arXiv:2502.02763v3 [cs.CV] 10 Jul 2026

Looking Locally: Object-Centric Vision Transformers as Foundation Models for Efficient Segmentation

Manuel Traub [1] **Martin V. Butz** [1]

Abstract

Current state-of-the-art segmentation models encode entire images before focusing on specific objects. This wastes computational resources. We introduce FLIP (Fovea-Like Input Patching), a parameter-efficient vision model that realizes object segmentation through biologically-inspired top-down attention. FLIP selectively samples multi-resolution patches centered on objects of interest from the input. As a result, it allocates high-resolution processing to object centers while maintaining coarser peripheral context. This off-grid, scale-invariant design enables FLIP to outperform META's Segment Anything models (SAM, SAM2 and fast variants) by large margins: With more than 440× fewer parameters, FLIP-Tiny (0.51M parameters) reaches a mean IoU of 79.90% while SAM2-L reaches 75.87% IoU (224.45M parameters). FLIP-Large even achieves 83.26% mean IoU (96.6M parameters), still running about 2× faster than SAM2-L. We evaluate on six benchmarks in total. In five established benchmarks (Hypersim, KITTI-360, OpenImages, COCO, LVIS) FLIP consistently outperforms SAM and various variants of it. In our novel ObjaScale dataset, which stress-tests scale invariance with objects ranging from 0.0001% up to 25% of the image area, we show that FLIP segments even very small objects accurately, where existing models fail severely. FLIP opens new possibilities for real-time, object-centric vision and offers much higher energy efficiency. We believe that FLIP can act as a powerful foundation model, as it is very well-suited to track objects over time, for example, when being integrated into slot-based scene segmentation architectures.

1. Introduction

Object-centric models have emerged as a powerful paradigm for structured perception in visual tasks. They offer the potential to represent complex scenes in a more interpretable and compositional manner. While traditional architectures such as Convolutional Neural Networks (CNNs) (Liu et al., 2022) and Vision Transformers (ViTs) (Dosovitskiy et al., 2020) have demonstrated impressive performance on large-scale datasets, they often lack the nuanced object-level understanding required for robust scene parsing. Furthermore, these models typically require massive amounts of labeled data and exhibit vulnerabilities to adversarial perturbations.

Recent advances in object-centric learning, such as Slot Attention (Locatello et al., 2020), have sought to address these challenges by enabling the model to discover and represent objects within a scene as distinct entities. Models like

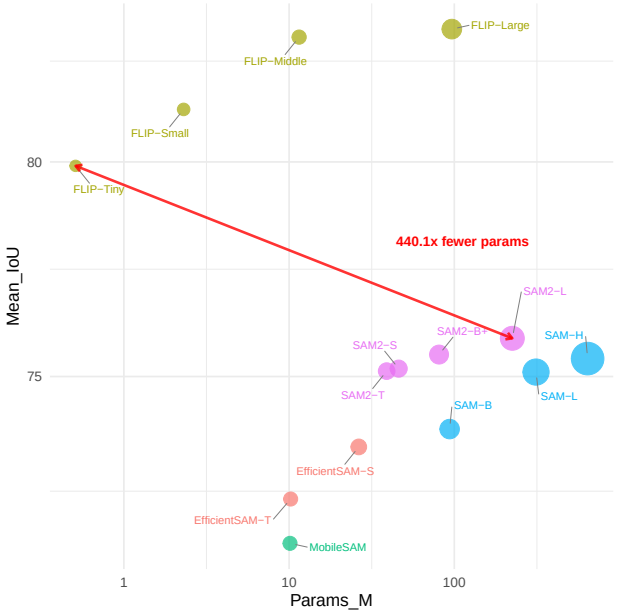


Figure 1. Comparison of mean IoU across six diverse datasets (Hypersim, KITTI-360, OpenImages, COCO, LVIS, ObjaScale) plotted against model size (in millions of parameters). FLIP variants consistently outperform SAM variants while using several orders of magnitude fewer parameters.

Project Page: https://cognitivemodeling.github.io/FLIP/
[1]Neuro-Cognitive Modeling Group, University of Tübingen, Germany. Correspondence to: <manuel.traub@uni-tuebingen.de>.

Accepted at ICML 2026.

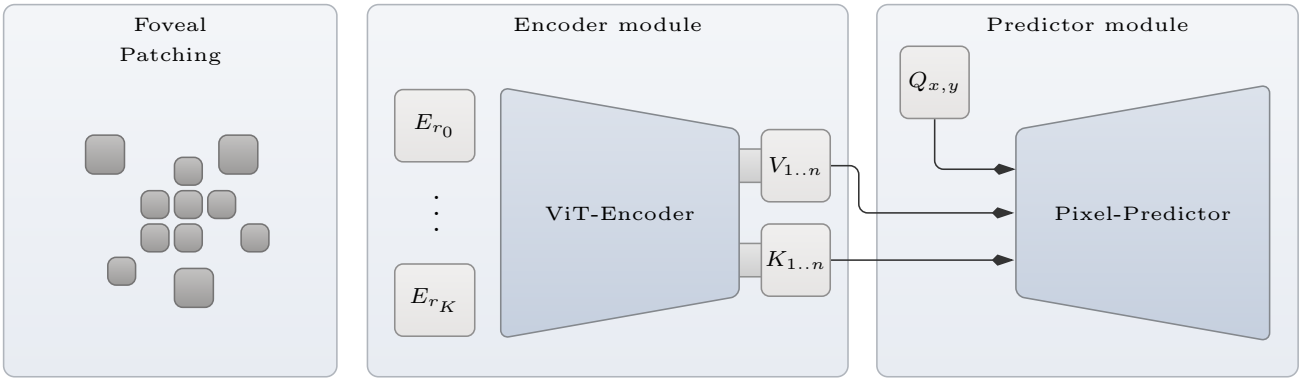


Figure 2. Overview of the FLIP architecture. The Foveal Patching module dynamically samples multi-resolution patches centered around objects of interest. These patches are embedded into a unified latent space using resolution-specific Patch Embedding Modules ($E_{r_0}, \ldots, E_{r_K}$). The Vision Transformer Encoder processes the embedded patches, generating keys $K_{1..n}$ and values $V_{1..n}$. The Pixel-Predictor performs attention over queries derived from pixel coordinates $Q_{x,y}$, enabling instance segmentation with pixel-level precision.

SAVi++ (Elsayed et al., 2022), VideoSAUR (Zadaianchuk et al., 2024), and Loci (Traub et al., 2023; 2024a;b) have advanced the state-of-the-art in unsupervised object-centric learning. They are able to disentangle objects from complex backgrounds and track their identities over time. Still, these approaches struggle to scale effectively to more complex, real-world data.

In contrast, the Segment Anything (SAM) model and its successor SAM2 (Kirillov et al., 2023; Ravi et al., 2024) have introduced a paradigm shift in object-centric learning. SAM learns from a vast array of diverse data that is segmented by a two stage segmentation process. First, a powerful transformer-based foundational model encodes the complete image. Second, a query-based focusing mechanism specifies which object to segment. Only this second mechanism targets one image area or object and leads to the production of the targeted output mask. SAM2 marks the state-of-the-art in object segmentation tasks. However, despite its impressive performance, SAM models have their limits. First, the transformer-based encoder requires very large computational resources. Second, the encoder backbone encodes the complete image, potentially wasting processing resources, particularly when small objects are to be segmented.

Derivative model variants like EfficientSAM, MobileSAM, and FastSAM (Xiong et al., 2023; Zhang et al., 2023; Zhao et al., 2023) address the former limitation. In our evaluation of single-object segmentation performance across multiple datasets, EfficientSAM employs SAMI (SAM-leveraged masked image pretraining), training lightweight ViT encoders to reconstruct features from SAM's ViT-H encoder, and achieves 72.29% mean IoU with only 10.22M parameters (EfficientSAM-T) compared to SAM2-L's 75.87% IoU with 224.45M parameters. MobileSAM applies decoupled distillation, replacing SAM's ViT-H encoder (632M parameters) with TinyViT (5M parameters) (Wu et al., 2022), achieving 71.33% mean IoU with an over 60× parameter reduction and ∼21ms runtime per image. FastSAM replaces the first-stage transformer-based encoder in SAM with a convolutional ANN (CNN) that is pre-trained to segment full images. It achieves a runtime per image of under 10ms but exhibits significantly reduced performance with only 44.58% mean IoU (FastSAM-s).

At this point, nearly all segmentation techniques, including

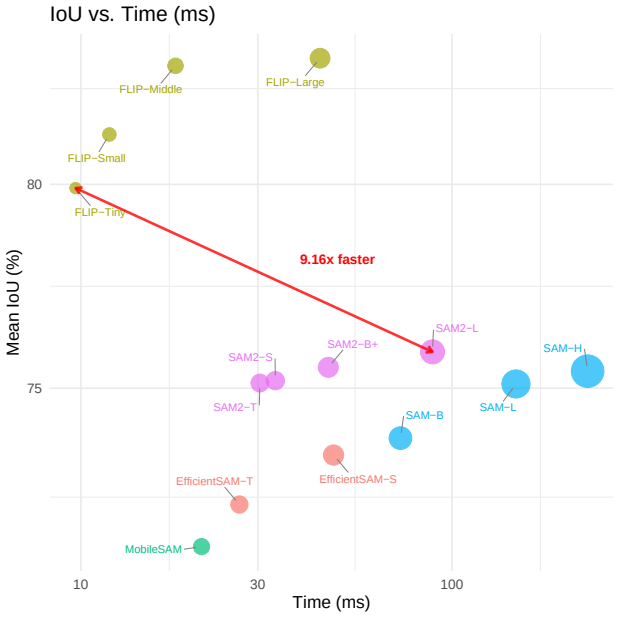


Figure 3. Comparison of mean IoU across six diverse datasets (Hypersim, KITTI-360, OpenImages, COCO, LVIS, ObjaScale) plotted against inference time (in milliseconds). FLIP variants consistently outperform SAM variants while being significantly faster.

SAM, SAM2 and its efficient variants, rely on a full image encoder. This is also the case for most object-centric models, such as Slot Attention, VideoSAUR, and related work (Locatello et al., 2020; Elsayed et al., 2022; Singh et al., 2022; Zadaianchuk et al., 2024), which first encode the entire image before assigning information to slots. The challenge to computationally efficiently segment and track small but potentially high-resolution objects across diverse and complex scenes remains. Here, we introduce a prompted vision transformer-based foundational model for efficient object-centric segmentation that utilizes a local, pixel-patch-based image encoder to generate accurate masks.

In particular, we introduce FLIP: a fovea-like input patching approach that is integrated in an object-centric, off-grid vision framework. FLIP dynamically adapts its processing pipeline to the object's size and spatial characteristics. It ignores currently irrelevant image subregions and focuses on critical regions with a flexible, multi-resolution approach. Our key contributions are:

- **Off-Grid, Scale-Invariant Object Encoding**: We introduce a fovea-inspired pixel-based patch sampling method that directly encodes image regions off-grid, adaptively focusing on objects of interest in a multi-resolution fashion. This scale-invariant approach is robust to large variations in object size and image resolution. It enables the detailed encoding of very small objects even in high-resolution scenes.
- **Exceptional Parameter and Computational Efficiency**: FLIP achieves superior segmentation performance while using orders of magnitude fewer parameters than existing state-of-the-art models. With lightweight encoders and efficient processing, FLIP provides significant energy savings and faster inference times, making it suitable for real-time applications and resource-constrained environments.
- **State-of-the-Art Segmentation Performance with High Parameter Efficiency**: Despite using significantly fewer parameters compared to state-of-the-art models from the SAM family, FLIP achieves superior segmentation accuracy on standard benchmarks such as Hypersim, KITTI-360, OpenImages, COCO and LVIS.
- **Superior performance on novel dataset *ObjaScale*** We introduce *ObjaScale*, a high-resolution evaluation benchmark with Blender-rendered objects at varying scales on real-world High Dynamic Range Image (HDRI) backgrounds, designed to stress-test scale invariance. As in the other datasets, FLIP outperforms all SAM variants on ObjaScale and shows that it can segment even rather tiny objects accurately.

