Octopus: A Multi-modal LLM with Parallel Recognition and Sequential Understanding

Chuyang Zhao[1*] **Yuxin Song**[1*] **Junru Chen**[1] **Kang Rong**[1] **Haocheng Feng**[1]
Gang Zhang[1] **Shufan Ji**[2] **Jingdong Wang**[1] **Errui Ding**[1] **Yifan Sun**[1✉]

[1]Baidu VIS [2] Beihang University
[*] Equal Contribution ✉ Corresponding Author
{zhaochuyang,songyuxin02,sunyifan01}@baidu.com

Abstract

A mainstream of Multi-modal Large Language Models (MLLMs) have two essential functions, *i.e.*, visual recognition (*e.g.*, grounding) and understanding (*e.g.*, visual question answering). Presently, all these MLLMs integrate visual recognition and understanding in a same sequential manner in the LLM head, *i.e.*, generating the response token-by-token for both recognition and understanding. We think unifying them in the same sequential manner is not optimal for two reasons: 1) parallel recognition is more efficient than sequential recognition and is actually prevailing in deep visual recognition, and 2) the recognition results can be integrated to help high-level cognition (while the current manner does not). Such motivated, this paper proposes a novel "parallel recognition $\rightarrow$ sequential understanding" framework for MLLMs. The bottom LLM layers are utilized for parallel recognition and the recognition results are relayed into the top LLM layers for sequential understanding. Specifically, parallel recognition in the bottom LLM layers is implemented via object queries, a popular mechanism in DEtection TRansformer, which we find to harmonize well with the LLM layers. Empirical studies show our MLLM named Octopus improves accuracy on popular MLLM tasks and is up to $5\times$ faster on visual grounding tasks.

1 Introduction

Visual recognition and understanding are two essential abilities for Multi-modal Large Language Models (MLLMs). While earlier MLLMs [1, 2, 3, 4] focused on the high-level visual understanding ability (*e.g.*, visual question answering), recent literature finds that visual recognition ability (*i.e.*, identifying and locating the objects) are no less important. The importance lies in two aspects: 1) Many newly-merged MLLM usages are directly related to visual recognition, *e.g.*, visual grounding [5, 6] and referential dialog [6]. 2) More generally, visual recognition is potential to benefit all understanding tasks as recognition results are important compositions for understanding. In this paper, we are interested in better harmonizing these two essentials for MLLM.

Presently, MLLMs unify visual recognition and understanding in a sequential paradigm. In this paper, the terms "sequential" and "parallel" refer specifically to the inference of LLM head, rather than the visual encoder. Typically, an MLLM consists of a visual encoder and an LLM head. During inference, the LLM head sequentially generates the response token-by-token, regardless of whether the task is more aligned with visual recognition (*e.g.*, grounding) or understanding (*e.g.*, visual question answering), as in Fig. 1 (left). Sequentially referring the recognition results, particularly textualized coordinates, is relatively slow. This sequential manner is a legacy of the LLM structure and, more fundamentally, stems from the inherently sequential nature of language.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

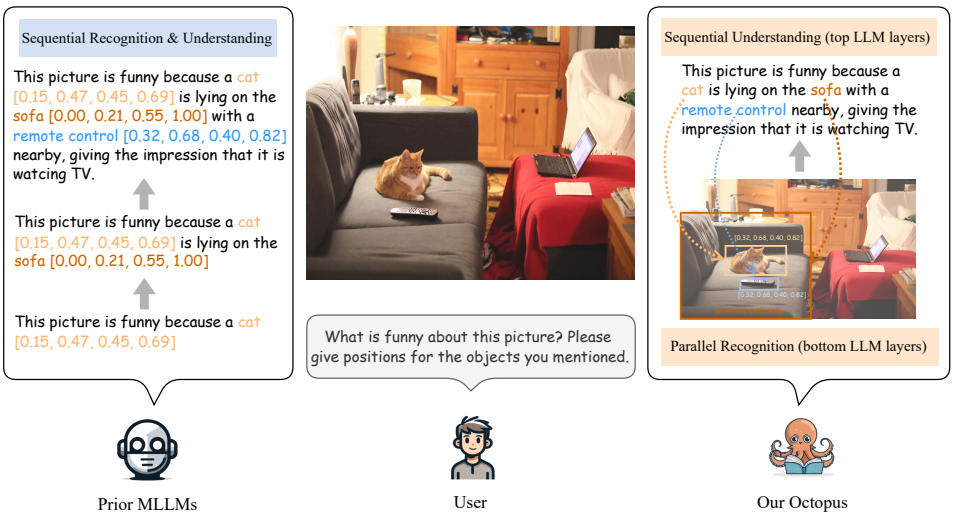

Figure 1: Comparison between prior MLLMs (left) and our Octopus (right). Left: Prior MLLMs typically adopt the purely sequential inference: the LLM head infers the response token-by-token, regardless whether the response token is more aligned with recognition (*e.g.*, detection) or understanding. Sequentially inferring the position is slow. Right: In contrast, Octopus establishes a "parallel recognition $\rightarrow$ sequential understanding" framework. The bottom LLM layers first recognize potential objects via visual grounding or referring segmentation (in Appendix:B). The recognition results (coupled with visual tokens) are relayed into top LLM layers. The top LLM layers thus do NOT infer the object position but instead, they select boxes (or masks) that have already been detected. The entire Octopus LLM head (recognition + understanding) is trained end-to-end.

We conjecture the purely sequential paradigm might not be an optimistic framework for MLLM, especially regarding visual recognition and its cooperation with understanding. There are two reasons:

• First, for both human and deep learning, visual recognition relies heavily on parallel processing for high efficiency. Before the MLLMs era, most deep recognition models are built on the parallel paradigm. For instance, the segmentation model infers the semantic for all pixels simultaneously, and the detection model detects all the objects using parallel anchors [7, 8] or queries [9, 10, 11, 12]. Humans also use parallel processing for simple recognition [13, 14, 15]. In contrast, current purely sequential paradigm lacks efficiency for visual recognition.

