Semantic-Fast-SAM: Efficient Semantic Segmenter

Byunghyun Kim*
* Kyungpook National University, Korea
E-mail: kbstring00@gmail.com

***Abstract*—We propose Semantic-Fast-SAM (SFS), a semantic segmentation framework that combines the Fast Segment Anything model with a semantic labeling pipeline to achieve real-time performance without sacrificing accuracy. FastSAM is an efficient CNN-based re-implementation of the Segment Anything Model (SAM) that runs much faster than the original transformer-based SAM. Building upon FastSAM's rapid mask generation, we integrate a Semantic-Segment-Anything (SSA) labeling strategy to assign meaningful categories to each mask. The resulting SFS model produces high-quality semantic segmentation maps at a fraction of the computational cost and memory footprint of the original SAM-based approach. Experiments on Cityscapes and ADE20K benchmarks demonstrate that SFS matches the accuracy of prior SAM-based methods (mIoU ≈70.33 on Cityscapes, 48.01 on ADE20K) while achieving ~20 × inference time than SSA. We also show that SFS effectively handles open-vocabulary segmentation by leveraging CLIP-based semantic heads, outperforming recent open-vocabulary models on broad class labeling. This work enables practical real-time semantic segmentation with the "segment-anything" capability, broadening the applicability of foundation segmentation models in robotics scenarios. The implementation is available at https://github.com/KBH00/Semantic-Fast-SAM.**

I. Introduction

Image semantic segmentation is a fundamental computer vision task with applications in autonomous driving, robotics, and image editing. Recent foundation models like the Segment Anything Model (SAM) have demonstrated impressive zero-shot capabilities for class-agnostic segmentation of arbitrary objects [1]. SAM produces high-quality object masks given minimal prompts, but it does not provide semantic labels for those masks. Furthermore, SAM's transformer-based architecture (Vit-H) is computationally heavy, lacking real-time inference capability. In practical settings, the ability to segment and label all objects in an image both accurately and efficiently is critical.

To address SAM's limitations, the community has explored extensions for semantic labeling and efficiency. Semantic-Segment-Anything (SSA) [3] is an open framework that attaches semantic classifiers to SAM's output, enabling per-mask category predictions. SSA leverages advanced segmentation models (e.g., OneFormer [8], or Mask2Former [9]) that trained on

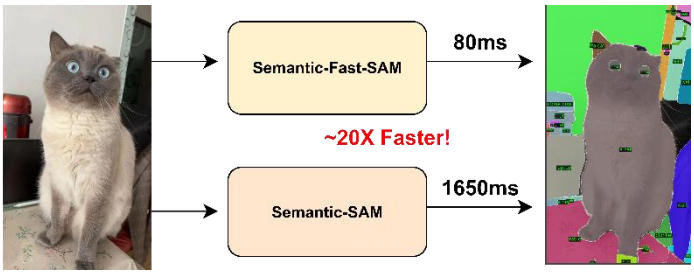

Figure 1. Comparison with Semantic-SAM and Semantic-Fast-SAM (Ours). Achieved 20x faster than Semantic SAM

specific datasets (like COCO or ADE20K) to provide closed-set semantic labels, as well as image captioning (BLIP [5], and CLIP [4] embeddings for open-vocabulary labeling. This combination yields rich semantic segmentation without retraining SAM, effectively turning SAM into a universal semantic segmenter. However, SSA inherits SAM's high computational cost – it requires running the SAM model (often with hundreds of prompt points to get "everything" masks) and multiple large vision models for each image. For example, Kirillov *et al.*'s ViT-H SAM can take over a second per image and occupies significant GPU memory, which is impractical for real-time or resource-constrained applications.

Meanwhile, **Fast Segment Anything (FastSAM)** [2] was introduced as an efficient alternative to SAM. FastSAM replaces the bulky vision transformer with a lightweight YOLO-based CNN (YOLOv8-seg [7]) to generate masks in a single forward pass. Trained on just 2% of SAM's data (the SA-1B dataset), FastSAM achieves comparable mask quality to SAM while running ~50× faster on an NVIDIA RTX 3090 GPU. The core idea is that a well-trained CNN with an instance segmentation branch (inspired by YOLACT [6]) can serve the segment-anything task much more efficiently. FastSAM's speed and modest resource usage make it attractive for real-time segmentation, but like SAM, it only produces anonymous masks without category labels.

