Solo:按位置分割物体

概述

和语义分割不同，实例分割的对象数目是不固定的，使其更有挑战性。

实例分割的主流方法

1、先检测再分割，如 Mask R-CNN

2、先预测嵌入向量，再对像素进行聚类形成单个实例。

本文通过引入“实例类别”的概念从一个新的角度看待实例分割问题，根据实例的位置和尺寸，对每个实例的像素分配类别，将实例分割任务转为一个可分类的问题。这样实例分割就转为两个分类任务。论文证明了该方法更加简单、灵活，性能也更好，它可以取得与 Mask RCNN 相似的准确率，超越其它的单阶段实例分割算法。

引言

本文的出发点为：图像中物体实例的本质差异是什么？

以 MS COCO 数据集为例，它的验证集中总共有 36780 个物体,98.3%的物体对之间的中心距离超过 30 个像素点。剩下的 1.7%物体对中，40.5%的大小比例超过 1.5× 。这里，我们不考虑极端案例，比如两个物体呈× 状。总而言之，大多数情况下图像中的任意两个物体，要么它们的中心位置不同，要么其物体大小不同。

这个发现就让我们去猜测，是否可以通过中心位置和物体的大小来直接区分实例。

在相近的领域（语义分割）中，现在主导的方法就是利用一个全卷积网络，输出 N 个通道的密集预测。每个输出通道负责一个语义类别（包括背景类）。语义分割旨在区分不同的语义类别。相似地，为了区分图像中的物体实例，本文提出了“实例类别”的概念，即量化的中心位置和物体大小，使我们通过位置信息就可以分割物体，因此该方法命名为 SOLO。

SOLO 的核心想法就是通过位置和大小信息来区分物体实例。

位置：

一张图片可以分割为 S×S 个网格，这样就有了 S^2 个中心位置类别。根据物体中心的坐标，将物体实例指派给一个网格，作为其中心位置类别。本文将中心位置类别编码为通道维度，这和语义分割中的语义类别相似。每个输出通道负责一个中心位置类别，相应的通道图预测该类别的物体实例掩码。因此，结构几何信息自然地就保留在了图片宽高的空间矩阵中。

大小：

为了区分不同大小的实例，论文使用特征金字塔网络（FPN）。将不同大小的物体分配到不同的特征图层级，作为物体大小类别。这样所有的实例都区分对待，使我们可以通过“实例类别”来对物体进行分类。注意，FPN 的目的在于检测图像中不同大小的物体。

