

UFO: A Unified Approach to Fine-grained Visual Perception via Open-ended Language Interface

Hao Tang^{1,2} Chenwei Xie² Haiyang Wang¹ Xiaoyi Bao^{2,3}
Tingyu Weng² Pandeng Li² Yun Zheng^{2†} Liwei Wang^{1†}

¹Center for Data Science, Peking University ²Alibaba Group

³Institute of Automation, Chinese Academy of Sciences

{tanghao@stu, wanghaiyang@stu, wanglw@cis}.pku.edu.cn baoxiaoyi2021@ia.ac.cn
{eniaticxcw, wengtingyu.wty, lipandeng.lpd, zhengyun.zy}@alibaba-inc.com

Abstract

Generalist models have achieved remarkable success in both language and vision-language tasks, showcasing the potential of unified modeling. However, effectively integrating fine-grained perception tasks like detection and segmentation into these models remains a significant challenge. This is primarily because these tasks often rely heavily on task-specific designs and architectures that can complicate the modeling process. To address this challenge, we present UFO, a framework that Unifies Fine-grained visual perception tasks through an Open-ended language interface. By transforming all perception targets into the language space, UFO unifies object-level detection, pixel-level segmentation, and image-level vision-language tasks into a single model. Additionally, we introduce a novel embedding retrieval approach that relies solely on the language interface to support segmentation tasks. Our framework bridges the gap between fine-grained perception and vision-language tasks, significantly simplifying architectural design and training strategies while achieving comparable or superior performance to methods with intricate task-specific designs. After multi-task training on five standard visual perception datasets, UFO outperforms the previous state-of-the-art generalist models by 12.3 mAP on COCO instance segmentation and 3.3 mIoU on ADE20K semantic segmentation. Furthermore, our method seamlessly integrates with existing MLLMs, effectively combining fine-grained perception capabilities with their advanced language abilities, thereby enabling more challenging tasks such as reasoning segmentation. Code and models are available at <https://github.com/nnnth/UFO>.

1. Introduction

Multimodal large language models (MLLMs) [1, 12, 35, 41, 45, 81] have made significant progress, exhibiting out-

standing performance on various visual tasks. Despite these achievements, their scopes are largely confined to image-level vision-language tasks, leaving fine-grained perception (e.g., detection and segmentation) as a critical weakness. Recent studies have shown that enabling MLLMs to collaborate with off-the-shelf detectors and segmenters can enhance precise visual understanding [18, 71] and facilitate advanced applications such as mobile agents [36, 62, 63, 73], indicating that endowing MLLMs with fine-grained perception capabilities is beneficial. However, seamlessly integrating these tasks into MLLMs poses challenges because traditional specialized methods heavily rely on complex and task-specific designs, such as RPN [53] and mask decoders [30].

A straightforward approach adopted by many existing works [33, 47, 51, 54, 68] is to augment MLLMs with task decoders. As shown in Figure 1 (a), LISA [33] introduces SAM [30] to support segmentation. Similarly, VisionLLM v2 [68] employs extra box and mask decoders. However, this combination introduces several limitations. First, task decoders add architectural complexity, necessitating compatibility among multiple components whenever the LLM is scaled up or new task decoders are introduced. Second, it complicates the end-to-end training. For example, the last stage in VisionLLM v2 [68] is dedicated to finetuning the task decoders because they fail to converge in earlier stages. These issues create a significant discrepancy with traditional vision-language modeling, limiting their broader application in general-purpose MLLMs. To remove task decoders, another line of research [6, 46, 66, 72] adopt text-based methods, converting boxes into location tokens or textual numbers. However, many text-based methods do not support segmentation due to the challenges of encoding masks of arbitrary shapes and complexity [3] with limited text tokens.

To address the above limitations, GiT [61] designs a language interface to support both box and mask outputs without task decoders. However, it still falls short of unifying

[†]Corresponding author.

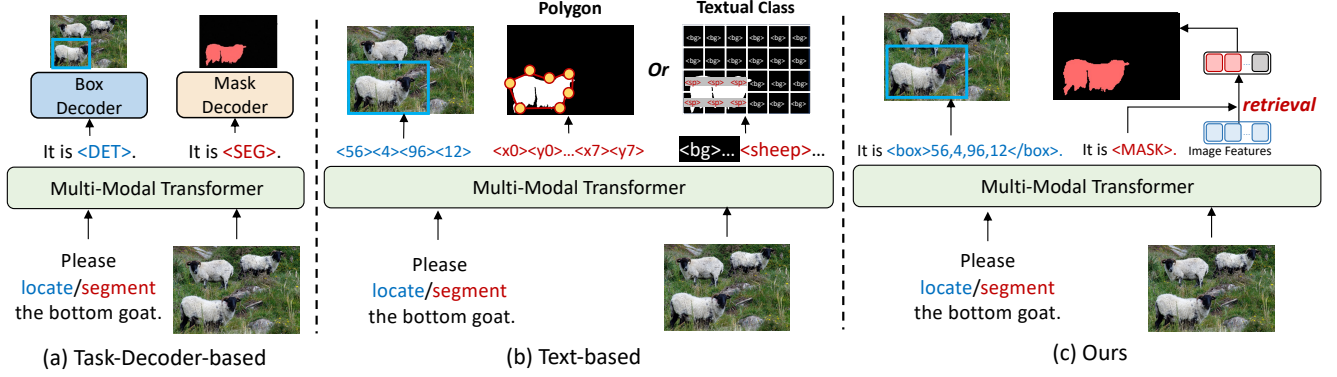


Figure 1. Methods to augment MLLMs with fine-grained perception tasks. (a) Relying on task decoders [33, 68], (b) Previous text-based methods represent boxes with location tokens [46, 61] and represent masks with suboptimal polygons [61, 66] or textual classes [34, 61], (c) Ours: predicting open-ended text sequences while using a simple yet effective embedding retrieval approach to support masks.

fine-grained perception tasks into MLLMs. For segmentation, GiT utilizes two suboptimal representations, illustrated in Figure 1 (b). In instance segmentation, it predicts polygon coordinates, which introduce quantization errors. In semantic segmentation, it predicts textual classes for each pixel, resulting in overly long sequences. For detection, GiT translates boxes into four location tokens and restricts the output space exclusively to these tokens, incompatible with other open-ended vision-language tasks that generate text of arbitrary length. In addition, GiT does not scale to MLLM and uses a Vision Transformer (ViT [21]) for multimodal tasks, resulting in poor language abilities. Hence, it is essential to develop a more effective approach to unify fine-grained perception into MLLMs. This method should effortlessly align with open-ended language interfaces, involve minimal structural complexity and deliver excellent performance.

In this paper, we present UFO, which unifies fine-grained perception tasks through the same open-ended language interface as vision-language tasks, without any task decoders. By translating all task outputs into open-ended text sequences, we demonstrate that competitive performance can be achieved without complex task-specific designs. As illustrated in Figure 1 (c), we reformulate segmentation as an embedding retrieval problem, where the mask token embedding computes similarity with image features by dot product, retrieving high-similarity positions to generate the mask. This design effectively leverages the output image features processed by MLLMs, which are often overlooked in previous methods. Our intuition is that since MLLMs achieve strong visual understanding, the mask information is already in the image features and we just need to retrieve it. Furthermore, we introduce a novel method that upsamples output masks by predicting multiple mask tokens, resulting in more refined masks and improved performance.

We first validate our method following GiT [61], which uses a lightweight ViT for multimodal processing but can share the same formulation as MLLMs (see Table 1), allowing efficient validation. GiT constructs a comprehensive

multi-task benchmark, which covers various granularity fine-grained perception tasks. Under the same evaluation protocols, UFO outperforms GiT by 12.3 mAP and 3.3 mIoU in COCO instance segmentation and ADE20K semantic segmentation, respectively. We then scale our method to larger MLLMs. Experimental results demonstrate that UFO effectively integrates fine-grained perception capabilities into MLLMs, achieving strong performance without task decoders. Leveraging the powerful language capabilities of MLLMs, UFO can extend to more language-dependent tasks, such as reasoning segmentation.

In summary, our contributions are listed as follows:

- (1) We introduce UFO, the first framework to unify diverse fine-grained perception tasks through the same open-ended language interface as vision-language tasks, without relying on task-specific decoders.
- (2) We reformulate segmentation as an embedding retrieval problem, exploring both text generation and image representation abilities of the language interface, significantly outperforming previous text-based methods.
- (3) Our framework seamlessly integrates with MLLMs, delivering competitive performance with a wide range of fine-grained perception tasks of varying difficulties.

2. Related Work

2.1. Multimodal Large Language Models

Inspired by the success of large language models (LLMs), multimodal large language models (MLLMs) have rapidly advanced in recent years. Early efforts like LLaVA [39], MiniGPT-4 [81], and Instruct-BLIP [17] fine-tune LLMs with instruction-following datasets, demonstrating strong multimodal understanding. More advanced MLLMs like DeepSeek-VL [41], Qwen2-VL [65], and InternVL2 [12] have emerged recently, offering superior multimodal comprehension through larger model sizes and extensive training data. However, these models remain limited to image-level vision-language tasks, with less exploration of fine-grained

Model	Image Tokenizer	Text Tokenizer	Multimodal Transformer
LLaVA [39]	CLIP [49],MLP	Llama Tokenizer [59]	Vicuna [14]
EVE [20]	Patch embedding	Llama Tokenizer [59]	Vicuna [14]
GiT [61]	Patch embedding	Bert Tokenizer [19]	ViT [21]
UFO-ViT	Patch Embedding	Bert Tokenizer [19]	ViT [21]
UFO-LLaVA-1.5-7B	CLIP [49],MLP	Llama Tokenizer [59]	Vicuna 1.5 [14]
UFO-InternVL2-8B	InternViT [12],MLP	InternLM2.5 Tokenizer [77]	InternLM2.5-7B [77]

Table 1. We abstract current multimodal architectures into three components: (1) Image tokenizer, converting images into visual tokens; (2) Text tokenizer, outputting text tokens; (3) Multimodal transformer, jointly processing visual and text tokens. We construct three variants by this formulation and list their architecture details.

visual perception, thereby restricting their application scope.