• Second, regarding the cooperation between visual recognition and understanding, there is a hierarchy of "parallel recognition $\rightarrow$ sequential understanding", as revealed by psychology and neurobiology discoveries [14, 15]. Relatively easier and low-level recognition results are integrated via more complex mental operations to help high-level cognition [13]. This hierarchy allows the understanding to take advantage of the recognition results, while the purely sequential paradigm does not offer such a benefit.

This paper proposes the Octopus (the octopus animal has a central brain and multiple parallel "auxiliary brains") framework to improve the efficiency of recognition, and to harness the benefit of the aforementioned cognition hierarchy. Octopus separates visual recognition and understanding into parallel and sequential processes, respectively, and then re-integrates them in a "parallel recognition $\rightarrow$ sequential understanding" hierarchy. The comparison between purely sequential MLLMs and Octopus is illustrated in Fig. 1.

Given visual tokens from the visual backbone, Octopus's LLM head uses the bottom layers to detect the potential objects in parallel. The detection results, coupled with the visual tokens, are fed into the sub-sequential LLM layers for further understanding. Though the understanding remains sequentially token-by-token, it deviates from previous MLLMs by eliminating the need to infer the position of objects. Instead, it selects previously-detected boxes and associates them with the objects, markedly improving efficiency. For example, on Flickr30k dataset (average 4 objects per image), Octopus reduces the time of recognizing all objects to about 21% (3.80s to 0.82s per image, $5\times$ increase in speed). Moreover, we empirically find that Octopus improves the accuracy on a range of understanding tasks compared to its purely sequential counterpart. It indicates that the initial

parallel recognition can effectively support the understanding, revealing a clear advantage of the brained-inspired cognitive hierarchy.

Another significant feature of Octopus is: it can automatically adjust its recognition modes, oscillating between class-agnostic and class-specific, based on user instructions. This flexibility provides versatile usages for various tasks. For instance, in a grounding task where users specify the particular interest ,*e.g.*, a cat, the recognition part of Octopus becomes class-specific and and predicts multiple candidates for the object of interest. The understanding part then selects the best candidate for the final response. In contrast, in another scenario, where the users do not specify any particular interest and request a detailed enumeration (including the position), the recognition part automatically shifts into a class-agnostic detector. Correspondingly, the understanding part then assigns semantics to the detection results. This flexibility originates from the knowledge in the bottom LLM layers designated for recognition.

Our main contributions are as summarized as follows:

• We investigate the cooperation between recognition and understanding in current MLLMs. As a result, we identify an efficiency issue with the purely sequential paradigm, as well as a significant discrepancy from human cognitive processes.

• We propose the Octopus framework for MLLM. Octopus separates recognition and understanding into parallel and sequential processes, respectively, and re-integrates them into a "parallel recognition $\rightarrow$ sequential understanding" hierarchy.

• Extensive experiments show Octopus improves inference efficiency and enhances the accuracy, compared to the purely sequential counterpart.

2 Related Works

Multi-modal Large Language Model. The recent success of large language models (LLMs) has spurred research into integrating LLMs with computer vision for visual understanding. Flamingo [16] adds trainable cross-attention layers to each LLM decoder layer to learn visual information. BLIP-2 [3] introduces the Q-Former to align visual and language spaces. Mini-GPT4 [2] and mPLUG-OWL [17] also use the Q-Former for visual understanding. LLaVA [18] connects the pretrained CLIP [19] visual encoder to the LLM with a simple vision-language connector, achieving strong performance. These efforts demonstrate the potential of Multi-modal Large Language Models (MLLMs) for complex multi-modal tasks.

Using MLLMs for Visual Recognition. Inspired by that LLM have unified various NLP tasks into a generation problem in one architecture, recent MLLM works manifest to solve traditional visual recognition tasks in a unified MLLM architecture. Object detection, a key visual recognition task, poses a challenge in expressing positional information in language within MLLM frameworks. Some literature [6, 20, 21, 2] convert the bounding box into natural language format and directly generate them in text response. However, representing bounding boxes in text form may not be optimal since the bounding box coordinates are numerical data and are typically predicted by regression. Some works [22, 23] use output embeddings of LLM as the understanding pivot to call an object detector for detection. However, the LLM can not benefit from the object detection results from the detector to enhance its own understanding. A more challenging scenario is the segmentation task, in which the ground-truth can be in random shape and seems indescribable via language. To tackle this problem, VisionLLM [24] represents the segmentation mask in text format by representing the mask by the coordinates of the mask polygon. Sphinx [21] and LLaVA-Plus [25] utilize an offline model SAM [26] for segmentation. They first generate the bounding box of the object to segment using MLLM, then they prompt SAM to generate the segmentation mask. LiSA [27] integrates SAM into MLLM for joint training. They introduce a special "<SEG>" token to predict the segmentation mask. However, a single "<SEG>" token cannot differentiate between multiple instances, limiting it to outputting only one segmentation mask.

