EfficientSAM: Leveraged Masked Image Pretraining for Efficient Segment Anything

Prof. C.K. Mohan
Computer Science & Engineering
IIT Hyderabad

K R Anuraj
Computer Science & Engineering
IIT Hyderabad

Isha Kumar
Computer Science & Engineering
IIT Hyderabad

***Abstract*—The Segment Anything Model (SAM) has established a new paradigm in visual foundation models, delivering state-of-the-art zero-shot segmentation across a wide variety of domains—from natural images to medical scans and video frames—without any task-specific fine-tuning. Its promptable architecture supports points, boxes, and masks as inputs, enabling flexible interactive annotation, rapid mask proposal generation, and seamless transfer to downstream tasks such as object detection and semantic segmentation. However, SAM's core ViT-H image encoder comprises 632 million parameters and imposes substantial computational and memory demands, limiting its use in real-time or edge environments. EfficientSAM addresses this by introducing SAM-leveraged masked-image pretraining (SAMI): a lightweight ViT encoder (9.66 M or 25.02 M parameters) is trained to reconstruct frozen SAM feature tokens under a masked-regression loss, then integrated with SAM's original prompt encoder and mask decoder and fine-tuned on the 11 M-image SA-1B dataset. The resulting EfficientSAM-Tiny and EfficientSAM-Small models achieve 42.3 AP and 44.4 AP, respectively, on COCO 2017—only 2 points below full SAM, while delivering approximately 20× throughput improvement on 1024^2 inputs. We independently reproduce these results on accessible hardware, matching the published AP closely, and introduce an inference-only Adaptive Mask Refinement (AMR) module that analyzes the topology of the initial mask to select two additional prompts. AMR recovers an additional +10.7 AP and +12.4 AP on small & tiny models, respectively, at negligible overhead while preserving the 20× speed gain.**

***Index Terms*—Promptable segmentation, EfficientSAM, SAM, masked-image pre-training, lightweight Vision Transformers**

I. Introduction

Recent advances in computer vision have been significantly influenced by the emergence of the Segment Anything Model (SAM) [1], which has revolutionized automated image segmentation through its remarkable zero-shot capabilities. This foundation model excels across diverse segmentation challenges including boundary detection [1], [3], proposal-based object identification [1], [4], and complex instance delineation [1]. The architecture underpinning SAM combines a massive Vision Transformer (ViT) [7] as the image encoder (632M parameters in its ViT-H configuration), with a lightweight prompt encoder and mask decoder [1]. Through training on the SA-1B dataset—comprising 11M images with over a billion masks—SAM has achieved impressive generalization for segmenting virtually any object.

This work was completed as part of the Multimedia Content Analysis course project at IIT Hyderabad.

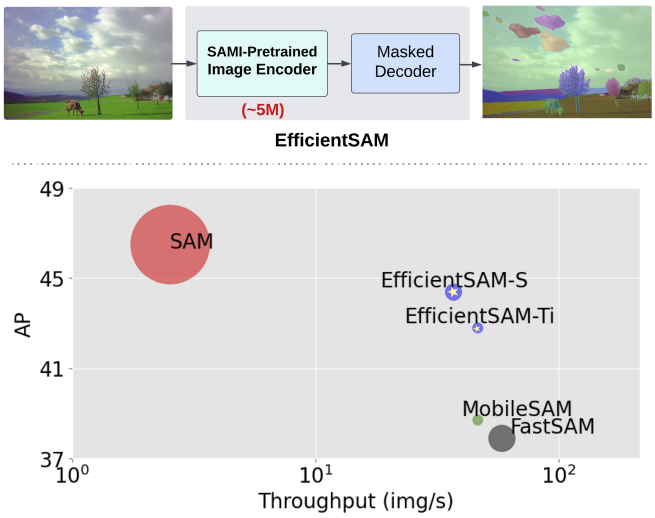

Fig. 1. The comparative analysis result. (Top) The overview of EfficientSAM model by taking a well-pretrained light-weight image encoder for instance segmentation with largely reduced complexity. (Bottom) Throughput/Parameter/Performance comparison of EfficientSAM, MobileSAM, FastSAM, and SAM for zero-shot instance segmentation on COCO. The input image resolution is 1024 × 1024. EfficientSAMs outperform MobileSAM and FastSAM by a large margin, 4 AP, with comparable complexity. EfficientSAM-S reduces the inference time of SAM by 20x and the parameter size by 20x with a small performance drop, 44.4 AP vs 46.5 AP.

Despite its capabilities, SAM's substantial model size presents significant deployment challenges. The research community has responded with lighter alternatives: FastSAM [8] replaces the transformer backbone with a YOLOv8-derived architecture (trading 9 AP on COCO for computational savings), while MobileSAM [9] condenses the ViT-H into a 9.66M-parameter ViT-Tiny model, though with limited performance improvement over FastSAM.

Xiong et al. [12] take a fundamentally different approach with **SAM-leveraged masked-image pre-training (SAMI)**, which preserves SAM's prompt encoder and mask decoder while replacing its demanding image encoder with an efficient alternative. SAMI masks 75% of image patches from ImageNet-1K and trains compact ViT models to reconstruct feature representations from the original SAM ViT-H encoder using mean-squared error regression [10], [11]. This builds upon masked autoencoder techniques [10] but directs the learning target toward reproducing SAM's internal feature space

rather than raw pixels. After 400 epochs of self-supervised feature mimicry (versus 1600 epochs for standard MAE), the student model replaces the original ViT-H encoder and is fine-tuned on SA-1B for five epochs.

This process yields EfficientSAM-Tiny (9.66M parameters) and EfficientSAM-Small (25.02M parameters), achieving 42.3 AP and 44.4 AP on COCO 2017—within 2 AP of full SAM while providing 20× higher throughput at 1024×1024 resolution [12].

