GOD: Generative Object Detection with Co-Training

Yang Liu[1,2], Yao Zhang[3], Feng Hou[3],
Yang Zhang[3], Yangzhou Du[3], Zhongchao Shi[3], Jianping Fan[3], Zhiqiang He[1,2,4]
Institute of Computing Technology (ICT), Chinese Academy of Sciences[1],
University of Chinese Academy of Sciences[2], Lenovo Research[3], Lenovo Ltd.[4]

Abstract

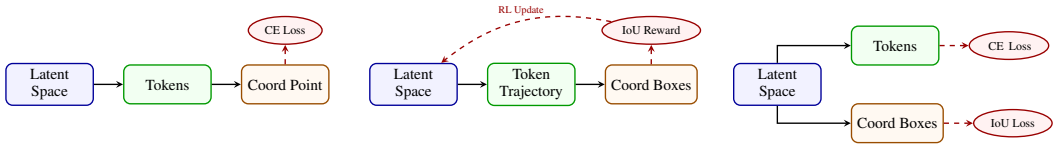
Recent progress in multimodal large language models (MLLMs) has strengthened visual understanding and instruction following, and opened a path to open-ended generative object detection, in which a model discovers, names, and localizes objects from free-form instructions without relying on a predefined category set. Nevertheless, current generative detectors remain less accurate than discriminative detectors, as token-level supervision provides only indirect and weak constraints on fine-grained visual perception. To equip MLLM with such an explicit optimization objective, we propose **GOD** (**G**enerative **O**bject **D**etection), a 3B-scale MLLM that injects object-level visual-prior supervision into generative detection by coupling visual-prior reconstruction with language generation. GOD attaches a lightweight DETR-style object-query branch to the MLLM visual encoder, maps the resulting object queries into the token space, and optimizes them jointly with autoregressive generation, MLLM-side object-query reconstruction, and DETR-side localization objectives. Specifically, in the training recipe, a multi-stage training pipeline first aligns the object decoder with the MLLM and then uses an interleaved-mask curriculum to transfer detector-grounded priors from proposal-assisted training to reference-free generation. Experiments on standard object detection and referring expression comprehension benchmarks show that GOD improves direct coordinate generation while preserving discriminative localization and language-conditioned grounding. These gains suggest that object-level visual priors are most effective when learned within the MLLM, rather than supplied only as external proposals, providing a practical bridge between generative language modeling and localization-aware perception.

1 Introduction

Object detection aims to recognize and localize object instances in images, yet conventional detectors are tied to the closed category set used during training, *e.g.*, the 80 classes in COCO [24], and therefore struggle with the diverse and evolving concepts in open-world scenes. Vision-language pre-training [14, 39] has advanced open-vocabulary object detection (OVD) by aligning region representations with language prompts and formulating detection as grounding or region-text matching [22, 31, 32, 46, 53]. However, existing discriminative OVD models still require predefined vocabularies or user-provided category prompts at inference, limiting their applicability when target objects are unknown or difficult to enumerate.

With the rapid progress of Multimodal Large Language Models (MLLMs) in visual understanding and cross-modal reasoning, object detection is gradually evolving from a prompt-conditioned discriminative paradigm toward an *open-ended generative paradigm* driven by free-form language instructions. In this setting, the model is expected to jointly perform object semantic understanding, spatial localization, and structured result generation within a unified vision-language framework,

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Traditional SFT (b) RL Optimization (c) GOD (Ours)

Figure 1: Comparison of optimization paradigms for generative object detection. Traditional SFT optimizes coordinates indirectly through next-token prediction, while RL uses IoU rewards but requires costly rollouts. GOD directly injects object-level visual priors into the MLLM, improving localization while preserving efficient token generation.

autonomously discovering, naming, and localizing objects from user instructions without enumerating candidate categories in advance. Existing generative object detection methods can be broadly grouped into two paradigms. The first is *direct generation* [3, 15, 37], which discretizes coordinates into text or specific tokens and autoregressively emits detection results token-by-token. The second is *two-stage retrieval* [7, 16, 29, 34], which first generates class-agnostic regions using an external detection or segmentation module [19, 31] and then asks the language model to index, retrieve, or reason over these candidates; while this strategy partially alleviates the localization difficulty of pure generation, detection quality becomes heavily dependent on the external proposal module, and the proposal stage remains decoupled from the language generation stage, precluding end-to-end optimization within a unified MLLM framework.

We argue that these limitations stem from the lack of explicit object-level visual priors during generation, reflecting a *structural mismatch* between token-level optimization and object-level localization. As illustrated in Figure 1, discriminative detectors directly constrain bounding boxes with object-level objectives such as IoU, L1 regression, and classification losses, enabling efficient optimization of object position and scale. By contrast, generative MLLMs usually decompose a box into discrete coordinate tokens and optimize them with cross-entropy, which provides gradients only for individual numeric symbols rather than for the box as a spatial entity. Reinforcement learning can partially bridge this gap by introducing vision task-aware rewards [11, 15, 41], such as IoU-based rewards, but it requires costly long-sequence rollouts, suffers from low training efficiency, and still does not natively integrate multimodal visual priors. Therefore, *how to explicitly inject object-level visual priors while preserving the open-ended generative capability of MLLMs during pre-training* remains an open challenge for open-ended generative object detection.

To address this optimization mismatch, we propose **GOD** (**G**enerative **O**bject **D**etection), a 3B-scale MLLM that establishes a co-training framework by jointly reconstructing visual priors and generating language descriptions. By introducing an object-query branch and visual optimization objectives directly inside the MLLM, GOD unifies generative language modeling and visual detection in a shared representation space, achieving superior open-vocabulary detection while preserving strong visual perception and reasoning abilities. Specifically, GOD is characterized by three core designs:

- *Architecture Design*: On top of a shared MLLM visual encoder, we introduce a lightweight DETR-style proposal network that extracts object-level query embeddings and candidate proposal boxes from multi-scale visual features. The resulting query embeddings are projected into the LLM token space and injected as conditioning object tokens together with mixed proposals (positive, negative, or none), removing the need for a cumbersome two-tower architecture or an external proposal system.

- *Co-Training Paradigm*: We jointly optimize three complementary objectives: the next-token prediction loss, the MLLM-side object-query reconstruction loss, and the DETR-side detection loss. These losses explicitly align autoregressive language generation with localization-aware visual modeling within a single end-to-end framework.

- *Multi-Stage Training Pipeline*: We first train the DETR-style branch for detection and projector alignment with the MLLM's visual encoder. Subsequently, we conduct full-parameter co-training under all three objectives. Finally, we apply interleaved mask to progressively optimize the MLLM to perform accurate detection without reference proposals.

Extensive experiments on standard detection and grounding benchmarks demonstrate the effectiveness of GOD. The results show that injecting object-level visual priors and optimizing them with co-training substantially improves generative object detection, while preserving the perception, grounding, and language understanding capabilities of the underlying MLLM.

2 Related Work

VLP for Open-Vocabulary Object Detection. Vision-Language Pre-training (VLP) extends conventional closed-set detection by aligning region representations with language so that detectors can generalize to novel categories. Early CLIP-based methods transfer vision-language knowledge to region-level detection through distillation or region-text pre-training [10, 36, 56, 57]. Another influential line formulates detection as grounding and learns fine-grained region-text alignment from grounding, detection, and image-text data [18, 22, 31, 32, 32, 45, 46, 53]. Although these methods substantially improve zero-shot transfer, they still rely on predefined category names or prompts at inference, which limits their applicability when object concepts are unknown or difficult to enumerate.

MLLMs for Open-Ended Object Detection. Conventional OVD remains prompt-conditioned: it requires predefined category names and performs discriminative region-text matching, making it less effective when object concepts are unknown, ambiguous, or better expressed in free-form language. Recent MLLM-based detectors relax this constraint through three main paradigms. Direct generation casts categories and coordinates as autoregressive tokens, extending from sequence-based detectors to grounding-capable and generalist MLLMs [1, 3, 4, 15, 37, 43]. Proposal-assisted retrieval first provides class-agnostic regions or localized visual tokens and then lets the LLM select or reason over them [16, 34, 44]. Labeling or tool-augmented cascades predict object labels or prompts and invoke a box generator, decoder, or external tool for localization [23, 30, 44, 55]. While these approaches broaden detection toward open-ended perception, they either rely on token-level coordinate supervision, decouple proposal generation from language generation, or depend on external modules, leaving visual priors and text generation weakly integrated into the MLLM.

3 Method

In this section, we present GOD, a co-training paradigm for generative object detection that empowers the MLLM to learn task-specific vision priors during fine-tuning. We first review the preliminaries of OVD and MLLMs in Section 3.1, then describe GOD architecture in Section 3.2 along with its co-training supervision in Section 3.3. Finally, we describe the training pipeline in Section 3.4.

3.1 Preliminaries

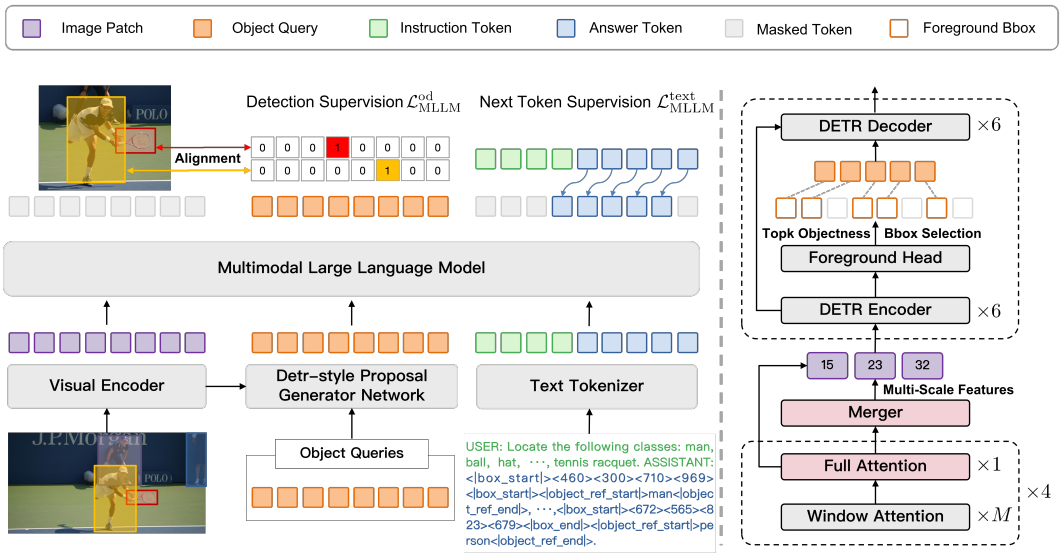
Open-Vocabulary Object Detection. A typical detection model feeds an input image into a CNN or Transformer visual encoder Enc_I and extracts object features $\boldsymbol{O} \in \mathbb{R}^{N\times D}$. Each object feature is fed into a category classifier and a box regressor. Open-vocabulary detection can be viewed as category-conditioned region–text alignment. The category-name prompt is tokenized and encoded by Enc_T to produce category embeddings $\boldsymbol{X} \in \mathbb{R}^{T\times D}$. The classifier applies a sigmoid σ to the region–text dot products to obtain scores $\boldsymbol{S} \in [0,1]^{N\times T}$, supervised by the ground-truth label matrix $\boldsymbol{Y} \in [0,1]^{N\times T}$:

$$\mathcal{L}_{\text{ovd}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}}, \qquad \mathcal{L}_{\text{cls}} = \text{BCE}\big(\sigma(\boldsymbol{O}\boldsymbol{X}^\top), \boldsymbol{Y}\big). \tag{1}$$

