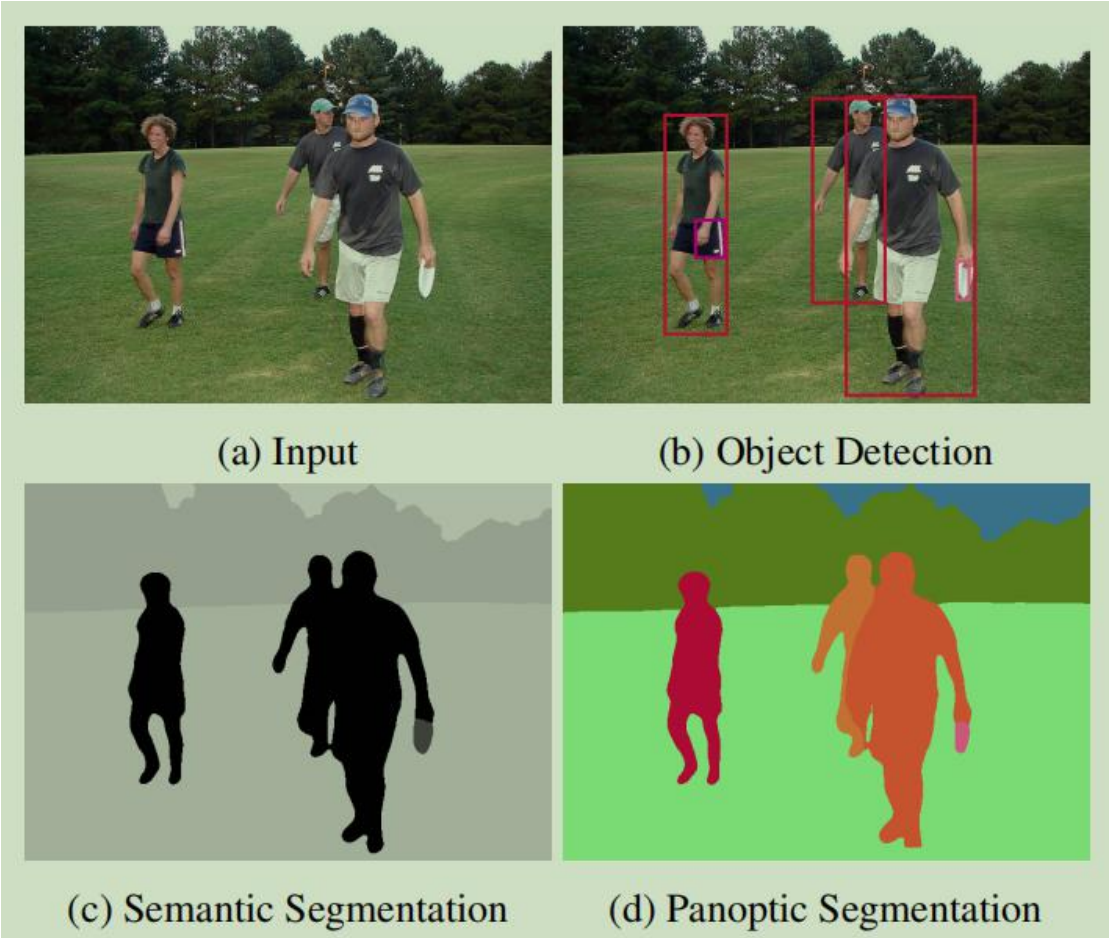


Single-Shot Panoptic Segmentation

单步全景分割

问题

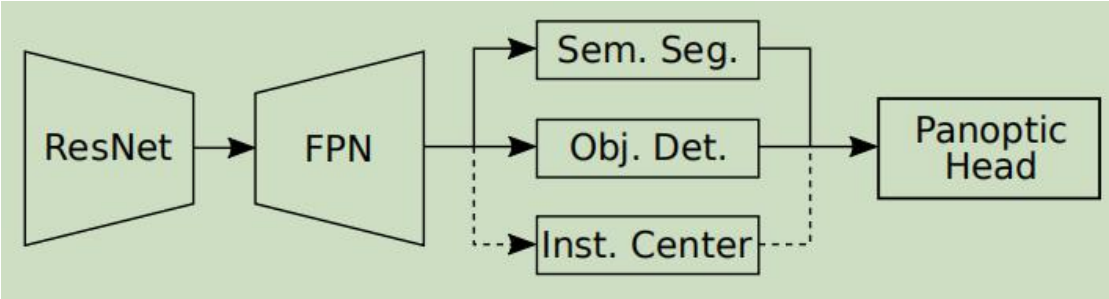
在全景分割中，目前最高水平的方法速度远低于视频帧速率，此论文提出的方法，速度达到了接近视频帧速率，精度下降不到 10 个百分点。

