YOLOE：实时看清一切

王鹜 [1*] 刘力豪 [1*] 陈辉 [1] 林子家 [1] 韩军光 [1] 丁贵广 [1] [1] 清华大学

摘要

目标检测和分割在计算机视觉应用中被广泛使用，然而像YOLO系列这样的传统模型，虽然高效且准确，但受到预定义类别的限制，在开放场景中的适应性较差。最近的开放集方法利用文本提示、视觉线索或无提示范式来克服这一问题，但由于高计算需求或部署复杂性，往往在性能和效率之间做出妥协。在这项工作中，我们引入了YOLOE，它在一个高效的模型中集成了多种开放提示机制的检测和分割，实现了实时看到任何东西。对于文本提示，我们提出了可重新参数化区域-文本对齐（RepRTA）策略。它通过一个可重新参数化的轻量级辅助网络来细化预训练的文本嵌入，并以零推理和转移开销增强视觉-文本对齐。对于视觉提示，我们提出了语义激活视觉提示编码器（SAVPE）。它采用解耦的语义和激活分支，以最小的复杂度带来改进的视觉嵌入和准确性。对于无提示场景，我们引入了惰性区域-提示对比（LRPC）策略。它利用内置的大词汇表和专门的嵌入来识别所有对象，避免了对昂贵语言模型的依赖。大量实验表明，YOLOE具有出色的零样本性能和可转移性，推理效率高，训练成本低。值得注意的是，在LVIS上，YOLOE-v8-S的训练成本降低了3 $\times$，推理速度提高了1.4 $\times$，比YOLO-Worldv2-S的平均精度（AP）高出3.5。当转移到COCO时，YOLOE-v8-L比封闭集的YOLOv8-L的平均精度（AP）提高了0.6 ^{b}，训练时间减少了近 4$\times$，增益为0.4 AP^m。代码和模型可在https://github.com/THU-MIG/yoloe获得。

1. 引言

目标检测和分割是计算机视觉中的基础任务[15, 48]，具有广泛的应用，涵盖自动驾驶[2]、医学分析[55]和机器人技术[8]等。

*共同贡献。

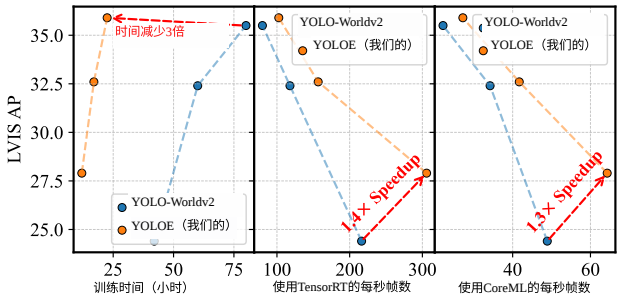


图1. YOLOE（我们的方法）与先进的YOLO-Worldv2在开放文本提示方面的性能、训练成本和推理效率比较。LVIS AP在小验证集上评估，使用TensorRT和CoreML的FPS分别在T4 GPU和iPhone 12上测量。结果突出了我们的优势。

像YOLO系列 [1, 3, 21, 47] 这样的传统方法，利用卷积神经网络实现了卓越的实时性能。然而，它们对预定义对象类别的依赖限制了实际开放场景中的灵活性。这种场景越来越需要能够在各种提示机制（如文本、视觉线索或无提示）的引导下检测和分割任意对象的模型。

鉴于此，最近的研究工作已转向使模型能够对开放提示进行泛化[5, 20, 49, 80]。它们针对单一提示类型，例如GLIP[32]，或以统一方式针对多种提示类型，例如DINO-X[49]。具体而言，通过区域级视觉语言预训练[32, 37, 65]，文本提示通常由文本编码器处理，以作为区域特征的对比目标[20, 49]，从而实现对任意类别的识别，例如YOLO-World[5]。对于视觉提示，它们通常被编码为与指定区域相关联的类别嵌入，以便通过与图像特征或语言对齐的视觉编码器[5, 19, 30, 49]进行交互来识别相似对象，例如T-Rex2[20]。在无提示场景中，现有方法通常集成语言模型，依次根据区域特征找到所有对象并生成相应的类别名称[49, 62]，例如GenerateU[33]。

尽管取得了显著进展，但仍缺乏一个能够高效、准确地支持对任意对象的多种开放提示的单一模型。

例如，DINO-X [49] 具有统一架构，然而，这会带来资源密集型的训练和推理开销。此外，不同作品中针对不同提示的单独设计在性能和效率之间表现出次优的权衡，使得难以直接将它们组合成一个模型。例如，由于跨模态融合的复杂性 [5, 32, 37, 49]，文本提示方法在纳入大词汇表时通常会产生大量计算开销。视觉提示方法通常由于重型变压器设计或对额外视觉编码器的依赖 [20, 30, 67] 而损害了在边缘设备上的可部署性。同时，无提示方法依赖于大语言模型，带来了相当大的内存和延迟成本 [33, 49]。

鉴于这些情况，在本文中，我们介绍了YOLOE(ye)，这是一种高效、统一且开放的目标检测与分割模型，类似于人眼，适用于不同的提示机制，如文本、视觉输入和无提示范式。我们从已被广泛证明有效的YOLO模型开始。对于文本提示，我们提出了一种可重新参数化的区域-文本对齐（RepRTA）策略，该策略采用一个轻量级辅助网络来改进预训练的文本嵌入，以实现更好的视觉-语义对齐。在训练期间，预缓存的文本嵌入仅需辅助网络来处理文本提示，与封闭集训练相比，额外成本较低。在推理和迁移时，辅助网络无缝地重新参数化为分类头，产生一个与YOLO模型架构相同且无额外开销的架构。对于视觉提示，我们设计了一种语义激活的视觉提示编码器（SAVPE）。通过将感兴趣区域形式化为掩码，SAVPE将它们与来自PAN的多尺度特征融合，在激活分支中生成低维的分组提示感知权重，并在语义分支中提取与提示无关的语义特征。通过对它们进行聚合得到提示嵌入，从而以最小的复杂度获得良好的性能。对于无提示场景，我们引入了惰性区域-提示对比（LRPC）策略。LRPC不依赖于昂贵的语言模型，而是利用一种专门的提示嵌入来找到所有对象，并使用一个内置的大词汇表进行类别检索。通过仅将与已识别对象匹配的锚点与词汇表进行匹配，LRPC确保了低开销下的高性能。