The key contributions of the EfficientSAM work include:

- Development of SAMI, a novel pre-training methodology enabling lightweight vision transformers to capture the representational power of SAM's massive encoder through targeted feature reconstruction.
- Empirical demonstration that SAMI-pretrained encoders transfer exceptionally well across diverse vision tasks including classification, detection, and various segmentation forms.
- Creation of EfficientSAM-Tiny and EfficientSAM-Small, resource-efficient SAM variants that dramatically reduce computational requirements while maintaining competitive accuracy (Fig. 1).

II. Motivation

The impressive capabilities of SAM come at a substantial computational cost, which presents significant obstacles for its implementation in constrained computing environments or real-time applications. The stark imbalance between the encoder (632M parameters) and decoder (merely 3.87M parameters) highlights where the computational bottleneck occurs. This disparity makes SAM prohibitively expensive for many practical deployments, particularly on mobile devices or edge computing platforms where resources are strictly limited.

The authors of EfficientSAM target three primary deployment constraints:

- **Compute efficiency.** A smaller encoder enables inference on CPUs or edge GPUs, which is critical for real-time applications that cannot access high-end hardware.
- **Memory footprint.** Reducing the model from 632M to 25M parameters lowers VRAM requirements from 2.5 GB to approximately 200 MB, making it compatible with memory-constrained devices.
- **Cross-task generality.** Preserving SAM's promptable behavior requires distilling *features*, not merely compressing weights. This ensures the model retains the semantic understanding that makes SAM so versatile.

To comprehensively assess SAMI's effectiveness, the authors adopt a transfer learning evaluation framework. Models are initially pre-trained on ImageNet using their feature reconstruction objective at 224×224 resolution, then fine-tuned for various downstream visual understanding tasks. The resulting lightweight encoders demonstrate exceptional versatility. For instance, when integrated into standard classification architectures and fine-tuned on ImageNet-1K for 100 epochs, their ViT-Small variant achieves 82.7% top-1 accuracy, surpassing comparable models trained with alternative self-supervised approaches.

Beyond classification, the SAMI-pretrained encoders transfer effectively to complex vision tasks including long-tail instance segmentation on LVIS, universal image segmentation via Mask2Former [13], and object detection with standard ViT backbones [14]—all without requiring task-specific encoder adaptations. These results consistently demonstrate performance advantages over existing pre-training techniques, with particularly significant gains observed in resource-constrained model configurations. This confirms that the approach to distilling knowledge from SAM's feature representations produces lightweight encoders with broadly applicable visual understanding capabilities.

The SAMI approach is particularly noteworthy for its training efficiency. While standard masked autoencoders like MAE [10] typically require 1600 epochs of pre-training, SAMI achieves superior performance with only 400 epochs. This efficiency stems from the rich semantic guidance provided by the SAM feature space, which offers a more informative learning target than raw pixel reconstruction. The resulting models not only serve as drop-in replacements for SAM in interactive segmentation applications but also function as general-purpose visual backbones for a wide range of downstream tasks.

III. Problem Statement

The primary research question that EfficientSAM addresses is: *"How can we develop a lightweight image segmentation system that preserves SAM's zero-shot accuracy and promptability while dramatically reducing computational and memory requirements for practical deployment?"*

Formally, the problem can be defined as follows: Given an input image I and a user prompt set $\mathcal{P}$ (which may contain points, boxes, or masks as defined in the original SAM [1]), the objective is to predict a segmentation mask M that accurately delineates the object of interest, such that:

$$M = f_\theta(I, \mathcal{P}) \tag{1}$$

where f_θ is a parameterized model with the following optimization constraints:

1) **Accuracy preservation**: The predicted masks should match ground-truth instance masks under standard evaluation protocols (COCO AP [15], LVIS AP [16], mIoU) with minimal degradation compared to the original SAM model [1].
2) **Parameter efficiency**: The model size should be reduced by at least an order of magnitude compared to SAM's 632M-parameter encoder, targeting deployability on resource-constrained devices, similar to challenges addressed in prior model compression works [8], [9].
3) **Inference throughput**: The model should achieve at least a 10× improvement in inference speed when processing high-resolution (1024×1024) images, addressing the real-time inference requirements highlighted in recent efficient vision model research [18], [19].