2.2. Extend MLLMs with Fine-grained Perception

Extend MLLMs with Task Decoders. Recent works [2, 33, 47, 51, 54, 68, 69, 78, 79] introduce task decoders to extend MLLMs with tasks like detection and segmentation. These models treat the MLLM as a coarse proposal extractor, passing the task-relevant embeddings to specialized decoders. The decoders then manage task-specific details, such as regressing boxes or generating masks. Although this approach yields strong performance, extra task decoders complicate architectures and training, undermining the unified design of MLLMs and limiting their potential.

Extend MLLMs with Text Outputs. To augment MLLMs with object-level perception tasks, previous methods [5, 6, 46, 48, 60, 66, 72] have employed location tokens or textual numbers to represent boxes. However, these approaches face challenges with pixel-level tasks such as segmentation, often requiring polygonal approximations [48, 66]. Although VistaLLM [48] reduces the errors of polygons through adaptive sampling and achieves competitive performance in referring segmentation, it remains inadequate for general segmentation tasks. First, polygons struggle to accurately represent “stuff” categories with amorphous regions (e.g., roads with parked cars), which are common in real world [3, 16, 74, 80]. Second, polygons inherently cause information loss, especially for detailed structures like retinal vessels [58]. In contrast, UFO leverages the multimodal outputs of MLLMs to generate precise masks for any shape, offering greater expressiveness and improved performance.

2.3. Vision Generalist Models

Vision generalist models aim to establish a unified framework supporting various vision-centric tasks. Inspired by the seq2seq framework in NLP, previous generalist models [8, 42, 61, 64] transform visual tasks into sequence generation problems. Notably, GiT [61] unifies five core visual tasks through a language interface, supporting box, mask, and text outputs. However, these models typically focus solely on visual tasks and lack the advanced language capabilities required for complex multimodal tasks.

3. Methods

As our method is applicable to various multimodal architectures, we first present a unified architectural abstraction in Section 3.1. Then, in Sections 3.2 and 3.3, we explain how to integrate box and mask representations into the open-ended language interface. Finally, in Section 3.4, we describe our multi-task data template for joint training.

3.1. Preliminary

Our goal is to unify fine-grained perception tasks into the open-ended language interface, thereby ensuring compatibility with any multimodal architecture that supports the same interface. We abstract existing multimodal architecture into three components based on the modalities they process: image tokenizer, text tokenizer and multimodal transformer, as shown in Tabel 1. For example, in LLaVA [39], the image tokenizer includes a vision encoder and MLP connector that extract visual features and map them into the LLM’s input space, while the multimodal transformer corresponds to the LLM. This abstraction applies not only to MLLMs with various image tokenizers [20, 35, 39] but also to vision generalist models with similar architectures [61], significantly broadening the scope of our method. To avoid confusion, we will refer to MLLMs by default in the following sections.

3.2. Bounding Box Representation

To align with the open-ended language interface while avoiding the addition of extra location tokens, we directly translate boxes into textual numbers. Each box is represented by the coordinates of its top-left (x_1, y_1) and bottom-right (x_2, y_2) corners. The continuous values of these coordinates are discretized into integers within $[0, \text{range}]$, enclosed by `<box>` and `</box>` tokens. If a class label is required, we simply prepend the textual class before the `<box>` token. For example, a box of a person can be represented as:

person, <box>465, 268, 589, 344</box>

This approach uses open-ended sequences to represent boxes, effectively aligning with vision-language tasks.

3.3. Mask Representation

Representing masks via the language interface is more challenging because masks contain more detailed information than boxes. Previous methods either use polygon formats, which sacrifice details, or assign textual classes to each pixel, resulting in overly long sequences. Therefore, a more efficient method to represent detailed masks is needed.

We observe that in MLLMs, the language interface is actually multimodal, where projected image features and text features are combined and jointly processed by the LLM. However, most existing methods ignore the output image features processed by the LLM. We argue that since MLLMs can express where and what objects are in text form, the

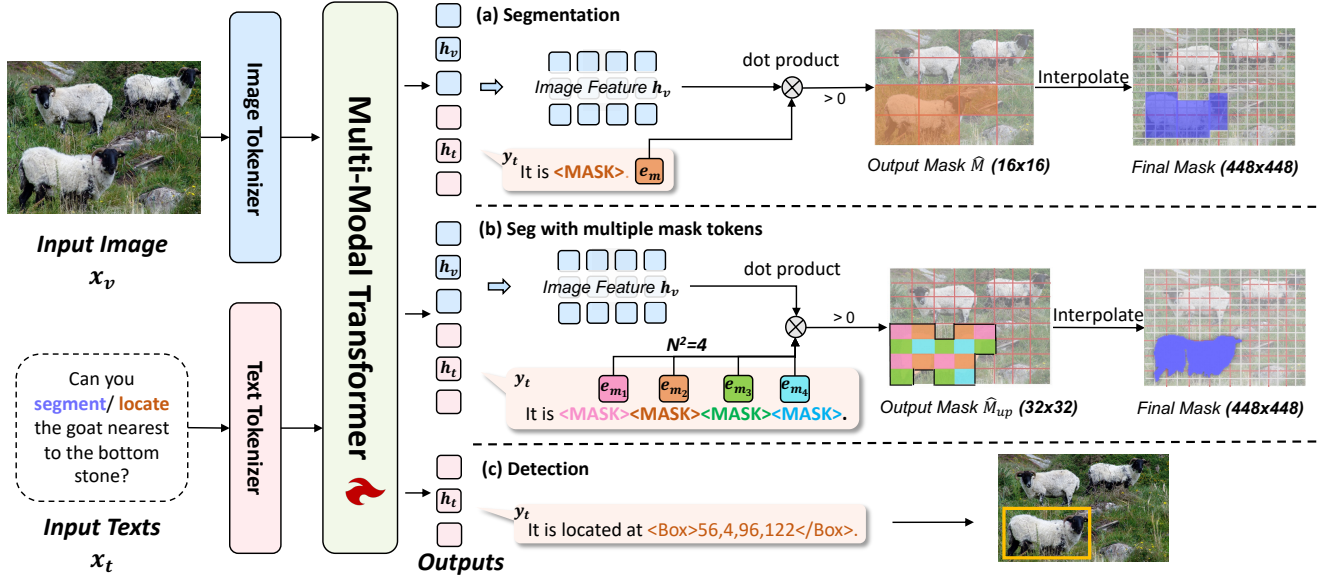


Figure 2. Overview of our approach. (a) Segmentation modeling: the mask token embedding retrieves similar image features to generate masks (shown with matching colors). (b) Upsampling masks by multiple mask tokens, retrieving more details by more tokens. We use $N=2$ to illustrate while using $N=4$ in implementation. (c) We output open-ended text sequences with textual numbers for detection.

mask information is already encoded in the image features. We just need to teach the model to decode this information. Therefore, we design a representation method based on image features and text embeddings. Instead of storing mask information in text embeddings, we use the text embeddings as query embeddings to extract mask information from the image features. The detailed approach is described below.

Segmentation by Embedding Retrieval. To incorporate the segmentation task using only the language interface, we reformulate it as an embedding retrieval problem. Specifically, we approach segmentation via mask prediction. We first augment the basic vocabulary of the model with a $\langle \text{MASK} \rangle$ token, which serves as the indicator for mask generation. When performing segmentation, the model is trained to output the $\langle \text{MASK} \rangle$ token, as shown in Figure 2 (a). Formally, given an input image \mathbf{x}_v and a segmentation prompt \mathbf{x}_t , the model \mathcal{F} generates the text response \mathbf{y}_t and corresponding output embeddings \mathbf{h}_t , image features \mathbf{h}_v as:

$$\mathbf{h}_v, \mathbf{y}_t, \mathbf{h}_t = \mathcal{F}(\mathbf{x}_v, \mathbf{x}_t). \quad (1)$$

We extract the mask token embedding \mathbf{e}_m corresponding to the $\langle \text{MASK} \rangle$ token from \mathbf{h}_t . To generate the segmentation mask, we compute the similarity between the mask token embedding \mathbf{e}_m and the image features \mathbf{h}_v via a scaled dot product. Positive scores are retrieved to form the binary mask $\hat{\mathbf{M}}$. This process is expressed as:

$$\mathbf{s} = \frac{\mathbf{e}_m \mathbf{h}_v^\top}{\sqrt{d}}, \quad \hat{\mathbf{M}} = \mathbb{I}(\mathbf{s} > 0), \quad (2)$$

where d is the embedding dimension, \mathbf{s} represents the similarity scores, and \mathbb{I} is the indicator function that converts the

similarity scores into a binary mask. By computing the dot product similarity between the mask token embedding and image features, we retrieve the most relevant image features corresponding to the mask token, thereby producing a mask aligned with the original image.

Our approach removes the need for task decoders while leveraging the inherent capabilities of the MLLMs to effectively support segmentation tasks. We hypothesize that, in well-encoded image features, features sharing the same semantics will group into clusters. Therefore, generating a mask token embedding equates to identifying the center of the relevant image feature cluster, while computing the similarity between embeddings reflects this relationship.

Upsampling by Multiple Mask Tokens. Due to the redundancy in visual information, it is common to process visual features at reduced resolutions. For example, the CLIP-L/14 [49] model downsamples image features by a factor of 14. In our segmentation method, similarities are computed using downsampled image features, resulting in low-resolution masks. However, directly upsampling by interpolation leads to coarse results and suboptimal performance due to the high interpolation factor.