得益于它们，YOLOE在单个模型中的各种开放提示机制下的检测和分割方面表现出色，具有高推理效率和低训练成本。值得注意的是，如图1所示，在 $3\times$ 更低的训练成本下，YOLOE-v8-S在LVIS[14]上比YOLO-Worldv2-S[5]显著高出3.5 AP，在T4和iPhone 12上的推理速度分别提高了1.4 $\times$ 和 $1.3\times$ 。在视觉提示和无提示设置中，YOLOE-v8-L分别在 $2\times$ 更少的训练数据和 $6.3\times$ 更少的参数下，比T-Rex2高出3.3 AP_r ，比GenerateU高出0.4 AP。为了迁移到COCO [34]，YOLOE-v8-M/L比YOLOv8-M/L分别高出0.4/0.6 AP^b 和0.4/ 0.4AP^m ，且训练时间减少了近 $4\times$ 。我们希望YOLOE能够建立一个强大的基线，并激发实时开放提示驱动视觉任务的进一步发展。

2. 相关工作

传统检测与分割。传统的目标检测与分割方法主要在封闭集范式下运行。早期的两阶段框架[4, 12, 15, 48]，以Faster R-CNN[48]为例，引入了区域提议网络（RPN），随后进行感兴趣区域（ROI）分类和回归。同时，单阶段检测器[10, 35, 38, 56, 72]通过在单个网络内基于网格的预测来优先考虑速度。YOLO系列[1, 21, 27, 47, 59, 60]在这一范式中发挥了重要作用，并在现实世界中广泛使用。此外，DETR[28]及其变体[28, 69, 77]通过采用基于Transformer的架构去除启发式驱动的组件，标志着一个重大转变。为了获得更细粒度的结果，现有的实例分割方法预测像素级掩码而不是边界框坐标[15]。为此，YOLACT[3]通过整合原型掩码和掩码系数来促进实时实例分割。基于DINO[69]，MaskDINO[29]利用查询嵌入和高分辨率像素嵌入图来生成二进制掩码。

文本提示检测与分割。开放词汇目标检测[13,25,61,68,74 - 76]的最新进展集中在通过将视觉特征与文本嵌入对齐来检测新类别。具体来说，GLIP[32]通过对大规模图像 - 文本对进行有基础的预训练，统一了目标检测和短语定位，展示了强大的零样本性能。DetCLIP[65]通过用描述丰富概念来促进开放词汇学习。此外，Grounding DINO[37]通过将跨模态融合集成到DINO中增强了这一点，改善了文本提示与视觉表示之间的对齐。YOLO - World[5]进一步展示了基于YOLO架构预训练具有开放识别能力的小型检测器的潜力。YOLO - UniOW[36]在YOLO - World的基础上利用自适应决策学习策略。同样，一些开放词汇实例分割模型 $[11, 18, 26, 45, 63]$ 从先进的基础模型中学习丰富的视觉语义知识，以对新的目标类别进行分割。例如，X - Decoder[79]和OpenSeeD[71]探索了开放词汇检测和分割任务。APE[54]引入了一种通用视觉感知模型，该模型使用各种文本提示对图像中的所有对象进行对齐和提示。

视觉提示检测与分割。

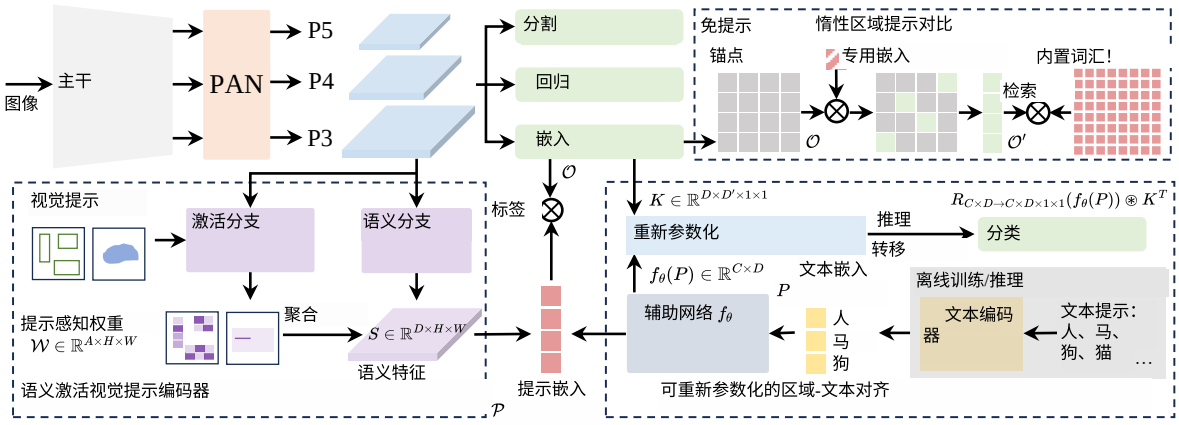


图2. YOLOE概述，其支持针对各种开放提示机制的检测和分割。对于文本提示，我们设计了一种可重新参数化的区域-文本对齐策略，以在零推理和传输开销的情况下提高性能。对于视觉提示，采用SAVPE以最小成本通过增强的提示嵌入对视觉线索进行编码。对于无提示设置，我们引入了惰性区域-提示对比策略，以通过检索有效地为所有识别出的对象提供类别名称。

虽然文本提示提供了一般性描述，但某些对象仅用语言描述可能具有挑战性，例如那些需要专业领域知识的对象。在这种情况下，视觉提示可以更灵活、更具体地指导检测和分割，补充文本提示[19, 20]。OV-DETR[67]和OWL-ViT[41]利用CLIP编码器来处理文本和图像提示。MQ-Det[64]用来自查询图像的特定类视觉信息增强文本查询。DINOv[30]将视觉提示作为通用和引用视觉任务的上下文示例进行探索。T-Rex2[20]通过区域级对比对齐集成视觉和文本提示。对于分割，基于大规模数据，SAM[23]提出了一个灵活且强大的模型，该模型可以进行交互式和迭代式提示。SEEM[80]进一步探索了用更多种提示类型分割对象。语义-SAM[31]在语义理解和粒度检测方面表现出色，处理全景和部分分割任务。

无提示检测与分割。现有的方法在进行开放集检测与分割推理时仍依赖显式提示。为解决这一局限性，一些工作[33, 40, 49, 62, 66]探索与生成式语言模型集成，为所有检测到的对象生成对象描述。例如，GRiT[62]在密集字幕和对象检测任务中都使用了文本解码器。DetCLIPv3[66]在大规模数据上训练对象字幕器，使模型能够生成丰富的标签信息。GenerateU[33]利用语言模型以自由形式生成对象名称。

结束语。据我们所知，除了DINO-X[49]，很少有工作能在单一架构中实现跨各种开放提示机制的对象检测与分割。然而，DINO-X需要大量的训练成本和显著的推理开销，严重限制了其在实际边缘部署中的实用性。

相比之下，我们的YOLOE旨在提供一个高效且统一的模型，该模型具有实时性能和效率，且易于部署。

3. 方法

在本节中，我们详细介绍YOLOE的设计。基于YOLOs（3.1节），YOLOE通过RepRTA（3.2节）支持文本提示，通过SAVPE（3.3节）支持视觉提示，并通过LRPC（3.4节）支持无提示场景。

