PixelLM: Pixel Reasoning with Large Multimodal Model

Zhongwei Ren[1*], Zhicheng Huang[2*], Yunchao Wei[1†], Yao Zhao[1],
Dongmei Fu[2], Jiashi Feng[3], Xiaojie Jin[3*†‡]
[1]Beijing Jiaotong University, [2]University of Science and Technology Beijing, [3]ByteDance Inc.
https://pixellm.github.io/

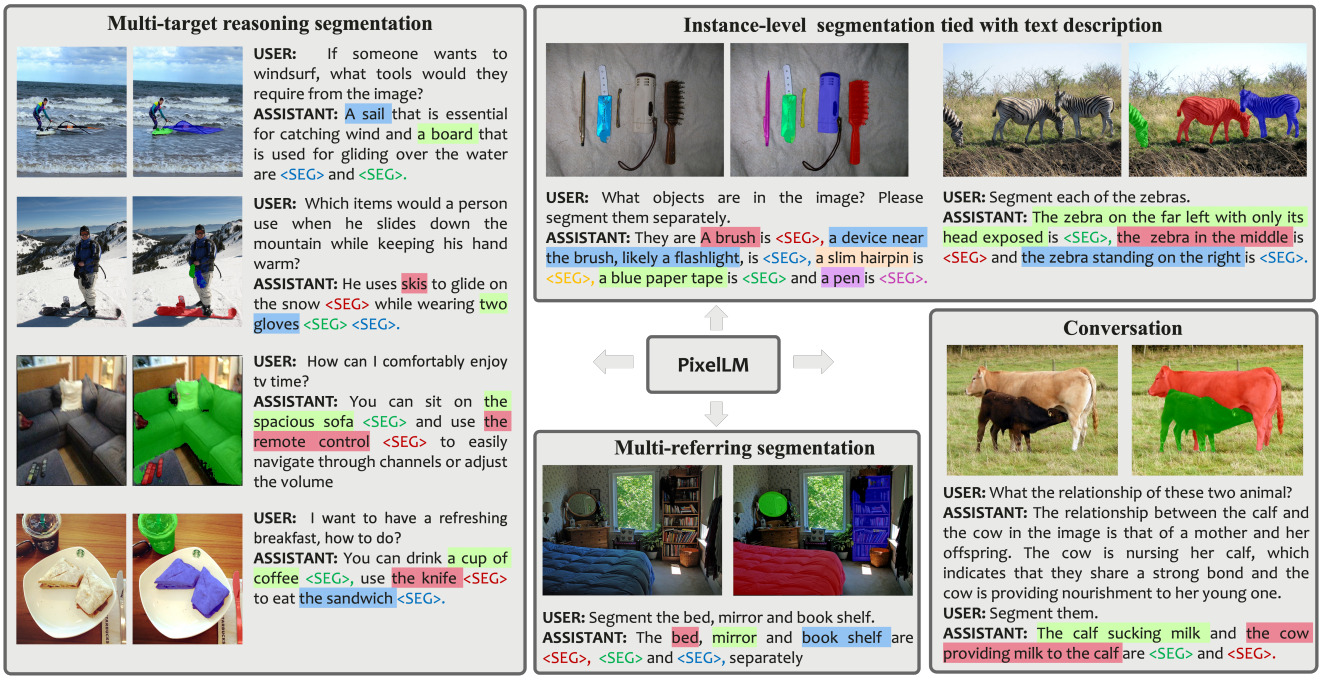


Figure 1. PixelLM is an effective and efficient LMM for pixel-level reasoning and understanding. We show its visualization results in following scenarios: 1) Multi-target reasoning segmentation; 2) Instance-level segmentation tied with text description; 3) Multi-referring segmentation; 4) Conversation

Abstract

While large multimodal models (LMMs) have achieved remarkable progress, generating pixel-level masks for image reasoning tasks involving multiple open-world targets remains a challenge. To bridge this gap, we introduce ***PixelLM****, an effective and efficient LMM for pixel-level reasoning and understanding. Central to PixelLM is a novel, lightweight pixel decoder and a comprehensive segmentation codebook. The decoder efficiently produces masks from the hidden embeddings of the codebook tokens, which encode detailed target-relevant information. With this design, PixelLM harmonizes with the structure of popular LMMs and avoids the need for additional costly segmentation models. Furthermore, we propose a target refinement loss to enhance the model's ability to differentiate between multiple targets, leading to substantially improved mask quality. To advance research in this area, we construct MUSE, a high-quality multi-target reasoning segmentation benchmark. PixelLM excels across various pixel-level image reasoning and understanding tasks, outperforming well-established methods in multiple benchmarks, including MUSE, single- and multi-referring segmentation. Comprehensive ablations confirm the efficacy of each proposed component. All code, models, and datasets will be publicly available.*

1. Introduction

Built upon the success of Large Language Models (LLMs) [6, 24, 25, 37], large multimodal models (LMMs)

*Equal contribution. Work done when Zhongwei and Zhicheng interned at ByteDance Inc. †Correspondence to Xiaojie Jin <jinxiaojie@bytedance.com> and Yunchao Wei <yunchao.wei@bjtu.edu.cn>.
‡ Project lead.

have significantly enhanced high-level visual perception and user interaction experiences [2, 15, 17, 41]. Yet, most of them generate textual descriptions for global images or regions, with limited capability for pixel-level responses like object masks. This research gap limits the practical application of multimodal systems in fine-grained tasks, such as image editing, autonomous driving, and robotics.

Recent work [14] explores using LLMs to produce object masks in a novel reasoning segmentation task, which is more challenging and flexible for real-world applications. In contrast to traditional segmentation which explicitly specifies objects (e.g., "orange"), reasoning segmentation requires complex reasoning for more intricate instructions (e.g., "the fruit high in Vitamin-C"), which aligns well with the capabilities of LMMs. However, this method has two major drawbacks: 1) it cannot handle tasks involving multiple target objects, which are indispensable in real-world scenarios, and 2) it depends on a pre-trained image segmentation model like SAM [13]. This reliance incurs substantial computational demands and confines the overall model's performance to the capability of segmentation model, consequently impeding the model's potential to enhance its performance through further training scaling up.

In this paper, we introduce **PixelLM**, an effective and efficient LMM for pixel-level reasoning and understanding. PixelLM proficiently handles tasks with an arbitrary number of open-set targets and diverse reasoning complexities. Its design maintains the fundamental structure of LMMs while avoiding additional, costly segmentation models, enhancing both efficiency and transferability to diverse applications. Fig. 1 showcases PixelLM's capabilities in diverse segmentation tasks, producing high-quality target masks.

At the core of PixelLM is a novel pixel decoder and a segmentation codebook. The codebook contains learnable tokens that encode contexts and knowledge pertinent to targets referencing at different visual scales. The pixel decoder then produces target masks based on the hidden embeddings from the codebook tokens in conjunction with image features. Thanks to this design, PixelLM can generate high-quality masks without external segmentation models, significantly boosting its efficiency. Furthermore, we propose a target refinement loss to enhance the model's capability of differentiating between multiple targets, thus further improving the mask quality.

To facilitate model training and evaluation in this research area, we construct MUSE, a comprehensive multi-target reasoning segmentation dataset. Utilizing a GPT-4V [25]-aided data curation pipeline, we create 246k question-answer pairs, covering 0.9 million instances. Our extensive ablation studies confirm the dataset's effectiveness in cultivating the model's pixel reasoning capabilities.

PixelLM's effectiveness is clearly demonstrated across a variety of benchmarks. It achieves state-of-the-art performance on benchmarks including MUSE, single- and multi-referring segmentation, significantly outperforming baseline models. Notably, in comparison to models that rely on external segmentation models, such as [14], PixelLM reduces computational costs by up to 50% while either maintaining or improving performance.

In summary, our contributions are:

- We present PixelLM, a novel LMM for pixel-level reasoning and understanding. It handles tasks with diverse reasoning complexities, maintaining high efficiency.
- We construct MUSE, a high-quality multi-target reasoning segmentation dataset, facilitating model training and evaluation in future research.
- PixelLM achieves state-of-the-art results across a diverse range of benchmarks, clearly demonstrating its superior efficacy and efficiency compared to competing methods.