目前最高水平的方法主要依靠实例分割和语义背景分割相结合。

此论文使用目标检测放宽了此要求（替代实例分割），但仍然可以解决类内和类间的重叠问题，达到无重叠分割。

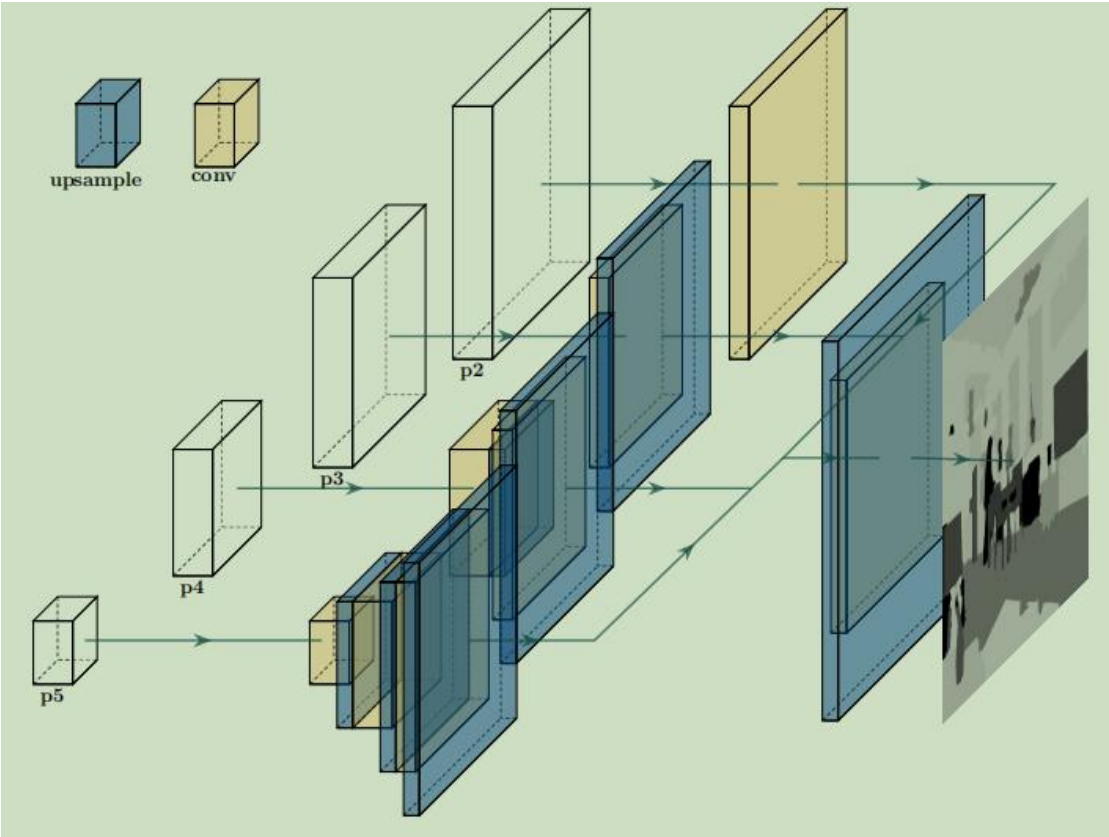
网络框架

一个端到端网络，编码解码主干，三个分支和一个全景头部产生分割。



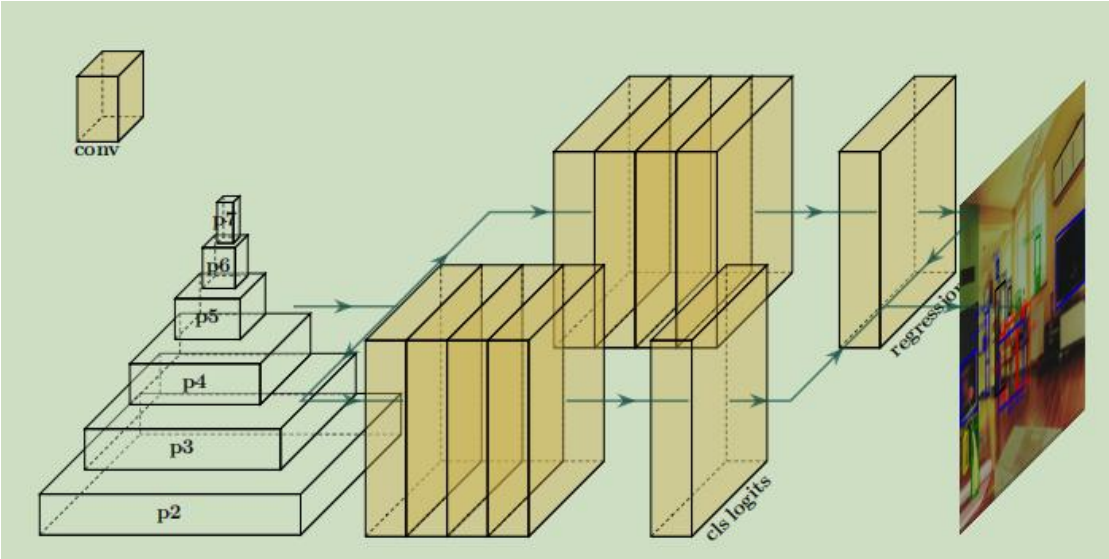
主干以编码解码的方式提取多尺度特征。
使用残差网络为编码器，特征金字塔网络（FPN）作为解码器。
选择标准的 ResNet50 FPN 来与以前的方法公平的比较。
金字塔分辨率 1/128 到 1/4.每个都有 256 个特征维度。

语义分割分支



使用 p5 到 p2 的金字塔层，对应 1/32 到 1/4 的分辨率。
交替使用卷积和上采样操作，直到金字塔层分辨率达到 1/4.
之后使用 GroupNorm 做归一化，使用 ReLU 做激活函数。
再将金字塔层相加，再使用卷积上采样层，得到语义分割，预测所有的类别。

目标检测分支



使用 RetinaNet 架构。因为它有良好的速度和精度的折中。
有两个全卷积的子网络分别用于分类和锚点回归。
从 1/128 到 1/4 分辨率，所有的金字塔层使用相同的网络权重来检测不同大小的前景目标。
保留标准的超参数证明开箱即用性能。

全景头部

创造一个输出 Y，类似语义分割，又能提供实例信息。
将语义输出 X 分割为前景 Xthings 和背景 Xstuff。
由于 Xstuff 不需要实例 ID，将其直接复制到 Y。
