [147]. The small ViT-based image encoder, TinyViT [127], is distilled from SAM's ViT-H and the prompt-guided mask decoder is inherited from SAM directly.

which aims to transfer the visual representation capabilities of ViT-H to TinyViT. The loss function used is a simple Mean Square Error (MSE) between the output image embeddings of the two encoders. Further fine-tuning of the prompt encoder or mask decoder is optional and can lead to improved accuracy.

Similar to MobileSAM, the later proposed ESAM [162] utilizes EfficientFormerV2 [62] as its backbone, aiming for improved performance in CPU environments, particularly resource-constrained medical devices. Given that expert models [12, 120, 125] often outperform SAM in medical applications, ESAM incorporates a novel Knowledge Distillation (KD) strategy called Holistic Knowledge Distillation (HKD) to transfer knowledge from an expert model to ESAM. HKD involves two components: distillation on feature maps and distillation on output masks. For the feature map distillation, three different methods with varying focuses [13, 102, 138] are combined to guide the learning process. For the output mask distillation, ESAM uses the Mean Square Error (MSE) loss between the teacher's and student's masks, supplemented by a Binary Cross-Entropy (BCE) loss between the teacher's masks and the ground truth masks. To further align the feature maps between the expert model and ESAM, a Teacher Guided Module (TGM) is introduced as illustrated in Fig. 9.

Shu et al. [103] conducted an analysis of MobileSAM, identifying that encoder-only distillation can lead to significant performance degradation. To address this issue, they propose the Hard Mining Full-Stage Knowledge Distillation strategy for more effective distillation, as illustrated in Fig. 10. With a structure identical to MobileSAM, they developed a new SAM variant called TinySAM by training with this improved KD strategy. Specifically, the strategy supervises not only the image embeddings but also the output tokens and output masks, all using L1 Loss. To further enhance the distillation process, they introduced the Hard Mask Weighting strategy, which assigns larger weights to masks that are harder to predict, thereby improving learning efficiency. The factor H is calculated as follows,

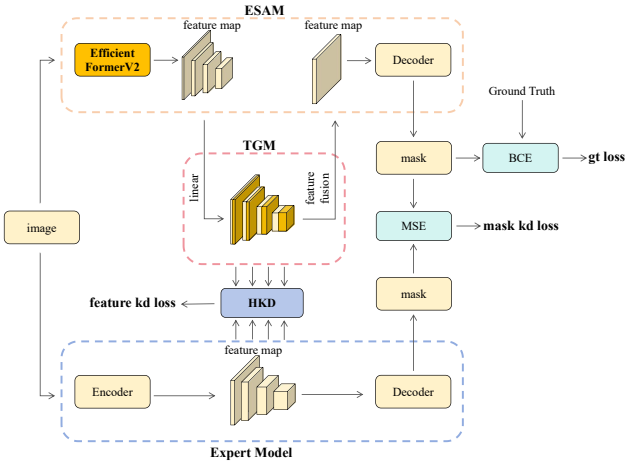


Figure 9. The framework of distilling ESAM [162] from expert model. ESAM adopts the EfficientFormerV2 [62] as image encoder and a Teacher Guided Module (TGM) is applied to align the feature maps for holistic konwledge distillation (HKD).

$$H = sigmoid(\frac{IoU(M^T, M^{GT})}{IoU(M^S, M^{GT}) + \epsilon}) \quad (6)$$

where M^T, M^S, M^{GT} refers to the predicted masks of the teacher, the student and the ground truth mask respectively. An extra strategy named Hard Prompt Sampling is adopted to sample prompts from areas with prediction failure to make the model focus more on image's hard regions.

(ii) CNN-based Encoders

Researchers from NVIDIA introduced a new SAM variant, NanoSAM [84], based on MobileSAM. It is designed to achieve real-time performance on NVIDIA Jetson Orin platforms using NVIDIA TensorRT. NanoSAM replaces the ViT-based encoder with a pure convolutional network, specifically ResNet18 [43], while retaining the other com-

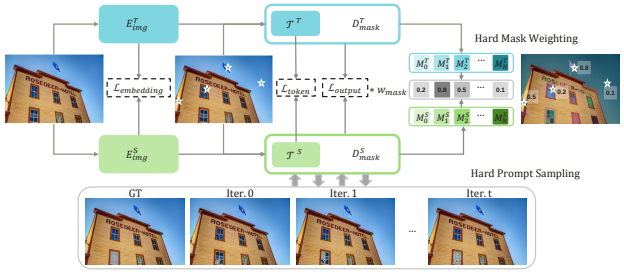


Figure 10. The framework of the hard mining full-stage knowledge distillation for TinySAM [103]. It contains two novel strategies to improve the quality of distillation: the Hard Mask Weighting and the Hard Prompt Sampling.

ponents from MobileSAM. NanoSAM is distilled from MobileSAM, and both models are retrained with TensorRT for optimized performance. The image encoder of MobileSAM is optimized with FP32 precision, whereas NanoSAM's image encoder is built using FP16 precision for faster execution. Inference latency results on Jetson Orin Nano and Jetson AGX Orin demonstrate that NanoSAM is 5x faster than MobileSAM, with minimal accuracy loss.

Wang et al. [117] developed an efficient SAM variant, RepViT-SAM, using their newly proposed CNN-based backbone, RepViT [118], as the image encoder. The core idea behind RepViT is to integrate the effective design principles of efficient Vision Transformers (ViTs) into lightweight CNNs. These design principles are applied at three levels: block-level, macro-level, and micro-level. At the block level, RepViT separates the token mixer and channel mixer [143], reduces the expansion ratio [38], and increases the width of blocks. For the macro design, it incorporates early convolutions [129] as the input stem, deepens the down-sampling layers [73], employs a simpler classifier, and adjusts the block ratios across stages [90]. At the micro level, only 3x3 convolutions are used, and Squeeze-and-Excitation layers [47] are applied only in odd-numbered blocks. RepViT-SAM is trained using knowledge distillation, following the same pipeline as in [147], achieving a 10x increase in inference speed compared to MobileSAM.

Concurrent with the development of RepViT-SAM, Zhou et al. [164] observe that MobileSAM [147] still struggles to achieve real-time performance when deployed on edge devices, such as mobile phones. To address this, they introduced EdgeSAM, which replaces the Transformer-based encoder with the more lightweight and efficient, pure CNN-based RepViT, aiming for better performance on resource-constrained devices. Similar to the approach in [162], Zhou et al. argue that encoder-only distillation is inadequate, as it is task-agnostic and does not fully capture the model's task-specific needs. To overcome this, they propose the Prompt-in-the-Loop Distillation method, which adds additional supervision to the output masks. The "Prompt-in-the-Loop" refers to a dynamic sampling strategy that iteratively samples new prompts from the non-overlapping zones of the teacher's and student's predicted masks. After several iterations, the accumulated loss is backpropagated to update both the encoder and decoder. To further enhance the output quality, EdgeSAM offers an optional module that embeds granularity priors from specific datasets. The overall framework for distilling EdgeSAM is illustrated in Fig. 11.

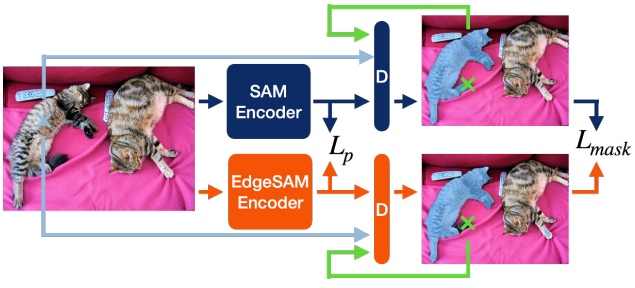


Figure 11. The framework of distilling EdgeSAM [164]. It consists of the encoder-only distillation and the prompt-in-the-loop distillation.

(iii) Attention-modified Encoders

Zhang et al. [159] introduced EfficientViT-SAM, which utilizes EfficientViT [10] as the image encoder. The primary advantage of EfficientViT is its use of the ReLU linear attention mechanism, which facilitates global information interaction while enhancing hardware efficiency. By eliminating the hardware-unfriendly softmax operations and replacing them with ReLU, the attention calculation is reformulated as follows,

$$O_i = \frac{ReLU(Q_i)(\sum_{j=1}^{N} ReLU(K_j)^T V_j)}{ReLU(Q_i)(\sum_{j=1}^{N} ReLU(K_j)^T)} \quad (7)$$

once the sub-expressions $(\sum_{j=1}^{N} ReLU(K_j)^T V_j)$ and $(\sum_{j=1}^{N} ReLU(K_j)^T)$ are computed, they can be reused for each query Q_i. Therefore, the complexity is decreased to $\mathcal{O}(N)$. The training of EfficientViT-SAM is divided into two steps. The first step is distilling from ViT-H to EfficientViT and the second step is train the entire model end-to-end on the SA-1B dataset. This approach achieves nearly no accuracy loss as well as huge improvement in efficiency.