2. Related Work

2.1. Large Multimodal Models

Large multimodal models (LMMs) have significantly enhanced the performance of tasks requiring the understanding of diverse modalities. These models generally fall into two categories based on their use of large language models (LLMs).

The first category [21, 31, 36] involves models trained from scratch or employing relatively smaller language models like BERT for text processing. They typically utilize a blend of contrastive and generative objectives to address a range of multimodal tasks. However, their limited language understanding capacity often hinders their performance in tasks that demand extensive world knowledge and reasoning abilities.

The advent of LLMs has spurred a new direction in LMM development, where LLMs are augmented with multimodal comprehension capabilities [26, 29, 30, 38]. Common approaches in this framework involve integrating adapters to align visual and textual representations within LLMs. Notable examples include BLIP-2 [15], Flamingo [1], MiniGPT-4 [41], llama-adapter [10, 37], LLaVA [17], InstructBLIP [8], InternGPT [19], and Qwen-VL [2]. While these methods have shown improved performance in vision-language tasks through instructional tuning, their primary limitation lies in generating only textual outputs about the entire image, thus constraining their utility in tasks requiring more granular, region-level or pixel-level understanding.

2.2. Fine-Grained LMMs

In many practical applications, understanding visual inputs at a more detailed level is crucial, such as specific regions or even at the pixel level. Several methods have been proposed to endow LLMs with this fine-grained un-

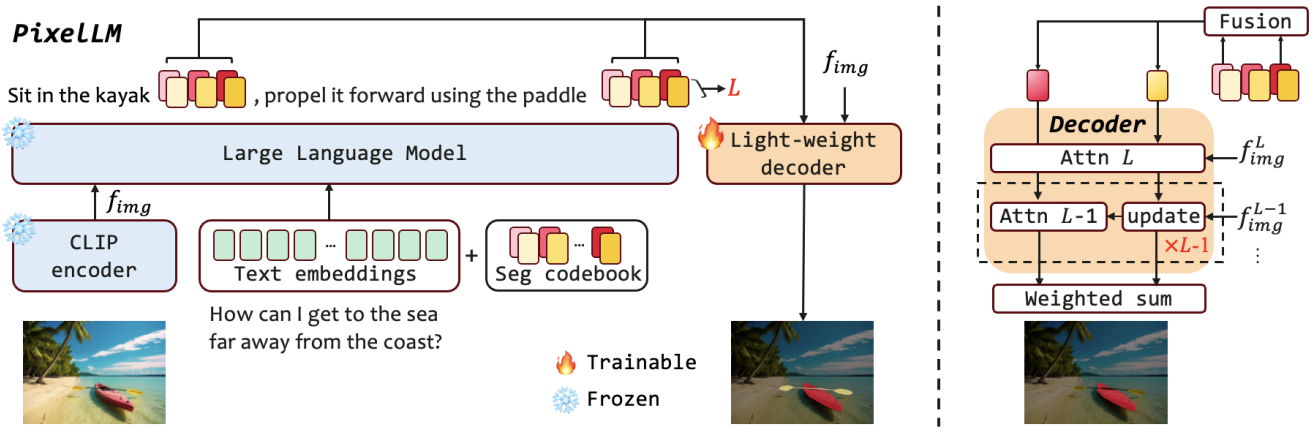


Figure 2. Overview of the proposed PixelLM model architecture. (Left) Overall architecture. (Right) The proposed lightweight pixel decoder. Trainable LoRA parameters are incorporated into the LLM. All parameters except those for the CLIP encoder and LLM are trainable.

derstanding capability. GPT4RoI [39], Shikra [5], Vision-LLM [32], Kosmos-2 [27], InternGPT [20], and Ferret [35] offer grounding capabilities to specified image regions, typically encoding location coordinates as tokens for integration with LLMs. Unlike these methods, which lack the ability to generate pixel-wise masks, LISA [14] integrates SAM with LLMs for segmentation tasks. Moreover, LISA explores the use of LMMs for complex instruction reasoning, which differs from traditional tasks that rely on explicit human instructions for object or category identification. However, LISA is constrained to handling single targets in images, and the incorporation of SAM adds significant computational overhead. In contrast, our goal is to develop an efficient LMM capable of pixel-level image reasoning and understanding, accommodating a varied number of targets and diverse reasoning complexities.

3. Method

We first outline the framework in Sec. 3.1, elucidating two key designs: pixel decoder and segmentation codebook. Training objectives are introduced in Sec. 3.2.

3.1. Model Design

Framework overview As depicted in Fig. 2, PixelLM features a streamlined architecture, comprising four main parts: *i*) a pre-trained CLIP-ViT vision encoder $\mathcal{I}$ which aligns with text, *ii*) a large language model $\mathcal{F}$, *iii*) a lightweight pixel decoder $\mathcal{D}$ and *iv*) a segmentation codebook C_{seg}. PixelLM processes image x_{img} and query text x_{txt}, yielding interleaved text description and corresponding masks for varied number of targets.

While components *i*) and *ii*) adhere to well-established LMM architectures for ensuring compatibility, the pixel decoder and segmentation codebook are pivotal in empowering LMMs with mask generation capabilities across diverse scenarios. We utilize the segmentation codebook to encode target-relevant information into the embeddings of the codebook tokens, which the pixel decoder then transforms in conjunction with image features into precise masks. In the following, we detail their designs.

Segmentation codebook Aiming to enrich the encoding of target-specific information and thereby facilitate the generation of high-quality masks, we devise a comprehensive segmentation codebook. This codebook includes various groups of tokens, each representing different levels of granularity or scales in visual concepts, tailored to meet the demands of segmentation tasks. For example, proficient segmentation requires comprehension of both semantic categories and nuanced geometric shapes. These elements are typically represented at distinct network layers. In response, our single codebook integrates diverse visual information, effectively capturing both semantic and geometric aspects necessary for accurate segmentation.

Specifically, the codebook consists of multiple token groups, each corresponding to a semantic scale of visual features from the image encoder. Formally, we define $C_{\text{seg}} = \{c_n^\ell \in \mathbb{R}^d\}_{n=1,\ell=1}^{N,L}$, where L and N denote the number of visual scales and tokens per group, respectively, and d represents the hidden dimension in LMMs. For clarity, we first set $N = 1$ and introduce how the codebook tokens are integrated within the LMMs to encode requisite information for target mask generation.

For an input image x_{img}, the vision encoder $\mathcal{I}$ extracts a spectrum of multi-scale visual features $I_{\text{img}} = \{I_{\text{img}}^\ell\}_{\ell=1}^{L}$ from $\mathcal{I}(x_{\text{img}})$, comprising L visual features output at select layers of $\mathcal{I}$. The output of the final layer, I_{img}^L, en-

capsulates global image information and is transformed into the language space via a vision-to-language projection layer $p_{V\to T}$. Simultaneously, a vision-to-decoder projection $p_{V\to D}$ transforms all I_{img} features, resulting in $f_{\text{img}} = \left\{ f^{\ell}_{\text{img}} = p_{V\to D}(I^{\ell}_{\text{img}}) \right\}^{L}_{\ell=1}$. The codebook tokens, combined with the input image and text, are then processed by the LLM to generate interleaved response y_{res} in an auto-regressive way:

$$y_{\text{res}} = \mathcal{F}(p_{V\to T}(I^{L}_{\text{img}}), x_{\text{txt}}, C_{\text{seg}}).$$

To help understand this process, consider an example of text query "`Segment the apple on the left`". Then, the output y_{res} contains L tokens of C_{seg}: "`The apple is` $c^1, \ldots, c^L$". The corresponding hidden embeddings (i.e. the output of last layer of $\mathcal{F}$) of C_{seg} are represented as $h = \{h^{\ell}\}^{L}_{\ell=1}$, which are inputs to the pixel decoder $\mathcal{D}$ alongside image features f_{img} for mask generation.