SOLO

问题分析

给定一张图片，实例分割方法需要判断它里面是否存在语义物体的实例；如果有，该方法需要返回分割掩码。

SOLO 的核心思想就是将实例分割问题重新表示为两个同时发生的子问题：类别预测和实例掩码生成。

具体点就是，该方法将输入图像划分为若干一致的网格，即 S×S。如果物体的中心落在某网格内，该网格就要负责：1）预测语义类别，2）分割该物体实例。

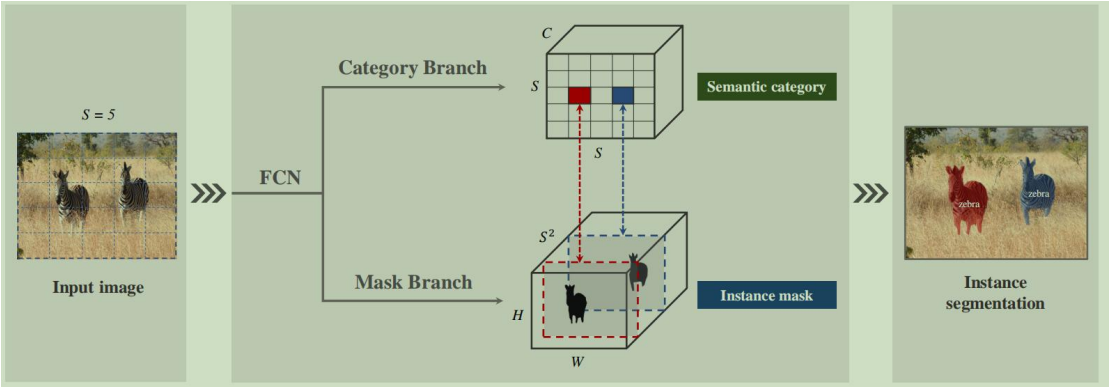


Figure 2 – SOLO framework. We reformulate the instance segmentation as two sub-tasks: category prediction and instance mask generation problems. An input image is divided into a uniform grids, *i.e.*, $S \times S$. Here we illustrate the grid with $S = 5$. If the center of an object falls into a grid cell, that grid cell is responsible for predicting the semantic category (top) and masks of instances (bottom). We do not show the feature pyramid network (FPN) here for simpler illustration.

语义类别

对每一个网格，SOLO 预测一个 C 维度的输出，表示语义类别的概率，C 是类别的个数。这些概率值取决于这些网格。如果我们将图像划分为 S×S 个网格，则输出空间为 S×S×C ，如上图所示。这样设计是基于一个假定，即每一个网格都属于一个单独的实例，因此只属于一个语义类别。在推理时，C 维度的输出表示每个实例的类别概率。

实例掩码

和语义类别预测并行，每个 positive 网格生成对应的实例分割。例如输入一张图片 I，分成 S×S 网格，总共最多预测 S^2 个实例掩码。在 3D 输出张量中，在第三个维度（通道）中直接对这些掩码进行编码。具体的，实例掩码输出维度为 $H_I \times W_I \times S_2$,第 k 个通道负责网格(i,j)的分割实例，$k=i \cdot S+j$(i 和 j 都是从 0 开始)。这样，我们就在语义类别和掩码之间构建起了一一对应的关系。

一个直接预测实例掩码的方法就是采用全卷积网络，如语义分割中的 FCN。但是，传统的卷积运算在某种程度上是空间不变的。空间不变性对某些任务非常重要，如图像分类，它能带来鲁棒性。但是相反，这里我们想要的是空间变动性，或更具体地说就是，位置敏感性，因为我们的分割掩码依赖于这些网格，必须被不同的特征通道分隔开。

本文的方法非常简单：在网络的初始阶段，我们直接将归一化后的像素坐标输入进网络，类似‘CoordConv’操作。我们构建一个和输入空间大小一样的张量，它包含归一化后的像素坐标，介于$[-1,1]$。该张量然后拼接到输入特征，传递到后面的层。将输入的坐标信息给到卷积操作，我们就给传统的 FCN 模型添加了空间功能。如果原来的特征张量大小是 H×W×D ，现在新的张量就是 H×W×(D+2) ，最后两个通道为 x−y 像素坐标。