Algorithm 1 Fovea-like patch sampling

Require: Image $I \in [0,255]^{H\times W\times C}$, Gaussian query (μ, Σ), patch sizes $\{p_i\}_{i=1}^K$, patch budget N, coverage $c \in [0.1, 2]$, overlap threshold $\tau \in [0, 4]$
Ensure: Per-resolution patch buffers $\{P_i\}_{i=1}^K$ and normalized coordinate buffers $\{\tilde{\mathcal{X}}_i\}_{i=1}^K$

$(\sigma_x, \sigma_y, R) \leftarrow \text{eig}(\Sigma)$
area $\leftarrow 2\pi\sigma_x\sigma_y$
for $i = 1, \dots, K$ **do**
  $\hat{N}_i \leftarrow \lfloor c \cdot \text{area}/p_i^2 \rfloor$
end for
for $i = K, \dots, 1$ **do**
  $N_i \leftarrow \min\left(\hat{N}_i, N - \sum_{j=i+1}^K N_j\right)$
end for
$\mathcal{H} \leftarrow \textsc{InitSpatialHash}(\{p_i\}, \{N_i\})$
for $i = 1, \dots, K$ **do**
  attempts $\leftarrow 0$
  while $|P_i| < N_i$ **and** attempts $< 10N_i$ **do**
    attempts $\leftarrow$ attempts $+ 1$
    $(x, y) \sim \mathcal{N}(\mu, \Sigma)$
    if (x, y, p_i) not entirely within I **then**
      continue
    end if
    if $\textsc{Overlap}(\mathcal{H}, (x, y, p_i)) > \tau$ **then**
      continue
    end if
    $\mathcal{H} \leftarrow \textsc{AddBox}(\mathcal{H}, (x, y, p_i))$
    $P \leftarrow \textsc{BilinearCrop}(I, (x, y), p_i)$
    $\tilde{\mathbf{x}} \leftarrow \left((x - W/2)/128,\ (y - H/2)/128\right)$
    $P_i \leftarrow \textsc{Push}(P_i,\ P)$
    $\tilde{\mathcal{X}}_i \leftarrow \textsc{Push}(\tilde{\mathcal{X}}_i,\ \tilde{\mathbf{x}})$
  end while
end for

2. Related Work

Several lines of research contain methods that share thematically similar ideas in transformer-based segmentation, handling multi-scale objects, dynamic sampling, or biologically inspired foveation. Mask-classification frameworks such as MaskFormer and Mask2Former (Cheng et al., 2021; 2022) provide a strong transformer-based route to full-image semantic, instance, and panoptic segmentation, but their learned-query full-scene prediction setting differs from prompted single-object segmentation. *Deformable Convolutional Networks* (Dai et al., 2017) and their successors Zhu et al. (2019); Xiong et al. (2024), as well as deformable-attention vision transformers such as DAT (Xia et al., 2022), introduce learnable offsets or selective kernels to better handle varying spatial structures. *Focal Sparse Convolutional Networks* (Chen et al., 2022) focus computation on salient 3D regions in point clouds. Additionally, biologically inspired foveation and overview-first / look-

Figure 4. Visualization of our FLIP (Fovea-Like Input Patching) approach applied to an image from the KITTI-360 dataset, showcasing potential applications in autonomous driving. The figure illustrates how our model dynamically focuses on multiple objects within a complex urban scene by allocating multi-resolution patches centered around prompted object locations. Higher-resolution patches (smaller sizes) are concentrated on critical areas such as vehicles and road signs, emulating a foveal vision system, while lower-resolution patches (larger sizes) cover peripheral regions to enable the consideration of the surrounding context. Patches are color-coded by size: purple for 16×16 patches, yellow for 8×8, green for 4×4, blue for 2×2, and red for 1×1.

closely-next processing have been explored in works like Lukanov et al. (2021); Kaplanyan et al. (2019); Thavamani et al. (2021) and in recent backbones such as OverLoCK (Lou & Yu, 2025) and TransNeXt (Shi, 2024). Most closely related to our setting, Segment This Thing (STT) (Schmidt & Newcombe, 2025) applies a fixed, point-centered foveated tokenization to reduce encoder tokens, whereas FLIP performs object-conditioned, off-grid multi-resolution patch sampling driven by a 2D Gaussian prior. FLIP is therefore complementary to unsupervised object-centric models: rather than discovering objects or disentangling full scenes by itself, it assumes an object-level prompt and uses foveated sampling to segment that target efficiently.

3. Methods

In this section, we present the overall Fovea-Like Input Patching (FLIP) architecture. Its main processing pipeline is shown in Figure 2. FLIP is a supervised, prompted vision model for efficient object-centric single-object segmentation It combines a fovea-inspired patching mechanism with a Vision Transformer (ViT)-based encoder. The encoded information is then used to generate the targeted object's segmentation mask.

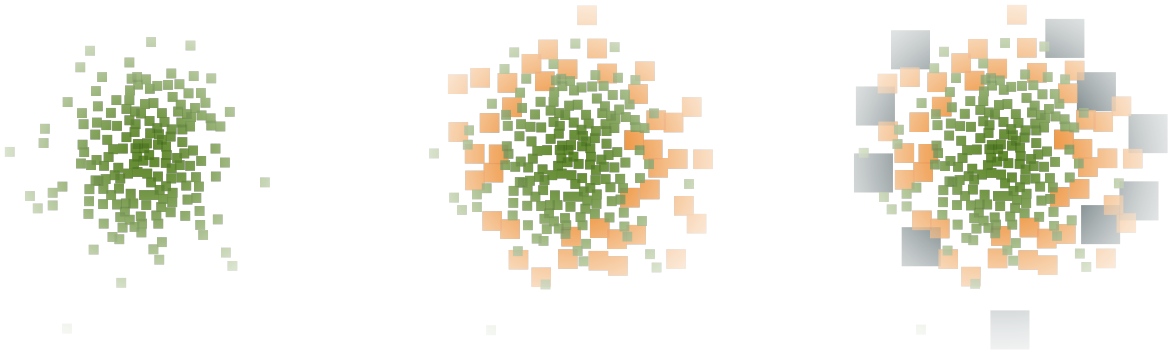
The fovea-like patching mechanism dynamically selects multi-resolution patches based on a 2D Gaussian distribution, which has an effect similar to the query in SAM variants but acts directly on the input image. The supplied Gaussian therefore anchors both the encoder input and the sparse mask queries around the target object. High-resolution patches focus on the object center, capturing fine details, while coarser patches cover peripheral regions, which inform FLIP about the surrounding context.

The ViT encoder processes the sampled patches and outputs latent tokens from which we compute keys $K_{1..n}$ and values $V_{1..n}$. The Pixel-Predictor then computes queries $Q_{x,y}$ from pixel coordinates and uses an attention mechanism to predict the segmentation mask with pixel level accuracy (see 9c).

FLIP is trained end-to-end with a sparse supervised loss for mask prediction. In the following subsections, we detail the fovea-like sampling mechanism and summarize the ViT encoder and Pixel-Predictor; full mathematical descriptions of the latter are provided in Appendix A.

3.1. Fovea-like Input Patching

We equip our ViT-based model with a multi-resolution, fovea-inspired patching mechanism that centers around the object of interest. The mechanism preserves high-resolution detail at the object center and coarser coverage in peripheral regions. Specifically, we derive the object center $\mu = (\mu_x, \mu_y)$ and covariance Σ from the ground-truth mask, yielding a 2D Gaussian $\mathcal{N}(\mu, \Sigma)$ that approximates the object's spatial extent and orientation in compressed



(a) 4x4 Patches (b) 8x8 Patches (c) 16x16 Patches

Figure 5. Example of FLIP's iterative, per-resolution sampling: we sample 4×4 patches first, then 8×8, and finally 16×16. Enforcing a maximum overlap biases larger patches toward the perimeter, yielding a fovea-like patching.

form. For input sampling, the Gaussian serves as a spatial input query—similar to the prompts in SAM variants—from which we then draw patches at multiple scales. Mask-derived Gaussian prompts use image moments of the target mask to estimate both the object center and a full covariance matrix, so they can encode scale and orientation. This differs from box-derived prompts, which use only an axis-aligned box and therefore do not contain mask-shape or orientation information beyond the box extents.

We define K patch sizes $p_1 < p_2 < \cdots < p_K$, from smallest (highest resolution) to largest. We fix the total number of patches N and choose N_i patches for patch size p_i as follows. First, we compute default numbers $\hat{N}_i$ by approximating the integral of the 2D Gaussian and dividing by p_i^2, scaled by a coverage parameter $c \in (0.1, 2)$ that controls the density of patch sampling. Then, starting at the coarsest resolution (i.e., p_K), we compute $N_i = \min(\hat{N}_i, N - \sum_{j=i+1}^{K} N_j)$. As a result, the chosen patches cover the Gaussian inner area with a density proportional to c and the number of chosen patches per size distributes itself from coarse to fine until the total number of patches N is reached ($N = \sum_i^K N_i$).

We sample N_i patches from $\mathcal{N}(\mu, \Sigma)$ with continuous coordinates, allowing patches to be centered at real-valued positions $(x, y) \in \mathbb{R}^2$. Pixel values are extracted using bilinear interpolation, enabling sub-pixel precision. To control patch overlap, we employ a spatial hash grid that tracks placed patches. When sampling, we reject candidate positions that would result in overlap exceeding the maximum overlap threshold $\tau \in [0, 4]$. This ensures efficient spatial coverage while allowing for controlled redundancy. We implement this sampling mechanism as a Python C extension in the data loading pipeline, achieving approximately 1ms average extraction time per image. Each sampled patch is first flattened and then mapped to a common embedding space via resolution-specific encoders E_{r_i} (similarly to Jaegle et al. (2021)), yielding a set of tokens $T = \{t_1, ..., t_N\}$ of equal tensor size. These embedded tokens are concatenated and fed to the main ViT layers. Note that the approach is in principle independent of the full image size, because sampling depends on μ and Σ only.

Patch Sampling Details Algorithm Algorithm 1 summarizes the extraction routine: μ and Σ parametrize the Gaussian object query, $p_1 < \cdots < p_K$ denotes the K patch sizes, N the total patch budget, c the coverage factor, and τ the maximal allowed overlap count per patch.

The helper INITSPATIALHASH instantiates the bounding-box hash grid such that every accepted patch overlaps previous ones at most τ times. BILINEARCROP evaluates the RGB values on a $p_i \times p_i$ grid centered at (x, y), enabling sub-pixel sampling of the Gaussian draw; $\tilde{\mathbf{x}}$ stores the normalized center coordinates used to build the positional encodings, i.e., $\tilde{x} = (x - W/2)/128$ and $\tilde{y} = (y - H/2)/128$. If the rejection steps exhaust the $10N_i$ attempts before filling N_i, the collected count $|P_i|$ is kept. Finally, every patch tensor is flattened and fed through its resolution-specific encoder E_{r_i}, producing $T = \{t_1, \ldots, t_{\sum_i |P_i|}\}$, the token sequence consumed by the transformer-based encoder.

3.2. Encoder and Pixel-Level Mask Prediction

The sampled patch tokens are processed by a ViT-based encoder with LaPE-style positional embedding injection and separate layer normalizations for queries, keys, and values. Encoder outputs feed into a pixel-level predictor that builds positional queries from target coordinates and attends to the encoded patch tokens to produce mask logits. Detailed mathematical formulations and architectural schematics, including attention block and pixel predictor diagrams, are

(a) Fire Hydrant (b) Apple (c) Truck (d) Airplane

Figure 6. Examples from our synthetic dataset. Objects from various categories are rendered with HDRI Haven backgrounds.

provided in Appendix A (9b, 9c).

3.3. Training

We train FLIP exclusively on META's SA-1B Dataset (Kirillov et al., 2023) with four model variants: FLIP-Tiny (0.51M parameters), FLIP-Small (2.3M parameters), FLIP-Middle (11.5M parameters), and FLIP-Large (96.6M parameters), performing approximately 8.5M, 7.4M, 5.8M and 2.7M updates respectively with a batch size of 256. These models are significantly more parameter-efficient than SAM variants (up to 641.1M parameters).

To fairly compare against bounding-box-prompted baselines, we also report FLIP$_{bb}$-* variants that consume 2D Gaussian prompts constructed from bounding boxes rather than mask-derived Gaussians. For a box $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, we set

$$\begin{aligned}\mu &= \left(\frac{x_{\min}+x_{\max}}{2}, \frac{y_{\min}+y_{\max}}{2}\right),\\ \sigma_x &= 0.188\,(x_{\max}-x_{\min}),\\ \sigma_y &= 0.34\,(y_{\max}-y_{\min}),\\ \Sigma &= \mathrm{diag}(\sigma_x^2, \sigma_y^2).\end{aligned}$$

Each FLIP$_{bb}$ model is fine-tuned on SA-1B with these box-derived Gaussians to match the information content available to the other bounding-box baselines.

For data augmentation, we apply random perturbations to the 2D Gaussian derived from ground truth masks by shifting center coordinates, stretching or compressing σ_x and σ_y, and introducing small rotations. The number of sampled tokens N is varied around $\mu = 512$ during training to improve generalization. Additionally the coverage c parameter is randomly sampled from $[0.1, 2]$ and the overlap threshold parameter τ is sampled from $[0, 4]$.

In a final learning phase we finetune FLIP variants either with bounding box derived prompts or full 2d Gaussian prompts without data augmentation. In this phase, we fix $N = 512$, $c = 1.44$ and $\tau = 2.67$. The coverage and overlap values were chosen to maintain a high information density during the patch sampling process.