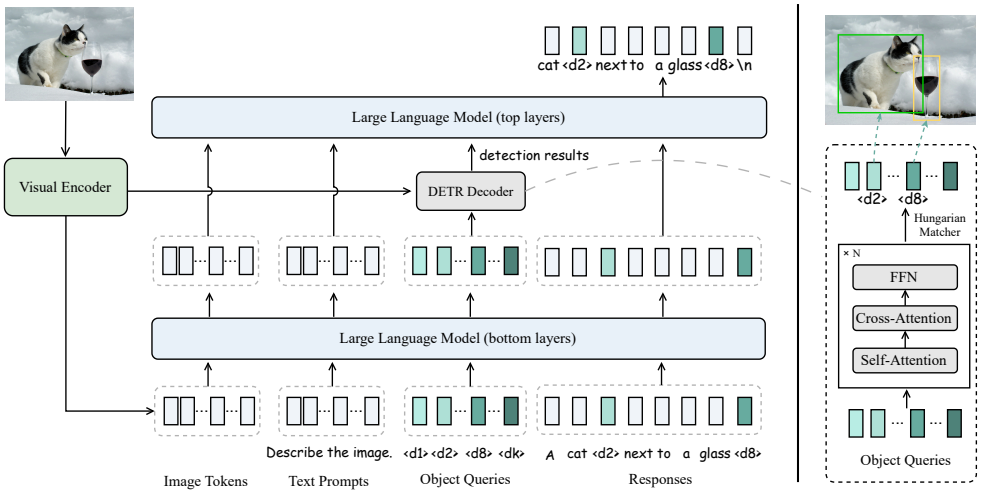

Figure 2: The overall training process of Octopus. We omit the visual backbone to highlight the LLM head. Octopus inputs multiple object queries into the LLM head, in addition to the image tokens and text prompt tokens,. After passing through several bottom LLM layers, the object queries are fed into a DETR decoder for parallel recognition. The recognition is not the canonical close-set detection, but can be class-agnostic detection, visual grounding, referring segmentation, *etc.*, depending on the user prompt (as shown in Fig.3). Afterwards, these object queries, coupled with the image token and text tokens, are then sent into the upper LLM layers for sequential understanding. When the users ask for spotting the mentioned objects (*e.g.*, visual grounding), Octopus finds the object query that detects each object (*e.g.*, the 2nd object query detects the cat) and points out this object query via a corresponding index token "<d2>". The structure of Octopus and its training details are elaborated in Section 3.2 and Section 3.3.

3 Approach

3.1 Preliminaries

Sequential inference in prior MLLMs. Prior MLLMs [1, 2, 21] solve traditional visual recognition tasks (e.g., object detection, instance segmentation) in a sequential generation paradigm. Specifically, they represent the outputs of visual recognition in a natural language format. For example, the bounding boxes are denoted by the coordinates of their corners: "$[x_{min}, y_{min}, x_{max}, y_{max}]$", and segmentation masks are represented by coordinates of points in polygon mask: "$[x_1, y_1, x_2, y_2, \dots]$". Hereby, each numerical value is expressed as multiple text tokens. Sequentially generating all these position tokens can be quite time-consuming, *e.g.*, 25 tokens for a bounding box "[0.152,0.475,0.451,0.692]". Moreover, generating each token at its core, is based on classification. In contrast, coordinates are inherently numerical and are typically predicted through regression. This discrepancy suggests that sequential generation may not be the optimal approach for visual recognition.

A revisit to DEtection TRansformer (DETR). Octopus uses a light-weight DETR decoder to cooperate with the bottom LLM layers for parallel visual recognition, as illustrated in Fig. 2. We first give a brief revisit to DETR below.

DETR [9] is an end-to-end object detection approach based on transformer. A DETR model consists of a visual backbone, an encoder, and a transformer decoder. The backbone and the encoder transforms an input image into image feature $\mathbf{F}$. Afterwards, the DETR decoder employs a set of parallel object queries $\mathbf{Q} = \{q_1, q_2, \dots, q_k\}$ to absorb image features through stacked cross-attention layers, which is formulated as:

$$\bar{\mathbf{Q}} = \texttt{Decoder}(\mathbf{F}, \mathbf{Q}), \tag{1}$$

in which $\bar{\mathbf{Q}}$ is the output state of object queries. Finally, the DETR decoder append class and box predictors upon the object queries to predict their category and bounding box, respectively:

$$\mathbf{B} = \texttt{box}(\bar{\mathbf{Q}}), \quad \mathbf{S} = \texttt{cls}(\bar{\mathbf{Q}}), \tag{2}$$

(a) (b) (c)

Figure 3: **Visualization the Results.** We visualize the detection results and the LLM output from Octopus on three tasks: (a) Spotting Caption, (b) Referring Expression Comprehension (REC), and (c) Visual Dialogue. In the Spotting Caption task, DETR identifies all foreground objects (highlighted in blue), while in REC, DETR locates only the object-of-interest (highlighted in green) as dictated by the prompts. DETR seamlessly transitions between these two modes based on user input. In the case of Visual Dialogue, since the users do NOT ask the MLLM to output object positions, the LLM output does not contain box information, correspondingly. However, Octopus can still localize the objects from the intermediate recognition results (we visualize the detection results with a classification score greater than 0.5).

in which $\mathbf{B}$ is the bounding boxes and $\mathbf{S}$ is the classification scores. In Octopus we employ DETR as a class-agnostic detector, $\mathbf{S} \in \mathbb{R}^1$ denotes whether an object is foreground or background. Consequently, DETR is independent of the number of classes in the training dataset.

We choose DETR decoder to implement visual recognition for the following reasons: 1) The object queries allow for parallel recognition. 2) The versatility of DETR offers the potential to handle more visual recognition tasks. In addition to object detection, DETR can be adapted for many other recognition tasks, *e.g.*, segmentation [28, 29, 30], pose estimation [31, 32, 33], and object tracking [34, 35, 36]. 3) The object queries can absorb user prompts through attention layers. This allows Octopus to recognize random objects described in natural language, as will be detailed in the following section.

3.2 The structure of Octopus

The overall architecture of Octopus is shown in Fig. 2. The image features $\mathbf{F}$ extracted the visual encoder is projected into image tokens $\mathbf{V} = \{v_1, v_2, \ldots, v_m\}$. The text prompts are tokenized and encoded into text tokens $\mathbf{T} = \{t_1, t_2, \ldots, t_n\}$. Based on this standard MLLM structure, Octopus additionally employs k object queries $\mathbf{Q} = \{q_1, q_2, \ldots, q_k\}$. Each $q_i \in \mathbb{R}^D$ has the same dimension D as the image and text tokens. The object queries are placed after image tokens and prompt text tokens. Consequently, $\mathbf{V}$,$\mathbf{T}$ and $\mathbf{Q}$ are concatenated and jointly fed into the LLM head, which facilitates *parallel recognition* and then *sequential understanding* as below.

