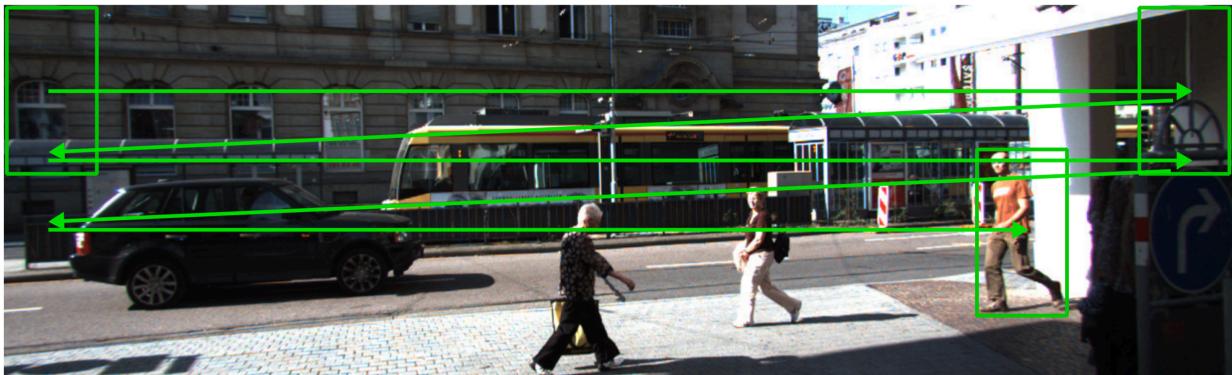


## Sliding-Window Object Detection

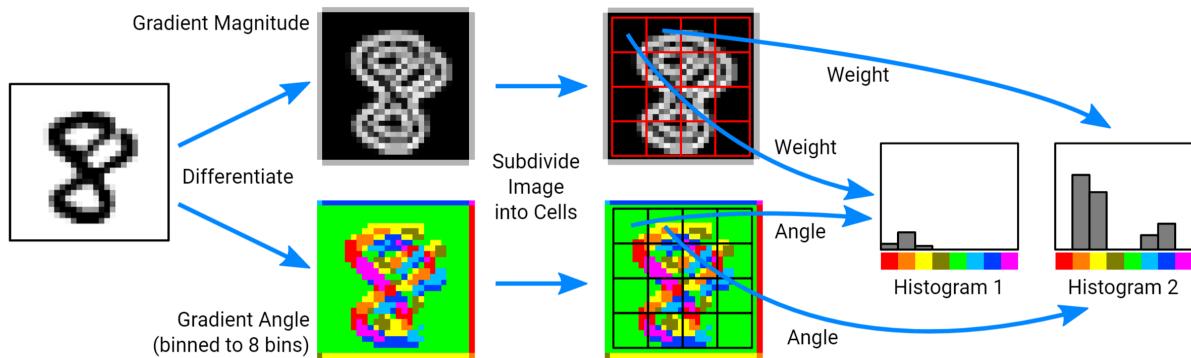
- Sliding-Window Object Detection



- 在图像上滑动一个固定大小的窗口，对于每个窗口提取其特征
- 将每个窗口截下来的 crop，使用例如 SVM 或随机森林等算法，对其进行分类，返回一个置信度
- 由于检测对象的大小、形状等特征在图像中是未知的——因此，我们可能需要改变窗口的尺寸并运行很多轮这样的过程，由此可见这样会导致计算量非常庞大
- 并且我们还需要施加 non-maxima 抑制——由于在滑动到正确的位置前后会有部分内容增加返回结果的置信度，我们需要对这些值进行抑制，防止根据阈值筛选结果后这些结果也出现，也即出现很多的 False Positive 结果

下面我们来具体考虑这个问题——首先提出的问题是，我们该在这个 crop 中使用什么特征作为判断依据？很显然，RGB 像素并不是一个好主意

- 最简单的选择当然是 RGB 像素空间，然而它存在两个方面的问题
  - 不具有视角不变性 (viewpoint invariance)
    - 图像从不同视角拍摄时，RGB值会发生显著变化
  - 不具有光照不变性 (illumination invariance)
    - 光线强度的变化会显著影响像素值
- 因此，Dalal and Triggs 等人提出一种新的表示方法——
  - HoG (Histograms of Oriented Gradients) [Dalal and Triggs: Histograms of Oriented Gradients for Human Detection. CVPR, 2005.](#)