We then explain the rationale behind setting $N > 1$. As depicted in Fig. 3, scenarios involving multiple targets or increased complexity reveal that a single token may not adequately capture the full scope of target semantics, despite the LLM providing precise textual responses. To bolster the model's capacity for interpreting complex reasoning scenarios, we introduce multiple tokens within each scale group and perform a token fusion operation, denoted as $c^{\ell} = \{c^{\ell}_{n}\}^{N}_{n=1}$. Before the decoding process, a linear projection layer ϕ is utilized to convert the hidden states of these grouped tokens into a unified form, represented as $h^{\ell} = \phi(h^{\ell}_1, \ldots, h^{\ell}_N)$. Fig. 3 showcases the application of multiple tokens within each group. The post-decoding attention map visualizations demonstrate that individual tokens contribute distinct, yet complementary information, leading to a more effective mask generation than is possible with a single token approach.

Pixel decoder We design a novel and lightweight pixel decoder $\mathcal{D}$ to adeptly harness the multi-scale features from the vision encoder. This decoder is tasked with learning the transformation of these features, in conjunction with the hidden embeddings from C_{seg}, into precise segmentation masks. Such a design obviates the need for extra costly segmentation models like SAM [14], thus significantly improving efficiency.

As depicted in the right panel of Fig. 2, $\mathcal{D}$ consists of L attention blocks $\{Attn^{\ell}\}^{L}_{\ell=1}$, each corresponding to distinct scales of image features and the codebook. For each targeted mask generation, $\mathcal{D}$ sequentially produces a mask score map m^{ℓ} at each scale ℓ, which then directs the model's attention to regions of higher relevance in the subsequent scale $\ell - 1$. This strategy works by guiding the model to focus on areas with high confidence scores in m^{ℓ}, thereby facilitating more accurate mask generation.

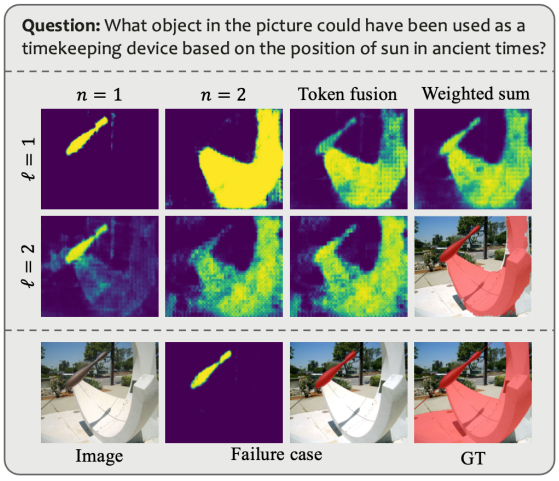


Figure 3. The segmentation codebook example comprises two scales with two tokens each. Each attention map results from the interaction between one token and its corresponding image feature in the decoder. The first two rows depict the token fusion mechanism, while the final row demonstrates a failure case arising from the utilization of only one token.

$$f^{\ell'}_{img} = \begin{cases} f^{L}_{img} & \ell = L \\ f^{\ell}_{img} \odot (\sigma(m^{\ell+1}) + 1) & \ell < L \end{cases} \qquad (1)$$
$$m^{\ell} = Attn^{\ell}(h^{\ell}, f^{\ell'}_{img}).$$

where $f^{\ell'}_{img}$ is the modulated feature at scale ℓ, σ is sigmoid function and $\odot$ is element-wise multiplication. Finally, we learn the weighting factors $\gamma = [\gamma^{\ell}]^{L}_{\ell=1}$ to combine mask maps at all scales to get the final segmentation result: $\hat{M} = \sum^{L}_{\ell=1} \gamma^{\ell} m^{\ell}$ where $|\gamma| = 1$. The detailed structure of the decoder is provided in Supp. A.

3.2. Traning Objectives

Target refinement loss As the number of targets increases, the likelihood of the model encountering confusion and producing overlapping masks intensifies. To mitigate this issue, we introduce a target refinement loss. This simple yet effective strategy focuses on unclear pixels where multiple targets are predicted together. This helps the model in clearly identifying and learning different targets.

Denote the mask predictions as $\left\{ \hat{M}_k \in \mathbb{R}^{H\times W} \right\}^{K}_{k=1}$ where K is the total number of targets, H and W are the shape of mask. $\hat{M}_{k_i} \in \mathbb{R}$ represents the binary value of each pixel. Then we define a map A to assign increased weights to regions predicting multiple targets:

$$A_i = \begin{cases} \alpha, & \sum_k \hat{M}_{k_i} \geq 2 \\ 1, & \sum_k \hat{M}_{k_i} < 2 \end{cases}$$

where α is a hyper-parameter. The weighted loss is com-

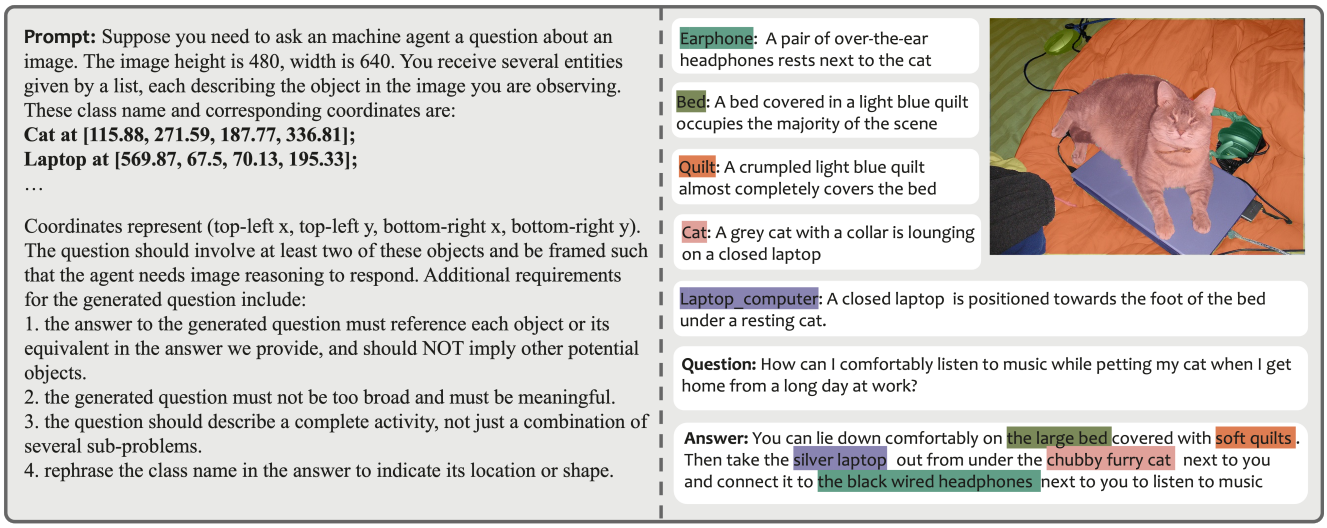


Figure 4. The left panel illustrates the prompt employed in our GPT-4V data generation pipeline. The right panel showcases an example of the generated data.

puted against the ground-truth M_k for each mask as follows:

$$\mathcal{L}_{ref} = \frac{1}{KHW} \sum_k \sum_i A_i \mathcal{L}_{BCE}(\hat{M}_{k_i}, M_{k_i})$$

where $\mathcal{L}_{BCE}$ is per-pixel binary cross-entropy loss.

Overall loss The model is trained end-to-end using an auto-regressive cross-entropy loss $\mathcal{L}_{txt}$ for text generation (note the tokens from the segmentation codebook are also included in the loss calculation), a DICE loss $\mathcal{L}_{dice}$ and a target refinement loss $\mathcal{L}_{ref}$ for mask generation. The overall objective $\mathcal{L}$ constitutes the weighted sum of these losses, calibrated by λ_{ref} and λ_{dice}:

$$\mathcal{L} = \mathcal{L}_{txt} + \lambda_{ref}\mathcal{L}_{ref} + \lambda_{dice}\mathcal{L}_{dice}$$

4. Multi-target Reasoning Segmentation

Our objective is to develop a LLM capable of handling tasks involving an arbitrary number of open-set targets and diverse reasoning complexities. A primary challenge is the absence of a suitable dataset for model training. We review existing public datasets and identify critical limitations: 1) Inadequate detail and object representation in segmentation masks; 2) A lack of question-answer pairs featuring complex reasoning and a varied number of objectives.