To address this issue, we propose an upsampling method by predicting multiple mask tokens. For an image $\mathbf{x}_v \in \mathbb{R}^{H \times W \times 3}$, we obtain image features $\mathbf{h}_v \in \mathbb{R}^{H_p \times W_p \times d}$ downsampled by the patch size p , where d represents the feature dimension. Our target is to upsample the generated mask by N times, producing $\hat{\mathbf{M}}_{\text{up}} \in \mathbb{R}^{(H_p N) \times (W_p N)}$ from image features $\mathbf{h}_v \in \mathbb{R}^{H_p \times W_p \times d}$. This requires decoding an $N \times N$ mask for each position in the image features. To achieve this, we train the model to autoregressively pre-

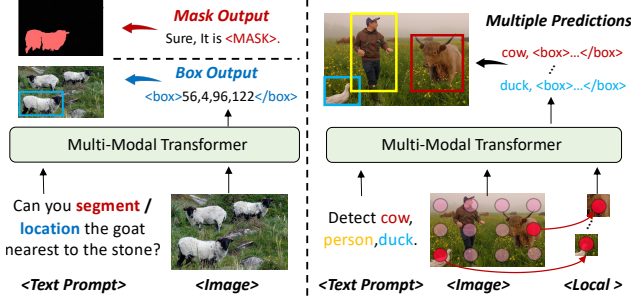


Figure 3. Multi-task data template examples. Red dots represent sampled grid point features, acting as local visual prompts for generating text sequences for nearby objects.

dict N^2 $\langle \text{MASK} \rangle$ tokens with embeddings $\{\mathbf{e}_{\text{mi}}\}_{i=1}^{N^2}$. Each token corresponds to a single position in the $N \times N$ upsampling grid, as illustrated in Figure 2 (b). For each mask token embedding \mathbf{e}_{mi} , we compute the similarity with the visual features \mathbf{h}_v :

$$\mathbf{s}_i = \frac{\mathbf{e}_{\text{mi}} \mathbf{h}_v^\top}{\sqrt{d}}, \quad (3)$$

where $\mathbf{e}_{\text{mi}} \in \mathbb{R}^{1 \times d}$, $\mathbf{h}_v^\top \in \mathbb{R}^{d \times H_p \times W_p}$, and $\mathbf{s}_i \in \mathbb{R}^{1 \times H_p \times W_p}$. These similarity scores $\{\mathbf{s}_i\}_{i=1}^{N^2}$ are then concatenated and reshaped into an upsampled similarity map:

$$\mathbf{s}_{\text{concat}} = \text{concat}(\{\mathbf{s}_i\}_{i=1}^{N^2}), \quad \mathbf{s}_{\text{concat}} \in \mathbb{R}^{N^2 \times H_p \times W_p}, \quad (4)$$

$$\mathbf{s}_{\text{up}} = \text{reshape}(\mathbf{s}_{\text{concat}}), \quad \mathbf{s}_{\text{up}} \in \mathbb{R}^{(H_p N) \times (W_p N)}. \quad (5)$$

Finally, we retrieve positive scores in \mathbf{s}_{up} to generate the upsampled binary mask $\hat{\mathbf{M}}_{\text{up}}$. By default, we set $N = 4$, predicting 16 $\langle \text{MASK} \rangle$ tokens, which upsamples the output mask by a factor of 4. The mask is then aligned with the original image resolution through interpolation.

Our method effectively leverages mask token embeddings as upsampling parameters, offering greater flexibility than traditional methods like bilinear interpolation and transposed convolution. Bilinear interpolation uses non-learnable parameters, while transposed convolution allows for learnable parameters, the same parameters are applied to all images after training. In contrast, we use embeddings generated by the network as the parameters, which can be customized for each image. This approach enables the model to generate optimal upsampling parameters dynamically, enhancing flexibility while achieving better performance.

3.4. Multi-Task Data Template

Based on the above designs, we construct multi-task data templates for joint training. We classify tasks into two categories based on output prediction number: single-prediction tasks like visual grounding produce one box or mask, and multi-prediction tasks like object detection generate several boxes. Merging multiple outputs into a long sequence is inefficient and the order among them is hard to define,

making autoregressive learning of the sequence difficult [7]. Inspired by GiT [61], we adopt a parallel decoding approach that splits multi-prediction tasks into independent subtasks, each handling one prediction in parallel. This strategy effectively accelerates inference and enhances task scalability.

Single prediction. For tasks only require a single prediction, our task template is:

$$\langle \text{Text Prompt} \rangle \langle \text{Image} \rangle \langle \text{Text Response} \rangle$$

As shown in Figure 3, we follow previous methods to construct text prompts and use our unified box and mask representation for text responses.

Multiple predictions. To efficiently support multi-prediction tasks, we split complex tasks into independent subtasks with single prediction, enabling parallel decoding within a batch. The key to achieving parallelism is to ensure all subtasks are independent. Typically, multiple boxes and masks correspond to different locations. Therefore, we introduce local image features in the input to differentiate these sub-tasks, serving as visual prompts. The template is structured as follows:

$$\langle \text{Text Prompt} \rangle \langle \text{Image} \rangle \langle \text{Local} \rangle \langle \text{Text Response} \rangle$$

where $\langle \text{Local} \rangle$ refers to local image features interpolated by grids sampled on the image. Specifically, we uniformly sample M grid points on the whole image and interpolate the local image features at each grid location. We then match the M grid points with K predictions, assigning each prediction to the nearest grid point. As shown in Figure 3, example sub-sequences might be:

Detect cow, ... $\langle \text{Image} \rangle \langle \text{Local} \rangle$ cow, $\langle \text{box} \rangle$...
Segment cow, ... $\langle \text{Image} \rangle \langle \text{Local} \rangle$ cow, $\langle \text{MASK} \rangle$...

The remaining $M - K$ grid points predict the end token. In this way, M grid points correspond to M independent subtasks. They share the text prompt and image features but have distinct local features and text responses, forming M separate sub-sequences that can be predicted in parallel within a batch. By breaking down complex tasks into simple single-prediction subtasks, this approach enhances efficiency while simplifying learning. We adopt this template for object detection, instance segmentation and semantic segmentation.

4. Training

To ensure efficient validation and fair comparison, we first follow GiT [61], using a smaller ViT as the multimodal transformer for multi-task training across five standard visual perception tasks. We then scale to MLLMs, validating on the same multi-task benchmark. Finally, we enrich the data by incorporating more diverse datasets, enabling fine-grained instruction tuning of MLLMs. After instruction tuning, the fine-grained perception capabilities are seamlessly integrated with the robust language abilities of MLLMs, thereby applying to perception tasks that require advanced language capabilities, such as reasoning segmentation.

Table 2. Results on GiT [61]’s multi-task benchmark. “single-task” refers to models trained on each task separately, while “multi-task” indicates models trained jointly across all selected tasks. “*” denotes the model is capable of the task, though no number is reported. “-” means incapability in that specific task. We highlight the top-1 entries of one-stage multi-task generalist models and joint training improvements with **bold** font. We follow [61] to list specific modules, which are designed for specific functions in architecture.

Methods	Specific Modules		#Params	Object Detection			Instance Seg			Semantic Seg	Captioning		REC
	Examples	Num		AP	AP ₅₀	AP ₇₅	AP	AP ₅₀	AP ₇₅	mIoU(SS)	BLEU-4	CIDEr	Acc@0.5
Specialist Models													
Deformable-DETR [83]	RegressionHead	5	40M	45.4	64.7	49.0	-	-	-	-	-	-	-
Pix2Seq [7]	Text Decoder	3	37M	43.0	61.0	45.6	-	-	-	-	-	-	-
Mask R-CNN [24]	FPN,RPN	6	46M	41.0	61.7	44.9	37.1	58.4	40.1	-	-	-	-
Polar Mask [70]	CenterNetHead	5	55M	-	-	-	30.5	52.0	31.1	-	-	-	-
Mask2Former [13]	PixelDecoder	5	44M	-	-	-	43.7	-	-	47.2	-	-	-
VL-T5 [15]	Faster R-CNN	3	440M	-	-	-	-	-	-	-	34.5	116.5	-
MDETR [27]	RoBERTa,DETR	6	188M	-	-	-	-	-	-	-	-	-	86.8
Generalist Models (MultiTask-Training)													
Uni-Perceiver [84]	None	1	124M	-	-	-	-	-	-	-	32.0	★	★
Uni-Perceiver-MoE [82]	None	1	167M	-	-	-	-	-	-	-	33.2	★	★
VisionLLM-R50 [66]	Deform-DETR	6	7B	44.6	64.0	48.1	25.1	50.0	22.4	-	31.0	112.5	80.6
GiT-B _{single-task} [61]	None	1	131M	45.1	62.7	49.1	31.4	54.8	31.2	47.7	33.7	107.9	83.3
GiT-B _{multi-task} [61]	None	1	131M	46.7	64.2	50.7	31.9	56.4	31.4	47.8	35.4	112.6	85.8
GiT-L _{multi-task} [61]	None	1	387M	51.3	69.2	55.9	35.1	61.4	34.7	50.6	35.7	116.0	88.4
GiT-H _{multi-task} [61]	None	1	756M	52.9	71.0	57.8	35.8	62.6	35.6	52.4	36.2	118.2	89.2
UFO-ViT-B _{single-task}	None	1	131M	47.8	65.7	52.0	42.6	65.8	46.1	49.5	34.2	111.1	83.6
UFO-ViT-B _{multi-task}	None	1	131M	48.3	66.6	52.6	43.5	66.2	47.0	50.2	35.3	114.2	85.8
<i>Improvement</i> (single→multi)				+0.5	+0.9	+0.6	+0.9	+0.4	+0.9	+0.7	+1.1	+3.1	+2.2
UFO-ViT-L _{multi-task}	None	1	387M	52.9	71.3	57.9	47.3	70.9	51.6	54.0	35.9	118.6	88.5
UFO-ViT-H _{multi-task}	None	1	756M	54.1	72.4	58.9	48.1	71.6	53.0	55.7	37.6	123.6	89.2
UFO-InternVL2-8B _{multi-task}	None	1	8B	52.3	71.7	56.5	45.8	69.5	49.7	54.6	39.6	131.6	90.4

Task	Example Sources	Count	Size
Visual Question Answering	LLaVA-v1.5- mix665k [39]	1	0.7m
Refer Expression Comprehension	RefCOCO [75], RefCLEF [28]	4	0.8m
Refer Expression Segmentation	RefCOCO [75], RefCLEF [28]	4	0.8m
Object Detection	Objects365 [56], COCO [37]	5	3.8m
Instance Segmentation	OpenImages [32], LVIS [23]	6	1.4m
Semantic Segmentation	COCOStuff [3], ADE20K [80]	4	0.3m

Table 3. Datasets used in instruction tuning. The columns, from left to right, represent task name, dataset examples, dataset count, and data amount. More details are provided in the appendix.