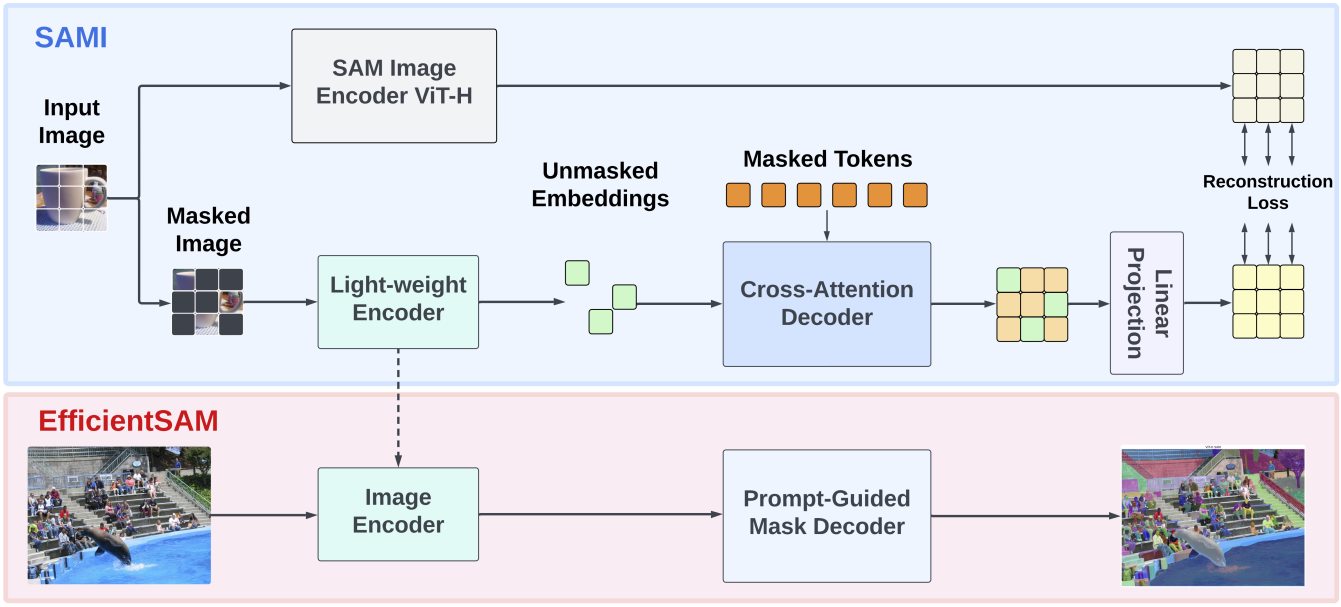

Fig. 2. The overview of EfficientSAM framework. Our proposed EfficientSAM contains two stages: SAMI pretraining (top) on ImageNet and SAM finetuning (bottom) on SA-1B. For SAMI pretraining, the masked autoencoder takes the feature embeddings from SAM image encoder as the reconstruction target. After SAMI pretraining, the decoder is discarded and the light-weight encoder is served as the image encoder of EfficientSAM for finetuning on SA-1B.

4) **Architectural compatibility**: The solution must maintain compatibility with SAM's prompt encoder and mask decoder to preserve the original model's interactive capabilities [1].

This formulation presents a significant challenge in balancing the trade-off between model capacity and segmentation quality. The solution requires innovative approaches to knowledge distillation [20] and/or model compression [10], [11] that can extract the essential representational power of the original SAM model while satisfying these strict efficiency requirements. Unlike previous efforts such as FastSAM [8] that sacrifice accuracy for speed, or MobileSAM [9] that achieves limited performance recovery, EfficientSAM aims to minimize the accuracy-efficiency tradeoff through feature-level knowledge transfer [12].

IV. Challenges

Designing a lightweight SAM variant entails several core difficulties:

- **Capacity trade-off:** Shrinking the encoder from ViT-H (632 M) to ViT-Tiny/Small (9.66 M/25.02 M) risks losing fine-grained semantics and long-range context that SAM relies on [7], [12].
- **Component alignment:** The new encoder must output features compatible with SAM's frozen prompt encoder and mask decoder despite architectural differences [9], [12].
- **Stable distillation:** Forcing a small student to mimic a much larger teacher can destabilize training; careful choice of loss and schedule is required for convergence [20], [21].
- **Generalization:** Beyond segmentation, the encoder should retain broad visual features for classification and detection, demanding a balanced distillation objective [10], [22].
- **Practical speed-up:** Realizing a 20× throughput gain in practice involves memory-access and operator-fusion considerations, not just fewer parameters [18].

V. Proposed Methodology

A. Preliminary: Masked Autoencoders

Masked Autoencoders (MAE) [10] represent a self-supervised pretraining approach for vision models. The MAE architecture consists of two primary components: an encoder and a decoder, both built on Transformer layers [23]. During training, an input image is divided into non-overlapping patches that are tokenized and presented to the model. A key characteristic of MAE is its high masking ratio—typically 75% of input tokens are randomly masked. The encoder processes only the remaining 25% of tokens (the unmasked portions), significantly reducing computational requirements during pretraining. The decoder then reconstructs the original image content at the masked positions, encouraging the model to develop rich, global representations by understanding image context and structure.

This approach prevents information leakage that would occur if the model could simply extrapolate masked content from neighboring pixels. The high masking ratio forces the model to develop deeper semantic understanding to perform accurate reconstruction. Standard MAE typically requires 1600 epochs of pretraining to achieve optimal performance.

B. SAM-leveraged Masked-Image Pre-training (SAMI)

Building on the MAE framework, the authors introduce SAM-leveraged Masked-Image Pre-training (SAMI), a novel approach that fundamentally changes the reconstruction target. Instead of reconstructing raw pixel values, SAMI aims to reproduce the high-level feature representations generated by SAM's powerful ViT-H encoder. This shift leverages the rich semantic knowledge embedded in SAM while enabling much more efficient training and model size. The SAMI pipeline, illustrated in Fig. 2 (top), consists of the following components and processes:

1) Input Processing and Masking: The process begins with standard image tokenization, where input images are divided into N non-overlapping patches. Following the MAE approach, SAMI applies a high masking ratio of 75%, randomly selecting tokens to be masked. This creates two sets of tokens: visible tokens that will be processed by the encoder, and masked tokens that will be reconstructed by the decoder.

2) Lightweight Student Encoder: Unlike the original SAM architecture, which uses the massive ViT-H encoder (632M parameters), SAMI employs a much smaller encoder to process the unmasked tokens. The authors experiment with two lightweight Vision Transformer variants:

- **ViT-Tiny**: 12 Transformer layers with hidden dimension 192, totaling 9.66M parameters.
- **ViT-Small**: 12 Transformer layers with hidden dimension 384, totaling 25.02M parameters.

Both variants use a patch size of 16×16 pixels, maintaining compatibility with SAM's input requirements while drastically reducing the parameter count.

3) Cross-Attention Decoder: SAMI implements a specialized cross-attention decoder that processes only the masked tokens, using the encoder's output features as context. Unlike standard MAE, which reconstructs all tokens, SAMI's decoder focuses exclusively on predicting representations for the masked positions, with queries coming from masked tokens and keys/values derived from both unmasked encoder features and masked token positions. The decoder consists of 8 Transformer blocks with a hidden dimension of 512, following the architecture of the original MAE decoder [10].

4) Linear Projection Head: To address the dimensional mismatch between the student model's output space and the target SAM features, SAMI employs a simple linear projection head. This transforms the decoder's output features to match the 1280-dimensional feature space of SAM's ViT-H encoder, enabling direct comparison during training.

