单步全景分割

摘要

我们提出了一种端到端的单步的方法，以接近视频帧速率将可计数的对象实例和背景区域分割为无重叠的全景分割。目前最高水平的方法速度远低于视频帧速率，并且主要依靠将实例分割和语义背景分割相结合。我们的方法通过使用目标检测放宽了此要求，但是仍然可以解决类间和类内的重叠问题，达到无重叠分割。在共享的编码解码器主干上，我们利用多分支进行语义分割，目标检测和实例中心点预测。最后，我们的全景头将所有输出组合成全景分割，全景头甚至可以处理分支间的冲突预测以及某些错误预测。我们的网络以 21.8FPS 在 MS-COCO 数据集上达到了 32.6%PQ，将全景分割推向了更广泛的应用领域。

引言

场景理解是计算机视觉中基础但又具有挑战性的任务之一。传统的场景理解人物包括目标检测，实例分割和语义分割。前者人物对可计数前景分类和定位，后者任务在不区分不同实例下分割同类的图像区域。这两者都需要对场景全面的理解。但是，近年来，这些任务使用的方法已经变得不同了。例如，候选区域网络被用于目标检测和实例分割，全卷积网络已被用于语义分割。

全景分割由 Kirillov 等引入，旨在解决在视觉场景理解领域中不同任务有不同的建模策略问题。目前最高水平的全景分割方法是基于 Mask R-CNN 扩展的两阶段方法。与目标检测和实例分割相似，高精度但是运行时间较慢，这使他们在实际应用中困难。相比之下，单步方法有潜力更快和更简单，以满足实质高效率要求的实际应用，例如机器人技术和自动驾驶。在目标检测中，单步方法如 YOLO，SSD，和它们的后继者已经对机器人技术有实质的影响。因此，推进具有绝佳性能的算法和探索速度和准确性的平衡一样重要。

本文中，我们提出统一的单步方法解决全景分割任务。不像以前的方法只为精度最大化，我们在速度和精度权衡的基础上设计我们的模型。

我们使用概念上简单但有效的组件进行目标检测和语义分割任务。与启发式的合并不同分支的输出的许多相关方法相比，我们研究了更复杂的合并过程。我们提出了一种全景头部，能融合检测到的目标和分割的区域，生成连贯的全景分割。此外，我们提出了一种新技术，使用实例中心预测与目标检测和语义分割结合，扩展了最初的简单合并。我们在流行的 COCO 数据集上广泛的评估了我们的方法，展示了我们方法的有效性。特别是与最高水平的方法相比，我们获得了巨大的运行时间收益。此外，在单步全景分割中我们还将我们的方法与其它并发方法比较，获得了更好的速度与精度的权衡。

相关工作

语义分割：最高水平的语义分割利用全卷积网络并遵循编码解码方案。多种方法通过聚合多尺度特征来合并上下文信息。使用空洞卷积允许方法重点利用更大图片的上下文信息而不用增加模型参数。最近，使用金字塔池化变体的方法成为主流。PSPNet 使用空间金字塔池化，DeepLab 提出了多孔的空间金字塔池化模块（ASPP）。借助这种多尺度上下文信息，这些方法在标准基准上证明了很高的准确性。我们网络结构中的全卷积部分利用金字塔的多尺度特征专门用于语义分割，但是由于空洞卷积在计算上更昂贵而没有使用。

目标检测和实例分割：与语义分割相比，目标检测和实例分割的方法识别同一类的不同实例。使用区域候选的两阶段方法成为高精度目标检测和实例分割的主导概念，例如 Mask R-CNN。在这些方法中，生成一组候选区域，之后被子网络分类和分割。虽然这些方法达到最高水平的精度，由于计算成本不能被很多应用使用。对目标检测来说，单步方法解决了速度和精度的权衡问题且有显著的结果。除了 YOLO 和 SSD，新方法使用锚点的密集区域或使用热点图来生成边界框，如 RetinaNet 和 CornerNet。

当单步检测和两阶段检测的准确度差距在缩小时，单步实例分割方法落后于最高水平的方

法。一些方法学习嵌入，通过聚类来产生实例。尽管这个概念不复杂，现在的技术也只达到较低的精度，并且不一定比两阶段方法快。另一种产生实例的方法是预测每个像素的实例中心，与预测的矩形框组合得到实例。然而，此技术远未达到最高水平的准确性。由于单次实例分割的准确性远低于两阶段方法，我们仅选择使用单次目标检测网络（RetinaNet）预测对象边界框。

全景分割：全景分割可以理解为多任务学习问题。以前的多任务学习工作着重于通过共享主干学习多个任务。但是，多个分支的冲突预测通常尚未解决。全景分割，正如定义的那样，期望对每一个像素进行唯一的预测，要求对每一个像素分配一个语义类别并为属于事物类别的所有像素分配实例 ID。自从引入全景分割以来，所有发布的监督学习方法都集中于两阶段架构，在 COCO 全景挑战赛中所有高分参赛作品，情况也是一样。

典型的网络设计通过添加并行的语义分割分支并以一个融合步骤将实例分割和语义分割结合产生全景分割来扩展 Mask R-CNN。融合步骤需要解决两个主要问题。第一个，Mask R-CNN 预测的实例可以重叠；第二个，两个分支的输出可能发生冲突。例如，一个像素可能被一个分支预测属于汽车实例，被另一个分支预测属于天空。通过基于高于某个阈值的置信度得分对实例进行排序并定义一个 IoU 阈值来跳过与其它实例重叠过多的实例，所有解决方案（Panoptic Segmentation， Panoptic Feature Pyramid Networks， Learning to Fuse Things and Stuff，Attention-Guided Unified Network for Panoptic Segmentation）【18,17,22,23】都解决了第一个问题。（An End-To-End Network for Panoptic Segmentation，Learning Instance Occlusion for Panoptic Segmentation）【27,21】提出了一个模块，分别学习不同语义类别或实例的空间关系。在【17】的基础架构之上，【23】增加了注意力层减少不同分支的冲突。【22】提出一个损失因子强迫分支间保持一致性。前面提到的方法仅通过获得 Mask R-CNN 的输出来解决第二个问题，只对未赋值的像素使用语义分割。唯一更复杂的方法是 UPSNet，使用一个全景头部将语义分割和实例分割结合预测实例感知对数变换（logits）。