Shen et al. [101] introduced FastSAM3D, an efficient segment-anything model specifically designed for 3D volumetric medical images. The key contribution of this work is the development of the 3D Sparse Flash Attention mechanism. This novel attention approach combines the advantages of the 3D Dilated Attention [26], which extends the receptive field, with FlashAttention [23] to accelerate computations. FastSAM3D utilizes a modified ViT-Tiny as the image encoder, distilled from the ViT-Base encoder, ensuring efficiency without compromising performance. The authors implemented a layer-wise progressive distillation strategy to iteratively align feature maps between the two encoders.

Building on the approach of FastSAM3D, Song et al. [104] introduced a 2D variant called SAM-Lightening. Like its 3D counterpart, SAM-Lightening combines Sparse/Dilated Attention and FlashAttention as a replacement for the standard attention mechanism, while retaining the same image encoder. The key difference lies in the Knowledge Distillation (KD) strategy, referred to as Dynamic Layer-Wise Distillation (DLD). In DLD, a series of time-varying weights $\in [0, 1]$ are applied to determine which layers need to be updated and how much each layer contributes to the

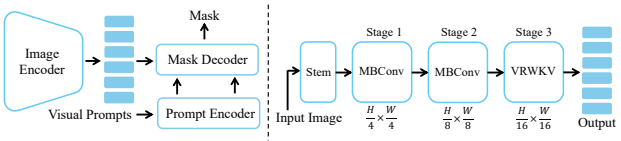


Figure 12. The overview of RWKV-SAM [145] (left) and its image encoder (right). The image encoder is made up of three stages, where the first two stages use Mobile Convolution Blocks [97] and the last stage uses Vision RWKV blocks [30].

update process. As training progresses, the entire architecture is gradually optimized. SAM-Lightening reportedly achieves a 30× acceleration compared to SAM-H. The overall distillation framework is similar to FastSAM3D, with the main change being the substitution of the distillation strategy for dynamic layer-wise distillation.

A recent work by Yuan et al. [145], RWKV-SAM, represents a significant advancement in accelerating SAM by introducing the popular linear attention model [39, 87] as an efficient backbone. In their study, they compare RWKV-based and Mamba-based architectures and select the RWKV-based approach to construct a lightweight version of SAM. The backbone is a hybrid design, with the first two stages consisting of Mobile Convolution Blocks from [97], and the final stage built using Vision RWKV blocks [30]. More details on RWKV can be found in Section 2.2.2. Additionally, a refinement module is incorporated into the SAM-like architecture to enhance mask quality by fusing features from different levels generated at each stage. The overall architecture of RWKV-SAM is depicted in Fig. 12. The model is trained using a "distillation-finetune" strategy, where knowledge from SAM-H is first distilled into the backbone, followed by fine-tuning of the entire model. RWKV-SAM demonstrates significant improvements in efficiency, while maintaining comparable segmentation performance to SAM.

3.1.3 Quantization based Methods

As detailed in previous section, TinySAM [103] leverages the Hard Mining Full-Stage Distillation to improve the effectiveness of transferring knowledge. To further contract the scale of TinySAM, Shu et al. adopt a post-training quantization to the encoder of TinySAM and the quantified version is named as Q-TinySAM. They take the quantization basically following the instructions in [146]. For the matrix multiplication in ViT, $O = AB$ would be quantified to $\hat{O} = \hat{A}\hat{B}$, with scaling factors s_A, s_B. The researchers takes a alternative and iterative search strategy for the best s_A, s_B which can minimize the distance between O and $\hat{O}$. The hessian guided metric is used to measure the distance

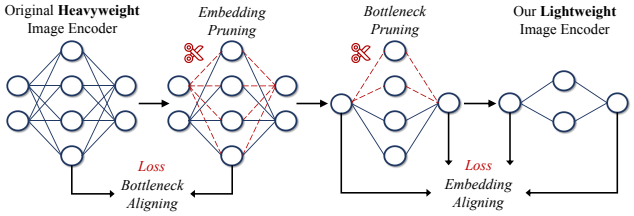


Figure 13. The overall pipeline of the alternate slimming strategy for SlimSAM [18]. It takes two steps to contract the heavy image encoder of SAM into a light-weight one: 1) Embedding pruning and bottleneck aligning; 2) Bottleneck pruning and embedding aligning.

which is approximately calculated as follows:

$$\min_{\Delta} \mathbb{E}[M^T diag((\frac{\partial L}{\partial O_1^l})^2, \ldots, (\frac{\partial L}{\partial O_{|O^l|}^l})^2) M] \quad (8)$$

where $M = (\hat{O}^l - O^l)$ and the Kullback-Leible (KL) divergence of masks and IoUs are used for task loss.

Different from taking quantization in an encoder-only manner used for Q-TinySAM, Lv et al. [77] propose a novel framework that can directly take post-training quantization on SAM, namely PTQ4SAM. They start the work by first discovering the two challenges after traditional PTQ: 1) the existence of bimodal distribution which has negative effects on quantization quality and 2) the obvious discrepancy in the distribution of different attention mechanisms. Therefore, researchers bring up two strategies to solve them respectively: the Bimodal Integration (BIG) and the Adaptive Granularity Quantization (AGQ). For the Bimodal Integration, a sign factor γ is introduced to convert the bimodal distribution into a normal distribution. For the Adaptive Granularity Quantization, the key is to adopt an adaptive parameter τ to adjust the base in the Log2 quantizer [81]. With the use of the AGQ strategy, the discrepancy between post-softmax distributions of different attention is expected to narrow down effectively. The PTQ4SAM is a plug-and-play SAM variant that can be easily deployed to downstream tasks.

3.1.4 Pruning based Methods

Chen et al. [18] were the first to develop an effective pruning strategy for reducing the size and complexity of SAM, resulting in the model known as SlimSAM. As discussed in Section 2.3.3, pruning algorithms aim to remove redundant parameters either structurally or individually. When applied to SAM's heavy encoder, the initial step involves estimating the importance of weights and activation values to determine which should be pruned. The core idea behind assessing importance is to evaluate the discrepancy in loss generated with and without the given parameter. SlimSAM

introduces the Disturbed Taylor Importance method, which uses the first-order Taylor expansion to approximate the importance of parameters and introduces Gaussian Noise $\mathcal{N}$ to prevent gradients from becoming zero. This process is formulated as follows,

$$\begin{aligned} I_{w_i} &\approx |\mathcal{L}_{w_i}(x_i, t_i + \mathcal{N}) - \mathcal{L}_{w_i=0}(x_i, t_i + \mathcal{N})| \\ &\approx \left| \frac{\partial \mathcal{L}(x_i, t_i + \mathcal{N})}{\partial w_i} w_i \right| \end{aligned} \tag{9}$$

Once the importance of parameters is estimated, a strategy called Alternate Slimming is employed to perform structural pruning and post-alignment. The ViT-based encoder is first divided into two substructures: the embedding layer and the bottleneck layer. The strategy alternates between pruning the embedding/bottleneck layers to reduce model size and aligning the bottleneck/embedding layers to maintain model quality, ensuring both efficiency and performance. The workflow for this process is illustrated in Fig. 13.

3.1.5 Code Refactorization

The Segment Anything Fast model (SAMfast), developed by the PyTorch team [88], is a rewritten version of SAM that leverages pure, native PyTorch optimizations. SAMfast is reported to be 8x faster than the original implementation while maintaining nearly the same accuracy. This improvement is the result of a systematic process of identifying bottlenecks and applying targeted optimizations. Initially, the team identified long function calls that caused synchronous blocking, leading them to rewrite the corresponding code. Another significant bottleneck was the time-consuming matrix multiplication, which was mitigated by using bfloat16 precision. Following these adjustments, the team utilized torch.compile to fuse smaller operations and adopted PyTorch's scaled_dot_product_attention (SDPA) to accelerate attention computations on the GPU. Additionally, by integrating the new kernel built with Triton, memory usage on the GPU was further reduced. When SAM uses the batch prediction method, input tensors of varying sizes are unified into NestedTensors, which significantly improve throughput. Despite these optimizations, matrix multiplications remained a key bottleneck. To address this, the team implemented int8 quantization and approximated matrix multiplication using semi-structured sparsity. For more details on the step-by-step optimization process, it is recommended to go through the official blog for more details.

3.2. Accelerating SegEvery Tasks

As described in Section 3.1, the primary efficiency bottleneck for the SegAny task lies in the heavy image encoder. Any SAM variant with a more lightweight architecture will inherently be able to segment everything faster than the original SAM. However, as analyzed by Zhang et al. [148], the main challenge of the SegEvery task stems from the dense grid sampling strategy. This strategy first predicts numerous masks based on a point grid and then selects valid masks, which is computationally expensive. Consequently, designing a more efficient sampling strategy to reduce the number of predicted masks has become the core approach to accelerating the SegEvery task. Another potential solution is to convert the SegEvery task into another well-established task, such as all-instance segmentation, as done in FastSAM [161]. In this part, we will review works that have specifically proposed optimized sampling strategies to accelerate the SegEvery task.