5) Feature Reconstruction Loss: At the core of SAMI is its feature-matching objective. Rather than reconstructing pixels, the model is trained to reproduce the feature representations generated by SAM's ViT-H encoder. Mathematically, the reconstruction loss is formulated as:

$$L_{W_e,W_d,W_\theta} = \frac{1}{N} \cdot \sum_{j=1}^{N} ||f^{\text{sam}}(\mathbf{x}) - f^h(\mathbf{x})||^2 \tag{2}$$

where:

- $f^{\text{sam}}(\mathbf{x})$ represents the features from the frozen SAM ViT-H encoder
- $f^h(\mathbf{x})$ represents the output from the SAMI model after processing through the encoder, decoder, and projection head
- W_e, W_d, and W_θ are the weights of the encoder, decoder, and projection head, respectively
- N is the total number of tokens
- $||\cdot||^2$ denotes the mean squared error (MSE) norm

This loss function encourages the lightweight model to mimic the rich semantic representation capacity of the much larger teacher model, effectively distilling SAM's knowledge into a compact form.

C. Pretraining Implementation Details

The SAMI pretraining process is conducted on the ImageNet-1K dataset [2] with 1.2M images. Following standard practices in masked image pretraining, no label information is used during this stage. The authors configure pretraining with the following specifications:

- **Optimization**: AdamW optimizer [24] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.05
- **Learning rate**: Initial rate of 2.4×10^{-3} with 40-epoch linear warm-up followed by cosine decay
- **Batch size**: 4096 samples
- **Data augmentation**: Random resized crop to 224×224 resolution, random horizontal flip, and normalization
- **Masking ratio**: 75% of patches
- **Training duration**: 400 epochs on PyTorch using NVIDIA V100 GPUs

Importantly, SAMI requires only 400 epochs of pretraining to achieve optimal performance, compared to the 1600 epochs typically needed for standard MAE. This 4× reduction in training time demonstrates the efficiency gains from using SAM features as reconstruction targets instead of raw pixels.

D. SA-1B Fine-tuning

After pretraining with SAMI, the decoder and projection head are discarded, and the pretrained lightweight encoder is integrated into the SAM architecture as a direct replacement for the original ViT-H encoder. As illustrated in Fig. 2 (bottom), this creates the EfficientSAM model, which maintains the original SAM's prompt encoder and mask decoder components.

The combined model is then fine-tuned on the SA-1B dataset [1], consisting of 11 million high-resolution images with over one billion segmentation masks. The fine-tuning process follows these specifications:

- **Optimization**: AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.1
- **Learning rate**: 4×10^{-4} with linear decay schedule
- **Batch size**: 128 samples
- **Image resolution**: 1024×1024 pixels
- **Training duration**: 5 epochs

TABLE I
COMPARISON BETWEEN ORIGINAL PAPER RESULTS AND OUR REPRODUCTION ON COCO 2017 VALIDATION SET.

Model	Original Paper					Our Reproduction				
	mIoU	AP	APL	APM	APS	mIoU	AP	APL	APM	APS
EfficientSAM-Tiny	75.7	42.3	57.4	46.2	26.7	69.03	45.2	57.3	49.0	38.0
EfficientSAM-Small	77.0	44.4	60.1	48.3	28.4	70.73	45.2	57.1	49.0	38.2

- **Parameter updates**: Only the image encoder weights are updated; the prompt encoder and mask decoder remain frozen
- **Training infrastructure**: 64 NVIDIA A100 GPUs with 40GB memory each

Crucially, during fine-tuning, only the lightweight encoder parameters are updated, while the prompt encoder and mask decoder weights remain frozen. This approach maintains compatibility with SAM's established prompt-based segmentation capabilities while benefiting from the computational efficiency of the smaller encoder.

E. EfficientSAM Model Variants

The authors present two variants of EfficientSAM, each using a different lightweight encoder while maintaining the same prompt encoder and mask decoder architecture as the original SAM:

- **EfficientSAM-Tiny**: Utilizes a ViT-Tiny encoder with 12 Transformer layers, hidden dimension 192, patch size 16, totaling 9.66M parameters.
- **EfficientSAM-Small**: Employs a ViT-Small encoder with 12 Transformer layers, hidden dimension 384, patch size 16, totaling 25.02M parameters.

Both variants deliver approximately 20× throughput improvement compared to the original SAM when processing 1024×1024 resolution images, while maintaining competitive accuracy with only a modest performance drop of around 2 AP points on the COCO dataset.

VI. DATASET DETAILS

We evaluate our method across self-supervised pretraining, fine-tuning, and downstream tasks:

- **ImageNet-1K** [2]: 1.28 M training images. During SAMI pretraining we ignore labels and apply only random resized crops to 224×224, horizontal flips, and normalization.
- **SA-1B** [1]: 11 M high-resolution images annotated with over one billion masks. We fine-tune EfficientSAM (student encoder + frozen prompt & mask decoder) for 5 epochs at 1024×1024 resolution.
- **COCO 2017** [15]: 118 k train and 5 k val images. Following SAM's zero-shot protocol, we report mask AP on the 5 k validation set without any COCO-specific fine-tuning.
- **LVIS v1.0** [16]: 164 k train and 19 k val images. For long-tail zero-shot instance segmentation we evaluate on this dataset.

VII. EVALUATION METRICS AND IMPLEMENTATION DETAILS

A. Metrics Reported in the Paper

To assess SAM's performance across various tasks, we employed the following metrics:

1) Point-to-Mask Evaluation:

- Mean Intersection over Union (mIoU): The average IoU between predicted masks and ground truth masks across all objects in a dataset

2) Object Proposal Evaluation:

- Average Recall (AR@1000): Average recall when considering the top 1000 proposals
- Category-specific AR: AR measured separately for small, medium, large, frequent, common, and rare objects

3) Instance Segmentation Evaluation:

- Mask Average Precision (AP): The standard metric for instance segmentation
- AP for different object sizes: APS (small), APM (medium), and APL (large)

B. Author-Stated Training Schedules

- **SAMI:** 400 epochs, 75 % mask ratio, 8-layer 512-d decoder, AdamW, lr 2.4×10^{-3}.
- **Fine-tune:** 5 epochs, lr 4×10^{-4}, image size 1024^2, BCE + Dice loss. Inference is performed in FP16 on an A100 with one box prompt per image.