对 Xthings 使用目标检测结果。
目标检测结果包含每个目标的类别 c 和矩形框 b。
通过 c 索引 Xthings 选择对应的切片通道，使用矩形框 b 过滤边界框之外的输出。得到和语义分割结果同宽高的 Xc,b。
将 Xc,b 和 Xstuff 堆叠起来。
Y 通道数取决于检测到的目标个数，因此，随输入变化。
例子
假设检测到三台车，网络会选择 Xc=car 进行三次切片，对应的裁剪矩形框 b1，b2，b3.与 Xstuff 堆叠获得输出 Y。因此，我们得到三个新的通道，每个对应一台车。
没有被目标检测器检测到的区域将丢弃。错误的检测对象在语义分割中具有较低的值，不会显示在最终的全景头部。
因此，一个分支的某些错误可以被另一个分支纠正。

同类重叠框，由于选择了相同的语义通道切片，在结果中具有相同的值。提出了三种不同的策略。

最高置信度策略

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

用来衡量全景分割质量的 PQ 指标，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:实例中心点预测分支

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。
此分支是与类无关的实例中心点预测，这里指定为矩形框中心点预测。
预测从像素点位置沿 x 方向和 y 方向的绝对偏移。
将预测的偏移和像素点坐标相加，计算包含该像素点的所有矩形框中心的 L2 距离解决重叠。
最小距离给出了每个像素最可能的实例。