由于所有的这些方法是基于 Mask R-CNN，它们都承受高昂的计算成本，因此不能应用于需要实时操作的应用。据我们了解，全景分割没有公开的单步方法。我们也将我们的方法与 DeeperLab 的衍生工作比较。DeeperLab 是基于 DeepLab 和 PersonLab 的单步架构。与类无关的实例通过关键点热图预测。它们的语义类别由语义预测中的多数投票决定。预测的填充类别只用于剩下的像素。

在这项工作中，我们提出单步架构，因此不是扩展 Mask R-CNN。与现在大多数顶尖方法不同，在我们的结构中使用了一个复杂的合并头部解决预测冲突问题。与 UPSNet 相比，我们的合并头能处理边界框而不是分割，这使我们的网络构架能大范围的用于可行的检测。与 DeeperLab 不同，我们证明了我们的端到端网络架构达到了接近视频帧速率的性能。

方法

我们提出的端到端网络由共享的卷积主干组成，以解码编码的方式提取多尺度特征。在此之上，我们为任务语义分割，目标检检测和可选的实例中心预测添加了多个分支。所有的这些预测都输入到我们最后的全景头部产生全景分割而无需任何后期处理合并步骤。整个网络架构如图 2 所示。我们网络设计的目的是准确性，速度，简单性和可比较性。

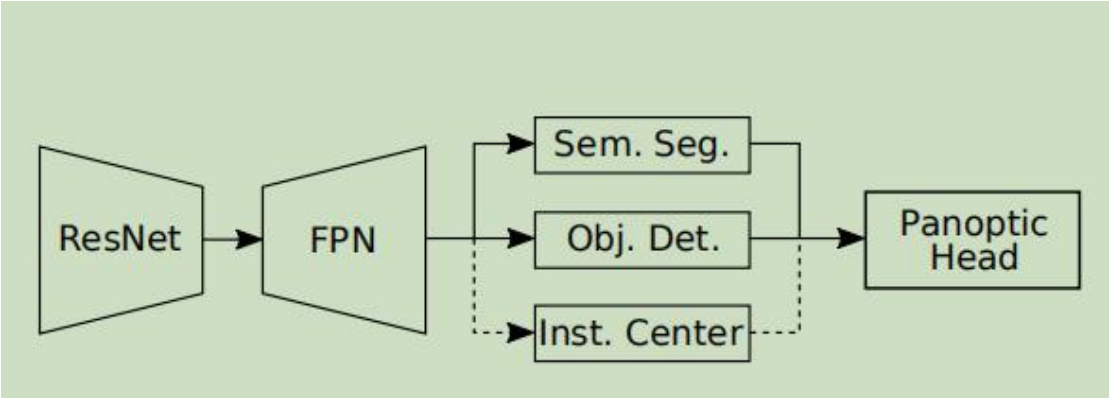

图 2：我们提出的网络架构利用编码解码主干，多达三个分支和一个全景头部产生全景分割。

提出的架构

主干：我们的主干使用残差网络作为编码器，特征金字塔网络作为解码器。因此，主干提供的多尺度上下文信息能被用于语义分割和与目标检测。我们选择标准的 ResNet-50 FPN 尽可能公平的与以前的方法作比较。我们的共享主干生成的金字塔级别，分辨率从 1/128 到 1/4，每个级别有 256 个特征维度。

语义分割分支：和【17】类似，我们使用轻量的全卷积网络来预测语义分割。我们使用金字塔级别从 p5 到 p2，对应 1/32 到 1/4 的分辨率，交替使用卷积和上采样操作直到所有金字塔层到 1/4 分辨率。卷积后，我们使用 GroupNorm 层做归一化，ReLU 做激活函数。最后，我们通过累加合并所有的上采样金字塔层并使用附加的卷积层来产生语义的对数变换（logits）。网络概述如图 3 所示。先前所有的工作中，语义分割的实现仅略有变化【27,37,23】。由于都达到了类似的性能，所以我们选择了轻量级的分支达到我们的目标。然而，与大多数相关工作不同，我们的语义分割分支预测所有的前景类（things）和背景类（stuff），这是我们全景头部所需要的。