Building on the structure of SAM, Zhang et al. [148] introduced an object-aware prompt sampling strategy to enhance the efficiency of the SegEvery task. This project, named MobileSAMv2, operates independently of their previous work [147] focused on accelerating the SegAny task. In MobileSAMv2, the researchers employ the YOLOv8 model, trained on a small subset of SA-1B, for object discovery. This model generates a large number of bounding boxes corresponding to latent objects. Highly overlapping boxes are filtered using Non-Maximum Suppression (NMS), and the remaining boxes are used as box prompts. By using these filtered boxes as prompts, MobileSAMv2 eliminates the need to filter predicted masks—a much more time-consuming process. With the maximum number of prompts set to 320, the new strategy is reported to be 16 times faster than the traditional 32*32 grid sampling strategy. Additionally, MobileSAMv2 can be integrated with MobileSAM to create a unified model that achieves high efficiency in both SegAny and SegEvery tasks.

Shu et al. [103] observed that using a dense points grid (e.g., 32*32, 64*64) often generates a large number of redundant masks, which are later filtered out during post-processing, an operation that incurs significant time costs. In reality, only a few points within the grid are necessary to produce confident masks. To address this inefficiency, they proposed a hierarchical strategy for efficient sampling, which progressively selects optimal points for mask generation. This strategy involves two rounds of prompt generation. In the first round, a sparse grid is used, incorporating only a fraction (approximately 1/4) of the default points along each side. Masks are generated based on these points, and after filtering, only high-confidence masks are retained as final predictions. In the second round, a denser grid is applied, following the default configuration. However, points located in regions already covered by confident masks are excluded, significantly reducing the number of points. The predicted results from both rounds are then fused to produce the final output. The pipeline for this hierarchical strategy is illustrated in Fig. 14. By employing this two-round

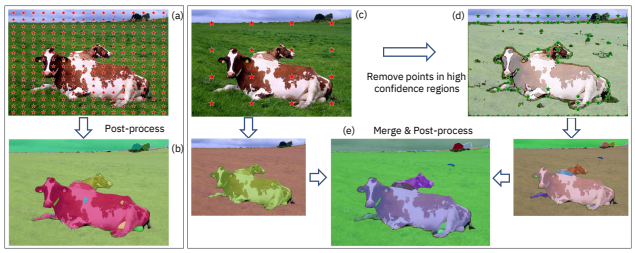


Figure 14. Comparison between the default sampling strategy (a-b) for SAM and the hierarchical sampling strategy (c-e) for TinySAM [103]. The hierarchical strategy contains two rounds of sampling: one round for sparse sampling (c) and the other round for dense sampling (d). The final prediction comes from the fusion of results in two rounds.

approach, the sampling process becomes both more time-efficient and fine-grained, leading to a notable acceleration in the SegEvery task with minimal performance degradation.

Different from all aforementioned works, Fu et al. [33] propose an end-to-end training pipeline specifically designed for the SegEvery task, aiming to develop a SAM variant capable of segmenting everything more efficiently. Their model, named Lite-SAM, retains the overall architecture of the original SAM but replaces the heavy image encoder with a more lightweight solution. The architecture overview of Lite-SAM is shown in Fig. 15. Lite-SAM employs a CNN-Transformer hybrid structure called Lite-ViT, which consists of four stages with 2, 2, 6, and 2 Lite-ViT blocks, respectively. The key innovation in Lite-ViT is the Multi-Scale Pooling Module (MSPM), which serves as an alternative to the traditional attention mechanism. Adapted from the PoolFormer block [142], MSPM utilizes channel-wise LayerNorm and extends the pooling operation to multiple scales. As discussed earlier, another major bottleneck in SAM lies in the time-consuming grid sampling strategy. To address this, Lite-SAM introduces an Automated Prompt Proposal Network (AutoPPN) to improve sampling efficiency. AutoPPN takes the feature maps generated by the encoder as input and directly predicts point and box prompts. To ensure high-quality prompts, Lite-SAM uses a more powerful MSPM-based network instead of CNNs, and incorporates distance transform to estimate the confidence of point prompts. While Lite-SAM is primarily designed to accelerate the SegEvery task, it also demonstrates improved efficiency in the SegAny task due to its lightweight image encoder.

3.3. Future Research Directions

Through this comprehensive review of efficient SAM variants, we have outlined the current advancements in accelerating SAM. However, there remain opportunities for further exploration and innovation. Below, we present several potential future research directions, offering preliminary insights with the hope of inspiring readers to contribute to the ongoing development in this field.

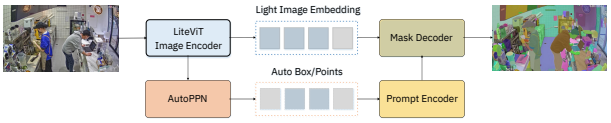


Figure 15. The architecture of Lite-SAM [33]. It takes the Lite-ViT as an image encoder and proposes an automated prompt proposal network (AutoPPN) for efficient sampling, while the prompt encoder and the mask decoder are copied from the original SAM.

Exploring Advanced Architectures. While current SAM variants have demonstrated efficiency gains through the adoption of efficient architectures and model compression techniques, there is substantial potential for further improvement. The exploration of Transformer-alternative models, such as Mamba [39], RetNet [105], KAN [74], and TTT [106], presents an exciting opportunity to design more lightweight and efficient structures. These models may offer advantages in computational efficiency without sacrificing segmentation accuracy. In addition to alternative models, refining the efficiency of the attention mechanism across both the image encoder and mask decoder remains critical. Methods such as linear attention, low-rank factorization, or hybrid architectures combining convolutional and attention-based designs should be further investigated. Addressing bottlenecks in both computation and memory usage will enhance SAM's deployment across diverse hardware environments.

Exploiting Sparsity and Acceleration Techniques. The inherent sparsity observed in deep neural networks, where only a subset of parameters significantly contributes to model output, offers a promising avenue for improving SAM's efficiency. Techniques like pruning, quantization, and structured sparsity could further reduce SAM's computational demands. While initial efforts in sparsification, such as those in [18], have shown success, future research could focus on understanding the distribution and dynamics of sparsity in SAM's architecture. This includes investigating the optimal layers or components of SAM that can be pruned or sparsified without impacting performance. Additionally, techniques such as sparse attention mechanisms, dynamic pruning during inference, and low-precision training should be explored to balance accuracy and efficiency, especially for large-scale deployments. By combining these with advanced knowledge distillation techniques, more compact, efficient variants of SAM could be realized.

Hardware-Specific Optimizations. Optimizing SAM for

specific hardware platforms, including GPUs, TPUs, specialized AI accelerators (e.g., NVIDIA's TensorRT or Google's Edge TPU), and edge devices, can unlock significant gains in performance and efficiency. Hardware-aware model optimization techniques, such as operator fusion, quantization-aware training, and custom CUDA kernels, can maximize throughput and reduce latency when deploying SAM on modern hardware platforms. In the context of edge devices, which face extreme constraints in storage, computational power, and energy supply, these optimizations are crucial for real-time applications, such as segmentation on UAVs or IoT devices. Future research could explore hierarchical cloud-edge architectures to offload computationally expensive tasks to the cloud, while running lightweight models locally on edge devices. Additionally, leveraging specialized AI hardware like Field-Programmable Gate Arrays (FPGAs) or using techniques such as hardware-aware neural architecture search (NAS) and mixed-precision quantization could further optimize SAM for low-latency, resource-constrained environments, ensuring that the model operates effectively across diverse hardware platforms.

Efficient Segmentation for Video and Multi-Modality Data. Video and multi-modality tasks, which deal with complex, dynamic environments, are rapidly gaining relevance across numerous real-world applications. While some initial efforts, such as SAM 2 [93] for video segmentation and MM-SAM [128] for multi-modality tasks, have extended SAM's applicability, efficiency remains a pressing issue. Video data contains temporal redundancy, while multimodal data often exhibits correlations between modalities. Leveraging these inherent redundancies through techniques like temporal aggregation and cross-modal feature sharing could significantly reduce computational costs. Future work could focus on optimizing SAM's runtime complexity by utilizing spatiotemporal attention, efficient memory mechanisms for temporal data, and early-fusion techniques to reduce the number of modality-specific computations. Developing frameworks that adapt dynamically to different levels of redundancy across frames or modalities can further drive computational efficiency in real-world applications.

4. Evaluation

In this section, we systematically compare the efficiency and accuracy of the previously described SAM variants. With reference to the experiments conducted in these works, we choose the tasks that most of the works have carried out and evaluate them on their commonly used datasets with corresponding metrics. Our evaluations are conducted on a single *24GB RTX 3090 GPU* with a 14 vCPU *Intel(R) Xeon(R) Gold 6330 Processor @ 2.00GHz*. More details are provided in the following subsections: Section 4.1 introduces the datasets and metrics used for evaluation; Sections 4.2 and 4.3 report the quantitative results of efficiency and accuracy, respectively.