Sparse Mask Prediction. Instead of dense mask prediction, we employ a computationally efficient sparse sampling strategy that focuses on boundary regions. Pixels are grouped into seven distance-based categories according to their L1 distance from mask boundaries: [0-2), [2-4), [4-8), [8-16), [16-32), [32-64), and [64-∞) pixels. For each sample, we select 2048 pixels distributed across these groups with equal sampling from inside and outside the mask to maintain class balance.

We implement adaptive sampling that dynamically redistributes pixels across distance groups based on prediction difficulty. Groups with lower IoU performance receive more samples in subsequent iterations, ensuring computational focus on challenging boundary regions.

The sparse mask loss reduces computational complexity from $O(HW)$ to $O(k)$ where $k \ll HW$:

$$\mathcal{L} = \mathrm{BCE}(\hat{M}_{\text{sparse}}, M_{\text{sparse}}) \tag{1}$$

where $\hat{M}_{\text{sparse}}$ and M_{sparse} represent predicted and ground truth values at sampled locations.

3.4. Inference

During inference, FLIP employs two strategies to efficiently generate segmentation masks while maintaining the sparse computation paradigm.

5-Sigma Bounding Box Prediction Instead of predicting mask values for all pixels in the image, we exploit the Gaussian prior to restrict predictions to a 5-sigma bounding box around the object center. Given the Gaussian parameters $(\mu_x, \mu_y, \sigma_x, \sigma_y)$ from the input prompt, we compute a bounding box using $\sigma_{\text{iso}} = \max(\sigma_x, \sigma_y)$ and sample pixels only within the region $[\mu_x \pm 5\sigma_{\text{iso}}, \mu_y \pm 5\sigma_{\text{iso}}]$. This reduces the number of mask queries from $O(HW)$ to $O(k^2)$ where

$k \propto \sigma_{\text{iso}}$, providing significant speedup for small objects while maintaining full accuracy for larger ones.

Bounding-Box Derived 2D Gaussian Prompts When an axis-aligned bounding box is provided as the input prompt, we convert it into a 2D Gaussian prompt (center at the box midpoint, σ_x and σ_y set as fixed fractions of the box width and height, covariance kept axis-aligned). We then predict mask values only for pixels inside the prompted 2D bounding box. Pixels outside the box are never queried and their values in the predicted final mask are set to zero.

4. Results

We evaluate FLIP on five standard benchmarks: Hypersim, KITTI-360, OpenImages, COCO and LVIS (Roberts et al., 2021; Liao et al., 2022; Kuznetsova et al., 2020; Lin et al., 2014; Gupta et al., 2019). Additionally, we evaluate FLIP on our newly constructed synthetic dataset ObjaScale, designed as a controlled scale-invariance stress test rather than a realistic object-segmentation benchmark. Like SAM, FLIP was trained on the SA-1B dataset (Kirillov et al., 2023), but was neither trained nor fine-tuned on the ObjaScale or any of the other evaluation datasets. Our experiments compare FLIP against state-of-the-art segmentation methods namely SAM2 (Ravi et al., 2024), SAM (Kirillov et al., 2023), EfficientSAM (Xiong et al., 2023), MobileSAM (Zhang et al., 2023), and FastSAM (Zhao et al., 2023). Additionally, we compare our method to another foveated segmentation approach, STT (Schmidt & Newcombe, 2025). However, this comparison is not directly comparable to the other segmentation baselines, since STT relies solely on point prompts, which are ambiguous regarding whether they indicate an entire object or only a part of it. In contrast, bounding-box or 2D-Gaussian prompts are unambiguous in this respect. Therefore we only report STT evaluation results on ObjaScale in Figure 7, where the identification of the object in question is much easier in comparison to the other datasets and the challenge is the extreme scale invariance. Point-only prompts are therefore not the main focus of this evaluation, and the STT comparison is limited to ObjaScale because the single centered target reduces part-whole ambiguity. Besides comparing the overall performance, we focus on handling small objects, achieving high IoU, and balancing parameter/runtime trade-offs.

4.1. Experimental Setup and Dataset Creation

To isolate scale effects, we created ObjaScale as a synthetic dataset with a single centered target object on a random HDRI background. We selected 68 diverse categories from Objaverse (Deitke et al., 2023) and combined each with high-resolution HDRI Haven (Zaal, Greg) backgrounds in Blender, yielding 10,200 synthetic images. Image resolution was randomized from 512 to 8192 pixels, while object dimensions were constrained to a maximum width/height of 256 pixels, producing masks spanning minuscule ($< 0.0001\%$) to large fractions of the image area (up to 25%) (Figure 6). For SAM variants, bounding-box prompts computed from ground-truth masks were used while FLIP employed 2D Gaussian prompts by design. For STT, we follow the point-prompt selection procedure described in their paper: the prompt is chosen as the point inside the ground-truth mask that lies farthest away from any mask boundary.

4.2. Overall Segmentation Performance

We denote bounding-box–prompted variants with the subscript "$_{bb}$" (e.g. FLIP$_{bb}$-Small); these models consume 2D Gaussian prompts derived from bounding boxes to match the information available to box-prompted baselines.

Table 1 presents results across all datasets. FLIP consistently outperforms SAM variants with fewer parameters. FLIP-Large attains 83.26% mean IoU with 96.6M parameters, outperforming SAM-H's 75.41% IoU (641.1M parameters) and SAM2-L's 75.87% IoU (224.5M parameters)—a $6.6\times$ and $2.3\times$ parameter reduction with improved accuracy. FLIP-Tiny (0.51M parameters) achieves 79.90% mean IoU, surpassing all SAM variants including SAM2-L. In the box-prompted setting, FLIP$_{bb}$-Large has a SAM-B-scale parameter count, remains faster, and outperforms SAM-B on all six benchmarks in Table 1, supporting the efficiency result under matched box-prompt information. The large standard deviations in Table 1 reflect substantial per-instance IoU variability caused by heterogeneous and challenging benchmarks with diverse objects, shapes, backgrounds, weak contrast, clutter, crowded scenes, and geometric configurations, while their similar magnitudes across methods indicate that this difficulty affects all approaches comparably.

ObjaScale should be interpreted as a controlled diagnostic rather than as a stand-alone benchmark for general segmentation: its single centered target reduces prompt and object-identity ambiguity, allowing Figure 7 to isolate how performance changes with absolute and relative mask size. On this diagnostic, FLIP's advantage is particularly pronounced, achieving 88.6–90.7% IoU versus SAM-H's 73.76%, demonstrating superior scale invariance across diverse object sizes. Notably, SAM2 dropped in performance on ObjaScale in comparison to SAM. Only STT, the other foveated segmentation model we evaluated, can maintain consistent performance on ObjaScale even for relatively small objects (see Figure 7), but it is still over 15 percentage points behind FLIP. In this setting, FLIP remains strong beyond the smallest objects and degrades less than full-image baselines as relative object area shrinks. ObjaScale therefore exposes how fixed-resolution full-image encoders allocate

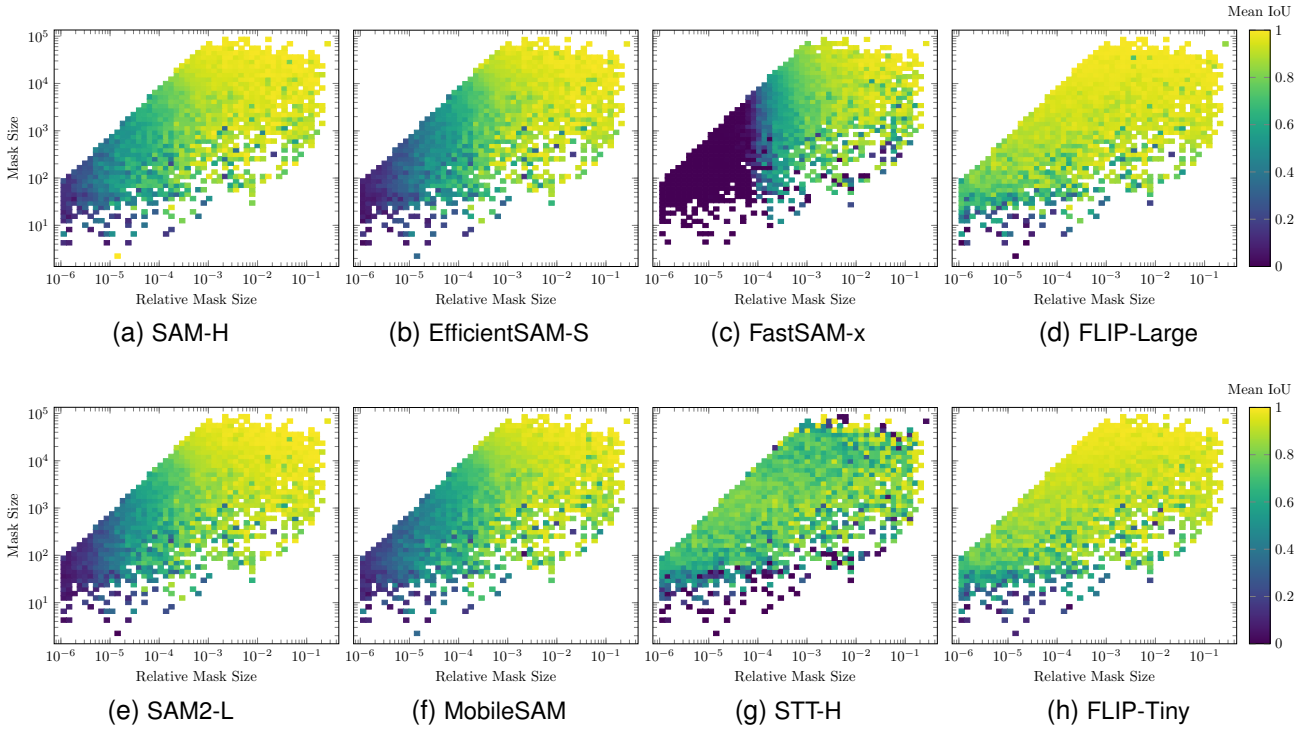


Figure 7. Heatmaps illustrating IoU performance in relation to relative (X-axis) vs. absolute mask size (Y-axis) on the ObjaScale dataset. The SAM-variants' full-image methods lose performance the smaller the object is in relation to the overall image size (left side of the graphs), with FastSAM-x failing completely for objects smaller than 0.1% of the image area. Only foveated methods like STT and FLIP maintain performance for relatively small objects. Notably, FLIP still maintains performance for relatively large objects in comparison with STT and is also stronger when the absolute mask sizes reach the detection limit at masks with fewer than 100 pixels (bottom part of the graphs).

Table 1. Comparison of Mean IoU (%) and Std IoU (%) across different datasets.