Multi-Modal Large Language Models. A conventional LLM models a text-token sequence $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{T}$ in a canonical causal distribution:

$$\text{LLM}(\boldsymbol{X}) = \prod_{i=1}^{T} \text{LLM}(\boldsymbol{x}_i \mid \boldsymbol{X}_{<i}). \tag{2}$$

To extend this formulation to the multimodal setting, recent MLLMs regard a sequence of visual tokens as prefix tokens. Given an image I, a visual encoder Enc_I and a learnable projector Proj_I map it into a sequence of visual tokens $\boldsymbol{V} \in \mathbb{R}^{M\times D}$, which are treated as soft token embeddings that the LLM can attend to. Together with the tokenized user instruction $\boldsymbol{X}_{\text{instruct}}$ (e.g., "*Describe*



(a) The overview of GOD (b) Object query generation

Figure 2: Overview of GOD. The shared visual encoder provides visual tokens to the MLLM and multi-scale features to a DETR-style proposal decoder. Object-query embeddings are injected alongside visual and instruction tokens for structured generation. The displayed token sequence is schematic; task-specific JSON output formats are provided in Appendix B.

this image"), the MLLM generates the output response $\boldsymbol{X}_{\text{ans}}$ autoregressively as

$$\text{MLLM}(\boldsymbol{X}_{\text{ans}} \mid \boldsymbol{V}, \boldsymbol{X}_{\text{instruct}}) = \prod_{i=1}^{T} \text{MLLM}(\boldsymbol{x}_{\text{ans},i} \mid \boldsymbol{V}, \boldsymbol{X}_{\text{instruct}}, \boldsymbol{X}_{\text{ans},<i}), \tag{3}$$

where $\boldsymbol{X}_{\text{ans}}$ denotes the target answer tokens produced by the assistant.

3.2 Model Architecture

Overview. The key idea behind GOD is to unlock the visual subspace within the MLLM that is dedicated to object-level representation, so that the model acquires fine-grained localization ability for visual-centric tasks while preserving its general semantic perception. To this end, we introduce a plug-and-play architectural extension: a lightweight DETR-style decoder is grafted onto the MLLM's visual encoder to extract object-level proposal embeddings (Figure 2). During training, the resulting object query embeddings are fed into the MLLM alongside the original visual and instruction tokens, and the model is jointly supervised by two complementary objectives: reconstruction of the object queries through the detection branch and autoregressive text generation through the language branch. This co-training design separates *object-centric perception* from *response generation* at the module level, yet reunites them within a shared token sequence, enabling the MLLM to leverage detector-grounded evidence without sacrificing its generative strengths.

Object Query Embedding Generation. To adapt the DETR-style decoder to the MLLM without introducing a separate visual encoders [7, 16], GOD reuses the Enc_I of MLLM as the shared feature source for both the language and detection branches. Based on the input image I, the Enc_I produces visual tokens $\boldsymbol{V} \in \mathbb{R}^{M \times D}$ that enter the MLLM as holistic scene context. In parallel, the hidden states of visual features from L global-attention layers of Enc_I are extracted for the detection branch; shallower layers retain fine spatial detail while deeper layers provide stronger semantic abstraction. The selected hidden states are reshaped into spatial feature maps and projected to a common channel dimension, forming a multi-scale feature pyramid $\{\boldsymbol{F}_v^l\}_{l=1}^{L}$ that serves as input to the object decoder.

The key difference from DETR-style detectors arises at the deep fusion and proposal generation stage. Unlike discriminative OVD, where text embeddings are fused with object features to guide proposal selection (Eq. 1), generative detection produces category names autoregressively, making

text-object fusion unavailable at proposal time. Therefore, as illustrated in Figure 2(b), GOD removes all text-object fusion modules from the decoder and replaces the category-dependent proposal scoring with a binary objectness head, Head_{obj} that predicts foreground likelihood at each encoder location. The proposal boxes $\boldsymbol{B} \in \mathbb{R}^{N\times 4}$ at the top-N objectness-scoring locations initialize the decoder reference points, and the decoder refines them into object query embeddings $\boldsymbol{O}^{(L)}$:

$$\begin{aligned} \boldsymbol{H} &= \text{DETR}_{\text{Enc}}\big(\{\boldsymbol{F}_v^l\}_{l=1}^L\big), \quad \mathcal{I}_N = \text{TopN}\big(\text{Head}_{obj}(\boldsymbol{H})\big), \\ \boldsymbol{B} &= \text{Head}_{box}(\boldsymbol{H})_{\mathcal{I}_N}, \quad \boldsymbol{O}^{(L)} = \text{DETR}_{\text{Dec}}\big(\{\boldsymbol{F}_v^l\}_{l=1}^L, \boldsymbol{O}^{(0)}, \boldsymbol{B}\big), \end{aligned} \tag{4}$$

where $\mathcal{I}_N$ indexes the N highest-objectness locations, Head_{box} predicts proposal boxes, and $\boldsymbol{O}^{(0)}$ denotes the initial object queries. Notably, although proposal generation is class-agnostic, the decoder is still trained with region-text alignment supervision as in Eq. 1, so the resulting query embeddings encode category-discriminative semantics facilitating MLLM for grounded detection.

To inject these object-level representations into the language model, the query embeddings are linearly projected into the MLLM token space:

$$\boldsymbol{O} = \text{Proj}_O(\boldsymbol{O}^{(L)}) \in \mathbb{R}^{N\times D}. \tag{5}$$

Input Formatting. The projected queries $\boldsymbol{O}$ are concatenated with the visual tokens $\boldsymbol{V}$ and the instruction tokens $\boldsymbol{X}_{\text{instruct}}$ to form the complete conditioning context. The MLLM then generates the output response autoregressively as

$$\text{MLLM}(\boldsymbol{X}_{\text{ans}} \mid \boldsymbol{V}, \boldsymbol{O}, \boldsymbol{X}_{\text{instruct}}) = \prod_{i=1}^{T} \text{MLLM}\big(\boldsymbol{x}_{\text{ans},i} \mid \boldsymbol{V}, \boldsymbol{O}, \boldsymbol{X}_{\text{instruct}}, \boldsymbol{X}_{\text{ans},<i}\big). \tag{6}$$

A schematic prompt-response pair for open-vocabulary object detection

User: <|image_start|> <image_tokens> <|image_end|> <|object_start|> <object$_1$> $\cdots$ <object$_n$> <|object_end|>. Locate the classes in the image: [man, ball, hat, $\cdots$, tennis racquet].

Assistant: <|box_start|> <460> <300> <710> <969> <|box_end|> <|object_ref_start|> man <|object_ref_end|>, $\cdots$, <|box_start|> <672> <565> <823> <679> <|box_end|> <|object_ref_start|> tennis racquet <|object_ref_end|>.

The prompt-response pair in the box above illustrates one concrete instance of this serialization strategy. On the user side, the instruction text and the image placeholder are followed by a reserved object segment delimited by <|object_start|> and <|object_end|>, within which <object$_i$> denotes the i-th inserted object-query embedding $\boldsymbol{o}_i$. On the assistant side, the model directly produces grounded textual content, including category names and quantized coordinate values between <0> and <999>, each represented as a special token in the MLLM's vocabulary. This example clarifies that the object query embeddings enter the MLLM as regularization tokens for visual task comprehension rather than as external detection results for the MLLM to reference [16, 34].

Denoising Augmentation. After specifying the architecture and input serialization, we introduce denoising augmentation to improve localization robustness. During co-training, GOD uses two complementary denoising mechanisms. First, on the detection branch, perturbed noise objects are injected into the initial query set $\boldsymbol{O}^{(0)}$, encouraging the DETR-style decoder to recover high-quality proposals from corrupted references [31, 52]. Second, on the generation branch, noise objects are appended to the assistant response as auxiliary targets, but next-token prediction supervision is imposed only on their category-name tokens; the coordinate tokens are excluded from the loss to prevent the MLLM from memorizing corrupted coordinates as valid outputs. By combining denoising detection with noise-injected sequence modeling [3], this design helps mitigate duplicate predictions and coordinate drift in generative detection. Table 5 analyzes how CDN queries should be masked under the causal generation setting. The MLLM-side denoising supervision can be written as

$$\mathcal{L}_{\text{MLLM}}^{\text{noise}} = -\sum_{j\in\mathcal{J}_{\text{cls}}} \log \text{MLLM}\big(\boldsymbol{x}_{\text{noise},j} \mid \boldsymbol{V}, \boldsymbol{O}, \boldsymbol{X}_{\text{instruct}}, \boldsymbol{X}_{\text{noise},<j}\big). \tag{7}$$

Here, $\mathcal{J}_{\text{cls}}$ denotes the positions of category-name tokens in the noise response sequence. The remaining positions, such as discretized coordinate tokens from perturbed boxes, are used only as known prefixes in the autoregressive forward pass and are omitted from the cross-entropy objective. For notational simplicity, we absorb $\mathcal{L}_{\text{MLLM}}^{\text{noise}}$ into $\mathcal{L}_{\text{MLLM}}^{\text{text}}$ in the following loss formulation.