Parallel recognition. In the bottom LLM layers, the object queries $\mathbf{Q}$ interact with the image tokens $\mathbf{V}$ and text tokens $\mathbf{T}$ through attention. This interaction aligns the object queries with the hidden states of user prompts and image tokens. In benefit of the language knowledge that is embedded in the LLM layers, the object queries thus absorb information from the user prompts and get an understanding of the user interest. It makes the recognition adaptive to user prompts and is crucial for visual grounding tasks, *e.g.*, referring expression comprehension (REC). Afterwards, the object queries are fed into a lightweight (4-layers) DETR decoder to detect the objects. Both the $\mathbf{Q}$-$\mathbf{V}$-$\mathbf{T}$ interaction and the object detection DETR decoder are in parallel, yielding the complete recognition for Octopus.

Discussion: It's worth noting that the key difference between the integrated DETR decoder in Octopus and traditional DETR trained on closed-set is that in Octopus the DETR decoder makes predictions based on user prompts. The predictions made by Octopus DETR can be class-agnostic detection, visual grounding, referring segmentation, *etc.*, depending on the user prompt. Fig. 3 shows DETR predicts the objects-of-interest depending on user prompts. For example, *e.g.*, it tries to locate all objects in the image, given the user prompt "Please detect all objects in the image". Given another

user prompt "Please spot the black-and-white cat in the image", DETR focuses on identifying the specified cat rather than other objects.

Sequential understanding. The recognition results, *i.e.*, the object queries output from the DETR decoder, coupled with the hidden states of $\mathbf{V}$ and $\mathbf{T}$, are relayed into the following LLM layers. These recognition results will be absorbed into the final output tokens which form the model response, *e.g.*, image captions or visual question answers, in a sequential manner.

In visual grounding tasks, the model response is expected to contain bounding boxes for the mentioned objects. In the previous MLLMs, the bounding box is represented in natural language form, which takes multiple tokens and is time-consuming to generate (Section 3.1). In contrast, Octopus does not generate the text of the bounding box in a sequential manner, but instead, it selects the detection boxes predicted by the DETR. For example, in Fig. 2, the 2nd object query from the DETR decoder detects the cat. In the LLM output, Octopus generates a special token "<d2>" which indexes the 2nd object query after the "cat" token. We name the token that indexes an object query as **index token**. Consequently, Octopus becomes aware of the position of the cat by selecting the bounding box corresponding to the predicted index token. How Octopus learns to predict the index tokens is elaborated in the following Section 3.3.

3.3 Training Octopus

Training Octopus involves supervision of two components: the DETR output and the LLM output. These components are trained jointly, meaning that both the DETR decoder and the LLM are optimized together.

Supervision on the DETR output. Given the detection predictions $\mathbf{Y} = \{(s_1, \mathbf{b}_1), \ldots, (s_k, \mathbf{b}_k)\}$ and the ground-truth objects $\bar{\mathbf{Y}} = \{(1, \bar{\mathbf{b}}_1), \ldots, (1, \bar{\mathbf{b}}_N)\}$, DETR uses the Hungarian algorithm to find the optimal assignment $\sigma(\cdot)$, where each ground-truth object is assigned to its best-matched prediction. Here, $(s_i, \mathbf{b}_i)$ indicates the predicted classification scores and bounding boxes from query q_i. All objects in the targets are treated as foreground objects, and thus, a binary classifier is used to predict whether an object query is foreground or background (non-object). The classification loss $\ell_{\text{cls}}(\cdot)$ is computed using binary cross-entropy, and the box regression loss $\ell_{\text{box}}(\cdot)$ is computed using L1 box distance and GIoU loss:

$$\mathcal{L}_{\text{DETR}} = \sum_{n=1}^{N} (\ell_{\text{cls}}(s_{\sigma(n)}, \bar{s}_n) + \ell_{\text{box}}(\mathbf{b}_{\sigma(n)}, \bar{\mathbf{b}}_n)), \tag{3}$$

Supervision on the LLM output. The LLM output is supervised through the next-token-prediction manner. However, supervising the object position differs significantly. We recall that for spotting objects, the LLM head does not generate the bounding boxes through text, but predicts an index token that points to the corresponding detection result. Correspondingly, during training, Octopus is trained to predict the index token following each mentioned object. The ground-truth index token is not fixed, but dynamically determined on-the-fly in each training iteration.

Identifying the index token requires matching the ground-truth object to its nearest object query. To this end, we get the predicted bounding box $\bar{\mathbf{b}}$ (the subscript is omitted) of all object queries, and then find the nearest query for the ground-truth object at $\mathbf{b}$ by:

$$C_{\text{loc}} = \left\| \bar{\mathbf{b}} - \mathbf{b} \right\| + \left(1 - \text{GIoU}(\bar{\mathbf{b}}, \mathbf{b})\right), \tag{4}$$

We replace the bounding boxes in the response with the index tokens to the matched object queries. This process introduces slight overload to the training (increasing $\sim 10\%$ training time). In inference, the index tokens are directly generated in the response. Since we exclusively use index tokens to represent bounding boxes without the need to generate bounding boxes token-by-token, our method is significantly faster than previous MLLMs.

The LLM computes the training loss as the language modeling loss using next-token prediction, the same as prior MLLMs.

$$\mathcal{L}_{\text{LM}} = -\sum_{i=1}^{K} \log P(y_i | y_{<i}), \tag{5}$$

where y_i represents the target token at position i. The overall training objective is the combination of DETR training loss and language modeling loss.