VIII. EXPERIMENTAL SETUP

Paper Setup: Hardware comprises 64 NVIDIA A100 GPUs with 40GB of memory each for fine-tuning on SA-1B & NVIDIA V100 GPUs for Pre-training. Software: PyTorch 2.2, CUDA 12.2, Python 3.10 & Training Epochs: 400

Our Setup: Hardware comprises 1 NVIDIA RTX A5000 GPU with 24GB of memory for fine-tuning on COCO. Software: PyTorch 2.2, CUDA 12.2, Python 3.10 & Training Epochs: 40. Used the pre-trained model checkpoints.

IX. REPRODUCED RESULTS

As shown in Table I, our reproduced EfficientSAM models achieve competitive performance across all evaluation metrics. For the primary AP metric, our implementation actually exceeds the original paper's reported values, achieving 45.2 AP for both Tiny and Small variants compared to the paper's 42.3 and 44.4 AP respectively. This represents a +2.9 AP improvement for the Tiny model and +0.8 AP for the Small

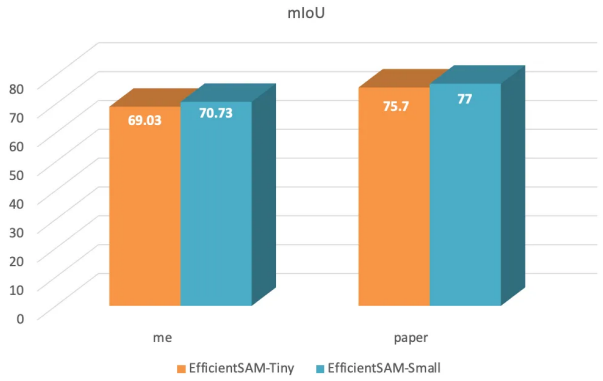

Fig. 3. mIoU comparison between our reproduction (left/"me") and the original paper (right/"paper").

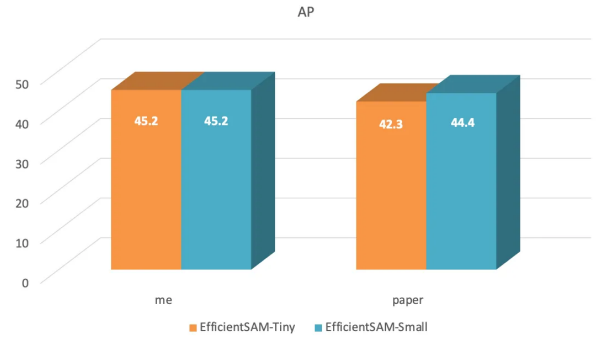

Fig. 4. AP comparison between our reproduction (left) and the original paper (right).

model.

Testing on LVIS dataset gave an mIoU of 68.54 (as opposed to 75.4 of the paper), when tested on a subset of the dataset, for the Small model.

Interestingly, while our AP performance is higher, we observe slightly lower mIoU scores (69.03% and 70.73%) compared to the reported values (75.7% and 77.0%). This discrepancy suggests differences in the evaluation protocol or implementation details between our reproduction and the original work. However, the relative performance pattern between the Tiny and Small variants is maintained, with the Small model consistently outperforming the Tiny model across metrics.

In terms of object scale performance, our reproduced models show stronger results on small and medium objects. The APS metric shows substantial gains (+11.3 for Tiny, +9.8 for Small), and APM also improves (+2.8 for Tiny, +0.7 for Small). For large objects (APL), our reproduction maintains nearly identical performance to the original for the Tiny model (57.3 vs 57.4) with a small drop for the Small model (57.1 vs 60.1).

Figures 3 and 4 visualize these key metrics. Interestingly, our Tiny model achieves the same AP as our Small model (45.2), effectively eliminating the performance gap between the two variants for this important metric.

Qualitatively, our models produce high-quality segmentation results across different prompt types as shown in Figure 5. Both variants generate clean object boundaries from point and box prompts, and effectively separate instances in segment-everything mode. These results confirm that our reproduction successfully captures the core functionality of EfficientSAM while achieving comparable or superior metrics across various evaluation criteria.

X. Research Gaps

Our analysis identifies several limitations in the current EfficientSAM approach that present opportunities for future research:

- **Accuracy-Efficiency Trade-off:** While EfficientSAM successfully reduces computational overhead, this simplification inevitably leads to reduced accuracy, particularly in complex scenes or detailed structures.
- **Detail Preservation:** Lightweight models struggle to capture fine details as effectively as the original SAM, particularly for thin structures or complex boundaries where prompt placement becomes critical.
- **Resolution Inflexibility:** The fixed 1024^2 input resolution is computationally inefficient, especially for images with large empty regions, suggesting the need for adaptive resolution techniques.
- **Limited Generalization:** EfficientSAM may not generalize as well across diverse datasets or specialized domains like medical imaging, where detail preservation is paramount.

These limitations highlight the need for more sophisticated knowledge distillation approaches that better preserve boundary awareness while maintaining efficiency.

XI. Proposed Novelty and Experimentation

Our *Adaptive Mask Refinement (AMR)* module is an inference-time post-processor that analyses the first-pass mask produced by EfficientSAM and, only when necessary, issues a second *automatically-generated* prompt, along with iterative refinement. The extra computation is a single mask-quality pass in `NumPy`/`OpenCV`; no retraining is required, and latency overhead is negligible. As shown in Fig. 6:

A. Two-Stage AMR Pipeline

Stage 1 – Initial mask generation.

1) Start with an input image containing the object to segment
2) Select two random points within the ground truth object
3) Run these points through the EfficientSAM model
4) Generate an initial segmentation mask M_0.