4.1. Datasets and Metrics

We select COCO 2017 [65] and LVIS v1 [40] as the evaluation datasets. COCO is a large-scale dataset designed for object detection, segmentation, and captioning, containing 330K images and 1.5 million object instances across 80 object categories. LVIS is tailored for large vocabulary instance segmentation, featuring over 2 million high-quality segmentation masks across more than 1200 categories in 164K images. For our evaluation, we use the validation sets of both datasets, which include 36,781 instances in 5,000 images for COCO and 244,707 instances in 19,809 images for LVIS. To evaluate efficiency, we first test several soft indicators like $\#Params$, $FLOPs$, $MACs$, and $Memory\,Usage$. And we further calculate the Efficient Error Rate (EER) a more comprehensive evaluation, as outlined in [86]. EER is defined as,

$$EER = \frac{1}{N}\sum_{i=1}^{N}(\frac{M_i}{R_i}) \tag{10}$$

where N is the number of metrics and M_i, R_i refer to the i-th metric of the tested model and the reference one, respectively. In our evaluation, we set the reference model to be SAM-H. In addition to these metrics, we also report the runtime and throughput of the models. For accuracy evaluation, we use mean intersection over union ($mIoU$) for the SegAny task and mean average precision (AP) for instance segmentation.

4.2. Efficiency Comparison

We first report the efficiency results of SAM and its variants. With the picture *groceries.jpg*[1] used in SAM's official example as input, we utilize a bounding box as prompt and evaluate models' $\#Params$, $FLOPs$ and $MACs$ by leveraging the tool *calflops*[2]. We also calculate the ERR for comprehensive comparison. The results are illustrated in Tab. 2. Among the efficient variants, we observe that EdgeSAM has the lowest number of parameters, $FLOPs$, $MACs$ and consequently the EER, while EfficientViT-SAM-XL1 takes the highest numbers whose EER is 3% more than SAM-B. When compared to the heaviest SAM-H, all variants present obvious reduction in both model size and calculation.

We also measure the models' inference time in both SegAny and SegEvery modes, using 100 images from the COCO validation set as evaluation data. For the SegAny task, each image is prompted with 50 fixed bounding boxes.

[1] https://github.com/facebookresearch/segment-anything/blob/main/notebooks/images/groceries.jpg

[2] https://github.com/MrYxJ/calculate-flops.pytorch

Model	#Params	GFLOPs	GMACs	EER (%)
SAM-H	641.09M	5490	2730	100
SAM-L	312.34M	2640	1310	48.26
SAM-B	93.74M	746.4	370.47	13.93
SAM2-B+	80.83M	533.87	266.44	10.70
FastSAM	68M	887.6	-	13.39
MobileSAM	10.13M	76.39	37.49	1.45
EdgeSAM	**9.58M**	**44.02**	**21.81**	**1.03**
EfficientSAM-Ti	10.22M	53.64	26.74	1.18
EfficientSAM-S	26.41M	371.68	185.56	5.90
RepVit-SAM	27.22M	231.65	115.07	4.23
EfficientViT-SAM-L0	34.79M	154.24	76.89	3.68
EfficientViT-SAM-XL1	203.35M	523.07	261.21	16.94
SlimSAM-50	28.01M	196.21	96.73	3.83
SlimSAM-77	9.85M	47.66	23.19	1.08
TinySAM	10.13M	76.39	37.94	1.45

Table 2. Quantitative results of **#Params**, **GFLOPs**, **GMACs** and **EER**. P.S. The results of FastSAM [161] are collected from [164] and others are evaluated under our framework.

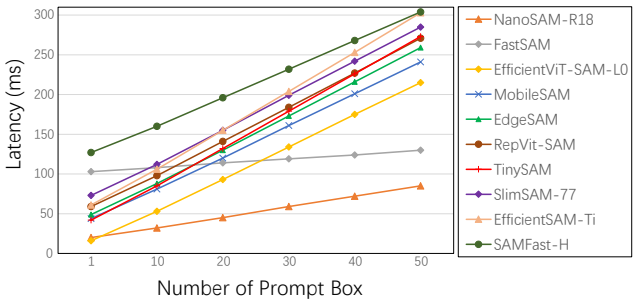


Figure 16. Latency results on GPU with increasing numbers of box prompts.

We report the cumulative time for every 10 boxes through a curve chart (shown in Fig. 16). Based on this, we calculate the average time required to process one image with one box prompt and report it as the inference time for the SegAny task. The evaluation is conducted on both CPU and GPU environments, and we simultaneously record GPU memory usage. Additionally, we test the throughput of each variant on the COCO validation set, using the ground truth boxes. The results are summarized in Tab. 3. Our findings reveal that EfficientViT-SAM-L0 achieves the shortest inference time for the SegAny task, offering nearly 30x acceleration on GPU and almost 50x acceleration on CPU compared to the heaviest model, SAM-H. EdgeSAM also shows impressive performance with a low CPU latency of 259 ms, while NanoSAM achieves a 20 ms latency on GPU, both coming close to the best results. In the throughput test on the COCO dataset, NanoSAM leads with 27.9 images processed per second. Two other variants, EfficientSAM-Ti and EfficientViT-SAM-L0, also demonstrate strong throughput, each exceeding 20 images per second.

For the SegEvery task, we report the mean time required to generate all masks for an image using different point grid sizes (16*16, 32*32, 64*64) or specialized sampling strategies. The results are presented in Tab. 4. With the default 32*32 grid, SAMfast-H demonstrates the highest efficiency, with a latency of 848 ms—more than twice as fast as SAM-H. EfficientViT-SAM-L0 performs best with the 16*16 and 64*64 grids, achieving latencies of 258 ms and 3938 ms, respectively. Interestingly, we observe that EfficientSAM-S is slower than SAM-H when using lower grid densities, showing latencies of 1100 ms for the 16x16 grid and 2290 ms for the 32*32 grid. Models employing alternative sampling strategies demonstrate significant improvements in efficiency, particularly FastSAM, which records a latency of 196 ms, and MobileSAMv2, with a latency of 173 ms.

4.3. Accuracy Comparison

In this subsection, we report the accuracy results of SAM and its variants on SegAny task (with point/box prompts) and instance segmentation task, respectively. We follow the evaluation framework in [159] and adapt it by introducing other evaluation modules to conduct a unified evaluation on variants.

To evaluate on the SegAny task, we adopt two types of points as prompts: 1) the center point of the ground truth bounding box, and 2) random points uniformly sampled from ground truth masks, following the setting in [131]. We evaluate the variants on COCO and LVIS, and the mean intersection over union (mIoU) is reported in Tab. 5. When prompted with the center point, SAM2-B+ and EfficientViT-SAM-XL1 achieve the highest mIoU of 54.3% on COCO, surpassing SAM-H with 53.6% mIoU,

Model	Memory (GB)	Latency-CPU (ms)	Latency-GPU (ms)	Throughput (img/s)
SAM-H	7.46	9470	472	2.04
SAM-L	6.03	6800	289	3.33
SAM-B	4.39	3294	129	6.78
SAMfast-H	2.13	-	127	6.36
SAM2-B+	1.38	1221	46	13.82
FastSAM	4.96	850	103	8.42
MobileSAM	1.81	424	44	16.79
EdgeSAM	1.72	259	49	17.56
EfficientSAM-Ti	4.46	1159	61	20.04
EfficientSAM-S	7.53	2605	86	19.7
RepVit-SAM	1.87	1013	59	13.49
EfficientViT-SAM-L0	1.73	**194**	**16**	20.92
EfficientViT-SAM-XL1	2.57	1334	67	12.28
SlimSAM-50	4.49	2312	87	9.13
SlimSAM-77	4.34	2230	73	10.28
TinySAM	2.2	422	42	16.53
Q-TinySAM	2.34	697	73	7.03
NanoSAM	**0.94**	-	20	**27.9**

Table 3. Quantitative results of **inference latency** of SegAny task and **throughput** on COCO dataset. P.S. 1) We also report the GPU memory usage in first column when models are inferring on GPU; 2) SAMfast [88] and NanoSAM [84] can not run on CPU environment.

Model	16*16 grid (ms)	32*32 grid (ms)	64*64 grid (ms)	other prompt strategies (ms)
SAM-H	824	2269	8103	-
SAM-L	665	2053	7518	-
SAM-B	422	1399	5417	-
SAMfast-H	315	**848**	OOM	-
SAM2-B+	442	1610	6356	-
FastSAM	-	-	-	**106**
MobileSAM	347	1313	5022	
MobileSAMv2	-	-	-	173
EdgeSAM	404	1553	5877	-
EfficientSAM-Ti	753	1833	5670	-
EfficientSAM-S	1100	2290	6210	-
RepVit-SAM	512	1890	7437	-
EfficientViT-SAM-L0	**258**	1023	**3938**	-
EfficientViT-SAM-XL1	432	1546	6094	-
SlimSAM-50	400	1374	5213	-
SlimSAM-77	385	1342	5148	-
TinySAM	-	-	-	776
Q-TinySAM	-	-	-	1318

Table 4. Quantitative results of **inference latency** of SegEvery task with grid sampling in different scales or other efficient strategies. P.S. OOM denotes out-of-memory.

while SAMfast-H, also with 53.6%, exhibits the best performance among variants on LVIS. Under the setting of random point prompts, it is demonstrated that EfficientViT-SAM-XL1 performs better than SAM-H, particularly when prompted with 3 points, with an increase of 2.7% and 0.7%, respectively. From the perspective of datasets, we observe that the results on LVIS are generally lower than those on COCO, especially for FastSAM and EfficientSAM-Ti, whose accuracy decreases to below 30% on LVIS.

