

语义分割

✓ 图像分割

✎ 分割任务就是在原始图像中逐像素的找到你需要的家伙!



(检测任务)

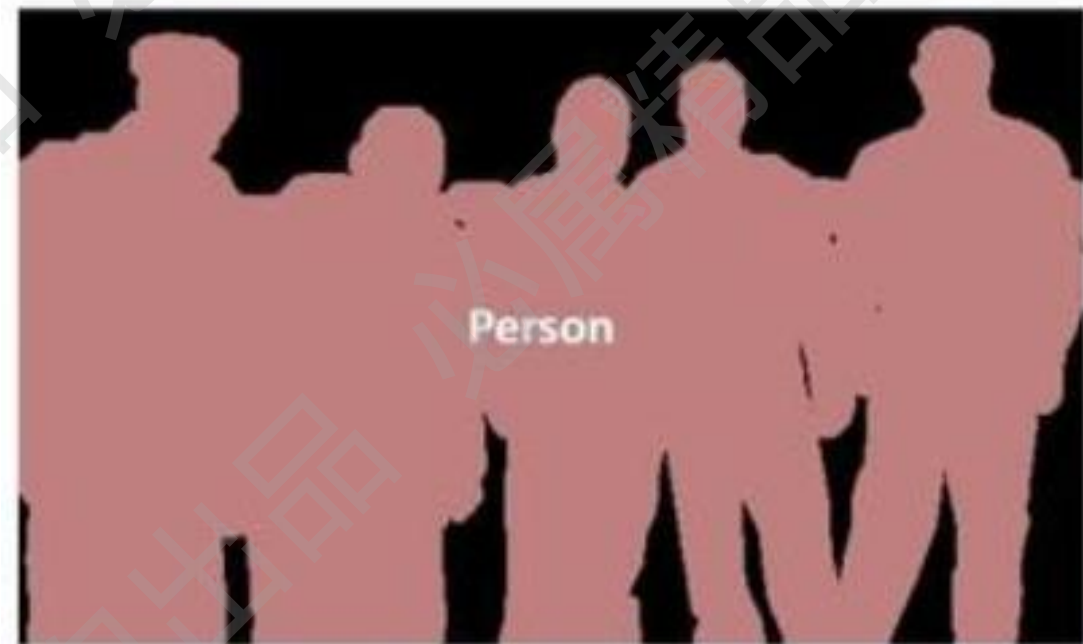


(分割任务)

语义分割

✓ 图像分割

📌 语义分割就是把每个像素都打上标签（这个像素点是人，树，背景等）
（语义分割只区分类别，不区分类别中具体单位）



语义分割

✓ 实例分割

📌 实例分割不光要区别类别，还要区分类别中每一个个体



语义分割

✓ 损失函数:

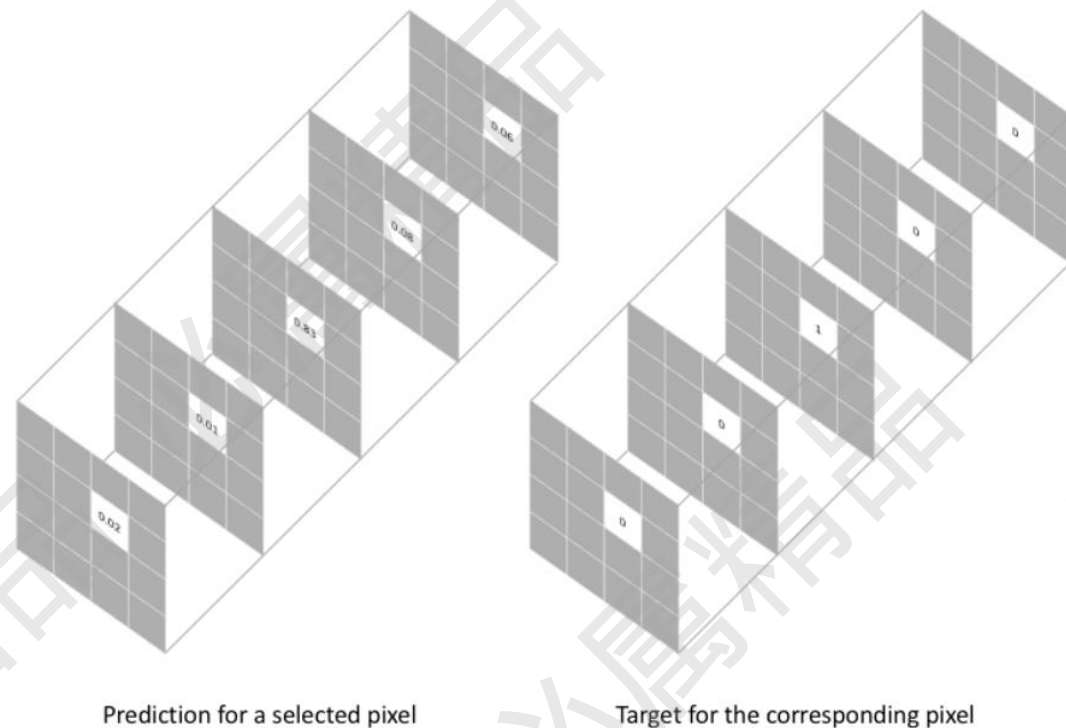
✎ 逐像素的交叉熵:

✎ 还经常需要考虑样本均衡问题

✎ 交叉熵损失函数公式如下:

$$\text{pos_weight} = \frac{\text{num_neg}}{\text{num_pos}}$$

$$\text{loss} = -\text{pos_weight} \times y_{\text{true}} \log(y_{\text{pred}}) - (1 - y_{\text{true}}) \log(1 - y_{\text{pred}})$$



语义分割

✓ Focal loss

✎ 样本也由难易之分，就跟玩游戏一样，难度越高的BOSS奖励越高

$$-(1 - y_{pred})^\gamma \times y_{true} \log(y_{pred}) - y_{pred}^\gamma \times (1 - y_{true}) \log(1 - y_{pred})$$

✎ Gamma通常设置为2，例如预测正样本概率0.95, $(1 - 0.95)^2 = 0.0025$
如果预测正样本概率0.4, $(1 - 0.5)^2 = 0.25$ （相当于样本的难易权值）

$$-\alpha(1 - y_{pred})^\gamma \times y_{true} \log(y_{pred}) - (1 - \alpha)y_{pred}^\gamma \times (1 - y_{true}) \log(1 - y_{pred})$$

（再结合样本数量的权值就是Focal Loss）

语义分割

✓ IOU计算

📌 多分类任务时: $iou_dog = 801 / true_dog + predict_dog - 801$

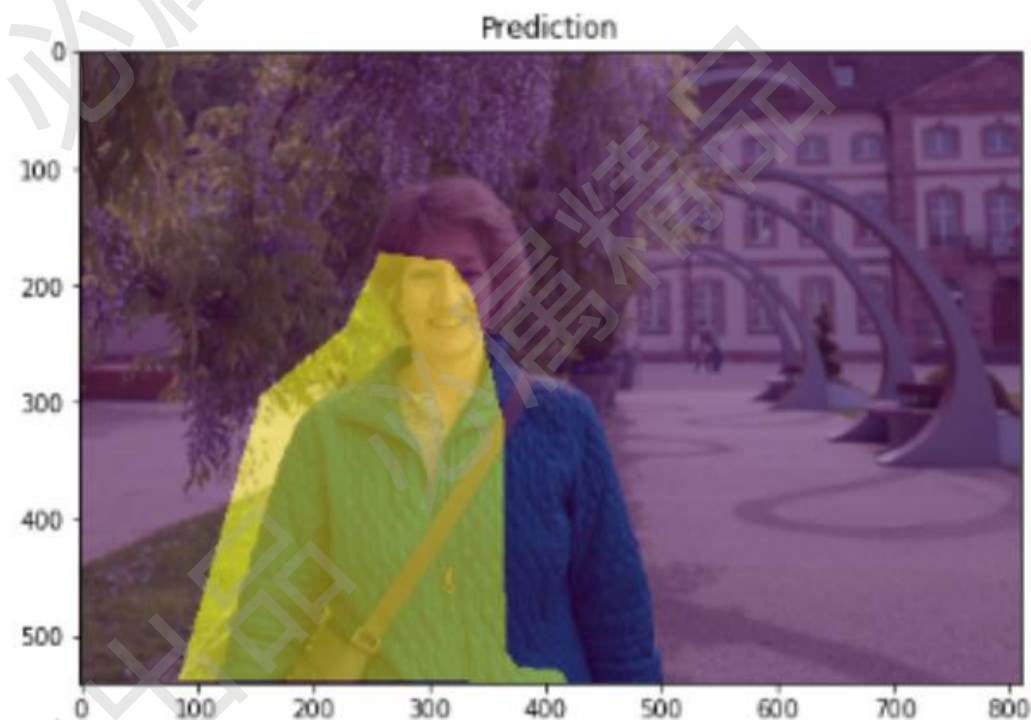
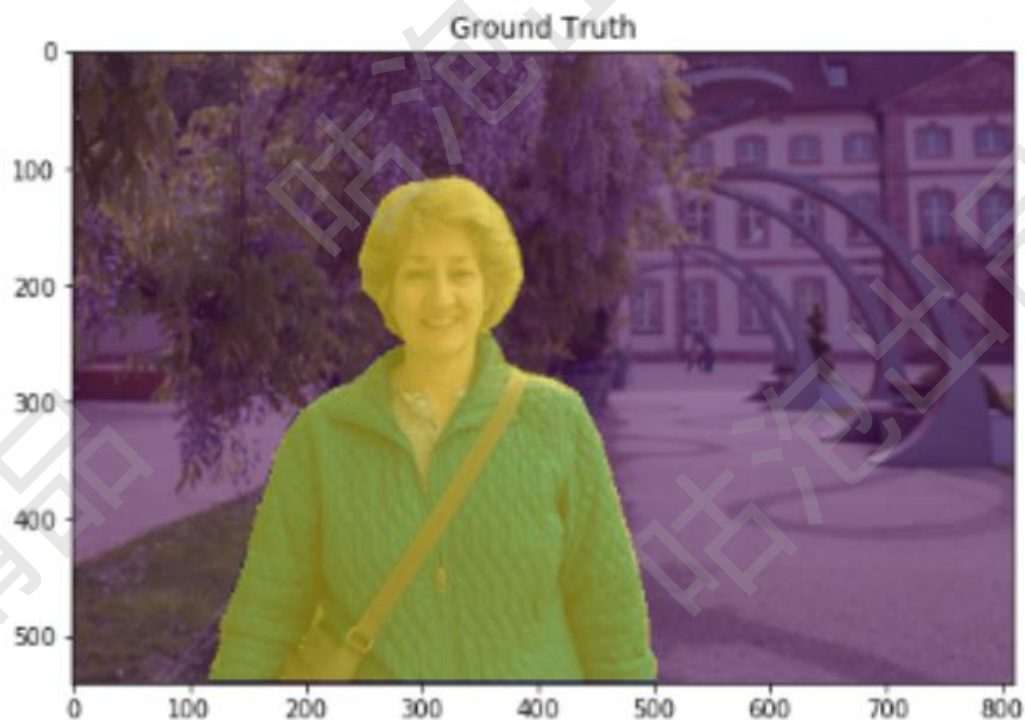
True Class	airplane	923	4	21	8	4	1	5	5	23	6
	automobile	5	972	2					1	5	15
	bird	26	2	892	30	13	8	17	5	4	3
	cat	12	4	32	826	24	48	30	12	5	7
	deer	5	1	28	24	898	13	14	14	2	1
	dog	7	2	28	111	18	801	13	17		3
	frog	5		16	27	3	4	943	1	1	
	horse	9	1	14	13	22	17	3	915	2	4
	ship	37	10	4	4		1	2	1	931	10
	truck	20	39	3	3			2	1	9	923
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Predicted Class											

True Class	airplane	923	4	21	8	4	1	5	5	23	6	
	automobile	5	972	2					1	5	15	
	bird	26	2	892	30	13	8	17	5	4	3	
	cat	12	4	32	826	24	48	30	12	5	7	
	deer	5	1	28	24	898	13	14	14	2	1	
	dog	7	2	28	111	18	801	13	17		3	
	frog	5		16	27	3	4	943	1	1		
	horse	9	1	14	13	22	17	3	915	2	4	
	ship	37	10	4	4		1	2	1	931	10	
	truck	20	39	3	3			2	1	9	923	
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
		Predicted Class										

语义分割

✓ MIOU指标:

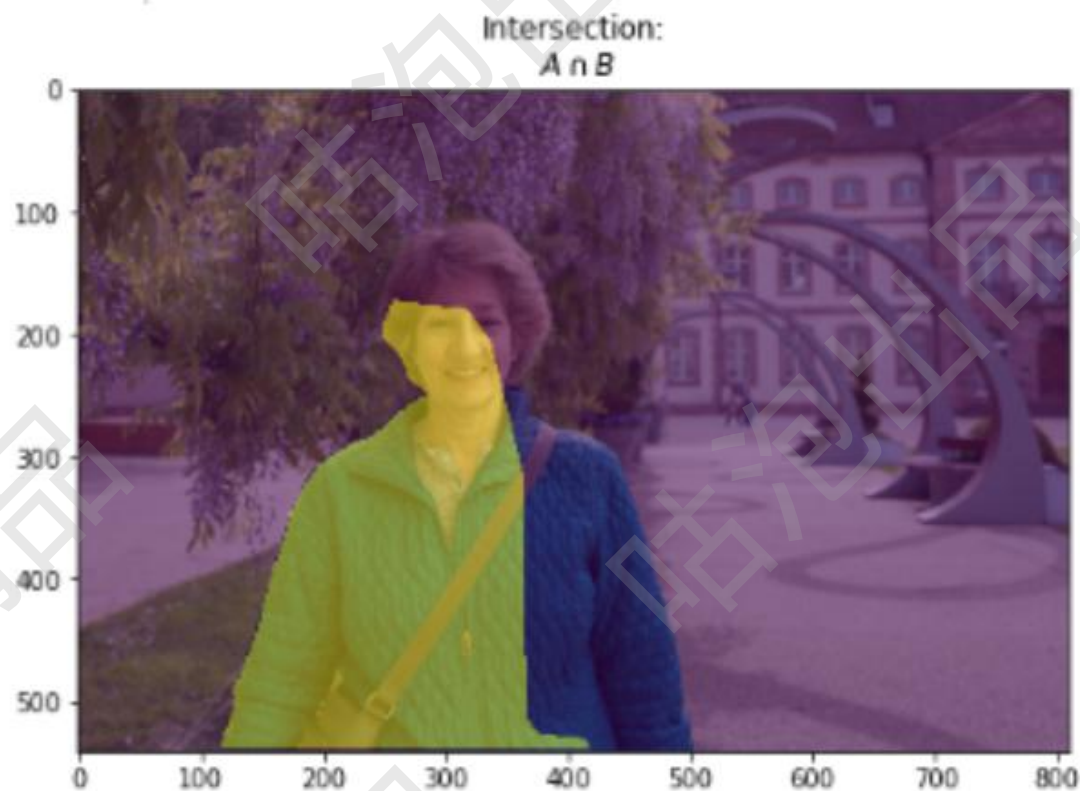
✎ IoU(Intersection over Union, 交并比)



语义分割

✓ MIOU指标:

✎ MIOU就是计算所有类别的平均值, 一般当作分割任务评估指标



语义分割

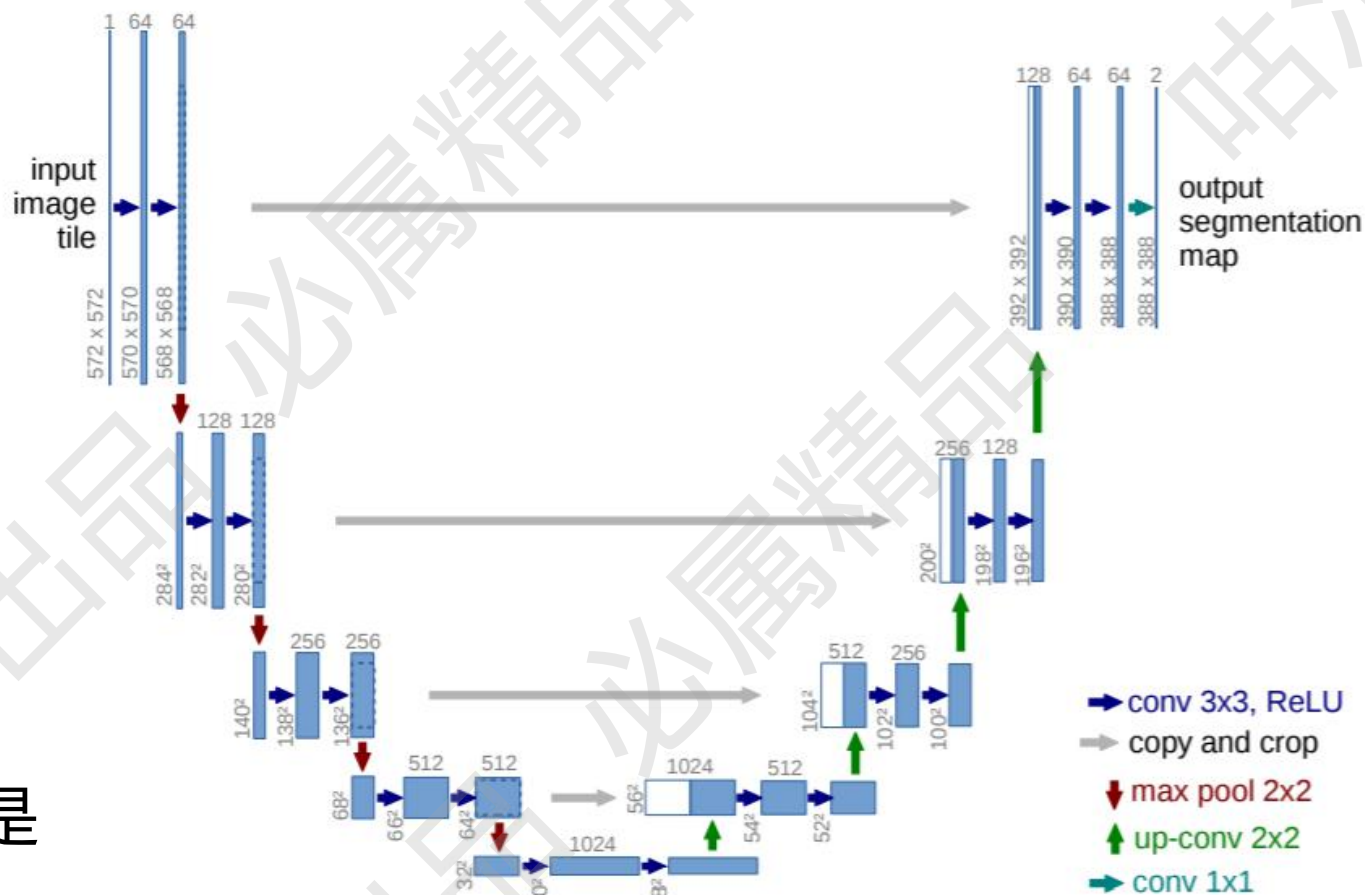
✓ U-net

📎 整体结构:

📎 概述就是编码解码过程

📎 简单但是很实用, 应用广

📎 起初是做医学方向, 现在也是



语义分割

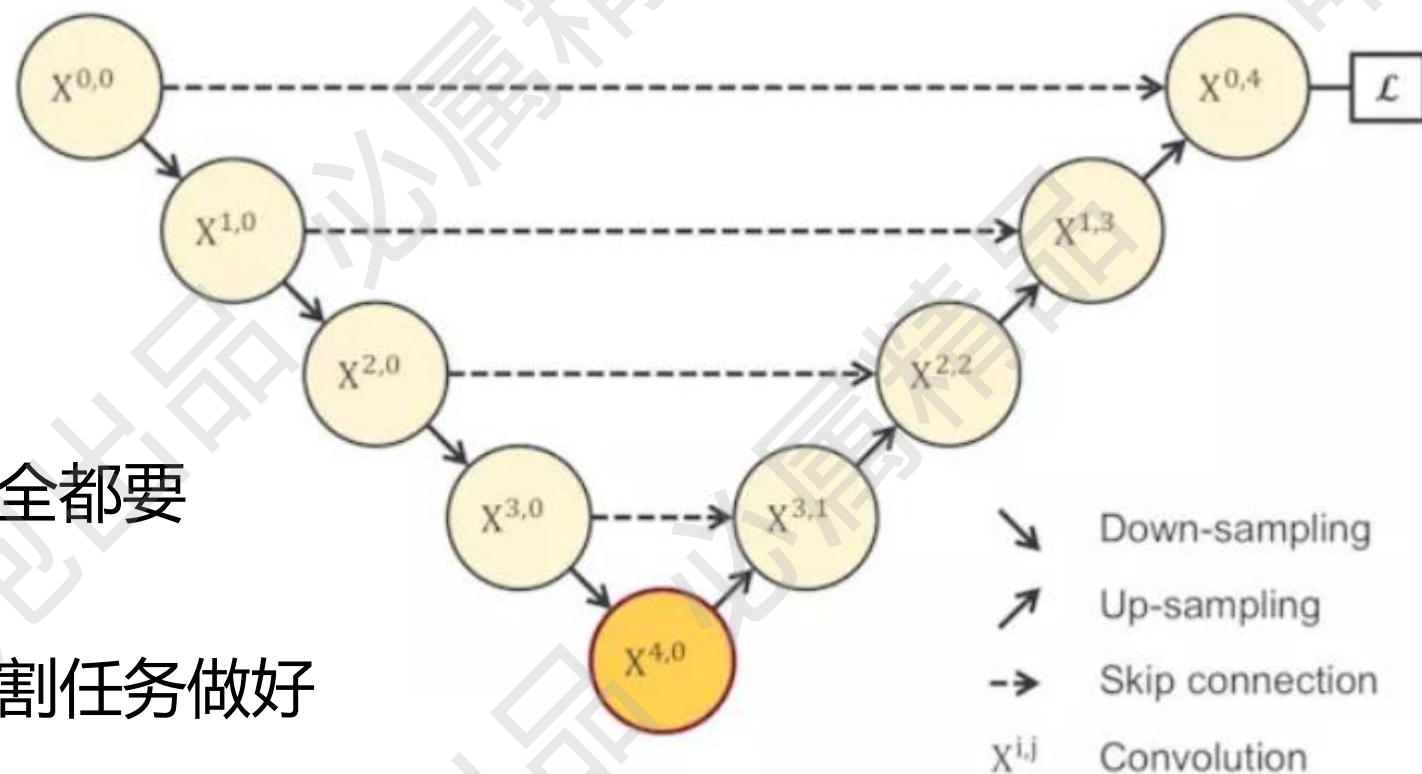
✓ U-net

✎ 主要网络结构:

✎ 还引入了特征拼接操作

✎ 以前我们都是加法, 现在全都要

✎ 这么简单的结构就能把分割任务做好



语义分割

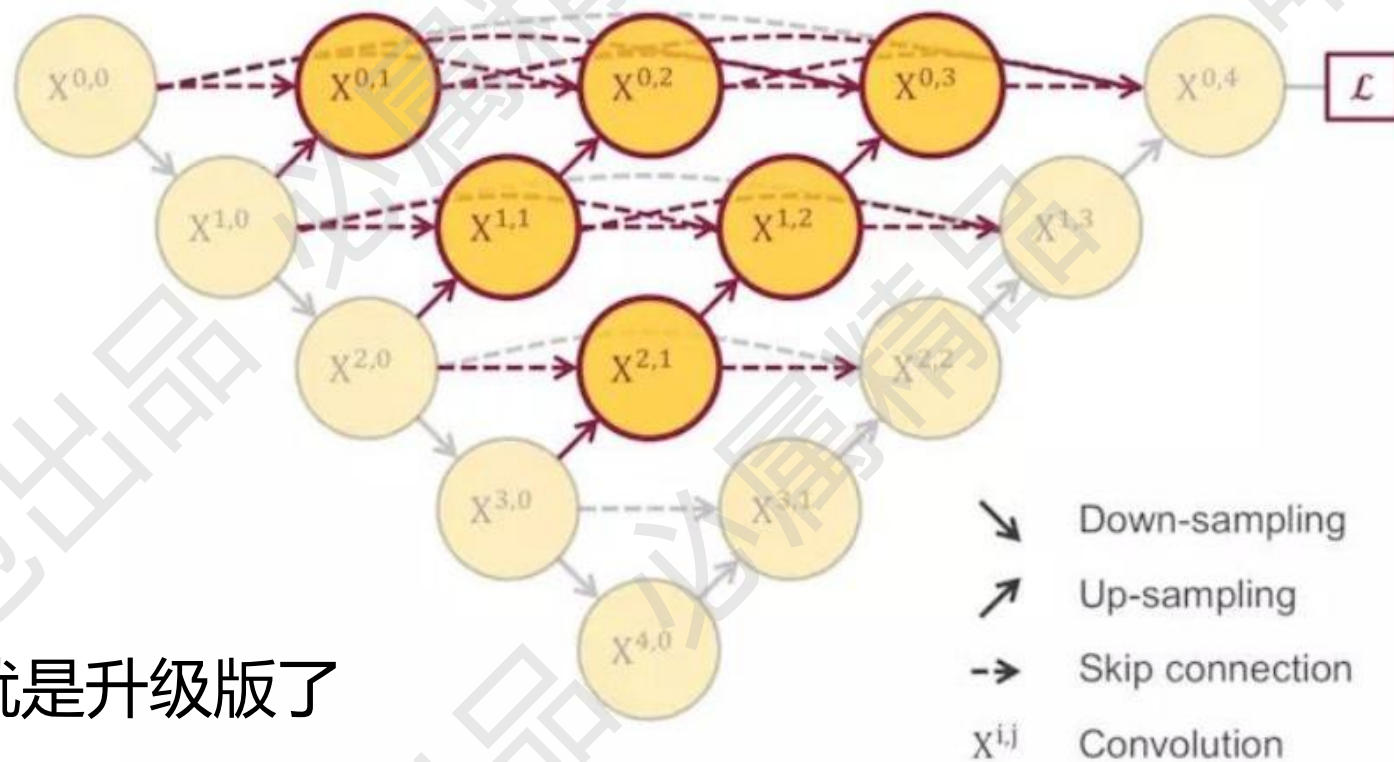
✓ U-net++

✎ 整体网络结构：

