

# 基于 Grounding DINO 和 SAM 的开集目标检测与分割

## 摘要

本研究基于 Grounding DINO 开集检测模型和 Segment Anything Model (SAM) 分割模型，提出了一种自动化的图像物体检测与分割方法。Grounding DINO 能够通过人类输入的信息（如类别名称或指称表达）检测图像中的任意物体。Grounding DINO 模型包括三个关键阶段：特征增强器、语言引导的查询选择和跨模态解码器。特征增强器通过细致的特征提取增强模型的感知能力，而语言引导的查询选择则允许用户通过自然语言输入精确定位目标。最后，跨模态解码器将视觉信息与语言信息融合，确保检测结果的准确性与鲁棒性。基于这些检测框，SAM 能够从中提取出精确的物体分割掩膜。本研究通过结合图像与语言模型的优势，使得分割任务不仅能适应不同类型的物体，还能处理多样化的图像内容。我们的方法通过简单的指令输入，如文本描述带有分割物体的信息，即可生成目标物体的精确掩膜，从而大大降低了人工标注的工作量。将 Grounding DINO 和 SAM 模型结合后，我们的方法实现了从物体检测到精确分割的无缝对接，提升了从图像理解到分割执行的整体性能。可视化结果表明，本研究在多个实际应用场景下表现出了优越的效果，验证了该算法在物体检测与分割中的高效性与可靠性。

**关键词：**开集目标检测；图像分割；

## 1 引言

人工智能系统能力的关键指标之一是其处理开放世界场景的能力。在现实世界中，物体的种类、形态以及相互关系都是高度动态和多变的，这使得传统的目标检测方法在处理未知物体时存在显著的局限性。特别是在开放集目标检测中，系统不仅要识别已知类别的物体，还必须能够处理从未见过的新类别物体。这一挑战使得开放集目标检测成为当前计算机视觉领域中的一个重要研究方向。相比传统的封闭集目标检测，开放集检测的最大优势在于它能够在没有先验知识的情况下，自适应地扩展检测能力，极大地提升了系统在实际应用中的灵活性和适应性。

在这一背景下，本研究旨在开发一种强大的开放集目标检测与分割系统，能够基于人类语言输入自动检测并分割图像中的任意物体。这一任务不仅是技术上的挑战，更是实现通用目标检测器的关键一步。随着生成式模型和深度学习的飞速发展，开放集目标检测正逐渐成为图像处理和计算机视觉技术的核心能力之一。例如，在图像编辑应用中，用户可以通过自然语言描述指定的物体进行编辑（如图 1 所示），这种交互方式大大简化了操作流程，提高了

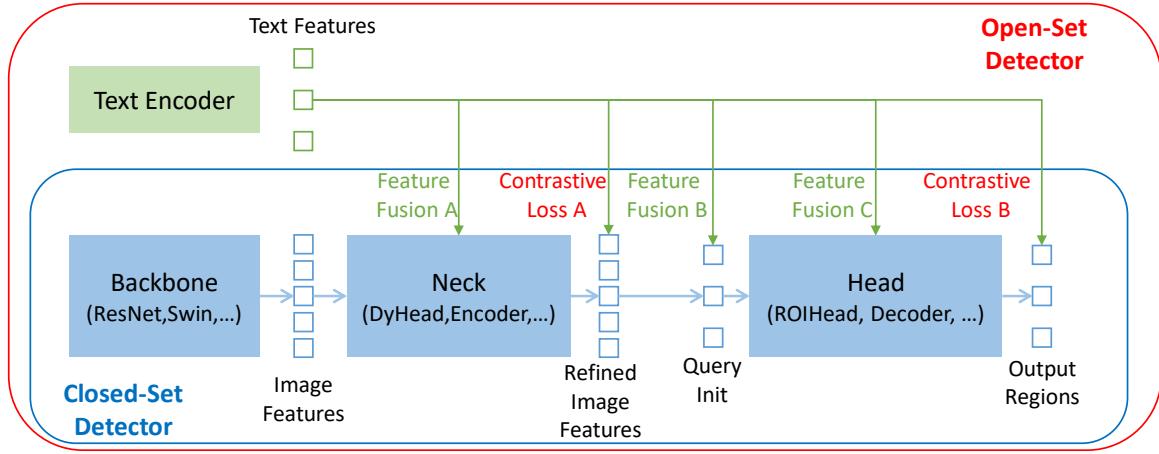


图 1. 将闭集检测器扩展到开集场景

用户体验。因此，开放集目标检测技术的成熟将为多个领域带来广泛的应用潜力，尤其是在自动化内容创作、增强现实和智能监控等领域。

为了实现这一目标，本研究结合了现有的开放集目标检测模型 Grounding DINO 和分割模型 Segment Anything Model (SAM)，提出了一种全新的自动化检测与分割算法。Grounding DINO 采用大规模语义预训练，使其具备强大的概念泛化能力，能够从人类语言输入（如类别名称或指称表达）中准确检测图像中的任意物体。Grounding DINO 模型的核心技术包括特征增强器、语言引导的查询选择以及跨模态解码器，这使得模型不仅能够高效地识别目标物体，还能够在复杂背景下定位物体，并精确地融合视觉与语言信息。

在此基础上，SAM 进一步提升了图像的编辑能力。SAM 可以基于检测框的输入，能够精确地从图像中提取目标物体的掩膜，进而实现物体的精细分割。SAM 的优势在于其高度的适应性和灵活性，它能够在没有大量标注数据的情况下执行高质量的图像分割任务。这一特点使得 SAM 与 Grounding DINO 的结合，形成了一个从物体检测到精细分割的完整工作流，显著提升了系统在开放世界中的表现。

我们通过简单的文本输入即可实现带有分割目标的物体描述，进一步简化了物体分割任务的交互过程，降低了人工标注的工作量。通过将 Grounding DINO 与 SAM 模型结合，我们的算法不仅能够适应多种不同类型的物体，还能够应对各种复杂的图像场景，极大提升了检测与分割的准确性和效率。实验结果表明，我们的算法在多个实际应用场景中表现出了优异的性能，能够在面对多样化的图像内容时，准确、高效地执行物体检测与分割任务。我们的研究为图像理解领域提供了一个强有力的新工具，并展示了开放集目标检测与分割技术在实际应用中的巨大潜力。

## 2 相关工作

### 2.1 传统的目标检测

传统的目标检测方法通常依赖于人工设计的特征和基于候选区域的方法。经典的 R-CNN 系列方法 [8, 9, 23] 首先提出了利用区域建议进行目标检测的方法。在这些方法中，首先生成可能包含目标的候选区域，然后对每个区域进行分类和边界框回归。Fast R-CNN [8] 改进了 R-CNN，通过共享卷积特征来加速检测过程，而 Faster R-CNN [23] 则进一步通过引入 Region

Proposal Network 自动化了候选区域的生成过程。尽管这些方法在精度上取得了较好的表现，但仍然依赖于预定义的类别和手工设计的特征，难以适应开放集目标检测任务。

近年来，基于卷积神经网络的方法，如 YOLO 系列 [1] 和 SSD [18]，通过将目标检测转化为回归问题，使得检测过程更加高效。YOLO 系列方法采用单一的网络结构，通过回归多个预测框和类别的参数实现端到端的目标检测。SSD 则通过在不同尺度的特征图上进行检测来提高对小物体的检测能力。尽管这些方法在速度和效率上取得了显著的进展，但它们同样面临着开放集检测的问题，无法应对未知类别的检测需求。