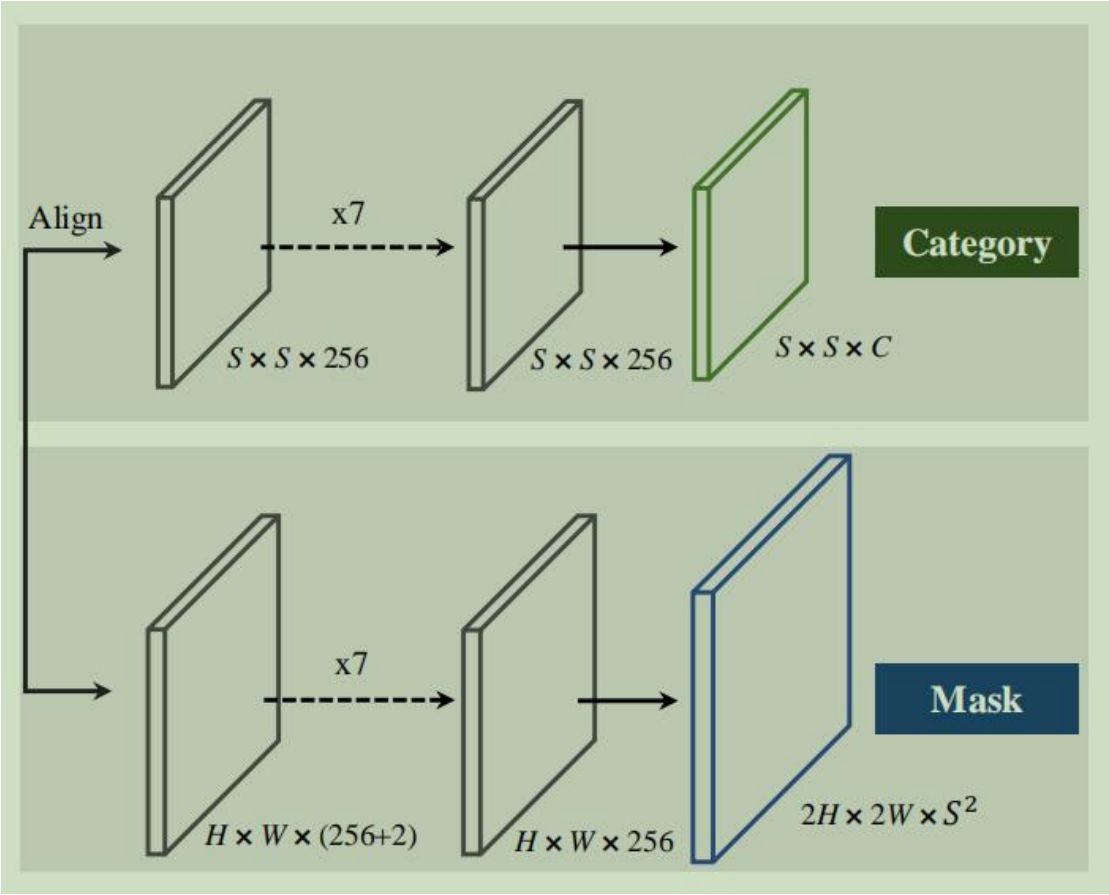
构建实例分割

在 SOLO 中，类别预测和掩码很自然地就与网格关联起来，即 k=i·S+j。于是对于每个网格，我们就可以直接得到其最终的实例分割结果。然后我们将所有的网格结果汇总起来，就可以得到该图像的实例分割结果。最后，我们使用 NMS 来得到最终的实例分割结果，而不再需要其他的后处理操作。

网络结构

SOLO 加在一个卷积主干网络后，使用 FPN，输出一个特征金字塔，每个层的特征图大小不同，通道数固定（256 维）。这些特征图作为每个预测 head 的输入：语义类别和实例掩码。在不同的层级上，head 的权重共享。在不同的金字塔中网格数可能不同。最后的 1x1 卷积不共享。

Head 结构如下图所示



SOLO 训练

标签生成：

在类别预测分支中，网络要给出 S×S 个网格中物体的类别概率。具体的，网格（i，j）如果落入任意一个真实掩码的中心区域（center region），视为正样本，反之为负样本。在最近的目标检测中中心采样是有效的，这里利用相似的技术来对掩码类别分类。给定真实掩码的质心（cx,cy）,宽 w 高 h,中心区域（center region）通过缩放系数ε控制（cx,cy，εw，εh），设置ε=0.2，每个真实掩码上平均有三个正样本。

除了实例类别的标签，对每个正样本也进行了二元分割。每张图有 S^2 个网格，所以也有 S^2 个输出掩码。对每个正样本，标注对应的二元掩码。

损失函数：

损失函数定义如下

$$L = L_{cate} + \lambda L_{mask}, \tag{1}$$

L_{cate} 是传统的 focal loss，针对语义类别分类。

L_{mask} 是掩码预测的损失函数

$$L_{mask} = \frac{1}{N_{pos}} \sum_{k} \mathbb{1}_{\{\mathbf{p}^*_{i,j}>0\}} d_{mask}(\mathbf{m}_k, \mathbf{m}^*_k), \tag{2}$$