To address these issues, we introduce a structured annotation pipeline for constructing the multi-target reasoning segmentation (MUSE) data. Examples of MUSE are shown in Fig. 1. MUSE stands out with its open-set concepts, detailed object descriptions, complex multi-target question-answer pairs, and instance-level mask annotations. In the next, we elaborate on the construction of MUSE.

4.1. MUSE Dataset

Two public datasets, RefCOCO series [12] and ReasonSeg dataset [14], are pertinent to our goal. RefCOCO guides segmentation with explicit target object names, e.g. "orange", lacking more complicated instructions, e.g. "the fruit high in Vitamin-C". Besides, they also fall short in offering multi-target question-answer pairs with target descriptions directly connected to segmentation masks, which however is a common requirement in real-world scenarios, like "How to make fruit salad?"

To this end, a total of 910k high-quality instance segmentation masks are selected from the LVIS dataset, along with detailed textual descriptions based on image content. Utilizing these instances, we construct 246k question-answer pairs, averaging 3.7 targets per answer. This dataset is then divided into three splits: `train`, `val`, and `test`, containing 239k, 2.8k, and 4.3k question-answer pairs, respectively. The test split comprises two parts: the number of targets involved in the question are less or more than three. Please see Supp. C.2 for more analysis of MUSE.

4.2. Dataset Generation Pipeline

We first try to use GPT-4 to construct our dataset: using LLAVA [17] for image captioning and then GPT-4 for generating questions about multiple image regions. We utilize images with pre-existing mask annotations to reduce annotation costs. The image caption, manually selected object names, and bounding box coordinates from the image are input into GPT-4 to facilitate answer selection and question formulation. However, due to the inability to directly perceive image content, the content of question-answer pairs generated by this method is often confined to the caption's description, significantly limiting the diversity of the data.

To address these limitations, the pipeline is refined in two key ways. First, we convert to the more advanced GPT-4V which shows strong capabilities in understanding visual contents [25]. This model has been instrumental in generat-

ing more nuanced and naturalistic questions. Additionally, we implement a more dynamic approach for answer generation. Specifically, we feed all the instance category names and corresponding bounding box coordinates in the image to GPT-4V. Using carefully crafted prompts, GPT-4V autonomously selects instances to construct question-answer pairs relevant to the image content. An example of such a prompt is illustrated in Fig. 4. More details of data filtering and the GPT-4 data generation pipeline are provided in Supp. C.

4.3. Evaluation

To evaluate our task, we focus on three main aspects: *i*) the generation of natural text descriptions aligned with corresponding object masks, *ii*) the accuracy of the match between object masks and text descriptions, and *iii*) the quality of the masks. Since we focus on the mask quality of each target, we do not evaluate image-level captioning.

The evaluation process for each question involves four steps: **First**, we match the predicted masks with ground-truth masks based on mask IoU scores using bipartite matching, similar to DETR [4]. Any unassigned predictions or ground-truths are assigned an empty mask. **Second**, we replace the position of the mask in the generated text with the corresponding ground-truth object description. For example, as shown in Fig. 2, the text might read: "`Sit in the kayak (a red kayak parked on the beach), propel it forward using the paddle (a double-bladed paddle on the kayak)`". The content in brackets is a ground-truth description. **Third**, this modified text prompts GPT-3.5 to score each prediction from 1 to 10, with higher scores indicating better quality, and unassigned predictions receiving a score of 0. **Fourth**, the final score for each prediction is the product of the GPT and IoU scores, from which we calculate the gIoU and cIoU metrics. For more evaluation details, please refer to Supp. A.

5. Experiment

In this section, we first present the implementation details and then show the comparison results on benchmarks. Finally, we ablate on the key components in PixelLM.

5.1. Implementation Details

We use pre-trained multimodal model from LlaVA-7B and LlaVA-llama2-13B, with LoRA adopted for efficient fine-tuning. The vision encoder uses a fixed CLIP-ViT-L/14-336 model, modified with linearly interpolated position embeddings for processing 448x resolution images. The trainable parts of our model include the pixel decoder $\mathcal{D}$, LoRA parameters, segmentation codebook C_{seg}, the vision-to-language and vision-to-decoder projection layers $p_{V\to T}$ and $p_{V\to D}$. To facilitate task evaluation, we convert refCOCO series datasets into multi-referring segmentation datasets (detailed in Supp. C.3). Training involves random sampling from multi-referring segmentation datasets (ADE20K [40], COCO-Stuff [3], LVIS-PACO [28], refCOCO series [12]), VQA data (LLAVA-150k), and MUSE. The training steps follow LISA [14], requiring approximately 1.5/2 days on 8 A100 GPUs for the 7B/13B model.

5.2. Benchmarks and Baselines

Benchmarks We evaluate PixelLM on three benchmarks with a varied number of targets: MUSE, multi-referring segmentation and the conventional referring segmentation (refCOCO series). The former two involve multiple targets and the last one focuses on a single object. Through this evaluation, we validate the versatility of PixelLM in diverse mask generation tasks. For the MUSE benchmark, we use the GPT-based evaluation metric as detailed in Sec. 4.3. In multi-referring segmentation, we formulate multi-target queries from the refCOCO dataset's annotations, asking models to segment 3 to 6 objects per image. These queries are structured as "`Please segment the <objects>`", with <objects> being comma-separated object descriptors. The responses involve generating masks corresponding to the listed objects' sequence. For refCOCO series, we follow previous methods to calculate the gIoU and cIoU scores.

Baselines To our best knowledge, PixelLM is the first to handle complex pixel reasoning tasks involving multiple targets. To demonstrate the effectiveness of PixelLM, we establish four strong baselines for comparative analysis on the aforementioned benchmarks. Three baselines evolve from LISA [14], which is the most relevant work with PixelLM, and one additional non-LLM based model.

- LISA [14]: The original model, designed for single-target segmentation, employs SAM for mask generation.
- LISA$_{\text{rec}}$: To address the limits of LISA, this variant employs a two-step approach: first identify target objects in text form and then ask LISA to segment them one by one. Specifically, LLAVA-13B extracts noun phrase responses from multi-target queries using the prompt: "`<question>, Please only use noun phrases in answer`", which are then input to LISA for mask generation.
- LISA$_{\text{aug}}$: An augmented version of LISA, adapted to generate masks for multiple targets and additionally trained with the MUSE dataset.
- SEEM [43]: A state-of-the-art image segmentation model, SEEM is limited to producing segmentation masks only. Hence, its evaluation is confined to mask quality assessment across each benchmark.

5.3. Results on MUSE

Tab. 1 compares PixelLM with competing methods on the multi-target reasoning segmentation task. For SEEM and the original LISA, since they can only generate a single

Method	w/o SAM	TFLOPs	Val		Test					
			overall		few targets		many targets		overall	
			gIoU	cIoU	gIoU	cIoU	gIoU	cIoU	gIoU	cIoU
SEEM [43]	✓	0.43	13.6	16.2	23.6	24.9	8.5	13.2	11.7	15.7
LISA-7B [14]	✗	7.16	18.8	29.0	24.7	36.5	9.6	24.5	12.8	27.1
LISA-7B$_{rec}$	✗	7.16	24.5	31.1	30.0	30.9	12.4	23.2	16.2	24.8
LISA-7B$_{aug}$	✗	7.16	42.0	46.1	43.5	52.0	37.7	42.3	38.9	44.4
PixelLM-7B†	✓	**3.57**	39.9	48.0	43.1	56.7	36.0	38.2	37.5	42.2
PixelLM-7B	✓	**3.57**	**42.6**	**50.7**	**44.6**	**59.2**	**37.7**	**42.8**	**39.2**	**46.3**
LISA-Llama2-13B [14]	✗	10.24	20.4	29.2	27.5	38.5	10.9	25.6	14.4	28.4
LISA-Llama2-13B$_{aug}$	✗	10.24	43.6	50.2	44.7	60.0	41.2	**47.9**	41.9	50.5
PixelLM-Llama2-13B†	✓	**6.65**	43.0	51.7	44.8	61.6	39.3	44.6	40.5	48.2
PixelLM-Llama2-13B	✓	**6.65**	**44.8**	**54.1**	**45.2**	**62.9**	**41.5**	47.6	**42.3**	**51.0**

Table 1. **Comparison on MUSE benchmark.** † denotes PixelLM w/o using the token fusion mechanism and target refinement loss.