## 2.2 开集目标检测

**检测变换器 (Detection Transformers)** Grounding DINO 构建于类似 DETR 的模型 DINO [29] 之上（如图 2 所示），这是一个端到端基于 Transformer 的检测器。DETR 首次在 [2] 中提出，并且在过去几年中从多个方向得到了改进 [3,4,7,12,21,26,30]。DAB-DETR [17] 引入了锚框作为 DETR 查询，以提高框预测的准确性。DN-DETR [15] 提出了查询去噪方法，用于稳定二分匹配。DINO [29] 进一步发展了包括对比去噪等技术，并在 COCO 物体检测基准上创造了新的纪录。然而，这些检测器主要集中在封闭集检测上，且由于预定义类别的限制，难以泛化到新的类别。

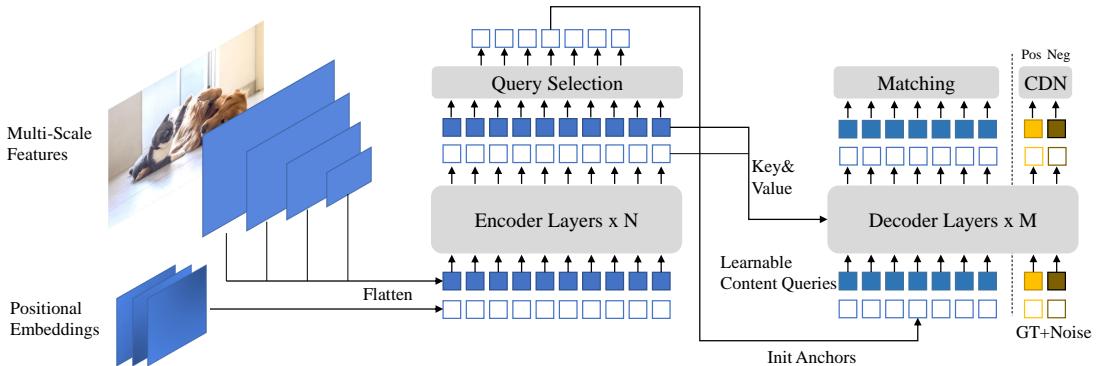


图 2. DINO 模型的结构图

**开放集物体检测 (Open-Set Object Detection)**。开放集物体检测通过现有的边界框标注进行训练，旨在利用语言泛化检测任意类别。OV-DETR [28] 使用由 CLIP 模型编码的图像和文本嵌入作为查询，以解码 DETR 框架中的类别特定边界框 [2]。ViLD [10] 从 CLIP 教师模型中提取知识，转换为类似 R-CNN 的检测器，使得学习到的区域嵌入包含语言语义。GLIP [6] 将物体检测视为一个定位问题，并利用额外的定位数据帮助学习在短语和区域层次上的语义对齐。研究表明，这种表述甚至可以在完全监督的检测基准上取得更强的性能。DetCLIP [27] 涉及大规模图像标注数据集，并使用生成的伪标签来扩展知识库。生成的伪标签有效地帮助扩展了泛化能力。

然而，先前的工作仅在部分阶段融合多模态信息，这可能导致语言泛化能力的次优。例如，GLIP 只在特征增强阶段（阶段 A）进行融合，而 OV-DETR 仅在解码器输入时注入语言信息（阶段 B）。此外，REC 任务通常在评估中被忽视，而该任务对于开放集检测来说是一个重要场景。为了提高模型的开放集能力，近年来一些方法尝试通过更深入的多模态融合，

提升语言与视觉特征的协同作用，例如通过在多个阶段同时融合图像与文本信息，来提高模型的泛化能力和识别能力。

### 2.3 图像分割

图像分割是计算机视觉中的一个基础任务，旨在将图像划分成多个有意义的区域或目标。早期的图像分割方法主要依赖于基于边缘检测和区域生长的传统算法，如 Canny 边缘检测和分水岭算法。这些方法在处理简单图像时表现良好，但在处理复杂背景或重叠目标时往往力不从心。随着深度学习的迅速发展，基于卷积神经网络的图像分割方法取得了显著进展。FCN（全卷积网络）[20]（如图 3）是最早的深度学习图像分割方法之一，它通过将全连接层替换为卷积层，使得网络能够接受任意尺寸的输入图像并进行像素级预测。在 FCN 之后，U-

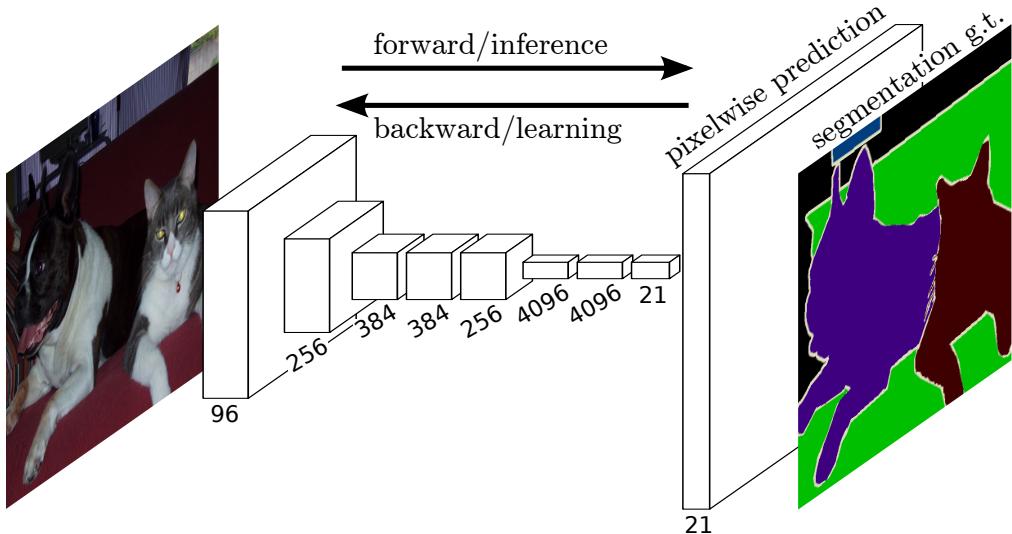


图 3. FCN 模型的结构图

Net [24] 作为一种用于医学图像分割的网络架构，通过引入跳跃连接，有效地提高了分割精度和效率。Mask R-CNN [11] 将目标检测与图像分割结合起来，通过生成物体的分割掩码，使得模型能够同时输出边界框和分割结果。近年来，基于 Transformers 的图像分割方法，例如 Swin Transformer [19] 和 Segmenter [25]，通过在局部和全局范围内建模图像的上下文信息，进一步提升了分割性能。

近年来，大规模视觉模型的出现进一步推动了图像分割的发展，代表性的模型包括 Segment Anything Model [14]。SAM 是一种通用的分割模型，能够在一次训练中实现对任意图像的分割。SAM 的核心思想是利用大规模预训练数据，通过引入一种“promptable segmentation”框架，使得用户可以通过点、框或者文本等多种提示方式对任意目标进行交互式分割。得益于大规模数据的泛化能力和多模态提示机制，SAM 在多种下游任务中表现出了卓越的性能。