索引 i=⌊k/S⌋,j=k mod S,网格索引顺序为从上到下，从左到右。Npos 表示正样本数量，p*和 m*表示分别表示类别和掩码。1 是指示函数，$p^*_{i,j}>0$ 时为 1，反之为 0。

在实现的时候，比较了多个 dmask 实现：二元交叉熵损失（BCE），Focal loss 和 Dice Loss。最终使用 Dice loss，因为它在训练中稳定且有效。λ设为 3。

Dice Loss 定义如下

$$L_{Dice} = 1 - D(\mathbf{p}, \mathbf{q}), \tag{3}$$

D 是 Dice 系数

$$D(\mathbf{p}, \mathbf{q}) = \frac{2\sum_{x,y}(\mathbf{p}_{x,y} \cdot \mathbf{q}_{x,y})}{\sum_{x,y} \mathbf{p}^2_{x,y} + \sum_{x,y} \mathbf{q}^2_{x,y}}. \tag{4}$$

$p_{x,y}$ 和 $q_{x,y}$ 是位于（x，y）的像素预测的掩码 p 和真实的掩码 q 的值。

推理：

推理步骤直接。输入一张图片，传入主干网络和 FPN，获得网格(i , j)的类别得分 $p_{i,j}$ 和对应的掩码 m_k，其中 k=i·S+j。首先使用置信度 0.1 过滤掉低置信度的预测，之后选取前 500 个高得分的掩码，进行 NMS 操作，为了将预测的掩码转为二值掩码，使用阈值 0.5 对预测掩码二值化。保留前 100 个实例掩码进行评估。

实验

一个网格只会激活一个实例，但一个实例可能会被多个相邻的掩码通道预测到。在推理时，使用 NMS 抑制重复的掩码。

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
two-stage:							
MNC [3]	Res-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [10]	Res-101-C5	29.2	49.5	–	7.1	31.3	50.0
Mask R-CNN [7]	Res-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN* [2]	Res-50-FPN	36.8	59.2	39.3	17.1	38.7	52.1
Mask R-CNN* [2]	Res-101-FPN	38.3	61.2	40.8	18.2	40.6	54.1
one-stage:							
TensorMask [2]	Res-50-FPN	35.4	57.2	37.3	16.3	36.8	49.3
TensorMask [2]	Res-101-FPN	37.1	59.3	39.4	17.4	39.1	51.6
YOLACT [1]	Res-101-FPN	31.2	50.6	32.8	12.1	33.3	47.1
PolarMask [27]	Res-101-FPN	30.4	51.9	31.0	13.4	32.4	42.8
ours:							
SOLO	Res-50-FPN	36.8	58.6	39.0	15.9	39.5	52.1
SOLO	Res-101-FPN	37.8	59.5	40.4	16.4	40.6	54.2
SOLO	Res-DCN-101-FPN	40.4	62.7	43.3	17.6	43.3	58.9

Table 1 – Instance segmentation mask AP on COCO `test-dev`. All entries are *single-model* results. Here we adopt the "6×" schedule (72 epochs) for better results. Mask R-CNN* is the improved version in [2].

与现有的实例分割比较，性能更好。

消融实验

网格数目：

grid number	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
12	27.2	44.9	27.6	8.7	27.6	44.5
24	29.0	47.3	29.9	10.0	30.1	45.8
36	28.6	46.3	29.7	9.5	29.5	45.2
Pyramid	35.8	57.1	37.8	15.0	38.7	53.6

Table 2 – The impact of **grid number and FPN**. FPN (Table 3) significantly improves the performance thanks to its ability to deal with varying sizes of objects.