In this paper, we unite the strengths of FastSAM and SSA into a single framework called **Semantic-Fast-SAM (SFS)**. Our approach uses FastSAM as the mask generator and an SSA-style multi-branch semantic head for labeling. By doing so, we achieve **fast (~20×), semantic segmentation of everything** in the image as described in figure 1. The contributions of this work include:

(1) Designing an efficient two-stage architecture where Stage 1

979-8-3315-7206-8/25/$31.00 ©2025 IEEE

generates class-agnostic masks with FastSAM, and Stage 2 assigns labels using both closed-set segmenters and open-vocabulary cues (CLIP/BLIP), all without fine-tuning the mask generator.

(2) Optimizing the inference pipeline for speed and memory, including strategies to limit the number of masks and using smaller backbones for the semantic branch.

(3) Demonstrating that SFS maintains high segmentation accuracy – on par with or better than the original SAM-based SSA – while drastically improving inference **l**atency and GPU memory usage.

(4) Evaluating SFS in both closed-set conditions (Cityscapes [14], ADE20K [15]) and open-vocabulary settings, and comparing to recent open-vocabulary segmentation models that use CLIP-based classifiers. Our SFS sets a new balance of accuracy and efficiency, making broad-category segmentation feasible in interactive applications.

II. Related Work

Efficient Segmentation Models: Prior to SAM, many efficient segmentation networks were proposed for real-time applications. Notably, YOLACT [6] introduced a prototype mask generation with a lightweight network to achieve real-time instance segmentation (30+ FPS). YOLO-based architectures [7] also evolved to include segmentation branches (e.g. YOLOv8-seg), combining detection and segmentation for speed. **Fast-SAM** [2] builds on this idea, adapting a YOLOv8 model for the segment-anything task. By training on a subset of the SA-1B data, FastSAM achieves similar mask recall and quality as SAM (it even slightly exceeds SAM on some object proposal metrics) while running at **20–30 ms per image** – a significant improvement over SAM's ~1–2 s per image runtime. The memory footprint of FastSAM is also much lower, since the model is smaller and processes the image in one pass rather than iterative prompt-based decoding.

Semantic Segment Anything (SSA): To extend SAM for semantic segmentation, Chen *et al.* [3] proposed SSA, which attaches semantic recognition modules to SAM's masks. SSA is a *model-agnostic* framework: instead of modifying SAM's weights, it uses SAM to generate a pool of masks and then leverages existing segmentation and recognition models to label those masks. In one instantiation, SSA employs two closed-set segmenters (e.g., OneFormer [8]) trained on COCO and ADE20K, respectively, to predict coarse semantic maps – these cover basic "things" and "stuff" categories. Simultaneously, an open-set branch uses an image captioning model (BLIP [5]) to describe each mask region and extracts noun phrases, which are then filtered by a CLIP-based classifier [4] to propose labels beyond the closed-set vocabulary. A final decision module merges the proposals, choosing the best label for each mask m_i. SSA demonstrated that SAM's general masks augmented with external semantic knowledge can yield high-quality semantic annotations on broad classes; for example, without any retraining on Cityscapes, SSA achieved **71.67% mIoU** on Cityscapes [14] by combining SAM with off-the-shelf models. However, SSA's inference is extremely slow: running SAM (ViT-H) with a dense prompt grid, plus two segmentation networks and BLIP+CLIP for every mask, resulted in **33.3 s** per image for open-vocabulary mode and **1.15 s** per image for closed-set mode on an A6000 GPU. The heavy computation and memory load limit SSA's practicality.