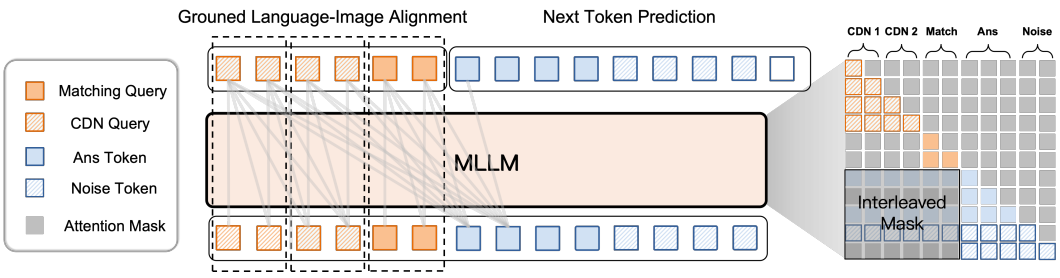


Figure 3: Interleaved attention mask with denoising augmentation.

3.3 Co-Training Objective

Co-Training Loss. To preserve both the generative ability of the MLLM and its object localization capability, GOD optimizes a three-part co-training objective:

$$\mathcal{L}_{\text{GOD}} = \mathcal{L}^{\text{text}}_{\text{MLLM}} + \mathcal{L}^{\text{ovd}}_{\text{MLLM}} + \mathcal{L}^{\text{ovd}}_{\text{DETR}}. \tag{8}$$

$\mathcal{L}^{\text{text}}_{\text{MLLM}}$ is the standard token-level prediction loss on assistant response tokens as in Eq. 6, which preserves the instruction-following and natural language generation ability of the MLLM. $\mathcal{L}^{\text{ovd}}_{\text{MLLM}}$ applies OVD supervision to the object-query tokens within the multimodal sequence, encouraging the language model to remodel object-level representations in its own token space so that they better serve detection-style generation. $\mathcal{L}^{\text{ovd}}_{\text{DETR}}$ is applied to the DETR-style proposal network to optimize proposal quality, objectness estimation, and box regression. Together, these terms bridge detector-style perception and language-model-style generation: using only generation loss provides no direct spatial constraint, whereas using only detection loss cannot teach the MLLM how to consume object-query embeddings during grounded response generation.

Curriculum Learning with Interleaved Attention Mask. The motivation for visual-prior remodeling is dual-mode inference: GOD should operate both with object query embeddings $\boldsymbol{O}$ and without explicitly provided $\boldsymbol{O}$. The former mode exploits detector-provided object-level priors for more accurate localization, whereas the latter ensures that the MLLM can still generate detection results from its own visual representations. However, if the answer tokens $\boldsymbol{X}_{\text{ans}}$ can always attend to $\boldsymbol{O}$ during training, the model may develop an overly strong dependency on explicit object queries and weaken its query-free inference ability. Therefore, as shown in Figure 3, GOD introduces an *interleaved attention mask* between $\boldsymbol{O}$ and $\boldsymbol{X}_{\text{ans}}$. At the beginning of training, the mask is set to all ones to fully learn to exploit detector-grounded priors. Next, the mask is randomly sparsified or zeroed out, forcing the model to rely more on its own visual representations when generating detection outputs.

Because the visibility between $\boldsymbol{O}$ and $\boldsymbol{X}_{\text{ans}}$ changes throughout training, the remodeling loss $\mathcal{L}^{\text{ovd}}_{\text{MLLM}}$ must be coordinated with the masking schedule. We therefore consider three supervision variants:

- *GOD-Full*: $\mathcal{L}^{\text{ovd}}_{\text{MLLM}}$ is applied to all object query embeddings $\boldsymbol{O}$, regardless of whether $\boldsymbol{O}$ is visible to $\boldsymbol{X}_{\text{ans}}$ under the current mask. This variant provides the strongest remodeling signal, requiring the model to maintain object-level semantics in $\boldsymbol{O}$ even when answer tokens cannot attend to the object queries.

- *GOD-Visible*: $\mathcal{L}^{\text{ovd}}_{\text{MLLM}}$ is still applied, but the answer-side remodeling supervision is restricted to positions where $\boldsymbol{O}$ remains visible to $\boldsymbol{X}_{\text{ans}}$ under the interleaved mask, rather than being applied uniformly across all answer positions. This variant keeps the supervision scope consistent with the actual information flow: the MLLM is asked to exploit object-level semantics only when it can attend to the object queries.

- *GOD-Detach*: The remodeling loss $\mathcal{L}^{\text{ovd}}_{\text{MLLM}}$ on $\boldsymbol{X}_{\text{ans}}$ is removed entirely; the proposal network is regularized solely by $\mathcal{L}^{\text{ovd}}_{\text{DETR}}$. This variant fully decouples object-query quality from the language generation branch, still encouraging high-quality proposals without imposing a sequence-level remodeling constraint on the MLLM.

Instead of maximizing query reconstruction alone, these variants identify the supervision configuration that best supports grounded generation in both inference modes.

Method	Base model	Paradigm	COCO val	ODinW13 test
			AP	AP_{avg}
LLMDet-L‡ [9]	Swin-L	Discriminative Two Tower	59.2	-
WeDetect-L [7]	ConvNeXt-L		54.5	-
ObjEmbedded-4B‡ [8]	SigLIP2-L		53.0	-
GOD-Stage1-DETR-OVD-3B	ViT-H	Discriminative Single Tower	46.0	-
GOD-Stage2-DETR-OVD-3B	ViT-H		53.2	-
GOD-Stage2-MLLM-OVD-3B	ViT-H		53.7	-
VLM-FO1-3B [29]	Qwen2.5-3B-VL	Generative Retrieval	44.4	44.0
ChatRex-7B [16]	Qwen2.5-7B		48.2*	-
WeDetect-Ref-4B [7]	Qwen3-4B-VL		50.0	47.3
Qwen2.5-7B-VL [38]	-	Generative Direct	17.7	37.3*
Grriffon-G-7B [51]	LLaMA-7B		39.8†	43.8*
Rex-Omni-3B [15]	Qwen2.5-3B-VL		30.8†	-
GOD-MLLM-3B	Qwen2.5-3B-VL		38.1	47.9

Table 1: Object detection results on mainstream benchmarks. The GOD COCO result of 38.1 is the MLLM-direct result; Table 16 separately reports the MLLM-ref result of 38.56. † denotes reproduction under our settings; * denotes prompts restricted to categories present in the image.

3.4 Training Pipeline

GOD is trained in two stages: (*i*) detection and alignment pre-training for the object decoder and its adapter to the MLLM, followed by (*ii*) curriculum fine-tuning for joint OVD and next-token prediction.

Stage-1: Detection and Alignment Pre-Training. The part *(a)* of pre-training attaches the DETR-style object decoder (DETR_{Enc}, DETR_{Dec}, Head_{obj}) to the shared visual encoder Enc_I. The visual encoder is kept frozen and provides multi-scale features; only the object decoder and objectness head are trainable, optimized with the detection loss:

$$\mathcal{L}_{\text{DETR}}^{\text{ovd}} = \mathcal{L}_{\text{DETR}}^{\text{box}} + \mathcal{L}_{\text{DETR}}^{\text{cls}} + \mathcal{L}_{\text{DETR}}^{\text{obj}}, \tag{9}$$

so that the decoder learns to generate accurate proposal boxes and semantically rich object query embeddings. The part *(b)* freezes the object decoder and trains only the projector Proj_O to map the resulting object query embeddings into the MLLM token space, while the MLLM itself also remains frozen.

Stage-2: Curriculum Fine-Tuning. In part *(a)*, all modules are unfrozen and jointly optimized for one epoch with the co-training objective $\mathcal{L}_{\text{GOD}}$ (Eq. 8). Part *(b)* then trains only the LLM for one epoch with the interleaved attention-mask schedule to progressively optimize detection without reference proposals, while the other modules remain frozen. Stage-2 therefore comprises two epochs in total.

Stage-2b fine-tunes only the LLM; the ViT, the BERT text encoder, and the DETR branch are all frozen. Its objective therefore contains only the two MLLM-side losses, $\mathcal{L}_{\text{MLLM}}^{\text{ovd}}$ and $\mathcal{L}_{\text{MLLM}}^{\text{text}}$; there is no $\mathcal{L}_{\text{DETR}}^{\text{ovd}}$ term, and hence no backpropagation into the shared encoder in this phase.

Building on the detection capability and token-space alignment obtained in Stage-1, this fine-tuning stage treats object-query remodeling as a regularizer that stabilizes object-query prediction and encourages the generation branch to continuously exploit reliable spatial priors; the differences among *GOD-Full*, *GOD-Visible*, and *GOD-Detach* are described in the preceding subsection. Appendices A and C provide the corresponding data composition and optimization settings.

4 Experiments

4.1 Training Detail

Setup. We instantiate GOD with Qwen2.5-VL-3B-Instruct [38] as the base MLLM. The DETR branch of GroundingDINO [31] serves as the object-query generator and is aligned with a BERT-Base text encoder [5]. Stage 1 sequentially trains the DETR branch and the object-query projector

Proj_O, while Stage 2 performs curriculum fine-tuning with one epoch each for Stage-2a and Stage-2b.

For the DETR branch, we use 900 object queries. For training efficiency, only the top 300 queries are used for reconstructive supervision, while the remaining queries are kept for DETR-style objectness prediction. The maximum new-token length of the MLLM is 1024, the maximum number of objects is 45, and the maximum number of alignment tokens for reconstructive detection is 256.

Training Dataset. We curate detection, grounding, grounded image captioning, referring expression comprehension, region-level tasks, and general image instruction mixtures. For MLLM generation labeling, object detection uses both category-conditioned and free-form prompt templates. For grounding, we use six instruction–answer forms that combine regions, categories, entity phrases, and captions or descriptions to produce direct grounding or grounded-captioning outputs. In reconstructive alignment, DETR supervision is tied to the category names and entity phrases expressed in the caption. Appendices A, B, and C provide the full details.

4.2 Main Results

We distinguish DETR outputs, MLLM-remodeled discriminative outputs, and the two generative inference modes using the notation defined in Appendix D.1.

The COCO comparison between GOD (38.1) and the reproduced Rex-Omni (30.8) uses the same general-category template and Qwen2.5-VL-3B backbone. Appendix D.2 reports the matched comparison and the evaluation protocol.