Moreover, we also evaluate the accuracy on SegAny task with two types of box prompts: 1) The ground truth bounding box and 2) The tightest bounding box corresponding to the ground truth mask, inspired by the experiments in [131, 159]. We reported the results of mIoU on COCO and LVIS which are illustrated in Tab. 6. We observe that EfficientViT-SAM-XL1 demonstrate the highest accu-

Model	COCO			LVIS		
	pt_1	1*pt_2	3*pt_2	pt_1	1*pt_2	3*pt_2
SAM-H	53.6	55.6	67.5	60	59.8	65.0
SAMfast-H	52.9	54.5	67.4	**53.6**	**57.2**	60.6
SAM2-B+	**54.3**	54.9	68.5	53.1	55.9	62.6
FastSAM	46.7	47.3	48.7	28.4	29.8	29.5
MobileSAM	48.6	49.2	59.1	45.6	47.2	50.0
EdgeSAM	47.2	46.8	61.6	45.2	46.5	54.4
EfficientSAM-Ti	43.8	44.6	61.6	25.4	26.9	52.5
EfficientSAM-S	41.1	39.4	60.1	52.3	53.1	**66.3**
RepViT-SAM	50.6	51.8	63.1	51.0	51.6	54.6
EfficientViT-SAM-L0	50.1	50.7	67.2	46.4	48.3	63.9
EfficientViT-SAM-XL1	**54.3**	**55.7**	**70.2**	53.5	54.4	65.7
SlimSAM-50	48.5	48.5	63.5	51.7	53.4	58.6
SlimSAM-77	47.2	47.2	61.1	48.6	50.4	55.3
TinySAM	42.4	42.1	65.1	50.0	51.0	60.9
NanoSAM	44.2	42.4	56.3	42.7	41.4	48.3

Table 5. The **mIoU** results on COCO and LVIS with points as prompts. P.S. pt_1 denotes the center point of the ground truth bounding box and pt_2 denotes points randomly sampled from ground truth masks.

Model	COCO		LVIS	
	box_1	box_2	box_1	box_2
SAM-H	77.4	77.8	78.0	79.1
SAMfast-H	77.3	77.7	77.0	78.0
SAM2-B+	75.7	76.6	70.4	72.4
FastSAM	60.5	60.6	50.4	50.6
MobileSAM	73.4	74.0	74.4	74.8
EdgeSAM	75.9	76.2	74.4	74.8
EfficientSAM-Ti	72.6	73.3	72.9	74.5
EfficientSAM-S	69.1	69.0	74.7	75.5
RepViT-SAM	75.1	75.7	73.6	74.8
EfficientViT-SAM-L0	78.5	78.2	78.0	77.9
EfficientViT-SAM-XL1	**79.9**	**79.8**	**79.9**	**79.7**
SlimSAM-50	75.5	76.3	74.3	76.0
SlimSAM-77	74.4	75.2	72.7	74.5
TinySAM	73.7	73.9	73.6	74.1
NanoSAM	69.7	70.2	65.1	66.5

Table 6. The **mIoU** results on COCO and LVIS with boxes as prompts. P.S. box_1 denotes the ground truth bounding box and pt_2 denotes the tightest bounding box with respect to the ground truth mask.

racy in every setting which exceeds SAM-H by 1.5%, 1.1%, 1.9% and 0.6%, respectively. SAMfast-H and EfficientViT-SAM-L0 also present performance close to SAM-H in the box-prompted segmentation tasks.

For instance segmentation tasks, we adopt the ViTDet [59], YOLOv8 [53], GrounddingDINO [69], Detic [165] and H-Deformable-DETR [52] with Swin-L [71] as the object detectors which helps to generate bounding boxes of latent objects, with reference to [117, 159, 164]. We evaluate the average precision (AP) of all objects, as well as AP of small, medium, and large objects. The results are reported in Tab. 7, 8 and 9. Similar to the previous results, we find that on COCO dataset EfficientViT-SAM-XL1 always presents the highest AP with any of the detectors (except H-Deformable-DETR). Under the setting of equipping ViTDet as detector and testing on LVIS dataset, SAMfast-H surpass all other variants with AP of 44.5%.

Based on the results in Section4.2 and Section 4.3, we further make a throughput-mIoU scatter to observe variants' efficiency-accuracy trade-off. Specifically, we select the throughput and mIoU that are evaluated on COCO dataset with ground truth bounding boxes as prompts. The results are illustrated in Fig. 17.

Model	COCO				LVIS			
	AP	AP^S	AP^M	AP^L	AP	AP^S	AP^M	AP^L
SAM-H	46.5	30.8	51.0	61.7	44.7	32.5	57.6	65.5
SAMfast-H	46.4	30.4	50.8	61.9	**44.5**	**32.3**	**57.5**	65.7
SAM2-B+	44.8	27.1	49.9	62.5	42.3	30.1	57.0	65.3
FastSAM	37.9	23.9	43.4	50.0	34.5	24.6	46.2	50.8
MobileSAM	38.7	23.7	42.2	54.3	37.0	24.7	47.8	59.1
EdgeSAM	42.1	26.6	46.7	56.9	39.8	28.6	51.3	59.3
EfficientSAM-Ti	42.3	26.7	46.2	57.4	39.9	28.9	51.0	59.9
EfficientSAM-S	35.6	27.6	41.4	42.6	31.7	29.5	43.1	44.3
RepViT-SAM	43.3	27.	47.4	59.6	40.2	28.0	52.3	61.4
SlimSAM-50	42.8	27.2	46.8	58.4	40.1	28.5	51.8	60.5
SlimSAM-77	41.3	25.7	44.9	57.5	38.3	26.8	49.7	59
TinySAM	42.3	26.3	45.8	58.8	38.9	27.0	50.3	60.2
Q-TinySAM	41.4	25.6	45.1	57.9	38.5	26.6	49.8	59.8
EfficientViT-SAM-L0	45.7	28.2	49.5	63.4	41.8	28.8	53.4	64.7
EfficientViT-SAM-XL1	**47.8**	**30.5**	**51.8**	**64.7**	44.4	31.6	57.0	**66.4**
NanoSAM	35.9	20.0	40.3	52.4	31.6	19.5	42.7	53.7

Table 7. Quantitative results of instance segmentation on COCO and LVIS with ViTDet [59] as object detector.

Model	YOLOv8				GroundingDINO			
	AP	AP^S	AP^M	AP^L	AP	AP^S	AP^M	AP^L
SAM-H	43.8	26.1	48.1	60.4	46.9	31.5	51.8	64.4
SAMfast-H	43.6	**26.0**	47.9	60.2	46.8	**31.5**	51.6	64.2
SAM2-B+	42.4	24.1	47.4	60.5	45.1	28.2	50.8	64.6
FastSAM	34.2	19.7	40.1	46.6	35.7	22.1	42.1	48.2
MobileSAM	39.0	21.7	42.8	56.5	41.4	25.7	45.6	60.5
EdgeSAM	39.5	22.6	43.7	56.0	42.5	27.4	46.9	59.5
EfficientSAM-Ti	36.6	23.0	41.3	49.4	39.6	27.2	44.7	53.4
EfficientSAM-S	32.9	24.1	38.6	40.7	36.5	29.4	42.3	44.9
RepViT-SAM	41.0	23.8	45.1	57.9	43.8	28.3	48.2	61.6
SlimSAM-50	40.6	23.9	44.7	57.0	43.5	28.7	47.8	60.8
SlimSAM-77	39.3	22.5	43.1	55.9	42.1	27.1	46.0	59.6
TinySAM	39.7	22.8	43.5	56.7	42.4	27.4	46.5	60.6
EfficientViT-SAM-L0	42.7	24.1	46.8	61.2	46.0	29.2	50.3	65.7
EfficientViT-SAM-XL1	**44.7**	**26.0**	**48.9**	**62.9**	**48.2**	**31.5**	**52.6**	**67.3**
NanoSAM	34.3	18.0	38.4	50.9	36.3	21.2	41.2	54.5

Table 8. Quantitative results of instance segmentation on COCO with YOLOv8 [53] or GrounddingDINO [69] as object detector.

5. Conclusion

In this survey, we have primarily discussed and evaluated the prominent works that focus on methods to efficiently segment anything and segment everything with reduced resource consumption and lower latency. For efficient SegAny tasks, most works adopt the approach of replacing either the image encoder or the entire architecture with a lightweight alternative, followed by training from scratch or through knowledge distillation. Other works aim to compress the original model by leveraging techniques such as quantization, pruning, or local optimization. For efficient SegEvery tasks, adopting an effective and efficient sampling strategy for prompt generation is essential. After reviewing these methods in detail, we have also outlined four potential future research directions that may drive new trends in this area. Additionally, we have evaluated the efficiency, accuracy, and the corresponding trade-offs of these models in a consistent environment, providing a fair and valuable comparison. Our analysis shows that some variants have already outperformed the original SAM in specific scenarios, and we believe their success will inspire further exploration and innovation in this field.