Model	Size (M)	Time (ms)	Hypersim	KITTI-360	OpenImage	COCO	LVIS	ObjaScale (ours)	Average
			Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)
FastSAM-s	11.8	9.94	36.80 ± 34.62	37.71 ± 32.76	61.14 ± 32.10	46.51 ± 33.73	37.11 ± 35.19	48.20 ± 35.69	44.58 ± 34.02
FastSAM-x	72.2	24.32	34.36 ± 34.90	38.92 ± 34.06	69.31 ± 29.25	55.31 ± 34.55	43.24 ± 37.47	47.11 ± 36.41	48.04 ± 34.44
MobileSAM	10.13	21.15	68.03 ± 21.36	64.64 ± 18.87	82.20 ± 16.74	73.80 ± 17.48	71.59 ± 21.36	67.72 ± 25.03	71.33 ± 20.14
EfficientSAM-T	10.22	26.75	68.35 ± 24.52	60.82 ± 24.72	84.32 ± 15.08	75.93 ± 17.04	75.56 ± 19.95	68.74 ± 26.43	72.29 ± 21.29
EfficientSAM-S	26.41	47.98	69.65 ± 22.20	64.88 ± 19.35	85.99 ± 14.54	77.07 ± 16.81	75.18 ± 21.06	67.80 ± 26.76	73.43 ± 20.12
SAM-B	93.7	72.67	71.46 ± 20.88	62.38 ± 21.41	84.72 ± 15.38	76.07 ± 16.95	76.93 ± 19.17	71.38 ± 25.36	73.82 ± 19.86
SAM-L	312.3	148.78	72.13 ± 21.21	62.73 ± 20.31	86.94 ± 13.41	78.19 ± 16.05	77.93 ± 19.43	72.68 ± 25.22	75.10 ± 19.27
SAM-H	641.1	232.04	72.37 ± 21.65	62.47 ± 20.52	87.06 ± 13.53	78.41 ± 16.15	78.36 ± 19.50	73.76 ± 24.59	75.41 ± 19.32
SAM2-T	38.96	30.38	71.56 ± 20.50	66.80 ± 20.40	87.08 ± 12.79	78.55 ± 15.37	76.85 ± 19.31	69.89 ± 26.28	75.12 ± 19.11
SAM2-S	46.06	33.43	71.37 ± 21.33	66.43 ± 21.83	87.61 ± 12.66	78.82 ± 15.54	77.16 ± 19.40	69.68 ± 26.43	75.18 ± 19.53
SAM2-B+	80.85	46.43	72.07 ± 20.53	65.77 ± 19.32	87.96 ± 11.97	79.35 ± 15.11	77.33 ± 19.46	70.50 ± 26.53	75.50 ± 18.82
SAM2-L	224.45	88.64	72.89 ± 20.53	67.30 ± 20.78	88.64 ± 11.35	79.63 ± 15.01	77.20 ± 19.66	69.54 ± 27.00	75.87 ± 19.05
FLIP-Tiny	**0.51**	9.68	74.14 ± 21.03	72.62 ± 16.56	86.74 ± 12.59	79.83 ± 14.27	77.55 ± 20.02	88.55 ± 15.08	79.90 ± 16.59
FLIP-Small	2.3	11.94	75.59 ± 20.95	**72.98** ± 16.83	88.96 ± 10.56	81.32 ± 14.02	79.12 ± 20.27	89.67 ± 14.91	81.27 ± 16.26
FLIP-Middle	11.5	18.00	79.14 ± 19.69	72.21 ± 16.90	90.84 ± 8.80	**82.86** ± 12.89	82.61 ± 17.78	**90.70** ± 13.86	83.06 ± 14.99
FLIP-Large	96.6	44.13	**80.34** ± 19.28	70.72 ± 17.51	**91.36** ± 7.72	82.66 ± 13.85	**83.83** ± 16.94	90.65 ± 13.80	**83.26** ± 14.85
FLIP$_{bb}$-Tiny	**0.51**	**8.81**	68.32 ± 21.37	66.77 ± 17.92	83.08 ± 14.58	75.38 ± 16.03	75.50 ± 19.72	86.80 ± 16.41	75.97 ± 17.67
FLIP$_{bb}$-Small	2.3	10.63	70.53 ± 21.15	67.09 ± 18.07	85.76 ± 12.86	77.48 ± 15.49	77.93 ± 18.96	88.39 ± 15.99	77.86 ± 17.09
FLIP$_{bb}$-Middle	11.5	14.28	72.21 ± 20.87	67.53 ± 18.34	87.67 ± 11.46	78.77 ± 15.77	79.90 ± 18.18	88.93 ± 15.63	79.17 ± 16.71
FLIP$_{bb}$-Large	96.6	31.45	72.83 ± 20.83	66.93 ± 19.14	88.59 ± 10.64	78.69 ± 16.56	80.79 ± 17.69	88.91 ± 15.65	79.46 ± 16.75

compute under extreme scale changes.

On established benchmarks, FLIP maintains competitive performance with 91.36% IoU on OpenImages (vs. SAM2-L's 88.64%) and 82.66% on COCO (vs. SAM2-L's 79.63%). Thus, the broader empirical claim rests on the complete benchmark suite in Table 1.

4.3. Prompt Robustness to Bounding-Box Noise

We evaluate prompt robustness on a fixed, reproducible random subset of our evaluation datasets shared by all models,

Table 2. Mean IoU (%) under bounding-box prompt noise, averaged equally over datasets.

Model	Size (M)	Clean	2.5%	5%	10%	20%
FastSAM-s	11.8	43.96	43.90	43.72	43.00	39.88
FastSAM-x	72.2	48.87	48.79	48.58	47.64	43.63
MobileSAM	10.13	71.27	70.85	69.26	62.81	44.34
EfficientSAM-T	10.22	72.03	72.10	70.56	64.75	46.59
EfficientSAM-S	26.41	73.33	73.16	72.17	**67.93**	**52.40**
SAM-B	93.7	73.70	73.24	71.41	64.05	43.76
SAM-L	312.3	74.99	74.68	73.24	65.93	45.86
SAM-H	641.1	75.36	75.08	**73.94**	67.91	47.43
SAM2-T	38.96	75.02	74.41	72.52	64.64	44.09
SAM2-S	46.06	75.06	74.52	72.58	64.28	42.33
SAM2-B+	80.85	75.41	74.95	73.21	65.65	45.29
SAM2-L	224.45	75.78	**75.31**	73.58	65.84	44.66
FLIP$_{bb}$-Tiny	**0.51**	76.09	71.20	64.04	50.47	30.76
FLIP$_{bb}$-Small	2.3	78.01	72.06	63.92	49.41	29.78
FLIP$_{bb}$-Middle	11.5	79.31	70.74	59.51	43.18	25.61
FLIP$_{bb}$-Large	96.6	**79.53**	68.55	56.38	41.98	26.34

using the same selected instance indices and prompt perturbations for every method. For each dataset, the subset contains $\max(10{,}000, 10\%)$ valid instances where available, and all valid instances otherwise. Clean denotes the unperturbed ground-truth box evaluated on this same ablation subset, rather than being copied from the full evaluation in Table 1; the clean values may therefore differ slightly from the corresponding Table 1 values. At noise level p, the box center is shifted randomly by $p\%$ of the box diagonal and the side lengths are scaled by $\pm p\%$, with noisy settings averaged over repeated perturbations. Table 2 reports the dataset-balanced mean IoU. FLIP$_{bb}$ performs best with clean boxes, but degrades more rapidly under noise because its box-derived Gaussian controls both patch sampling and sparse mask querying. This identifies prompt accuracy as the main practical limitation of the current FLIP$_{bb}$ interface.

4.4. Parameter Efficiency and Performance

Figure 1 illustrates the parameter efficiency of FLIP variants. While maintaining superior segmentation quality, FLIP models operate with orders of magnitude fewer parameters than SAM variants. The efficiency gains translate to faster inference times (Figure 3), with FLIP-Large requiring only 44.13ms compared to SAM2-L's 88.64ms, representing an approximate $2\times$ speedup alongside improved accuracy (measured on a Nvidia RTX4090 graphics card).

5. Conclusion

We have introduced FLIP, a novel vision model that fundamentally rethinks object segmentation through fovea-inspired input patching. By directly sampling multi-resolution patches centered on objects of interest, FLIP achieves state-of-the-art segmentation performance while using orders of magnitude fewer parameters than existing approaches.

Our key innovation lies in the scale-invariant patch sampling mechanism that adapts processing resolution to object characteristics, allocating computational resources where they matter most. This design enables FLIP-Large to achieve 83.26% mean IoU with only 96.6M parameters—outperforming SAM2-L's 75.87% IoU (224.45M parameters) while being about $2\times$ faster. Even our smallest model, FLIP-Tiny (0.51M parameters), surpasses all SAM variants with 79.90% mean IoU.

The introduction of ObjaScale reveals a critical limitation in current segmentation models: their inability to effectively handle very small objects in high-resolution scenes. FLIP addresses this gap through its fovea-like sampling, maintaining 88.6–90.7% IoU on ObjaScale where SAM variants achieve only 67.7–73.8%. This capability, combined with consistent performance improvements across Hypersim, KITTI-360, OpenImages, COCO, and LVIS, demonstrates FLIP's potential for applications requiring precise segmentation of objects across extreme scale variations.

The prompt-noise ablations also expose a limitation of the current box-prompted interface. Because FLIP$_{bb}$ converts an input box into a Gaussian that determines both where the encoder samples patches and which pixels are queried for mask prediction, inaccurate boxes can move computation away from the target and reduce robustness more sharply than in full-image encoders. This does not diminish the clean-prompt efficiency gains, but it makes prompt quality an important consideration and motivates future work on uncertainty-aware prompt construction and training.

Looking forward, FLIP's architectural efficiency and strong generalization open new possibilities for real-time object-centric vision systems. Our current formulation, however, assumes access to a 2D Gaussian prompt, which is not yet a drop-in replacement for the bounding-box- or point-based prompts used in most deployed detection and segmentation pipelines, so deriving such Gaussians from detector boxes, tracking filters, or slot-based scene models and studying robustness to imperfect, non-Gaussian prompts remains important future work. We nevertheless expect FLIP's foveated encoder and Gaussian object prompts to be particularly useful in tracking scenarios where an additional temporal module may predict future 2D Gaussian locations of objects across frames. Our results suggest that biologically inspired selective attention mechanisms can deliver both superior accuracy and computational efficiency, paving the way for more capable and sustainable computer vision applications in domains ranging from autonomous navigation to medical imaging.

Acknowledgement

This work received funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645 as well as from the Cyber Valley in Tübingen, CyVy-RF-2020-15. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Manuel Traub, and the Alexander von Humboldt Foundation for supporting Martin Butz.

Impact Statement

This paper presents work whose goal is to improve the efficiency and accuracy of prompted object segmentation, especially for small objects in high-resolution images. Such methods may reduce computational cost and energy consumption in vision systems and may support applications such as autonomous navigation, robotics, scientific imaging, and medical image analysis. At the same time, more efficient segmentation can also be used in sensitive settings such as surveillance, tracking, and automated decision-making. Deployments in these settings should account for privacy, consent, dataset bias, robustness to imperfect prompts, and the possibility of downstream harms when segmentation errors affect people. The method is a prompted single-object segmentation model rather than a complete decision system, and responsible use requires appropriate human oversight, validation on the target domain, and compliance with applicable laws and ethical standards.

References

Chen, Y., Li, Y., Zhang, X., Sun, J., and Jia, J. Focal sparse convolutional networks for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5428–5437, 2022.

Cheng, B., Schwing, A. G., and Kirillov, A. Per-pixel classification is not all you need for semantic segmentation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17864–17875, 2021.

Cheng, B., Misra, I., Schwing, A. G., Kirillov, A., and Girdhar, R. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1290–1299, 2022.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., and Farhadi, A. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13142–13153, June 2023.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Elsayed, G., Mahendran, A., van Steenkiste, S., Greff, K., Mozer, M. C., and Kipf, T. Savi++: Towards end-to-end object-centric learning from real-world videos. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 28940–28954. Curran Associates, Inc., 2022.

Gupta, A., Dollar, P., and Girshick, R. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5356–5364, 2019.

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

Kaplanyan, A. S., Sochenov, A., Leimkühler, T., Okunev, M., Goodall, T., and Rufo, G. Deepfovea: Neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision*, 128(7):1956–1981, 2020.

Liao, Y., Xie, J., and Geiger, A. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco:

Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pp. 740–755. Springer, 2014.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

Lou, M. and Yu, Y. Overlock: An overview-first-look-closely-next convnet with context-mixing dynamic kernels. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 128–138, 2025.

Lukanov, H., König, P., and Pipa, G. Biologically inspired deep learning model for efficient foveal-peripheral vision. *Frontiers in Computational Neuroscience*, 15:746204, 2021.

Menary, S., Kaski, S., and Freitas, A. Transformer normalisation layers and the independence of semantic subspaces. *arXiv preprint arXiv:2406.17837*, 2024.

Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10912–10922, October 2021.

Schmidt, T. and Newcombe, R. Segment this thing: Foveated tokenization for efficient point-prompted segmentation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 29428–29437, 2025.

Shi, D. Transnext: Robust foveal visual perception for vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17773–17783, 2024.

Singh, G., Wu, Y.-F., and Ahn, S. Simple unsupervised object-centric learning for complex and naturalistic videos. *Advances in Neural Information Processing Systems*, 35:18181–18196, 2022.

Thavamani, C., Li, M., Cebron, N., and Ramanan, D. Fovea: Foveated image magnification for autonomous navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 15539–15548, 2021.

Traub, M., Otte, S., Menge, T., Karlbauer, M., Thuemmel, J., and Butz, M. V. Learning what and where: Disentangling location and identity tracking without supervision. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NeDc-Ak-H_.

Traub, M., Becker, F., Otte, S., and Butz, M. V. Learning object permanence from videos via latent imaginations. In *International Conference on Artificial Neural Networks*, pp. 223–240. Springer, 2024a.

Traub, M., Becker, F., Sauter, A., Otte, S., and Butz, M. V. Loci-segmented: improving scene segmentation learning. In *International Conference on Artificial Neural Networks*, pp. 45–61. Springer, 2024b.

Wu, K., Zhang, J., Peng, H., Liu, M., Xiao, B., Fu, J., and Yuan, L. Tinyvit: Fast pretraining distillation for small vision transformers. In *European conference on computer vision*, pp. 68–85. Springer, 2022.

Xia, Z., Pan, X., Song, S., Li, L. E., and Huang, G. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4794–4803, 2022.

Xiong, Y., Varadarajan, B., Wu, L., Xiang, X., Xiao, F., Zhu, C., Dai, X., Wang, D., Sun, F., Iandola, F., Krishnamoorthi, R., and Chandra, V. Efficientsam: Leveraged masked image pretraining for efficient segment anything. *arXiv preprint arXiv:2312.00863*, 2023.

Xiong, Y., Li, Z., Chen, Y., Wang, F., Zhu, X., Luo, J., Wang, W., Lu, T., Li, H., Qiao, Y., et al. Efficient deformable convnets: Rethinking dynamic and sparse operator for vision applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5652–5661, 2024.