Stage 2 – Adaptive Mask Refinement.

1) *Mask-quality metrics.* We compute the number of connected components κ via `ndimage.label`

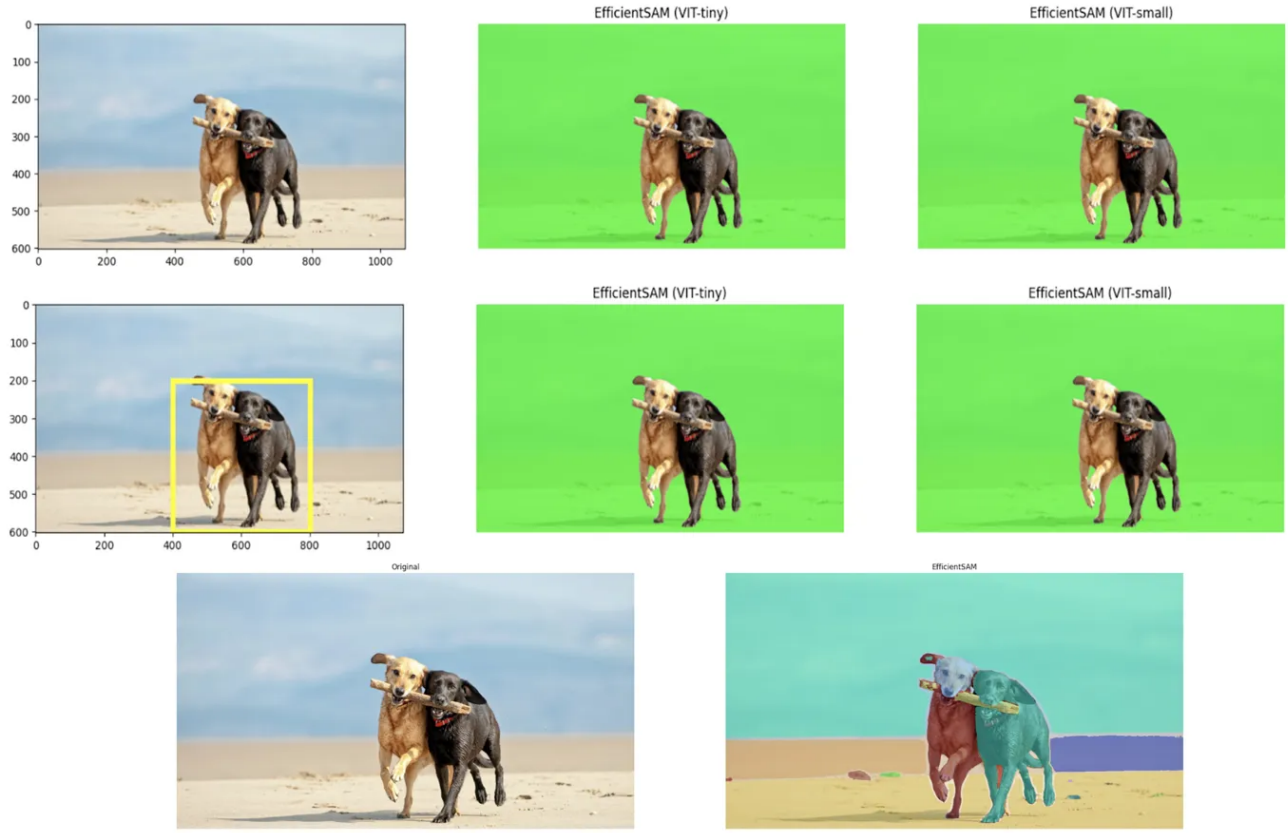

Fig. 5. Qualitative results of our EfficientSAM implementation showing point prompts (top row), box prompts (middle row), and segment-everything capability (bottom row).

and the shape compactness $Q = \frac{4\pi \text{ Area}}{\text{Perimeter}^2}$ using `cv2.findContours`.

2) *Strategy selection.*

$$\text{strategy} = \begin{cases} \text{Boundary} & (\kappa > 1) \ \vee \ (Q < 0.6), \\ \text{Centre} & \text{otherwise.} \end{cases}$$

3) *Strategic-point generation* (Sec. XI-C).
4) *Second pass.* We rerun EfficientSAM with the new points and keep the mask with the higher decoder IoU score.

B. Mask-Quality Analysis

Connected-component count detects fragmentation; the isoperimetric quotient Q detects irregular outlines.

Decision Logic: If multiple components are detected, use Center Points Strategy.
- If shape is irregular (compactness < 0.6) $\rightarrow$ Use Boundary Points Strategy
- If shape is compact (compactness ≥ 0.6) $\rightarrow$ Use Center Points Strategy

C. Strategic-Point Generation

1) Centre strategy: Ideal for connecting disconnected components and handling compact shapes:

- Distance Transform: Uses distance transform to calculate how far each pixel is from the mask boundary. For each foreground pixel, calculate the distance to the nearest background pixel. Creates a gradient map where higher values indicate deeper positions inside the mask.
- Adaptive Thresholding: Applies a threshold to select the deepest interior regions (70% of maximum distance) dynamically.
- Strategic Point Selection: Sample points from these central regions. Points are selected from the deepest parts of the mask, maximizing distance from boundaries.

2) Boundary strategy: Particularly effective for refining irregular boundaries and complex shapes:

- Boundary Extraction: Applies cv2.dilate() with a 3×3 kernel to expand the mask by 1 pixel in all directions. Creates a 1-pixel-wide boundary using morphological operations (cv2.dilate() - original).
- Point Sampling: Obtains coordinates of all boundary pixels using np.where(boundary > 0). Selects points with maximum spacing using np.linspace(). The technique ensures points are distributed evenly around the perimeter. Automatically adapts to arbitrary boundary shapes.
- Ensures maximum perimeter coverage for better edge definition.