尽管 SAM 等大模型极大地扩展了图像分割的应用范围，但其仍然面临一些挑战。例如，如何在高精度分割与计算资源消耗之间找到平衡，如何在弱监督或无监督场景下进一步提升性能，以及如何更有效地结合多模态信息（如文本和图像）来增强模型的泛化能力。未来的发展可能需要在数据质量、多模态融合、以及训练效率等方面继续探索，以实现更高效、更智能的分割模型。

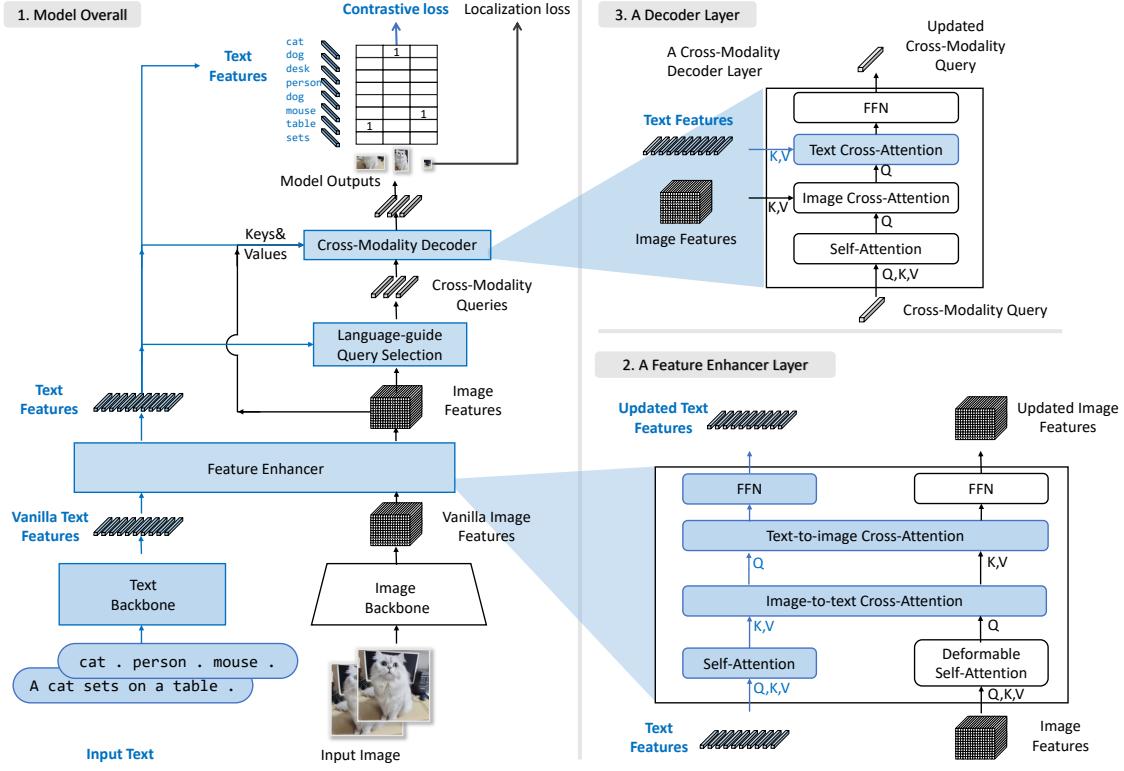


图 4. Grounding DINO 模型的结构图

### 3 本文方法

#### 3.1 本文方法概述

Grounding DINO 可以为给定的 (Image, Text) 对输出多个对象框和名词短语的配对。Grounding DINO 模型的结构如图 4 所示。模型从输入图像中定位到一只猫和一张桌子，并从输入文本中提取出词语 cat 和 table 作为对应的标签。对象检测和指向性表达理解任务都可以与该流程对齐。将所有类别名称拼接为对象检测任务的输入文本。表达理解任务要求每个文本输入对应一个边界框。将具有最高分数的输出对象作为最终的输出。

Grounding DINO 是一种双编码器-单解码器架构。它包括一个用于图像特征提取的图像主干网络、一个用于文本特征提取的文本主干网络、一个用于图像和文本特征融合的特征增强模块、一个用于查询初始化的语言引导查询选择模块以及一个用于框精细化的跨模态解码器。对于每个 (Image, Text) 对，首先通过图像主干网络和文本主干网络分别提取原始图像特征和原始文本特征。这两个原始特征被送入特征增强模块进行跨模态特征融合。在获得跨模态文本和图像特征后，使用语言引导查询选择模块从图像特征中选择跨模态查询。类似于大多数 DETR 类模型中的对象查询，这些跨模态查询将被送入跨模态解码器，从两个模态特征中提取所需特征并更新自身。最后解码器层的输出查询将用于预测对象框并提取对应的短语。

最后，得到的对象框还会被送入 SAM 中进行实例分割，SAM 接收 Grounding DINO 生成的边界框作为提示输入，针对每个边界框生成对应的精细分割掩码。这一过程可以有效将对象检测结果扩展为像素级的分割结果，使得检测更加细粒度并适用于更多下游任务。最终，Grounding DINO 的检测框与 SAM 的分割掩码相结合，能够实现从图像到文本的精准对齐以

及从边界框到像素级分割的全面覆盖。

### 3.2 Grounding DINO 特征提取模块模块

对于给定的 (Image, Text) 对，使用类似于 Swin Transformer [19] 的图像主干网络提取多尺度图像特征，并使用类似于 BERT [5] 的文本主干网络提取文本特征。参考以往的 DETR 类检测器 [29,30]，多尺度特征从不同层的输出中提取。在提取出原始图像特征和文本特征后，将它们送入特征增强模块进行跨模态特征融合。特征增强模块由多个特征增强层组成。在图 4 的第 2 块中展示了一个特征增强层的结构。

在特征增强过程中，利用可变形自注意力增强图像特征，并对文本特征增强器使用原始自注意力。受 GLIP [16] 的启发，为特征融合添加了图像到文本和文本到图像的交叉注意力模块。这些模块有助于对齐不同模态的特征。

### 3.3 Grounding DINO 语言引导查询选择

Grounding DINO 旨在通过输入文本指定的目标检测图像中的对象。为了有效利用输入文本引导对象检测，设计了一个语言引导查询选择模块，从中选择与输入文本更相关的特征作为解码器查询。

语言引导查询选择模块输出  $N_q$  个索引。可以基于选定的索引提取特征来初始化查询。参考 DINO [29]，采用混合查询选择来初始化解码器查询。每个解码器查询由两部分组成：内容部分和位置部分 [21]。其中，位置部分被表述为动态锚框 [17]，并通过编码器输出初始化。另一部分，即内容查询，则在训练过程中设置为可学习参数。

### 3.4 Grounding DINO 跨模态解码器模块

Grounding DINO 还开发了一个跨模态解码器，用于结合图像和文本模态特征，如图 4 中的模块 3 所示。每个跨模态查询会依次通过一个自注意力层、一个图像交叉注意力层以结合图像特征、一个文本交叉注意力层以结合文本特征，以及一个 FFN 层。这些组成部分构成了每一层跨模态解码器。与 DINO 解码器层相比，每层解码器额外添加了一个文本交叉注意力层，这是因为需要向查询中注入文本信息以实现更好的模态对齐。在先前的研究中，探索了两种文本提示方式，将其称为句子级表示和单词级表示。句子级表示 [22,27] 将整句话编码为一个特征。如果短语标注数据中的某些句子包含多个短语，句子级表示会提取这些短语并丢弃其他词汇。这种方式可以消除单词之间的影响，但同时也会丢失句子中的细粒度信息。