Table 1: **Results on Referring Expression Comprehension benchmarks.** We note that Octopus with resolution 224/336 is up to $5\times/4\times$ faster than a purely sequential counterpart Shikra (resolution 224) and achieves higher accuracy. More details of the inference speed comparison are reported in Sec. 4.5

Model Type	Method	Res.	RefCOCO			RefCOCO+			RefCOCOg	
			val	test-A	test-B	val	test-A	test-B	val	test
Generalists	OFA-L[37]	480	79.96	83.67	76.39	68.29	76.00	61.75	67.57	67.58
	VisionLLM[24]	224	-	86.70	-	-	-	-	-	-
	Shikra [6]	224	87.01	90.61	80.24	81.60	87.36	72.12	82.27	82.19
	MiniGPT-v2 [38]	448	88.06	91.29	84.30	79.58	85.52	73.32	84.19	84.31
	Octopus	224	**88.77**	**91.93**	82.28	**83.05**	**88.87**	**75.12**	83.11	**84.78**
	Octopus	336	**89.02**	**92.63**	83.42	**83.55**	**89.40**	**76.02**	**84.25**	**86.19**
Specialists	G-DINO-L [39]	512	90.56	93.19	88.24	82.75	88.95	75.92	86.13	87.02
	UNINEXT-H [40]	640	92.64	94.33	91.46	85.24	89.63	79.79	88.73	89.37

4 Experiments

4.1 Settings

Training details. We train Octopus via three stages, *i.e.*, stage-1 for pretraining vision-language alignment, stage-2 for pretraining the DETR recognition module, and stage-3 for end-to-end instruction fine-tuning. The details are as below:

1) Stage-1 pretrains the vision-language connector for vision-language alignment using LLaVA pretraining data [18]. The visual encoder and LLM are frozen and only the vision-language connector is trained. **2)** Stage-2 pretrains the DETR module on small-scale grounding detection datasets (RefCOCO [41], RefCOCO+ [42], RefCOCOg [42] and Flickr30k [43]) to quickly obtain the recognition ability. We freeze LLM and visual encoder and only train the DETR module in this stage. Stage-2 is mainly for training efficiency, *i.e.*, fast adapting the DETR decoder to the LLM layers. **3)** In stage-3, we finetune the whole LLM head and DETR decoder on LLaVA-Instruct [18], REC data (RefCOCO, RefCOCO+, RefCOCOg, Visual Genome [44]), and Flickr30k end-to-end. Please refer to the Appendix for details of the training datasets.

We adopt AdamW as the optimizer and cosine annealing scheduler. The learning rate is initialized to 1e-4 for stage-1 and stage-2, and 2e-5 for stage-3. The entire training takes about 2 hours for Stage-1 (1 epoch), 4 hours for Stage-2 (2 epochs) and 120 hours for Stage-3 on 8 NVIDIA A100 GPUS.

Architecture details. Octopus adopts the ViT pre-trained from CLIP as the visual encoder. All the vision-language connectors are one-layer MLP with random initialized. The LLM head is initialized with Vicuna-7B-v1.5 [45] and the DETR decoder consists of 4 standard DETR decoder layers. We employ 64 object queries and place the DETR decoder after the 16-th LLM layer.

4.2 Evaluation on REC datasets

We evaluate Octopus's recognition ability on 3 popular referring expression comprehension datasets, *i.e.*, RefCOCO, RefCOCO+ and RefCOCOg. In Table 1, Octopus achieves the highest accuracy on seven out of eight dataset splits, among the compared generalist MLLMs. For example, Octopus outperforms MiniGPT-v2 [46] by +0.96% on val split and +1.34% on test-A split of RefCOCO dataset. On RefCOCO+, the superiority is even larger, *e.g.*, surpassing MiniGPT-v2 by +3.97% on val split, +3.92% on test-A split and +2.70% on test-B split.

We particularly note the comparison against Shikra, a purely sequential counterpart that adopts the same backbone as our Octopus and shares the same training data. Octopus (resolution 224) is up to $5\times$ faster than Shikra under the same image resolution and consistently achieves higher accuracy, *e.g.*, +1.51% on RefCOCO+ test A. When scaling up the image resolution to 336, Octopus is still $4\times$ faster than Shikra (resolution 224) and further enlarges the accuracy superiority to + 2.04% on RefCOCO+ test A. These observations show that parallel recognition improves both the efficiency and accuracy for recognition.

Table 2: **Results on Visual Question Answering benchmarks.** Note that specialists are fine-tuned on each individual evaluation dataset. We gray out those specialists methods, as well as the fine-tuned results of generalists.

Model Type	Method	#LLM Params	Res.	VQAv2	OKVQA	GQA	VizWiz	SciQA
Generalists	BLIP2 [3]	11B	224	65.0	45.9	41.0	19.6	61.0
	InstructBLIP [4]	11B	224	-	-	49.2	34.5	60.5
	Unified-IO$_{XL}$ [47]	2.7B	256	77.9	54.0	-	-	-
	PaLM-E-12B [48]	12B	224	76.2	55.5	-	-	-
	Shikra [6]	7B	224	77.4	47.2	-	-	-
	Octopus	7B	224	**78.5**	**56.0**	**62.29**	**45.6**	65.7
	LLaVA-1.5 [49]	7B	336	78.5	-	62.0	50.0	66.8
	Qwen-VL-Chat [50]	7B	448	78.2	56.6	57.5	38.9	**68.2**
	Octopus	7B	336	**79.2**	**57.2**	**63.3**	**50.1**	67.7
Specialists	GIT [51]	0.7B	384	78.6	-	-	68.0	-
	GIT2 [51]	5.1B	384	81.7	-	-	71.0	-
	PaLI-17B [52]	17B	580	84.3	64.5	-	71.6	-

Table 3: **Results on popular VL benchmarks.** MMB is MMBench [53], LLaVA^W is LLaVA-Bench (In-the-Wild) [18] and MM-V is MM-Vet Benchmark[54]. POPE [55] is reported on the average F1 score of three splits (Adersarial, Popular and Random).