3.1. 模型架构

如图2所示，YOLOE采用了典型的YOLO系列架构[1, 21, 47]，由骨干网络、PAN、回归头、分割头和对象嵌入头组成。骨干网络和PAN为图像提取多尺度特征。对于每个锚点，回归头预测用于检测的边界框，分割头生成用于分割的原型和掩码系数[3]。对象嵌入头遵循YOLO系列中分类头的结构，不同之处在于最后 $1\times$ 卷积层的输出通道数从封闭集场景中的类别数改为嵌入维度。同时，给定文本和视觉提示，我们分别采用RepRTA和SAVPE将它们编码为归一化的提示嵌入 $\mathcal{P}$ 。它们用作分类权重，并与锚点的对象嵌入 $\mathcal{O}$ 进行对比以获得类别标签。该过程可以形式化为

$$\text{Label} = \mathcal{O} \cdot \mathcal{P}^T : \mathbb{R}^{N\times D} \times \mathbb{R}^{D\times C} \to \mathbb{R}^{N\times C}, \quad (1)$$

其中 N 表示锚点的数量， C 表示提示的数量， D 分别表示嵌入的特征维度。

3.2. 可重新参数化的区域-文本对齐

在开放集场景中，文本嵌入与对象嵌入之间的对齐决定了识别类别的准确性。先前的工作通常引入复杂的跨模态融合来改进视觉-文本表示以实现更好的对齐[5, 37]。然而，这些方法会带来显著的计算开销，尤其是在文本数量众多时。鉴于此，我们提出了可重新参数化的区域-文本对齐（RepRTA）策略，该策略通过可重新参数化的轻量级辅助网络在训练期间改进预训练的文本嵌入。文本与锚点对象嵌入之间的对齐可以通过零推理和转移成本来增强。

具体来说，对于长度为 C 的 T 文本提示，我们首先使用CLIP文本编码器 [44, 57] 来获得预训练的文本嵌入 $P =$ 文本编码器 (T) 。在训练之前，我们预先缓存数据集中所有文本的嵌入，并且可以移除文本编码器而无需额外的训练成本。同时，如图3.(a)所示，我们引入了一个轻量级辅助网络 f_θ ，它只有一个前馈块[53,58]，其中 θ 表示可训练参数，与封闭集训练相比引入的开销较小。它导出增强的文本嵌入 $\mathcal{P} = f_\theta(P) \in \mathbb{R}^{C\times D}$ ，用于在训练期间与锚点的对象嵌入进行对比，并导致视觉语义对齐得到改善。设 $K \in \mathbb{R}^{D\times D'\times 1\times 1}$ 是对象嵌入头中具有输入特征 $I \in \mathbb{R}^{D'\times H\times W}$ 的最后一个卷积层的内核参数， $\circledast$ 是卷积算子， R 是重塑函数，我们有

$$\text{Label} = R_{D\times H\times W\to HW\times D}(I \circledast K)\cdot(f_\theta(P))^T. \quad (2)$$

此外，在训练之后，辅助网络可以与目标嵌入头一起重新参数化为YOLO的相同分类头。重新参数化后最后卷积层的新内核参数 $K' \in \mathbb{R}^{C\times D'\times 1\times 1}$ 可以通过以下方式推导得出

$$K' = R_{C\times D\to C\times D\times 1\times 1}(f_\theta(P)) \circledast K^T. \quad (3)$$

最终预测可以通过标签 $= I \circledast K'$ 获得，这与原始的YOLO架构相同，从而在部署和转移到下游封闭集任务时实现零开销。

3.3. 语义激活视觉提示编码器

视觉提示旨在通过视觉线索（例如框和掩码）来指示感兴趣的对象类别。为了生成视觉提示嵌入，先前的工作通常采用重型Transformer设计[20, 30]，例如可变形注意力[78]，或额外的CLIP视觉编码器[44, 67]。然而，由于复杂的算子或高计算需求，这些方法在部署和效率方面带来了挑战。

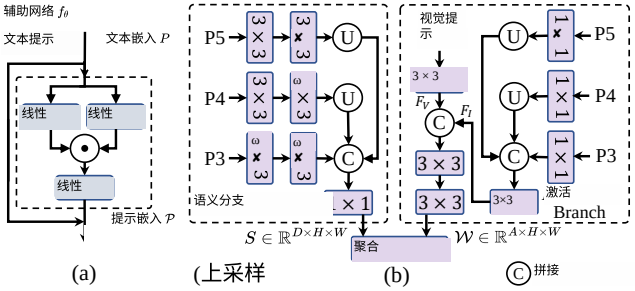


图3. (a) RepRTA中轻量级辅助网络的结构，它由一个SwiGLU FFN模块[53]组成。(b) SAVPE的结构，它由生成与提示无关的语义特征的语义分支和提供分组的提示感知权重的激活分支组成。因此，通过它们的聚合可以有效地导出视觉提示嵌入。

考虑到这一点，我们引入了语义激活视觉提示编码器(SAVPE)来有效处理视觉线索。它具有两个解耦的轻量级分支：(1) 语义分支在 D 通道中输出与提示无关的语义特征，而无需融合视觉线索的开销，并且(2) 激活分支通过在低成本下在少得多的通道中使视觉线索与图像特征相互作用来产生分组的提示感知权重。然后，它们的聚合在最小复杂度下导致信息丰富的提示嵌入。

如图3.(b)所示，在语义分支中，我们采用与对象嵌入头类似的结构。利用来自PAN的多尺度特征 $\{P_3, P_4, P_5\}$ ，我们分别对每个尺度使用两个 3×3 卷积。上采样后，将特征连接并投影以得到语义特征 $S \in \mathbb{R}^{D\times H\times W}$ 。在激活分支中，我们将视觉提示形式化为掩码，指示区域为1，其他区域为0。我们对其进行下采样并利用 3×3 卷积得到提示特征 $F_V \in \mathbb{R}^{A\times H\times W}$ 。此外，我们通过卷积从 $\{P_3, P_4, P_5\}$ 中获取图像特征 $F_I \in \mathbb{R}^{A\times H\times W}$ 以与之融合。然后将 F_V 和 F_I 连接起来并用于输出提示感知权重 $\mathcal{W} \in \mathbb{R}^{A\times H\times W}$ ，该权重在提示指示区域内使用softmax进行归一化。此外，我们将 S 的通道分成 A 组，每组 $\frac{D}{A}$ 个通道。第 i 组中的通道共享来自 $\mathcal{W}$ 的第 i 个通道的权重 $\mathcal{W}_{i:i+1}$ 。利用 $A \ll D$ ，我们可以在低维度下用图像特征处理视觉线索，带来最小的成本。此外，通过两个分支的聚合可以得到提示嵌入

$$\mathcal{P} = \text{Concat}(G_1, ..., G_A); G_i = \mathcal{W}_{i:i+1}\cdot S^T_{\frac{D}{A}*i:\frac{D}{A}*(i+1)}. \quad (4)$$

