MOBIUS: Big-to-Mobile Universal Instance Segmentation via Multi-modal Bottleneck Fusion and Calibrated Decoder Pruning

Supplementary Material

We here report additional implementation details (Sec. 6) and state-of-the-art comparison on additional datasets (Sec. 7). Moreover, we extend our ablation study and include analysis on the component-wise efficiency (Sec. 8.1), the different mobile encoders and the relative computational complexity of our decoders (Sec. 8.2), the FLOPs at low image resolution (Sec. 8.3), decoder design choices (Sec. 8.4), the effect of calibration on decoder pruning (Sec. 8.5), and different confidence trajectory functions (Sec. 8.6). Finally, we provide qualitative results for the different tasks supported by our foundational universal instance segmentation model (Sec. 9).

6. Implementation Details

Datasets. In Sec. 4.1, we have described the datasets that we used for training our model. We here report additional details in table Tab. 5. Notice that, unlike GLEE [58], MOBIUS is trained in a single stage across all listed datasets. The table also reports the sampling ratio for each dataset. Following GLEE, to ensure that objects from SA1B are at the object-level rather than the part-level, we apply mask IoU based NMS and use area as NMS score to eliminate part-level object annotations.

Additional Training Details. To ensure full reproducibility of our approach, we here report additional training details to the ones reported in Sec. 4.1. In particular, we train our model for 500,000 iterations on the joint set of datasets listed in Tab. 5. We use the AdamW [33] optimizer with learning rate 10^{-4} and weight decay of 0.05. We decay the learning rate twice by a factor of 0.1 after 400k and 500k iterations respectively. The learning rates of the image encoder and text encoder are multiplied by a factor of 0.1. We use multi-scale augmentation, and resize the input images such that the shortest side is at least 384 and at most 800 pixels while the longest at most 1333.

7. Additional State-of-the-art Comparisons

Low-resolution evaluation. For completeness, we provide the low-resolution performance of our big models (Tab. 6), so that they can be fairly compared to our mobile models in Tab. 1. This analysis further demonstrate the adaptability of MOBIUS models. At 89G FLOPs, **MOBIUS-3 (low-res)** achieves a COCO-val $\mathrm{AP_b}$ of 50.1 and LVIS $\mathrm{AP_b}$ of 40.1, with a modest performance drop compared to its high-resolution counterpart (COCO-val $\mathrm{AP_b}$ of 57.8 and LVIS $\mathrm{AP_b}$ of 50.3 at 456G FLOPs). Lower-tier models, such as **MOBIUS-0 (low-res)**, operate at just 33G FLOPs while maintaining competitive performance (COCO-val $\mathrm{AP_b}$ of 46.2). Nevertheless, the smallest big model still requires almost twice as many FLOPs as our mobile model based on MNv4-conv-M (Tab. 1, d). These results highlight the suitability of MOBIUS models for resource-constrained platforms, such as mobile and edge devices.

	Sizes		Annotations			Sampling Ratio
dataset	images	objects	semantic	box	mask	
Detection Data						
Objects365 [49]	1817287	26563198	category	✓	-	1.5
OpenImages [20]	1743042	14610091	category	✓	-	1.5
LVIS [12]	100170	1270141	category	✓	✓	1.5
COCO [30]	118287	860001	category	✓	✓	1.5
BDD [47]	69863	1274792	category	✓	✓	0.15
Grounding Data						
RefCOCO [37]	16994	42404	description	✓	✓	
RefCOCOg [37]	21899	42226	description	✓	✓	2.5†
RefCOCO+ [37]	16992	42278	description	✓	✓	
VisualGenome [19]	77396	3596689	description	✓	-	2
OpenWorld Data						
UVO [56]	16923	157624	-	✓	✓	0.2
SA1B [17]	2147712‡	99427126	-	✓	✓	2.5
Video Data						
YTVIS19 [64]	61845	97110	category	✓	✓	0.3
YTVIS21 [64]	90160	175384	category	✓	✓	0.3
OVIS [38]	42149	206092	category	✓	✓	0.3
RefVOS [48]	93857	159961	description	✓	✓	0.3

Table 5. **Training Datasets.** The datasets used to train MOBIUS and the corresponding sampling ratio. We here process each frame in video datasets independently. †: sampling ratio of the joint set including all RefCOCO datasets; ‡: we train on a subset of 500k images from the SA1B dataset.