Method	#LLM Params	Resolution	MMB	LLaVA^W	SEED	MM-V	POPE
BLIP-2[3]	13B	224	-	38.1	46.4	22.4	-
InstructBLIP [4]	7B	224	36.0	60.9	53.4	26.2	-
InstructBLIP [4]	13B	224	-	58.2	-	25.6	78.9
IDEFICS [16]	7B	480	48.2	-	-	-	-
IDEFICS [16]	65B	480	54.5	-	-	-	-
LLaVA-1.5 [49]	7B	336	64.3	65.4	58.6	31.1	84.2
QwenVL-Chat [50]	7B	448	60.6	-	58.2	-	-
Shikra [6]	7B	224	58.8	-	-	-	83.9
Octopus	7B	224	**66.2**	63.9	**58.6**	**32.1**	**84.8**

4.3 Evaluation on VQA datasets

General visual question answering (VQA) is a widely employed task for MLLMs. We compare Octopus on 5 VQA datasets against multiple competing MLLMs in Table 2. It is observed that Octopus achieves competitive results under both 224 and 336 resolution settings. For instance, at a 224 image resolution setting, Octopus outperforms Shikra [6] by +1.10Moreover, it surpasses InstructBLIP [4] by +13.09%, +11.1%, and +5.2% on GQA, VizWiz, and SciQA, respectively.

We note that on these datasets, MLLMs are not asked to provide position of the mentioned objects, but Octopus still gains benefit from parallel visual recognition. This suggests that the recognition effectively supports the subsequent understanding, validating the benefit of "parallel recognition $\rightarrow$ sequential understanding" hierarchy.

4.4 Evaluation on recent MLLM benchmarks

We report the performance on the recent popular MLLM benchmarks in Table 3. It is observed that Octopus performs favably against purely sequential counterparts on the five benchmarks. For example, Octopus is higher than Shikra by +7.4% on MMBench and +0.9% on POPE, respectively.

Table 4: **Comparison of inference speed.** We compared the inference speed of our method with the baseline method Shikra [6] on two benchmarks. The reported inference times (in seconds) are averaged for one record across the datasets. RefCOCO contains exactly 1 bounding box in outputs, and Flickr30k contains on average 4 boxes in outputs. Our method is much faster ($\sim 5\times$) than the baseline in both resolutions.

Method	#LLM Params	Resolution	RefCOCO		Flickr30k	
			infer time ↓	FPS ↑	infer time ↓	FPS ↑
Baseline	7B	224	0.89	1.12	3.80	0.26
Octopus (ours)	7B	224	0.17	5.88	0.82	1.21
Baseline	7B	336	1.16	0.86	6.08	0.16
Octopus (ours)	7B	336	0.27	3.70	1.49	0.67

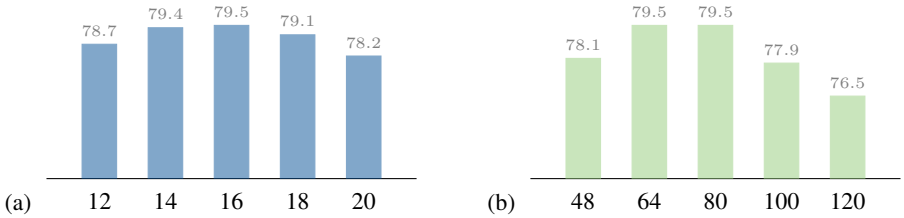

Figure 4: **Influence of the recognition layers and number of object queries.** We evaluate our method on RefCOCOg `val` benchmark. We did not use Visual Genome data [56] in training for efficiency. (a) Influence of the number of LLM layers employed for recognition. (b) Influence of the number of object queries.

Octopus also outperforms InstructBLIP by +3.0% on LLaVA-Bench, +5.2% on SEED and +6.5% on MM-Vet, respectively.

4.5 Ablation Study

The number of LLM layers employed for recognition. We study the optimal layer for positioning DETR within LLM. Fig. 4 (a) illustrates our method's performance on the RefCOCOg val benchmark when placing DETR after different LLM layers. We find that the optimal placement for DETR is after the 16th LLM layer, which is the middle layer of the LLM (consisting of 32 layers in total). We infer that positioning DETR on a lower layer might limit its comprehension of the instruction prompt, while placing it at a higher layer harms the ability of LLM to infer the correct index tokens for the mentioned objects.

Number of object queries. We compare the impact of different object query number on performance in Fig. 4 (b). We evaluate Octopus on the val split of RefCOCOg benchmark and observe that 64 and 80 object query achieved the best performance in our setup. We infer that using insufficient object queries will hamper the recognition capabilities of the bottom LLM layers. On the other hand, although an excess of object queries can enhance the LLM's perception ability for foreground objects, it also increases the complexity for the LLM to select the correct object queries in the final output.

Inference speed. We compare the inference speed of our method against the traditional MLLM on the grounding detection data. As shown in Table 4, our method is much faster ($5\times$) on all benchmarks. This is attributed to that our method does not require to generate bounding boxes in discrete tokens. For each bounding box, we only need to generate an index token to the corresponding detection results predicted by the DETR module. It saves 24 tokens in generating one box, which is more significant in scenarios where a larger number of objects exist.

Qualitative analysis. We visualize the recognition results predicted by DETR for non-grounding data, such as VQA and instruction dialogue, in which no bounding boxes are provided and generated. As shown in Fig. 3 (c), DETR still locates the mentioned objects in the user prompt even if it is not in grounding data format. The detection results are helpful for VQA and some tasks, where the MLLM needs to locate the mentioned objects in the image and answer specific questions.