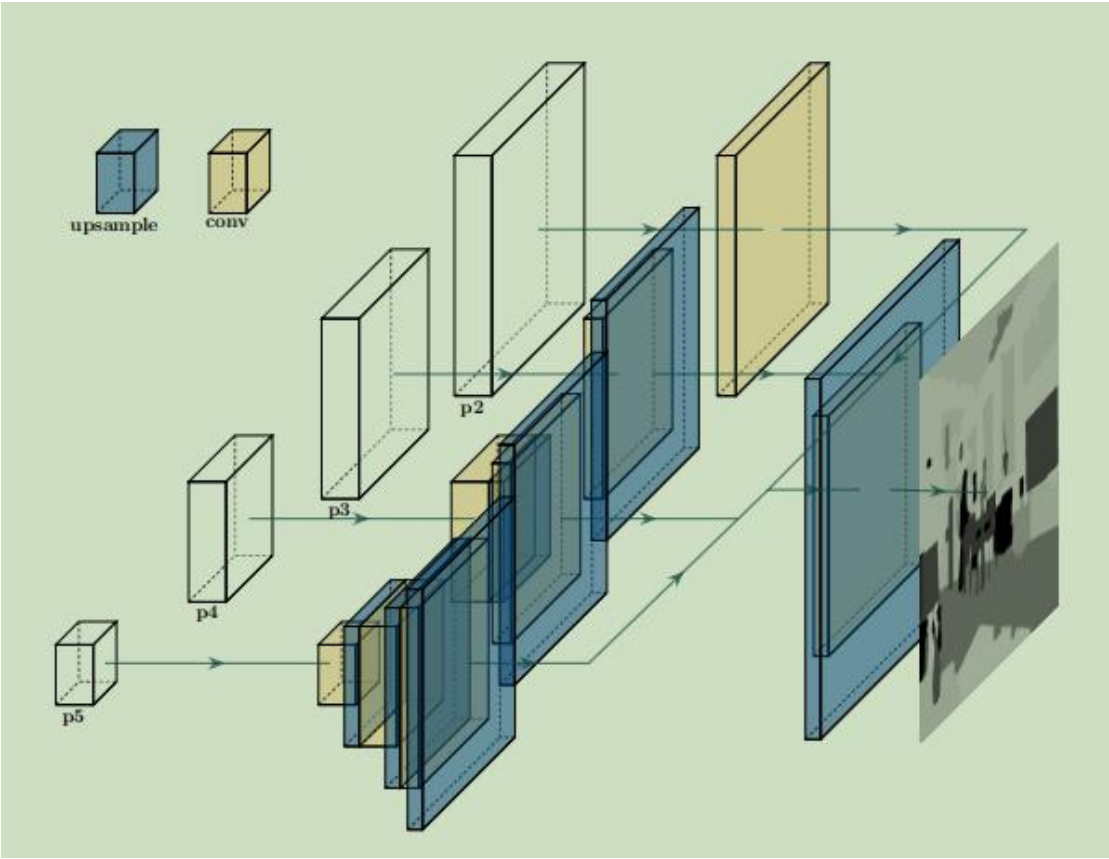

图 3：语义分割分支利用 p2 到 p5 级别产生对数变换（logits）。每一个特征金字塔张量被卷积（黄）和上采样（蓝）到相同的空间分辨率，最后在计算对数变换和上采样到原始分辨率之前逐个元素的求和。

目标检测分支：对我们的目标检测分支，我们使用 RetinaNet 架构，因为它具有良好的速度与精度折衷。RetinaNet 有 两个全卷积的子网络用于分类和回归锚点。从 1/128 到 1/4 所有的金字塔级别使用相同的网络权重来检测不同大小的物体（things）。图 4 展示了目标检测分支。我们保留标准的超参数来证明我们整个网络的开箱即用性能。

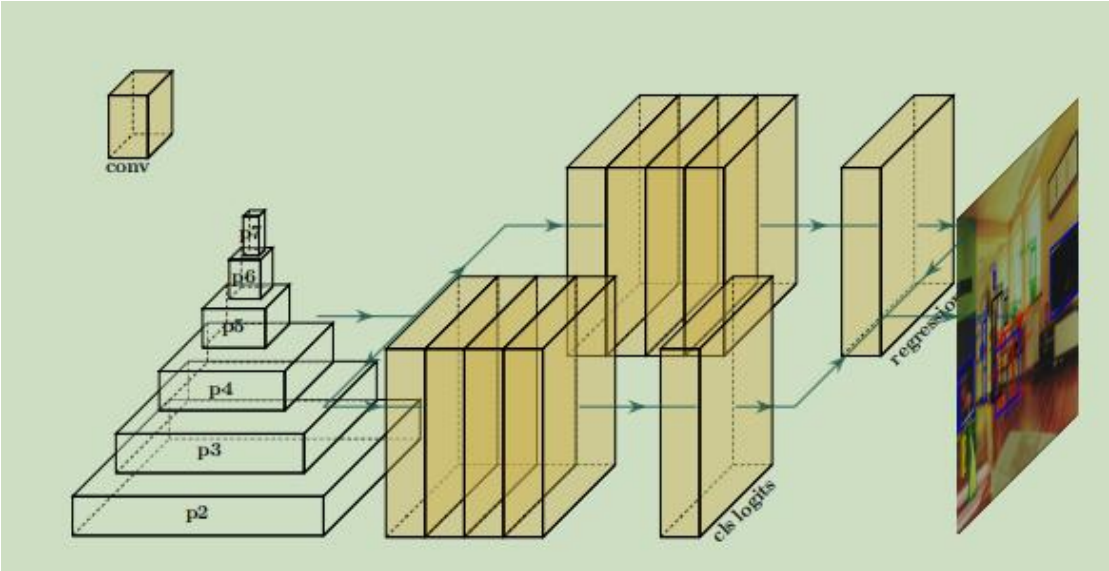

图 4：目标检测分支利用特征金字塔的所有级别来预测矩形框和对应的类别。每个任务都将

每个级别的金字塔输入 4 层卷积（黄）。最后，预测所有锚点的偏移量和对应的类别对数变换（logits）。

全景头部：受 UPSNet 启发，我们提出一个新的无参数的全景头部，能够处理目标检测而不是实例对数变换（logits）。我们的目标是创建实例感知对数变换（logits）Y，类似语义对数变换（logits）那样推断，又能提供额外的实例信息。因此，我们将我们的语义对数变换(logits)X 分割为 stuff X_{stuff} 和 things 对数变换（logits） X_{things}。我们仅将 X_{stuff} 复制到 Y，因为它们不需要实例 ID。对于 X_{things}，我们使用目标检测的结果，每个目标都包含一个类别 c 和矩形框 b。我们使用 c 做索引来从 X_{things} 选择对应的切片。二维的对数变换（logits）X_c 中，我们使用矩形框 b 来裁剪对数变换（logits），即，我们过滤边界框之外的所有的对数变换（logit）值。我们将这些对数变换（logits）$X_{c,\ b}$ 与 stuff 对数变换（logits）X_{stuff} 堆叠一起。在索引选择之后，$X_{c,\ b}$ 会有与语义对数变换（logits）一样的宽高。然而，Y 的通道数直接取决于检测到的目标个数，因此，每个输入都不同。
为了说明这点，考虑下面一个例子。假设检测到三台车。我们的网络会选择车的 logit 切片 $X_{c=car}$ 三次，裁剪对应的矩形框 b1,b2,b3，与 Xstuff 堆叠一起获得实例感知对数变换（logits）Y。因此，我们最终得到三个新的对数变换（logits）切片，每个对应一个汽车实例。被语义分割分割为 things 的区域，如果没有被目标检测器检测到目标将被丢弃。此外，错误的检测对象仍然可以从语义分割分支中获得较低的对数变换（logit）值，因此不会在最终的全景分割头部显示。所以，一个分支的某些错误可以被另一个分支的纠正。
由于保留了原始的对数（logits），即使检测器未提供此类信息，argmax 操作仍然能实现预测对象的轮廓。唯一的问题来自相同类别的重叠框，不同类别间没有此问题。由于这种情况下选择了相同的 logit 切片，因此重叠区域在我们的结果中具有相同的值。因此，argmax 操作不能选择正确的实例。为了克服这个问题，我们提出并研究了不同的策略。