	Method	FLOPs (G)	*Generic Detection & Segmentation*						*Zero-shot*
			COCO-val		LVIS				ODinW
			$\mathrm{AP_{box}}$	$\mathrm{AP_{mask}}$	$\mathrm{AP_{box}}$	$\mathrm{AP^r_{box}}$	$\mathrm{AP_{mask}}$	$\mathrm{AP^r_{mask}}$	$\mathrm{AP_{box}}$
Low-res	GLEE-Lite [58]	59	47.2	42.1	35.0	31.9	31.2	23.0	**40.5**
	MOBIUS-3	89	**50.1**	**45.4**	**40.1**	**36.8**	**37.3**	**34.7**	**40.5**
	MOBIUS-2	53	48.7	43.4	36.1	31.1	33.8	29.8	39.1
	MOBIUS-1	41	**47.4**	**42.3**	**35.2**	**32.6**	**32.8**	**32.4**	40.5
	MOBIUS-0	**33**	46.2	41.5	33.7	28.0	31.4	27.3	**43.8**

Table 6. **Comparison of big models at low-res.** We compare MOBIUS to GLEE [59] on object-level image tasks at low-resolution, rescaling the images to 384 on their short side while preserving aspect ratio. The models are ranked by descending FLOPs and divided into groups with similar FLOPs count. FLOPs are computed at 384x384 resolution, omitting the text encoder.

RefCOCO - Referring Object Detection and Segmentation. We report a state-of-the-art comparison on the RefCOCO, RefCOCO+ and RefCOCOg datasets in Tab. 7. For each dataset, we report the P@0.5 and the oIoU. We

	Method	RefCOCO		RefCOCO+		RefCOCOg	
		P@0.5	oIoU	P@0.5	oIoU	P@0.5	oIoU
Specialist	MDETR [16]	87.5	-	81.1	-	83.4	-
	SeqTR [75]	87.0	71.7	78.7	63.0	82.7	64.7
	PolyFormer (L) [31]	90.4	76.9	85.0	72.2	85.8	71.2
Generalist	UniTAB (B) [65]	88.6	-	81.0	-	84.6	-
	OFA (L) [55]	90.1	-	85.8	-	85.9	-
	UNINEXT (L) [28]	91.4	80.3	83.1	70.0	86.9	73.4
	UNINEXT (H) [28]	92.6	82.2	85.2	72.5	88.7	74.7
Foundation	GLEE-Plus [58]	90.6	79.5	81.6	68.3	85.0	70.6
	GLEE-Lite [58]	88.5	77.4	78.3	64.8	82.9	68.8
	MOBIUS-3	87.5	75.6	77.2	63.1	80.4	65.6
	MOBIUS-2	86.1	73.6	75.7	61.0	79.2	63.5
	MOBIUS-1	85.8	73.2	73.1	59.0	77.4	62.1
	MOBIUS-0	85.1	71.9	73.4	58.5	77.4	61.5
	MOBIUS-R50	86.6	74.3	76.0	61.5	79.5	64.2

Table 7. Comparison of methods on RefCOCO, RefCOCO+, and RefCOCOg datasets.

find that, despite the decreased number of FLOPs, our model remains effective in grounding referring expressions. However, we want to highlight that, while switching from ResNet-50 to FasterViT variants allowed us to leverage a more edge-friendly architecture, it seems that FasterViT provides a worse initialization for the referring tasks. We indeed report the performance of a MOBIUS variant trained with R50 and find that, despite having a number of FLOPs comparable to MOBIUS-0, it achieves much higher referring performance. We hope that this insight will guide future researchers towards choosing more suitable vision encoder initializations for referring and grounding.