Yu, R., Wang, Z., Wang, Y., Li, K., Liu, C., Duan, H., Ji, X., and Chen, J. Lape: Layer-adaptive position embedding for vision transformers with independent layer normalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5886–5896, 2023.

Zaal, Greg. Hdri haven, 2016. URL https://hdri-haven.com/. Accessed: 2024-11-20.

Zadaianchuk, A., Seitzer, M., and Martius, G. Object-centric learning for real-world videos by predicting temporal feature similarities. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhang, C., Han, D., Qiao, Y., Kim, J. U., Bae, S.-H., Lee, S., and Hong, C. S. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.

Zhao, X., Ding, W., An, Y., Du, Y., Yu, T., Li, M., Tang, M., and Wang, J. Fast segment anything, 2023.

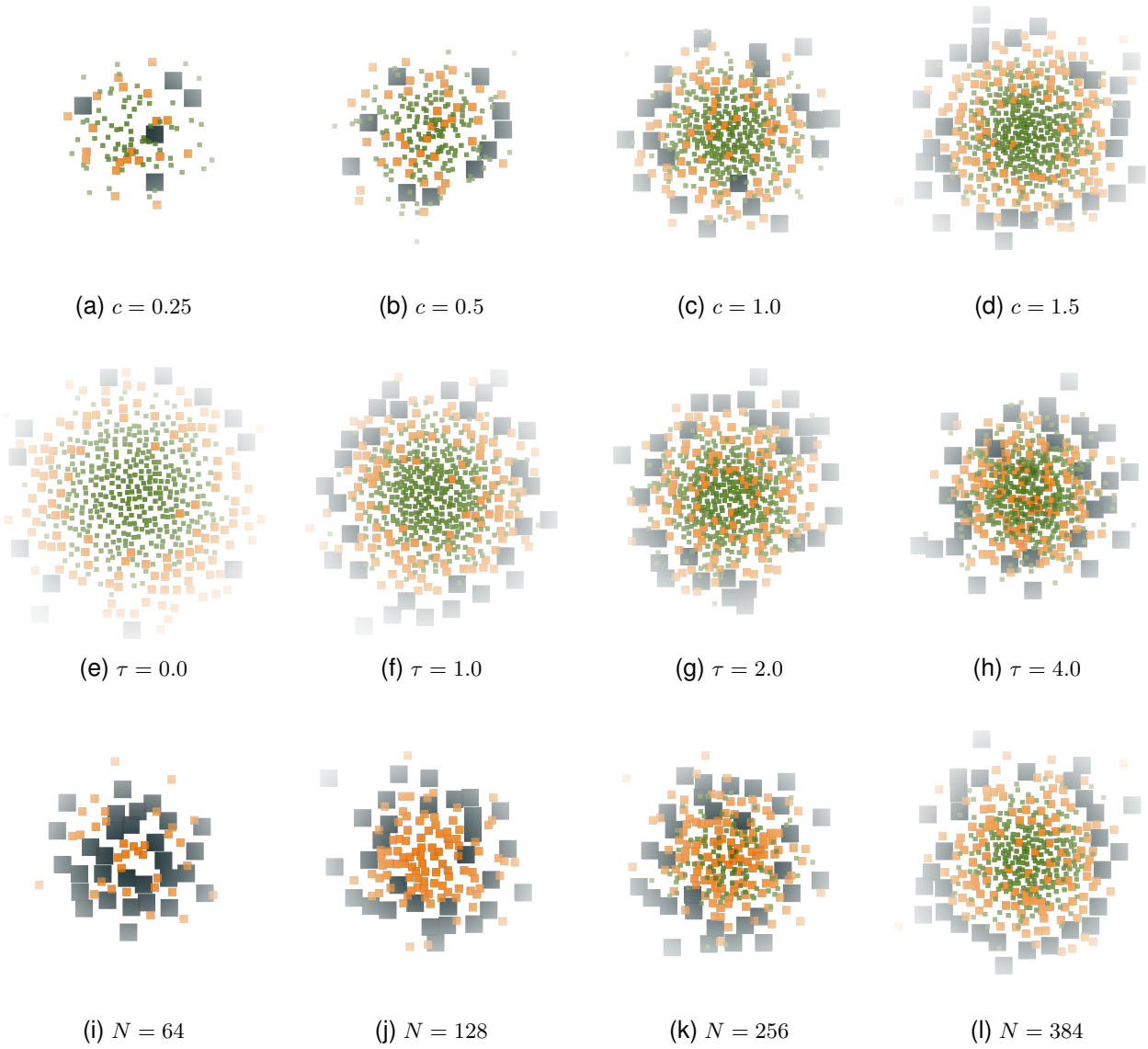
Zhu, X., Hu, H., Lin, S., and Dai, J. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9308–9316, 2019.

A. Encoder and Pixel-Level Mask Prediction

A.1. Encoder

For encoding the multi-resolution patches, we employ a Vision Transformer architecture with modified attention blocks inspired by LaPE (Yu et al., 2023). Our design differs from standard transformers by incorporating position-aware attention without position embedding normalization, maintaining separate layer normalization for queries, keys, and values, which relates to recent analyses of how normalization placement affects semantic subspaces in transformers (Menary et al., 2024).

9b illustrates the architecture of our attention blocks. Each block receives the set of tokens X_l from the previous layer as input. The first layer receives $X_1 = T$, which are the tokens produced by patch-size-specific embedding MLPs $E_{r_0}..E_{r_K}$ that process the foveal patches. Similar to LaPE, we inject relative positional embeddings PE into every layer.



(a) $c = 0.25$ (b) $c = 0.5$ (c) $c = 1.0$ (d) $c = 1.5$

(e) $\tau = 0.0$ (f) $\tau = 1.0$ (g) $\tau = 2.0$ (h) $\tau = 4.0$

(i) $N = 64$ (j) $N = 128$ (k) $N = 256$ (l) $N = 384$

Figure 8. Example of sampled patches from an isotropic Gaussian with varying coverage c, the maximal allowed overlap count per patch τ and patch budget N. **a-d** varying coverage c with fixed $\tau = 1.5$ and $N = 512$. The coverage also affects the number of sampled patches and leads to an overall sparser patching of the input for $c < 1$. **e-h** varying τ with fixed $c = 1.5$ and $N = 512$. Allowing for more overlap between patches leads to a denser sampling of the input. **i-l** varying N with fixed $c = 1.5$ and $\tau = 1.5$ directly affects the information density of the input sampling with more patches leads to a more fined grained input.

The relative positional embeddings $(\hat{x}_i, \hat{y}_i)$ are computed with respect to the 2D Gaussian input query. From Σ, we extract rotation (θ_a, θ_b) and scale (σ_x, σ_y) via a standard eigenvalue decomposition of the covariance matrix. For each token at position (x_i, y_i), we shift coordinates by (μ_x, μ_y), rotate by (θ_a, θ_b), and scale by (σ_x, σ_y). We then use separate MLPs to compute positional embeddings for queries, keys and values:

$$\begin{aligned} \mathrm{PE}_i^{q,k,v} &= \mathrm{MLP}_i^{q,k,v}(\hat{x}_i, \hat{y}_i), \\ \mathbf{Q}'_i = \mathbf{Q}_i + \mathrm{PE}_i^q, \quad \mathbf{K}'_i &= \mathbf{K}_i + \mathrm{PE}_i^k, \quad \mathbf{V}'_i = \mathbf{V}_i + \mathrm{PE}_i^v, \end{aligned}$$

where MLP is a two-layer MLP with SiLU activations that maps 2D coordinates to positional embeddings.

Unlike standard transformers, we apply individual layer normalization to queries, keys, and values before projection, which empirically improved performance over a single layer normalization (see Table 9):

$$Q = \mathrm{LN}_Q(X_l)W_Q, \quad K = \mathrm{LN}_K(X_l)W_K, \quad V = \mathrm{LN}_V(X_l)W_V,$$

where LN_Q, LN_K, and LN_V are separate LayerNorm modules.

The attention computation proceeds as:

$$\begin{aligned} Q' = Q + \mathrm{PE}_Q, \quad K' = K + \mathrm{PE}_K, \quad V' = V + \mathrm{PE}_V, \\ \mathrm{Attn}(Q', K', V) = \mathrm{softmax}\left(\frac{Q'K'^T}{\sqrt{d_k}}\right)V', \\ Y = X_l + \alpha \cdot \mathrm{Attn}(Q', K', V')W_{out}, \end{aligned}$$

where α is a learnable scaling parameter initialized to 0.1 for stable training, and d_k is the key dimension.

Each attention block is followed by a residual MLP:

$$X_{l+1} = Y + \beta \cdot \mathrm{MLP}(\mathrm{LN}(Y)),$$

where β is another learnable scaling parameter. The MLP uses a bottleneck design with SiLU activations and expansion factor of 4. Depending on the model configuration, we stack 3, 5, 8, or 24 of these blocks to form the complete encoder.

A.2. Pixel-Level Mask Prediction

The Pixel-Predictor generates precise segmentation masks by computing attention between encoded patch features and query pixels. Given the encoder output tokens X_e, we predict mask values for arbitrary pixel coordinates through a sparse attention mechanism.

The predictor consists of preprocessing and postprocessing residual MLP layers surrounding a specialized attention module. We first preprocess the encoder features:

$$\hat{X}_e = \mathrm{MLP}_{\mathrm{pre}}(X_e)$$

For each target pixel at coordinates (x_t, y_t), we compute relative positional embeddings with respect to the input 2D Gaussian by shifting coordinates by (μ_x, μ_y), rotating by (θ_a, θ_b), and scaling by (σ_x, σ_y) to get $(\hat{x}_t, \hat{y}_t)$:

$$\begin{aligned} \mathrm{PE}_t &= \mathrm{MLP}_t(\hat{x}_t, \hat{y}_t), \\ \mathrm{PE}_i &= \mathrm{MLP}_i(\hat{x}_i, \hat{y}_i), \end{aligned}$$

where MLP is a two-layer MLP with SiLU activations that maps 2D coordinates to positional embeddings, and PE_i represents the positional embeddings for each encoded and preprocessed input patch $(\hat{X}_e)$.

In contrast to the encoder, the attention mechanism in the Pixel-Predictor is not residual. We compute embedding-informed keys and values as follows:

$$\begin{aligned} K_{\text{enhanced}} &= \mathrm{MLP}_K(\mathrm{concat}[\mathrm{LN}_K(\hat{X}_e), \mathrm{PE}_i]), \\ V_{\text{enhanced}} &= \mathrm{LN}_V(\hat{X}_e)W_V + \mathrm{MLP}_V(\mathrm{PE}_i) \end{aligned}$$