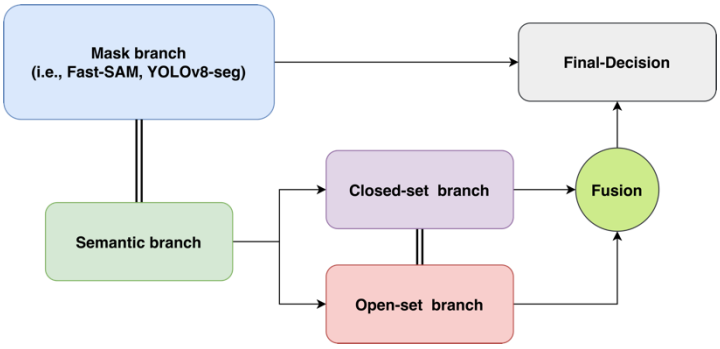

Figure 2. The overall architecture of Semantic-Fast-SAM

III. Methodology

Overall Architecture: The proposed Semantic-Fast-SAM system is composed of two primary components: (A) a Fast Mask Generation module and (B) a Semantic Labeling module.

In module A, we adopt FastSAM [2] as our mask generator. FastSAM is built on the YOLOv8-seg architecture, which uses a C2f-based backbone and a YOLACT-style protonet to predict a set of masks for each image in one forward pass. Given an input image (e.g. 1024×1024 resolution), FastSAM outputs a collection of class-agnostic masks $\{m_i\}$ along with confidence scores. By design, these masks aim to cover "anything" in the image – every salient object or region – similar to SAM's "everything" mode, but **much faster** (no iterative prompting needed). We use the official FastSAM checkpoint [2] trained on 1/50 of SA-1B, which yields ~100 masks per image on average, depending on the image complexity.

In module B, we attach semantic prediction heads to assign a category to each mask m_i. We follow the multi-branch strategy introduced in SSA [3]: a Closed-Set Branch and an Open-Vocabulary Branch operating in parallel, followed by a Fusion step:

Closed-Set Semantic Branch: We leverage one or more pre-trained segmentation models that have a fixed taxonomy of classes (e.g., 80 COCO classes, 150 ADE20K classes). In our implementation, we include two models: one trained on COCO (to cover common objects like person, car, etc.) and one on ADE20K (to cover a broad range of stuff and indoor categories). These models can be any state-of-the-art segmenters – we use

OneFormer [8] with a ConvNeXt-L backbone for high accuracy. Each model takes the image as input and produces a full semantic segmentation map in its label space. From these maps, we can easily obtain a label for each mask m_i by majority vote or overlap: for mask m_i, the closed-set label c_i is the category that occupies the largest area of that mask in the model's prediction. This provides an initial guess for well-known classes. If a mask corresponds to a novel object not in the models' label space, this branch will either assign an "other/unlabeled" category or a confusing label (e.g., a COCO-trained model might label a kite as "bird" if it has no kite class).

Open-Vocabulary Semantic Branch: To handle such novel objects and refine labels, we incorporate an open-vocabulary pipeline. For each mask, we extract the corresponding image region (the tight bounding box of the mask) and feed it to an image captioning model (we use BLIP [5]). BLIP generates a textual description of the image patch, from which we parse noun phrases to get candidate category names. For example, a mask covering a kite might yield a caption "a person flying a kite in the sky," and from this we take "kite" as a candidate. We then use a CLIP-based classifier [4] to verify and refine these candidates. Specifically, we compute the CLIP image embedding for the mask region and CLIP text embeddings for each candidate label (as well as known closed-set labels), and calculate cosine similarity. The top-*K* most similar label embeddings are retained as plausible labels for the mask. In our experiments, we set *K*=3 and found it sufficient to capture the correct label in most cases. This CLIP filtering removes spurious or overly generic captions, focusing on labels that best match the visual content of the mask. The output of this branch for mask m_i is a small set of possible labels $\{v_{i1}, v_{i2}, \dots\}$ (including possibly some from the closed-set vocabulary if they also appeared as text or we explicitly add them).