因此，它可以与锚点的对象嵌入进行对比，以识别感兴趣类别的对象。

3.4. 惰性区域提示对比

在没有明确指导的无提示场景中，模型需要识别图像中所有带有名称的对象。

先前的工作通常将这种设置表述为一个生成问题，即使用语言模型为密集发现的对象生成类别[33, 49, 62]。然而，这会带来显著的开销，例如在GenerateU[33]中使用具有250M参数的FlanT5-base[6]和在DINO-X[49]中使用OPT-125M[73]等语言模型，远远无法满足高效性要求。鉴于此，我们将这种设置重新表述为一个检索问题，并提出了LazyRegion-Prompt Contrast (LRPC)策略。它以经济高效的方式从内置的大词汇表中为带有对象的锚点懒检索类别名称。这种范式对语言模型零依赖，同时具有良好的效率和性能。

具体而言，对于预训练的YOLOE，我们引入了一种专门的提示嵌入，并专门对其进行训练以找到所有对象，其中对象被视为一个类别。同时，我们遵循[16]收集了一个涵盖各种类别的大词汇表，并将其用作内置的检索数据源。人们可以直接将大词汇表用作YOLOE的文本提示来识别所有对象，然而，通过将大量锚点的对象嵌入与大量文本嵌入进行对比，这会带来显著的计算成本。相反，我们使用专门的提示嵌入 $\mathcal{P}_s$ 通过以下方式找到与对象对应的锚点集 $\mathcal{O}'$ 。

$$\mathcal{O}' = \{o \in \mathcal{O} \mid o \cdot \mathcal{P}_s^T > \delta\}, \tag{5}$$

其中 $\mathcal{O}$ 表示所有锚点， δ 是用于过滤的阈值超参数。然后，仅对 $\mathcal{O}'$ 中的锚点与内置词汇表进行惰性匹配以检索类别名称，从而绕过无关锚点的成本。这进一步提高了效率而不会降低性能，便于实际应用。

3.5. 训练目标

在训练过程中，我们遵循[5]为每个镶嵌样本获取一个在线词汇表，将图像中涉及的文本作为正标签。遵循[21]，我们利用任务对齐的标签分配来使预测与真实情况相匹配。二元交叉熵损失用于分类，IoU损失和分布式焦点损失用于回归。对于分割，我们遵循[3]使用二元交叉熵损失来优化掩码。

4. 实验

4.1. 实现细节

模型。为了与[5]进行公平比较，我们对YOLOE采用相同的YOLOv8架构[21]。此外，为了验证其在其他YOLO上的良好通用性，我们还对YOLO11架构[21]进行了实验。对于这两种架构，我们提供三种模型规模，即小（S）、中（M）和大（L），以满足各种应用需求。文本提示使用预训练的MobileCLIP-B(LT)[57]文本编码器进行编码。默认情况下，我们在SAVPE中凭经验使用 $A = 16$ 。

数据。我们遵循[5]使用检测和定位数据集，包括Objects365（V1）[52]、GoldG[22]（包括GQA[17]和Flickr30k[43]），其中排除了来自COCO[34]的图像。此外，我们利用先进的SAM-2.1[46]模型，使用来自检测和定位数据集的真实边界框生成伪实例掩码，用于分割数据。这些掩码经过过滤和简化以消除噪声[9]。对于视觉提示数据，我们遵循[20]利用真实边界框作为视觉线索。在无提示任务中，我们使用相同的数据集，但将所有对象注释为单个类别以学习专门的提示嵌入。

训练。由于计算资源有限，与YOLO-World训练100个轮次不同，我们首先使用文本提示对YOLOE训练30个轮次。然后，我们仅使用视觉提示对SAVPE训练2个轮次，这避免了支持视觉提示带来的额外显著训练成本。最后，我们针对无提示场景仅对专门的提示嵌入训练1个轮次。在文本提示训练阶段，我们采用与[5]相同的设置。值得注意的是，YOLOE-v8-S / M / L可以在8块英伟达RTX4090 GPU上分别在12.0 / 17.0 / 22.5小时内完成训练，与YOLO-World相比成本降低了 $3\times$ 。对于视觉提示训练，我们冻结所有其他部分并采用与文本提示训练相同的设置。为了实现无提示能力，我们利用相同的数据训练一个专门的嵌入。我们可以看到，YOLOE不仅训练成本低，而且还展现出卓越的零样本性能。此外，为了验证YOLOE在下游任务上的良好可迁移性，我们在COCO [34]上对YOLOE进行微调以用于封闭集检测和分割。我们试验了两种不同的实际微调策略：(1) 线性探测：只有分类头是可学习的；(2) 完全微调：所有参数都是可训练的。对于线性探测，我们仅对所有模型训练10个轮次。对于完全微调，我们分别对包括YOLOE-v8-S / 11-S在内的小规模模型训练160个轮次，以及对包括YOLOE-v8-M / L和YOLOE-11-M / L在内的中大规模模型训练80个轮次。

对于文本提示评估，我们遵循开放词汇目标检测任务的标准协议，将基准中的所有类别名称用作输入。对于视觉提示评估，遵循[20]，对于每个类别，我们随机采样 N 训练图像（默认 $N = 16$ ），使用其真实边界框提取视觉嵌入，并计算平均提示嵌入。对于无提示评估，我们采用与[33]相同的协议。使用预训练的文本编码器[57]将开放式预测映射到基准内语义相似的类别名称。与[33]不同的是，我们通过选择最有信心的预测来简化映射过程，并且无需进行top-k选择和波束搜索。

表1. LVIS上的零样本检测评估。为了进行公平比较，在LVIS小验证集上以零样本方式报告固定平均精度（Fixed AP）。训练时间是针对文本提示的，基于8块英伟达V100 GPU用于[32, 65]，以及8块RTX4090 GPU用于YOLO-World和YOLOE。每秒帧数（FPS）分别在使用TensorRT的英伟达T4 GPU和使用CoreML的iPhone 12上进行测量。结果以文本提示（T）和视觉提示（V）类型提供。对于训练数据，OI、HT和CH分别表示OpenImages [24]、HierText [39]和CrowdHuman [51]。OG表示Objects365 [52]和GoldG [22]，G-20M代表Grounding-20M [50]。