Model	Detic				H-Deformable-DETR			
	AP	AP^S	AP^M	AP^L	AP	AP^S	AP^M	AP^L
SAM-H	39.2	26.7	44.4	51.3	46.8	31.8	51.0	63.6
FastSAM	31.1	19.4	38.6	40.9	35.3	21.6	41.2	48.0
MobileSAM	34.9	22.7	39.8	47.6	42.7	27.0	46.5	61.1
EdgeSAM	35.2	23.5	40.3	46.6	32.1	21.6	44.3	59.4
EfficientSAM-Ti	32.7	22.9	39.2	41.7	38.8	26.5	43.6	53.0
EfficientSAM-S	30.4	25.8	38.4	33.7	35.8	27.8	41.5	44.4
RepViT-SAM	36.5	24.7	41.7	48.2	**44.4**	**29.1**	**48.6**	**61.4**
SlimSAM-50	34.3	24.9	40.5	43.1	42.2	29.3	46.8	56.5
SlimSAM-77	34.9	22.7	39.6	47.0	26.8	21.6	43.9	59.0
TinySAM	35.2	23.2	39.9	47.5	43.1	27.4	46.6	61.2
EfficientViT-SAM-L0	37.9	24.1	42.7	51.6	-	-	-	-
EfficientViT-SAM-XL1	**39.5**	**26.3**	**44.6**	**52.2**	-	-	-	-
NanoSAM	25.1	13.2	31.9	37.6	-	-	-	-

Table 9. Quantitative results of instance segmentation on COCO with Detic [165] or H-Deformable-DETR [52] as the object detector.

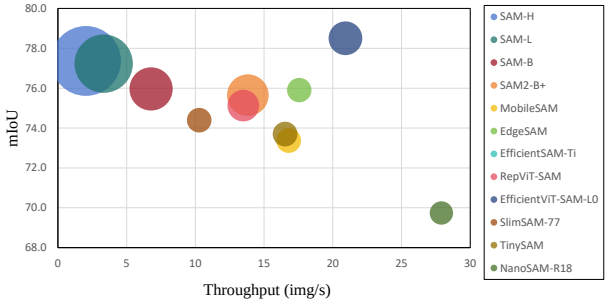


Figure 17. The trade-off between throughput and mIoU of SAM and its variants. The scale of circles responses to models' size.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[2] Mohsen Ahmadi, Ahmad Gholizadeh Lonbar, Abbas Sharifi, Ali Tarlani Beris, Mohammadsadegh Nouri, and Amir Sharifzadeh Javidi. Application of segment anything model for civil infrastructure defect assessment. *arXiv preprint arXiv:2304.12600*, 2023. 1, 3

[3] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023. 1

[4] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18, 2017. 5

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 7

[6] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems*, 2019. 5

[7] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2019. 7

[8] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 1

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020. 1

[10] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Multi-scale linear attention for high-resolution dense prediction. *arXiv preprint arXiv:2205.14756*, 2022. 4, 10

[11] Yunkang Cao, Xiaohao Xu, Chen Sun, Yuqi Cheng, Zongwei Du, Liang Gao, and Weiming Shen. Segment any anomaly without training via hybrid prompt regularization. *arXiv preprint arXiv:2305.10724*, 2023. 3

[12] Qi Chang, Danish Ahmad, Jennifer Toth, Rebecca Bascom, and William E Higgins. Esfpnet: efficient deep learning architecture for real-time lesion segmentation in autofluorescence bronchoscopic video. In *Medical Imaging 2023: Biomedical Applications in Molecular, Structural, and Functional Imaging*, 2023. 9

[13] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021. 9

[14] Tianrun Chen, Lanyun Zhu, Chaotao Deng, Runlong Cao, Yan Wang, Shangzhan Zhang, Zejian Li, Lingyun Sun,

Ying Zang, and Papa Mao. Sam-adapter: Adapting segment anything in underperformed scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 1

[15] Tianrun Chen, Lanyun Zhu, Chaotao Ding, Runlong Cao, Yan Wang, Zejian Li, Lingyun Sun, Papa Mao, and Ying Zang. Sam fails to segment anything?–sam-adapter: Adapting sam in underperformed scenes: Camouflage, shadow, medical image segmentation, and more. *arXiv preprint arXiv:2304.09148*, 2023. 1

[16] Xizi Chen, Jingyang Zhu, Jingbo Jiang, and Chi-Ying Tsui. Tight compression: Compressing cnn model tightly through unstructured pruning and simulated annealing based permutation. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020. 5

[17] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020. 8

[18] Zigeng Chen, Gongfan Fang, Xinyin Ma, and Xinchao Wang. 0.1% data makes segment anything slim. *arXiv preprint arXiv:2312.05284*, 2023. 2, 5, 6, 11, 13

[19] 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*, 2023. 3

[20] Yangming Cheng, Liulei Li, Yuanyou Xu, Xiaodi Li, Zongxin Yang, Wenguan Wang, and Yi Yang. Segment and track anything. *arXiv preprint arXiv:2305.06558*, 2023. 3

[21] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023. 1

[22] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, 2017. 8

[23] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, 2022. 4, 10

[24] Thanos Delatolas, Vicky Kalogeiton, and Dim P Papadopoulos. Learning the what and how of annotation in video object segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024. 4

[25] Ruining Deng, Can Cui, Quan Liu, Tianyuan Yao, Lucas W Remedios, Shunxing Bao, Bennett A Landman, Lee E Wheless, Lori A Coburn, Keith T Wilson, et al. Segment anything model (sam) for digital pathology: Assess zero-shot segmentation on whole slide imaging. *arXiv preprint arXiv:2304.04155*, 2023. 3

[26] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023. 10

[27] Haoyu Dong, Hanxue Gu, Yaqian Chen, Jichen Yang, and Maciej A Mazurowski. Segment anything model 2: an application to 2d and 3d medical images. *arXiv preprint arXiv:2408.00756*, 2024. 4

[28] Shichao Dong, Fayao Liu, and Guosheng Lin. Leveraging large-scale pretrained vision foundation models for label-efficient 3d point cloud segmentation. *arXiv preprint arXiv:2311.01989*, 2023. 1, 4

[29] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[30] Yuchen Duan, Weiyun Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, Yu Qiao, Hongsheng Li, Jifeng Dai, and Wenhai Wang. Vision-rwkv: Efficient and scalable visual perception with rwkv-like architectures. *arXiv preprint arXiv:2403.02308*, 2024. 5, 11

[31] Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Kronecker decomposition for gpt compression. *arXiv preprint arXiv:2110.08152*, 2021. 6

[32] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022. 5

[33] Jianhai Fu, Yuanjie Yu, Ningchuan Li, Yi Zhang, Qichao Chen, Jianping Xiong, Jun Yin, and Zhiyu Xiang. Lite-sam is actually what you need for segment everything. *arXiv preprint arXiv:2407.08965*, 2024. 4, 6, 13

[34] Zhe Gan, Linjie Li, Chunyuan Li, Lijuan Wang, Zicheng Liu, Jianfeng Gao, et al. Vision-language pre-training: Basics, recent advances, and future trends. *Foundations and Trends® in Computer Graphics and Vision*, 14(3–4):163–352, 2022. 1

[35] Ruochen Gao, Donghang Lyu, and Marius Staring. Swinlitemedsam: A lightweight box-based segment anything model for large-scale medical image datasets. *arXiv preprint arXiv:2409.07172*, 2024. 3

[36] Iraklis Giannakis, Anshuman Bhardwaj, Lydia Sam, and Georgios Leontidis. Deep learning universal crater detection using segment anything model (sam). *arXiv preprint arXiv:2304.07764*, 2023. 3

[37] Artem M Grachev, Dmitry I Ignatov, and Andrey V Savchenko. Neural networks compression for language modeling. In *International Conference on Pattern Recognition and Machine Intelligence*, 2017. 6

[38] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021. 10

[39] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 11, 13

[40] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019. 14

[41] Habib Hajimolahoseini, Walid Ahmed, Mehdi Rezagholizadeh, Vahid Partovinia, and Yang Liu. Strategies for applying low rank decomposition to transformer-based models. In *36th Conference on Neural Information Processing Systems (NeurIPS2022)*, 2022. 6

[42] Haibin He, Jing Zhang, Mengyang Xu, Juhua Liu, Bo Du, and Dacheng Tao. Scalable mask annotation for video text spotting. *arXiv preprint arXiv:2305.01443*, 2023. 1, 4

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 4, 7, 9

[44] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022. 2, 8

[45] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 5

[46] Chuanfei Hu, Tianyi Xia, Shenghong Ju, and Xinde Li. When sam meets medical images: An investigation of segment anything model (sam) on multi-phase liver tumor segmentation. *arXiv preprint arXiv:2304.08506*, 2023. 3

[47] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. 10

[48] Mingzhe Hu, Yuheng Li, and Xiaofeng Yang. Skinsam: Empowering skin cancer segmentation with segment anything model. *arXiv preprint arXiv:2304.13973*, 2023. 3

[49] You Huang, Wenbin Lai, Jiayi Ji, Liujuan Cao, Shengchuan Zhang, and Rongrong Ji. Hrsam: Efficiently segment anything in high-resolution images. *arXiv preprint arXiv:2407.02109*, 2024. 4

[50] Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 7

[51] Wei Ji, Jingjing Li, Qi Bi, Tingwei Liu, Wenbo Li, and Li Cheng. Segment anything is not always perfect: An investigation of sam on different real-world applications. *Machine Intelligence Research*, 21:617–630, 2024. 3