单词级表示 [6,13] 则允许通过一次前向传递对多个类别名称进行编码，但这种方式会引入类别之间不必要的依赖关系，特别是在输入文本是多个类别名称以任意顺序拼接时。在注意力机制中，一些不相关的词语可能会发生交互。为了避免这种不必要的单词交互，引入了注意力掩码，以阻止不相关类别名称之间的注意力交互，这种方法被称为“子句”级表示。该方法消除了不同类别名称之间的影响，同时保留每个词语的特征以实现细粒度的理解。

### 3.5 SAM 分割

将输入的图像和边界框提示分别送入 SAM 的图像编码器和提示编码器，如图 5 所示。图像经过标准化预处理后，被输入基于 ViT 架构的图像编码器，提取全局特征表示。图像被

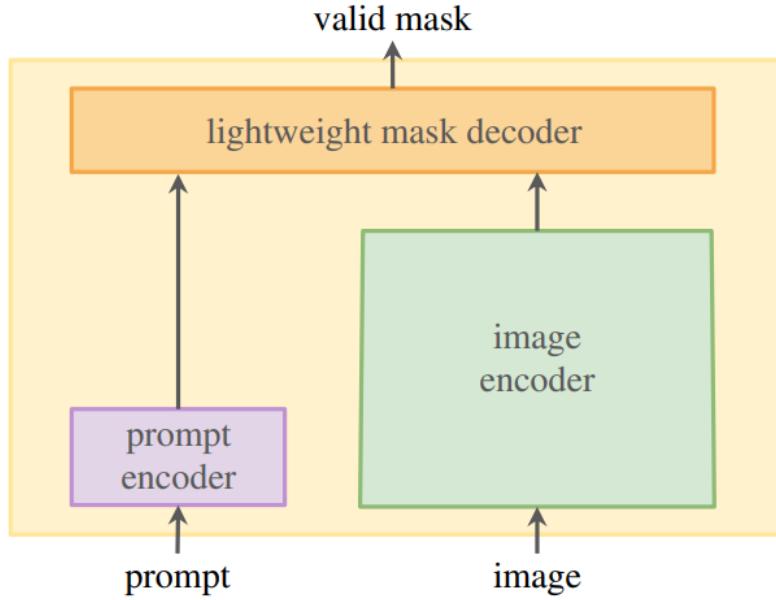


图 5. SAM 的流程图

分解为一系列固定大小的图像块，每个块通过自注意力机制与其他块交互，生成高维特征图  $\mathbf{F}_{\text{img}} \in \mathbb{R}^{H \times W \times d}$ ，其中  $H \times W$  为特征图的空间分辨率， $d$  为特征维度。同时，边界框提示作为位置信息被送入提示编码器，并被编码为高维特征向量  $\mathbf{F}_{\text{prompt}}$ ，与图像特征图兼容。在掩码解码器中，图像特征  $\mathbf{F}_{\text{img}}$  和提示特征  $\mathbf{F}_{\text{prompt}}$  通过交叉注意力机制融合，确保提示特征能够准确定位图像中的目标区域。随后，掩码解码器将融合后的特征投影到像素空间，生成分割掩码  $\mathbf{M} \in \mathbb{R}^{H \times W}$ 。SAM 输出的掩码以二值化形式表示目标的像素位置，其分辨率与输入图像一致。在与 Grounding DINO 集成的应用中，SAM 针对每个由 Grounding DINO 提供的边界框分别生成分割掩码，实现从边界框到像素级目标表示的转换，有效提升了分割结果的细粒度表现，并为复杂的下游视觉任务提供了更强的支持。

## 4 复现细节

本研究的算法流程如图 6 所示。首先，输入图像和相应的文本描述被传递给 Grounding DINO 模型，该模型利用大规模预训练的语义信息从图像中检测出目标物体的位置。接着，基于检测到的物体框，SAM 模型进一步对这些物体进行精确的分割，生成目标物体的掩膜。整个过程通过简洁的指令输入即可完成，从物体的检测到细粒度的分割，确保了算法的高效性与准确性。

### 4.1 实验环境搭建

为了复现本研究的方法，以下是我们搭建的实验环境：

**硬件环境：**

操作系统：Windows 11 GPU：NVIDIA RTX 3050 (8GB VRAM)；CPU：AMD 5600g；内存：16GB DDR4。

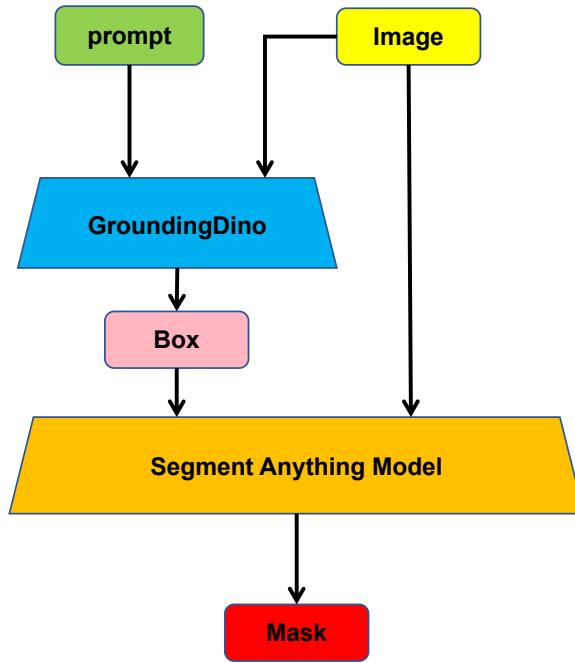


图 6. 本研究方法流程图

**软件环境:** Python 3.10.6; PyTorch 2.0.1+cu118; CUDA 12.7; TorchVision 0.15.2+cu118; Transformers 4.46.3; Matplotlib 3.4.3; Opencv-python 4.10.0.84。

**依赖库:** 确保安装以下依赖库：

- PyTorch：用于深度学习模型的训练与推理。
- Transformers：用于加载预训练的 Grounding DINO 模型。
- NumPy：用于矩阵计算和数据操作。
- Matplotlib：用于生成实验结果的可视化图表。

**预训练模型下载:**

- [Grounding DINO 预训练模型下载链接](#)

将下载的权重文件放到./GroundingDINO 下。

- [SAM 预训练模型下载链接](#)