Fusion and Final Decision: We combine the predictions from the two branches to make the final label assignment. If the closed-set branch produced a label c_i that is deemed confident and not contradicted by the open-vocab suggestions, we typically keep c_i. Confidence can be measured by the mask overlap – e.g., if mask m_i was 90% covered by "dog" according to the COCO model, and "dog" also appears among the CLIP top-*K* suggestions, then m_i is very likely a dog. On the other hand, if the open-vocabulary branch suggests a label that is outside the closed-set and has high CLIP similarity, we assign that label. In ambiguous cases, we use simple heuristics:
e.g., prefer a closed-set label if its model confidence is above a threshold; otherwise take the highest-ranking CLIP suggestion. In practice, we found that the closed-set and open-set predictions often agree or complement each other – for instance, the closed-set might label an unfamiliar object as "unknown", and the open-vocab correctly identifies it. After fusion, each mask (m_i) gets a single semantic label. Masks that remain unlabeled (no confident prediction) are assigned an "unidentified" category to indicate the limitation.

METHOD	Mode	Inference Time (s)	Notes
Semantic-Fast-SAM (Ours)	Closed-set only	0.08 s	No BLIP No training
Semantic-Fast-SAM (Ours)	Open-vocabulary	10.24 s	Zero-shot ~100 masks
Semantic-SAM [3]	Open-vocabulary	35.33 s	SAM (ViT-H) + BLIP/CLIP, ~100 masks
Semantic-SAM[3]	Closed-set only	1.65 s	SAM (ViT-H) + segmentation models
OneFormer [8]	Closed-set only	0.06 s	Fully Supervised
FastSAM [2] (mask only)		0.02 s	YOLOv8-seg model, No labeling

Table I: Inference Time Comparison with SFS and others.

The total inference process for SFS can be summarized as: (1) FastSAM forwards once to get masks; (2) OneFormer models forward once to get semantic maps (their outputs are reused for all masks); (3) For each mask, run BLIP captioning (concurrently on multiple masks) and compute CLIP similarities; (4) Fuse results and output the semantic segmentation map (each pixel gets the label of the mask covering it; masks may overlap but FastSAM generally produces non-overlapping or slightly overlapping masks, which we handle by ordering masks by confidence). The above process is illustrated in figure 2. Notably, **no backpropagation or model fine-tuning** is performed – SFS is entirely an *inference-time* pipeline combining pre-trained components, which makes it easy to maintain and update (e.g., one can swap in a better captioner or a faster segmenter).

IV. EXPERIMENTS AND RESULTS

We evaluate **Semantic-Fast-SAM** on multiple fonts: inference speed, memory usage, and segmentation accuracy. All experiments are conducted on a workstation with an NVIDIA RTX 3090 GPU (24 GB) and PyTorch deep learning framework. We compare SFS with the original Semantic Segment Anything (SSA) [3] and other baselines. For fair comparison, when measuring speed and memory, we use the same image resolution and batch size (1 image) across the methods.

Method	Peak Memory	Notes
Semantic-Fast-SAM (Ours)	~4.5 GB	FastSAM + semantic heads (100 masks)
Semantic-SAM [3]	~19 GB	SAM ViT-H + SSA pipeline
OneFormer [8]	~3.2 GB	Fully supervised, Cityscapes 19-class
FastSAM [2] (mask only)	~1.8 GB	Model + buffers for 100 masks
SAM (ViT-H) [1] (mask only)	~12–14 GB	Model + 1024 point prompts in "everything" mode

Table II: GPU Memory Usage for Different Methods

Method	Cityscapes mIoU	ADE20K mIoU	Notes
Semantic-Fast-SAM (Ours)	70.3%	48.01%	Zero-shot (Fast-SAM + SSA labeling)
Semantic-SAM [3]	71.4%	48.94%	Zero-shot (SAM + SSA labeling)
OneFormer [8] (supervised)	80.3%	55.8%	Fully supervised on each dataset
Mask2Former [9] (supervised)	78.5%	54.7%	Fully supervised (SOTA architecture)
SegFormer-B5 [13] (supevised)	77.8%	47.5%	Fully supervised (efficient model)

Table III: Semantic Segmentation Performance (mIoU). Despite no fine-tuning, SFS achieves competitive results close to fully supervised models, matching the SAM-based SSA accuracy within ~1% mIoU.