- 主要思想

- 将图像的 patch 区域表示为梯度方向的直方图，梯度方向的权重由梯度幅值 (gradient magnitude) 决定 (这一点，和 SIFT 方法非常相似)
- 具体过程如图所示
- 首先，对图片计算微分，得到图片的梯度幅值 (gradient magnitude) 和梯度方向 (gradient angle)，在这里我们将梯度方向量化为 8 个不同的方向，形成 8 个用不同颜色表示的 bin
- 然后将图片划分为若干网格，例如这里使用  $4 \times 4$  的网格进行划分
- 最后以梯度幅值作为对应梯度方向的权重，计算出直方图，也即图像的 HoG 特征描述符
- 特点
- HoG 对于较小程度的变换 (平移、缩放、旋转、透视等) 具有不变性

### 微分 (Differentiate)

由于 RGB 图像是离散的，一般会采用一些近似的思想来计算图像的微分，常用 Sobel 算子 (水平+竖直方向)

### 梯度幅值 (Gradient Magnitude)

考虑为所有方向梯度的平方和的平方根，例如  $g = \sqrt{G_x^2 + G_y^2}$

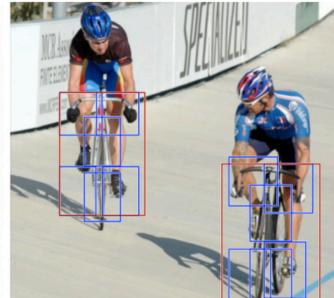
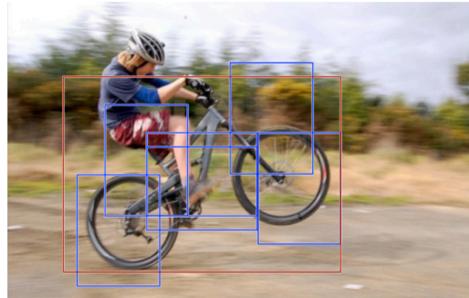
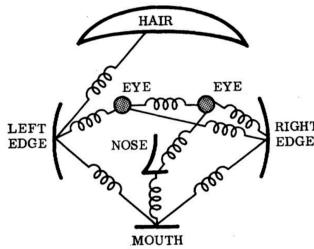
### 梯度方向 (Gradient Angle)

梯度本身的方向，我们可以将其量化到若干个方向，例如  $0^\circ \sim 90^\circ, 90^\circ \sim 180^\circ, 180^\circ \sim 270^\circ, 270^\circ \sim 360^\circ$ ，这样可以将梯度量化到 4 个方向上

HoG 相比 RGB 已经有了很大的改进，然而对于大幅度的变换，它仍然不具有不变性

- Part Based Models [Object Detection with Discriminatively Trained Part Based Models. Felzenszwalb, Girshick, McAllester and Ramanan. PAMI, 2010.](#)

○



- 主要思想

- 构建目标的各个部分的分布模型来进行目标检测——每个目标（例如人、车）可以被分解为若干个重要的部件，例如人的头部、眼睛、身体等部分；这些部件通过相对位置、形状、大小等特征来描述目标的整体形态
- 基于部件的模型 (Part Based Models) 能够很好的适应目标的**非刚性变形 (Non-rigid Deformations)**，它具有的不变性允许更大幅度的变换
- 然而，它导致推理速度很慢——你需要对每个部件的存在和位置进行推理，大幅增加计算量；而且尽管它在处理非刚性变形时表现优异，但是与 HoG 模型相比，效果提升并不显著，特别是在某些简单的目标检测任务中

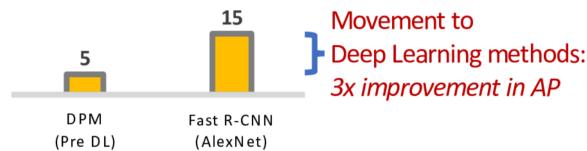
- Recognition 的转折点——深度学习

- 传统的方法并没有得到突破，它们只被应用于处理了一些，对安全需求并不是很高的任务（例如面部识别）
- 为什么呢？我们来总结一下原因
  - CV 的特征通常都非常难以手工设计
  - 我们仍然不清楚，一个好的表示方式应该是什么样的
  - 另一方面，sliding window 方法在推理时实际很慢
  - 使用复杂的模型，例如 part-based 模型会更慢
  - 计算结果通常不能有效地重用——大大减少了优化计算的可能性
- 然而，深度学习的出现改变了这一切