推理和训练

联合训练

训练时，结合了四个不同的损失函数。
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。
实例中心点预测分支对每个像素使用 L1 损失 L_i。

网络使用损失项加权的组合进行训练：$L= \lambda_o(L_{FL}+L_{reg})+ \lambda_s L_s+ \lambda_i L_i$。

实验

数据集：全景分割标注的 COCO 数据集。此数据集有 118K 训练集，5K 验证集和 20K 测试集，标注有 80things 类别和 53stuff 类别。
评估标准：语义分割使用平均 IoU。目标检测使用 AP。全景分割使用 PQ。
PQ^{th} 和 PQ^{st} 得分指 things 和 stuff 数据集子集的 PQ 分数。

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

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

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。
使用 PyTorch 实现模型并在两块 Tesla V100 上训练。
与大多数相关工作使用 batch size 为 16 相比，这里使用 batch size 为 4.
因此，在编码器里冻结 BatchNorm 层并忽略微调他们的统计信息。
使用 Adam 训练，学习率 1e-5，权重衰减为 1e-4，设置 $\beta 1 = 0.9$ 和 $\beta 2 = 0.99$。
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。
在训练中只使用了左右翻转作为数据增强。

策略

最高置信度策略

为了评估网络，必须在全景头部选择一个策略解决类内重叠问题。
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。
将 0.4 的置信度阈值作为检测器输出。
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 数据集上网络使用最高置信度策略的准确性。
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：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。

最小优先策略

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，如下表。

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	28.7	33.8	21.1	48.4	**30.5**
0.25	0.75	29.1	34.0	21.7	49.0	**30.5**
0.33	0.66	**29.3**	**34.3**	21.9	49.6	30.2
0.50	**0.50**	**29.3**	33.8	22.5	50.2	29.2
0.66	0.33	29.2	33.2	*23.0*	**50.4**	27.7
0.75	0.25	28.6	32.3	22.9	50.2	26.7
0.80	0.20	28.2	31.7	22.8	50.2	25.9

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。

最近中心策略

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，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。
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，但图 6 中的定性比较显示了这种方法的优势。
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。