✎ 特征融合，拼接更全面

✎ 其实跟densenet思想一致

✎ 把能拼能凑的特征全用上就是升级版了



语义分割

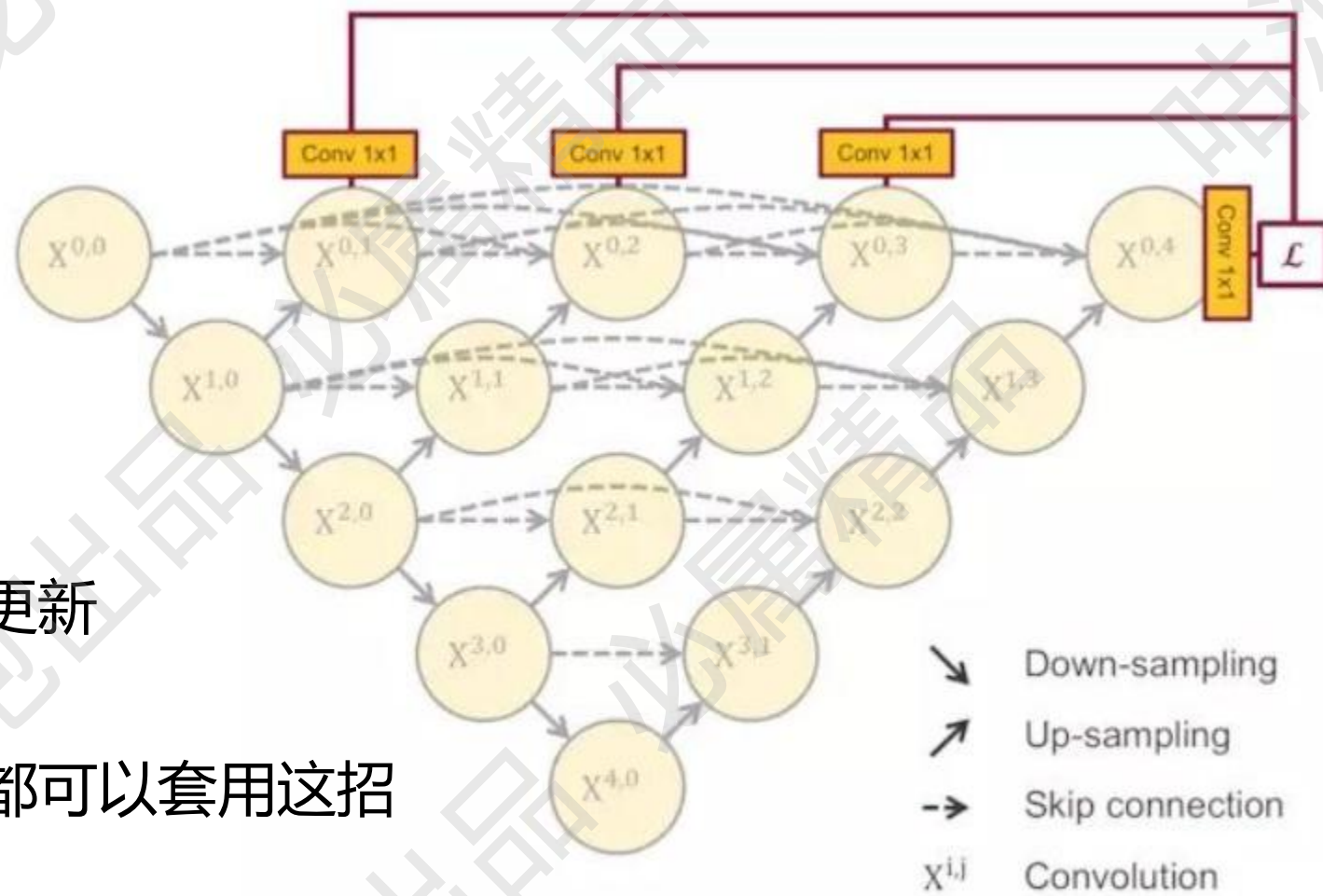
✓ U-net++

✎ Deep Supervision :