模型	提示类型	参数	训练数据	训练时间	FPST4 / 苹果手机	AP	AP_r	AP_c	AP_f
GLIP-T [32]	T	232M	OG,Cap4M	1337.6h	-1-	26.0	20.8	21.4	31.0
GLIPv2-T [70]	T	232M	OG,Cap4M	-	-1-	29.0	-	-	-
GDINO-T [37]	T	172M	OG,Cap4M	-	-1-	27.4	18.1	23.3	32.7
DetCLIP-T [65]	T	155M	OG	250.0h	-1-	34.4	26.9	33.9	36.3
G-1.5 Edge [50]	T	-	G-20M	-	-1-	33.5	28.0	34.3	33.9
T-Rex2 [20]	V	-	O365.OI,HT CH.SA-1B	-	-1-	37.4	29.9	33.9	41.8
YWorldv2-S [5]	T	13M	OG	41.7h	216.4 / 48.9	24.4	17.1	22.5	27.3
YWorldv2-M [5]	T	29M	OG	60.0h	117.9 / 34.2	32.4	28.4	29.6	35.5
YWorldv2-L [5]	T	48M	OG	80.0h	80.0 / 22.1	35.5	25.6	34.6	38.1
YOLOE-v8-S	T / V	12M / 13M	OG	12.0h	305.8 / 64.3	27.9 / 26.2	22.3 / 21.3	27.8 / 27.7	29.0 / 25.7
YOLOE-v8-M	T / V	27M / 30M	OG	17.0h	156.7 / 41.7	32.6 / 31.0	26.9 / 27.0	31.9 / 31.7	34.4 / 31.1
YOLOE-v8-L	T / V	45M / 50M	OG	22.5h	102.5 / 27.2	35.9 / 34.2	33.2 / 33.2	34.8 / 34.6	37.3 / 34.1
YOLOE-11-S	T / V	10M / 12M	OG	13.0h	301.2 / 73.3	27.5 / 26.3	21.4 / 22.5	26.8 / 27.1	29.3 / 26.4
YOLOE-11-M	T / V	21M / 27M	OG	18.5h	168.3 / 39.2	33.0 / 31.4	26.9 / 27.1	32.5 / 31.9	34.5 / 31.7
YOLOE-11-L	T / V	26M / 32M	OG	23.5h	130.5 / 35.1	35.2 / 33.7	29.1 / 28.1	35.0 / 34.6	36.5 / 33.8

我们使用来自[16]的标签列表作为内置大词汇表，共有4585个类别名称，默认情况下，经验性地将 $\delta = 0.001$ 用于LRPC。对于所有三种提示类型，按照[5, 20, 33]，以零样本方式在包含1203个类别的LVIS[14]上进行评估。默认情况下，报告LVIS小验证子集上的固定AP[7]。为了迁移到COCO，按照[1, 21]评估标准AP。此外，我们在配备TensorRT的英伟达T4 GPU和配备CoreML的移动设备iPhone 12上测量所有模型的FPS。

4.2. 文本和视觉提示评估

如表1所示，对于在LVIS数据集上的检测任务，YOLOE在不同模型规模下的效率和零样本性能之间展现出良好的权衡。我们还注意到，这些结果是在少得多的训练时间内取得的，例如，比YOLO-Worldv2快3倍。具体而言，YOLOE-v8-S / M / L分别比YOLOv8-Worldv2-S / M / L在 $3.5/0.2/0.4\mathrm{AP}$ 上表现更优，同时在T4和iPhone12上的推理速度分别加快了 $1.4\times/1.3\times/1.3\times$ 和 $1.3\times/1.2\times/1.2\times$ 。此外，对于具有挑战性的稀有类别，我们的YOLOE-v8-S和YOLOE-v8-L分别取得了 5.2% 和 $7.6\%\mathrm{AP}_r$ 的显著提升。此外，与YOLO-Worldv2相比，虽然YOLOE-v8-M / L实现了更低的 AP_f ，但这种性能差距主要源于YOLOE在一个模型中集成了检测和分割。这种多任务学习引入了一种权衡，对频繁出现类别的检测性能产生了不利影响，如表5所示。此外，采用YOLO11架构的YOLOE也展现出良好的性能和效率。

例如，YOLOE-11-L与YOLO-Worldv2-L的平均精度相当，但在T4和iPhone 12上推理速度显著加快 $1.6\times$ ，这突出了我们的YOLOE强大的通用性。

此外，视觉提示的加入进一步增强了 YOLOE 的通用性。与 T - Rex2 相比，YOLOE - v8 - L 在训练数据少 $2\times$ （T - Rex2：310 万，我们：140 万）且训练资源低得多（T - Rex2：16 块英伟达 A100 GPU，我们：8 块英伟达 RTX4090 GPU）的情况下，实现了 $3.3\mathrm{AP}_r$ 和 $0.9\mathrm{AP}_c$ 的提升。此外，对于视觉提示，虽然我们仅在冻结其他部分的情况下对 SAVPE 进行 2 个轮次的训练，但我们注意到，对于各种模型规模，它可以在 AP_r 和 AP_c 方面取得与文本提示相当的效果。这表明视觉提示在处理文本提示往往难以准确描述的低频对象方面是有效的，这与文献 [20] 中的观察结果相似。

此外，对于分割任务，我们展示了在LVIS验证集上使用表2中报告的标准 AP^m 的评估结果。结果表明，YOLOE通过利用文本提示和视觉提示展现出强大的性能。具体而言，YOLOE-v8-M / L以零样本方式分别达到20.8和 $23.5\mathrm{AP}^m$ ，显著优于在LVIS-Base数据集上微调的YOLO-Worldv2-M / L，分别高出3.0和 $3.7\mathrm{AP}^m$ 。这些结果充分显示了YOLOE的优越性。

4.3. 无提示评估

如表3所示，在无提示场景下，YOLOE也展现出卓越的性能和效率。

表2. LVIS上的分割评估。我们使用报告的标准 AP^m 在LVIS验证集上评估所有模型。YOLOE支持文本（T）和视觉线索（V）作为输入。† 表示预训练模型在LVIS-Base数据上针对分割头进行了微调。相比之下，我们以零样本方式评估YOLOE，在训练期间不使用来自LVIS的任何图像。

模型	提示	AP^m	AP^m_r	AP^m_c	AP^m_f
YWorld-M †	T	16.7	12.6	14.6	20.8
YWorld-L†	T	19.1	14.2	17.2	23.5
YWorldv2-M †	T	17.8	13.9	15.5	22.0
YWorldv2-L†	T	19.8	17.2	17.5	23.6
YOLOE-v8-S	T / V	17.7 / 16.8	15.5 / 13.5	16.3 / 16.7	20.3 / 18.2
YOLOE-v8-M	T / V	20.8 / 20.3	17.2 / 17.0	19.2 / 20.1	24.2 / 22.0
YOLOE-v8-L	T / V	23.5 / 22.0	21.9 / 16.5	21.6 / 22.1	26.4 / 24.3
YOLOE-11-S	T / V	17.6 / 17.1	16.1 / 14.4	15.6 / 16.8	20.5 / 18.6
YOLOE-11-M	T / V	21.1 / 21.0	17.2 / 18.3	19.6 / 20.6	24.4 / 22.6
YOLOE-11-L	T / V	22.6 / 22.5	19.3 / 20.5	20.9 / 21.7	26.0 / 24.1

表3. LVIS上的无提示评估。按照[33]中的协议，在LVIS小验证集上报告固定平均精度（Fixed AP）。每秒帧数（FPS）是在配备Pytorch [42]的英伟达T4 GPU上测量的。