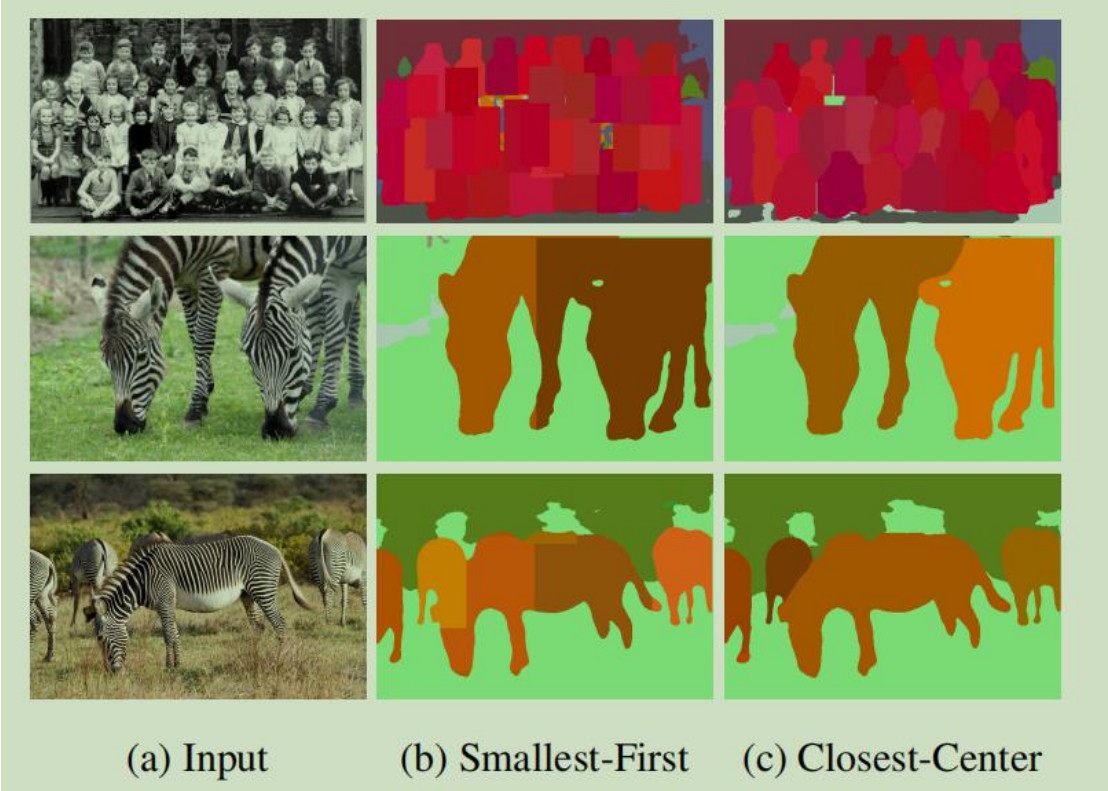


图 6：我们比较了我们最小优先和最近中心策略的全景分割。后一个实现了更合理的视觉结果。

未知预测

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，但前人的工作在某些情况下会使用未知的预测类别。

例如，删除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励。

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。

在作者的设置中，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，作者还研究了此类后处理步骤对提出方法的影响。

作者研究了未知预测的影响并移除了小的 small 的区域。

将 stuff（填充）区域的阈值设为 4096 个像素。

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，即对 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 指本文中的策略。

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。

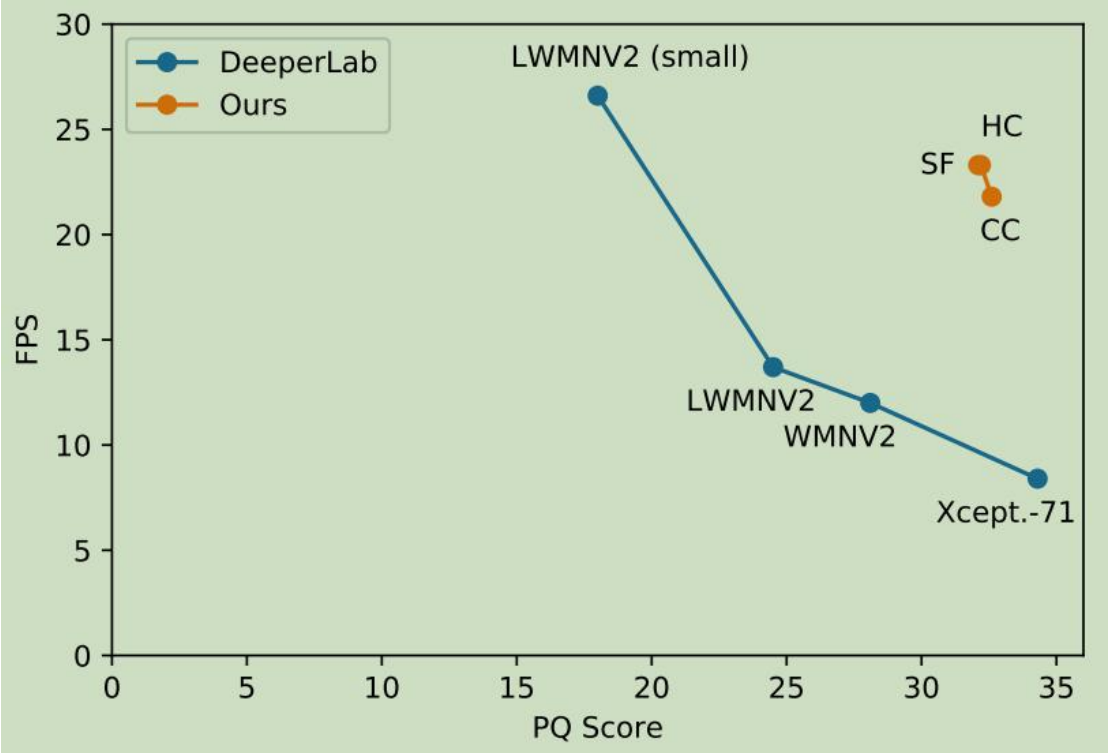


图 7：我们可视化了速度和精度的权衡。DeeperLab 在较慢的运行时间上取得了较好的结果，但速度远低于视频帧数率。该图表明，我们的网络实现了更好的权衡，因此对实际应用是有益的。