重叠部分解决
所有之前发布的工作都要求实例分割结果来产生全景分割。这项功能工作的一部分是放宽目标检测器的要求。虽然全景头解决了类别间的重叠问题，但是没有直接解决类别内的重叠问题。因此，我们提出是那种不同的策略来克服这一缺陷。结合全景头部，我们可以解决这两类重叠，因此使用实例分割的子网络是可选的。
最高置信度策略：这个策略与之前使用 Mask R-CNN 预测的重叠实例（任何类别）处理类似。假设置信度得分与对象存在的可能性密切相关，则以置信度降序排列重叠矩形框会导致得分更高。使用此策略，预测可能会掩盖误报。
最小优先策略：PQ 指标【18】，用来衡量全景分割质量，重点在于小实例，因为即使最小的实例也要和最大的实例一样计入分数。因此，以大小递增的顺序排列重叠实例防止大实例遮盖较小的实例。
最近中心策略：上面两种策略都会受到以下影响，重叠的部分只会分给一个实例。因此，它们引入了来自矩形框的直角的直线轮廓。由于类内重叠的矩形框数量受到一定的限制，这个问题不会很好的反应在最终的得分上，但是这些情况下结果的视觉效果会受到极大的影响。因此，在我们的架构中我们提出了第三个分支。第三个分支的任务是与类无关的实例中心点预测，我们指定为矩形框中心点的预测。
Uhrig 等【35】使用实例中心点预测直接聚类实例，但成功有限。然而，聚类预测实现实例分割被认为比确定已经预测的重叠矩形框中哪个更可能是对的困难的多。我们利用同样的语义分割的轻量级架构只预测从像素点位置沿 x 方向和 y 方向的绝对偏移。通过添加预测的偏移到像素点位置并计算包含该像素点所有矩形框中心的 L2 距离，可以解决重叠。最小距离

为每个像素点提供最可能的实例。假设实例中心预测准确，此策略能保证精确的轮廓。

推理与训练

未知的预测：虽然全景分割任务被定义为为每一个像素分配语义类别并为可计数的 things 分配实例 ID，之前的工作在某些情况下会使用未知的预测类别。例如，【17，37】删除了像素区域少于特定阈值的 stuff（填充）区域。此外，当它们的分支产生冲突分配时它们对某些 stuff 区域做出未知的预测。这些减弱任务需求的策略是值得商榷的，但是由于提高了 PQ 指标的得分而得到了鼓舞。对于一些像素，语义分割分支可能预测了一个 things（可计数的）类别，但是目标检测器没有在此位置检测到目标。在我们当前的设置中，全景头部会为这些像素预测最有可能出现的 stuff（填充）类别。但是，在推理过程中对这些情况做出未知预测可能会有所帮助。因此，我们还研究了此类后处理步骤对我们方法的影响。

联合训练：在训练期间，我们结合四个不同的损失函数。我们的目标检测分支使用 focal loss L_{FL} 用于分类，smooth L1 损失 L_{reg} 用于矩形框回归。语义分割分支对每个像素使用交叉熵损失 L_s，我们的实例中心点预测分支每个像素使用 L1 损失 L_i。我们的网络使用损失项的加权组合进行训练：$L=\lambda_o(L_{FL}+L_{reg})+\lambda_s L_s+\lambda_i L_i$。对于提出的最小优先和最高置信度策略，我们在没有实例中心预测的情况下训练了我们的网络，在使用最近中心策略时我们添加了这个分支。我们通过实验研究了不同的加权方案。我们注意到自动选择权重对我们不可行，由于它们代价太大了，例如，需要多次向后传播，或者使用 Adam 优化器时存在优化问题。

实验

我们的目标是证明我们的网络可以用作简单，快速而准确的全景分割算法。我们展示了我们各个分支达到了预期的准确度，并在多任务设置中保持了预期的准确性。我们确定我们的全景头部能结合目标检测和语义分割的结果到全景分割，因此放宽了对实例分割输入的需求。此外，我们证明了我们的全景头部能处理冲突预测和重叠。我们详尽评估了我们的策略，不同的权重，后处理步骤。

数据集：我们在具有挑战性的，有全景分割标注的 COCO 数据集上进行我们的所有实验。此数据集有 118K 训练集，5K 验证集和 80things 类别和 53stuff 类别标注的 20K 测试集。

评估标准：对单任务表现，我们使用平均的 Iou 和 AP 分别在语义分割和目标检测上。对全景分割，我们使用标准的 PQ 指标。PQ^{th} 和 PQ^{st} 得分指数据 things 和 stuff 子集的 PQ 分数。指标定义如下

$$PQ = \frac{\sum_{(p,q)\in \mathrm{TP}} \mathrm{IoU}(p,q)}{|\mathrm{TP}| + \frac{1}{2}|\mathrm{FP}| + \frac{1}{2}|\mathrm{FN}|}$$

TP，FP 和 FN 对应 true positives, false positives 和 false negatives。预测的分割与真实分割 IoU 大于 0.5 属于 TP。

实验设置：我们的 ResNet-50 编码器使用 ImageNet 的预训练权重，FPN 解码器和所有的分支从头开始训练。我们使用 PyTorch 实现我们的模型并在两块 Tesla V100 上训练。与大多数相关工作使用 batch size 为 16，我们使用 batch size 为 4.因此，我们在编码器里冻结 BatchNorm 层并忽略微调他们的统计信息。我们使用 Adam 训练，学习率 1e-5，权重衰减为 1e-4，设置 $\beta 1 = 0.9$ 和 $\beta 2 = 0.99$。我们将训练限制在 14 个周期，在第 7 和第 10 个周期将学习率衰减 10 倍。所有的图片被缩放为短的那边至少为 576，只要长的那边小于 864。在训练中我们只使用了左右翻转作为数据增强。