5 Limitation and Conclusion

Limitations. Due to limited GPU resources, we have not been able to explore how Octopus would perform when scaling up on larger LLM and more open-ended instruction-tuning data. Moreover, DETR is applicable to many recognition tasks beyond detection. We leave it as future works.

Conclusion. We propose Octopus, a novel MLLM framework that disrupts the purely sequential inference paradigm for LLM head. Octopus perform parallel recognition through the lower LLM layers and a lightweight DETR decoder, and then passes the recognition results to the upper LLM layers for further understanding. Consequently, it reformulates the LLM head into a "parallel recognition $\rightarrow$ sequential understanding" hierarchy. Empirical results show Octopus improves accuracy over a range of MLLM tasks and significantly enhances inference efficiency when the task include recognition objectives.

References

[1] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Barun Patra, et al. Language is not all you need: Aligning perception with language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[2] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models, 2023.

[3] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

[4] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023.

[5] 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, 2023.

[6] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal llm's referential dialogue magic, 2023.

[7] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[10] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr. *arXiv preprint arXiv:2201.12329*, 2022.

[11] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[12] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

[13] Elena Agliari, Adriano Barra, Andrea Galluzzi, Francesco Guerra, Daniele Tantari, and Flavia Tavani. Hierarchical neural networks perform both serial and parallel processing. *Neural Networks*, 66:22–35, 2015.

[14] Yoshihito Shigihara and Semir Zeki. Parallel processing in the brain's visual form system: an fmri study. *Frontiers in human neuroscience*, 8:506, 2014.

[15] Kang Li, Mikiko Kadohisa, Makoto Kusunoki, John Duncan, Claus Bundesen, and Susanne Ditlevsen. Distinguishing between parallel and serial processing in visual attention from neurobiological data. *Royal Society Open Science*, 7(1):191553, 2020.

[16] Jean-Baptiste Alayrac et al. Flamingo: a visual language model for few-shot learning. 2022.

[17] Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, Chenliang Li, Yuanhong Xu, Hehong Chen, Junfeng Tian, Qian Qi, Ji Zhang, and Fei Huang. mplug-owl: Modularization empowers large language models with multimodality, 2023.

[18] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[19] 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.

[20] 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. *Advances in Neural Information Processing Systems*, 36, 2024.

[21] Ziyi Lin, Chris Liu, Renrui Zhang, Peng Gao, Longtian Qiu, Han Xiao, Han Qiu, Chen Lin, Wenqi Shao, Keqin Chen, et al. Sphinx: The joint mixing of weights, tasks, and visual embeddings for multi-modal large language models. *arXiv preprint arXiv:2311.07575*, 2023.

[22] Yuhang Zang, Wei Li, Jun Han, Kaiyang Zhou, and Chen Change Loy. Contextual object detection with multimodal large language models. *International Journal of Computer Vision*, pages 1–19, 2024.

[23] Hao Zhang, Hongyang Li, Feng Li, Tianhe Ren, Xueyan Zou, Shilong Liu, Shijia Huang, Jianfeng Gao, Chunyuan Li, Jainwei Yang, et al. Llava-grounding: Grounded visual chat with large multimodal models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2025.

[24] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, and Jifeng Dai. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks, 2023.

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

[26] 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, 2023.

[27] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023.

[28] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *arXiv:2112.01527*, 2021.

[29] Bowen Cheng, Anwesa Choudhuri, Ishan Misra, Alexander Kirillov, Rohit Girdhar, and Alexander G Schwing. Mask2former for video instance segmentation. *arXiv:2112.10764*, 2021.

[30] Zhiqi Li, Wenhai Wang, Enze Xie, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, Ping Luo, and Tong Lu. Panoptic segformer: Delving deeper into panoptic segmentation with transformers. In *CVPR*, 2022.

[31] Lucas Stoffl, Maxime Vidal, and Alexander Mathis. End-to-end trainable multi-instance pose estimation with transformers. *arXiv:2103.12115*, 2021.

[32] Weian Mao, Yongtao Ge, Chunhua Shen, Zhi Tian, Xinlong Wang, and Zhibin Wang. Tfpose: Direct human pose estimation with transformers. *arXiv:2103.15320*, 2021.

[33] Guillem Brasó, Nikita Kister, and Laura Leal-Taixé. The center of attention: Center-keypoint grouping via attention for multi-person pose estimation. In *ICCV*, 2021.

[34] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, 2021.

[35] Moju Zhao, Kei Okada, and Masayuki Inaba. Trtr: Visual tracking with transformer. *arXiv:2105.03817*, 2021.

[36] Fangao Zeng, Bin Dong, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. *arXiv:2105.03247*, 2021.

[37] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*, pages 23318–23340. PMLR, 2022.

[38] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning, 2023.

[39] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2023.

[40] Bin Yan, Yi Jiang, Jiannan Wu, Dong Wang, Ping Luo, Zehuan Yuan, and Huchuan Lu. Universal instance perception as object discovery and retrieval, 2023.

[41] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014.

[42] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 69–85. Springer, 2016.

[43] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649, 2015.

[44] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123:32–73, 2017.

[45] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[46] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*, 2023.

[47] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks, 2022.

[48] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023.

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

[50] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023.

[51] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. Git: A generative image-to-text transformer for vision and language, 2022.

[52] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish Thapliyal, James Bradbury, Weicheng Kuo, Mojtaba Seyedhosseini, Chao Jia, Burcu Karagol Ayan, Carlos Riquelme, Andreas Steiner, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali: A jointly-scaled multilingual language-image model, 2023.

[53] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023.

[54] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2023.

[55] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models, 2023.

[56] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations, 2016.

A Comparison between the intermediate and final recognition results

In Octopus, there are two components that can give recognition results, *i.e.*, the intermediate outputs from the DETR and the final LLM outputs. The intermediate recognition results are parallel and redundant, *i.e.*, predicting multiple boxes for each object of interest. Based on these parallel recognition results, the final LLM outputs choose a single one for each object via the indexing tokens.