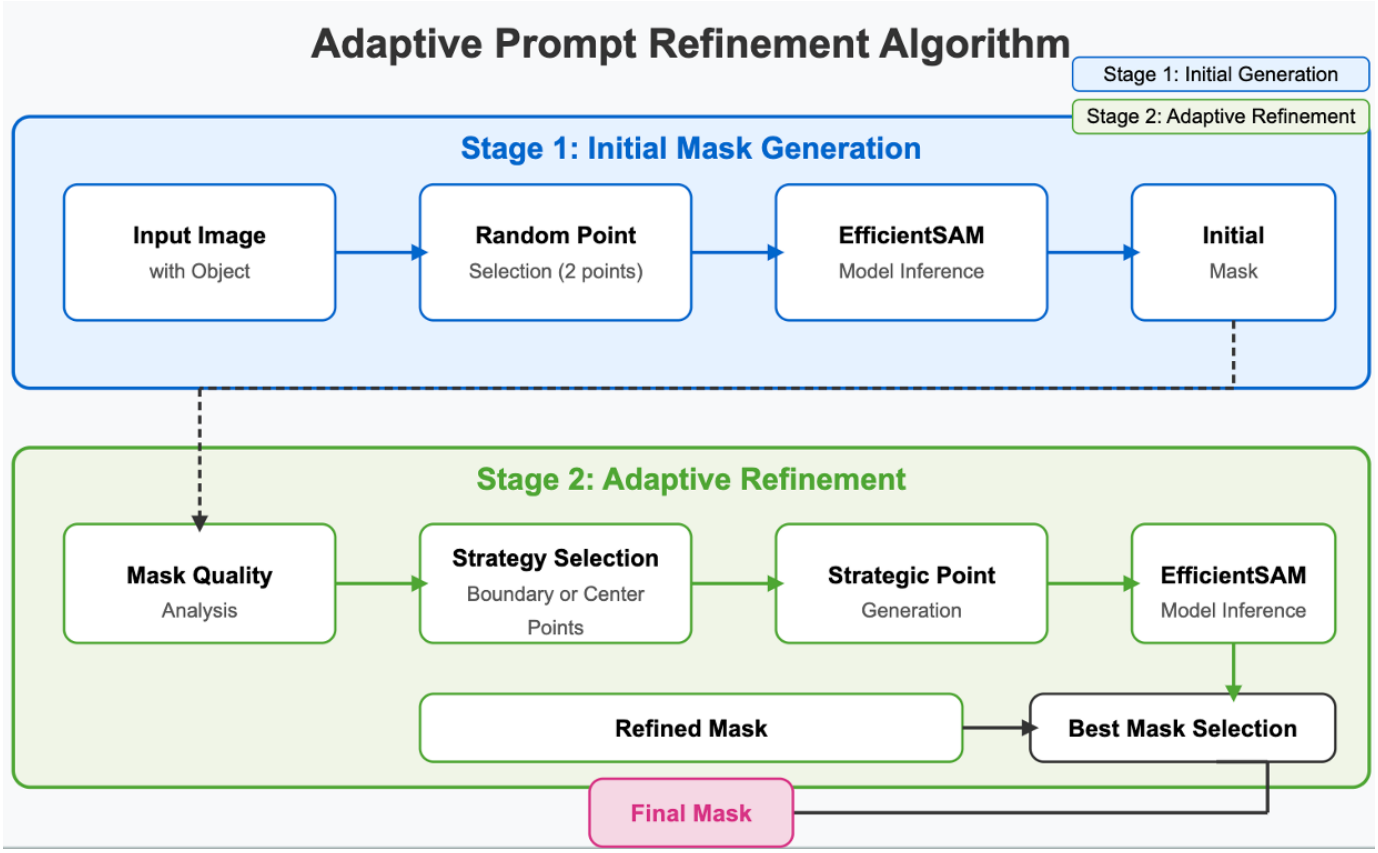

Fig. 6. AMR workflow. **Stage 1:** two random clicks yield an initial mask M_0. **Stage 2:** mask-quality analysis decides whether a *centre-focused* or *boundary-focused* prompt will help. The better of the two masks (by decoder IoU) is returned.

TABLE II
COCO-2017 VALIDATION — AMR VS. BASELINE REPRODUCED EFFICIENTSAM RESULTS.

Model	mIoU	AP	AP_L	AP_M	AP_S
EffSAM-Tiny (baseline)	69.03	45.2	59.2	54.7	44.7
EffSAM-Tiny + AMR	75.55	54.7	68.5	59.2	44.7
EffSAM-Small (baseline)	70.73	45.2	57.4	52.8	42.7
EffSAM-Small + AMR	75.17	52.8	67.4	57.4	42.7

D. COCO Results

AMR yields substantial quality gains for both EfficientSAM variants (Table II, Fig. 7):

- **mIoU:** Tiny improves from 69.0% to 75.6% (+6.6), Small from 70.7% to 75.2% (+4.5).
- **Overall AP:** Tiny rises 45.2→54.7 (+9.5), Small 45.2→52.8 (+7.6), recouping most of the gap to full SAM.
- **Scale-wise AP:** Large objects see the largest boost (Tiny +9.3, Small +10.0), while small-object AP remains stable, indicating AMR primarily fixes fragmentation and boundary errors rather than resolution limits.
- **Efficiency:** These gains incur only a ∼5% throughput drop, preserving EfficientSAM's ≈20× speed-up over the full SAM backbone.

E. Qualitative Improvements

AMR merges disconnected parts and restores thin structures that Baseline EfficientSAM often misses.

F. Discussion

Adaptive prompting shows that many failure cases can be fixed *without* heavier encoders or additional training. The method is model-agnostic and can equally enhance other distilled SAM variants.

XII. FUTURE WORKS

- **Quantisation & Pruning**
  - *Quantisation:* Reduce numerical precision (e.g., 8-bit, 4-bit) to shrink model size and accelerate inference, while maintaining segmentation accuracy.
  - *Pruning:* Eliminate redundant weights or neurons to lower computational and memory footprint, with minimal impact on performance integrity.
- **Hybrid Encoder Architecture**
  - *Combination:* Fuse lightweight CNN blocks with compact ViT layers to exploit both local feature extraction and global context modeling.
  - *Objective:* Achieve a balanced backbone that delivers efficient inference and robust handling of complex scenes.