4.1. Multi-Task Training

Architecture. To ensure fair comparison and validate the effectiveness of our approach across various architectures, we conduct multi-task training using two variants: UFO-ViT and UFO-InternVL2-8B. UFO-ViT strictly follows GiT [61], employing a SAM [30]-pretrained ViT [21] and a text tokenizer from BERT [19]. It is available in three sizes: ViT-B, ViT-L, and ViT-H. UFO-InternVL2-8B utilizes the pretraining weight of InternVL2-8B [12], with detailed model specifications provided in Table 1.

Datasets. We use the same multi-task dataset as GiT, which includes five standard visual perception tasks: COCO 2017 [37] for object detection and instance segmentation, COCO Caption [10] for image captioning, the RefCOCO series [43, 75] for referring expression comprehension (REC), and ADE20K [80] for semantic segmentation.

4.2. Fine-grained Instruction Tuning

Architecture. To demonstrate that our method is applicable to various MLLMs, we use not only InternVL2-8B [12] but also the LLaVA-1.5-7B [40] for pretraining, specifically UFO-LLaVA-1.5-7B. Architecture details are in Table 1.

Datasets. To enhance the model’s versatility, we enrich our training datasets with more datasets and tasks, extensively covering major visual perception tasks. Our training data includes 24 datasets across 6 tasks, as detailed in Table 3. More details of data composition are in the appendix.

5. Experiments

5.1. Experimental Settings

Multi-Task Training Details. To facilitate comparison and verify the effectiveness of our method, we also conduct single-task training independently on five selected tasks. For both single-task and multi-task training, we use a batch size of 24 and employ the AdamW [29] optimizer with a cosine annealing schedule, setting the initial learning rate to 0.0002. More details are in the appendix.

Fine-grained Instruction Tuning Details. In training, we use a batch size of 32 with gradient accumulation set to 16, running on 8 NVIDIA A100 GPUs for 120K iterations. The AdamW [29] optimizer and a cosine annealing schedule are employed, with a learning rate of 0.0002 and weight decay of 0.01. For efficient training, we employ LoRA [25] with a rank of 8, freezing the image tokenizer while keeping only

Methods	Referring Expression Comprehension (REC)									Referring Expression Segmentation (RES)								
	RefCOCO			RefCOCO+			RefCOCOg		Avg	RefCOCO			RefCOCO+			RefCOCOg		Avg
	val	testA	testB	val	testA	testB	val	test		val	testA	testB	val	testA	testB	val	test	
<i>MLLMs with Task Decoders</i>																		
LISA [33]	-	-	-	-	-	-	-	-	-	74.1	76.5	71.1	62.4	67.4	56.5	66.4	68.5	67.9
PixelLM [54]	-	-	-	-	-	-	-	-	-	73.0	76.5	68.2	66.3	71.7	58.3	69.3	70.5	69.2
SAM4MLLM-7B [11]	-	-	-	-	-	-	-	-	-	77.1	80.9	72.5	71.5	76.8	64.7	74.5	75.2	74.2
NExT-Chat [76]	85.5	90.0	77.9	77.2	84.5	68.0	80.1	79.8	80.4	74.7	78.9	69.5	65.1	71.9	56.7	67.0	67.0	68.9
PerceptionGPT [47]	88.6	92.5	84.6	82.1	88.6	74.2	84.1	85.2	85.0	75.1	78.6	71.7	68.5	73.9	61.3	70.3	71.7	71.4
VisionLLM v2 [68]	90.0	93.1	87.1	81.1	87.3	74.5	85.0	86.4	85.6	79.2	82.3	77.0	68.9	75.8	61.8	73.3	74.8	74.1
<i>MLLMs w/o Task Decoders</i>																		
Shirka-7B [6]	87.0	90.6	80.2	81.6	87.4	72.1	82.3	82.2	82.9	-	-	-	-	-	-	-	-	-
MiniGPT-v2-7B [5]	88.1	91.3	84.3	79.6	85.5	73.3	84.2	84.3	83.8	-	-	-	-	-	-	-	-	-
Ferret-7B [72]	87.5	91.4	82.5	80.8	87.4	73.1	83.9	84.8	83.9	-	-	-	-	-	-	-	-	-
VistaLLM [48]	88.1	91.5	83.0	82.9	89.8	74.8	83.6	84.4	84.8	74.5	76.0	72.7	69.1	73.7	64.0	69.0	70.9	71.2
UFO-LLaVA-1.5-7B	89.9	93.0	86.0	84.3	90.3	77.5	86.0	86.8	86.7	76.2	78.5	73.0	69.8	75.1	63.9	71.1	71.9	72.4
UFO-LLaVA-1.5-7B*	90.8	93.0	87.3	85.5	90.5	78.6	86.9	87.2	87.5	77.2	79.4	73.8	70.8	75.5	64.1	72.1	73.2	73.3
UFO-InternVL2-8B	90.7	94.0	87.4	85.4	90.1	79.2	86.7	87.0	87.6	77.3	79.2	74.7	71.3	75.7	66.5	72.8	73.2	73.8
UFO-InternVL2-8B*	91.4	93.8	88.2	85.7	90.7	79.7	86.8	87.4	88.0	78.0	79.7	75.6	72.3	76.8	66.6	73.7	74.3	74.6

Table 4. Comparison of referring expression comprehension (REC) and segmentation (RES) performance. Results on REC are reported based on P@0.5. Results for RES are reported based on cumulative IoU (cIoU). * denotes the model is specifically finetuned on the dataset.

the LLM trainable. More details are in the appendix.

Training Objectives. All tasks utilize a CrossEntropy Loss as they are unified under the open-ended language interface. For segmentation tasks, we additionally apply focal loss [55] and dice loss to supervise the binary mask output. The final loss for segmentation tasks is expressed as:

$$\mathcal{L}_{\text{seg}} = \lambda_{\text{CE}}\mathcal{L}_{\text{CE}} + \lambda_{\text{focal}}\mathcal{L}_{\text{focal}} + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}}$$

We empirically find that setting all weights to 1 offers better overall performance. See appendix for more details.

Methods	ReasonSeg		
	overall	short query	long query
GRES [38]	21.3	17.6	22.6
X-Decoder [85]	21.7	20.4	22.2
SEEM [86]	24.3	20.1	25.6
LISA-LLaVA-7B [33]	36.8	37.6	36.6
LISA-LLaVA-7B [33]*	47.3	40.6	49.4
LISA-LLaVA-1.5-7B [33]	48.7	47.1	49.2
LISA-LLaVA-1.5-7B [33]*	55.6	48.3	57.9
UFO-LLaVA-1.5-7B	53.8	40.1	58.2
UFO-LLaVA-1.5-7B*	58.0	46.3	61.7
UFO-InternVL2-8B	55.4	41.9	59.8
UFO-InternVL2-8B*	61.2	49.6	64.9

Table 5. Results (gIoU) on ReasonSeg test set. * denotes using 239 reasoning segmentation data samples to fine-tune the model.

5.2. Multi-Task Evaluation

We evaluate our model’s performance in both single-task and multi-task settings across five standard vision-centric tasks, benchmarking it against specialized and generalist models. Without extra task decoders, our model adapts to various tasks through the proposed open-ended language interface and achieves outstanding performance.

Comparison with Specialist Models. As shown in Table 2, our single-task model effectively bridges the performance gap with specialized models, achieving either superior or on-par performance. For example, we achieve 47.8 mAP in detection compared to 45.4 mAP with Deformable-DETR [83] and 49.5 mIoU in semantic segmentation against 47.2 mIoU with Mask2Former [13]. In instance segmentation, we also outperform specialized methods like Mask R-CNN [24] and Polar Mask [70] while matching the performance of Mask2Former [13].

Comparison with Generalist Models. To facilitate comparison with GiT [61], we adopt its one-stage training method without task-specific finetuning. This involves jointly training on a mixed dataset of the five tasks and directly testing on their respective validation or test sets. Table 2 shows that our model outperforms the previous leading generalist model, GiT, across all tasks, with the same pretraining and data. Notably, in the largest model size, we outperform GiT by 12.3 mAP on COCO instance segmentation and 3.3 mIoU on ADE20K semantic segmentation, demonstrating the superiority of our segmentation modeling. The improvement in captioning is mainly because we use a standard vocabulary for all tasks, while GiT uses task-specific vocabularies, hindering the learning of general text representation.

We also observe a multi-task synergy effect like GiT, with performance on instance segmentation improved by 0.9 mAP and captioning increased by 3.1 CIDEr. Our multi-task improvements on segmentation also outperform GiT (0.7 vs. 0.1 mIoU). We attribute this to unified modeling across segmentation tasks, whereas GiT employs separate methods for instance and semantic segmentation.

After scaling to MLLMs, we observe improved performance on captioning and REC, while performance on other tasks remains comparable to the UFO-ViT-L. We speculate

that this performance difference primarily arises from different pretraining. For UFO-ViT, we use SAM [30] pretraining, making it more aligned with detection and segmentation. In contrast, InternVL2-8B is mainly pre-trained on image-level vision-language tasks, which better suit captioning and REC.

5.3. Fine-grained Instruction Tuning Results

After training MLLMs with data shown in Table 3, we first evaluate their object and pixel-level perception performance on the visual grounding tasks. Then we evaluate on a more difficult task, reasoning segmentation, examining whether our method effectively combines fine-grained perception with the MLLMs’ language reasoning abilities.