Discriminative Object Detection. Table 1 first compares the discriminative outputs of GOD with representative two-tower detectors on COCO val. These baselines use detection-specific visual backbones separated from the MLLM; by contrast, GOD reuses the MLLM visual encoder in a single-tower architecture and does not introduce extra deep fusion modules. Under this more constrained design, Stage-2 co-training improves the DETR output from 46.0 to 53.2 AP, matching the results of two-tower detectors. GOD-Stage2-MLLM-OVD obtains a comparable 53.7 AP after remodeling the DETR queries in the MLLM sequence space, indicating that the injected object queries retain effective localization and category semantics after language-side modeling.

Generative Object Detection. Table 1 also evaluates generative detection on COCO val and ODinW13. Existing generative methods either retrieve indices from predefined regions or directly generate box coordinates; the former inherits localization quality from an external detector, while the latter requires the MLLM to perform spatial regression autoregressively. Following the direct coordinate-generation paradigm, GOD-MLLM-3B achieves 38.1 AP on COCO and 47.9 AP_{avg} on ODinW13, outperforming the same-scale Rex-Omni-3B by 7.3 AP under our reproduced setting and surpassing Qwen2.5-VL-7B by a large margin. These results show that co-training injects detector-derived object priors that substantially improve autoregressive localization, although retrieval-based methods can still benefit from externally supplied proposals.

Referring Expression Comprehension. Table 2 reports the REC task results on RefCOCO/+/g. GOD-MLLM-Ref-3B achieves accuracy comparable to or better than several 7B-scale methods, and reaches 82.0 on the linguistically challenging RefCOCO+ testB split. This suggests that the object-level representations learned through co-training not only support open-ended detection but also transfer effectively to language-conditioned region understanding.

4.3 Ablation

Co-Training Losses. Table 3 ablates $\mathcal{L}^{\text{ovd}}_{\text{DETR}}$ and $\mathcal{L}^{\text{ovd}}_{\text{MLLM}}$ in the lightweight COCO co-training setting, asking what supervision is required for detector proposals to benefit a single-tower generative framework. Without detection supervision (Row 2), injected proposals behave as unanchored noisy tokens: both with-proposal generation and the DETR output fall below the proposal-free baseline (14.2 *v.s.* 16.7). Using only $\mathcal{L}^{\text{ovd}}_{\text{DETR}}$ produces spatially reliable proposals but yields limited generative gains (Row 3); in contrast, using only $\mathcal{L}^{\text{ovd}}_{\text{MLLM}}$ remodels language-friendly query tokens while collapsing spatial anchoring (Row 4, $\boldsymbol{O}_{\text{MLLM}} = 43.9$ and $\boldsymbol{O}_{\text{DETR}} = 2.3$). Jointly optimizing both losses achieves the best with-proposal generation (32.4) and strong discriminative outputs (50.2/49.0), showing that $\mathcal{L}^{\text{ovd}}_{\text{DETR}}$ provides reliable foreground anchors while $\mathcal{L}^{\text{ovd}}_{\text{MLLM}}$ makes them interpretable to the language decoder. The proposal-free score in Row 5 remains 16.7, indicating a

Method	Paradigm	RefCOCO			RefCOCO+			RefCOCOg	
		val	testA	testB	val	testA	testB	val	test
Grounding-DINO-L [31]	Discriminative	90.6	93.2	88.2	82.8	89.0	75.9	86.1	87.0
VLM-FO1-3B [29]	Index-style gen.	91.1	93.7	87.6	86.4	91.9	80.6	88.9	88.3
Octopus-7B [54]		89.0	92.6	83.4	83.6	89.4	76.0	84.3	86.3
ChatRex-7B [16]		91.0	94.1	87.0	89.8	91.9	79.3	89.8	90.0
Qwen2.5-VL-3B [38]	Coordinate gen.	89.1	91.7	84.0	82.4	88.0	74.1	85.2	85.7
VLM-R1-3B [41]		90.1	92.3	85.2	84.2	89.4	76.8	85.6	86.8
Rex-Omni-3B [15]		86.6	89.5	82.8	79.6	84.8	71.4	85.3	86.2
GOD-MLLM-3B		92.1	**94.6**	**89.0**	87.5	91.7	**82.0**	89.5	**90.1**

Table 2: Referring expression comprehension results.

Row	Proposal curriculum	Gradient	Loss terms		Generative		Discriminative	
		DETR	$\mathcal{L}_{\text{DETR}}^{\text{ovd}}$	$\mathcal{L}_{\text{MLLM}}^{\text{ovd}}$	w/o prop.	w/ prop.	O_{DETR}	O_{MLLM}
1	○	-	-	-	16.7	-	-	-
2	●	✓	-	-	14.2	13.6	10.5	-
3	●	✓	✓	-	14.9	21.8	50.4	-
4	●	✓	-	✓	18.0	17.3	2.3	43.9
5	●	✓	✓	✓	16.7	32.4	50.2	49.0

Table 3: Ablation on co-training loss terms for GOD. Except for the gradients and loss terms listed, all other settings keep the visual encoder and language model trainable and always optimize $\mathcal{L}_{\text{MLLM}}^{\text{text}}$.

reliance on explicit object queries and motivating the following curriculum schedule. Those rows are always trained with proposals, but the proposal-free distribution is never optimized for. This train/inference mismatch leaves the result at the baseline (Row 5, 16.7 = Row 1). Table 5 isolates how CDN noise queries affect this co-training setting.

Curriculum and Interleaved-Mask Supervision. Table 4 examines how appropriately supervising reference proposals under the interleaved mask closes the training–inference gap introduced by proposal-conditioned co-training. Row 2 exposes this dependence: when proposals are fully visible, with-proposal generation reaches 32.4, while proposal-free generation remains at 16.7, matching the pure generative baseline. Directly applying the interleaved mask (Row 3) raises proposal-free/with-proposal accuracy to 33.3/33.7; the full two-stage curriculum (Row 5) improves them to 35.9/37.3 by first establishing object-level grounding with visible proposals and then gradually reducing proposal visibility. Both the curriculum and the no-curriculum variant are trained for 2 epochs in total; the curriculum allocates one epoch to Stage-2a and one to Stage-2b, so its gain does not come from longer training.

Among the Stage-2 variants, GOD-Full achieves the strongest generative performance while preserving stable discriminative outputs (52.9/52.6); by comparison, GOD-Visible provides insufficient remodeling supervision, and GOD-Detach collapses O_{MLLM} to 0.4 despite retaining reasonable generation. These results show that co-training remains essential throughout curriculum fine-tuning: $\mathcal{L}_{\text{MLLM}}^{\text{ovd}}$ acts as a semantic regularizer that continuously anchors DETR-derived object priors in the MLLM token space, enabling discriminative localization and generative decoding to improve within a unified framework. Appendix F provides a more detailed discussion of how this regularization maintains consistency between detection and generation.

Denoising Augmentation. Table 5 shows that CDN scarcely changes discriminative AP but improves generation. Because CDN queries are removed at inference, matched queries must remain independent of this training-only context: allowing both matched queries and answer tokens to attend to CDN reduces proposal-free AP to 6.1 (Row 3). Restricting CDN visibility to answer tokens yields the best proposal-assisted AP (32.4, Row 4), although its proposal-free AP (16.7) remains below Row 2 (21.2).

4.4 Open-Ended Detection and General Multimodal Ability

Neither the MLLM nor the detection branch receives a category list: the text branch uses only `object.`, and the instruction asks for boxes and free-form names. GOD emits 528 distinct labels and reaches 26.9 mAP on full COCO val2017 after offline sentence-embedding alignment to COCO's

Row	Proposal curriculum	Variant	Loss terms		Generative		Discriminative	
			$\mathcal{L}_{\text{MLLM}}^{\text{ovd}_{\text{ref}}}$	$\mathcal{L}_{\text{MLLM}}^{\text{ovd}_{\text{noref}}}$	w/o prop.	w/ prop.	O_{DETR}	O_{MLLM}
1	○	Baseline	-	-	16.7	-	-	-
2	●	Stage-2a	✓	-	16.7	32.4	50.2	49.0
3	◐	GOD-Full	✓	✓	33.3	33.7	51.0	51.2
4	◐	GOD-Visible	✓	-	33.3	33.7	50.9	50.4
5	● → ◐	GOD-Full	✓	✓	35.9	37.3	52.9	52.6
6	● → ◐	GOD-Visible	✓	-	33.0	33.7	51.2	51.1
7	● → ◐	GOD-Detach	-	-	33.9	34.2	51.2	0.4

Table 4: Ablation on interleaved-mask training variants for GOD.

Row	Proposal curriculum	Noise masks		Generative		Discriminative	
		Ans → CDN	Match → CDN	w/o prop.	w/ prop.	O_{DETR}	O_{MLLM}
1	Only Match	-	-	17.5	18.9	50.3	49.3
2	CDN + Match	-	-	21.2	22.5	50.0	50.4
3	CDN + Match	✓	✓	6.1	24.0	50.1	45.0
4	CDN + Match	✓	-	16.7	32.4	50.2	49.0

Table 5: Ablation on noise-augmentation configurations for GOD.

80 classes solely for evaluation. Table 6 also reports category-agnostic R@mIoU, which ignores category labels when matching boxes and measures localization independently of naming; Rex-Omni's generic "object" outputs explain its 0.1 mAP despite 38.3 R@mIoU.

To substantiate this empirically, we evaluated GOD on standard general-MLLM benchmarks, where it retains strong multimodal understanding (Table 7).

Model	mAP	R@mIoU
Rex-Omni	0.1	38.3
GoD-mllm-ref	26.9	30.1

Table 6: Category-free detection on COCO.

Benchmark	GOD-MLLM-Direct
MMMU	45
POPE	86
MME	1520

Table 7: General MLLM benchmark results.

Inference Overhead. Table 18 in Appendix D.4 compares identical images and prompts on the same GPU: MLLM-Ref adds $\sim$ 143 M parameters (+3.8%) and 302 input tokens, with the DETR branch running only during prefill. End-to-end generation over the 10 COCO images rises from 7.68 s to 8.66 s (+12.7%), while cached decode FLOPs are unchanged. This overhead is optional: MLLM-direct bypasses the DETR branch and its multi-scale feature path while retaining 38.1 AP on COCO (Table 1).