- 手工设计表示 vs 学习方法表示

Past  
(best circa  
2012)

Early  
2015

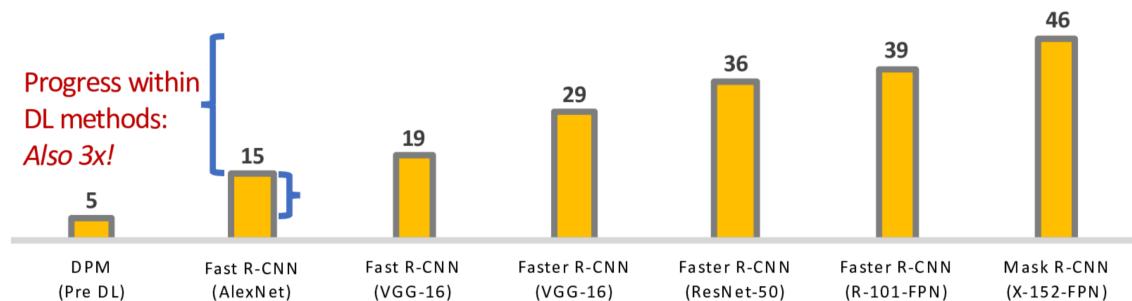


- ■ 在 2012 年提出的，采用 Deformable part based model (可变性的基于部件的模型) 加上 HoG 表示以及图模型推理，已经是传统方法可以达到的最高的水准
- 2015 年提出的首个基于深度学习的模型，在 AP 上是传统最优方法的 3 倍

Past  
(best circa  
2012)

Early  
2015

2.5 years



- ■ 而在 2015—2017 年 (准确的说是 2.5 年) 以来，基于深度学习的模型结果又提高了 3 倍，并且自 2017 年以来，性能还在持续增长

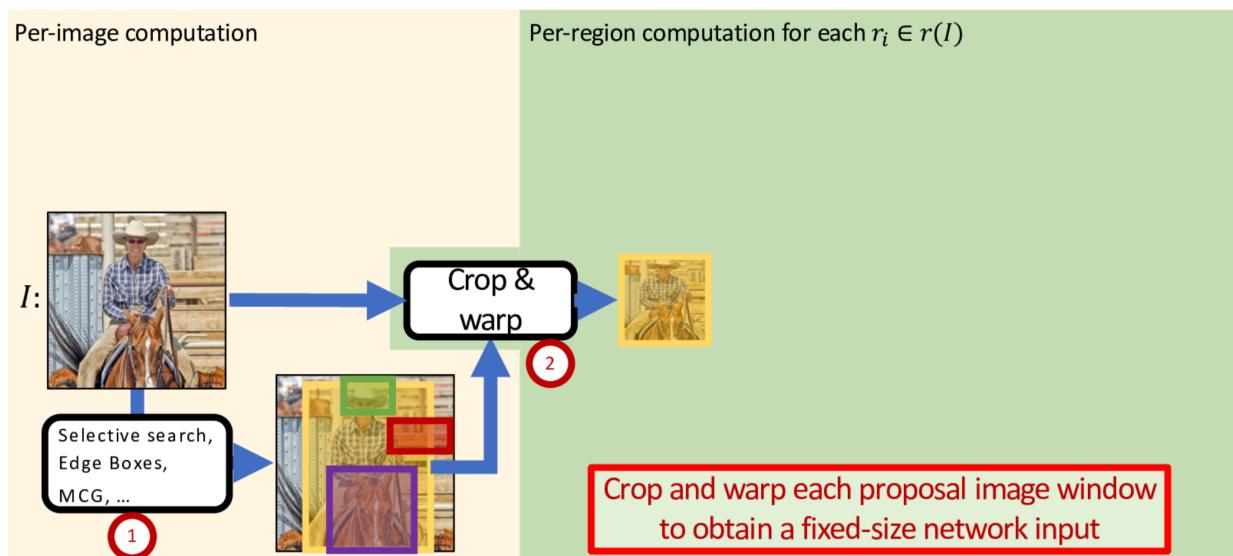
## Object Detection with Deep Neural Networks —— (Slide Credits: Ross Girshick)