ODinW - Zero-shot Object Detection. We report a state-of-the-art comparison on 13 ODinW [26] datasets in Tab. 9, benchmarking the zero-shot generalization of our models for the object detection task. We find that our model remains competitive with GLEE-Lite while achieving better efficiency, with MOBIUS-3 even outperforming GLEE-Lite (44.0 vs 43.2 average box AP)

SegInW - Zero-shot Instance Segmentation. We report a state-of-the-art comparison on 22 SegInW [79] datasets in Tab. 8, benchmarking the zero-shot generalization of our models for the instance segmentation task. Remarkably, we find that our model outperforms all prior methods (43.9 average mask AP) already in its smallest size MOBIUS-0.

8. Additional Ablation Studies

8.1. Component-wise Efficiency Analysis

In Tab. 10, we report the component-wise numerical FLOPs values used to generate Fig. 2.

8.2. Mobile Encoders

We show in Tab. 11 that further downscaling can be allowed by switching the vision encoder from FasterViT [13] to MobileNetv4 [39]. While FasterViT has been optimized for performance / throughput trade-off on high-end and edge GPUs, different versions of MobileNetv4 have also been optimized for performance / throughput trade-off on different mobile devices. As can be seen from our comparison, MobileNetv4 variants require significantly less FLOPs. Nevertheless, despite the larger FLOPs count, FasterViT retains good latency and provides significantly better detection performance. For this reason, we prefer leveraging the efficient FasterViT in our experiments in the main paper so to fairly compete with GLEE-Lite. Nevertheless, the results in Tab. 11 show that further downscaling of our model can be enabled by using one of the MobileNetv4 architectures, trading off performance for less compute requirements.

8.3. Low-resolution FLOPs

In Tab. 12 we compare the FLOPs requirements of different MOBIUS variants and GLEE under the low-resolution setting, where images are rescaled to 384 on their short side while preserving aspect ratio. The results show that the computational complexity of our pixel decoder and transformer scales down nicely with the input image size, still resulting in less FLOPs than the corresponding vision encoders (except for MOBIUS-0). Moreover, even at smaller resolution, using our bottleneck encoder as pixel decoder results in only 41% of GLEE's pixel decoder FLOPs. Finally, thanks to our single-scale processing, our transformer decoder only takes 50% on GLEE's.

8.4. Decoder Design

In Tab. 13 we ablate on different design choices for our pixel decoder. In particular, we ablate on the COCO dataset on the effect on FLOPs and performance of: type of self-attention used, bottleneck size, number of pixel decoder layers, whether to use single or multiple scales in the transformer decoder. We find that: (i) deformable self-attention - enabled by our smart design of the bottleneck representation as an individual scale from the feature scale pyramid - achieves the same performance as standard self-attention but with a significantly lower FLOPs count; (ii) the bottleneck size, measured according to the feature stride selected, saturates at stride 16, with the smaller stride 32 resulting in lower performance but better efficiency; (iii) the performance can greatly vary based on the number of pixel decoder layers, and we thus advise practitioners to choose the number of layers based on their computational budget; (iv) thanks to the multi-modal and multi-scale fusion happening within our pixel decoder, leveraging a single scale or multiple scales in the transformer decoder does not result in a significant difference, and we thus advise to use a single scale to improve efficiency.