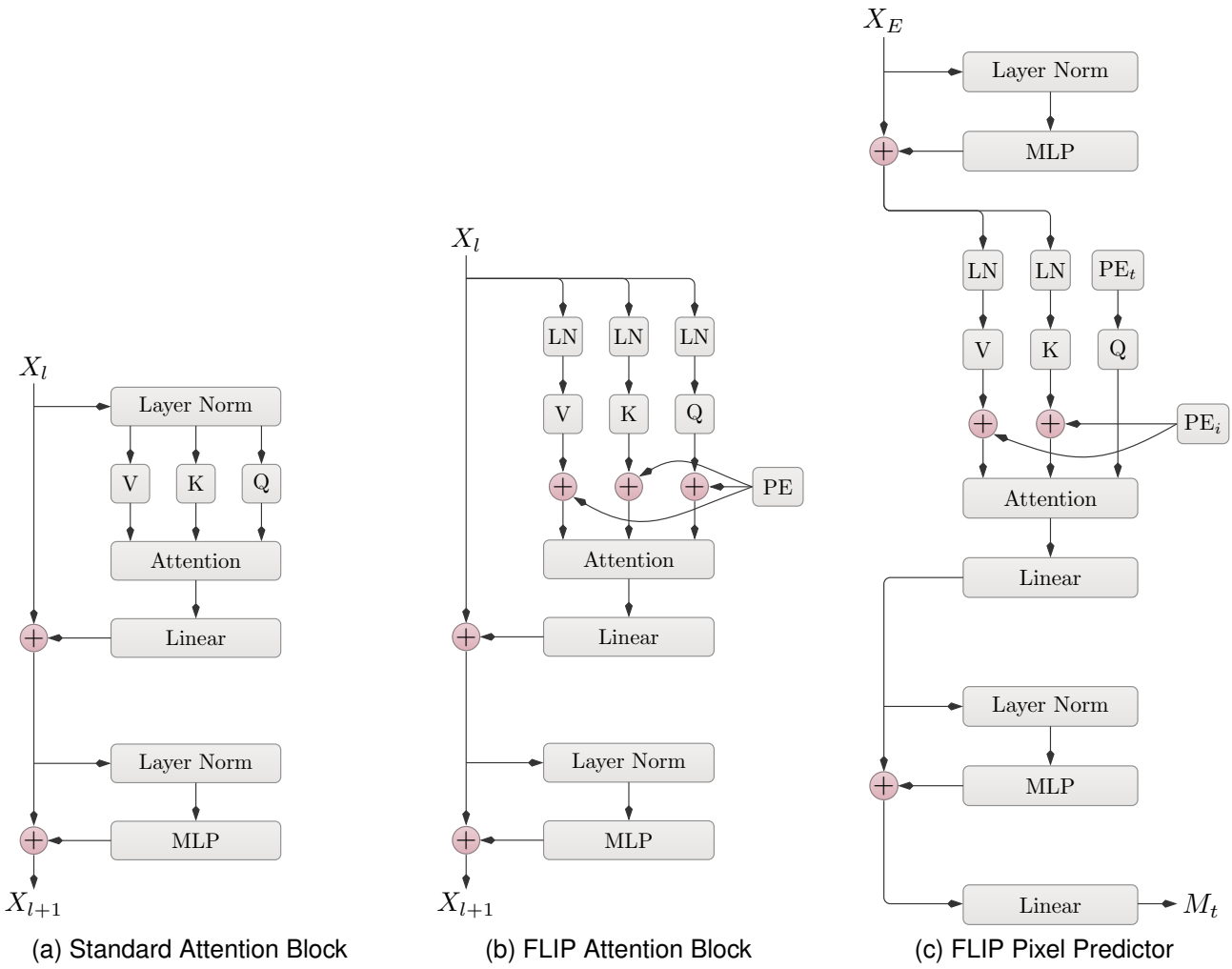


Figure 9. Architectural components of FLIP. (a) Standard pre-layer norm attention block for comparison. (b) Architecture of the FLIP attention block, where positional embeddings influence queries, keys and values independent of feature normalization. (c) Architecture of the FLIP Pixel-Predictor. Encoder features X_E are processed through a residual MLP. The attention mechanism computes keys (K) and values (V) enhanced with patch positional embeddings (PE_i), while queries (Q) use positional embeddings derived from the target pixels coordinates (PE_t). The attended features pass through a residual MLP and a final linear layer to produce mask predictions M_t for each query pixel.

For each query pixel, we compute:

$$\begin{aligned} Q_{\text{pixel}} &= \text{PE}_t, \\ \text{feat}_{\text{pixel}} &= \text{Attention}(Q_{\text{pixel}}, K_{\text{enhanced}}, V_{\text{enhanced}}), \\ M_{x,y} &= \text{Linear}(\text{feat}_{\text{pixel}} + \text{MLP}_{\text{post}}(\text{feat}_{\text{pixel}})) \end{aligned}$$

where $M_{x,y}$ is the predicted mask logit at position (x, y).

B. Small Object Segmentation

Table 3 reveals FLIP's capability for small object segmentation (objects <1% image area). FLIP-Large achieves 81.19% mean IoU on small objects compared to SAM2-L's 73.50%, demonstrating the effectiveness of our fovea-like patching mechanism. This advantage is most pronounced on ObjaScale, where FLIP-Tiny achieves 88.28% IoU on small objects compared to SAM-H's 72.26% or SAM2-L's 67.70%.

Table 3. Segmentation Accuracy on small-scale objects (< 1% image area)

Model	Size (M)	Time (ms)	Hypersim	KITTI-360	OpenImage	COCO	LVIS	ObjaScale (ours)	Average
			Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)
FastSAM-s	11.8	10.02	31.91 ± 32.85	32.55 ± 31.32	50.59 ± 32.66	38.50 ± 32.74	31.12 ± 33.01	45.91 ± 35.16	38.43 ± 32.96
FastSAM-x	72.2	24.37	24.77 ± 30.20	33.12 ± 33.34	56.58 ± 31.82	44.09 ± 34.05	34.96 ± 35.00	44.13 ± 35.62	39.61 ± 33.34
MobileSAM	10.13	21.83	65.67 ± 20.77	62.36 ± 18.41	76.76 ± 19.01	70.19 ± 17.36	68.46 ± 21.25	65.91 ± 24.86	68.22 ± 20.28
EfficientSAM-T	10.22	26.75	68.61 ± 20.18	63.11 ± 17.56	79.67 ± 16.78	72.91 ± 16.41	72.79 ± 19.80	67.11 ± 26.17	70.70 ± 19.48
EfficientSAM-S	26.41	47.98	66.74 ± 21.33	63.64 ± 18.01	80.82 ± 17.48	73.26 ± 16.78	71.64 ± 21.16	65.86 ± 26.63	70.33 ± 20.23
SAM-B	93.7	73.33	69.97 ± 19.58	63.05 ± 18.28	81.08 ± 16.87	74.48 ± 15.60	74.53 ± 19.11	69.79 ± 25.41	72.15 ± 19.14
SAM-L	312.3	149.45	69.76 ± 20.29	62.40 ± 18.44	82.76 ± 15.33	75.39 ± 15.59	74.75 ± 19.71	71.12 ± 25.32	72.70 ± 19.11
SAM-H	641.1	232.67	70.12 ± 20.66	62.02 ± 18.50	82.66 ± 15.81	75.62 ± 15.63	75.18 ± 19.83	72.26 ± 24.71	72.98 ± 19.19
SAM2-T	38.96	31.37	70.08 ± 18.88	67.29 ± 17.34	82.91 ± 14.38	75.65 ± 14.78	73.68 ± 19.44	68.09 ± 26.27	72.95 ± 18.51
SAM2-S	46.06	34.45	70.23 ± 18.89	67.63 ± 17.58	83.47 ± 14.23	75.91 ± 14.81	73.89 ± 19.51	67.87 ± 26.39	73.17 ± 18.57
SAM2-B+	80.85	47.43	70.13 ± 19.08	65.60 ± 17.15	83.69 ± 13.73	76.07 ± 14.83	73.81 ± 19.66	68.72 ± 26.55	73.00 ± 18.50
SAM2-L	224.45	89.61	70.95 ± 18.88	67.98 ± 17.56	84.48 ± 12.70	76.43 ± 14.49	73.48 ± 19.86	67.70 ± 26.99	73.50 ± 18.41
FLIP-Tiny	**0.51**	8.50	71.48 ± 21.90	**70.47** ± 16.86	84.39 ± 14.21	78.22 ± 14.93	75.25 ± 20.70	88.28 ± 15.33	78.01 ± 17.32
FLIP-Small	2.3	10.35	72.41 ± 21.77	69.99 ± 16.76	86.56 ± 11.92	79.19 ± 14.93	76.32 ± 21.20	89.38 ± 15.21	78.97 ± 16.96
FLIP-Middle	11.5	13.89	76.12 ± 20.57	68.82 ± 16.51	88.44 ± 10.21	81.15 ± 13.31	79.98 ± 18.69	**90.46** ± 14.10	80.83 ± 15.56
FLIP-Large	96.6	30.52	**77.21** ± 20.15	67.58 ± 17.21	**89.14** ± 8.57	**81.53** ± 13.28	**81.24** ± 17.89	90.45 ± 14.00	**81.19** ± 15.18
FLIP$_{bb}$-Tiny	**0.51**	**8.20**	66.86 ± 21.35	65.36 ± 18.14	80.29 ± 15.55	74.82 ± 15.86	73.81 ± 20.04	86.52 ± 16.68	74.61 ± 17.94
FLIP$_{bb}$-Small	2.3	9.85	68.47 ± 21.02	65.72 ± 17.93	82.56 ± 13.93	76.58 ± 15.08	75.79 ± 19.38	88.10 ± 16.30	76.20 ± 17.27
FLIP$_{bb}$-Middle	11.5	12.37	69.50 ± 20.76	65.56 ± 18.11	84.37 ± 12.50	77.82 ± 14.64	77.41 ± 18.68	88.60 ± 15.96	77.21 ± 16.78
FLIP$_{bb}$-Large	96.6	25.57	70.04 ± 20.61	65.29 ± 18.60	85.48 ± 11.45	78.22 ± 14.60	78.23 ± 18.19	88.60 ± 15.95	77.64 ± 16.57

Table 4. Mean IoU (%) and Std IoU (%) at 0% bounding-box prompt noise.

Model	Size (M)	Hypersim	KITTI-360	OpenImage	COCO	LVIS	ObjaScale (ours)	Average
		Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)
FastSAM-s	11.8	36.41 ± 34.56	37.52 ± 32.51	61.16 ± 32.09	46.15 ± 33.78	36.77 ± 35.18	45.74 ± 36.35	43.96 ± 34.08
FastSAM-x	72.2	42.94 ± 36.01	38.68 ± 33.85	69.19 ± 29.39	54.84 ± 34.85	42.86 ± 37.49	44.70 ± 36.95	48.87 ± 34.76
MobileSAM	10.13	68.11 ± 21.26	64.36 ± 18.90	82.20 ± 16.73	73.67 ± 17.46	71.55 ± 21.35	67.73 ± 25.03	71.27 ± 20.12
EfficientSAM-T	10.22	67.54 ± 24.83	60.50 ± 24.92	84.34 ± 15.01	75.74 ± 17.18	75.34 ± 20.09	68.74 ± 26.43	72.03 ± 21.41
EfficientSAM-S	26.41	69.44 ± 22.37	64.76 ± 19.24	86.07 ± 14.38	76.96 ± 16.95	74.97 ± 21.24	67.80 ± 26.76	73.33 ± 20.16
SAM-B	93.7	71.24 ± 20.97	62.15 ± 21.55	84.71 ± 15.33	75.96 ± 17.05	76.78 ± 19.24	71.38 ± 25.36	73.70 ± 19.92
SAM-L	312.3	71.98 ± 21.17	62.52 ± 20.35	86.89 ± 13.48	78.03 ± 16.25	77.83 ± 19.38	72.68 ± 25.23	74.99 ± 19.31
SAM-H	641.1	72.39 ± 21.51	62.36 ± 20.47	87.04 ± 13.58	78.31 ± 16.30	78.30 ± 19.41	73.75 ± 24.58	75.36 ± 19.31
SAM2-T	38.96	71.36 ± 20.55	66.58 ± 20.39	87.10 ± 12.67	78.42 ± 15.51	76.76 ± 19.31	69.88 ± 26.28	75.02 ± 19.12
SAM2-S	46.06	71.22 ± 21.36	66.06 ± 22.06	87.58 ± 12.72	78.73 ± 15.63	77.12 ± 19.31	69.68 ± 26.43	75.06 ± 19.58
SAM2-B+	80.85	71.95 ± 20.52	65.57 ± 19.35	87.97 ± 11.92	79.21 ± 15.30	77.26 ± 19.37	70.49 ± 26.53	75.41 ± 18.83
SAM2-L	224.45	72.84 ± 20.61	67.03 ± 20.89	**88.66** ± 11.24	**79.49** ± 15.13	77.15 ± 19.51	69.54 ± 27.00	75.78 ± 19.06
FLIP$_{bb}$-Tiny	**0.51**	69.00 ± 19.58	67.40 ± 17.28	83.05 ± 14.60	75.19 ± 16.03	75.11 ± 19.40	86.81 ± 16.35	76.09 ± 17.20
FLIP$_{bb}$-Small	2.3	71.26 ± 19.20	67.76 ± 17.37	85.68 ± 12.89	77.28 ± 15.51	77.69 ± 18.47	88.38 ± 15.92	78.01 ± 16.56
FLIP$_{bb}$-Middle	11.5	73.00 ± 18.69	**68.07** ± 17.76	87.68 ± 11.33	78.52 ± 15.91	79.62 ± 17.73	**88.98** ± 15.49	79.31 ± 16.15
FLIP$_{bb}$-Large	96.6	**73.54** ± 18.76	67.20 ± 18.82	88.60 ± 10.62	78.49 ± 16.57	**80.44** ± 17.34	88.91 ± 15.58	**79.53** ± 16.28

The heat-map analysis (Figure 7) offers clear evidence of FLIP's scale-invariant behavior: whereas SAM variants—and FastSAM-x in particular—suffer sharp performance drops when the mask-to-image ratio falls below 0.01%, FLIP's IoU remains stable until the object's absolute footprint shrinks to roughly 10×10 pixels. Because FLIP's accuracy is governed by absolute pixel coverage rather than relative image area, it robustly segments small objects even in ultra-high-resolution scenes—a critical advantage for real-world applications.

C. Prompt Noise Robustness Details

Table 4–Table 8 provide the per-dataset breakdown for the bounding-box noise experiment summarized in Table 2. All tables in this appendix use the fixed ablation subset described in the main text; accordingly, their "Clean" values are subset estimates and may differ slightly from the full-evaluation values in Table 1. The clean setting preserves the main box-prompted performance pattern, with FLIP$_{bb}$ competitive or strongest on several datasets. As prompt noise increases, the tables show a consistent degradation of FLIP$_{bb}$ across benchmarks, although the magnitude varies with object scale, clutter, and dataset composition. At larger perturbations, full-image encoders are more stable because their image representation is less tied to the prompt location, whereas FLIP$_{bb}$ uses the noisy box-derived Gaussian to determine both where to sample patches and where to query mask values.