Visual Grounding maps textual descriptions to objects in an image, which can be further categorized into referring expression comprehension (REC) and segmentation (RES). We comprehensively list the results for the two tasks in Table 4. We report results in two settings: direct evaluation after joint training and specifically finetuning. Without using box decoders, our best model surpasses the previous state-of-the-art VisionLLM v2 [68] by an average of 2.0%. After specific finetuning, our model achieves stronger performance with 88.0% on average. While all previous approaches rely on mask decoders or polygon approximations for segmentation, our method delivers superior or comparable performance without them. For instance, our InternVL2 variant outperforms the SAM [30]-based LISA [33] by an average of 5.9 cIoU and matches with VisionLLM v2 [68]. These outcomes validate the effectiveness of our method, demonstrating that with proper task modeling, MLLMs can handle fine-grained perception tasks without task decoders.

Reasoning Segmentation (ReasonSeg) is a challenging benchmark introduced by LISA [33], which presents more sophisticated and nuanced instructions, requiring models to leverage world knowledge and engage in deeper logical reasoning. We adhere to the evaluation protocol of LISA, reporting both zero-shot and finetuned results. As shown in Table 5, with the same pretraining, our LLaVA-1.5 variant outperforms LISA by 5.1 gIoU in zero-shot settings. With advanced MLLM InternVL2, our performance further improves, exceeding LISA by 6.7 gIoU. Finetuning also significantly improves performance, with our InternVL2 variant improving by 5.8 gIoU compared to zero-shot.

Since reasoning segmentation requires both language reasoning and precise segmentation, we attribute our improvement to better task integration through unified modeling. In LISA, the MLLM handles only language reasoning and generates coarse segmentation prompts, relying on an additional SAM for finer segmentation. This leads to information loss and insufficient synergy. In our unified modeling, the MLLM manages both language reasoning and precise segmentation, allowing different task capabilities to fully integrate within a shared parameter space, thereby enhancing synergy.

Decoding Rule	Beam Search	Positive Predictions	Detection mAP
✓	✓	9.1 100 67.0	43.0 45.1 45.1

Table 6. Ablation of open-ended decoding on UFO-ViT-B_{single-task}.

N^2	1	4	16	25
mAP	38.9	41.3	42.6	42.9
FPS	7.0	5.9	3.6	2.8

Table 7. Mask token number ablation on UFO-ViT-B_{single-task} for instance segmentation.

Open Ended	Beam Search	Embedding Retrieval	Copy Paste	Repeat GT	Detection AP	Ins Seg AP
✓					45.1	31.4
✓	✓				43.0	30.4
✓	✓	✓			45.1	31.3
✓	✓	✓	✓		45.1	39.2
✓	✓	✓	✓	✓	46.2	40.4
✓	✓	✓	✓	✓	47.8	42.6

Table 8. Ablation on COCO detection and instance segmentation.

5.4. Ablation Study

Open-ended decoding. We explore the impact of our open-ended decoding on single-task detection. As noted in Section 3.4, we split object detection into sub-tasks for each grid point (e.g., 625 grid points for a 1120×1120 image), which leads to an imbalance between positive and negative samples due to more grid points than objects. Our method utilizes a standard vocabulary (e.g., BERT’s 30,524 tokens) for open-ended decoding. This output space is much bigger than the range of positive classes (e.g., 80 in COCO), worsening class imbalance and reducing positive predictions in inference. As shown in Table 6, while using decoding rules like removing negative classes from the vocabulary [61] could force all outputs to be positive, this compromises our generality. Therefore, we use beam search [52], which allows the model to explore multiple potential sequences. This approach effectively increases positive predictions and improves performance. By default, we only apply beam search for COCO detection, instance segmentation, and image captioning.

Number of Mask tokens. We ablate the number of mask tokens on the COCO instance segmentation task. As shown in Table 7, using multiple tokens significantly improves performance compared to a single token, but the gains plateau after 16. Considering the increased training and inference costs, we set $N^2 = 16$ by default for a balanced trade-off.

Advanced training strategies. The sparsity of positive samples in multi-prediction tasks hampers effective learning. To mitigate this, we use two advanced strategies to increase the ratio of positive samples. First, copy-paste data augmentation [22], where objects from other images are pasted onto the target image. Second, we repeat ground truth k times [26], defaulting to 3. As seen in Table 8, copy-paste boosts mAP by 1.1 for detection and 1.2 for instance segmentation, while repeating ground truth further boosts mAP by 1.6 and 2.2. This demonstrates that the sparsity of is a key bottleneck, and our performance can be effectively improved with these strategies. By default, we only use these strategies for COCO detection and instance segmentation.

6. Conclusion

In this paper, we present UFO, a unified approach for various fine-grained visual perception tasks with an open-ended language interface. We translate all perception targets into open-ended text sequences and introduce a novel embedding retrieval method for segmentation. Experiments show that our method can achieve excellent performance on MLLMs without requiring architecture modifications. Our unification fully aligns with vision-language tasks, providing a flexible, effective, and scalable solution to enhance the fine-grained perception capabilities of MLLMs, paving the way to build more general multimodal models.

References

- [1] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv:2308.12966*, 2023. 1
- [2] Xiaoyi Bao, Siyang Sun, Shuailei Ma, Kecheng Zheng, Yuxin Guo, Guosheng Zhao, Yun Zheng, and Xingang Wang. Cores: Orchestrating the dance of reasoning and segmentation. In *ECCV*, 2024. 3
- [3] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *CVPR*, 2018. 1, 3, 6, 2
- [4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 2
- [5] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yuniang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv:2310.09478*, 2023. 3, 7
- [6] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal llm’s referential dialogue magic. *arXiv:2306.15195*, 2023. 1, 3, 7
- [7] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. In *ICLR*, 2022. 5, 6
- [8] Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J Fleet, and Geoffrey E Hinton. A unified sequence interface for vision tasks. In *NeurIPS*, 2022. 3, 4
- [9] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014. 2
- [10] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv:1504.00325*, 2015. 6
- [11] Yi-Chia Chen, Wei-Hua Li, Cheng Sun, Yu-Chiang Frank Wang, and Chu-Song Chen. Sam4mllm: Enhance multimodal large language model for referring expression segmentation. In *ECCV*, 2024. 7
- [12] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024. 1, 2, 3, 6
- [13] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 6, 7
- [14] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. <https://vicuna.lmsys.org>, 2023. 3
- [15] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. Unifying vision-and-language tasks via text generation. In *ICML*, 2021. 6
- [16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 3
- [17] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. In *NeurIPS*, 2023. 2
- [18] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv:2409.17146*, 2024. 1
- [19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018. 3, 6
- [20] Haiwen Diao, Yufeng Cui, Xiaotong Li, Yueze Wang, Huchuan Lu, and Xinlong Wang. Unveiling encoder-free vision-language models. *arXiv:2406.11832*, 2024. 3
- [21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2, 3, 6
- [22] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, 2021. 8
- [23] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 6, 2
- [24] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 6, 7
- [25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022. 6

- [26] Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. Detrs with hybrid matching. In *CVPR*, 2023. 8
- [27] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *ICCV*, 2021. 6
- [28] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 6, 2
- [29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 6
- [30] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 1, 6, 8
- [31] Harold W Kuhn. The hungarian method for the assignment problem. In *NRL*, 1955. 1
- [32] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Mallocci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. In *IJCV*, 2020. 6, 2
- [33] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. In *CVPR*, 2024. 1, 2, 3, 7, 8
- [34] Mengcheng Lan, Chaofeng Chen, Yue Zhou, Jiaying Xu, Yiping Ke, Xinjiang Wang, Litong Feng, and Wayne Zhang. Text4seg: Reimagining image segmentation as text generation. In *ICLR*, 2025. 2
- [35] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. 1, 3
- [36] Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui 2: Mastering universal user interface understanding across platforms. *arXiv:2410.18967*, 2024. 1
- [37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6, 2
- [38] Chang Liu, Henghui Ding, and Xudong Jiang. Gres: Generalized referring expression segmentation. In *CVPR*, 2023. 7
- [39] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 2, 3, 6
- [40] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *CVPR*, 2024. 6
- [41] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Yaofeng Sun, et al. Deepseek-vl: towards real-world vision-language understanding. *arXiv:2403.05525*, 2024. 1, 2
- [42] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Motlaghi, and Aniruddha Kembhavi. UNIFIED-IO: A unified model for vision, language, and multi-modal tasks. In *ICLR*, 2023. 3, 4
- [43] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016. 6
- [44] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Buló, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 2
- [45] OpenAI. Gpt-4 technical report, 2023. 1
- [46] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shao-han Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv:2306.14824*, 2023. 1, 2, 3
- [47] Renjie Pi, Lewei Yao, Jiahui Gao, Jipeng Zhang, and Tong Zhang. Perceptiongpt: Effectively fusing visual perception into llm. In *CVPR*, 2024. 1, 3, 7
- [48] Shravan Pramanick, Guangxing Han, Rui Hou, Sayan Nag, Ser-Nam Lim, Nicolas Ballas, Qifan Wang, Rama Chellappa, and Amjad Almahairi. Jack of all tasks master of many: Designing general-purpose coarse-to-fine vision-language model. In *CVPR*, 2024. 3, 7
- [49] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 3, 4
- [50] Vignesh Ramanathan, Anmol Kalia, Vladan Petrovic, Yi Wen, Baixue Zheng, Baishan Guo, Rui Wang, Aaron Marquez, Rama Kovvuri, Abhishek Kadian, et al. Paco: Parts and attributes of common objects. In *CVPR*, 2023. 2
- [51] Hanoona Rasheed, Muhammad Maaz, Sahal Shaji, Abdelrahman Shaker, Salman Khan, Hisham Cholakkal, Rao M Anwer, Eric Xing, Ming-Hsuan Yang, and Fahad S Khan. Glamm: Pixel grounding large multimodal model. In *CVPR*, 2024. 1, 3
- [52] Raj Reddy. Speech understanding systems: A summary of results of the five-year research effort at carnegie mellon university. *Technical Report*, 1977. 8
- [53] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1
- [54] Zhongwei Ren, Zhicheng Huang, Yunchao Wei, Yao Zhao, Dongmei Fu, Jiashi Feng, and Xiaoje Jin. Pixellm: Pixel reasoning with large multimodal model. In *CVPR*, 2024. 1, 3, 7
- [55] T-YLPG Ross and GKHP Dollár. Focal loss for dense object detection. In *CVPR*, 2017. 7
- [56] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *ICCV*, 2019. 6, 2
- [57] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgb-d images. In *ECCV*, 2012. 4
- [58] Joes Staal, Michael D Abràmoff, Meindert Niemeijer, Max A Viergever, and Bram Van Ginneken. Ridge-based vessel segmentation in color images of the retina. *TMI*, 2004. 3