5 Conclusion

In this paper, we present GOD, a generative object detection model that brings object-level visual priors into MLLM training. By adding a lightweight DETR-style query branch and co-training it with the language generation objective, GOD reduces the mismatch between token-level prediction and spatial localization. This allows the model to preserve the flexibility of open-ended generation while learning object-centric representations that remain useful for detection. The interleaved-mask curriculum further helps the generative branch operate without always relying on explicit proposal tokens. Experiments show consistent gains on open-ended detection and referring expression comprehension, while the ablations confirm the importance of co-training and curriculum masking for transferring detector priors to the generative branch. A remaining challenge is to make such generative detectors reliable under free form prompts and more standardized evaluation protocols.

References

[1] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[2] Guiming Hardy Chen, Shunian Chen, Ruifei Zhang, Junying Chen, Xiangbo Wu, Zhiyi Zhang, Zhihong Chen, Jianquan Li, Xiang Wan, and Benyou Wang. ALLaVA: Harnessing GPT-4V-synthesized data for a lite vision-language model. *arXiv preprint arXiv:2402.11684*, 2024.

[3] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852*, 2021.

[4] 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.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[6] Dawei Du, Pengfei Zhu, Longyin Wen, Xiao Bian, Haibin Lin, Qinghua Hu, Tao Peng, Jiayu Zheng, Xinyao Wang, Yue Zhang, et al. VisDrone-DET2019: The vision meets drone object detection in image challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 0–0, 2019.

[7] Shenghao Fu, Yukun Su, Fengyun Rao, Jing Lyu, Xiaohua Xie, and Wei-Shi Zheng. Wedetect: Fast open-vocabulary object detection as retrieval. *arXiv preprint arXiv:2512.12309*, 2025.

[8] Shenghao Fu, Yukun Su, Fengyun Rao, Jing Lyu, Xiaohua Xie, and Wei-Shi Zheng. ObjEmbed: Towards universal multimodal object embeddings. *arXiv preprint arXiv:2602.01753*, 2026.

[9] Shenghao Fu, Qize Yang, Qijie Mo, Junkai Yan, Xihan Wei, Jingke Meng, Xiaohua Xie, and Wei-Shi Zheng. Llmdet: Learning strong open-vocabulary object detectors under the supervision of large language models. In *CVPR*, 2025.

[10] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *International Conference on Learning Representations*, 2021.

[11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[12] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.

[13] Shuting He, Henghui Ding, Chang Liu, and Xudong Jiang. GREC: Generalized referring expression comprehension. *arXiv preprint arXiv:2308.16182*, 2023.

[14] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021.

[15] Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. *arXiv preprint arXiv:2510.12798*, 2025.

[16] Qing Jiang, Gen Luo, Yuqin Yang, Yuda Xiong, Yihao Chen, Zhaoyang Zeng, Tianhe Ren, and Lei Zhang. Chatrex: Taming multimodal llm for joint perception and understanding. *arXiv preprint arXiv:2411.18363*, 2024.

[17] Qing Jiang, Lin Wu, Zhaoyang Zeng, Tianhe Ren, Yuda Xiong, Yihao Chen, Qin Liu, and Lei Zhang. Referring to any person. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[18] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021.

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[20] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

[21] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M. Ni, and Lei Zhang. DN-DETR: Accelerate DETR training by introducing query DeNoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13619–13627, 2022.

[22] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022.

[23] Chuang Lin, Yi Jiang, Lizhen Qu, Zehuan Yuan, and Jianfei Cai. Generative region-language pretraining for open-ended object detection. In *CVPR*, 2024.

[24] 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 *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[25] 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 *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[26] Weifeng Lin, Xinyu Wei, Ruichuan An, Peng Gao, Bocheng Zou, Yulin Luo, Siyuan Huang, Shanghang Zhang, and Hongsheng Li. Draw-and-understand: Leveraging visual prompts to enable MLLMs to comprehend what you want. *arXiv preprint arXiv:2403.20271*, 2024.

[27] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023.

[28] Junzhuo Liu, Xuzheng Yang, Weiwei Li, and Peng Wang. FineCops-Ref: A new dataset and task for fine-grained compositional referring expression comprehension. *arXiv preprint arXiv:2409.14750*, 2024.

[29] Peng Liu, Haozhan Shen, Chunxin Fang, Zhicheng Sun, Jiajia Liao, and Tiancheng Zhao. Vlm-fo1: Bridging the gap between high-level reasoning and fine-grained perception in vlms. *arXiv preprint arXiv:2509.25916*, 2025.

[30] Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, Lei Zhang, Jianfeng Gao, and Chunyuan Li. Llava-plus: Learning to use tools for creating multimodal agents. *arXiv preprint arXiv:2311.05437*, 2023.

[31] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[32] Yang Liu, Feng Hou, Yunjie Peng, Gangjian Zhang, Yao Zhang, Dong Xie, Peng Wang, Yang Zhang, Jiang Tian, Zhongchao Shi, et al. Doga: Enhancing grounded object detection via grouped pre-training with attributes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 5658–5666, 2025.

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[34] Chuofan Ma, Yi Jiang, Jiannan Wu, Zehuan Yuan, and Xiaojuan Qi. Groma: Localized visual tokenization for grounding multimodal large language models. In *ECCV*, 2024.

[35] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2016.

[36] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *European Conference on Computer Vision*, pages 728–755. Springer, 2022.

[37] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023.

[38] Qwen Team. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[39] 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 *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[40] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8430–8439, 2019.

[41] Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, Ruochen Xu, and Tiancheng Zhao. VLM-R1: A stable and generalizable R1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

[42] Jiaqi Wang, Pan Zhang, Tao Chu, Yuhang Cao, Yujie Zhou, Tong Wu, Bin Wang, Conghui He, and Dahua Lin. V3Det: Vast vocabulary visual detection dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19844–19854, 2023.

[43] 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. *NeurIPS*, 36, 2023.

[44] Jiannan Wu, Muyan Zhong, Sen Xing, Zeqiang Lai, Zhaoyang Liu, Zhe Chen, Wenhai Wang, 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. *Advances in Neural Information Processing Systems*, 37:69925–69975, 2024.

[45] Lewei Yao, Jianhua Han, Xiaodan Liang, Dan Xu, Wei Zhang, Zhenguo Li, and Hang Xu. Detclipv2: Scalable open-vocabulary object detection pre-training via word-region alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23497–23506, 2023.

[46] Lewei Yao, Jianhua Han, Youpeng Wen, Xiaodan Liang, Dan Xu, Wei Zhang, Zhenguo Li, Chunjing Xu, and Hang Xu. Detclip: Dictionary-enriched visual-concept paralleled pre-training for open-world detection. In *Advances in Neural Information Processing Systems*, volume 35, pages 9125–9138, 2022.

[47] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. In *Transactions of the Association for Computational Linguistics*, volume 2, pages 67–78. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2014.

[48] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer, 2016.

[49] Yuqian Yuan, Wentong Li, Jian Liu, Dongqi Tang, Xinjie Luo, Chi Qin, Lei Zhang, and Jianke Zhu. Osprey: Pixel understanding with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28202–28211, 2024.

[50] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6720–6731, 2019.

[51] Yufei Zhan, Hongyin Zhao, Yousong Zhu, Fan Yang, Ming Tang, and Jinqiao Wang. Griffon-g: Bridging vision-language and vision-centric tasks via large multimodal models. *arXiv preprint arXiv:2410.16163*, 2024.

[52] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *The Eleventh International Conference on Learning Representations*, 2022.

[53] Haotian Zhang, Pengchuan Zhang, Xiaowei Hu, Yen-Chun Chen, Liunian Li, Xiyang Dai, Lijuan Wang, Lu Yuan, Jenq-Neng Hwang, and Jianfeng Gao. Glipv2: Unifying localization and vision-language understanding. In *Advances in Neural Information Processing Systems*, volume 35, pages 36067–36080, 2022.

[54] Chuyang Zhao, YuXin Song, Junru Chen, Kang Rong, Haocheng Feng, Gang Zhang, Shufan Ji, Jingdong Wang, Errui Ding, and Yifan Sun. Octopus: A multi-modal LLM with parallel recognition and sequential understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[55] Haoyu Zhao, Wenhang Ge, and Ying-Cong Chen. Llm-optic: Unveiling the capabilities of large language models for universal visual grounding. *arXiv preprint arXiv:2405.17104*, 2024.

[56] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, et al. Regionclip: Region-based language-image pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16793–16803, 2022.

[57] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *European Conference on Computer Vision*, pages 350–368. Springer, 2022.

Technical appendices and supplementary material

A Training Data

To supplement the training pipeline in Section 3.4 and the experimental setup in Section 4.1, we give the detailed data composition below. GOD is trained in two stages: *region–language alignment pre-training* (Stage 1) and *curriculum fine-tuning* (Stage 2). The two stages correspond to different granularities of supervision. The pre-training stage mainly uses object detection and referring-style supervision, so that visual region representations and language semantics are stably aligned at the object and referring-expression levels. The curriculum fine-tuning stage extends this alignment to fine-grained perceptual abilities such as region description, region question answering, grounded image captioning, and multi-target referring comprehension; general visual instruction data are also mixed in to alleviate catastrophic forgetting. We organize the training data by task type in Tables 8 and 9.

Pre-training stage. As shown in Table 8, Stage 1 pre-trains the DETR detection branch and optimizes the projection layer that maps object-query embeddings into the MLLM feature space. During DETR training, we use two types of tasks.

- *Object detection* combines COCO [25], LVIS [12], ODinW [22], Objects365 [40], V3Det [42], and VisDrone2019 [6]. These datasets cover general scenes, long-tail categories, fine-grained category systems, and drone-view imagery. Together, they provide diverse visual–semantic correspondences for box-level localization and open-vocabulary category modeling, with about 2.24M images in total. Objects365, with 1.72M images, forms the main body of detection supervision; LVIS and V3Det focus on long-tail distributions and fine-grained categories, respectively; ODinW and VisDrone introduce domain-specific scenarios such as industrial and aerial-view scenes.

- *Referring-expression comprehension and phrase grounding* use GoldG [18], RefCOCO/+/g [35, 48], and gRefCOCO [13]. GoldG, introduced with MDETR, is built from sources such as Flickr30k Entities and Visual Genome and provides multi-entity phrase–region correspondences; the RefCOCO family focuses on localizing a single referred object; gRefCOCO supports generalized referring-expression comprehension. These grounding sources contain about 1.40M samples in total. Through joint pre-training on the above detection and grounding tasks, the DETR branch learns to generate high-quality proposals and category-discriminative object-query embeddings. During the object-query mapping step, we warm up the projection layer with ALLaVA-4V-Caption [2], completing the interface alignment between the detection branch and the language branch.