将下载的权重文件放到./segment-anything 下。

## 4.2 复现代码细节说明

本报告中所展示的推理和可视化界面的代码都为本人撰写，并不在开源代码中。

#### 4.2.1 Grounding Dino 得到检测框

在该部分，我们使用 Grounding DINO 模型来获取输入图像中物体的检测框。首先，使用给定的文本提示对模型进行推理，生成与提示相关的边界框。在代码中，get\_boxes 函数是执行这一任务的核心，首先加载图像并对文本提示进行处理。文本提示被分割成多个子提示，然后依次传入模型进行预测。每个提示都会返回相关的边界框坐标、置信度分数以及物体的标签。通过设定合适的阈值（如 BOX\_THRESHOLD 和 TEXT\_THRESHOLD），我们能够控制框的筛选标准，确保返回的框是模型认为最相关的物体区域。最终，get\_boxes 函数返回所有检测到的边界框坐标、相应的置信度和物体的描述标签。

```
1 def load_GD_model(config_path, model_path):
2     model = load_model(config_path, model_path)
3     return model
4
5 def get_boxes(image_path, prompts, model):
6     _, image = load_image(image_path)
7     prompt_list = prompts.split(".")
8     boxes_list = []
9     logits_list = []
10    phrases_list = []
11    for prompt in prompt_list:
12        boxes, logits, phrases = predict(
13            model=model,
14            image=image,
15            caption=prompt,
16            box_threshold=BOX_THRESHOLD,
17            text_threshold=TEXT_THRESHOLD
18        )
19        for box in boxes:
20            boxes_list.append(box.tolist())
21        for logit in logits:
22            logits_list.append(logit.tolist())
23        for phrase in phrases:
24            phrases_list.append(phrase)
25    return boxes_list, logits_list, phrases_list
```

Listing 1: Grounding Dino 检测的代码

#### 4.2.2 SAM 得到分割掩膜

在获得物体的边界框后，接下来通过 SAM 模型来生成与边界框对应的分割掩膜。在代码中，get\_masks 函数首先使用 Grounding DINO 得到边界框列表，然后加载 SAM 模型并根据边界框信息进行掩膜预测。SamPredictor 负责加载模型和进行预测。通过输入的边界框信息，SAM 会生成图像中对应区域的掩膜，这些掩膜可以用来精确地分割出图像中的物体。为了确保掩膜能够准确地覆盖到物体区域，我们对边界框进行了坐标缩放，使得框的坐标符合图像的实际尺寸。每个生成的掩膜被绘制在原始图像上，最终返回包含边界框和掩膜的图像。

```
1
2 def load_sam_img(image_path):
```

```

3
4     image = cv2.imread(image_path) # 读取的图像以NumPy数组的形式存储在变量image
    中
5     image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # 将图像从BGR颜色空间转换为
    RGB颜色空间，还原图片色彩（图像处理库所认同的格式）
6     return image
7 def load_sam_model(sam_checkpoint, model_type):
8     sam_model = sam_model_registry[model_type](checkpoint=sam_checkpoint)
9     sam_model.to(device="cuda:0")
10    predictor = SamPredictor(sam_model)
11    return predictor
12 def get_masks(image_path, prompts):
13     #得到box
14     config_path = "GroundingDINO/groundingdino/config/GroundingDINO_SwinT_OGC.py
        "
15     model_path = "GroundingDINO/groundingdino_swint_ogc.pth"
16     GD_model = load_GD_model(config_path, model_path)
17     print("GD_model\u2014loaded")
18     boxes_list, logits_list, phrases_list = get_boxes(image_path, prompts,
19             GD_model)
20     # 完成后释放 GD_model
21     del GD_model # 删除模型对象
22     torch.cuda.empty_cache() # 清空CUDA缓存（如果使用GPU）

23     print("GD_model\u2014released")
24     #得到mask
25     sam_checkpoint = "segment-anything/sam_vit_h_4b8939.pth" # 定义模型路径
26     model_type = "vit_h" # 定义模型类型
27     predictor = load_sam_model(sam_checkpoint, model_type)
28     print("Sam_model\u2014loaded")
29     image = load_sam_img(image_path)
30     if len(boxes_list) == 0:
31         return Image.fromarray(image)
32     h, w, _ = image.shape
33     for i in range(len(boxes_list)):
34         coordinates = torch.tensor(boxes_list[i])
35         label = phrases_list[i]
36         box = coordinates * torch.Tensor([w, h, w, h]) # Scale the coordinates
            to the image size
37         input_box = box_convert(boxes=box, in_fmt="cxcywh", out_fmt="xyxy").
            numpy()

38         predictor.set_image(image)
39         masks, _, _ = predictor.predict(
40             point_coords=None,
41             point_labels=None,
42             box=input_box[None, :, :],
43             multimask_output=False,
44

```

```

45
46
47     mask = masks[0] # 获取第一个 mask
48
49     # 将 mask 绘制到原图上
50     image = draw_mask_and_box_on_image(image, mask, input_box, label)
51
52     # 你可以选择保存或显示这个带有 mask 的图像
53
54
55     del predictor # 删除模型对象
56     torch.cuda.empty_cache() # 清空 CUDA 缓存 (如果使用 GPU)
57     print("Sam_model released")
58     return Image.fromarray(image)

```

Listing 2: SAM 分割的代码

#### 4.2.3 基于 tkinter 的可视化界面

为了提高用户交互体验，我们在程序中集成了 tkinter 库，构建了一个简易的可视化界面。这个界面允许用户上传图片并输入提示文本，然后调用前述的 Grounding DINO 和 SAM 模型进行图像处理。在界面上，用户可以点击按钮选择要处理的图像文件，并在文本框中输入描述图像的文本提示。当用户点击“处理”按钮时，图像和提示将被传递给后台的处理函数，经过模型的推理后，生成带有边界框和分割掩膜的图像。处理后的图像会实时显示在界面上，用户可以选择将结果保存到本地。这个界面使得模型的使用更加直观和易于操作，特别适合非技术用户使用。

```

1 import tkinter as tk
2 from tkinter import filedialog
3 from PIL import Image, ImageTk
4 from utils import process_image
5
6 # 初始化全局变量
7 image1_path = None
8
9 def upload_image1():
10     """上传图片并显示"""
11     global image1_path
12     file_path = filedialog.askopenfilename()
13     if file_path:
14         img = Image.open(file_path)
15
16         # 调整图片大小，保持最大尺寸为512x512
17         img.thumbnail((512, 512), Image.Resampling.LANCZOS)
18
19         # 创建Tkinter兼容格式
20         img_tk = ImageTk.PhotoImage(img)
21

```

```

22     # 更新图片显示
23     label_img1.config(image=img_tk)
24     label_img1.image = img_tk  # 保存引用
25
26     # 存储图片路径以便后续处理
27     image1_path = file_path
28
29 def process_and_display():
30     """处理图像并显示结果"""
31     global image1_path
32     if image1_path:
33         # 从文本框获取用户输入
34         prompt = entry_text.get()
35         processed_img = process_image(image1_path, prompt)  # 传递文本到处理函数
36
37         # 确保处理后的图片大小为512x512
38         processed_img.thumbnail((512, 512), Image.Resampling.LANCZOS)
39
40         # 转换为Tkinter图像格式
41         img_tk = ImageTk.PhotoImage(processed_img)
42
43         # 更新处理后的图片显示
44         label_img2.config(image=img_tk)
45         label_img2.image = img_tk  # 保存引用
46
47 def exit_program():
48     """退出程序"""
49     root.quit()
50     root.destroy()
51
52 def setup_window():
53     """设置窗口的大小、位置及其他基本属性"""
54     root = tk.Tk()
55     root.title("GroundingDino_SAM_Process_Image")
56
57     # 设置窗口大小并居中显示
58     window_width = 1200
59     window_height = 800
60     screen_width = root.winfo_screenwidth()
61     screen_height = root.winfo_screenheight()
62
63     # 计算窗口的初始位置，使其居中
64     x = (screen_width - window_width) // 2
65     y = (screen_height - window_height) // 2
66     root.geometry(f"{window_width}x{window_height}+{x}+{y}")
67
68     # 设置窗口的最小尺寸
69     root.minsize(600, 400)