- [59] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv:2302.13971*, 2023. 3
- [60] Haiyang Wang, Yue Fan, Muhammad Ferjad Naeem, Yongqin Xian, Jan Eric Lenssen, Liwei Wang, Federico Tomba, and Bernt Schiele. Tokenformer: Rethinking transformer scaling with tokenized model parameters. *arXiv:2410.23168*, 2024. 3
- [61] Haiyang Wang, Hao Tang, Li Jiang, Shaoshuai Shi, Muhammad Ferjad Naeem, Hongsheng Li, Bernt Schiele, and Liwei Wang. Git: Towards generalist vision transformer through universal language interface. In *ECCV*, 2024. 1, 2, 3, 5, 6, 7, 8, 4
- [62] Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *arXiv:2406.01014*, 2024. 1
- [63] Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv:2401.16158*, 2024. 1
- [64] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *ICML*, 2022. 3
- [65] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv:2409.12191*, 2024. 2
- [66] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *arXiv:2305.11175*, 2023. 1, 2, 3, 6
- [67] Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist painter for in-context visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6830–6839, 2023. 4
- [68] Jiannan Wu, Muyan Zhong, Sen Xing, Zeqiang Lai, Zhaoyang Liu, Wenhai Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, et al. Visionllm v2: An end-to-end generalist multimodal large language model for hundreds of vision-language tasks. In *NeurIPS*, 2024. 1, 2, 3, 7, 8
- [69] Zhuofan Xia, Dongchen Han, Yizeng Han, Xuran Pan, Shiji Song, and Gao Huang. Gsva: Generalized segmentation via multimodal large language models. In *CVPR*, 2024. 3
- [70] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *CVPR*, 2020. 6, 7
- [71] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv:2310.11441*, 2023. 1
- [72] Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. Ferret: Refer and ground anything anywhere at any granularity. *arXiv:2310.07704*, 2023. 1, 3, 7
- [73] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *arXiv:2404.05719*, 2024. 1
- [74] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020. 3
- [75] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *ECCV*, 2016. 6, 2
- [76] Ao Zhang, Liming Zhao, Chen-Wei Xie, Yun Zheng, Wei Ji, and Tat-Seng Chua. Next-chat: An lmm for chat, detection and segmentation. *arXiv:2311.04498*, 2023. 7
- [77] Pan Zhang, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Rui Qian, Lin Chen, Qipeng Guo, Haodong Duan, Bin Wang, Linke Ouyang, et al. Internlm-xcomposer-2.5: A versatile large vision language model supporting long-contextual input and output. *arXiv:2407.03320*, 2024. 3
- [78] Tao Zhang, Xiangtai Li, Hao Fei, Haobo Yuan, Shengqiong Wu, Shunping Ji, Chen Change Loy, and Shuicheng Yan. Omg-llava: Bridging image-level, object-level, pixel-level reasoning and understanding. *arXiv:2406.19389*, 2024. 3
- [79] Yichi Zhang, Ziqiao Ma, Xiaofeng Gao, Suhaila Shakiah, Qiaozi Gao, and Joyce Chai. Groundhog: Grounding large language models to holistic segmentation. In *CVPR*, 2024. 3
- [80] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 3, 6, 2
- [81] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv:2304.10592*, 2023. 1, 2
- [82] Jinguo Zhu, Xizhou Zhu, Wenhai Wang, Xiaohua Wang, Hongsheng Li, Xiaogang Wang, and Jifeng Dai. Uni-perceiver-moe: Learning sparse generalist models with conditional moes. In *NeurIPS*, 2022. 6
- [83] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2020. 6, 7
- [84] Xizhou Zhu, Jinguo Zhu, Hao Li, Xiaoshi Wu, Hongsheng Li, Xiaohua Wang, and Jifeng Dai. Uni-perceiver: Pre-training unified architecture for generic perception for zero-shot and few-shot tasks. In *CVPR*, 2022. 6
- [85] Xueyan Zou, Zi-Yi Dou, Jianwei Yang, Zhe Gan, Linjie Li, Chunyuan Li, Xiyang Dai, Harkirat Behl, Jianfeng Wang, Lu Yuan, et al. Generalized decoding for pixel, image, and language. In *CVPR*, 2023. 7
- [86] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Wang, Lijuan Wang, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. In *NeurIPS*, 2024. 7

UFO: A Unified Approach to Fine-grained Visual Perception via Open-ended Language Interface

Supplementary Material

Our supplementary material provides detailed information, including implementation details (§7), dataset descriptions (§8), and training specifics (§9). Inference speed and additional ablation studies are covered in §10 and §11, respectively. Experiments on more tasks and discussions with previous work are included in §12 and §13. Qualitative results from two training setups and visualizations of multiple mask tokens are presented in §14.

7. Implementation Details

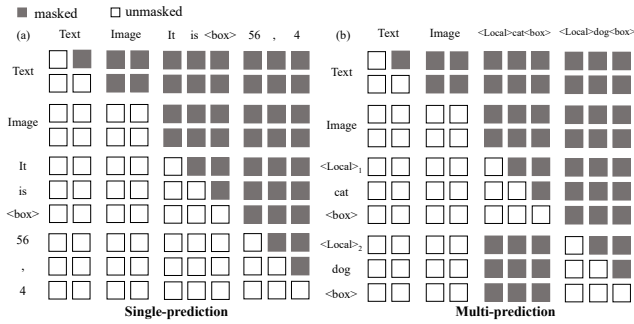


Figure 4. Attention mask visualizations. (a) We apply bidirectional attention for image features. (b) For multi-prediction tasks, we mask each subsequence from seeing others.

ViT Architecture. Our ViT architecture follows the same design as GiT [61], using the SAM variant of ViT. We also follow GiT to add 6 newly initialized transformer layers upon the original ViT, leading to better performance. For example, UFO-ViT-B consists of 18 layers.

Grid Generation. As mentioned in the multi-task template (Section 3.4), we divide multi-prediction tasks into sub-tasks, each corresponding to the nearest grid point. The number of grid points is roughly proportional to the image resolution, as shown in Table 11. Calculating the loss for all sub-tasks is computationally expensive. Therefore, we sample grid points and compute the loss on a subset, as presented in Table 11. When sampling, we prioritize positive samples. If the number of positive samples is less than the target size, we then sample from negative samples. During multi-task training on InternVL2-8B, we adjust the resolution based on the default resolution of the image tokenizer (using multiples of 448) and modify the grid point configurations accordingly. In fine-grained instruction tuning, we use the default resolution of the image tokenizer, such as 448^2 for InternVL2 and 336^2 for LLaVA-1.5. For the multi-prediction tasks of the two MLLMs, we use 100 grids and sample 40 of them.

Attention Mask. To accurately model multi-modal features and manage relationships among sub-tasks in multi-prediction tasks, we customize the attention mask based on autoregressive attention. As shown in Figure 4 (a), for tasks with a single prediction, we use autoregressive attention but apply bidirectional attention to the image features to better capture inter-image relationships. For tasks with multiple predictions, during training, we concatenate multiple sub-task sequences into a long sequence but use attention masks to prevent different sub-tasks from seeing each other, as illustrated in Figure 4 (b). This approach enables different sub-tasks to be decoded in parallel during inference. Assuming there are M sub-tasks, we first process the `<Text Prompt>` and `<Image>` and save their key-value (KV) caches. These KV caches are then duplicated M times to create caches for M subsequences. By batching these subsequences together, the model can decode them in parallel, thereby accelerating the inference speed.

Label Assignment. In multi-prediction tasks, we use the Hungarian algorithm [31] to match sub-tasks with grid points, specifically to associate boxes and masks with grid points. For boxes, the matching is based on the distance between the center of each box and the grid points. For masks, we first convert them into box format before matching.

Coordinate Discretization. We follow GiT to convert continuous coordinates into discrete numbers within the range $[0, \text{range}]$. Specifically, we define the range as twice the image resolution. For example, for a 448×448 image, we convert coordinate values into integers within $[0, 896]$.

Data Augmentation. For object detection, instance and semantic segmentation, we use `RandomFlip` and `RandomResizedCrop`. We also use `CopyPaste` for object detection and instance segmentation. For other tasks, we simply resize the images to the required resolution.

Indicator function. In segmentation, we use sigmoid as the indicator and then get a mask with a 0.5 threshold.

8. Dataset Details

Fine-grained Instruction Tuning Datasets As shown in Table 9, our Fine-grained Instruction Tuning datasets build upon GiT [61]’s universal datasets with the following modifications: (1) Replacing captioning datasets with VQA datasets to maintain the multimodal conversation capabilities of MLLMs. (2) Excluding datasets with smaller data volumes, such as Pascal VOC and BDD100K. (3) Following LISA [33], adding PACO-LVIS and Pascal Part to enhance segmentation of object parts.

Task	Sources	Size
VQA	<i>LLaVA-v1.5- mix665k</i> [39]	0.7m
REC & RES	<i>RefCOCO</i> [75], <i>RefCOCO+</i> [75], <i>RefCOCog</i> [75], <i>RefCLEF</i> [28]	0.8m
Object Detection	<i>Objects365</i> [56], <i>OpenImages</i> [32], <i>COCO</i> [37], <i>LVIS</i> [23], <i>nuImages</i> [4]	3.8m
Instance Seg	<i>OpenImages</i> [32], <i>LVIS</i> [23], <i>COCO</i> [37], <i>PACO-LVIS</i> [50], <i>PASCAL-Part</i> [9], <i>nuImages</i> [4],	1.4m
Semantic Seg	<i>COCOStuff</i> [3], <i>Mapillary</i> [44] <i>nuImages</i> [4], <i>ADE20K</i> [80]	0.3m

Table 9. Fine-grained instruction tuning datasets.