Curriculum fine-tuning stage. As shown in Table 9, Stage 2 keeps the detection and referring supervision from Stage 1 and adds five dataset groups: region description, region QA, grounded image captioning, multi-target referring comprehension, and general visual instruction data. In our prompt design, region description and region QA share a region-level instruction objective, so the five dataset groups are organized into four task families. Because the annotation formats of existing datasets do not directly match our task definitions, we convert the data for each task accordingly.

- *Region description and region QA* directly use the original annotation formats of Osprey [49], Rexverse-2M [16], VCR [50], and related datasets. These samples do not participate in the object-query reconstruction loss or the DETR detection loss; their optimization objective is the same as that of general visual instruction data, and they are used to strengthen the model's understanding of the mapping between region coordinates and semantic descriptions.

- *Grounded image captioning* requires the model to generate an image-level description while annotating every mentioned entity with bounding-box coordinates, and therefore requires high spatial coverage. To construct this task, we first reorganize phrase-level region annotations from sources such as Flickr30k [47], gRefCOCO [13], and FineCops-Ref [28] into coherent image-level descriptions. We then keep samples whose entity boxes sufficiently cover the main image content, so that the grounded captions learned by the model retain global coverage. For DETR-branch alignment, the entities mentioned in the caption are used as supervision signals for object queries.

Task	Dataset		Volume
Object Detection	COCO [25] (117K) LVIS [12] (96K) ODinW [22] (123K)	Objects365 [40] (1.72M) V3Det [42] (180K) VisDrone2019 [6] (3.8K)	2.24M
REC & Grounding	GoldG [18] (0.8M) RefCOCO/+/g [35, 48] (321K)	gRefCOCO [13] (278K)	1.40M
Image Caption	ALLaVA-4V-Caption [2] (636K)		636K

Table 8: Training data for GOD in the pre-training stage, including object detection, grounding supervision, and image-caption data. Sizes in parentheses are dataset volumes.

Task	Dataset		Volume
Region Description	Osprey [49] (197K)	Rexverse-2M [16] (500K)	697K
Region QA	Osprey [49] (521K)	VCR [50] (213K)	734K
REC	Flickr30k [47] (30K) gRefCOCO [13] (15K)	FineCops-Ref [28] (29K)	74K
Region REC	Rexverse-2M [16] (500K) MDVP [26] (752K)	HumanRef [17] (42K)	1.29M
General Visual Instruction	ALLaVA-4V-Instruct [2] (663K) LLaVA-1.5-Instruct [27] (665K)	LLaVA-OneVision [20] (1.2M)	2.53M

Table 9: Region-related and general data for GOD in the curriculum fine-tuning stage, including region captioning, region QA, grounded image captioning, multi-target referring-expression comprehension, and general visual instruction data. Sizes in parentheses are post-deduplication counts; "Volume" is the sum over listed datasets.

- *Multi-target referring comprehension* asks the model, given a region description and several referring expressions, to output the bounding box and object category for each expression. However, the original annotations in Rexverse-2M [16], MDVP [26], HumanRef [17], and related datasets usually provide only a binary region–description correspondence. We therefore convert them into (region, entity phrase, description) triples: resolvable entity phrases are extracted from the description and bound to the corresponding region, enabling the model to localize targets and output detection boxes conditioned on both entity names and contextual descriptions.

- *General visual instruction* includes ALLaVA-4V-Instruct [2], LLaVA-1.5-Instruct [27], and LLaVA-OneVision [20]. These data help maintain the model's high-level scene understanding and multimodal dialogue ability, preventing overfitting to region-level tasks from degrading general conversational performance.

B Prompt Templates

Prompt construction, briefly summarized in Section 4.1, is detailed below. During instruction tuning, all training samples are organized with natural-language instruction templates before being fed into the model. As shown in Tables 10–13, each task is assigned a set of instruction templates, from which one instruction is randomly sampled during training. For tasks with structured outputs, we append an output-format constraint; for detection and grounding tasks, we additionally use a phase-dependent noise-box policy. The templates give different tasks a unified input–output interface and keep the supervision of the DETR object-query branch aligned with the MLLM generation branch. Based on the training stage and task granularity, we divide the prompt templates into the following two groups.

Detection and phrase-grounding templates. These templates are used for object detection and referring/phrase-grounding tasks that appear in both training stages and cover four instruction types.

- *Object Detection (OD)* takes a comma-separated list of category names as input, and the model outputs the bounding box and category label of each target object as a structured JSON array.
- *Open-Ended Detection (OED)* provides no candidate category set; the model autonomously discovers and names objects, using the same output format as OD.
- *Grounding from Caption* provides a complete image caption, and the model localizes all entities mentioned in the caption and returns their bounding boxes.
- *Grounded Captioning from Caption* also takes a full caption as input, but asks the model to generate natural-language text while appending inline annotations after each entity phrase, including the category name and bounding-box coordinates.

Each of the four task types contains 15 diverse English instruction variants, and Table 10 lists five examples for each type. During training, the sampled instruction is appended with a unified output-format constraint that specifies the JSON keys and coordinate format; the exact constraints are shown in Table 11. In addition, for detection and grounding tasks with structured outputs, a noise-box instruction can be optionally attached depending on the training or inference phase. During training, the model is allowed to output low-confidence boxes labeled as "noise" to improve robustness to hard negatives; during inference, such outputs are forbidden to avoid invalid predictions.

Region understanding and referring-comprehension templates. These templates serve the region-level perception tasks newly introduced in Stage 2 and also contain four instruction types.

- *Region Caption* specifies a region in the image with bounding-box coordinates, and the model generates a short natural-language description of the region without outputting coordinates.
- *Grounded Image Captioning from Region* uses the same region-coordinate input, but requires the model to annotate every mentioned entity with an inline category name and bounding box while generating the description, so that the regional description contains fine-grained spatial grounding.
- *Referring Comprehension* $\rightarrow$ *Region + Category* provides both a description and several referring expressions to be resolved; for each expression, the model returns the corresponding bounding box and object category as a structured JSON array.
- *Referring Comprehension* $\rightarrow$ *Grounded Caption* uses the same input as the previous task, but changes the output form: instead of a JSON array, the model directly inserts the corresponding bounding-box annotation for each referring expression into the original description, producing a complete sentence with spatial annotations while preserving linguistic coherence.

These four task types also contain 15 instruction variants each; Table 12 lists five examples, and their corresponding output-format constraints are given in Table 13. Overall, the two groups of templates correspond to the two-stage training data described above. The first group covers detection and phrase grounding and is used throughout both stages, while the second group covers region captioning, grounded image captioning from regions, and multi-target referring comprehension, and is dedicated to Stage 2 region-level instruction tuning.

C Optimization and Implementation Details

The following settings specify the optimization details behind the two-stage training pipeline in Section 3.4. Table 14 reports the trainable modules, objectives, batch sizes, learning rates, and epochs for each sub-stage. Stage 1 first trains the DETR branch and then aligns the object-query projector Proj_O. Stage 2 performs curriculum fine-tuning for two epochs: one epoch of full-parameter co-training in Stage-2a, followed by one epoch of LLM-only fine-tuning with the interleaved mask in Stage-2b. We use AdamW [33] with a cosine learning-rate schedule, 1,000 warmup steps, and decay over the final 30% of total training steps; the visual backbone and BERT-Base use peak learning rates set to $0.1\times$ the corresponding main learning rate. All supervised loss weights are set to 1.0.

OD

1. Detect all objects: {}
2. Locate the following objects in the image: {}
3. Find all instances of: {}
4. Identify and locate: {}
5. Detect {}

OED

1. Detect all objects in the image and output their bounding boxes and labels.
2. Find every salient object in the scene. For each one, give a box and a short category name you choose.
3. Perform open-ended object detection: list all objects you see, with bounding boxes and your own labels.
4. Scan the image and detect anything worth localizing. Use natural category names and tight boxes.
5. Identify all distinct objects in the image. You may use any descriptive labels; output bbox and label for each.

Grounding from Caption

1. Given the caption: "{}", locate all mentioned entities in the image and provide their bounding boxes.
2. Caption: "{}". Ground every entity referenced in this caption.
3. Find and localize all objects mentioned in the following caption: "{}".
4. Using the caption "{}", identify and box each entity it describes.
5. The caption says: "{}". Detect all entities mentioned and return their bounding boxes.

Grounded from Caption

1. Given the caption "{}", produce a grounded version with bounding boxes for each entity.
2. Rewrite the caption "{}" as a grounded caption, linking each entity to its region.
3. Convert caption "{}" into a grounded description with bbox annotations.
4. Caption: "{}". Output the same description with each entity grounded to a bounding box.
5. Produce a grounded image caption consistent with: "{}".

Table 10: Instruction templates for detection and phrase grounding in GOD.

_json_schema_hint
Return ONLY valid JSON as an array of objects. Each object must be exactly in this format: {"bbox_2d":[x1,y1,x2,y2],"label":"class_name"}. Do not add extra keys, comments, markdown, or trailing commas.

_grounding_schema_hint
Return ONLY valid JSON as an array of objects, one per entity. Each object must follow exactly: {"bbox_2d":[x1,y1,x2,y2],"label":"entity_name"}. Use the exact entity wording from the caption as the label value. Do not add extra keys, comments, markdown, or trailing commas.

_grounded_captioning_schema_hint
Output the caption as natural language. Immediately after each grounded entity phrase, insert its annotation in this exact format: {"category":"entity_phrase","bbox_2d":[[x1,y1,x2,y2]]}. A single entity may have multiple boxes: {"category":"entity_phrase","bbox_2d":[[x1,y1,x2,y2],[x1,y1,x2,y2]]}. Do not output anything outside the annotated caption string.

_noise_train_hint
For hard negatives or uncertain boxes, you can output the noise box like {"label":"noise", "bbox_2d":[x1,y1,x2,y2]}.

_noise_infer_hint
Do NOT output the noise box like {"label":"noise", "bbox_2d":[x1,y1,x2,y2]}.

Table 11: Output-format and noise-box instructions for detection and phrase grounding.

Region Description

1. Describe the content within the region {}.
2. What is in the region {}? Provide a brief caption.
3. Generate a caption for the region {} in the image.
4. Given the region {}, describe what you see.
5. Provide a short description of the area marked by {}.

Grounded Image Captioning

1. Based on the region {}, generate a grounded caption with category labels and the region boxes.
2. For region {}, describe the content and annotate each entity with its category and bounding box.
3. Produce a grounded caption for region {}: include category names and bbox for every mentioned entity.
4. Region {}: generate a caption and ground each entity with its category and spatial location.
5. Describe region {} and mark each entity with an inline category label and bounding box.