网格数目对单张特征图性能的影响。特征图是融合 resnet 的 C3，C4，C5 的输出。12 到 24 有提升，同时展示了 FPN 层对性能的巨大提升。

多层预测：

pyramid	P2	P3	P4	P5	P6
re-scaled stride	8	8	16	32	32
grid number	40	36	24	16	12
instance scale	<96	48∼192	96∼384	192∼768	≥ 384

Table 3 – we use five **FPN pyramids** to segment objects of different scales. The grid number increases for smaller instances due to larger existence space.

使用五层 FPN 来分割不同尺寸的物体。通过使用多层预测，达到了上表中的 35.8AP。

CoordConv：

#CoordConv	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
0	32.2	52.6	33.7	11.5	34.3	51.6
1	35.8	57.1	37.8	15.0	38.7	53.6
2	35.7	57.0	37.7	14.9	38.7	53.3
3	35.8	57.4	37.7	15.7	39.0	53.0

Table 4 – Conv vs. CoordConv. CoordConv can considerably improve AP upon standard convolution. Two or more layers of CoordConv are not necessary.

标准的卷积有一定的空间敏感性，当连接额外的坐标通道使卷积访问自己的输入坐标时，获得了 3.6AP 的提升。更多的 CoordConv 没有带来显著的提升，没有必要。

损失函数：

mask loss	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
BCE	30.0	50.4	31.0	10.1	32.5	47.7
FL	31.6	51.1	33.3	9.9	34.9	49.8
DL	35.8	57.1	37.8	15.0	38.7	53.6

Table 5 – Different loss functions may be employed in the mask branch. The Dice loss (DL) leads to best AP and is more stable to train.

Focal Loss 比 Binary Cross Loss 效果好，这是因为一个实例掩码大部分像素是背景，Focal Loss 可以平衡正负样本。当时 Dice Loss 取得了最好的效果，且不用调整超参数。Dice Loss 将像素视为一个整体自动平衡前景和背景的像素。