模型	骨干网络	参数	AP	AP_r	AP_c	AP_f	FPS
GenerateU [33]	斯温-T	297M	26.8	20.0	24.9	29.8	0.48
生成U [33]	斯温-L	467M	27.9	22.3	25.2	31.4	0.40
YOLOE-v8-S	YOLOv8-S	13M	21.0	19.1	21.3	21.0	95.8
YOLOE-v8-M	YOLOv8-M	29M	24.7	22.2	24.5	25.3	45.9
YOLOE-v8-L	YOLOv8-L	47M	27.2	23.5	27.0	28.0	25.3
YOLOE-11-S	YOLO11-S	11M	20.6	18.4	20.2	21.3	93.0
YOLOE-11-M	YOLO11-M	24M	25.5	21.6	25.5	26.1	42.5
YOLOE-11-L	YOLO11-L	29M	26.3	22.7	25.8	27.5	34.9

具体而言，YOLO-v8-L的平均精度均值（AP）达到27.2，以及23.5 AP_r，比采用Swin-T骨干网络的GenerateU在AP上高出0.4，在 AP_r 上高出3.5，同时参数减少了 $6.3\times$，推理速度加快了 $53\times$。这表明通过将开放式问题重新表述为针对内置大词汇量的检索任务，YOLOE是有效的，并强调了其在不依赖明确提示的情况下跨广泛类别进行泛化的潜力。这种功能还增强了YOLOE的实用性，使其能够应用于更广泛的实际场景中。

4.4. 下游迁移

如表4所示，在向下游封闭集检测和分割任务转移到COCO数据集时，在两种微调策略下，YOLOE在有限的训练轮次下都展现出良好的性能。具体而言，对于线性探测，在不到2%的训练时间内，YOLOE - 11 - M / L分别可以达到YOLO11 - M / L性能的80%以上。这突出了YOLOE强大的可迁移性。对于完全微调，YOLOE可以在有限的训练成本下进一步提升性能。

表4. 在COCO数据集上的下游迁移。我们在COCO数据集上对YOLOE进行微调，并报告检测和分割的标准AP。我们采用两种实际的微调策略进行实验，即线性探测和完全微调。

模型	轮次	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
从头开始训练							
YOLOv8 - S	500	44.7	61.4	48.7	36.6	58.0	38.6
YOLOv8-M	300	50.0	66.8	54.8	40.5	63.4	43.3
YOLOv8-L	300	52.4	69.3	57.2	42.3	66.0	44.9
YOLO11-S	500	46.6	63.3	50.6	37.8	59.7	40.0
YOLO11-M	600	51.5	68.5	55.7	41.5	65.0	43.9
YOLO11-L	600	53.3	70.1	58.2	42.8	66.8	45.5
线性探测							
YOLOE-v8-S	10	35.6	51.5	38.9	30.3	48.2	32.0
YOLOE-v8-M	10	42.2	59.2	46.3	35.5	55.6	37.7
YOLOE-v8-L	10	45.4	63.3	50.0	38.3	59.6	40.8
YOLOE-11-S	10	37.0	52.9	40.4	31.5	49.7	33.5
YOLOE-11-M	10	43.1	60.6	47.4	36.5	56.9	39.0
YOLOE-11-L	10	45.1	62.8	49.5	38.0	59.2	40.6
完全调优							
YOLOE-v8-S	160	45.0	61.6	49.1	36.7	58.3	39.1
YOLOE-v8-M	80	50.4	67.0	55.2	40.9	63.7	43.5
YOLOE-v8-L	80	53.0	69.8	57.9	42.7	66.5	45.6
YOLOE-11-S	160	46.2	62.9	50.0	37.6	59.3	40.1
YOLOE-11-M	80	51.3	68.3	56.0	41.5	64.8	44.3
YOLOE-11-L	80	52.6	69.7	57.5	42.4	66.2	45.2

表5. 基于文本提示的YOLOE路线图。标准AP以零样本方式在LVIS小验证集上报告。FPS分别在配备TensorRT (T)和CoreML (C)的英伟达T4 GPU和iPhone 12上测量。

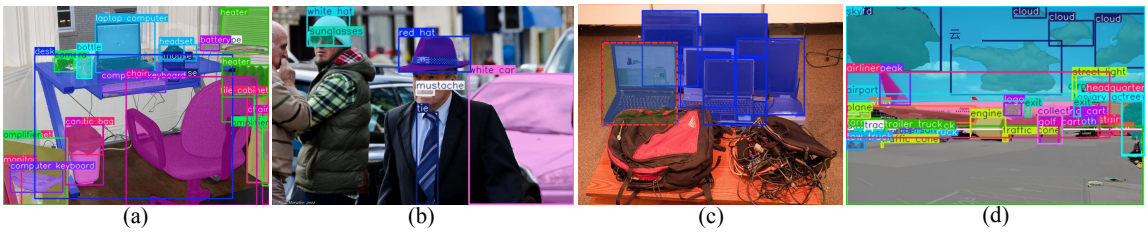
模型	轮次	AP	AP_r	AP_c	AP_f	每秒帧数（测试/训练）
YOLO-Worldv2-L	100	33.0	22.6	32.0	35.8	80.0 / 22.1
+ 更少的训练轮次	30	31.0	22.6	28.8	34.2	80.0 / 22.1
+ 全局负样本字典	30	31.9	22.8	31.0	34.4	80.0 / 22.1
• 跨模态融合	30	30.0	19.1	28.0	33.9	102.5 / 27.2
+ 移动CLIP编码器	30	31.5	20.2	30.5	34.4	102.5 / 27.2
+ RepRTA	30	33.5	29.5	32.0	35.5	102.5 / 27.2
+ 分割。(YOLOE)	30	33.3	30.8	32.2	34.6	102.5 / 27.2

例如，在训练轮次减少近 $4\times$ 的情况下，YOLOE - v8 - M / L的性能分别比YOLOv8 - M / L高出 $0.4AP^m$ 和0.6 AP^b。在训练时间减少 $3\times$ 的情况下，YOLO - v8 - S在检测和分割任务上也比YOLOv8 - S表现更好。这些结果充分表明，YOLOE可以作为向下游任务迁移的有力起点。

4.5. 消融研究

我们进一步对YOLOE中设计的有效性进行了广泛分析。实验在YOLOE - v8 - L上进行，默认情况下，在LVIS小验证集上报告标准AP用于零样本评估。

YOLOE路线图。我们概述了从基线模型YOLOv8-Worldv2-L到我们的逐步进展



(a) (b) (c) (d)

图4。(a) 在LVIS上的零样本推理。(b) 使用定制文本提示的结果，其中“白帽子、红帽子、白色汽车、太阳镜、胡子、领带”作为文本提示。(c) 使用视觉提示的结果，其中红色虚线边界框用作视觉线索。(d) 在无提示场景下的结果，其中未提供明确提示。更多示例请参考补充材料。