独立组件

我们通过分别训练他们专门的任务来验证我们每个分支的生效。对于语义分割，我们在 92 类别的 COCO stuff 数据集上比较了单尺寸得分。我们在 80 类别的 COCO 检测数据集上与官方的 RetinaNet 比较得分验证了我们目标检测分支。通过记录在 COCO 全景数据集上正确解决的在真实矩形框上的类内重叠问题个数，我们评估了实例中心预测。在补充材料中我们提供了各个组件的消融实验。

语义分割：由于我们的基准使用相同的架构，我们想证明相似的性能。【17】使用更深，更大的 ResNeXt-152 与更大的输入，但是网络只根据自定义的训练数据进行训练。因此，分数不能直接比较。然而，我们 26.9%的 mIoU 得分接近 27.8%的 mIoU 参考得分，因此，证实了我们的网络确实能执行语义分割。

目标检测：我们在我们的 RetinaNet 实现中使用 Lin 等【25】提出的超参数。此外，我们排除了权重衰减，但是把偏差参数的学习率加倍了。参考实现使用 batch size 16 和 SGD 训练。我们使用同样的 ResNet-50 主干以类似的输入来比较得分。与 34.3%AP 的参考得分相比，我们获得了 33.1%AP 稍差的结果，我们归结于不同的训练配置。

实例中心预测：在我们提出的网络中，实例中心预测被用于解决类内重叠的实例 id 分配。因此，我们通过跟踪真实边界框重叠的正确的 id 分配来评估该分支。对于属于同一类别多个框的 90.8%的像素，分支能分配正确的真实边界框。因此，这的确可以被用于该任务。我们注意到由于‘crowd’标注，很多重叠没有被涉及。所以，我们在图 5 中显示了定性的例子 。

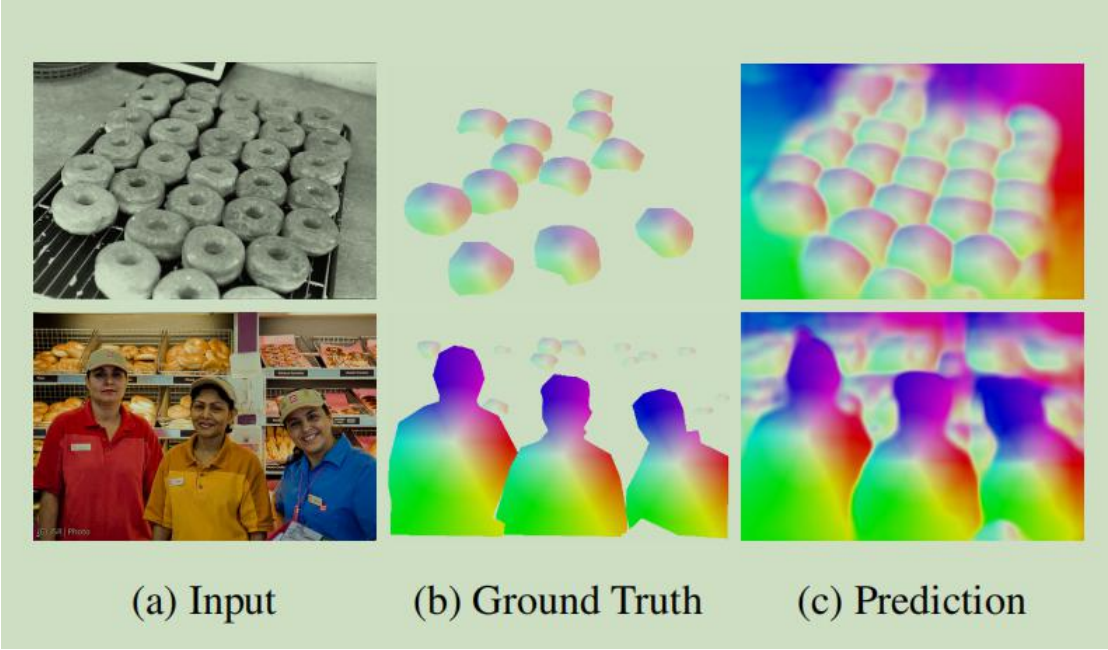

图 5：研究结果表明，实例中心预测分支可以实现同一任务重叠实例的平滑轮廓。可视化反映了向量偏移的方向和大小。

全景分割

确定了我们的组件可以单独工作后，我们现在研究我们全景头部的性能。此外，我们展示了常用的未知检测的效果。最后，我们检查了速度精度的取舍，并与之前的工作和并行衍生的工作比较。

策略：为了评估我们的网络，我们已必须在我们的全景头部选择一个策略解决类内重叠问题。我们选择最高置信度策略，因为在先前的工作中使用此策略来解决 Mask R-CNN 的重叠问题。此外，我们将 0.4 的置信度阈值作为检测器输出。我们在补充材料中提供了此阈值的实验证

明。多任务训练中的一个挑战是设置损失阈值 λ_o，λ_s，λ_i，使整体准确度是最高的。表 1 展示了在 COCO val2017 数据集上我们提出的网络使用最高置信度策略的准确性。我们注意到单任务模型是在具有相同数据集上使用相同训练配置训练的，而之前，我们在不同训练集上报告了得分。这些分数提供了证据，表明相同的权重在全景分割中 things 和 stuff 间达到了最佳的平衡。结果表明还有一系列的权重性能同样的好。值得注意的是语义分割的单任务性能能适应权重的变化甚至能从多任务训练中轻微获利。然而，检测任务受多任务训练影响，得分很大程度上依靠于权重值。

COCO Panoptic						
Weights		Pan. Seg.			Seg.	Det.
Sem.	Obj.	**PQ**	PQ^{th}	PQ^{st}	mIoU	AP
1.00	-	-	-	-	**50.0**	-
-	1.00	-	-	-	-	**31.2**
0.20	0.80	28.9	34.1	21.1	48.4	**30.5**
0.25	0.75	29.4	34.6	21.7	49.0	**30.5**
0.33	0.66	**29.6**	**34.7**	21.9	49.6	30.2
0.50	**0.50**	**29.6**	34.2	22.5	50.2	29.2
0.66	0.33	29.4	33.6	**23.0**	**50.4**	27.7
0.75	0.25	28.8	32.7	22.9	50.2	26.7
0.80	0.20	28.4	32.2	22.8	50.2	25.9