We compare these two different results for recognition objectives in Table 5. Since the intermediate recognition results (DETR results) have multiple predictions, we evaluate the rank-k accuracy, which indicates the recall of the ground-truth. We draw two observations as below:

First, the final LLM outputs yield higher accuracy than the rank-1 DETR results, *e.g.*, +1.18% and + 2.12% under the 224 and 336 resolution settings, respectively. It shows that the LLM top layers further improves recognition accuracy after understanding all the recognition results.

Second, the redundant DETR results can achieve high recall of the object-of-interest, *e.g.*, 94.56% rank-5 accuracy under the 224 resolution. Since these intermediate detection results are relayed into the LLM top layers, we infer the high recall is an important reason for the LLM to improve recognition accuracy by selecting one result for each object-of-interest.

In addition to the benefit of improving recognition accuracy, the understanding part of Octopus has another important role: it organizes multiple recognition results into a complete description or response, *e.g.*, in the spotting caption task (Fig. 5 in the main part).

Table 5: **Comparison of Detection Performance.**

Method	Resolution	Octopus Accuracy	DETR results				
			R@1	R@2	R@3	R@4	R@5
Octopus	224	83.11	81.93	90.54	92.85	93.89	94.56
Octopus	336	84.25	82.13	92.79	94.58	95.36	95.95

B Application on referring segmentation

In addition to detection, Octopus is potential to acquire a broader range of recognition abilities, taking advantage of the versatility of the DETR mechanism. Hereby, we endow Octopus with the referring segmentation ability by modifying its object queries into "mask" queries, a common

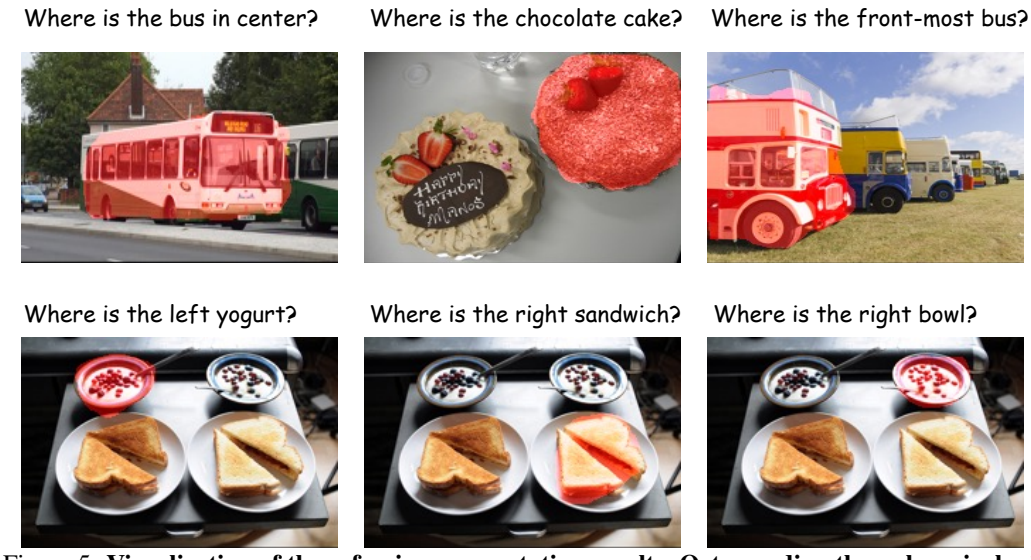

Figure 5: **Visualization of the referring segmentation results. Octopus directly makes pixel-wise predictions rather than predicting the vertexes of the segmentation mask.**

application of DETRs. We add an additional mask head on top of the DETR decoder to predict the segmentation mask for each mask query. Under this setting, we train Octopus on the training datasets of referring segmentation datasets (RefCOCO, RefCOCO+, RefCOCOg). Visualization in Fig. 5 shows that Octopus gains referring segmentation ability. The ability of directly predicting pixel-wise segmentation results rather than vertexes of segmentation masks is valuable for MLLMs.

Table 6: **Referring segmentation results on RefCOCO.**

Method	Resolution	val	testA	testB
Octopus	224	63.6	66.6	61.3

We quantitatively evaluate the referring segmentation performance on the RefCOCO benchmark using cIoU. Table. 6 show that our model achieves reasonable results. Due to time limit, we only implement Octopus using the low-resolution CLIP-ViT features (224×224) for this experiment. The input size is very small for segmentation task and is an important reason that limits our performance (*e.g.*, 63.6 cIoU on RefCOCO validation set). We note a recent state-of-the-art method LISA achieves 74.6. LISA uses an external strong segmentation model, SAM [26], that is pretrained on SA-1B datasets and uses large input size. We conjecture that enlarging the input size and adding training data will bring further improvement to Octopus, as well.

To sum up, by incorporating the bottom layers of LLM head with the DETR query mechanism, Octopus is potential to acquire various recognition abilities. We will explore more forms of recognition abilities for Octopus.

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: We provide our contributions both in the abstract and introduction.

   Guidelines:

   - The answer NA 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 NA 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 express our limitations in Section 5.

   Guidelines:

   - The answer NA 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: [NA]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer NA 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: Yes, we claim our experimental settings in Section 4.1

   Guidelines:

   - The answer NA 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: Our code is currently being organized. We will release the code and data after the review process is complete.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, we provide all training and test details in Section 4.1.

   Guidelines:

   - The answer NA 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: Error bars are not reported because it would be too computationally expensive.

   Guidelines:

   - The answer NA 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 these information in Section 4.1.

   Guidelines:

   - The answer NA 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: Our research follows the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA 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: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA 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 have credited and cited the used assets in our paper.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: Our paper does not release new assets.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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.