表6。SAVPE的有效性。

模型	AP	AP_r	AP_c	AP_f
掩码池	30.4	27.6	31.3	30.2
SAVPE	31.9	29.4	32.5	31.7
$A=1$	30.9	28.2	31.9	30.4
$A=16$	31.9	29.4	32.5	31.7
$A=32$	31.9	28.2	33.0	31.7

表7。LRPC的有效性。

模型	LRPC	AP	AP_r	AP_c	AP_f	FPS
v8-S	✘	21.0	19.1	21.4	21.0	56.5
	$\delta=1e^{-3}$	21.0	19.1	21.3	21.0	95.8
	$\delta=1e^{-4}$	21.0	19.1	21.3	21.0	66.1
	$\delta=1e^{-2}$	20.8	19.1	21.2	20.8	106
v8-L	✘	27.2	23.5	27.0	28.0	19.9
	$\delta=1e^{-3}$	27.2	23.5	27.0	28.0	25.3

表格5中关于文本提示的YOLOE-v8-L。初始基线指标为33.0% AP，由于计算资源有限，我们首先将训练轮次减少到30，导致AP为31.0%。此外，我们不是使用空字符串作为基础数据的负文本，而是遵循[65]，通过维护一个全局字典来采样更多样化的负提示。全局字典是通过选择在训练数据中出现超过100次的类别名称构建的。这导致AP提高了0.9%。接下来，我们移除跨模态融合以避免昂贵的视觉-文本特征交互，这导致AP下降了1.9%，但在T4和iPhone 12上分别实现了 1.28× 和 1.23× 的推理加速。为了解决这种下降，我们使用更强的MobileCLIP-B(LT)文本编码器[57]来获得更好的预训练文本嵌入，这将AP恢复到31.5%。此外，我们采用RepRTA来增强锚点的对象和文本嵌入之间的对齐，这导致AP显著提高了2.3%，且推理开销为零，显示了其有效性。最后，我们引入分割头并同时训练用于检测和分割的YOLOE。尽管由于多任务学习导致 0.2% AP和 0.9AP_f 下降，但YOLOE获得了分割任意对象的能力。

SAVPE的有效性。为了验证SAVPE对视觉输入的有效性，我们移除了激活分支，仅利用掩码池化，通过公式化的视觉提示掩码聚合语义特征。如表6所示，SAVPE显著优于“掩码池”，AP值高出1.5。这是因为“掩码池”忽略了提示指示区域内不同位置的语义重要性差异，而我们的激活分支有效地对这种差异进行了建模，从而改善了语义特征的聚合，并为对比提供了更好的提示嵌入。我们还研究了激活分支中不同组数（即A）的影响。

如表6所示，仅使用一组（即 $A=1$ ）也可以提高性能。此外，在 $A=16$ 条件下，我们可以实现31.9 AP的强劲性能，获得良好的平衡，更多组导致的性能差异很小。

长期相对位置编码（LRPC）的有效性。为了验证LRPC在无提示设置下的有效性，我们引入了一种基线方法，该方法直接利用内置的大词汇表作为文本提示，让YOLOE识别所有物体。表7展示了比较结果。我们观察到，在相同性能下，我们的LRPC通过懒检索已发现物体的锚点类别并跳过大量无关类别，分别为YOLOE-v8-S / L显著获得了 1.7 × 1.3× 的推理加速。这些结果充分突出了其有效性和实用性。此外，通过设置不同的过滤阈值 δ ，LRPC可以实现不同的性能和效率权衡，例如，对于YOLOE-v8-S，仅 0.2AP 的性能下降就能实现 1.9× 的加速。

4.6. 可视化分析

我们在四种场景下对YOLOE进行可视化分析：(1) 图4.(a)中对LVIS进行零样本推理，其类别名称为文本提示；(2) 图4.(b)中的文本提示，其中可以输入任意文本作为提示；(3) 图4.(c)中的视觉提示，其中可以绘制视觉线索作为提示；(4) 图4.(d)中无明确提示，模型识别所有对象。我们可以看到，YOLOE在这些不同场景下表现良好，能够准确检测和分割各种对象，进一步展示了其在各种应用中的有效性和实用性。

5. 结论

在本文中，我们提出了YOLOE，这是一个单一的高效模型，它通过各种开放提示机制无缝集成了目标检测和分割。具体来说，我们引入了RepRTA、SAVPE和LRPC，以使YOLO能够以良好的性能和低成本处理文本提示、视觉线索和无提示范式。得益于它们，YOLOE在各种提示方式下都具有强大能力和高效率，能够实时“所见即所得”。我们希望它能作为一个强大的基线，以激发进一步的进展。

A. 更多实现细节

数据。我们使用Objects365[52]、GoldG [22]（包括GQA[17]和Flickr30k [43]）来训练YOLOE。表8展示了它们的详细信息。我们利用SAM-2.1-Hiera-Large [46]，以带真实边界框作为提示来生成高质量的分割掩码伪标签。我们过滤掉面积过小的伪标签。为了增强掩码边缘的平滑度，我们对掩码应用高斯核，分别对小掩码和大掩码使用 3×3 和 7×7 核。此外，我们按照[9]中的方法对掩码进行细化，该方法迭代地简化掩码轮廓。这在保留整体结构的同时减少了噪声像素。

表8. YOLOE训练的数据细节。

数据集	类型	边界框	面具	图像	注释。
Objects365 [52]	检测	✓	✓	609k	8,530k
GQA [17]	基础	✓	✓	621k	3,662k
Flickr [43]	基础	✓	✓	149k	638k

训练。对于所有模型，我们采用AdamW优化器，初始学习率为0.002。批量大小和权重衰减分别设置为128和0.025。数据增强包括颜色抖动、随机仿射变换、随机水平翻转和马赛克增强。在迁移到COCO时，对于线性探测和完全微调，我们使用初始学习率为0.001的AdamW优化器，将批量大小和权重衰减分别设置为128和0.025。

B. 对LRPC的更多分析

为了定性地展示LRPC策略的有效性，我们可视化了过滤后用于类别检索的保留锚点数量。我们在图5中展示了它们在LVIS小验证集上随过滤阈值 δ 变化的平均数量。结果表明，随着 δ 的增加，不同模型中保留的锚点数量大幅减少。与使用总共8400个锚点的基线场景相比，这种减少显著降低了计算开销。例如，对于YOLOE-v8-S，在 $\delta = 0.001$ 时，有效锚点数量减少了 80% ，在相同性能下推理速度提高了 $1.7\times$ （见论文中的表7）。结果进一步证实了类别检索中锚点的显著冗余性，并验证了LRPC的有效性。