Method	Brain Tumor	Chicken	Cows	Electric Shaver	Elephants	Fruits	Garbage	Ginger Garlic	Hand	Hand Metal	HouseHold Items	NutterflySquirrel	Phones	Poles	Puppies	Rail	Salmon Fillet	Strawberry	Tablets	Toolkits	Trash	Watermelon	Avg
X-Decoder(L) [79]	2.2	8.6	44.9	7.5	66.0	79.2	33.0	11.6	75.9	42.1	53.0	68.4	15.6	20.1	59.0	2.3	19.0	67.1	22.5	9.9	22.3	13.8	32.3
OpenSEED(L) [70]	2.1	82.9	40.9	4.7	72.9	76.4	16.9	13.6	92.7	38.7	50.0	40.0	7.6	4.6	74.6	1.8	15.6	82.8	47.4	15.4	15.3	52.3	36.1
ODISE(L) [61]	2.9	84.1	41.6	18.3	74.9	81.3	39.8	23.0	41.4	51.4	60.4	71.9	43.8	0.4	65.4	2.8	30.2	79.9	9.1	15.0	28.6	37.5	38.7
SAN(L) [62]	2.6	69.2	44.0	11.4	67.4	77.4	46.5	23.3	88.8	62.9	60.1	82.2	10.4	1.8	60.1	2.9	20.0	81.8	35.1	31.2	41.4	43.5	41.4
HIPIE(H) [57]	1.9	46.5	50.1	76.1	68.6	61.1	31.2	24.3	94.2	64.0	53.4	79.7	7.0	6.7	64.6	2.2	41.8	81.5	8.8	17.9	31.2	50.6	41.2
UNINEXT(L) [63]	2.6	75.2	52.1	71.2	72.1	81.1	16.9	23.7	93.7	57.0	54.0	84.1	6.1	13.4	64.6	0.0	44.4	80.7	21.0	10.1	10.8	56.3	42.1
MOBIUS-3	15.4	75.5	40.5	14.1	78.1	82.2	15.9	40.5	32.5	29.6	51.8	83.0	26.2	6.2	67.1	2.2	26.5	78.9	4.3	9.6	32.0	64.9	39.9
MOBIUS-2	1.0	65.8	23.8	0.4	75.7	74.3	16.1	40.7	94.9	73.4	28.4	77.5	54.8	12.6	68.3	12.9	31.8	83.8	1.7	25.0	29.9	44.8	**42.6**
MOBIUS-1	3.1	63.2	23.3	0.9	72.6	84.0	17.0	39.9	94.3	58.7	31.7	73.9	53.5	23.7	62.8	6.4	19.0	82.8	1.5	18.7	33.9	57.6	41.9
MOBIUS-0	4.4	79.4	29.9	0.1	76.2	77.7	14.7	33.4	92.2	72.7	54.0	70.2	40.5	22.2	70.0	7.4	24.0	88.3	0.3	15.6	29.1	62.8	**43.9**

Table 8. Results on SeginW benchmark across 22 datasets. We report the AP mask.

Model	PascalVOC	AerialDrone	Aquarium	Rabbits	EgoHands	Mushrooms	Packages	Raccoon	Shellfish	Vehicles	Pistols	Pothole	Thermal	Avg
GLIP-T [69]	56.2	12.5	18.4	70.2	50.0	73.8	72.3	57.8	26.3	56.0	49.6	17.7	44.1	46.5
GLIP-L [69]	61.7	7.1	26.9	75.0	45.5	49.0	62.8	63.3	68.9	57.3	68.6	25.7	66.0	52.1
GLEE-Plus [58]	67.8	10.8	38.3	76.1	47.4	19.2	29.4	63.8	66.7	63.8	62.6	15.3	66.5	48.3
GLEE-Lite [58]	61.7	7.9	23.2	72.6	41.9	51.6	32.9	51.1	35.0	59.4	45.6	21.8	56.9	43.2
MOBIUS-3	67.1	18.9	31.7	54.7	35.8	40.0	51.2	48.6	45.8	60.1	62.6	12.3	43.1	**44.0**
MOBIUS-2	64.9	13.0	23.2	75.6	12.9	18.1	62.9	50.1	37.2	61.1	49.0	17.7	40.4	40.5
MOBIUS-1	64.5	11.1	17.9	79.1	10.5	29.6	65.1	50.7	27.2	61.5	54.5	15.2	40.3	40.5
MOBIUS-0	63.3	13.7	24.7	75.5	6.0	39.0	47.5	53.4	34.8	61.1	55.7	20.4	34.1	**40.7**

Table 9. Zero-shot performance on 13 ODinW datasets.