```

```

70
71     return root
72
73 def setup_widgets(root):
74     """设置窗口中的所有组件"""
75     # 添加退出按钮
76     button_exit = tk.Button(root, text="Exit", command=exit_program, width=10)
77     button_exit.grid(row=0, column=0, padx=10, pady=10, sticky="nw")
78
79     # 图片展示框1
80     label_img1 = tk.Label(root, text="UploadedImage1")
81     label_img1.grid(row=0, column=0, padx=10, pady=10)
82
83     # 图片展示框2
84     label_img2 = tk.Label(root, text="ProcessedImage")
85     label_img2.grid(row=0, column=1, padx=10, pady=10)
86
87     # 上传按钮
88     button_upload = tk.Button(root, text="UploadImage", command=upload_image1,
89                               width=20)
90     button_upload.grid(row=1, column=0, pady=10, padx=10, sticky="ew")
91
92     # 文本输入框
93     entry_text = tk.Entry(root, width=20)
94     entry_text.grid(row=1, column=1, pady=10, padx=10, sticky="ew")
95     entry_text.insert(0, "EnterTextHere") # 提示默认文本
96
97     # 处理按钮
98     button_process = tk.Button(root, text="ProcessedImage", command=
99                               process_and_display, width=20)
100    button_process.grid(row=2, column=0, columnspan=2, pady=20, padx=10, sticky=
101                          "ew")
102
103    # 配置列和行的权重，确保组件的大小可以调整
104    root.grid_rowconfigure(0, weight=1)
105    root.grid_rowconfigure(1, weight=0)
106    root.grid_rowconfigure(2, weight=0)
107    root.grid_columnconfigure(0, weight=1)
108    root.grid_columnconfigure(1, weight=1)
109
110   # 将 `label_img1` 和 `label_img2` 引用传递到函数中
111   return label_img1, label_img2, entry_text
112
113 def main():
114     """主函数"""
115     global label_img1, label_img2, entry_text, root
116
117     # 设置窗口

```

```

115 root = setup_window()
116
117 # 设置组件
118 label_img1, label_img2, entry_text = setup_widgets(root)
119
120 # 运行Tkinter事件循环
121 root.mainloop()
122
123 if __name__ == '__main__':
124     main()

```

Listing 3: 交互页面代码

### 4.3 界面分析与使用说明

可视化界面设计简洁直观，主要分为左侧的原始图像展示和右侧的结果展示区域。用户首先通过界面底部的“Upload Image”按钮上传一张图像，上传后的图像会实时显示在左侧展示区域。同时，用户在右侧的输入框中输入文本提示词（例如“dog”，如图 7 所示），然后点击“Process Image”按钮触发图像处理流程。在处理完成后，右侧展示区域会显示根据提示词生成的检测结果，包括目标物体的边界框和分割掩膜，如图 8 所示。若用户需要重新操作，可随时更改文本提示词或重新上传图像，以实现灵活的交互功能。