在这里，我们快速展示一下 Object Detection 领域最具代表性的一些工作的 overview；该领域的主要推动者之一——Ross Girshick，这部分 PPT 素材主要取自他的一篇教程，向他致谢

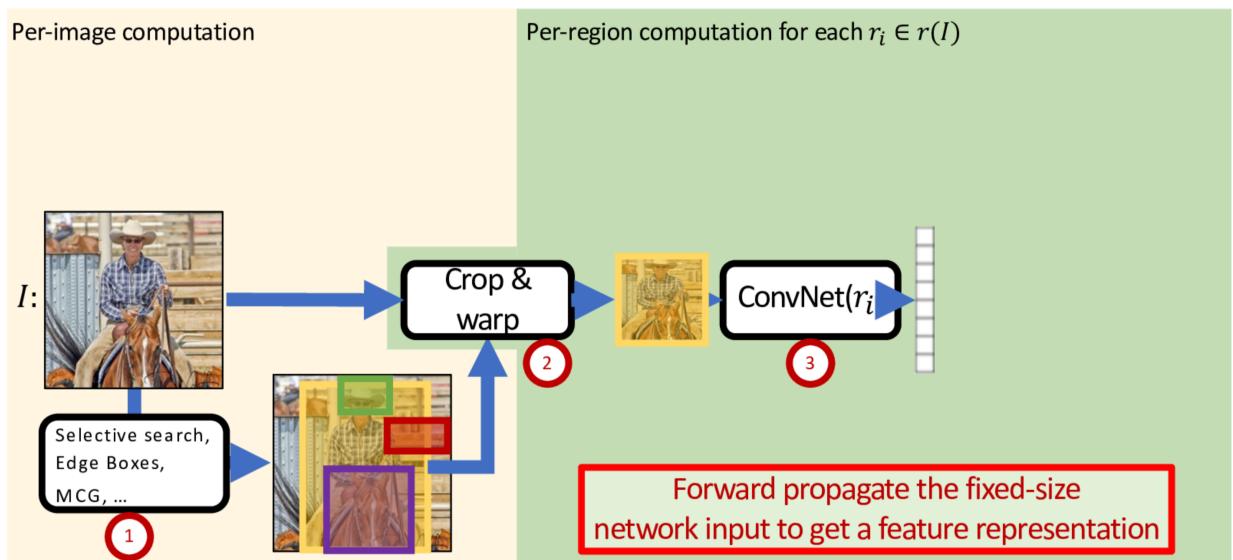
- R-CNN: Region-based Convolutional Neural Network [Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.](#)  
- 第 1 步



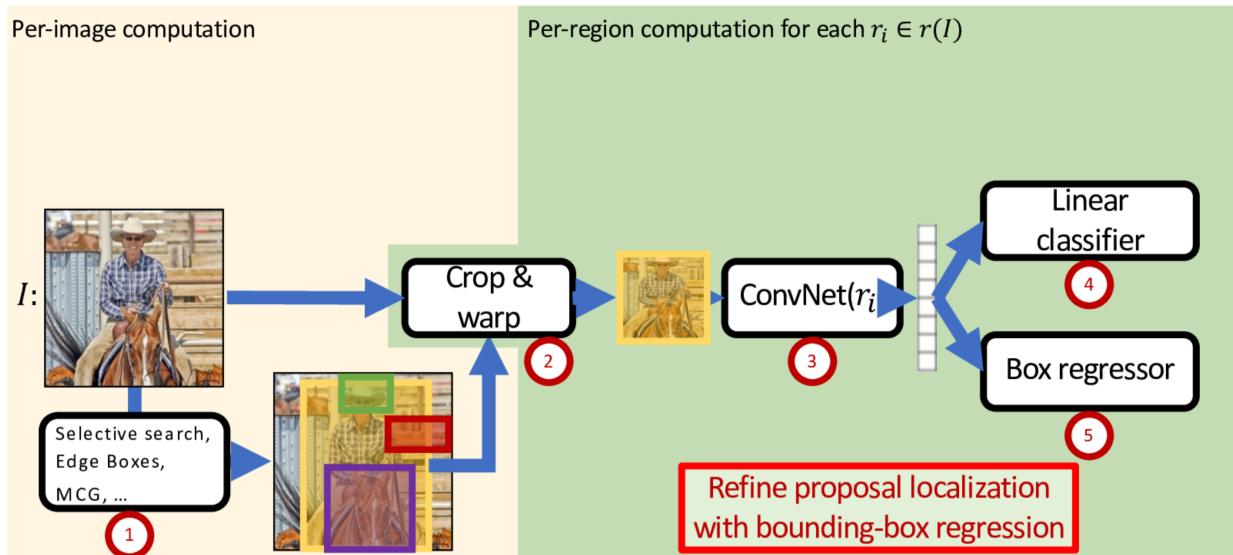
- 这是一个很简单的想法，首先使用一个现成的 region/object/detection proposal (生成候选区域/对象/检测) 算法，例如 Selective Search、Edge Boxes、MCG 等算法
- 每张图片大概会生成 2000 个左右的 candidates 结果，这已经是传统方法中非常好的一个结果了，2000 已经是一个相对来说较小的输出空间了
- 第2步