C. 更多可视化结果

为了定性地展示YOLOE的有效性，我们在各种场景下展示了更多关于它的可视化结果。

在LVIS上的零样本推理。在图6中，我们展示了YOLOE在LVIS上的零样本推理能力。

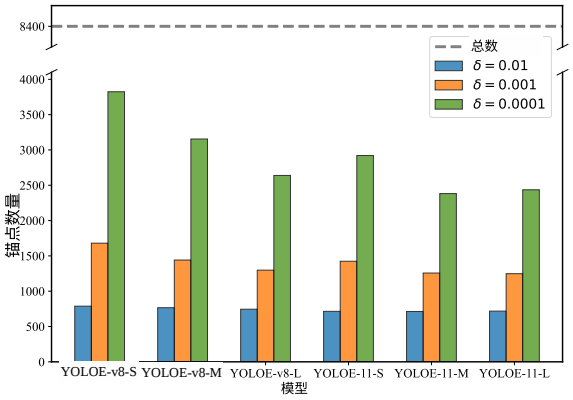


图5. LRPC中不同过滤阈值下保留的锚点数量。虚线表示总数。

通过利用1203个类别名称作为文本提示，该模型展示了其在各种图像中检测和分割不同对象的能力。

使用定制文本进行提示。图7展示了使用定制文本提示的结果。我们可以看到，YOLOE能够解释通用和特定的文本输入，实现精确的目标检测和细粒度分割。这种能力允许用户通过定义不同粒度级别的输入提示来调整模型的行为，以满足特定需求。

使用视觉输入进行提示。在图8中，我们展示了以视觉输入作为提示的YOLOE的结果。视觉输入可以采用各种形式，如边界框、点或手工制作的形状。它也可以跨图像提供。我们可以看到，通过视觉提示指示目标对象，YOLOE可以准确找到同一类别的其他实例。此外，它在不同的对象和图像上表现良好，展现出强大的能力。

无提示推理。图9展示了采用无提示范式的YOLOE的结果。我们可以看到，在这种设置下，YOLOE能够对各种物体实现有效的识别。这突出了其在预定义输入不可用或不切实际的场景中的实用性。

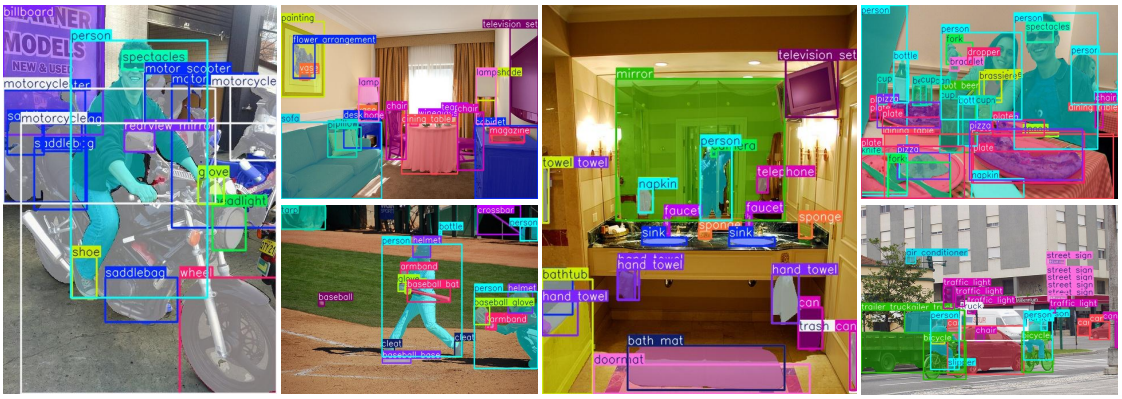


图6. LVIS上的零样本推理。LVIS的类别作为文本提示提供。

{食物，饮料}　　{红色汽车，黑色汽车，黄色汽车}　　{岩石，树木，河流，动物}

图7. 带有定制文本的提示。YOLOE适用于通用和特定文本提示，以便灵活使用。

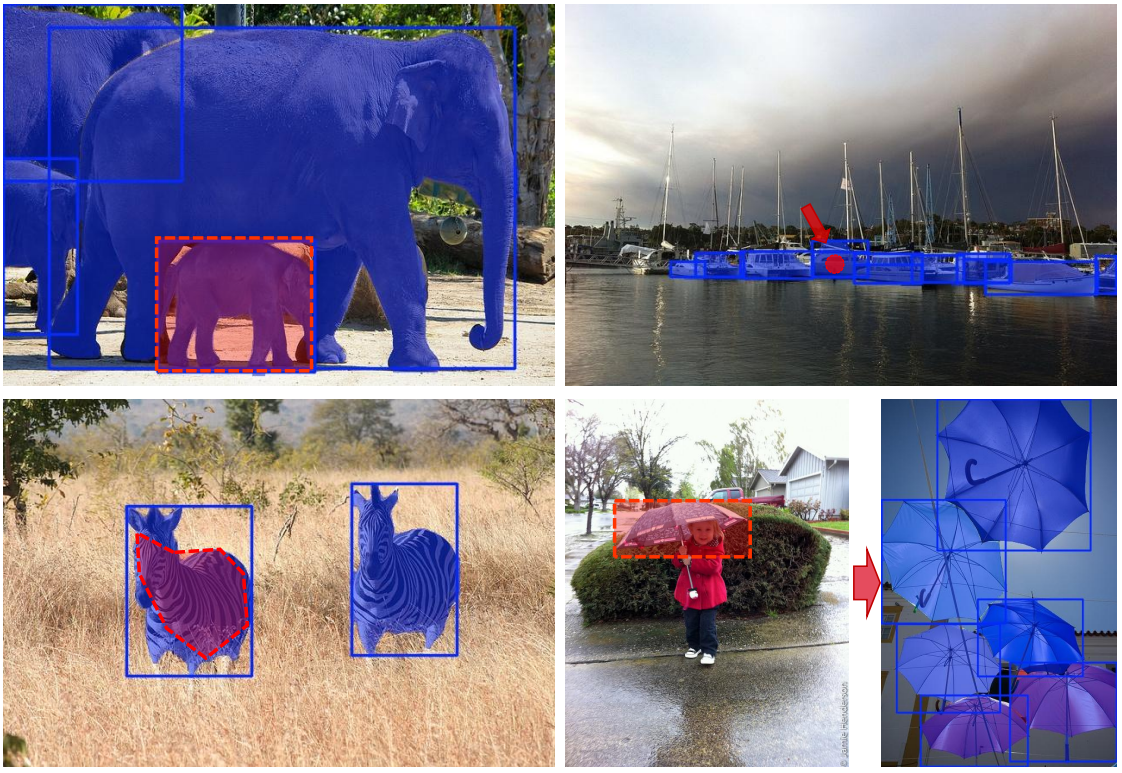


图8. 带有视觉输入的提示。YOLOE展示了在各种视觉提示（如边界框（左上角）、点（右上角）、手工形状（左下角））引导下识别物体的能力。视觉提示也可以应用于跨图像（右下角）。