Region REC

1. Given the caption "{}" and the referring expressions [{}], locate each expression in the image and output its bounding box and category.
2. Caption: "{}". For each of the following phrases [{}], find the corresponding region and provide its bbox and category label.
3. Resolve the referring expressions [{}] in the context of caption "{}". Return the bounding box and category for each.
4. Using the caption "{}", ground each of these expressions [{}] to a region with bounding box and category.
5. Referring comprehension: caption is "{}", expressions are [{}]. Output bbox and category for each.

Referring Comprehension → Grounded Caption

1. Given the caption "{}" and the referring expressions [{}], rewrite the caption with each expression grounded to its bounding box.
2. Caption: "{}". Annotate each of [{}] in-place with its bounding box to produce a grounded caption.
3. Produce a grounded version of caption "{}" where each expression in [{}] is tagged with its region.
4. Rewrite "{}" as a grounded caption: for each phrase in [{}], insert its bounding box annotation inline.
5. Grounded referring: given caption "{}" and expressions [{}], output the caption with bbox annotations on each expression.

Table 12: Instruction templates for region and referring comprehension.

_region_caption_schema_hint
Output a brief natural-language caption describing the content of the specified region. Do not include bounding box coordinates in your output.

_grounded_captioning_from_region_schema_hint
Output the caption as natural language. Immediately after each entity phrase, insert its annotation in this exact format: {"category":"entity_name","bbox_2d":[[x1,y1,x2,y2]]}. Do not output anything outside the annotated caption string.

_referring_region_schema_hint
Return ONLY valid JSON as an array of objects, one per referring expression. Each object must follow exactly: {"expression":"phrase","category":"object_category","bbox_2d":[x1,y1,x2,y2]}. Do not add extra keys, comments, markdown, or trailing commas.

_referring_grounded_schema_hint
Output the caption as natural language. Immediately after each resolved referring expression, insert its annotation in this exact format: {"expression":"phrase","bbox_2d":[[x1,y1,x2,y2]]}. Do not output anything outside the annotated caption string.

Table 13: Output-format instructions for region and referring comprehension in GOD.

Setting	Stage 1		Stage 2	
	(a)	(b)	(a)	(b)
Trainable modules	DETR	Proj_O	MLLM + DETR + Proj_O	LLM
Objective	$\mathcal{L}^{\text{ovd}}_{\text{DETR}}$	$\mathcal{L}^{\text{text}}_{\text{MLLM}} + \mathcal{L}^{\text{ovd}}_{\text{MLLM}}$	$\mathcal{L}^{\text{text}}_{\text{MLLM}} + \mathcal{L}^{\text{ovd}}_{\text{MLLM}} + \mathcal{L}^{\text{ovd}}_{\text{DETR}}$	$\mathcal{L}^{\text{text}}_{\text{MLLM}} + \mathcal{L}^{\text{ovd}}_{\text{MLLM}}$
Batch size	256	256	128	128
Learning rate	$2e^{-4}$	$1e^{-3}$	$2e^{-5}$	$2e^{-5}$
Epochs	12	1	1	1

Table 14: Training configuration for each stage of GOD.

Training images are capped at 768×768, with at most 45 objects per image. For the DETR branch, we use 900 object queries; for efficiency, only the top 300 queries are used for object-query reconstruction, while the remaining queries are kept for DETR-style objectness prediction. End-to-end training is implemented with DeepSpeed, ZeRO-1/2 sharding, mixed precision, and FlashAttention-2. Under this configuration, Stage 1 and Stage 2 take approximately 36 and 48 hours, respectively, on 8 NVIDIA A100 GPUs.

D Additional Experiments

We first define the experimental output notation, then present matched comparisons and analyses of MLLM remodeling, box quality, and inference overhead.

D.1 Experimental Output Notation

This notation is used in the main comparisons in Section 4.2. GOD contains both a discriminative DETR branch and a generative MLLM branch, and the generative branch can be evaluated either with or without reference object queries. We therefore use the following notation to distinguish output sources and inference modes in the experiments.

- **GOD-DETR-OVD** denotes the discriminative detection output of the DETR branch. The object queries produced by the DETR proposal network are directly aligned with text category embeddings through region–text matching, yielding bounding-box and category predictions.
- **GOD-MLLM-OVD** denotes the discriminative detection output after MLLM-side remodeling. The object queries are projected into the MLLM through Proj_O and remodeled in the sequence space; the resulting query embeddings are then used for region–text alignment. The comparison evaluates whether the MLLM refines object-level representations in a way that remains useful for discriminative detection.
- **GOD-MLLM-Direct** denotes the generative detection output of the MLLM in reference-free mode. During inference, the mask fully blocks the object queries $\boldsymbol{O}$ from the answer sequence, so the MLLM autoregressively generates structured detection outputs using only the visual tokens $\boldsymbol{V}$. It measures the model's standalone generative detection ability without explicit detector priors.
- **GOD-MLLM-Ref** denotes the generative detection output of the MLLM in reference-assisted mode. During inference, the object queries $\boldsymbol{O}$ are fully visible to the answer sequence, allowing the MLLM to generate detection results with object-level priors from the DETR branch. It measures generative detection under detector-anchored guidance.

The first two notations refer to discriminative outputs based on region–text matching; they differ in whether the query embeddings come directly from the DETR branch or are remodeled by the MLLM. The last two notations refer to autoregressive generative outputs; they differ in whether object queries are visible to the generation branch at inference time. Unless otherwise specified, we use GOD-MLLM-Ref as the default generative inference setting. Stage or model-size suffixes, such as Stage1, Stage2, or 3B, indicate the corresponding training checkpoint or model scale when needed.

All third-party datasets, pretrained models, and code components used in GOD are credited through their original references and are used only under their official licenses and terms of use; we do not redistribute any third-party asset beyond those terms.

D.2 Evaluation under the Rex-Omni Protocol

What differs across settings is only which category list enters the prompt.

- Restricted (image) category: only the categories that occur in the image (the positive labels) are listed.
- General category: the full dataset category list is listed, so it necessarily includes categories absent from the image, i.e. negatives that invite hallucination.

Rex-Omni also differs in scoring, not only in prompting. It does not report mAP; it proposes its own mIoU-based metric (R@mIoU / P@mIoU / F1@mIoU), and we re-evaluated GOD under that metric for a like-for-like comparison. Its fast variant removes all negative-category predictions before scoring, so it reflects only the positive classes. We instead retain negatives, so hallucinated categories and low-quality detections are penalized rather than filtered out; we report this stricter setting by default.

Index generation and discriminative merging. After the proposal-based results are produced, we weighted merge the generated boxes that related proposal via GIoU and use that proposal to refine the probability of generated category. This combination reaches 43.7, on par with VLM-FO1's 44.4; and it attains this comparable result *without a dual-tower backbone*, using only Qwen2.5-VL-3B, whereas VLM-FO1 additionally relies on a separate DaViT-Large vision tower.

Method	Paradigm	Backbone	AP
VLM-FO1	Index Gen.	Qwen2.5-VL-3B	44.4
GoD-mllm-ref-merged	Coord. Gen. + Disc. Merge	Qwen2.5-VL-3B	<u>43.7</u>

Table 15: Comparison with VLM-FO1 on COCO.

IoU-threshold metrics

Model	P@0.50	R@0.50	F1@0.50	P@0.95	R@0.95	F1@0.95
Rex-Omni	75.51	**68.07**	71.60	**17.64**	**14.74**	**16.06**
GoD-mllm_direct	77.80	67.20	**72.11**	17.33	13.72	15.32
GoD-mllm_ref	**78.11**	66.87	72.05	17.59	13.90	15.53

Aggregated metrics

Model	P@mIoU	R@mIoU	F1@mIoU	mAP
Rex-Omni	55.91	49.80	52.68	30.8
GoD-mllm_direct	60.17	**51.49**	55.50	38.1
GoD-mllm_ref	**60.59**	51.23	**55.52**	**38.56**

Table 16: Detection results under the Rex-Omni protocol on COCO.

D.3 MLLM Remodeling and Box Quality

The ordering in fact *reverses* over training: in Stage-1 and Stage-2a the MLLM discriminative output is actually below the DETR head, and the crossover only emerges in Stage-2b.

During Stage-2b, where only the LLM is fine-tuned while the vision encoder and the DETR branch stay frozen, we observe MLLM-OVD first rising and then declining, while the generative output keeps improving throughout. We read this as follows: because only the LLM is updated, the model first learns, over an intermediate range, to refine the frozen proposals, which lifts MLLM-OVD

above DETR-OVD; as the text loss increasingly dominates, the model then reallocates capacity toward generation, so MLLM-OVD eases off while MLLM-text continues to grow.

To probe the two outputs directly, we compared MLLM-text against MLLM-OVD on their overlapping boxes (same category, GIoU > 0.5). Counter to the heuristic that generated coordinates should be coarser and therefore worse, the generative boxes align better with the ground truth and win more often (Table 17).

Overlap Same Category	Matched pairs	Win Rate Text / OVD	Mean GT GIoU Text / OVD
GIoU > 0.30	32,927	51.9 / 48.1	0.623 / 0.616
GIoU > 0.50	31,581	51.9 / 48.1	0.645 / 0.637
GIoU > 0.70	28,047	51.8 / 48.2	0.691 / 0.684

Table 17: Generative and discriminative box quality on COCO.

We further analyzed where the two outputs differ on COCO. The generative branch's lower headline score stems mainly from a small number of missed objects and a few hallucinations: it emits only ~ 37.8 K boxes in total, of which just 6.2 K have no discriminative counterpart. The discriminative branch instead emits ~ 1.48 M boxes, and the ~ 1.45 M that the generative branch does not produce are almost all very-low-probability detections; precisely the low-confidence boxes that generation omits.

D.4 Inference Overhead

Table 18 provides the detailed profiling results for the inference modes discussed in Section 4.4.

Metric	MLLM-direct Number queries $= 0$	MLLM-Ref Number queries $= 900$	Change
Parameters (B)	3.755	3.898	$+3.8\%$
Prefill FLOPs (T)	138.8	159.9	$+15.2\%$
Prefill peak memory (MiB)	31,664	37,366	$+18.0\%$
Decode peak increment (MiB)	104.8	128.6	$+22.7\%$
Input tokens	1,333	1,635	$+302$ tokens
End-to-end generation (s)	7.68	8.66	$+12.7\%$

Table 18: Inference overhead of the DETR branch.