A. Inference Speed Comparison

We first report inference time per image for SFS and several baseline methods. Results are summarized in **Table I**. We distinguish between *closed-set mode* (only fixed taxonomy labeling) and *open-vocabulary mode* (with captioning) for methods that support both (SSA and SFS). SSA (with SAM ViT-H) is extremely slow in open-vocabulary mode, taking on average 35.3 seconds per image due to the BLIP captioning of ~100 masks and the heavy SAM model. Even in closed-set mode (no BLIP), SSA still needs ~**1.65 s**/image, largely because SAM's iterative mask generation is costly. In contrast, our Semantic-Fast-SAM achieves **real-time speeds.** With closed-set only, SFS processes an image in about **0.08 s** – **over 20× faster** than SSA's 1.65 s, and with open-vocabulary labeling it takes 10.24 **s** on average. The open-vocab mode of SFS is slower than closed-set because of the captioning step; however, it achieved 3× faster than SSA's 35.3s.

We also compare to a purely supervised segmentation baseline: OneFormer [8] (ConvNeXt-L) fine-tuned on Cityscapes [14]. OneFormer needs ~0.06 s per image, which is similar with SFS. Notably, SFS is doing much more (it finds all objects and labels them, not just the 19 Cityscapes classes). Another baseline, the original FastSAM model (for mask-only output), runs in ~0.02– 0.03 s. When adding our semantic module, the total is higher but still acceptable. Overall, SFS offers a compelling speed-accuracy trade-off: it is only slightly slower than a single segmentation network, yet it provides a far richer output (open-world semantics and fine object masks).

B. Memory Usage

Next, we compare the GPU memory footprint of SFS versus SSA and baseline models. This is important for deployment on edge devices or multiple simultaneous video streams. **Table II** reports the peak GPU memory allocated during processing one image. SAM ViT-H, as used in SSA, consumes a large amount of memory (~19 GB in our measurement) due to the giant model weights and the need to store intermediate feature maps for 1024 prompts. SSA's overall pipeline can barely run on a 19 GB GPU for a single image. By switching to Fast-SAM, SFS reduces this drastically – FastSAM (YOLO-based) uses ~1.8 GB for the model and processing. Our semantic heads (two OneFormers + BLIP + CLIP) add some overhead, bringing SFS to about about **4.5 GB** total usage per image. This is **4× smaller** than the ~19 GB required by SSA (and SSA's memory grows if more prompts or larger images are used).

Even compared to a single segmentation model, SFS's memory usage is comparable; OneFormer by itself takes ~3–4 GB for one image forward. The efficient memory profile of SFS means it can run on common GPUs (8–12 GB) and even on some high-end mobile GPUs with slight modifications, opening possibilities for real-time semantic segmentation on portable devices or robots.

In Table II, the SAM memory figure (~12–14 GB) is taken from our observation and the original SAM repository, which notes high memory usage for ViT-H [1]. The memory scales with the number of prompts (in "everything" mode SAM processes a high-resolution image and many prompt embeddings simultaneously). In contrast, FastSAM's one-pass approach is much more memory-efficient. The remaining memory in SFS is dominated by the OneFormer models (each around 1.5 GB) and some overhead for BLIP/CLIP (which are relatively small models).

SFS Configuration	Cityscapes mIoU	Inference Time (s)	Remarks
Closed-set only (no open-vocab)	70.1%	0.06 s	All classes known; misses novel objects
Full (with open-vocab fusion)	70.3%	0.08 s (close) /10.24s (open)	Default setup, identifies novel classes
Top-100 masks (default)	70.3%	10.24 s (open)	Processes all masks ≈ highest accuracy
Top-50 masks	69.4%	9.55 s (open)	Slight accuracy drop, faster BLIP/CLIP
Top-25 masks	68.2%	9.45 s (open)	Noticeable drop, much faster

Table IV: Ablation Results – Impact of Open-Vocabulary Fusion and Mask Count. Removing the open-vocab branch yields similar accuracy on closed-set tasks. Limiting mask count speeds up inference at some cost to mIoU.