[52] Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. Detrs with hybrid matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023. 17, 19

[53] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Yolo by ultralytics, 2023. 7, 17, 18

[54] 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*, 2023. 4

[55] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 1, 2, 3, 8

[56] César Laurent, Camille Ballas, Thomas George, Nicolas Ballas, and Pascal Vincent. Revisiting loss modelling for unstructured pruning. *arXiv preprint arXiv:2006.12279*, 2020. 5

[57] Bao-Hiep Le, Dang-Khoa Nguyen-Vu, Trong-Hieu Nguyen-Mau, Hai-Dang Nguyen, and Minh-Triet Tran. Medficientsam: A robust medical segmentation model with optimized inference pipeline for limited clinical settings. In *Submitted to CVPR 2024: Segment Anything In Medical Images On Laptop*, 2024. 1, 3

[58] Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, Jianfeng Gao, et al. Multimodal foundation models: From specialists to general-purpose assistants. *Foundations and Trends® in Computer Graphics and Vision*, 16(1-2):1–214, 2024. 1

[59] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European conference on computer vision*, 2022. 17, 18

[60] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022. 8

[61] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. In *Advances in Neural Information Processing Systems*, 2022. 4

[62] Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Rethinking vision transformers for mobilenet size and speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 4, 9

[63] Yaqin Li, Dandan Wang, Cao Yuan, Hao Li, and Jing Hu. Enhancing agricultural image segmentation with an agricultural segment anything model adapter. *Sensors*, 23(18): 7884, 2023. 3

[64] Yuheng Li, Mingzhe Hu, and Xiaofeng Yang. Polyp-sam: Transfer sam for polyp segmentation. In *Medical Imaging 2024: Computer-Aided Diagnosis*, 2024. 3

[65] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, 2014. 14

[66] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. 8

[67] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, 2023. 1

[68] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9): 1–35, 2023. 2

[69] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 17, 18

[70] Xinyu Liu, Jing Zhang, Kexin Zhang, Xu Liu, and Lingling Li. Lsvos challenge 3rd place report: Sam2 and cutie based vos. *arXiv preprint arXiv:2408.10469*, 2024. 1, 4

[71] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, 2021. 17

[72] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. In *Advances in Neural Information Processing Systems*, 2021. 5

[73] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022. 10

[74] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024. 13

[75] Ange Lou, Yamin Li, Yike Zhang, Robert F Labadie, and Jack Noble. Zero-shot surgical tool segmentation in monocular video using segment anything model 2. *arXiv preprint arXiv:2408.01648*, 2024. 4

[76] Zhihe Lu, Zeyu Xiao, Jiawang Bai, Zhiwei Xiong, and Xinchao Wang. Can sam boost video super-resolution? *arXiv preprint arXiv:2305.06524*, 2023. 1, 4

[77] Chengtao Lv, Hong Chen, Jinyang Guo, Yifu Ding, and Xianglong Liu. Ptq4sam: Post-training quantization for segment anything. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2024. 2, 5, 6, 11

[78] Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 15(1):654, 2024. 1, 3, 4

[79] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023. 1

[80] Sachin Mehta and Mohammad Rastegari. Mobilevit: lightweight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021. 4

[81] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016. 11

[82] Sovesh Mohapatra, Advait Gosai, and Gottfried Schlaug. Sam vs bet: A comparative study for brain extraction and segmentation of magnetic resonance images using deep learning. *arXiv preprint arXiv:2304.04738*, 2023. 3

[83] Yury Nahshan, Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Alex M Bronstein, and Avi Mendelson. Loss aware post-training quantization. *Machine Learning*, 110(11):3245–3262, 2021. 5

[84] NVIDIA-AI-IOT. Nanosam, 2023. 5, 6, 9, 16

[85] Lucas Prado Osco, Qiusheng Wu, Eduardo Lopes de Lemos, Wesley Nunes Gonçalves, Ana Paula Marques Ramos, Jonathan Li, and José Marcato Junior. The segment anything model (sam) for remote sensing applications: From zero to one shot. *International Journal of Applied Earth Observation and Geoinformation*, 124:103540, 2023. 3

[86] Lorenzo Papa, Paolo Russo, Irene Amerini, and Luping Zhou. A survey on efficient vision transformers: algorithms, techniques, and performance benchmarking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2, 14

[87] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023. 4, 5, 6, 11

[88] PyTorch. Accelerating generative ai with pytorch: Segment anything, fast, 2023. 6, 12, 16

[89] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 2021. 1, 7

[90] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020. 10

[91] Osher Rafaeli, Tal Svoray, and Ariel Nahlieli. Prompt-based segmentation at multiple resolutions and lighting conditions using segment anything model 2. *arXiv preprint arXiv:2408.06970*, 2024. 4

[92] Frano Rajič, Lei Ke, Yu-Wing Tai, Chi-Keung Tang, Martin Danelljan, and Fisher Yu. Segment anything meets point tracking. *arXiv preprint arXiv:2307.01197*, 2023. 3

[93] 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, 6, 8, 14

[94] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015. 4, 7

[95] Saikat Roy, Tassilo Wald, Gregor Koehler, Maximilian R Rokuss, Nico Disch, Julius Holzschuh, David Zimmerer, and Klaus H Maier-Hein. Sam. md: Zero-shot medical

image segmentation capabilities of the segment anything model. *arXiv preprint arXiv:2304.05396*, 2023. 1, 3

[96] 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 *International Conference on Machine Learning*, 2023. 3, 8

[97] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. 4, 5, 11

[98] Tal Shaharabany, Aviad Dahan, Raja Giryes, and Lior Wolf. Autosam: Adapting sam to medical images by overloading the prompt encoder. *arXiv preprint arXiv:2306.06370*, 2023. 3

[99] Chuyun Shen, Wenhao Li, Yuhang Shi, and Xiangfeng Wang. Interactive 3d medical image segmentation with sam 2. *arXiv preprint arXiv:2408.02635*, 2024. 1, 4

[100] Qiuhong Shen, Xingyi Yang, and Xinchao Wang. Anything-3d: Towards single-view anything reconstruction in the wild. *arXiv preprint arXiv:2304.10261*, 2023. 1, 4

[101] Yiqing Shen, Jingxing Li, Xinyuan Shao, Blanca Inigo Romillo, Ankush Jindal, David Dreizin, and Mathias Unberath. Fastsam3d: An efficient segment anything model for 3d volumetric medical images. *arXiv preprint arXiv:2403.09827*, 2024. 4, 6, 10

[102] Changyong Shu, Yifan Liu, Jianfei Gao, Zheng Yan, and Chunhua Shen. Channel-wise knowledge distillation for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 9

[103] 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. *arXiv preprint arXiv:2312.13789*, 2023. 2, 4, 5, 6, 9, 11, 12, 13

[104] Yanfei Songa, Bangzheng Pua, Peng Wanga, Hongxu Jiang, Dong Donga, and Yiqing Shen. Sam-lightening: A lightweight segment anything model with dilated flash attention to achieve 30 times acceleration. *arXiv preprint arXiv:2403.09195*, 2024. 4, 6, 10

[105] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023. 13

[106] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024. 13

[107] Marzieh S Tahaei, Ella Charlaix, Vahid Partovi Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. Kroneckerbert: Learning kronecker decomposition for pre-trained language models via knowledge distillation. *arXiv preprint arXiv:2109.06243*, 2021. 6

[108] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015. 6

[109] George Tang, William Zhao, Logan Ford, David Benhaim, and Paul Zhang. Segment any mesh: Zero-shot mesh part segmentation via lifting segment anything 2 to 3d. *arXiv preprint arXiv:2408.13679*, 2024. 4

[110] Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, et al. Video understanding with large language models: A survey. *arXiv preprint arXiv:2312.17432*, 2023. 1

[111] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1

[112] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 1

[113] Tuyen Tran. The 2nd solution for lsvos challenge rvos track: Spatial-temporal refinement for consistent semantic segmentation. *arXiv preprint arXiv:2408.12447*, 2024. 1, 4

[114] Balakrishnan Varadarajan, Bilge Soran, Forrest Iandola, Xiaoyu Xiang, Yunyang Xiong, Chenchen Zhu, Raghuraman Krishnamoorthi, and Vikas Chandra. Squeeze-sam: User friendly mobile interactive segmentation. *arXiv preprint arXiv:2312.06736*, 2023. 2, 6, 7

[115] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 1, 2

[116] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, et al. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 2023. 1

[117] Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Repvit-sam: Towards real-time segmenting anything. *arXiv preprint arXiv:2312.05760*, 2023. 4, 5, 10, 17

[118] 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*, 2024. 2, 6, 10

[119] Di Wang, Jing Zhang, Bo Du, Minqiang Xu, Lin Liu, Dacheng Tao, and Liangpei Zhang. Samrs: Scaling-up remote sensing segmentation dataset with segment anything model. In *Advances in Neural Information Processing Systems*, 2023. 3

[120] Jinfeng Wang, Qiming Huang, Feilong Tang, Jia Meng, Jionglong Su, and Sifan Song. Stepwise feature fusion: Local guides global. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2022. 9