Table 5. Mean IoU (%) and Std IoU (%) at 2.5% bounding-box prompt noise.

Model	Size (M)	Hypersim	KITTI-360	OpenImage	COCO	LVIS	ObjaScale (ours)	Average
		Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)
FastSAM-s	11.8	36.31 ± 34.57	37.44 ± 32.52	61.10 ± 32.12	46.09 ± 33.81	36.73 ± 35.17	45.74 ± 36.34	43.90 ± 34.09
FastSAM-x	72.2	42.79 ± 36.03	38.59 ± 33.86	69.10 ± 29.46	54.76 ± 34.89	42.79 ± 37.49	44.69 ± 36.95	48.79 ± 34.78
MobileSAM	10.13	67.36 ± 21.77	63.97 ± 19.15	81.97 ± 16.90	73.09 ± 17.86	71.05 ± 21.70	67.66 ± 25.06	70.85 ± 20.41
EfficientSAM-T	10.22	67.97 ± 23.87	61.91 ± 22.36	84.19 ± 15.28	74.88 ± 18.18	74.88 ± 20.52	68.80 ± 26.36	72.10 ± 21.09
EfficientSAM-S	26.41	69.13 ± 22.63	64.90 ± 19.21	85.91 ± 14.65	76.62 ± 17.27	74.66 ± 21.49	67.76 ± 26.76	73.16 ± 20.34
SAM-B	93.7	70.54 ± 21.67	61.81 ± 21.81	84.38 ± 15.73	75.06 ± 17.87	76.35 ± 19.62	71.32 ± 25.36	73.24 ± 20.34
SAM-L	312.3	71.41 ± 21.81	62.25 ± 20.62	86.76 ± 13.62	77.53 ± 16.77	77.52 ± 19.76	72.64 ± 25.22	74.68 ± 19.63
SAM-H	641.1	71.83 ± 22.17	62.08 ± 20.75	86.90 ± 13.86	77.95 ± 16.74	**77.98** ± 19.82	73.72 ± 24.59	75.08 ± 19.65
SAM2-T	38.96	70.31 ± 21.50	65.83 ± 21.13	86.83 ± 13.03	77.60 ± 16.16	76.08 ± 19.75	69.81 ± 26.27	74.41 ± 19.64
SAM2-S	46.06	70.25 ± 22.19	65.53 ± 22.48	87.39 ± 12.88	77.86 ± 16.54	76.44 ± 19.81	69.62 ± 26.42	74.52 ± 20.05
SAM2-B+	80.85	71.09 ± 21.27	65.08 ± 19.87	87.81 ± 12.20	78.53 ± 15.91	76.75 ± 19.77	70.43 ± 26.51	74.95 ± 19.26
SAM2-L	224.45	**72.05** ± 21.34	**66.51** ± 21.37	**88.48** ± 11.57	**78.74** ± 15.99	76.57 ± 20.00	69.49 ± 26.98	**75.31** ± 19.54
FLIP$_{bb}$-Tiny	**0.51**	64.90 ± 20.76	63.80 ± 17.97	77.57 ± 16.52	70.30 ± 17.25	69.51 ± 20.56	**81.11** ± 17.65	71.20 ± 18.45
FLIP$_{bb}$-Small	2.3	66.25 ± 21.06	63.72 ± 18.41	78.74 ± 16.87	71.50 ± 17.58	71.38 ± 20.34	80.80 ± 18.03	72.06 ± 18.72
FLIP$_{bb}$-Middle	11.5	66.32 ± 21.63	62.82 ± 19.69	76.19 ± 20.81	70.60 ± 19.80	71.50 ± 21.03	77.03 ± 20.38	70.74 ± 20.56
FLIP$_{bb}$-Large	96.6	65.64 ± 22.62	60.63 ± 21.86	72.95 ± 24.39	68.73 ± 21.80	70.45 ± 22.39	72.93 ± 22.86	68.55 ± 22.65

Table 6. Mean IoU (%) and Std IoU (%) at 5% bounding-box prompt noise.

Model	Size (M)	Hypersim	KITTI-360	OpenImage	COCO	LVIS	ObjaScale (ours)	Average
		Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)
FastSAM-s	11.8	36.08 ± 34.56	37.29 ± 32.54	60.90 ± 32.21	45.84 ± 33.85	36.52 ± 35.17	45.72 ± 36.34	43.72 ± 34.11
FastSAM-x	72.2	42.48 ± 36.06	38.41 ± 33.91	68.90 ± 29.61	54.43 ± 35.00	42.58 ± 37.49	44.68 ± 36.95	48.58 ± 34.84
MobileSAM	10.13	64.92 ± 23.43	62.35 ± 20.28	80.73 ± 17.82	70.91 ± 19.10	69.25 ± 22.78	67.42 ± 25.07	69.26 ± 21.41
EfficientSAM-T	10.22	65.73 ± 25.65	59.93 ± 23.75	83.18 ± 16.58	72.73 ± 19.94	73.15 ± 22.05	68.61 ± 26.37	70.56 ± 22.39
EfficientSAM-S	26.41	67.65 ± 23.70	63.50 ± 20.15	85.36 ± 15.16	75.31 ± 18.36	73.56 ± 22.35	67.62 ± 26.76	72.17 ± 21.08
SAM-B	93.7	67.98 ± 23.77	60.77 ± 22.36	82.53 ± 17.65	71.92 ± 20.44	74.30 ± 21.37	70.97 ± 25.33	71.41 ± 21.82
SAM-L	312.3	69.33 ± 23.62	61.25 ± 21.51	85.48 ± 15.21	75.16 ± 18.86	75.94 ± 21.23	72.27 ± 25.23	73.24 ± 20.94
SAM-H	641.1	**69.96** ± 23.90	61.14 ± 21.56	86.12 ± 14.98	76.16 ± 18.59	**76.72** ± 21.13	**73.51** ± 24.60	**73.94** ± 20.79
SAM2-T	38.96	67.60 ± 23.17	64.13 ± 21.87	85.04 ± 14.48	75.09 ± 17.80	73.80 ± 21.13	69.46 ± 26.20	72.52 ± 20.77
SAM2-S	46.06	67.41 ± 24.02	64.02 ± 23.07	85.49 ± 14.87	75.22 ± 18.56	74.09 ± 21.44	69.27 ± 26.40	72.58 ± 21.39
SAM2-B+	80.85	68.56 ± 22.98	63.65 ± 20.81	86.12 ± 14.06	76.14 ± 17.81	74.73 ± 21.21	70.07 ± 26.48	73.21 ± 20.56
SAM2-L	224.45	69.49 ± 23.22	**65.03** ± 22.21	**86.90** ± 13.51	**76.45** ± 18.11	74.51 ± 21.47	69.12 ± 26.96	73.58 ± 20.91
FLIP$_{bb}$-Tiny	**0.51**	58.17 ± 22.01	58.29 ± 19.19	68.73 ± 19.30	63.41 ± 18.61	62.81 ± 21.46	72.80 ± 19.81	64.04 ± 20.06
FLIP$_{bb}$-Small	2.3	58.78 ± 22.74	58.04 ± 19.65	67.76 ± 21.42	63.72 ± 19.63	63.58 ± 21.92	71.64 ± 20.44	63.92 ± 20.97
FLIP$_{bb}$-Middle	11.5	57.31 ± 24.01	55.77 ± 21.71	59.73 ± 27.00	60.39 ± 23.04	61.26 ± 23.91	62.58 ± 25.65	59.51 ± 24.22
FLIP$_{bb}$-Large	96.6	56.18 ± 25.09	52.55 ± 24.06	54.96 ± 29.54	57.52 ± 24.92	59.44 ± 25.45	57.63 ± 26.45	56.38 ± 25.92

Table 7. Mean IoU (%) and Std IoU (%) at 10% bounding-box prompt noise.

Model	Size (M)	Hypersim	KITTI-360	OpenImage	COCO	LVIS	ObjaScale (ours)	Average
		Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)
FastSAM-s	11.8	35.30 ± 34.55	36.52 ± 32.70	60.04 ± 32.65	44.80 ± 34.16	35.75 ± 35.14	45.57 ± 36.31	43.00 ± 34.25
FastSAM-x	72.2	41.39 ± 36.11	37.58 ± 34.10	67.68 ± 30.52	53.18 ± 35.53	41.43 ± 37.53	44.56 ± 36.94	47.64 ± 35.12
MobileSAM	10.13	57.43 ± 26.60	57.26 ± 22.87	71.71 ± 22.89	62.69 ± 22.99	62.10 ± 25.72	65.66 ± 24.83	62.81 ± 24.32
EfficientSAM-T	10.22	59.49 ± 28.94	54.68 ± 25.92	74.91 ± 23.91	65.14 ± 24.95	67.02 ± 26.41	67.26 ± 26.44	64.75 ± 26.09
EfficientSAM-S	26.41	62.56 ± 26.67	**60.03** ± 21.94	**80.01** ± 20.18	**69.58** ± 22.47	68.71 ± 25.55	66.71 ± 26.60	**67.93** ± 23.90
SAM-B	93.7	60.09 ± 27.88	55.22 ± 24.87	71.76 ± 25.83	62.06 ± 26.11	66.55 ± 26.38	68.64 ± 25.22	64.05 ± 26.05
SAM-L	312.3	61.53 ± 27.68	56.45 ± 24.53	74.07 ± 24.46	65.44 ± 25.20	68.43 ± 26.22	69.68 ± 25.09	65.93 ± 25.53
SAM-H	641.1	**62.92** ± 28.05	56.81 ± 24.52	77.77 ± 22.98	68.02 ± 24.68	**70.15** ± 26.11	**71.77** ± 24.57	67.91 ± 25.15
SAM2-T	38.96	58.35 ± 26.24	57.65 ± 23.26	74.67 ± 21.62	65.32 ± 22.75	65.16 ± 24.74	66.68 ± 25.71	64.64 ± 24.05
SAM2-S	46.06	57.76 ± 27.26	57.20 ± 24.85	74.30 ± 24.01	65.12 ± 24.39	64.98 ± 25.55	66.32 ± 26.07	64.28 ± 25.36
SAM2-B+	80.85	59.71 ± 26.21	57.69 ± 23.30	75.82 ± 22.21	66.86 ± 23.26	66.50 ± 25.03	67.29 ± 26.09	65.65 ± 24.35
SAM2-L	224.45	60.28 ± 27.23	58.76 ± 24.18	76.91 ± 22.49	66.77 ± 24.22	65.90 ± 25.58	66.41 ± 26.63	65.84 ± 25.06
FLIP$_{bb}$-Tiny	**0.51**	46.38 ± 22.55	47.31 ± 20.51	52.46 ± 21.80	50.17 ± 20.29	49.96 ± 22.21	56.51 ± 22.50	50.47 ± 21.64
FLIP$_{bb}$-Small	2.3	46.25 ± 23.37	46.83 ± 21.11	49.66 ± 24.48	49.73 ± 21.57	49.55 ± 23.16	54.45 ± 23.51	49.41 ± 22.87
FLIP$_{bb}$-Middle	11.5	43.98 ± 24.43	43.12 ± 23.15	40.25 ± 27.06	45.01 ± 24.17	45.85 ± 24.85	40.88 ± 27.65	43.18 ± 25.22
FLIP$_{bb}$-Large	96.6	43.27 ± 25.19	39.62 ± 24.81	38.33 ± 27.53	43.30 ± 24.74	44.78 ± 25.54	42.57 ± 25.08	41.98 ± 25.48

D. Architectural Robustness and Ablation Analysis

Our ablation study (Table 9) reveals that FLIP is remarkably robust to architectural variations while certain design choices provide measurable improvements. Unless stated otherwise, all ablations are evaluated without any prompt-specific fine-tuning. The baseline FLIP-Small (79.29% mean IoU) serves as our reference point for evaluating architectural components.

Table 8. Mean IoU (%) and Std IoU (%) at 20% bounding-box prompt noise.