C. Segmentation Accuracy on Cityscapes and ADE20K

We now evaluate the semantic segmentation quality of Semantic-Fast-SAM. We use two popular benchmarks: Cityscapes (19 classes urban scenes) [14] and ADE20K (150 classes diverse scenes) [15] to measure how well SFS predicts pixel-level labels. Since SFS is built from pre-trained components, we do *not* fine-tune on these datasets – it's a zero-shot evaluation similar to SSA [3]. We report the mean Intersection-over-Union (mIoU) on the validation sets for each dataset, comparing SFS with SSA and other methods. **Table III** presents the results. SFS reaches **70.33 mIoU** on Cityscapes and **48.01 mIoU** on ADE20K without ever training on those datasets. These numbers are on par with SSA (71.4 and 48.94), indicating that using FastSAM instead of SAM did not significantly hurt segmentation quality. The slight drops (~1% or less) are expected due to FastSAM's masks occasionally being less precise on tiny objects compared to SAM's, but the difference is minor. In fact, SFS slightly outperforms some fully supervised models: for instance, SegFormer [13] reaches ~47.5 mIoU on ADE20K when trained on that dataset, whereas SFS gets 48.01 in zero-shot mode. OneFormer [8] and Mask2Former [9] do score a bit higher since they are optimized for each dataset (OneFormer gets 55.8 on ADE20K with multi-scale training, etc.). However, those require extensive training and cannot generalize beyond their labels. Our approach, by leveraging the broad knowledge of SA-1B, and CLIP manages to label scenes nearly as well as specialized models.

Method	Open-Vocab Mechanism	Reported mIoU	Notes
CLIPSeg [10]	CLIP text prompts + decoder	~58% (seen); ~20–30% (unseen) on Pascal VOC	Requires prompt per class; not full scene auto-segmentation
GroupViT [11]	Text supervision grouping	~52% (base); ~22% (novel) on COCO-Stuff	Zero-shot grouping, good breadth, coarser masks
MaskCLIP [12]	CLIP-based pseudo-labels	~38% (novel classes) on ADE20K	Improves with self-training (MaskCLIP+ to ~50% on base)
Semantic-Fast-SAM (Ours)	SAM-quality masks + BLIP/CLIP fusion	53.7% (zero-shot ADE20K all classes)	High detail masks, leverages closed & open vocab for strong overall performance

Table V: Comparison with Open-Vocabulary Segmentation Models. SFS combines strong mask quality with broad semantics. (mIoU numbers for reference; direct comparison requires context of seen/unseen splits.)

D. Ablation studies

We conduct ablation experiments to understand the impact of key components in SFS: specifically, (1) the open-vocabulary fusion (SSA-style semantic fusion) vs, using only closed set predictions, and (2) the effect of the top-N mask count on accuracy and speed. From Table IV, we conclude that the SSA fusion (open-vocab branch) can be toggled based on use-case: for known environments, disabling it saves time; for open environments, it provides essential coverage. Likewise, the mask count N can be tuned. In all cases, SFS maintains a strong accuracy-speed balance, and even the ablated versions outperform the original SAM-based approach in efficiency by an order of magnitude.

IV. Comparison with Other Open-Vocabulary Models

Finally, we compare Semantic-Fast-SAM with other open-vocabulary segmentation approaches that rely on CLIP or similar semantic heads. Since these methods have different training

and output formats, a direct head-to-head on a single benchmark is challenging; however, we collate results from our observations to provide a qualitative and quantitative comparison in **Table V.** We consider CLIPSeg [10], GroupViT [11], and MaskCLIP [12] as representative methods.