9. Training Details

Multi-Task Training Details. Multi-task training setting is in Table 10. For single-task training, we simply reduce the training iterations to 120k. For multi-task training on InternVL2-8B, we keep all parameters trainable and reduce the iterations to 400k because of its faster convergence.

Progressive High-resolution Training of MLLMs. In multi-task training, object detection, instance segmentation, and semantic segmentation require predicting a large number of targets, including many small objects, making performance highly sensitive to resolution. For the relatively small UFO-ViT, we train directly using high resolution. However, for multi-task training on MLLMs, high resolution significantly increases training costs. Therefore, we adopt a progressive high-resolution training strategy: first training at a resolution of 448^2 for 300k iterations, then at 896^2 for 60k iterations, and finally at 1344^2 for 40k iterations. We utilize InternVL2-8B’s dynamic resolution to support high-resolution inputs. As shown in Table 12, increasing the resolution leads to substantial improvements in detection and segmentation performance, even with fewer iterations.

Fine-grained Instruction Tuning Details. Training setting is in Table 10. The training data for the MLLM includes six tasks, each containing multiple datasets. We first sample evenly from the six tasks, with a sampling ratio of 1/6 for each task. Then, within each task, we sample evenly from the corresponding datasets. For example, for object detection, the sampling ratio for each dataset is 1/30. When fine-tuning on a specific dataset, we maintain the same training setup but train for only 20k iterations.

10. Inference Speed

Table 11 shows the speed of UFO-ViT-B. By using parallel decoding for multi-prediction tasks, we achieve inference speeds comparable to single-prediction tasks, despite higher resolutions (1120^2 vs. 224^2) and more predictions. Table 13 shows the speed on MLLMs. Our LLaVA-1.5 variant is slower than InternVL2 on REC because its tokenizer converts textual numbers into longer token sequences. In embedding retrieval, the extra scaled-dot product operation only costs a negligible 0.17 ms for InternVL2-8B on an A100.

config	ViT	MLLM	MLLM-instruct
optimizer	AdamW	AdamW	AdamW
learning rate	2e-4	2e-4	2e-4
weight decay	0.05	0.01	0.01
layer-wise lr decay	0.85	0.85	-
schedule	cosine	cosine	cosine
gradient norm clip	0.1	1.0	1.0
warmup iters	1k	1k	1k
training iters	640k	400k	120k
batch size	24	24	32
gradient accumulation	-	-	16
LoRA rank	-	-	8
LoRA alpha	-	-	16
LoRA dropout	-	-	0.05
LoRA modules	-	-	LLMs
drop path	0.1(B), 0.4(L,H)	-	-
precision	FP16	BF16	BF16
GPUS	$24 \times \text{V100}$	$8 \times \text{A100}$	$8 \times \text{A100}$

Table 10. Multi-task training and instruction tuning settings.

Task	Resolution	Grid	Sample Grid	Speed
Object Detection	1120^2	625	250	4.1
Instance Seg	1120^2	625	250	3.6
Semantic Seg	672^2	225	90	4.8
Image Captioning	224^2	0	0	7.7
REC	224^2	0	0	9.1

Table 11. Resolution, grid number and sample grid number for the five tasks in multi-task training on UFO-ViT. Speed is measured on UFO-ViT-B, single A100 with batch size 1.

Resolution	Iters	Detection	Ins Seg	Sem Seg
448^2	300k	44.3	37.4	53.9
896^2	60k	51.7	44.1	54.6
1344^2	40k	52.3	45.8	-

Table 12. Iterations and performance in progressive high-resolution training on UFO-InternVL2-8B.

Model	REC	RES	ReasonSeg
UFO-InternVL2-8B	1.10	0.58	0.57
UFO-LLaVA-1.5-7B	0.78	0.67	0.65

Table 13. Inference speed on MLLMs. Speed is measured on an A100 GPU with batch size 1.

Beam Number	Detection mAP	Instance Seg mAP	Captioning BLEU-4	CIDEr
1	45.6	40.9	33.0	108.5
2	47.4	42.1	34.2	111.1
3	47.8	42.6	34.0	110.8
5	47.9	42.6	33.9	111.0

Table 14. Ablation of beam number on UFO-ViT-B_{single-task}.

11. More ablation studies

Beam Search. In Table 14, we present ablation studies on the beam number. As the beam number increases, performance initially improves and then stabilizes, but further

CE:Focal	1:1	1:3	3:1	Method	P@0.5	FPS
Ins Seg	43.5	43.7	43.4	box	90.7	1.10
Caption	35.3	34.9	35.3	mask2box	89.5	0.57

Table 15. Loss weight ablation.

Table 16. REC performance.

REC	RES	VQA	Ins Seg	Sem Seg	Det	RefCOCO val REC	RefCOCO val RES	ReasonSeg test
✓						88.2	-	-
✓	✓					88.6	73.5	47.7
✓	✓	✓				89.2	74.4	50.2
✓	✓	✓	✓	✓		89.6	76.7	54.9
✓	✓	✓	✓	✓	✓	90.7	77.3	55.4

Table 17. Ablation of MLLM training data on UFO-InternVL2-8B.

Model	Det	Ins Seg	Sem Seg	REC	RES
InternVL2-8B [9]	10.5	-	-	87.1	-
UFO-InternVL2-8B	52.3	45.8	54.6	90.7	77.3

Table 18. Comparisons with baseline MLLMs.

increases cause a slight performance drop in the captioning. Since larger beam numbers increase inference time, we select the optimal beam number: 3 for object detection and instance segmentation and 2 for captioning.

Loss weights. In Table 15, we ablate loss weights on UFO-ViT-B. We jointly training both instance segmentation and image captioning tasks. Although increasing the focal loss weight is slightly better for segmentation, it leads to a drop in captioning. Since our goal is better overall multi-task performance, we set all weights to 1 to avoid task competition.

Data contributions. In Table 17, we illustrate the impact of different task data on performance. We observe that adding VQA data significantly enhances performance on ReasonSeg, likely because it helps maintain the language reasoning abilities of MLLMs. The inclusion of instance and semantic segmentation data benefits both RES and ReasonSeg by providing dense mask annotations. Additionally, incorporating detection data leads to noticeable improvements in the REC task, as it offers dense bounding box annotations and potentially exposes the model to a broader range of concepts.

Comparisons with baseline MLLMs. In Table 18, we provide comparisons between UFO and the baseline MLLM. Firstly, UFO significantly expands the task range, enabling the model to handle all types of segmentation. Secondly, although InternVL2-8B can support REC and perform detection by breaking it into multiple single-category prediction tasks, its performance is markedly inferior to us, especially in detection. This is primarily because InternVL2-8B cannot predict multiple boxes for a single category nor model the relationships among multiple categories, leading to insufficient and contradictory predictions. In contrast, UFO effectively supports multi-prediction tasks through local prompts, allowing it to accommodate any prediction number.

Mask2Box. A simple way to output box based on segmentation is `mask2box`, which uses bounding rectangle of masks. Table 16 shows the performance of `mask2box`, which is slightly lower than directly predicting boxes. This is mainly

because some masks have outlier predictions, distorting the converted boxes. Moreover, boxes are generally shorter than masks, resulting in a faster speed (see Table 13). Notably, our box and mask representations are unified through the language interface. The only difference is that for boxes, textual numbers are converted into coordinates, and for masks, embedding retrieval is used. These operations are as simple as `mask2box`, which greatly reduces task-specific details compared to methods that use task decoders.

12. Extended Experiments

Retinal Vessel Segmentation. Our embedding retrieval method offers superior expressive capability compared to polygons, particularly in highly complex and detailed masks, which require a large number of vertices when using polygons. To further illustrate this, we fine-tune our model on the retinal vessel segmentation, where the vessels possess very irregular and narrow shapes, which are hard to represent as polygons. We follow the few-shot settings in GiT [61], fine-tuning both UFO-ViT-H and UFO-InternVL2-8B on the DRIVE [58] training set for only 100 steps. As shown in Figure 5, UFO accurately segments the retinal vessels. In performance, UFO-ViT-H achieves 77.4 Dice, outperforming GiT-H with 57.9 Dice. UFO-InternVL2-8B also achieves a competitive 76.3 Dice. This result validates the effectiveness of our segmentation method on extremely fine-grained structures, enabling support for more general segmentation.

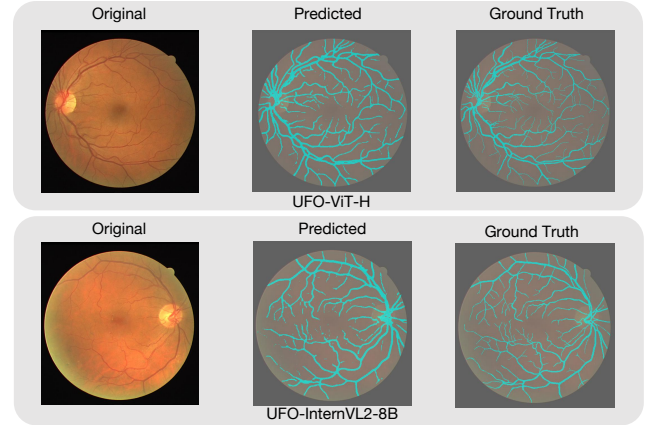


Figure 5. Visualizations of retinal vessel segmentation.

Depth Estimation. Thanks to the flexibility of our method, we can easily extend it to depth estimation similar to segmentation. We can apply a sigmoid to the dot product result to interpret it as relative depth r , which can be then mapped into absolute depth. For depth \hat{D} within $[D_{min}, D_{max}]$, we can predict it as follows:

$$r = \sigma\left(\frac{\mathbf{e}_d \mathbf{h}_v^\top}{\sqrt{d}}\right), \quad \hat{D} = r \cdot D_{max} + (1 - r) \cdot D_{min} \quad (6)$$

\mathbf{e}_d is the embedding of `<DEPTH>` token. As shown in Table 19, we can achieve competitive performance.