Model	Size (M)	Hypersim	KITTI-360	OpenImage	COCO	LVIS	ObjaScale (ours)	Average
		Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)
FastSAM-s	11.8	32.51 ± 34.24	33.48 ± 33.00	55.37 ± 34.80	40.24 ± 34.75	32.73 ± 34.82	44.93 ± 36.17	39.88 ± 34.63
FastSAM-x	72.2	37.51 ± 36.02	34.12 ± 34.36	**61.88** ± 34.09	47.09 ± 37.00	37.32 ± 37.26	43.87 ± 36.80	43.63 ± 35.92
MobileSAM	10.13	39.15 ± 28.35	40.59 ± 25.90	47.18 ± 30.49	41.27 ± 27.98	42.29 ± 28.68	55.56 ± 26.10	44.34 ± 27.92
EfficientSAM-T	10.22	42.47 ± 31.72	38.16 ± 28.42	47.97 ± 34.23	43.05 ± 31.16	48.05 ± 32.04	59.87 ± 28.14	46.59 ± 30.95
EfficientSAM-S	26.41	**48.02** ± 30.33	**45.48** ± 26.39	57.35 ± 31.11	**49.73** ± 30.01	**52.43** ± 30.80	**61.41** ± 27.20	**52.40** ± 29.31
SAM-B	93.7	40.82 ± 30.67	36.70 ± 27.32	43.31 ± 34.02	39.27 ± 30.66	44.96 ± 31.56	57.47 ± 27.32	43.76 ± 30.26
SAM-L	312.3	42.59 ± 31.31	39.57 ± 28.06	45.54 ± 34.24	41.90 ± 31.71	47.04 ± 32.28	58.54 ± 27.34	45.86 ± 30.82
SAM-H	641.1	43.21 ± 31.72	39.92 ± 28.44	49.23 ± 34.62	43.63 ± 32.21	48.04 ± 32.62	60.56 ± 27.09	47.43 ± 31.12
SAM2-T	38.96	38.23 ± 27.53	38.41 ± 25.50	48.21 ± 31.18	41.68 ± 28.41	43.22 ± 28.18	54.81 ± 27.05	44.09 ± 27.97
SAM2-S	46.06	37.06 ± 28.14	37.58 ± 26.42	45.42 ± 33.37	40.20 ± 29.65	42.01 ± 28.93	51.68 ± 28.11	42.33 ± 29.10
SAM2-B+	80.85	40.14 ± 28.10	38.74 ± 26.47	49.41 ± 31.98	43.20 ± 29.28	44.83 ± 28.88	55.40 ± 27.66	45.29 ± 28.73
SAM2-L	224.45	39.94 ± 29.53	39.42 ± 26.77	48.47 ± 33.47	42.06 ± 30.46	43.66 ± 29.33	54.39 ± 27.77	44.66 ± 29.55
FLIP$_{bb}$-Tiny	**0.51**	29.63 ± 20.36	30.58 ± 19.64	30.51 ± 20.42	30.70 ± 19.48	31.00 ± 20.06	32.16 ± 21.21	30.76 ± 20.20
FLIP$_{bb}$-Small	2.3	29.20 ± 21.11	29.78 ± 20.20	28.59 ± 22.17	30.22 ± 20.63	30.33 ± 20.95	30.57 ± 22.56	29.78 ± 21.27
FLIP$_{bb}$-Middle	11.5	27.74 ± 21.56	27.01 ± 21.13	23.89 ± 21.95	27.20 ± 21.20	27.95 ± 21.32	19.86 ± 21.15	25.61 ± 21.39
FLIP$_{bb}$-Large	96.6	27.75 ± 22.23	25.20 ± 21.71	24.50 ± 22.44	27.13 ± 21.55	27.95 ± 21.78	25.53 ± 21.64	26.34 ± 21.89

Positional Embedding Strategies. The comparison between baseline and "InitialPosEmb" (79.23% vs. 79.29%) shows that our layer-wise positional embedding injection provides marginal but consistent benefits over initial embedding only. However, the "LaPE-SNorm" variant, which applies shared normalization across positional embeddings, achieves the best performance at 79.50% mean IoU, while underperforming in the hierarchical inference setting.

Patch Resolution Configurations. Testing different patch size ranges reveals interesting trade-offs. Restricting maximum patch size to 4×4 ("MaxPatch4x4") reduces parameters to 1.7M with only modest performance degradation (78.81% IoU), demonstrating FLIP's efficiency even with limited resolution diversity. Conversely, extending to 64×64 patches ("MaxPatch64x64") slightly improves performance (79.44%) at the cost of increased parameters (3.0M), suggesting diminishing returns from very large patches.

Removing the smallest 1×1 patches ("MinPatch4x4") significantly impacts performance on LVIS (72.08% vs. 77.92%), highlighting the importance of fine-grained detail capture for complex segmentation tasks, while maintaining strong performance on other datasets.

Mask Target Pixel Sampling Strategy Impact. The "NoDynamicSampling" ablation (78.63% vs. 79.29%) demonstrates that our adaptive target pixel sampling strategy provides consistent benefits across datasets. Without dynamic sampling, each distance group uses the same number of pixels rather than adaptively redistributing based on prediction difficulty. The performance gap is particularly notable on Hypersim (71.03% vs. 73.07%), where focusing training on challenging boundary regions better handles the diverse object shapes in simulated indoor scenes.

Table 9. FLIP-Small ablation study

Ablation	Params (M)	Time (ms)	Hypersim	KITTI-360	OpenImages	COCO	LVIS	ObjaScale (ours)	Average
			Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
Baseline	2.3	12.19	73.07 ± 19.80	69.91 ± 15.69	86.65 ± 11.27	79.23 ± 13.32	77.92 ± 18.34	88.96 ± 15.14	79.29 ± 15.59
Baseline$_h$	2.3	10.94	71.93 ± 18.99	68.34 ± 15.57	85.73 ± 11.29	78.15 ± 13.04	76.71 ± 17.41	86.97 ± 14.68	77.97 ± 15.16
InitialPosEmb	2.2	10.37	73.41 ± 19.73	68.96 ± 15.48	86.29 ± 11.54	79.22 ± 13.14	**78.42** ± 17.98	89.05 ± 14.98	79.23 ± 15.47
InitialPosEmb$_h$	2.2	8.98	72.03 ± 18.92	67.36 ± 15.27	85.45 ± 11.51	77.94 ± 12.98	76.92 ± 17.15	86.84 ± 14.63	77.76 ± 15.08
LaPE-SNorm	2.3	10.88	**73.86** ± 19.37	69.92 ± 15.20	86.41 ± 11.40	79.29 ± 13.12	78.42 ± 18.00	**89.12** ± 14.77	**79.50** ± 15.31
LaPE-SNorm$_h$	2.3	**8.79**	71.03 ± 17.74	65.87 ± 13.78	80.20 ± 11.98	74.42 ± 12.71	74.34 ± 16.30	83.01 ± 14.61	74.81 ± 14.52
MaxPatch4x4	**1.7**	11.46	72.69 ± 19.71	69.43 ± 15.57	86.01 ± 11.55	78.73 ± 13.28	77.55 ± 18.30	88.48 ± 14.96	78.81 ± 15.56
MaxPatch4x4$_h$	**1.7**	10.09	71.45 ± 18.97	67.87 ± 15.46	85.11 ± 11.47	77.61 ± 13.07	76.31 ± 17.43	86.55 ± 14.57	77.48 ± 15.16
MaxPatch64x64	3.0	12.54	73.45 ± 19.52	69.78 ± 15.93	86.67 ± 11.37	**79.33** ± 13.15	78.39 ± 18.02	89.04 ± 14.89	79.44 ± 15.48
MaxPatch64x64$_h$	3.0	11.10	72.19 ± 18.80	68.21 ± 15.80	85.73 ± 11.32	78.16 ± 12.91	77.04 ± 17.08	87.05 ± 14.41	78.06 ± 15.05
MinPatch4x4	2.2	11.45	69.39 ± 23.77	**70.30** ± 15.60	86.42 ± 11.60	77.27 ± 16.62	72.08 ± 25.85	88.14 ± 17.28	77.27 ± 18.45
MinPatch4x4$_h$	2.2	10.07	68.53 ± 22.58	68.82 ± 15.46	85.53 ± 11.60	76.31 ± 16.23	71.07 ± 24.74	86.05 ± 16.78	76.05 ± 17.90
NoDynamicSampling	2.3	12.17	71.03 ± 20.14	70.03 ± 15.46	**87.47** ± 11.07	79.04 ± 13.65	76.40 ± 19.00	87.82 ± 15.95	78.63 ± 15.88
NoDynamicSampling$_h$	2.3	10.69	70.57 ± 19.46	68.54 ± 15.26	86.63 ± 11.07	78.29 ± 13.38	75.89 ± 18.11	86.66 ± 15.20	77.76 ± 15.41

Figure 10. Examples of multi-resolution patch inputs (top row) and corresponding mask predictions (bottom row) from FLIP. Input patches are color-coded by size: purple (16×16), yellow (8×8), green (4×4), blue (2×2), and red (1×1). Higher-resolution patches focus on object centers for detail, while lower-resolution patches cover peripheral regions for efficiency. Mask predictions show accurate segmentation with optimized resource allocation.

Table 10. Non-finetuned 2d Gaussian prompted FLIP models (standard and hierarchical).

Model	Size (M)	Time (ms)	Hypersim	KITTI-360	OpenImage	COCO	LVIS	ObjaScale (ours)	Average
			Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)	Mean ± Std IoU (%)
FLIP-Tiny	**0.51**	9.82	73.17 ± 19.36	**70.04** ± 15.27	83.83 ± 12.72	77.76 ± 13.55	76.88 ± 18.33	87.76 ± 14.93	78.24 ± 15.69
FLIP-Small	2.3	12.19	73.07 ± 19.80	69.91 ± 15.69	86.65 ± 11.27	79.23 ± 13.32	77.92 ± 18.34	88.96 ± 15.14	79.29 ± 15.59
FLIP-Middle	11.5	17.54	**73.68** ± 20.13	69.86 ± 15.76	87.89 ± 10.08	80.16 ± 13.15	78.88 ± 18.31	**89.08** ± 15.04	79.93 ± 15.41
FLIP-Large	96.6	38.65	73.66 ± 20.51	69.75 ± 16.03	**89.33** ± 8.76	**80.79** ± 13.06	**79.42** ± 18.13	89.02 ± 15.21	**80.33** ± 15.28
$FLIP_h$-Tiny	**0.51**	**9.05**	71.36 ± 18.65	68.53 ± 15.08	83.03 ± 12.65	76.57 ± 13.25	75.41 ± 17.37	85.72 ± 14.52	76.77 ± 15.25
$FLIP_h$-Small	2.3	10.94	71.93 ± 18.99	68.34 ± 15.57	85.73 ± 11.29	78.15 ± 13.04	76.71 ± 17.41	86.97 ± 14.68	77.97 ± 15.16
$FLIP_h$-Middle	11.5	14.34	72.65 ± 19.21	68.28 ± 15.68	86.98 ± 10.01	78.98 ± 12.89	77.58 ± 17.25	86.95 ± 14.68	78.57 ± 14.95
$FLIP_h$-Large	96.6	29.36	73.19 ± 19.56	68.05 ± 16.18	88.28 ± 8.96	79.67 ± 12.83	78.40 ± 17.10	87.36 ± 14.61	79.16 ± 14.87

E. Cross-Dataset Generalization

FLIP demonstrates robust generalization across diverse evaluation benchmarks without dataset-specific fine-tuning. The performance gains are consistent across synthetic (Hypersim), automotive (KITTI-360), and large-scale natural image datasets (OpenImages, COCO, LVIS). Notably, FLIP-Tiny achieves 72.62% IoU on KITTI-360 compared to SAM-H's 62.47%, despite using 1,257× fewer parameters. This 10.15 percentage point improvement highlights how FLIP's fovea-like attention mechanism effectively handles automotive scenarios characterized by numerous small, distant objects.

The architectural ablations confirm that FLIP's performance is remarkably robust to design variations. While removing the smallest patches (MinPatch4x4) impacts performance on complex datasets like LVIS, and the adaptive pixel sampling during training (NoDynamicSampling ablation) provides consistent improvements, the overall architecture maintains strong performance across different configurations. This robustness suggests that FLIP's core fovea-like attention mechanism, rather than any single architectural choice, drives its superior accuracy, particularly for challenging small objects.

F. Hierarchical Refinement

Procedure For further efficiency, we introduce a hierarchical inference strategy that progressively refines a 5-sigma bounding-box mask prediction. Starting from a coarse resolution, we predict initial mask values on a coarse grid and identify uncertain regions where predictions fall within $[\tau, 1-\tau]$ (default $\tau = 0.01$). The mask is then upsampled by a factor α (default $\alpha = 4$) using bilinear interpolation, but only the uncertain pixels are re-queried at the higher resolution. This process repeats iteratively until the target resolution is reached, focusing refinement exclusively on ambiguous boundary regions. The number of refinement steps n is automatically determined as $n = \lfloor \log_\alpha(k_{\text{target}}/k_{\text{init}}) \rfloor$, where k_{init} and k_{target} are the initial and target resolutions.

Empirical Trade-offs. The hierarchical inference variants ($FLIP_h$) demonstrate the effectiveness of our progressive refinement strategy, achieving comparable performance with reduced computational cost. Comparing standard inference

with hierarchical variants reveals a general pattern: hierarchical inference achieves 98–99% of the original performance while providing $\approx$ 8–24% speed improvements. For instance, FLIP$_h$-Large maintains 79.16% mean IoU while reducing inference time to 29.36ms. This trade-off is favorable for real-time applications where slight accuracy reductions are acceptable for substantial computational savings.

Non-finetuned Gaussian Variants. Table 10 summarizes the Gaussian-query FLIP and FLIP$_h$ variants. These results reuse the checkpoints from the main training before prompt-specific finetuning, isolating the effect of the progressive inference scheme.