E Denoising and Noise-Augmentation Ablation

Relation to DN-DETR and DINO. DN-DETR [21] introduces denoising to accelerate the optimization of positive queries under bipartite matching. In this discriminative setting, denoising groups are isolated from one another, and matched queries are prevented from accessing the noisy ground-truth context to avoid information leakage. GOD builds on this denoising lineage, including the contrastive denoising of DINO [52], but additionally uses noisy object-level context to support autoregressive generation. High-quality positive proposal queries provide references that reduce interference from inaccurate proposals early in training, while negative noise improves the robustness of generated outputs.

The attention strategy is adapted accordingly. GOD's *interleaved attention mask* allows answer tokens to attend across query groups, jointly strengthening their use of positive references and their robustness to negative context. Interactions between denoising groups follow the causal sequence ordering in the MLLM, as illustrated in Figure 3. This cross-group context for generation is distinct from group-isolated discriminative denoising; it does not require matched queries to depend on CDN queries, which remain masked in the preferred proposal-assisted configuration.

Denoising and complementary objectives. GOD uses contrastive denoising (CDN) to supply high-quality positive proposal queries as references for generation, reducing interference from inaccurate queries early in training, while additional *negative* noise improves output robustness. During co-training, $\mathcal{L}_{\mathrm{DETR}}^{\mathrm{ovd}}$ shapes the DETR decoder and objectness head, $\mathcal{L}_{\mathrm{MLLM}}^{\mathrm{text}}$ drives autoregressive generation, and $\mathcal{L}_{\mathrm{MLLM}}^{\mathrm{ovd}}$ aligns the projected object queries in the MLLM sequence space. The detection and generation objectives therefore have complementary roles, with projected object queries linking localization supervision to the generative context. The *interleaved attention mask* lets answer tokens attend across query groups, allowing generation to benefit from positive references and additional negative context.

Noise masks and training–inference consistency. To complement the denoising design in Section 3.2 and the ablations in Section 4.3, we examine how this noisy context should be exposed to answer tokens and matched target queries. In discriminative DETR-style training, CDN noise queries and matched queries can be separated by a bidirectional attention mask to prevent information leakage. In a generative framework, the causal attention mask introduces a further constraint: during training, both CDN queries and Hungarian-matched target queries precede the answer sequence, whereas CDN queries are removed at inference time. If matched queries depend on this training-only noise context, the MLLM receives a different context structure at inference time. The ablation below therefore distinguishes the benefit of exposing answer tokens to CDN queries from the risk of allowing matched queries to depend on them.

Noise-augmentation ablation. Table 5 isolates the masking relation between CDN noise queries and target queries during co-training. The columns Ans→CDN and Match→CDN indicate whether the answer sequence and the matched target queries can attend to CDN noise queries, respectively. Row 1 uses only matched queries without CDN, giving $17.5/18.9$ proposal-free/proposal-assisted generative accuracy and discriminative scores of $50.3/49.3$ for $\boldsymbol{O}_{\mathrm{DETR}}$ and $\boldsymbol{O}_{\mathrm{MLLM}}$. When CDN queries are added but both Ans→CDN and Match→CDN are masked (Row 2), the CDN queries participate only in contrastive denoising inside the DETR decoder and pass no information to either the generation branch or the matched queries. Both proposal-free and proposal-assisted generation improve to $21.2/22.5$, suggesting that even masked CDN queries can improve the shared visual space through $\mathcal{L}_{\mathrm{DETR}}^{\mathrm{ovd}}$.

Allowing both the answer sequence and matched queries to attend to CDN queries (Row 3) raises proposal-assisted generation to 24.0, but proposal-free generation collapses to 6.1 and $\boldsymbol{O}_{\mathrm{MLLM}}$ drops from 50.4 to 45.0. This pattern points to information leakage: matched-query representations become dependent on noise context that is absent at inference time, weakening the remodeling effect of $\mathcal{L}_{\mathrm{MLLM}}^{\mathrm{ovd}}$. When generation also relies on this joint context, the loss of noise-dependent target-query information further increases the training–inference gap.

For proposal-assisted generation, the preferred configuration is Row 4, where the answer sequence can attend to CDN queries but matched queries cannot. Row 4 gives the strongest proposal-assisted result and balanced discriminative scores, with $16.7/32.4$ proposal-free/proposal-assisted accuracy and $50.2/49.0$ discriminative scores; its proposal-free score remains below Row 2. This configuration separates the two roles of CDN: matched queries remain independent of training-only noise context, while the answer sequence receives additional noisy object-level context during training. Together, these results support complementary uses of denoising: improving the shared visual space through detection supervision and augmenting the context available to generation. The latter requires preventing matched queries from depending on CDN queries to preserve training–inference consistency.

F Analysis

Relation to existing open-ended detection paradigms. The discussion here expands the methodological analysis in Section 3 and the ablation findings in Section 4.3. Existing MLLM-based open-ended detectors can be broadly grouped into direct generation, two-stage retrieval, and label- or tool-driven approaches. Direct generation is architecturally simple, but it asks the language model to handle category generation and coordinate regression at the same time, making it strongly constrained by token-level prediction. Two-stage retrieval first produces candidate regions and then lets the language model select or reason over them, which improves localization accuracy but usually decouples detection and understanding into two separate modules. Label- or tool-driven methods

rely on expert modules to improve task performance, but their training and inference pipelines are often more complex. GOD occupies a middle ground between these paradigms: it does not force the MLLM to recover spatial structure only from visual tokens, nor does it fully separate the proposal network from the language model. Instead, GOD introduces a lightweight object-query branch on top of the shared visual encoder and uses co-training to make object-level representations part of the generative context. The resulting model preserves the flexibility of open-ended interaction while introducing detector-like spatial inductive bias, linking open-ended detection with grounded generation in a single training process.

Co-training mechanism. Optimization-wise, the key to GOD is not simply attaching a detector in front of a language model, but jointly constraining the shared visual space with three losses. $\mathcal{L}_{\mathrm{DETR}}^{\mathrm{ovd}}$ encourages object queries to correspond to high-quality foreground proposals; $\mathcal{L}_{\mathrm{MLLM}}^{\mathrm{ovd}}$ remodels these query embeddings into object-level semantic tokens that are easier for the language model to use; and $\mathcal{L}_{\mathrm{MLLM}}^{\mathrm{text}}$ preserves natural-language generation. Removing any one of these components either weakens the spatial prior or prevents the generative branch from effectively using that prior. The curriculum schedule is designed to reduce a training–inference gap in generative detection. If answer tokens always attend to object queries during training, the model can become overly dependent on explicit proposals; if object queries are hidden from the beginning, optimization becomes much harder. The interleaved attention mask and staged training address this by first letting the model learn from detector-provided priors and then gradually reducing its dependence on them. The ablation results show that GOD-Full not only achieves the best generative accuracy, but also keeps $\boldsymbol{O}_{\mathrm{MLLM}}$ close to, and in some settings even above, $\boldsymbol{O}_{\mathrm{DETR}}$, suggesting that $\mathcal{L}_{\mathrm{MLLM}}^{\mathrm{ovd}}$ acts both as an implicit regularizer and as a semantic refinement signal for object queries. Thus, co-training is not only the starting point of GOD's design, but also a mechanism that maintains consistency between detection and generation throughout training.

G Limitations

Limitations of the generative paradigm. The following points provide additional context for the evaluation setting in Section 4.2. Although GOD injects object-level visual priors into the generation process through a DETR-style query branch, its output side remains in the token space. Bounding-box coordinates are discretized into numeric tokens and optimized token by token with cross-entropy, together with category names. As a result, coordinate prediction is still modeled as a classification problem rather than a regression problem, and token-level supervision cannot directly measure the spatial quality of a predicted box. Index-based retrieval methods face a related limitation: although they map bounding boxes into an index space, their final outputs are still token-level predictions and therefore do not remove the structural gap between token-level optimization and visual-level optimization. A possible direction is to introduce a dual-mode output mechanism into the autoregressive sequence. When spatial information is required, a special token could switch the model from token prediction to continuous feature output, e.g., by directly generating an object-query vector or a box-regression feature, followed by a lightweight prediction head for coordinate regression. When language information such as category names is required, the model would still use standard token prediction and cross-entropy supervision. Such a hybrid output design would convert part of the spatial prediction problem from classification back to regression, allowing token-level language supervision and object-level visual supervision to coexist in the same generative framework.

Limitations of evaluation. Open-ended generative detection also faces fundamental evaluation challenges. Unlike closed-set or open-vocabulary detection, an open-ended model outputs category names in free-form natural language, which cannot be directly matched one-to-one with a predefined label set. Mainstream benchmarks such as COCO and LVIS are built around fixed category taxonomies, and therefore cannot fully measure whether a model can autonomously name and localize objects in genuinely open-ended scenes. Heuristic post-processing, such as semantic-similarity matching or canonical label mapping, can partly reduce the mismatch between free-form names and standard labels, but the resulting scores still depend on similarity thresholds, matching strategies, and even the choice of judging model. This limits the objectivity and reproducibility of current evaluations. Standardized protocols for open-ended generative detection, including unified category-alignment rules, multidimensional metrics, and low-subjectivity automatic judging mechanisms, remain necessary for future progress.

Model size. The object queries enter the LLM through the projector Proj_O, which maps them into whatever hidden size the backbone uses, so the DETR branch is *architecture-agnostic* rather than tuned to 3B.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: A summary of the paper's contributions is provided at the end of the introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in the conclusion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The development of our method is theoretical driven, which bridges the training objective of generative model to our proposed loss function.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the implementation details of our method in the main results section, and more detailed dataset-specific training setups in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We will release our code and trainingdata after the paper is accepted.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide training details such as dataset, pretrained models, hyperparameters in Appendix A and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Because it would be computationally expensive for us to run all baselines and our methods multiple times.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the compute resources in the main results section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: We conduct foundational research based on the existing open-source datasets and models, which are publicly available for research use.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The safeguards of GOD's based model Qwen2.5-VL and Rex-omni are considered by their authors, our data collection and processing protocols are transparent.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite all original papers for the models and datasets used in this paper in Appendix A. Their licenses and terms of use are also mentioned in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: We have not yet released our assets. We will release our code and models, together with usage instruction files, once the paper is accepted.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components. Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.