Method	Pix. Dec. Type	FLOPs (G)				
		Vis. Enc.	Pix. Dec.	(+Modality Fusion)	Decoder	Total
GLEE† [59]	MaskDINO [23]	52.4	138	28.2	20.1	238.9
GLEE† [59]	RT-DETR [73]	52.4	69.2	**1.6**	20.0	143.1
MOBIUS (Ours)	Bottleneck	52.4	**61.4**	5.6	**10**	**129.8**

Table 10. **Component-wise Efficiency Analysis.** We compare the computational cost of MOBIUS and GLEE [59] variants using MaskDINO [23] or RT-DETR [73] decoders. FLOPs are reported for the vision encoder, pixel decoder, modality fusion, and decoder. All models use an R50 vision encoder at 800×800 resolution, excluding the text encoder from the total FLOPs count.

8.5. Effect of Uncertainty Calibration on Query Pruning

In Tab. 14, we investigate the effect of uncertainty calibration on query pruning on the COCO dataset. Importantly, we find that uncertainty calibration enables more meaningful differentiation of relevant vs. irrelevant queries, enabling better performance when applying query pruning at inference time.

	Vision Encoder	Vision Encoder Efficiency		COCO-val	
		FLOPs (G)	Latency (ms)	AP_{box}	AP_{mask}
MobileNetv4	MobileNetv4-conv-small	3	25.4	39.0	35.4
	MobileNetv4-conv-medium	15	39.0	43.6	39.2
	MobileNetv4-conv-large	38	48.4	47.2	42.3
	MobileNetv4-hybrid-medium	17	58.5	44.6	40.2
	MobileNetv4-hybrid-large	44	66.8	46.9	41.9
FasterViT	FasterViT-0	66	61.5	45.2	40.9
	FasterViT-1	105	72.3	46.3	41.9
	FasterViT-2	170	85.3	48.2	43.4
	FasterViT-3	358	99.8	49.3	44.5

Table 11. **Mobile encoders comparison.** We compare the latency, FLOPs, and performance on COCO val of MOBIUS models trained on COCO following the 1x schedule using MobileNetv4 and FasterViT image encoders. We report Average Precision (AP) for box and mask predictions. The latency (in ms) is measured on one NVIDIA A100 with the images resized to 800 on their shorter side while preserving aspect ratio.

8.6. Confidence Trajectory Functions

In Tab. 15 we investigate the effect of different confidence trajectories for our query pruning strategy. As explained in Sec. 3.2, our query pruning strategy relies on a threshold that increases layer-by-layer following a sigmoidal trajectory. We here compare to a logarithmic and exponential trajectory. Each strategy results in a different increase steepness for the confidence threshold at different layers. Empirically, we find that the sigmoidal trajectory, which enables slower increase at the beginning and end of the decoder with a steeper increase in the middle layers, works slightly better under its most FLOPs-efficient setting.

Exponential Interpolation Exponential interpolation gradually increases the confidence threshold in an exponential manner. This method is particularly useful when you

Model	FLOPs (G)					
	Text Encoder	Vision Encoder	Pixel Decoder w/o	Pixel Decoder w/	Decoder	Total w/
GLEE-Plus [59]	239	146	49.6	59.5	9.9	454.4
GLEE-Lite [59]	239	**16.1**	50	59.9	9.9	324.9
MOBIUS-3	239	90.5	19.8	24.7	4.9	354.2
MOBIUS-2	239	43.1	19.7	**24.6**	**4.9**	**311.6**
MOBIUS-1	239	29	18.7	23.6	4.9	296.5
MOBIUS-0	239	**16.7**	18.6	**23.5**	**4.9**	**278.1**

Table 12. **Low-resolution FLOPs comparison.** We compare the FLOPs for each model component in GLEE and MOBIUS. Notice that the text encoder is a fixed cost that can be removed by caching in most applications. We report its cost for processing the 80 COCO categories. We evaluate all models on low-resolution images rescaled to 384 on their short side while preserving aspect ratio. We compare the pixel decoder w/ and w/o early vision-language fusion.