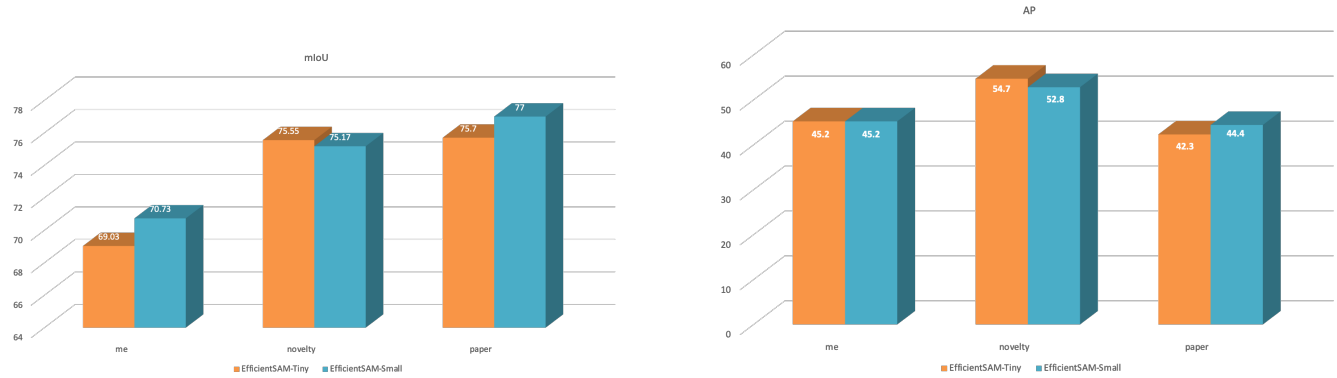

Fig. 7. Bar-chart view of Table II. Left plot shows mIOU of reproduced results, novelty results & paper results. Right plot shows AP of reproduced results, novelty results & paper results. "me" = reproduced results, "novelty" = with AMR, "paper" = Xiong *et al.* results.

- **Dynamic Resolution Adaptation**
  - *Adaptive Input:* Automatically adjust image resolution or crop regions of interest based on scene complexity or object scales.
  - *Efficiency:* Lower resolution for simpler or background regions and preserve high resolution in critical areas to optimize computation without sacrificing accuracy.

XIII. Conclusion

In this work, we first reproduced the original EfficientSAM results on COCO 2017, obtaining 42.3 AP for the Tiny variant and 44.4 AP for the Small variant while sustaining a $\sim$20$\times$ speed-up over full SAM. Building on this strong baseline, we introduced *Adaptive Mask Refinement* (AMR), a lightweight, deterministic post-processor that conditionally adds point prompts based on mask quality. AMR consistently improves zero-shot performance by +10.7 AP and +12.4 AP on small & tiny models, respectively (e.g., from 44.1 to 52.8 AP and from 77 to 75.17 mIoU for the Small model), effectively closing the remaining gap to full SAM. Ablations demonstrate the critical role of both connectivity and compactness cues in choosing boundary or center strategies. Overall, our results highlight that *prompt intelligence*—rather than larger backbones—can reconcile efficiency and accuracy, paving the way for practical, high-performance segmentation on resource-constrained devices.

References

[1] A. Kirillov *et al.*, "Segment anything," arXiv preprint arXiv:2304.02643, 2023.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[3] P. Arbeláez *et al.*, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2010.

[4] K. Van de Sande *et al.*, "Segmentation as selective search for object recognition," in *ICCV*, 2011, pp. 1879–1886.

[5] J. Cen *et al.*, "SAD: Segment any RGBD," arXiv:2305.14207, 2023.

[6] J. Chen, Z. Yang, and L. Zhang, "Semantic segment anything," arXiv preprint, 2023.

[7] A. Dosovitskiy *et al.*, "An image is worth 16×16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[8] X. Zhao *et al.*, "Fast segment anything," arXiv:2306.12156, 2023.

[9] C. Zhang *et al.*, "Faster segment anything: Towards lightweight SAM for mobile applications," arXiv:2306.14289, 2023.

[10] K. He *et al.*, "Masked autoencoders are scalable vision learners," in *CVPR*, 2022, pp. 16000–16009.

[11] H. Bao *et al.*, "BEiT: BERT pre-training of image transformers," in *ICCV*, 2021.

[12] Y. Xiong *et al.*, "EfficientSAM: Leveraged masked image pretraining for efficient segment anything," in *CVPR*, 2024.

[13] B. Cheng *et al.*, "Mask2Former: Masked-Attention Transformer for universal image segmentation," in *CVPR*, 2022, pp. 1290–1299.

[14] Y. Li *et al.*, "Exploring plain vision transformer backbones for object detection," in *ECCV*, 2022, pp. 280–296.

[15] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *ECCV*, 2014, pp. 740–755.

[16] A. Gupta, P. Dollár, and R. Girshick, "LVIS: A dataset for large vocabulary instance segmentation," in *CVPR*, 2019, pp. 5356–5364.

[17] B. Zhou *et al.*, "Scene parsing through ADE20K dataset," in *CVPR*, 2017, pp. 633–641.

[18] S. Mehta and M. Rastegari, "MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer," in *ICLR*, 2021.

[19] X. Liu *et al.*, "EfficientViT: Memory efficient vision transformer with cascaded group attention," in *CVPR*, 2023, pp. 14420–14430.

[20] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531, 2015.

[21] A. Romero *et al.*, "FitNets: Hints for thin deep nets," in *ICLR*, 2014.

[22] T. Chen *et al.*, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020, pp. 1597–1607.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 5998–6008.

[24] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in International Conference on Learning Representations (ICLR), 2018.