头部结构的深度：

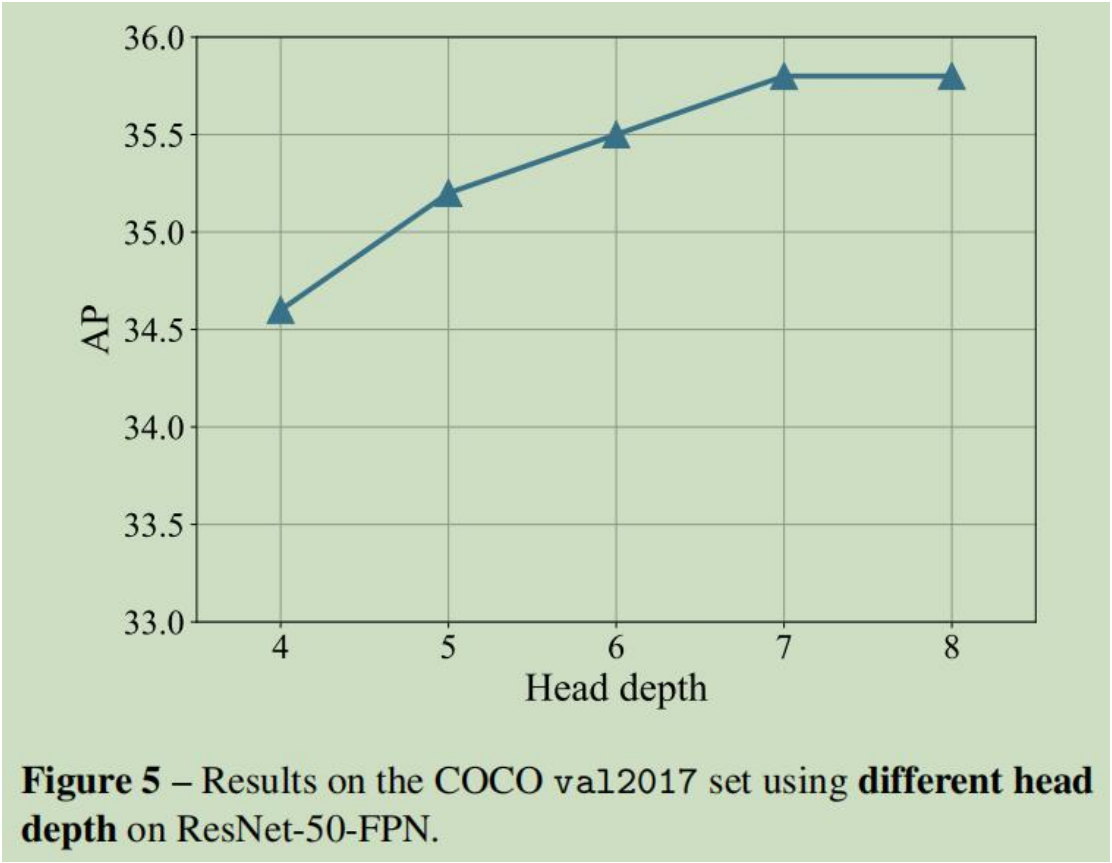


Figure 5 – Results on the COCO `val2017` set using **different head depth** on ResNet-50-FPN.

将头部结构深度从 4 到 7，提升 1.2AP。7 以后表现稳定，选择 7 层的深度。

之前的网络（例如 maskrcnn）采用 4 层卷积。在 SOLO 中，由于掩码分割添加了坐标，需要足够的表现力来学习这种变换。

SOLO-512：

	backbone	AP	AP_{50}	AP_{75}	fps
SOLO	ResNet-50-FPN	36.0	57.5	38.0	12.1
SOLO	ResNet-101-FPN	37.1	58.7	39.4	10.4
SOLO-512	ResNet-50-FPN	34.2	55.9	36.0	22.5
SOLO-512	ResNet-101-FPN	35.0	57.1	37.0	19.2

Table 6 – SOLO-512. SOLO-512 uses a model with smaller input size (shorter image size of 512 instead of 800). All models are evaluated on `val2017`. Here the models are trained with "6×" schedule.

使用一个更小的输入分辨率（800 替换为 512）。在 V100 显卡上 Fps 达到了 22.5，有实时实例分割应用的潜力。

Decoupled SOLO：

给定一个网格数，如 S=20，输出通道就有 S^2=400 个。这里有很大的冗余，一张图片中不太可能有那么多实例。这里介绍一个等价且更高效的 SOLO 变体，Decoupled SOLO。