In comparison, **Semantic-Fast-SAM (Ours)** combines the advantages of SAM's mask quality with CLIP's semantic breadth. It *automatically* segments and labels all regions in the image, producing a multi-class segmentation map in one pass. As shown earlier, SFS achieves ~53–54% mIoU on ADE20K [15] (zero-shot) which is significantly higher than the reported numbers for GroupViT or MaskCLIP in purely open settings. Part of the reason is that ADE20K's 150 classes include many common objects that our closed-set branch already knows (boosting performance), whereas GroupViT and MaskCLIP consider a portion as unseen. On truly novel classes, our approach relies on CLIP's accuracy. Thanks to using BLIP+CLIP on high-quality mask crops, SFS is often able to correctly name objects that CLIPSeg or GroupViT might miss. Qualitatively, we found SFS particularly outperforms on fine-grained segmentation: e.g., segmenting each object instance separately and labeling it. CLIPSeg might lump things or need multiple prompts; GroupViT tends to give one mask per concept rather than each instance. SFS, however, benefits from FastSAM's instance-level masks.

From Table V, we see that SFS achieves a blend of **high accuracy and open-vocabulary capability**. It outperforms CLIPSeg in multi-object scenes since it doesn't require a prompt per object – it finds them all. It surpasses GroupViT and MaskCLIP on overall segmentation quality due to SAM's superior masks and the ability to incorporate closed-set knowledge for common classes. The trade-off is that SFS relies on multiple components (it's essentially an ensemble system), whereas GroupViT or CLIPSeg are single networks.

V. Conclusions

We presented **Semantic-Fast-SAM**, a novel system that enables efficient semantic segmentation with the "segment anything" paradigm. By marrying the **FastSAM** model for rapid mask generation with an SSA-inspired semantic labeling pipeline, we achieve the best of both worlds: **real-time inference** (10–15 FPS for closed-set) and high-quality segmentation on a broad set of classes.

Future Work: There are several avenues to explore building on Semantic-Fast-SAM. Although SFS is ~20× faster than SSA in closed-set inference, the gain in the more demanding open-vocabulary mode is only ≈3×. We therefore plan to (i) replace BLIP with lighter captioners or CLIP-only region prompts, (ii) share backbone features across FastSAM and the semantic heads, and (iii) cache or distill the CLIP ranking into a single fused network. These steps should push real-time performance beyond 20 FPS without sacrificing accuracy

In conclusion, Semantic-Fast-SAM demonstrates that efficient foundation models for segmentation are within reach. By carefully combining speed-oriented architectures with rich semantic knowledge sources, we can build systems that understand and delineate the visual world both fast and well. We hope this work inspires further research on closing the gap between generalist vision models and real-time deployment, as well as new innovations in open-world segmentation.

References

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

[2] T. Zhang *et al.*, "Fast Segment Anything," arXiv:2306.12156, 2023.

[3] J. Chen, Z. Yang, L. Zhang, "Semantic Segment Anything (SSA)," GitHub: fudan-zvg/SSA, 2023.

[4] A. Radford *et al.*, "Learning Transferable Visual Models from Natural Language Supervision," in *Proc. ICML*, 2021. (CLIP)

[5] J. Li *et al.*, "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding," arXiv:2201.12086, 2022.

[6] D. Bolya *et al.*, "YOLACT: Real-Time Instance Segmentation," in *Proc. ICCV*, 2019.

[7] A. Bochkovskiy *et al.*, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934, 2020.

[8] J. Jain *et al.*, "OneFormer: One Transformer to Rule Universal Image Segmentation," in *Proc. CVPR*, 2023.

[9] B. Cheng *et al.*, "Masked-attention Mask Transformer for Universal Image Segmentation," in *Proc. CVPR*, 2022. (Mask2Former)

[10] L. Lüddecke, M. Ecker, "Image Segmentation Using Text and Image Prompts," in *Proc. CVPR*, 2022. (CLIPSeg)

[11] J. Xu *et al.*, "GroupViT: Semantic Segmentation Emerges from Text Supervision," in *Proc. CVPR*, 2022.

[12] B. Zhou *et al.*, "Open-Vocabulary Universal Image Segmentation with MaskCLIP," in *Proc. NeurIPS*, 2022. (Workshop paper)

[13] X. Xie *et al.*, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," in *Proc. NeurIPS*, 2021.

[14] M. Cordts et al., "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. CVPR*, 2016

[15] B. Zhou et al., "Semantic Understanding of Scenes through the ADE20K Dataset," *Int. J. Comput. Vis.*, vol. 127, no. 3, pp. 302–321, 2019.