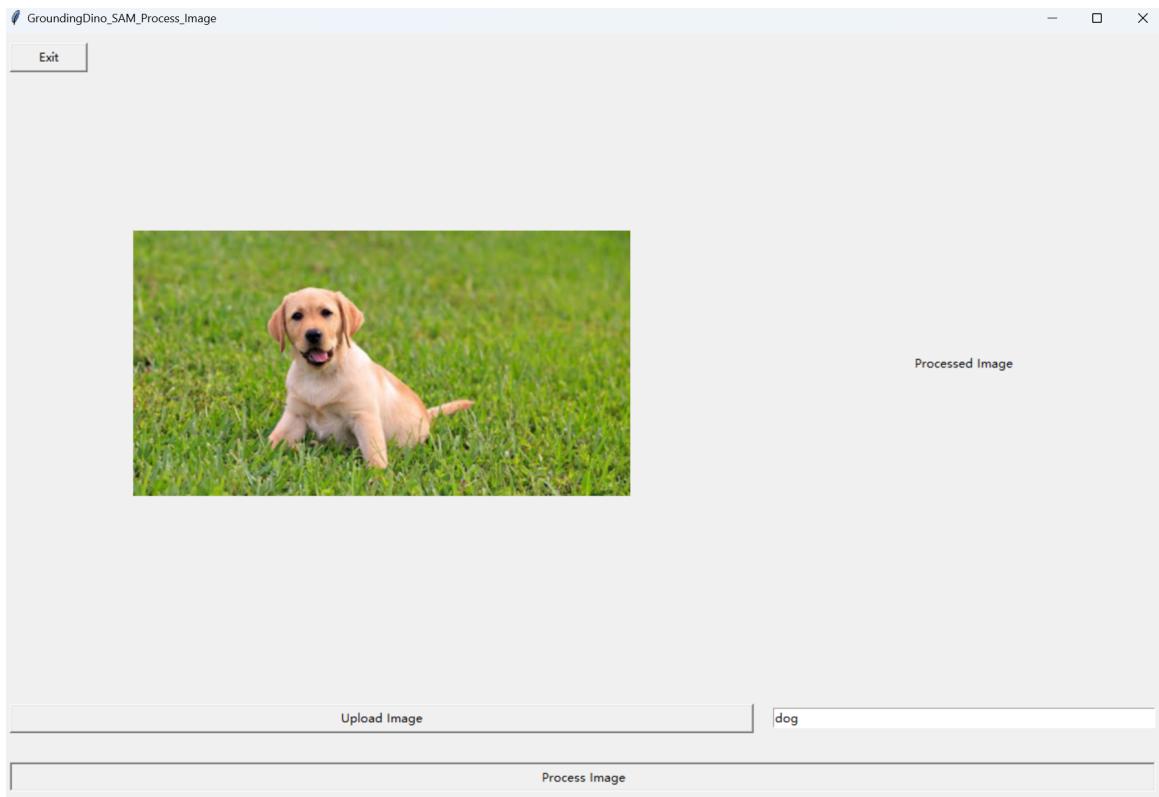


图 7. 初始可视化界面

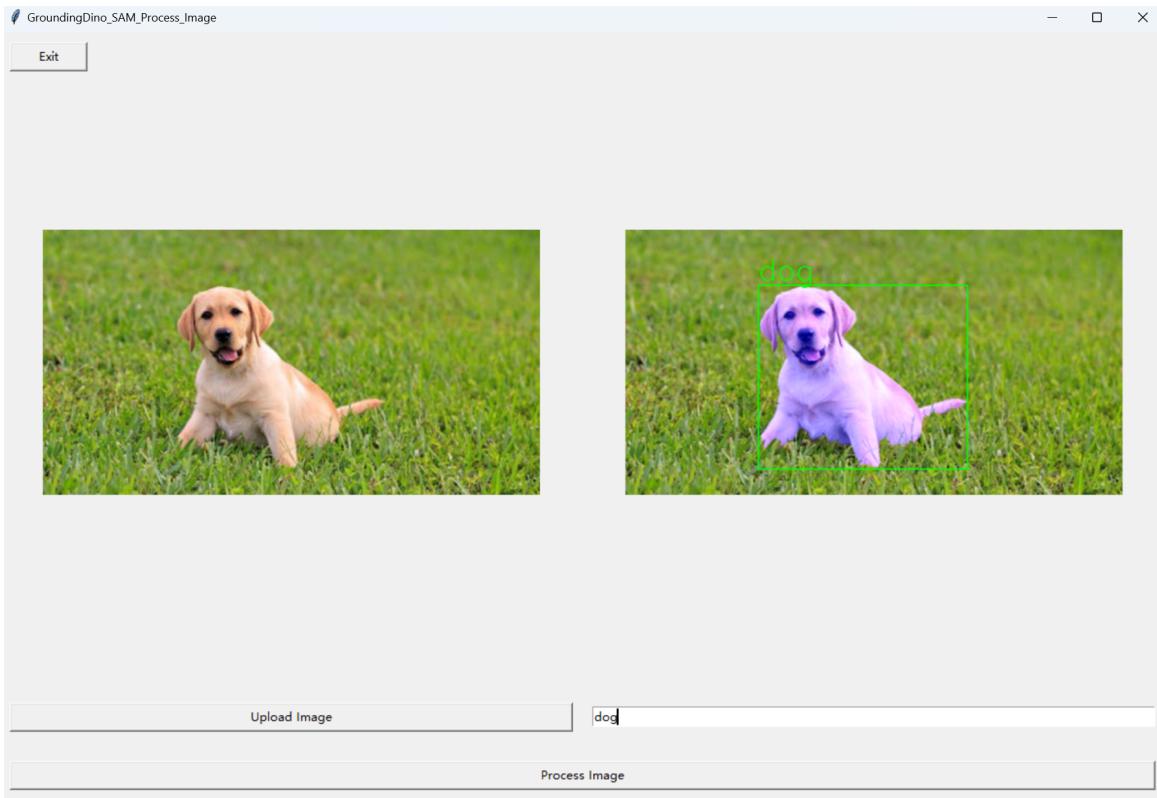


图 8. 结果可视化界面

#### 4.4 创新点

基于 Grounding DINO 和 SAM 的开集目标检测与分割有以下几个创新点：

- 本研究结合了 Grounding DINO 和 SAM 两个强大的视觉模型，其中 Grounding DINO 能够根据文本提示自动生成检测框，而 SAM 则用于进一步生成精确的分割掩膜。通过这种方式，实现了从文本描述到物体检测和精细分割的全过程。这种多模态的联动不仅提升了图像理解的能力，还增强了文本引导的准确性，弥补了传统图像分割和检测方法仅依赖视觉信息的局限性。
- 通过构建基于 tkinter 的用户界面，用户能够非常简便地上传图像、输入提示文本并获取处理后的图像。这种设计使得复杂的模型操作变得极其简洁和直观，尤其对于非技术背景的用户来说，这样的工具提供了更好的可用性和灵活性。它不仅可以作为一个研究工具，还能够在实际应用中为临床医生、图像处理专家等提供便捷的辅助工具，极大地降低了模型的使用门槛。
- 将 Grounding DINO 和 SAM 两种不同类型的模型进行协同使用。Grounding DINO 是一个基于文本提示进行物体检测的模型，而 SAM 是一个基于图像和边界框生成掩膜的模型。将这两者结合起来，不仅提升了物体检测的精准度，还能够对物体进行精细的像素级分割，填补了当前基于文本和边界框的模型在细粒度分割上的空白。这种跨模型的组合，体现了多任务、多目标的处理能力。

## 5 实验结果分析

### 5.1 单类别单物体

在单类别单物体的实验中，我们选取了多个图像样本，其中每张图像仅包含一个明确类别的目标物体。通过模型的推理，可以发现本项目中使用的检测与分割方法能够稳定地定位并识别目标物体，无论目标物体的大小、位置还是背景复杂度如何，检测框和分割掩膜均与目标物体高度匹配。从实验结果可以看出，Grounding Dino 生成的检测框精确覆盖了目标物体的边界，而 SAM 进一步生成的分割掩膜细化了物体的轮廓，边界清晰且无明显漏检或误检的情况，表现出较强的单类别单物体检测与分割能力。

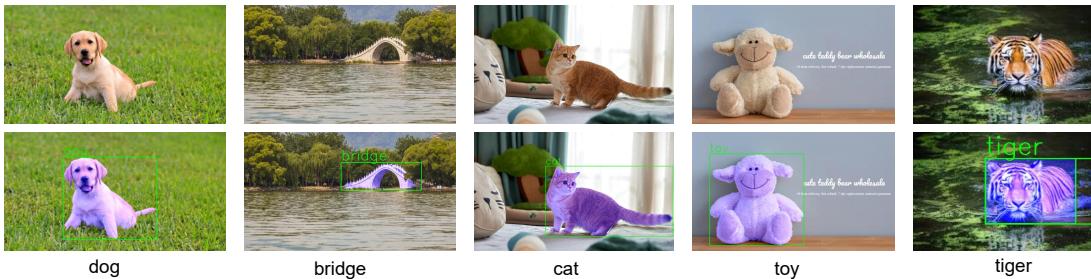


图 9. 单类别单物体的可视化结果

### 5.2 单类别多物体

在单类别多物体的实验中，我们测试了包含多个同类目标物体的图像样本。实验结果显示，模型能够同时检测并分割场景中的所有同类物体，无论它们在空间中的分布是否密集，或是存在部分遮挡的情况下，模型依然能够保证较高的检测与分割准确率。从定量分析结果来看，检测框与分割掩膜在物体的完整性和边界拟合上表现优秀，几乎所有同类别目标均被正确识别并分割，且不存在重复标注或遗漏现象。特别是在目标物体形状不规则或具有复杂边缘的情况下，SAM 生成的分割掩膜仍然保持了较高的细节还原能力，证明了其在单类别多物体场景中的稳定性。

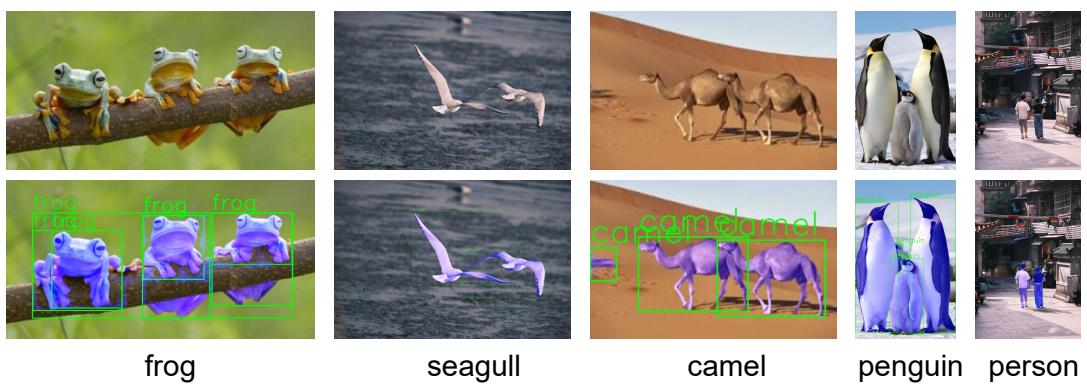


图 10. 单类别多物体的可视化结果

### 5.3 多类别单物体

在多类别单物体的实验中，我们测试了每张图像中包含一个目标物体，但该目标物体属于多个不同的类别。该实验的主要挑战在于如何精确认识别目标物体的类别，同时保持高质量的分割精度。实验结果表明，模型能够有效区分并识别同一物体的不同类别，且在物体的分割上表现出了很好的性能。尽管物体的外观可能由于不同类别的标签而有所不同，SAM 生成的分割掩膜仍能够准确覆盖目标区域，并且在类别判别上没有混淆现象。