表 1：我们展示了在 COCO val2017 数据集上加权对单任务和 PQ 分数的影响。总之，在全景分割中采用最高置信度策略使用相等的权重达到最佳平衡。

我们可以使用同样训练过的网络来测试最小优先策略，因为它仅在测试时期生效。与最高置信度策略类似，权重相等时性能最佳。我们在补充材料中提供了详细的分析。表 2 比较了两种策略。结果表明，通常使用置信度得分的重叠解决方案优于基于大小排序的。因此，仅使用我们网络的两个分支，应该使用最高置信度策略。

COCO Panoptic					
Policy	**PQ**	PQ^{th}	PQ^{st}	mIoU	AP
HC	**29.6**	**34.2**	**22.5**	50.2	29.2
SF	29.3	33.8	**22.5**	50.2	29.2

表 2：在 COCO val2017 数据集上评估了最高置信度和最小优先策略。证据支持基于置信度而不是大小解决类内重叠的问题。

由于使用最高置信度和最小优先策略的设计，分配类内重叠时会导致直角边。为了克服这个，我们结合了最近中心策略引入第三个分支来实现视觉上合理的轮廓。我们确定此设置可以实现全景分割并在表 3 中调查了不同的损失权重。由于实例中心预测的初始化权重高于其它分支两个数量级，第三分支使用较小权重。虽然得分略有增加，图 6 中的定性比较显示了我们方法的优势。我们认为这种方法的好处没有在 COCO 数据集上得到了很好的表现。最近中心策略在图片上的优势在处理大量同类重叠的实例。然而，这些区域通常没有被密集的标注，而是标记为人群排除在评估范围之外。

COCO Panoptic						
Weights		Pan. Seg.			Seg.	Det.
Sem./Obj.	Inst.	**PQ**	PQ^{th}	PQ^{st}	mIoU	AP
1.0/-	-	-	-	-	**50.0**	-
-/1.0	-	-	-	-	-	**31.2**
0.5	0.001	29.7	34.5	**22.6**	**50.3**	**29.5**
0.5	0.005	29.8	**34.8**	22.3	49.9	29.3
0.5	0.01	**29.9**	**34.8**	22.3	49.7	29.4
0.5	GT	31.4	37.3	22.5	50.2	29.2

表 3：在 COCO val2017 split 中，列出了最终的神经网络使用最近中心策略，在三种不同权重下的 PQ 分数。。与其他策略相比，性能提高了 0.3 – 0.6%PQ。我们还发现，单个任务的较好分数与 PQ 的较好分数并不直接对应，并给出了一个可以用真实的实例中心实现的上限。

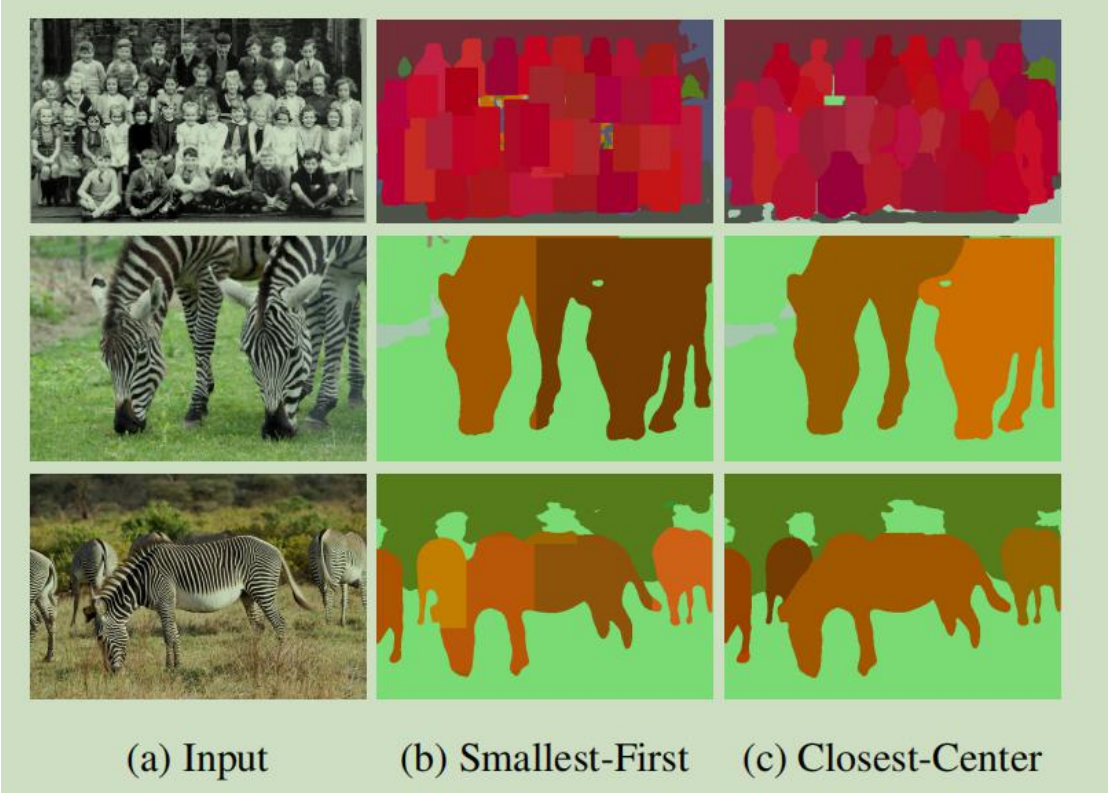

图 6：我们比较了我们最小优先和最近中心策略的全景分割。后一个实现了更合理的视觉结果。

未知预测：我们研究了未知预测的影响并移除了小的 small 的区域。我们将 stuff（填充）区域的阈值设为 4096 个像素，这是最佳做法【17】。表 4 列出了我们最近中心策略的后处理步骤的得分。值得注意的是，移除小的 stuff 去提高了 2.3%的 PQ 值。由于几种相关的方法使用相同的移除步骤而没有消融，我们假设使用语义分割分支有相同的问题，对 stuff 区域的过分割。