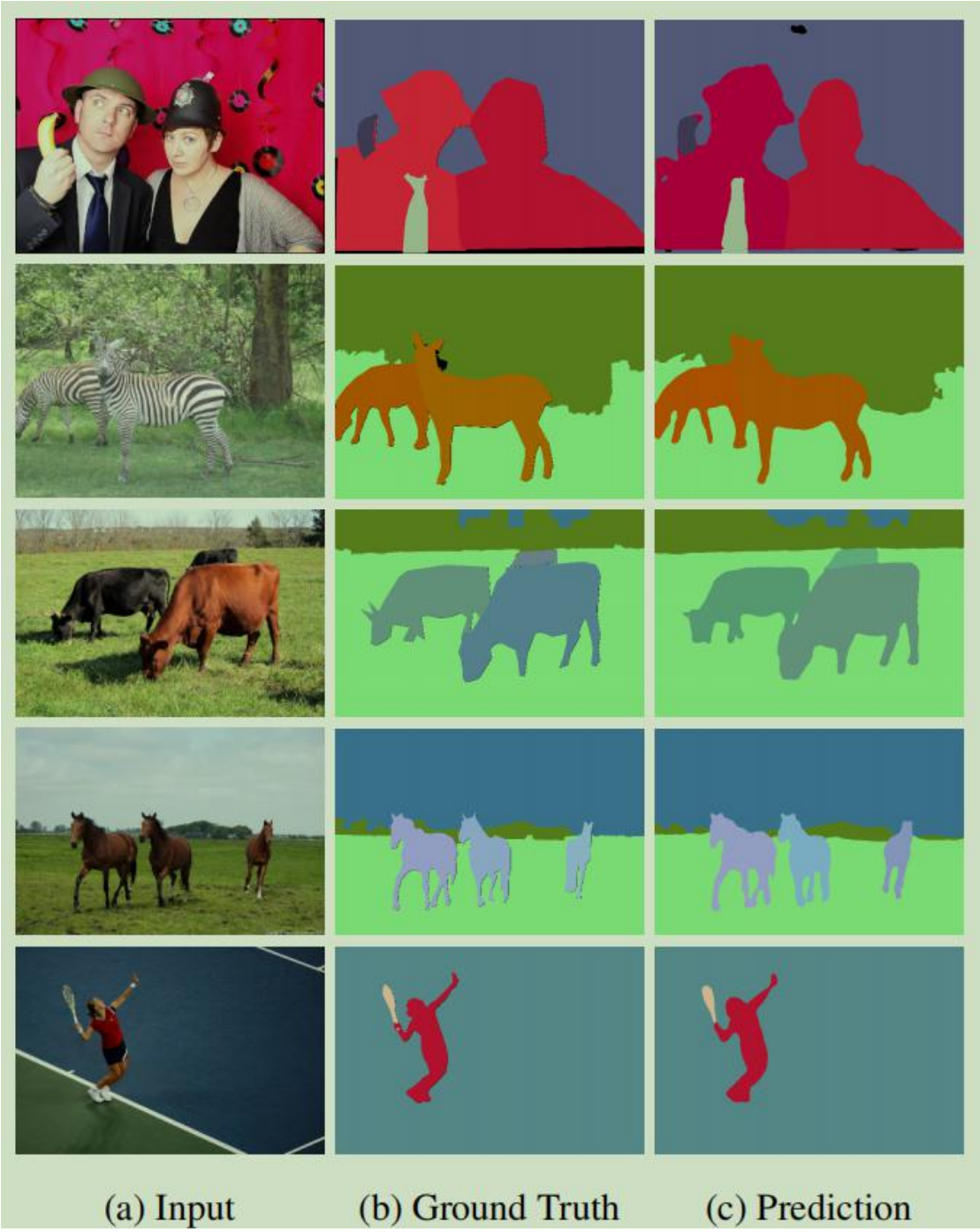


图 8：在 COCO val2017 数据集上使用最近中心策略的我们最终的网络的定性的结果。