[121] Teng Wang, Jinrui Zhang, Junjie Fei, Hao Zheng, Yunlong Tang, Zhe Li, Mingqi Gao, and Shanshan Zhao. Caption

anything: Interactive image description with diverse multimodal controls. *arXiv preprint arXiv:2305.02677*, 2023. 3

[122] Yun Wang and Qiang Liu. Aqa: An adaptive post-training quantization method for activations of cnns. *IEEE Transactions on Computers*, 73(08):2025–2035, 2024. 5

[123] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019. 5

[124] Genta Indra Winata, Andrea Madotto, Jamin Shin, Elham J Barezi, and Pascale Fung. On the effectiveness of low-rank matrix factorization for lstm model compression. *arXiv preprint arXiv:1908.09982*, 2019. 6

[125] Junde Wu, Wei Ji, Yuanpei Liu, Huazhu Fu, Min Xu, Yanwu Xu, and Yueming Jin. Medical sam adapter: Adapting segment anything model for medical image segmentation. *arXiv preprint arXiv:2304.12620*, 2023. 9

[126] Jay Zhangjie Wu, Xiuyu Li, Difei Gao, Zhen Dong, Jinbin Bai, Aishani Singh, Xiaoyu Xiang, Youzeng Li, Zuwei Huang, Yuanxi Sun, et al. Cvpr 2023 text guided video editing competition. *arXiv preprint arXiv:2310.16003*, 2023. 4

[127] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European conference on computer vision*, 2022. 4, 8, 9

[128] Aoran Xiao, Weihao Xuan, Heli Qi, Yun Xing, Naoto Yokoya, and Shijian Lu. Segment anything with multiple modalities. *arXiv preprint arXiv:2408.09085*, 2024. 14

[129] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. In *Advances in Neural Information Processing Systems*, 2021. 10

[130] Defeng Xie, Ruichen Wang, Jian Ma, Chen Chen, Haonan Lu, Dong Yang, Fobo Shi, and Xiaodong Lin. Edit everything: A text-guided generative system for images editing. *arXiv preprint arXiv:2304.14006*, 2023. 1, 3

[131] 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 *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2024. 4, 6, 7, 8, 15, 16

[132] Canwen Xu and Julian McAuley. A survey on model compression and acceleration for pretrained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. 2

[133] Shilin Xu, Haobo Yuan, Qingyu Shi, Lu Qi, Jingbo Wang, Yibo Yang, Yining Li, Kai Chen, Yunhai Tong, Bernard Ghanem, et al. Rap-sam: Towards real-time all-purpose segment anything. *arXiv preprint arXiv:2401.10228*, 2024. 6, 8

[134] Xiuwei Xu, Huangxing Chen, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. Embodiedsam: Online segment any 3d thing in real time. *arXiv preprint arXiv:2408.11811*, 2024. 1, 4

[135] Yosuke Yamagishi, Shouhei Hanaoka, Tomohiro Kikuchi, Takahiro Nakao, Yuta Nakamura, Yukihiro Nomura, Soichiro Miki, Takeharu Yoshikawa, and Osamu Abe. Zero-shot 3d segmentation of abdominal organs in ct scans using segment anything model 2: Adapting video tracking capabilities for 3d medical imaging. *arXiv preprint arXiv:2408.06170*, 2024. 1, 4

[136] Zhiling Yan, Weixiang Sun, Rong Zhou, Zhengqing Yuan, Kai Zhang, Yiwei Li, Tianming Liu, Quanzheng Li, Xiang Li, Lifang He, et al. Biomedical sam 2: Segment anything in biomedical images and videos. *arXiv preprint arXiv:2408.03286*, 2024. 4

[137] Jinyu Yang, Mingqi Gao, Zhe Li, Shang Gao, Fangjing Wang, and Feng Zheng. Track anything: Segment anything meets videos. *arXiv preprint arXiv:2304.11968*, 2023. 1, 3

[138] Zhendong Yang, Zhe Li, Xiaohu Jiang, Yuan Gong, Zehuan Yuan, Danpei Zhao, and Chun Yuan. Focal and global knowledge distillation for detectors. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2022. 9

[139] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In *Advances in Neural Information Processing Systems*, 2022. 5

[140] 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. 1, 4

[141] Tao Yu, Runseng Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, and Zhibo Chen. Inpaint anything: Segment anything meets image inpainting. *arXiv preprint arXiv:2304.06790*, 2023. 3

[142] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022. 4, 13

[143] Weihao Yu, Chenyang Si, Pan Zhou, Mi Luo, Yichen Zhou, Jiashi Feng, Shuicheng Yan, and Xinchao Wang. Metaformer baselines for vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(02):896–912, 2023. 10

[144] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. 6

[145] Haobo Yuan, Xiangtai Li, Lu Qi, Tao Zhang, Ming-Hsuan Yang, Shuicheng Yan, and Chen Change Loy. Mamba or rwkv: Exploring high-quality and high-efficiency segment anything model. *arXiv preprint arXiv:2406.19369*, 2024. 5, 11

[146] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In *European conference on computer vision*, 2022. 5, 11

[147] 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. 2, 4, 5, 6, 8, 9, 10, 12

[148] Chaoning Zhang, Dongshen Han, Sheng Zheng, Jinwoo Choi, Tae-Ho Kim, and Choong Seon Hong. Mobilesamv2: Faster segment anything to everything. *arXiv preprint arXiv:2312.09579*, 2023. 6, 12

[149] Chunhui Zhang, Li Liu, Yawen Cui, Guanjie Huang, Weilin Lin, Yiqian Yang, and Yuehong Hu. A comprehensive survey on segment anything model for vision and beyond. *arXiv preprint arXiv:2305.08196*, 2023. 2, 3

[150] Chaoning Zhang, Fachrina Dewi Puspitasari, Sheng Zheng, Chenghao Li, Yu Qiao, Taegoo Kang, Xinru Shan, Chenshuang Zhang, Caiyan Qin, Francois Rameau, et al. A survey on segment anything model (sam): Vision foundation model meets prompt engineering. *arXiv preprint arXiv:2306.06211*, 2023.

[151] Chunhui Zhang, Yawen Cui, Weilin Lin, Guanjie Huang, Yan Rong, Li Liu, and Shiguang Shan. Segment anything for videos: A systematic survey. *arXiv preprint arXiv:2408.08315*, 2024. 2, 3

[152] Leying Zhang, Xiaokang Deng, and Yu Lu. Segment anything model (sam) for medical image segmentation: A preliminary review. In *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2023. 2

[153] Mingya Zhang, Liang Wang, Limei Gu, Zhao Li, Yaohui Wang, Tingshen Ling, and Xianping Tao. Sam2-path: A better segment anything model for semantic segmentation in digital pathology. *arXiv preprint arXiv:2408.03651*, 2024. 4

[154] Renrui Zhang, Zhengkai Jiang, Ziyu Guo, Shilin Yan, Junting Pan, Xianzheng Ma, Hao Dong, Peng Gao, and Hongsheng Li. Personalize segment anything model with one shot. *arXiv preprint arXiv:2305.03048*, 2023. 1, 3

[155] Yichi Zhang and Zhenrong Shen. Unleashing the potential of sam2 for biomedical images and videos: A survey. *arXiv preprint arXiv:2408.12889*, 2024. 2, 4

[156] Yichi Zhang, Yuan Cheng, and Yuan Qi. Semisam: Exploring sam for enhancing semi-supervised medical image segmentation with extremely limited annotations. *arXiv preprint arXiv:2312.06316*, 2023. 3

[157] Yichi Zhang, Zhenrong Shen, and Rushi Jiao. Segment anything model for medical image segmentation: Current applications and future directions. *Computers in Biology and Medicine*, 171:108238, 2024. 2, 3

[158] Zhenghao Zhang, Zhichao Wei, Shengfan Zhang, Zuozhuo Dai, and Siyu Zhu. Uvosam: A mask-free paradigm for unsupervised video object segmentation via segment anything model. *arXiv preprint arXiv:2305.12659*, 2023. 3, 4

[159] Zhuoyang Zhang, Han Cai, and Song Han. Efficientvit-sam: Accelerated segment anything model without performance loss. *arXiv preprint arXiv:2402.05008*, 2024. 4, 5, 6, 10, 15, 16, 17

[160] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023. 1

[161] 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. 2, 4, 6, 7, 12, 15

[162] Yingwei Zhao. Efficient sam for medical image analysis, 2023. 4, 5, 6, 9, 10

[163] Yuyang Zhao, Enze Xie, Lanqing Hong, Zhenguo Li, and Gim Hee Lee. Make-a-protagonist: Generic video editing with an ensemble of experts. *arXiv preprint arXiv:2305.08850*, 2023. 4

[164] 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. 2, 4, 5, 6, 10, 15, 17

[165] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *European Conference on Computer Vision*, 2022. 17, 19

[166] Jiayuan Zhu, Yunli Qi, and Junde Wu. Medical sam 2: Segment medical images as video via segment anything model 2. *arXiv preprint arXiv:2408.00874*, 2024. 4

[167] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Wang, Lijuan Wang, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. *Advances in Neural Information Processing Systems*, 36, 2024. 1