Method	w/o SAM	MrefCOCO			MrefCOCO+			MrefCOCOg	
		val	testA	testB	val	testA	testB	val(U)	test(U)
LISA [14]	✗	36.7	38.3	36.4	34.0	36.3	32.1	34.5	36.2
LISA$_{aug}$	✗	68.9	70.8	66.3	59.8	62.2	54.1	62.3	63.9
PixelLM†	✓	70.3	74.2	66.2	64.4	69.6	57.0	64.0	67.0
PixelLM	✓	**72.7**	**76.2**	**68.1**	**65.7**	**71.3**	**57.7**	**65.8**	**67.7**

Table 2. **Results on the multi-referring segmentation benchmark.** The meaning of † is the same as Tab. 1.

Method	w/o SAM	refCOCO			refCOCO+			refCOCOg	
		val	testA	testB	val	testA	testB	val(U)	test(U)
MCN [22]	✓	62.4	64.2	59.7	50.6	55.0	44.7	49.2	49.4
VLT [9]	✓	67.5	70.5	65.2	56.3	61.0	50.1	55.0	57.7
CRIS [33]	✓	70.5	73.2	66.1	62.3	68.1	53.7	59.9	60.4
LAVT [34]	✓	72.7	75.8	68.8	62.1	68.4	55.1	61.2	62.1
ReLA [16]	✓	73.8	76.5	70.2	66.0	71.0	57.7	65.0	66.0
X-Decoder [42]	✓	-	-	-	-	-	-	64.6	-
SEEM [43]	✓	-	-	-	-	-	-	65.7	-
LISA [14]	✗	**74.1**	**76.5**	**71.1**	62.4	67.4	56.5	66.4	68.5
LISA$_{aug}$	✗	74.0	76.3	70.4	62.5	66.3	56.0	67.0	69.1
PixelLM	✓	73.0	**76.5**	68.2	**66.3**	**71.7**	**58.3**	**69.3**	**70.5**

Table 3. **Results on the referring segmentation benchmark.**

segmentation mask given query text, we report the IoU between prediction and the ground truth which merges all target masks. Although the text description for the target is ignored, thus leading to a simpler setting when compared with ours, we can observe their performance is still inferior, which demonstrates the challenges of this task. Although LISA$_{rec}$ improves upon LISA by simplifying the generation task by first generating the target for multi-target queries, it still largely lags behind our models. This highlights the importance of end-to-end training on the task. By training on MUSE, the performance of the augmented LISA LISA$_{aug}$ further improves on the dataset, which validates the benefits of our dataset in training an LMM with strong pixel reasoning capabilities.

PixelLM stands out in both efficiency and performance. Under the fair setting involving equivalent training datases and LMM sizes, PixelLM-7B and PixelLM-Llama2-13B consistently outperform their counterparts (LISA$_{aug}$ and LISA-Llama2-13B$_{aug}$, respectively) across almost all evaluated metrics. Notably, PixelLM eschews additional segmentation modules, thereby conferring substantial efficiency advantages. In terms of computational efficiency, PixelLM achieves a remarkable reduction in TFlops by 50% and 35% for the 7B and 13B model sizes, respectively, compared to LISA. Interestingly, while PixelLM benefits from model scaling up: PixelLM-Llama2-13B boosts upon the PixelLM with a 7B LMM, it still maintains smaller TFlops (6.65 vs. 7.16) than LISA$_{aug}$ which includes an extra SAM model. Furthermore, we conduct an ablation study to ascertain the impact of the proposed token fusion mechanism and target refinement loss. PixelLM, devoid of these two components, is denoted as PixelLM†. The results clearly demonstrate that the complete PixelLM configuration significantly outperforms PixelLM†, thereby affirming the contribution of these components to the model's efficacy. This trend is consistent in the 13B model variant as well. Detailed ablation analyses of each component are presented in Sec. 5.5.

5.4. Results on Referring Segmentation

Tab. 2 presents the results on the multi-target referring segmentation dataset, wherein PixelLM demonstrates superior performance across all data splits, significantly outperforming LISA and its enhanced variant, LISA$_{aug}$. Notably, PixelLM exhibits enhanced efficiency compared to the LISA models, as evidenced by its lower TFlops (as detailed in Tab. 1). The superiority of the token fusion and target refinement loss components within PixelLM is further corroborated through the comparison with PixelLM†.

We also compare results on the conventional single-target refCOCO series dataset. Despite the dataset's focus on single target segmentation, PixelLM attains commendable performance. As shown in Tab. 3, PixelLM achieves the highest scores on most metrics, particularly excelling in the more challenging refCOCO+/refCOCOg datasets.

Architecture	MUSE Val		refCOCOg	
	gIoU	cIoU	val(U)	test(U)
baseline	40.1	47.2	64.3	65.6
+ 2 scale feature-fusion	**42.6**	50.7	69.3	**70.5**
+ 3 scale feature-fusion	42.3	**51.4**	**69.8**	70.4

(a) Different number of scales.

Token number N per-group	MUSE Val		refCOCOg	
	gIoU	cIoU	val(U)	test(U)
1	40.7	48.6	68.0	68.3
2	42.0	49.5	69.0	69.9
3	42.6	**50.7**	**69.3**	70.5
4	**43.0**	50.3	69.1	**70.6**

(b) Different token number in each group.

Token fusion	Target refinement loss	MUSE Val		refCOCOg	
		gIoU	cIoU	val(U)	test(U)
		39.9	48.0	68.0	68.3
✓		41.5	50.1	69.3	70.5
	✓	40.7	48.6	68.0	68.3
✓	✓	**42.6**	**50.7**	**69.3**	**70.5**

(c) Token fusion mechanism and target refinement loss.

Data generator	Data amount	MUSE Val	
		gIoU	cIoU
GPT-4	30k	30.0	35.7
GPT-4V	30k	35.6	40.0
	100k	38.9	45.7
	200k	**42.6**	**50.7**

(d) Different data generators and number of question-answer pairs.

Table 4. **Ablations.** We conduct all experiments based on our 7B model.

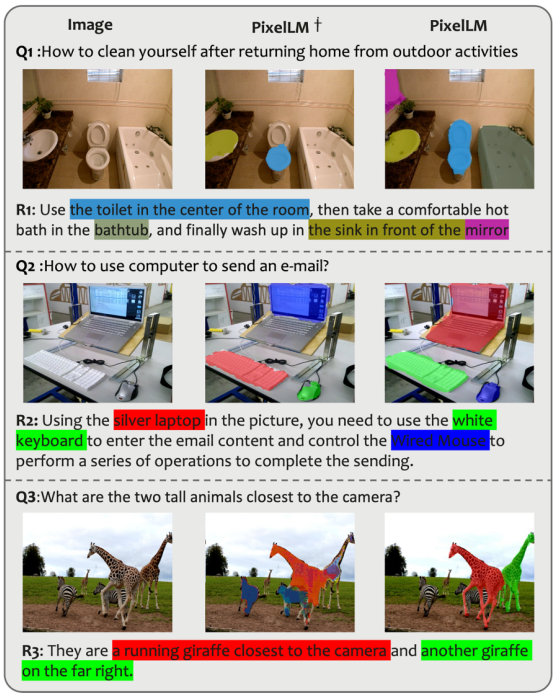


Figure 5. **Comparison between PixelLM and PixelLM†** (w/o token fusion mechanism and target refinement loss.)