COCO Panoptic				
Stuff Rem.	Unk.	**PQ**	PQ^{th}	PQ^{st}
-	-	29.9	34.8	22.5
✓	-	32.2 *(+2.3)*	34.8	28.4 *(+5.9)*
✓	✓	32.4 *(+2.5)*	34.8	28.6 *(+6.1)*

表 4：我们实验了未知预测（Unk.）和小 stuff 区域的移除（Stuff Rem.）。我们在 COCO val2017 上评估了使用最近中心策略的网络。对 stuff 区域使用阈值并添加未知预测增加了 2.5%的 PQ 得分。

最终结果：我们使用最后一段的后处理步骤来公布我们的最终得分。由于全景分割是一个新的且活跃的研究领域，我们也提供了衍生并发工作的得分。表 5 提供相关的两阶段方法和 DeeperLab 衍生并发的单步方法。与其它方法相比，出于运行时间的考虑，我们的得分不是

通过测试时的技巧或其他主干优化得到的，例如可变形卷积可以提高得分。而且，所有方法都能在更深更广的主干上始终取得更好的成绩。为了公平的比较我们的方法，我们公布了 ResNet-50 编码器的得分。值得注意的是，我们的单步方法和两阶段方法的准确度差距大多不超过 10 个百分点。我们的网络比两个 DeeperLab 变体表现的更好。使用 Xception-71 配置的表现稍好一点，但也使用了更深的主干。Xception-71 编码器有 42.1M 参数，我们的编码器只有 23.6M 参数。

COCO Panoptic				
Name	Backbone	**PQ**	PQ^{th}	PQ^{st}
PanFPN [17]	ResNet-50	39.0	45.9	28.7
OANet [27]	ResNet-50	39.0	48.3	24.9
AUNet [23]	ResNet-50	39.6	**49.1**	25.2
UPSNet [37]	ResNet-50	**42.5**	48.5	**33.4**
JSIS† [10]	ResNet-50	26.9	29.3	23.3
OCFusion† [21]	ResNet-50	41.0	49.0	29.0
AdaptIS† [34]	ResNet-50	35.9	40.3	29.3
DeeperLab† [38]	LWMNV2	24.1	-	-
DeeperLab† [38]	WMNV2	27.9	-	-
DeeperLab† [38]	Xception-71	33.8	-	-
Ours (SF)	ResNet-50	31.8	33.8	28.9
Ours (HC)	ResNet-50	32.1	34.2	28.8
Ours (CC)	ResNet-50	32.4	34.8	28.6

表 5：在 COCO val2017 数据集上，两阶段网络相关和衍生的工作，单步方法衍生的工作和此方法的性能。对于两阶段方法限制为 ResNet-50 主干，SF，HC 和 CC 指我们的策略。

使用大的 Xception-71 是最精准的 DeeperLab 的帧数率远低于视频帧数率的原因。我们在表 6 中发布了单步方法的准确性和 FPS。我们的方法和 DeeperLab 的运行时间都是在 Tesla-V100-SXM2 GPU 上测量的。对 DeeperLab，我们使用官方时间测量，但包括从中间结果到最后全景分割的时间。在 DeeperLab 中，中间结果是热图和语义分割，如前面所描述的那样。然而，我们认为 FPS 应该测量从输入到输出完整获得全景分割的运行时间。而且，全卷积网络在更小的输入上由于需要更少的计算量运行更快（通常性能更差）。因此，我们比较方法都在相似的输入条件下保证公平的评估。在这些条件下，我们所有的网络都比竞争对手 DeeperLab 明显更快。从 21.8 到 23.3 FPS，我们几乎达到了视频帧数率。图 7 展示了我们的网络取得了明显更好的速度精度权衡，是我们的方法可以更广泛的应用。我们在图 8 中展示了定性的结果。

COCO Panoptic Test						
Name	Backbone	Input size	**PQ**	PQ^{th}	PQ^{st}	FPS
Ours (SF)	ResNet-50	$576 \times [576, 864]$	32.1	34.3	28.8	23.3
Ours (HC)	ResNet-50	$576 \times [576, 864]$	32.2	34.5	28.8	23.3
Ours (CC)	ResNet-50	$576 \times [576, 864]$	32.6	35.0	29.0	21.8
DeeperLab [38]	LWMNV2	321×321	18.0	18.5	17.2	26.6
DeeperLab [38]	LWMNV2	641×641	24.5	26.9	20.9	13.7
DeeperLab [38]	WMNV2	641×641	28.1	30.8	24.1	12.0
DeeperLab [38]	Xcept.-71	641×641	**34.3**	**37.5**	**29.6**	8.4

表 6：在 COCO test-dev 上我们将我们的方法与 DeeperLab 进行了比较。FPS 包含了从输入到输出的时间。在相似输入的情况下，在运行时间方面，我们在所有情况下均优于 DeeperLab，在得分方面，我们在三分之二的情况下优于 DeeperLab。SF，HC 和 CC 指我们的最小优先，最高置信度和最近中心策略。