Self-attn Type	Bottleneck Size	Layers	Scales	FLOPs (G)	COCO-val AP_{box}	COCO-val AP_{mask}
No	16	6	Single	410	44.0	39.8
Standard	16	6	Single	432	45.4	40.8
Deformable	16	6	Single	**413**	**45.5**	**41.1**
	32	6	Multi	**399**	43.9	39.5
Deformable	16	6	Multi	434	45.5	41.0
	8	6	Multi	547	**45.7**	**41.2**
Deformable	16	3	Single	**395**	44.2	39.9
	16	6	Single	413	**45.5**	**41.1**

Table 13. **Design Choices for Bottleneck Decoder.** FLOPs and performance (AP) are reported for COCO-val under different configurations: attention mechanisms (self, deformable, or no self-attention), bottleneck size (1/8, 1/16, 1/32), number of layers (3 or 6), scales (single or multi), and comparisons with/without multi-scale decoding.

	Cal.	Strategy	Rule	Lower	Upper	Min	Layers	COCO-val AP_{box}	COCO-val AP_{mask}
COCO	-	Confidence	Sigmoid	0.05	0.2	100	6	45.1	40.0
COCO	✓	Confidence	Sigmoid	0.05	0.2	100	6	46.0	41.1

Table 14. **Ablation Study of Query Pruning Strategy on COCO only.** Comparison of different pruning strategies across COCO with variations in calibration, selection strategy, rule type, threshold bounds, minimum kept elements, and decoder layers. We report FLOPs for the decoder and results on COCO-val.

want to retain more queries in the early layers and prune more aggressively in the later layers.

$$\text{thr}(l) = \text{l} + (\text{u} - \text{l}) \times \frac{e^{\alpha \times \frac{l}{L-1}} - 1}{e^{\alpha} - 1} \tag{6}$$

Strategy	Rule	FLOPs	COCO-val AP_{box}	COCO-val AP_{mask}	LVIS-minival AP_{box}	LVIS-minival AP_{mask}
Confidence	Sigmoid	4.6–7.6	52.2–52.7	46.2–46.7	47.6–47.9	44.0–44.5
Confidence	Logarithm	4.1–7.6	51.7–52.7	45.8–46.7	47.3–47.9	44.0–44.5
Confidence	Exponential	4.2–7.6	51.9–52.7	45.9–46.7	47.4–47.9	44.0–44.5

Table 15. **Comparison of Sigmoid, Logarithm, and Exponential strategies.** Results show decoder FLOPs, AP_{box}, and AP_{mask} on COCO-val and LVIS-minival. We report the range of results for different hyperparameter configurations.

Here, l is the current layer index, L is the total number of layers, and α is a parameter that controls the steepness of the curve. The threshold starts at l and approaches u as l increases.

Logarithmic Interpolation Logarithmic interpolation increases the confidence threshold logarithmically. This method allows for a rapid increase in the threshold in the early layers, which then slows down in the later layers. It is ideal for scenarios where you want to prune more aggressively in the initial layers.

$$\text{thr}(l) = \text{l} + (\text{u} - \text{l}) \times \frac{\log(1 + \alpha \times \frac{l}{L-1})}{\log(1 + \alpha)} \tag{7}$$

In this equation, α is a parameter that controls the curve's steepness. The threshold starts at l and grows rapidly at first, then gradually levels off as it approaches u.

Sigmoid Interpolation Sigmoid interpolation provides a smooth, S-shaped curve that starts slowly, increases more rapidly in the middle layers, and slows down again as it approaches the upper layers. This method is useful when a balanced, gradual transition is desired.

$$\text{thr}(l) = \text{l} + (\text{u} - \text{l}) \times \frac{1}{1 + e^{-\beta \times \left(\frac{l - \frac{L}{2}}{L/10}\right)}} \tag{8}$$

In this formula, β controls the steepness of the transition. The threshold starts at l, increases more rapidly around the middle layers, and finally levels off as it approaches u.

9. Qualitative Results

In table Fig. 5 we show results for the following supported tasks for a variety of input images: (1) category-guided instance segmentation using COCO categories, (2) category-agnostic instance segmentation, (3) referring detection and segmentation.