5.5. Ablation Study

Number of scales. Tab. 4a shows the effects of increasing the number scales in PixelLM by adding more feature layers and corresponding codebook tokens to the decoder. We observe notable gains by adding just one extra scale (the 2nd row). The gain diminishes with more scales.

Token fusion mechanism In this ablation, the number of tokens in a group is 3. Table 4c shows that applying token fusion achieves up to a 2.4% and 2.2% increase in cIoU on MUSE val and refCOCOg, respectively. This demonstrates its benefits for both multi-target and single-target tasks. **Token number in each scale** Tab. 4b explores how changing the number of tokens per group (denoted as N in Section 3.1) affects performance. $N = 1$ corresponds to no token fusion. Increasing N to two and three consistently boosts performance, yielding increases of 0.9% to 2.1% in cIoU and 1.3% to 2.1% in gIoU. However, further increases yield only slight improvements in metrics.

Target refinement loss As Table 4c shows, target refinement loss results in a 0.8% improvement in gIoU and 0.6% in cIoU on MUSE val. This loss, designed for scenarios with multiple targets, does not affect performance in refCOCOg. Combining the loss with token fusion can lead to up to a 2.7% improvement in cIoU on MUSE. Fig. 5 compares the use (third column) and non-use (second column) of these two designs, wherein the former shows higher mask quality and better target completeness than the former.

GPT-4V *vs*. GPT-4 To showcase GPT-4V's advantage over GPT-4 in data generation, we create 30,000 extra multi-target question-answer pairs for a balanced comparison. As one can see from Tab. 4d, GPT-4V-generated data shows superior performance in the MUSE benchmark, with up to 7.0% gIoU and 8.6% cIoU gains using the same data amount. Additionally, increasing MUSE training data volume leads to consistent performance improvements, indicating that expanding and enhancing the MUSE dataset allows PixelLM to gradually improve its general pixel-level understanding.

6. Conclusion

In this study, we introduce PixelLM, an effective and efficient LMM for pixel-level image reasoning and understanding. Benefiting from novel designs, PixelLM is adept at producing high-quality masks for a variety of tasks. Moreover, we construct a comprehensive multi-target reasoning segmentation benchmark to bolster this research area. Through extensive experiments, PixelLM achieves promising results across various benchmarks. Future endeavors will focus on the expansion and enhancement of PixelLM's capabilities.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022. 2

[2] 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. *arXiv preprint arXiv:2308.12966*, 2023. 2

[3] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *Computer Vision and Pattern Recognition*, pages 1209–1218, 2018. 6

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. pages 213–229, 2020. 6, 11

[5] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal llm's referential dialogue magic. *arXiv preprint arXiv:2306.15195*, 2023. 3

[6] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90% chatgpt quality, 2023. 1

[7] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020. 11

[8] 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. 2

[9] Henghui Ding, Chang Liu, Suchen Wang, and Xudong Jiang. Vision-language transformer and query generation for referring segmentation. In *International Conference on Computer Vision*, pages 16321–16330, 2021. 7

[10] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*, 2023. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 12

[12] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Empirical Methods in Natural Language Processing*, pages 787–798, 2014. 5, 6

[13] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 2

[14] 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. 2, 3, 4, 5, 6, 7

[15] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023. 2

[16] Chang Liu, Henghui Ding, and Xudong Jiang. Gres: Generalized referring expression segmentation. In *Computer Vision and Pattern Recognition*, pages 23592–23601, 2023. 7

[17] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. 2, 5, 11

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 12

[19] Zhaoyang Liu, Yinan He, Wenhai Wang, Weiyun Wang, Yi Wang, Shoufa Chen, Qinglong Zhang, Zeqiang Lai, Yang Yang, Qingyun Li, Jiashuo Yu, et al. Interngpt: Solving vision-centric tasks by interacting with chatgpt beyond language. *arXiv preprint arXiv:2305.05662*, 2023. 2

[20] Zhaoyang Liu, Yinan He, Wenhai Wang, Weiyun Wang, Yi Wang, Shoufa Chen, Qinglong Zhang, Zeqiang Lai, Yang Yang, Qingyun Li, et al. Interngpt: Solving vision-centric tasks by interacting with chatgpt beyond language. *arXiv preprint arXiv:2305.05662*, 2023. 3

[21] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks. In *International Conference on Learning Representations*, 2022. 2

[22] Gen Luo, Yiyi Zhou, Xiaoshuai Sun, Liujuan Cao, Chenglin Wu, Cheng Deng, and Rongrong Ji. Multi-task collaborative network for joint referring expression comprehension and segmentation. In *Computer Vision and Pattern Recognition*, pages 10034–10043, 2020. 7

[23] Microsoft. Deepspeed. Technical report, Microsoft, 2023. 11

[24] OpenAI. Chatgpt: A language model for conversational ai. Technical report, OpenAI, 2023. 1

[25] OpenAI. Gpt-4 technical report, 2023. 1, 2, 5

[26] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023. 2

[27] 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. 3

[28] Vignesh Ramanathan, Anmol Kalia, Vladan Petrovic, Yi Wen, Baixue Zheng, Baishan Guo, Rui Wang, Aaron Marquez, Rama Kovvuri, Abhishek Kadian, et al. Paco: Parts and attributes of common objects. In *Computer Vision and Pattern Recognition*, pages 7141–7151, 2023. 6

[29] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023. 2

[30] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2, 11

[31] 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. 2

[32] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *arXiv preprint arXiv:2305.11175*, 2023. 3

[33] Zhaoqing Wang, Yu Lu, Qiang Li, Xunqiang Tao, Yandong Guo, Mingming Gong, and Tongliang Liu. Cris: Clip-driven referring image segmentation. In *Computer Vision and Pattern Recognition*, pages 11686–11695, 2022. 7

[34] Zhao Yang, Jiaqi Wang, Yansong Tang, Kai Chen, Hengshuang Zhao, and Philip HS Torr. Lavt: Language-aware vision transformer for referring image segmentation. In *Computer Vision and Pattern Recognition*, pages 18155–18165, 2022. 7

[35] Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. Ferret: Refer and ground anything anywhere at any granularity. *arXiv preprint arXiv:2310.07704*, 2023. 3

[36] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022. 2

[37] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023. 1, 2

[38] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. 2

[39] Shilong Zhang, Peize Sun, Shoufa Chen, Min Xiao, Wenqi Shao, Wenwei Zhang, Kai Chen, and Ping Luo. Gpt4roi: Instruction tuning large language model on region-of-interest. *arXiv preprint arXiv:2307.03601*, 2023. 3

[40] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 6

[41] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023. 2

[42] Xueyan Zou, Zi-Yi Dou, Jianwei Yang, Zhe Gan, Linjie Li, Chunyuan Li, Xiyang Dai, Harkirat Behl, Jianfeng Wang, Lu Yuan, et al. Generalized decoding for pixel, image, and language. In *Computer Vision and Pattern Recognition*, pages 15116–15127, 2023. 7

[43] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. *Advances in Neural Information Processing Systems*, 2023. 6, 7

PixelLM: Pixel Reasoning with Large Multimodal Model

Supplementary Material

In this supplementary material, we first detail the training configuration, decoder structure, flops calculation, and MUSE evaluation process in Sec.A. We then present an additional experimental analysis of our segmentation codebook and decoder in Sec.B. Furthermore, we offer a more comprehensive analysis of MUSE and the multi-referring segmentation dataset in Sec. C.

A. Implementation Details

Training details. We give the detailed training configuration in Tab. A.1, and we do *not* use color jittering, drop path or gradient clip. The gradient accumulation step is set to 10.

Structure of pixel decoder. The decoder can be divided into three parts based on their functions: *i*) the attention block for each scale; *ii*) using the output mask from one scale to modulate the features in the next scale; *iii*) the fusion of masks from all scales to obtain the final results. We present the PyTorch-style pseudocodes for the overall decoder and each part in Alg. 1.

Calculation of TFLOPs. We compare models' TFlops (trillion floating-point operations per second) in Tab. 1. The calculation follows the formula in [7] and the script inDeepSpeed [23]. Since the flops for LMMs vary with the generated token length, we standardize it at 512. This length aligns with the common default used by [17, 30] and suffices to accommodate dozens of target objects.