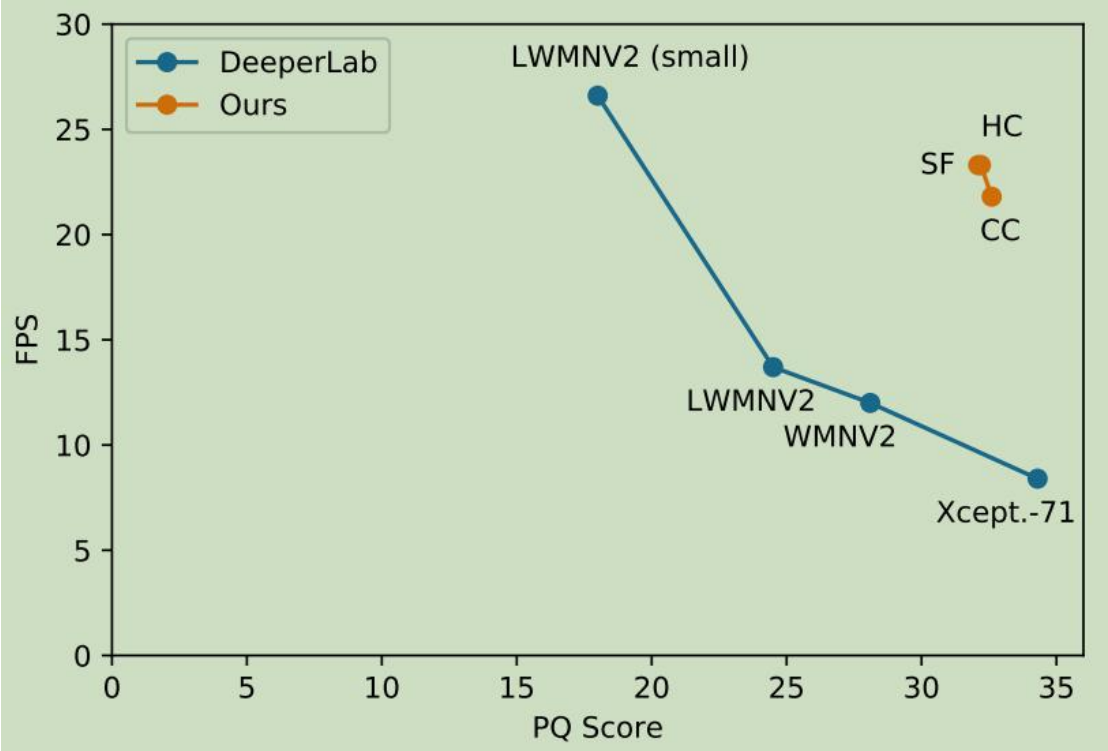

图 7：我们可视化了速度和精度的权衡。DeeperLab 在较慢的运行时间上取得了较好的结果，但速度远低于视频帧数率。该图表明，我们的网络实现了更好的权衡，因此对实际应用是有益的。

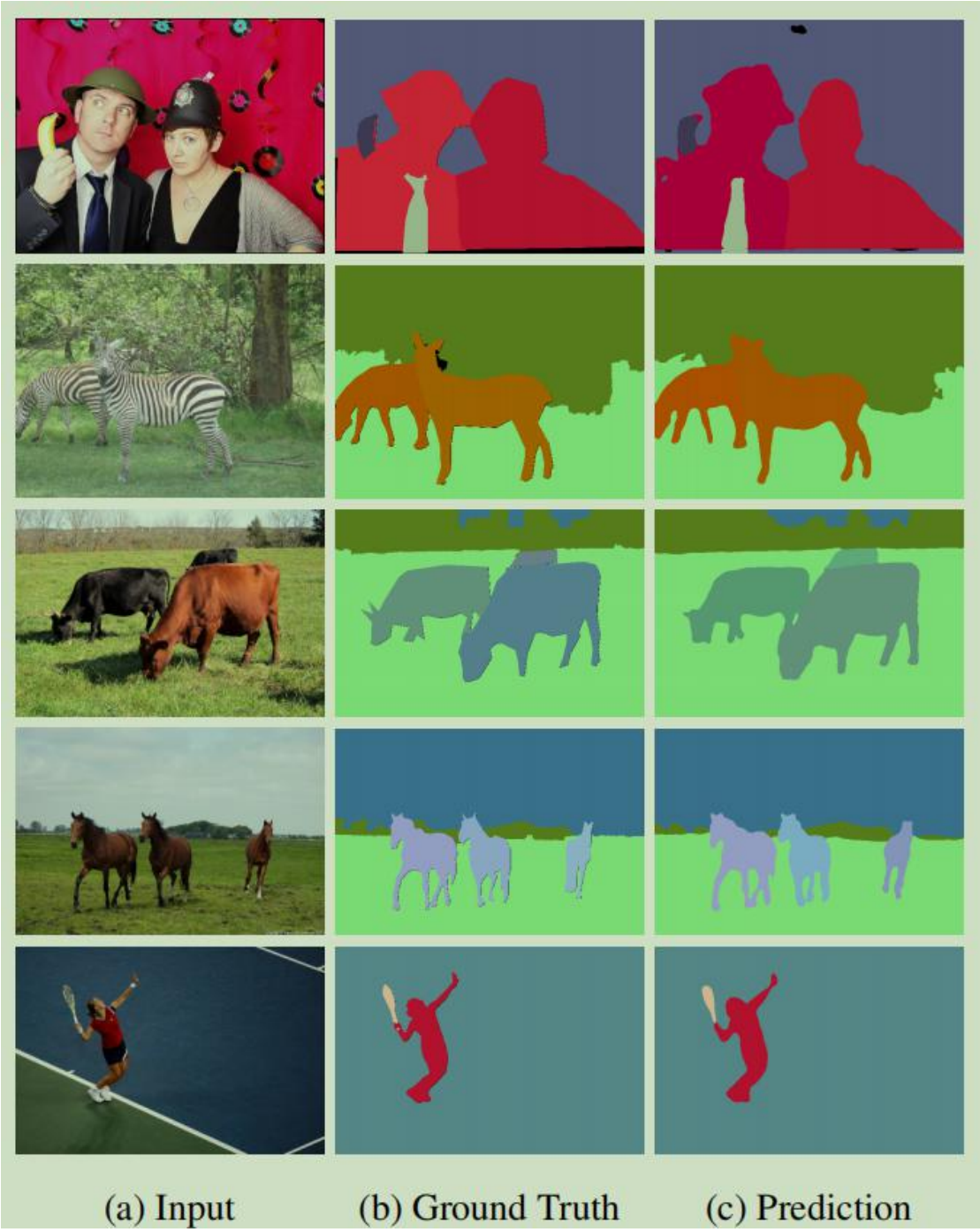

图 8：在 COCO val2017 数据集上使用最近中心策略的我们最终的网络的定性的结果。

结论

在本文中，我们提出一种新的端到端的单步全景分割方法，该方法在接近视频帧数率下取得了出色的精度。我们的全景头部不只是简单的合并子网络的输出，而且生成了实例感知的全景对数变换（logits）。相比于先前的工作，我们提出的方法不依赖与实例分割。同时，我们的全景头部通过结合语义分割，目标检测和实例中心预测解决类内和类间的重叠。该全景头部能在输出最终全景分割结果之前纠正其每个输入之间的错误。