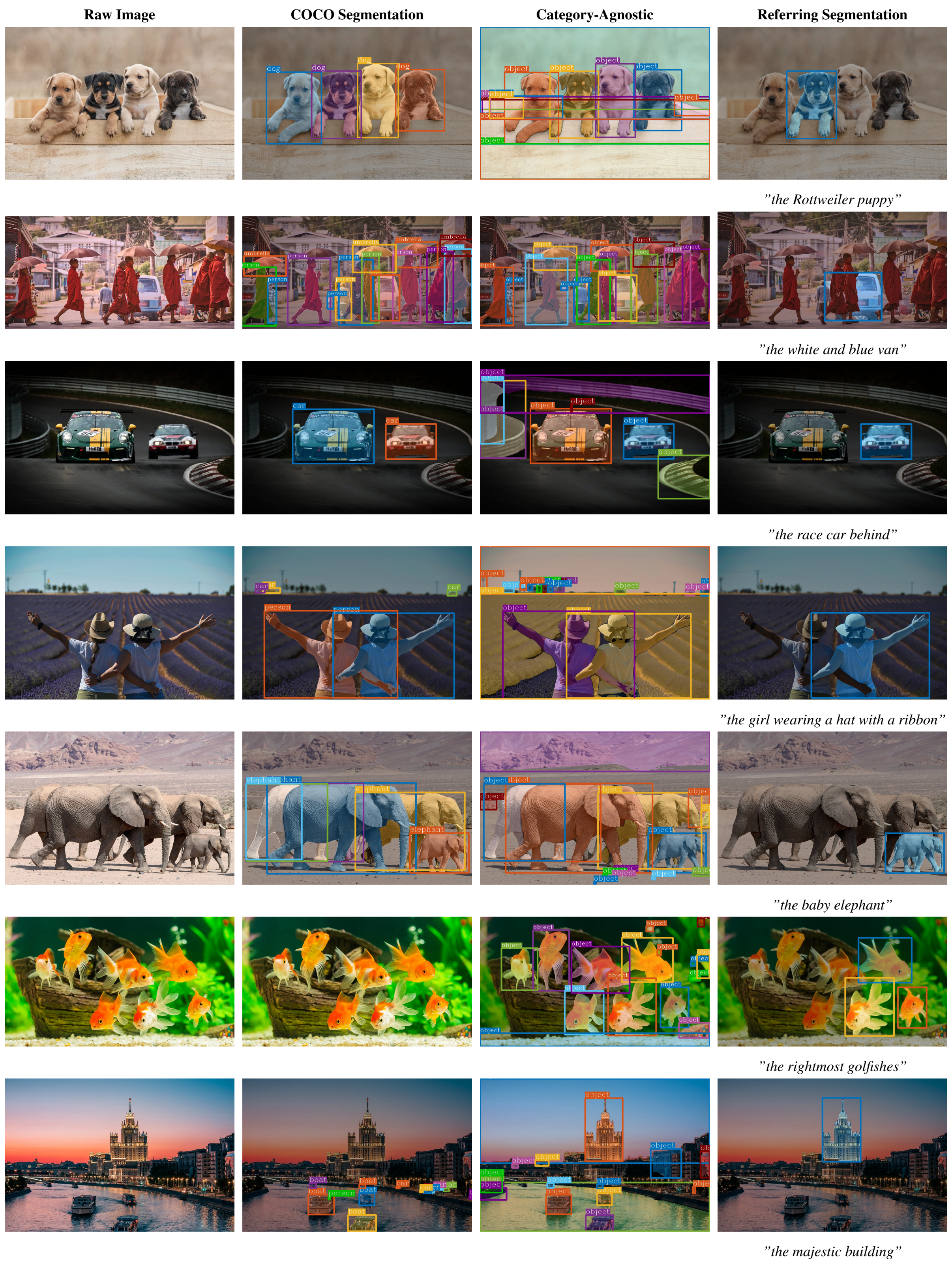

Figure 5. Qualitative results for different instance segmentation supported by our approach. In each row, we show the input image and report the instance segmentation results for (i) category-guided instance segmentation with COCO categories, (ii) category-agnostic instance segmentation, (iii) referring instance segmentation.