Details of the evaluation metric. Sec. 4.3 provides a concise overview of the MUSE evaluation pipeline. In this section, we delve into a more formal and detailed explanation of its design. Let us denote by $M = \{M_g\}_{g=1}^{G}$ the ground truth set of G objects, and $\hat{M} = \{\hat{M}_k\}_{k=1}^{K}$ the set of K predictions. Motivated by [4], assuming K is not equal to G, we use $\varnothing$ (no objects) to pad the smaller set and both sets finally have size $P = \max(G, K)$.

(1) We find a bipartite matching between these two sets by searching for a permutation of P elements, $\sigma \in \mathfrak{S}_P$, with the lowest cost:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_P} \sum_{i}^{P} \mathcal{L}_{match}(M_i, \hat{M}_{\sigma(i)})$$

where $\mathcal{L}_{match}(M_i, \hat{M}\sigma(i))$ is a pairwise matching cost between ground truth M_i and a prediction with index $\sigma(i)$. We compute this optimal assignment efficiently with the Hungarian algorithm. We define $\mathcal{L}_{match}(M_i, \hat{M}\sigma(i))$ as $\mathcal{L}_{bce}(M_i, \hat{M}\sigma(i)) + \mathcal{L}_{dice}(M_i, \hat{M}\sigma(i))$.

(2) Based on the matching results, we modify the generated response y_{res} to y'_{res}: since each $\hat{M}_i$ originates from a segmentation token sequence in y_{res}, we replace each sequence with the GPT-generated description of M_i.

(3) We use a carefully designed prompt for GPT-3.5 to assign a score s_i to each $\hat{M}_i$ in the answer in a single step. An example of this methodology is depicted in Fig. A.1. The empty predictions are directly scored with 0.

The above three steps assess the model's capability to generate outputs where masks are intertwined with text descriptions and evaluate how accurately these masks correspond to their respective text descriptions. Then we evaluate the quality of the masks.

(4) The final IoU of each prediction is:

$$\text{Intersection}_i = \begin{cases} \text{Intersection}_i & s_i > 0.5 \\ 0 & s_i \leq 0.5 \end{cases}$$

$$\text{IoU}_i = \text{Intersection}_i / \text{Union}_i$$

And the final IoU_{img} of each image is:

$$\text{IoU}_{img} = \sum\nolimits_i \text{IoU}_i / P$$

config	value
optimizer	AdamW
base learning rate	3.0e-4
weight decay	0
optimizer monmentum	β_1, β_2=0.9,0.95
batch size	16
learning rate schedule	WarmipDecayLR
warmup iterations	100
augmentations	None
α	2.0
λ_{ref}	2.0
λ_{dice}	0.5

Table A.1. **Training settings.**

Codebook design	MUSE Val		refCOCO+			refCOCOg	
	gIoU	cIoU	val	testA	testB	val(U)	test(U)
N	41.0	48.3	64.0	69.8	57.5	67.9	68.4
$N \times L$	**42.6**	**50.7**	**66.3**	**71.7**	**58.3**	**69.3**	**70.5**

Table A.2. **Sharing tokens across L scales.** The first row corresponds to results of sharing tokens across all feature scales.

Selected layers	MUSE Val		refCOCO+			refCOCOg	
	gIoU	cIoU	val	testA	testB	val(U)	test(U)
baseline	40.1	47.2	61.1	65.4	54.7	64.3	65.6
20, 17, 14	42.0	51.5	66.0	71.4	58.5	68.8	70.6
20, 17	41.2	49.1	65.9	71.2	58.0	68.0	66.4
23, 14, 10	41.8	48.0	65.1	68.3	57.9	67.5	68.0
23, 14, 20	42.3	**51.4**	66.0	71.5	**58.5**	**69.8**	70.4
23, 14	**42.6**	50.7	**66.3**	**71.7**	58.3	69.3	**70.5**

Table A.3. **Multi-scale layer selection.** CLIP-ViT consists of 24 layers. "Baseline" only uses the penultimate (i.e., the 23rd) layer.

Algorithm 1: Pseudo codes of our pixel decoder.

```
# Inputs: f_img:image features from L scales [L,C,H,W];
  h_seg:segmentation tokens for L scales [L, C];
# Variables: lev_token: Learnable embeddings for L scales
  [L,C]; out_token:Learnable embeddings [N, C];
  image_pe:position embedding of image features; gamma:mask
  weighting factors [L]
# Functions: SelfAttention();CrossAttention();MLP();
  up_scale(); down_scale();
def feature_update(f, mask):
    # f:image feature of one scale.[CxHxW]
    # mask:output from the attention block above f.[Cx4Hx4W]
    mask = down_scale(mask, size=(HxW)); # mask: [H,W]
    f = f * (sigmoid(mask) + 1) # update feature
    return f
def attention_block(h, f, l):
    # h, f:segmentation token and image feature of one
      scale.[C], [CxHxW]
    # l:scale index
    token = cat([out_token, h], dim=0) + lev_token[l];
      # token:[N+1,C]
    attn_out = SelfAttention(q=k=v=token);# self attention
      output:[N+1,C]
    token = norm(token + attn_out)
    key = f + image_pe; # key:[C,H,W]
    attn_out = CrossAttention(q=token, k=key, v=f);# cross
      attention output:[N+1,C]
    token = norm(token + attn_out + MLP(token)) ;# update
      token:[N+1,C]
    attn_out = CrossAttention(q=key, k=token,
      v=token);# cross attention output:[C,H,W]
    f = norm(f + attn_out); # update feature:[C,H,W]
    f_up = up_scale(f) # f_up:Cx4Hx4W
    token = MLP(token) # token:[N+1,C]
    mask = token @ f # mask:[N+1,4H,4W]
    mask = mean(mask, dim=0) # mask:[4H,4W]
    return mask
def mask_fusion(mask_list):
    # masks:a list of masks from each scale
    final_mask = zeros(4Hx4W); # initialize empty mask
    for l, m in enumerate(mask_list):
        final_mask = final_mask + gamma[l] * m
    return final_mask
# Main Function
def pixel_decoder(f_img, h_seg):
    # masks:a list of mask from each scale
    mask_list = []
    for l, (f, h) in enumerate(zip(f_img, h_seg)):
        if l < L-1:
            f = feature_update(f, mask)
        ; # update features after the scale L
        mask = attention_block(h, f, l)
        mask_list.append(mask)
    final_mask = mask_fusion(mask_list)
    return final_mask
```

Based on the IoU scores, we can calculate gIoU and cIoU metrics by referring segmentation dataset.

B. More Ablative Experiments.

Multi-scale tokens sharing. To clearly demonstrate the necessity of our multi-scale segmentation tokens in the codebook C_{seg}, we continue using L multi-scale features, but reduce the original codebook shape from $N \times L$ to N (recall that N is the number of tokens in each scale group). This reduction resulted in the remaining N tokens being identical (i.e., shared) across all L scales. As Tab. A.2 shows, using a dedicated segmentation token for each scale yields better performance.

Multi-scale layer selection. We experiment with different layer selections as presented in Tab. A.3. The CLIP-ViT-L used in PixelLM consists of 24 layers. To reduce the computational cost of the decoder, we avoid selecting features from all layers of ViT. Instead, we follow the multi-scale feature selection ratio from prior work [11, 18] and consider other selection options. Our experiments reveal that selecting features from layers before the middle does not yield benefits for our task (the fourth row in Tab. A.3). Therefore, we mainly focus on selections from the middle and rear layers. The results show that using features from layers 14 and 23 leads to the best outcomes.

C. More Details about MUSE

C.1. Data Filtering

GPT-4V filtering. Although GPT-4V can efficiently understand image content, there are still failure cases in the generated data, which can be summarized in the following two points:

- Questions are vague and open to multiple interpretations. For example, the question "What should I take with me on my outing?" is extremely vague because "outing" can mean a wide array of activities.
- Answers that omit semantically equivalent instances. For instance, a question might ask about "choosing a fruit for a snack", but the answer may only suggest "an apple", ignoring other fruit visible in the image.