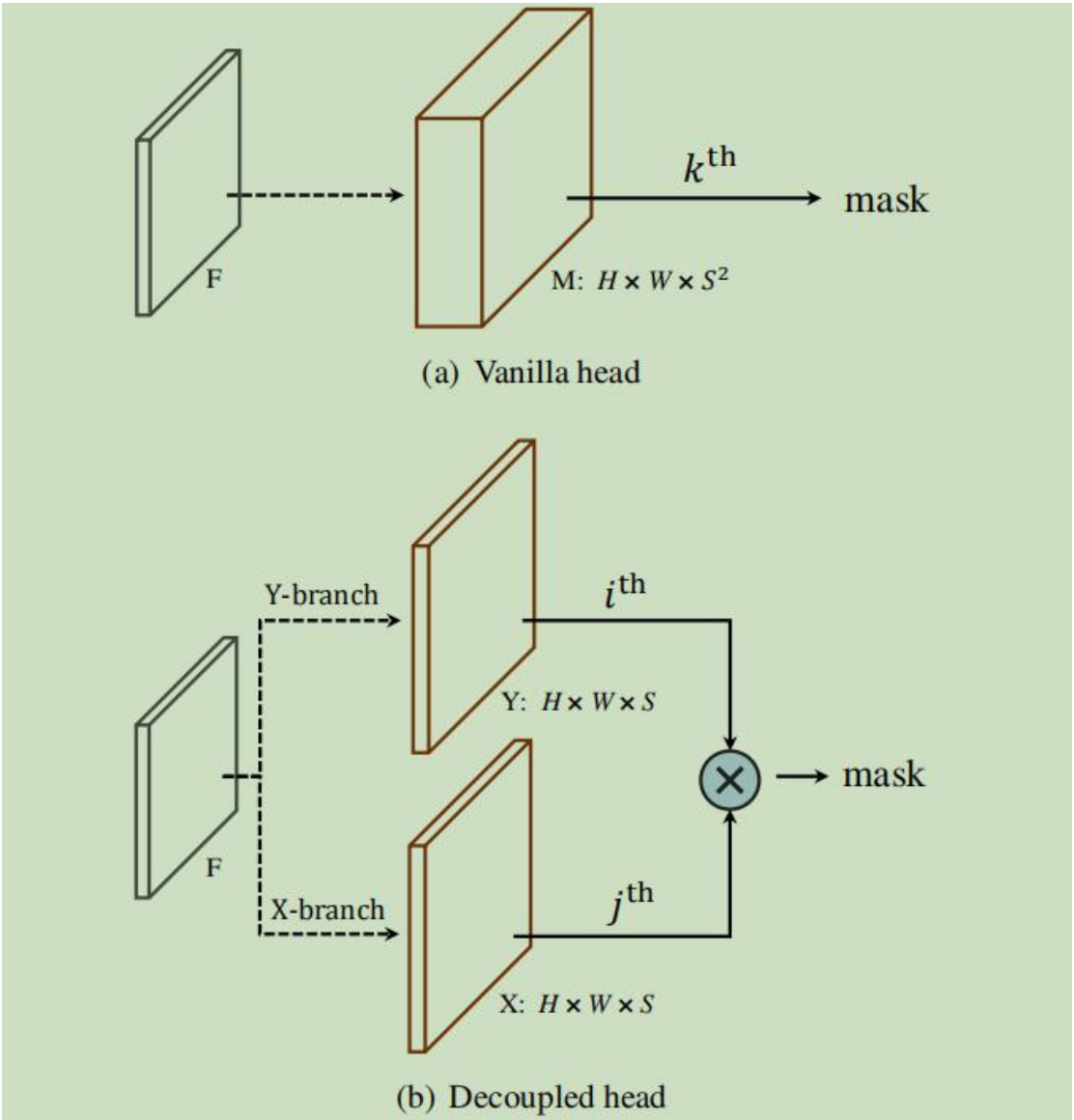


Figure 7 – Decoupled SOLO head. F is input feature. Dashed arrows denote convolutions. $k = i \cdot S + j$. '$\otimes$' denotes element-wise multiplication.

原始输出张量 $M \in R^{HxWxS2}$ 被替换为两个输出张量 $X \in R^{HxWxS}$ 和 $Y \in R^{HxWxS}$，对应两个坐标轴。

对于一个位于网格 (i,j) 中的目标，原来的 SOLO 在第 k 个通道输出张量 M，k=i·S+j。在 Decoupled SOLO 中，被定义为两个通道的元素相乘，

$$\mathbf{m}_k = \mathbf{x}_j \otimes \mathbf{y}_i, \tag{5}$$

其中 xj 和 yi 分别表示 X 的第 j 个通道特征图和 Y 的第 i 个通道特征图。

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Vanilla SOLO	35.8	57.1	37.8	15.0	38.7	53.6
Decoupled SOLO	35.8	57.2	37.7	16.3	39.1	52.2

Table 7 – Vanilla head vs. Decoupled head. The models are trained with"3×" schedule and evaluated on `val2017`.

上表显示使用相同超参数的两种 SOLO 性能相同。