The above approach essentially shares the same modeling as segmentation, differing only on how to interpret the model output. In segmentation, dot product results serve as confidence scores that are thresholded to create masks, whereas in depth estimation, they represent relative depth and are then converted to absolute depth. This process can be seen as a simple post-processing, which is very common in general-purpose models [8, 67]. For example, Painter [67] converts RGB values to categories and depth using task-specific rules. Our unification lies in modeling all tasks through the standard language interface. When specific outputs (e.g., masks, depth) are needed, the corresponding post-processing is then performed. This design unifies the core understanding capabilities across tasks while requiring only minimal, learning-free post-processing for various formats.

Methods	RMSE↓	$\delta 1 \uparrow$	REL↓	log10↓
Painter [67]	0.327	0.930	0.090	-
Unified-IO 2 [42]	0.423	-	-	-
UFO-InternVL2-8B	0.320	0.928	0.091	0.039

Table 19. Depth estimation performance on NYUv2 Depth [57].

13. Discussions

Comparisons with GiT. GiT [61] also aims to build a generalist model for fine-grained perception tasks. Compared with GiT, we provides six key improvements: 1) Segmentation by embedding retrieval, a simple yet intuitive way to support segmentation by language interface. GiT uses polygons and textual classes, leading to information loss or lengthy sequences. In contrast, UFO can accurately segment using only 16 mask tokens. 2) Alignment with the open-ended language interface: unlike GiT, which requires separate vocabularies and fixed output lengths per task, UFO uses shared vocabulary and outputs arbitrary-length sequences. Tables 6 and 18 demonstrate that open-ended detection is challenging due to severe class imbalance. We address this issue with a text-aligned beam search and achieve enhanced performance. 3) Scalability to MLLMs: while GiT only experiments on relatively small ViTs, UFO can easily scale to larger MLLMs thanks to the aligned language interface. 4) Exploring the image representation capabilities of the language interface. GiT is a purely text-based method, while UFO can effectively extract mask information from image features. 5) Better task universality: GiT uses different methods for instance and semantic segmentation (polygon and textual class), while we adopt a unified approach for both tasks because UFO can support masks with any shape. As UFO is aligned with vision-language tasks, we can effortlessly combine VQA reasoning and segmentation, enabling ReasonSeg. 6) Significant performance improvements. In Table 2, UFO-ViT-H outperforms GiT-H by 1.2 mAP and 12.3 mAP on COCO detection and instance segmentation, and 3.3 mIoU on ADE20K semantic segmentation.

14. Visualization

Multi-task Training Results. In Figure 7, we visualize the multi-task training results of UFO-ViT-H. The model can not only handle simple perception tasks but also accurately detect and segment multiple objects in complex scenarios.

Instruction Tuning Results. We present qualitative results of UFO-InternVL2-8B in Figure 8. Leveraging the language capabilities of MLLMs, the model can accurately locate and segment based on both simple phrases and complex queries.

Multiple Mask Tokens. In Figure 9, we visualize the masks corresponding to each of multiple mask tokens. Each token captures specific details, such as different legs of a horse or the tail of a dog. Therefore, combining all the mask tokens results in a higher-resolution, more detailed mask. In Figure 10, we visualize the results for different numbers of mask tokens. Using only one mask token results in rough edges, while increasing the number of mask tokens produces more refined masks, leading to better performance.

Failure cases. We visualize the failure cases of UFO-InternVL2-8B on REC, RES and ReasonSeg in Figure 6. In the first example, the model misunderstands “fourth” and detects the third guy from the right, showing its inaccuracy in counting. In the second example, the model only marks visible table parts (blocked by food), while the human guesses hidden areas. This is mainly because the semantics of image features on occluded areas are different. However, whether to include occluded areas is also ambiguous. In the third example, the model fails to identify the answer as “river” due to limited knowledge, leading to segmentation errors.



Figure 6. Failure case visualizations of UFO-InternVL2-8B on REC, RES and ReasonSeg respectively.



Figure 7. Qualitative results of multi-task training. The first three rows correspond to object detection, instance segmentation, and semantic segmentation, while the last row shows results on captioning and referring expression comprehension.

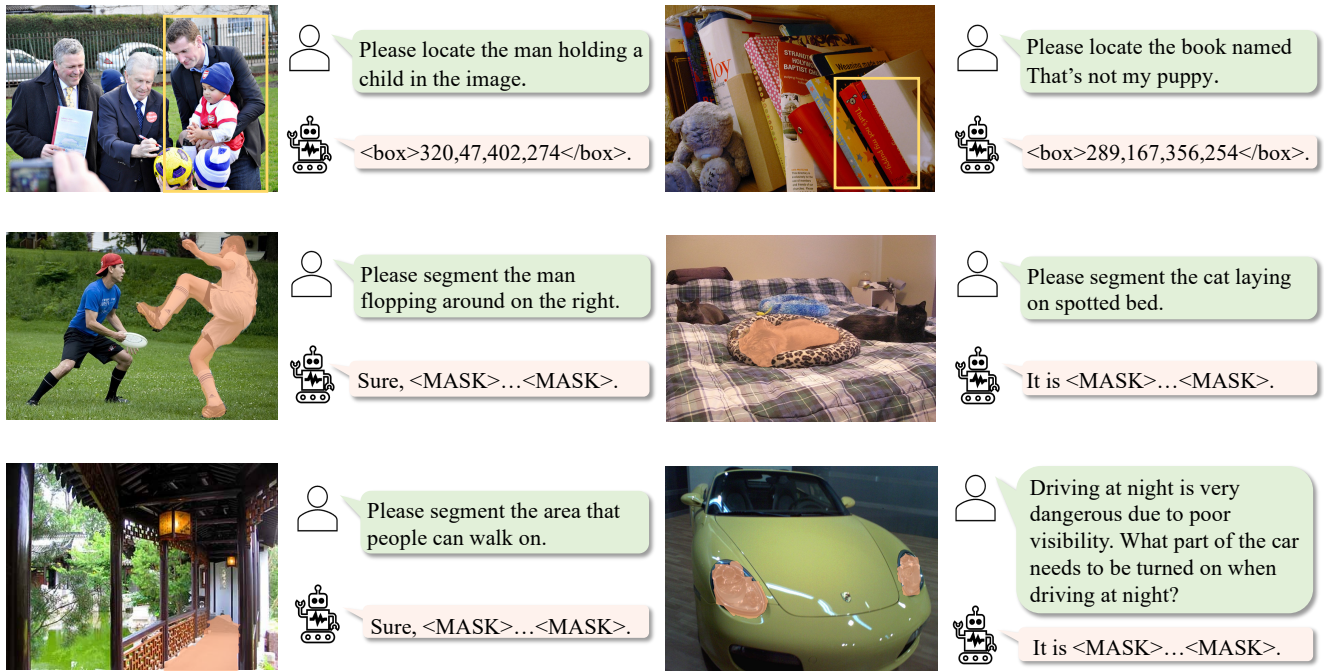


Figure 8. Qualitative results of Fine-grained Instruction Tuning. The three rows correspond to REC, RES, and reasoning segmentation in order.



Figure 9. Visualization of multiple mask tokens. We illustrate with four mask tokens (with $N=2$). Employing multiple mask tokens allows for capturing finer details, such as the horse leg and the dog tail, resulting in more precise and refined masks.

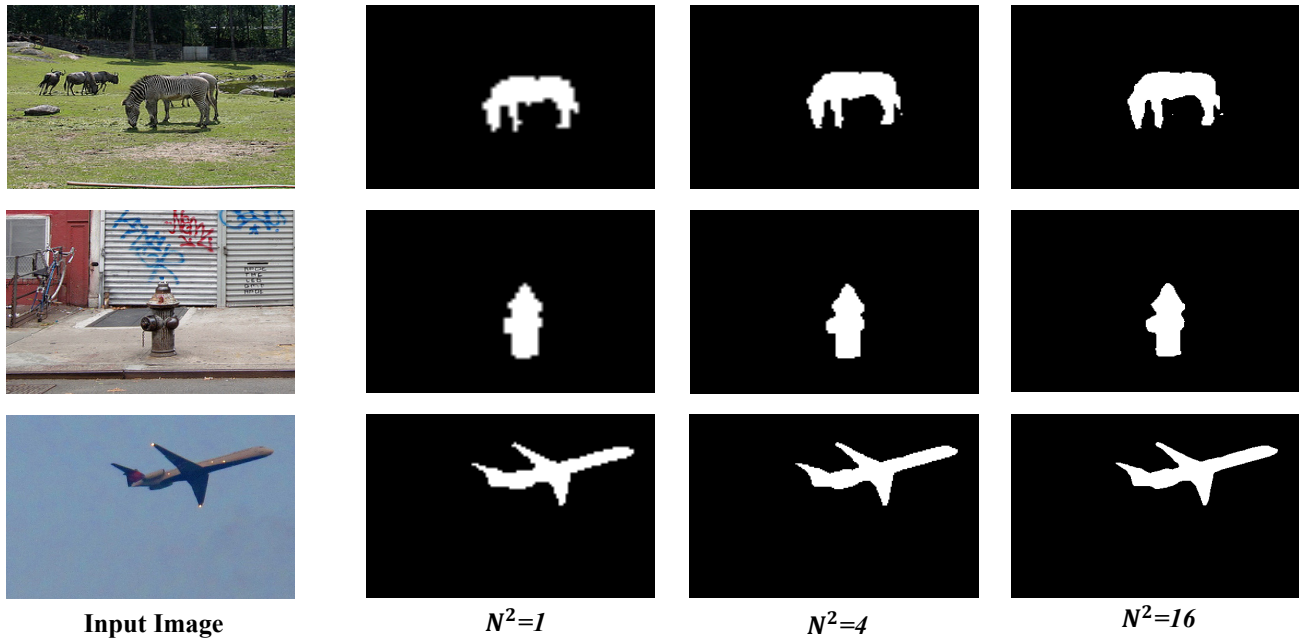


Figure 10. Instance segmentation results with different mask token number (N^2) on UFO-ViT-B_{single-task}.