Therefore, it is necessary to employ a stringent sample filtering process to guarantee the quality of the data. Toward this goal, we develop a GPT-4V assisted data filtering pipeline. This pipeline operates by prompting GPT-4V to evaluate all initial question-answer pairs, based on identified common failure modes. Pairs classified by GPT-4V as falling within these failure categories are removed. This procedure effectively excludes approximately 20% of the preliminary data. The specific prompts used in this process are detailed in Fig. A.1.

Human verification. To ensure high quality in the generated question-answer pairs during the evaluation stage, we further engage experienced human annotators to double-check our evaluation set. Our approach is driven by two primary objectives:

- The questions should follow an intuitive and logical sequence that a person would typically think of when viewing the image.
- The answers should correspond closely to the way a human would naturally respond to the question

The above filtering process effectively ensures that the questions in the MUSE dataset are sufficiently challenging for

Prompt: You are an intelligent chatbot designed to evaluate the correctness of generative outputs for question-answer pairs. You receive a list of objects, each described in the image you are observing. The image has a height of 480 and a width of 640. The image caption, objects in the image, and their respective bounding box coordinates are as follows:
Global Caption;
A grey cat with a collar is lounging on a closed laptop [115.88, 271.59, 187.77, 336.81];
...

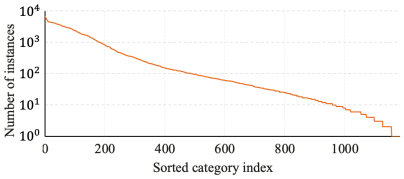
Coordinates represent (top-left x, top-left y, bottom-right x, bottom-right y). The question should involve at least two of these objects and must be framed in a way that requires image reasoning for response. Additional requirements for the generated question include:
1.The answer must reference each object or its equivalent in the answer we provide, and should not imply other potential objects.
2.The question must be specific and meaningful, not overly broad.
3.The question should describe a complete activity, rather than just combining several sub-problems.
4.In the answer, rephrase the class name to indicate its location or shape.

Question: How can I comfortably listen to music while petting my cat when I get home from a long day at work?

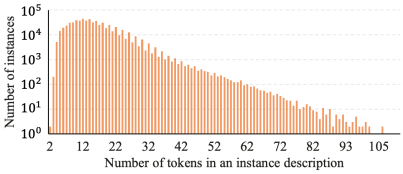
Generated Answer: You can lie down comfortably on the large bed <SEG> covered with soft quilts <SEG> . Then take the silver laptop <SEG> out from under the chubby furry cat <SEG> next to you and connect it to the black wired headphones <SEG> next to you to listen to music.

Modified Answer: You can lie down comfortably on the large bed (A bed covered in a light blue quilt occupies the majority of the scene) covered with soft quilts (A crumpled light blue quilt almost completely covers the bed). Then take the silver laptop (A closed laptop is positioned towards the foot of the bed under a resting cat) out from under the chubby furry cat (A grey cat with a collar is lounging on a closed laptop) next to you and connect it to the black wired headphones (A pair of over-the-ear headphones rests next to the cat) next to you to listen to music.

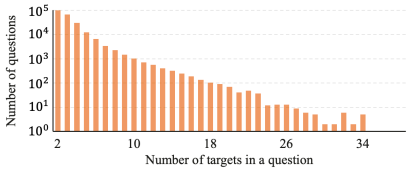
Figure A.1. **Evaluation example.** The left panel illustrates the prompt employed in our GPT evaluation pipeline. The right panel showcases an example of a predicted answer alongside its corresponding modified version, as input to GPT.



(a) The number of instances per category



(b) Distribution of token count in instance descriptions.



(c) Distribution of target count in questions.

Figure A.2. **Dataset statistics**. Best viewed digitally.

reasoning, while the answers remain detailed and accurate.

C.2. Dataset Statistics

In this section, we systematically analyze our dataset. First, our question-answer pairs are based on over 1000 categories, encompassing a wide spectrum of objects found in daily scenes. Additionally, the descriptions of objects in the answers go beyond mere category names limited to a few tokens. Instead, they offer context-specific descriptions extending to over 100 tokens. This demonstrates that our dataset is rich in perception information, crucial for real-world applications. Finally, we present the statistics regarding the number of objects involved in a data sample.

Category statistics. There are over 1000 categories in MUSE from the original LVIS dataset, and 0.9 million instances with unique descriptions that vary based on the context of the question-answer pairs. Fig. A.2a shows the number of instances per category on all question-answer pairs. The distribution inherits the low-shot nature of LVIS.

Token count. Fig. A.2b presents the distribution of instances by token count in their descriptions, highlighting a wide range that exceeds 100 tokens in the most extensive cases. These descriptions are not limited to simple category names; rather, they are substantially enriched with detailed information about each instance, encompassing aspects like appearance, attributes, and relationships with other objects, thanks to our GPT-4V-based data generation pipeline. The depth and variety of information in the dataset bolster the trained model's generalization capabilities, enabling it to effectively address open-set questions.

Target count. Fig. A.2c presents statistics on the number of targets in each question-answer pair. The average number of targets is 3.7, with the maximum number of targets in a single pair reaching up to 34. This number can cover most scenarios of target reasoning for a single image.

C.3. Multi-referring Segmentation

As mentioned in Sec. 5.1, we transform conventional referring segmentation datasets into a multi-referring format for model training. In this subsection, we detail this process. The transformation involves selecting one to three distinct target objects from the annotations of each image.

Prompt: You are an intelligent chatbot designed to evaluate the correctness of generative outputs for question-answer pairs. You receive a list of objects, each describing an object in the image you are observing. The image has a height of 480 and a width of 640. The image caption, objects in the image, and their respective bounding box coordinates are as follows:
Global Caption: The image depicts a cozy scene with a cat lying on top of a laptop computer, which is situated on a bed. The cat is in the center of the image, covering a significant portion of the laptop. The bed serves as the main background element, with the laptop and cat as the primary subjects of the scene.
Cat at [115.88, 271.59, 187.77, 336.81];
Laptop at [569.87, 67.5, 70.13, 195.33];
…
Coordinates represent (top-left x, top-left y, bottom-right x, bottom-right y). The question must involve at least two of these objects and be framed to require image reasoning for a response. Additional requirements for the generated question include:
1.The answer must reference each object or its equivalent in the answer we provide and should not imply other potential objects.
2.The question must not be too broad and must be meaningful.
3.The question should describe a complete activity, not just a combination of several sub-problems.
4.Rephrase the class name in the answer to indicate its location or shape.

Question: Can you identify the object in the picture that is typically used in office, and the item often used to provide a comfortable resting environment ?

Answer: A laptop under the cat and the bed.

Figure C.3. **Example of GPT-4 data generation.** The corresponding image is the same as in Fig. A.1.

These objects are used to construct questions in the format: `Please segment the <objects> in the image`, where `<objects>` represents a list of comma-separated object descriptors. The response format is a list of comma-separated `<object> is <SEG>`, with `<object>` being the description of each object. We require that the order of predictions in the answer matches the order of object names in the question and calculate gIoU and cIoU for each prediction based on this.

C.4. More Details about GPT-4 Generated Data.

In Sec. 5.5, we create 30,000 additional multi-target question-answer pairs to compare GPT-4 and GPT-4V. The prompts for GPT-4 adhere to similar generation principles but incorporate detailed image captions to offset the absence of visual content. Fig. C.3 demonstrates an example of our GPT-4 data generation process and its typical failures. Given that image content is not directly perceivable, conveying as detailed an image description as possible into GPT-4 is crucial to compensate for this information gap. However, this method often leads to a lack of diversity in the generated question-answer pairs (refer to the question-answer pair at the bottom of Fig. C.3), characterized by: *i*) Numerous questions are composed of simple referring style sub-questions; *ii*) The question and answer content is limited to what the image caption describes; *iii*) Challenges in generating detailed object descriptions. To address these issues, it might be necessary to introduce more complex annotation details like object relationships, which significantly increases the complexity and burden of data generation.