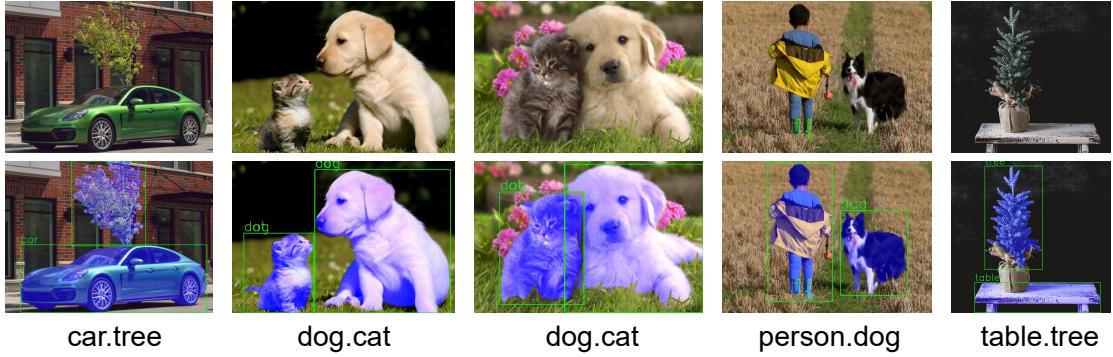


图 11. 多类别单物体的可视化结果

### 5.4 多类别多物体

多类别多物体的实验是对模型性能的进一步考验，图像中同时包含多个类别的目标物体，每个类别可能包含多个实例。实验结果表明，系统在多类别多物体场景中依然表现出色。Grounding Dino 能够准确区分不同类别的目标，并为每个目标生成独立的检测框，同时 SAM 生成的分割掩膜也能够精准覆盖目标区域，并对不同类别进行了清晰的分割。在测试样本中，即使存在物体遮挡或类别边界模糊的情况，模型仍然能够正确识别目标类别，且分割掩膜边界精确、无明显重叠。此外，在某些复杂场景中，例如物体类别多达五种以上的情况下，模型依然保持了较高的检测与分割准确率，充分证明了本系统在多类别多物体场景下的强大泛化能力和实际应用潜力。

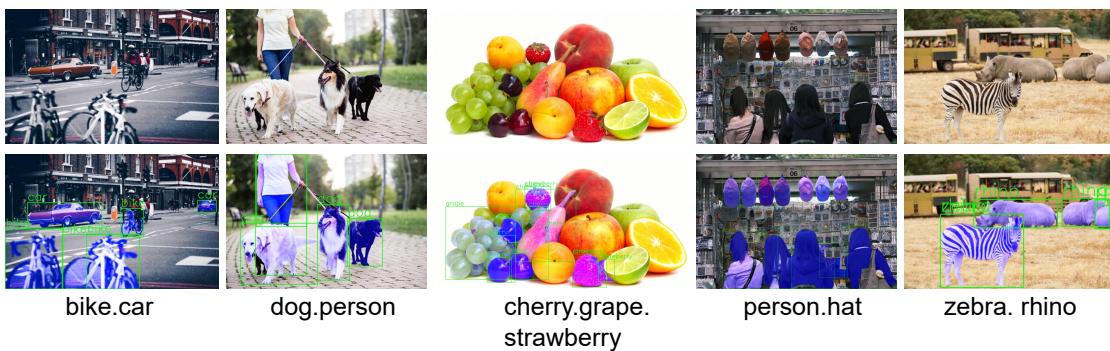


图 12. 多类别多物体的可视化结果

## 6 总结与展望

本研究提出了一种基于 Grounding DINO 和 SAM 的自动化图像物体检测与分割算法，旨在解决开放集目标检测与精细分割任务中的挑战。通过结合开放集检测技术与先进的分割模型，我们的算法能够根据人类语言输入自动检测并精确分割图像中的任意物体，显著提升了检测与分割的灵活性与适应性。Grounding DINO 模型通过大规模语义预训练，具备了强大的概念泛化能力，能够准确识别和定位未知物体；而 SAM 模型则在此基础上，通过高效的分割能力将物体识别扩展到精细的分割任务。所提出的算法在多个实际应用场景中均表现出了优异的性能，能够处理各种复杂背景和多样化的物体类型。特别是在开放集场景中，模型展现了较强的适应能力和鲁棒性，能够应对未知类别的物体并准确进行检测与分割。此外，通过简单的语言指令输入，系统能够快速生成目标物体的精确掩膜，大大减少了人工标注的工作量，为实际应用提供了高效的解决方案。本研究的成果不仅在目标检测与分割领域具有重要的学术价值，还为图像编辑、智能监控、自动驾驶等领域提供了有力的技术支持。未来的研究可以进一步优化模型的精度和效率，探索如何在更复杂的开放世界场景中实现更广泛的应用，同时，结合更多的多模态输入和跨领域数据，也有望进一步提升算法的适应性和泛化能力。

## 参考文献

- [1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
- [2] 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.
- [3] Qiang Chen, Xiaokang Chen, Jian Wang, Haocheng Feng, Junyu Han, Errui Ding, Gang Zeng, and Jingdong Wang. Group DETR: Fast detr training with group-wise one-to-many assignment. 2022.
- [4] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7373–7382, 2021.
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [6] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv preprint arXiv:2110.04544*, 2021.

- [7] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of DETR with spatially modulated co-attention. *arXiv preprint arXiv:2101.07448*, 2021.
- [8] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.
- [9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [10] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. *Learning*, 2021.
- [11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
- [12] Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. DETRs with hybrid matching. 2022.
- [13] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. MDETR-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021.
- [14] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.
- [15] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. DN-DETR: Accelerate DETR training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13619–13627, 2022.
- [16] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. *arXiv preprint arXiv:2112.03857*, 2021.
- [17] 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. In *International Conference on Learning Representations*, 2022.
- [18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

- [19] 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. *arXiv preprint arXiv:2103.14030*, 2021.
- [20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [21] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional DETR for fast training convergence. *arXiv preprint arXiv:2108.06152*, 2021.
- [22] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. Simple open-vocabulary object detection with vision transformers. 2022.
- [23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.
- [24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.
- [25] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021.
- [26] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor DETR: Query design for transformer-based detector. *national conference on artificial intelligence*, 2021.
- [27] Lewei Yao, Jianhua Han, Youpeng Wen, Xiaodan Liang, Dan Xu, Wei Zhang, Zhengu Li, Chunjing Xu, and Hang Xu. Detclip: Dictionary-enriched visual-concept paralleled pre-training for open-world detection. 2022.
- [28] Alireza Zareian, Kevin Dela Rosa, Derek Hao Hu, and Shih-Fu Chang. Open-vocabulary object detection using captions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14393–14402, 2021.
- [29] 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, 2022.

- [30] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR 2021: The Ninth International Conference on Learning Representations*, 2021.