✎ 也是很常见的事, 多输出

✎ 损失由多个位置计算, 再更新

✎ 现在来看, 很多视觉任务都可以套用这招



语义分割

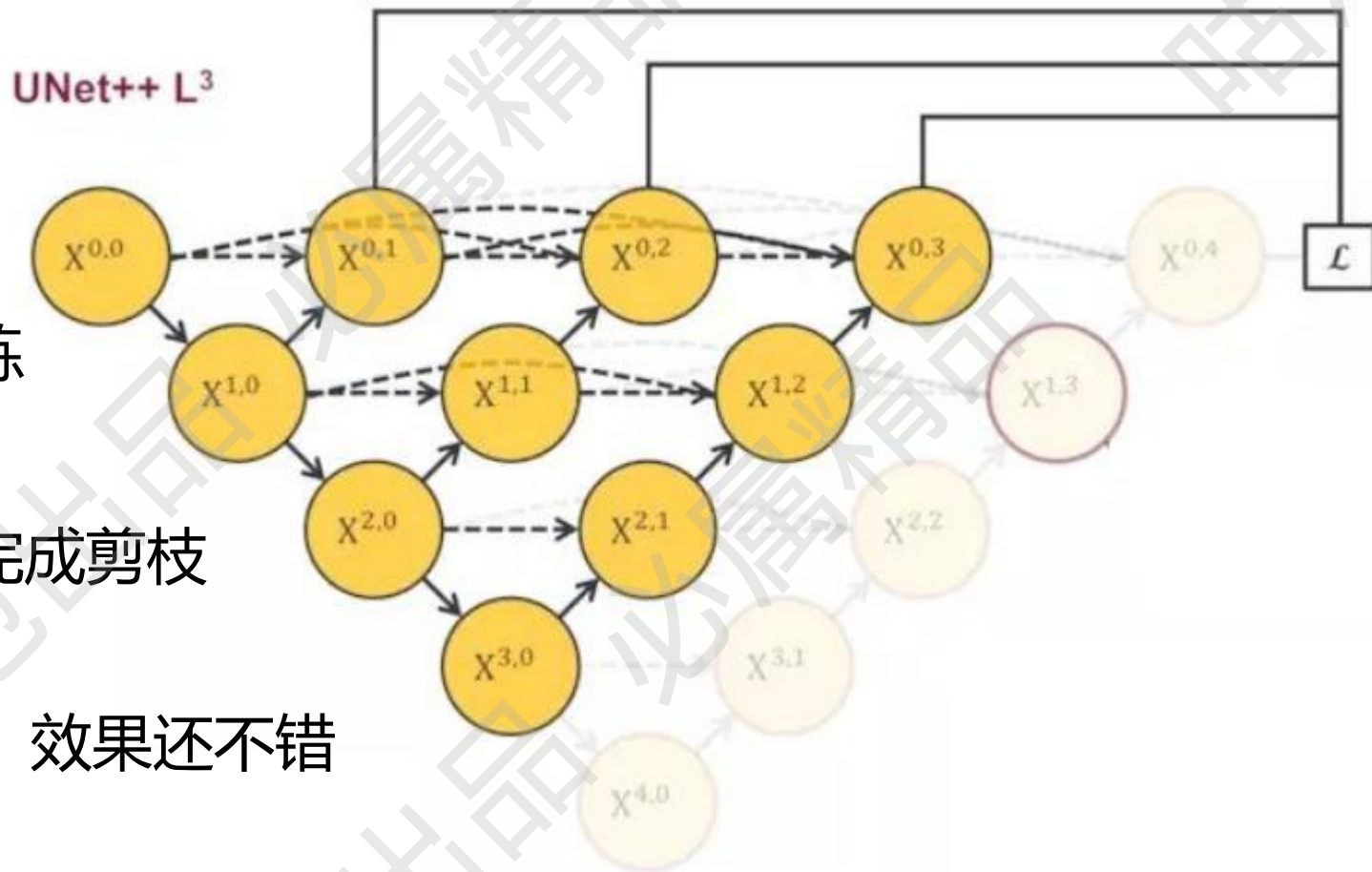
✓ U-net++

✎ 可以更容易剪枝:

✎ 因为前面也单独有监督训练

✎ 可以根据速度要求来快速完成剪枝

✎ 训练的时候同样会用到L4, 效果还不错



语义分割

✓ U-net+++ (了解下就行)

✎ 不同的max pool整合低阶特征
(X_1 和 X_2 , 轮廓之类的)

✎ 上采样整合高阶特征
(感受野大的, 全局的)

✎ 各层统一用卷积得到64个特征图

✎ $5 \times 64 = 320$, 最终组合得到全部特征