- 我们将每个 candidate 结果裁剪并变形到固定的大小，从而使模型的输入能够标准化
- 第3步



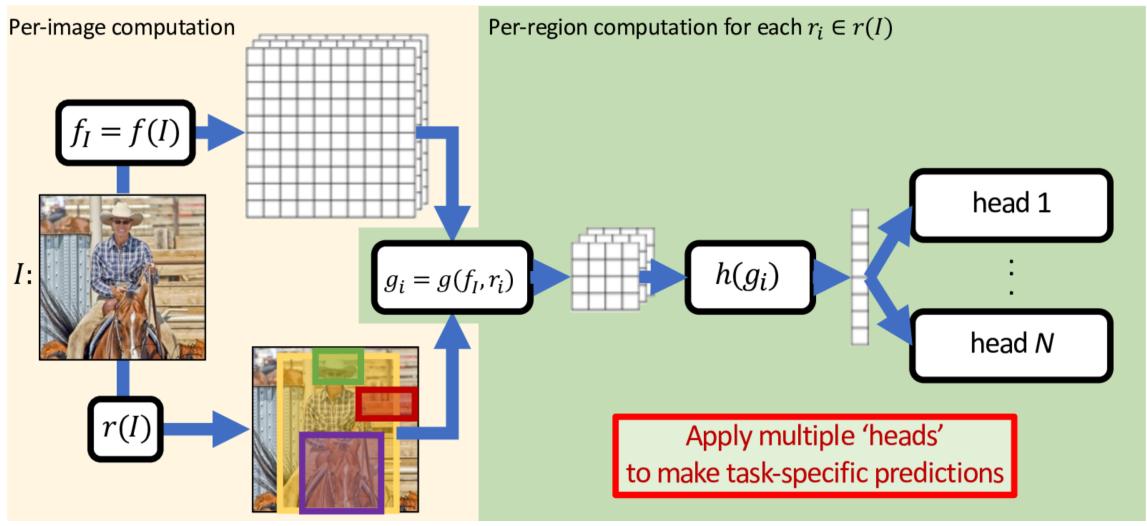
- 然后我们将这些裁剪好的图片作为一个卷积网络的输入
- 至此，我们将对象检测问题转化为了一个图像分类问题
- 第4步



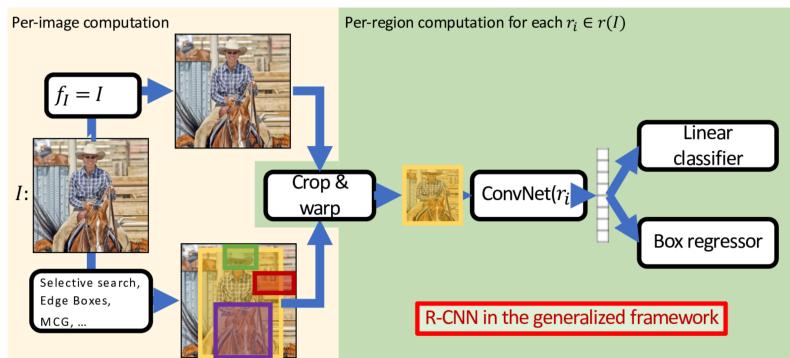
- 最后，该模型会输出一个线性分类器
- 与此同时，由于开始的 proposals 算法并不是特别的精确，因此模型还会对边框做一轮回归检测，这个过程称为 Box regression

我们可以将这个过程泛化为一个一般的方法

- Generalized Framework



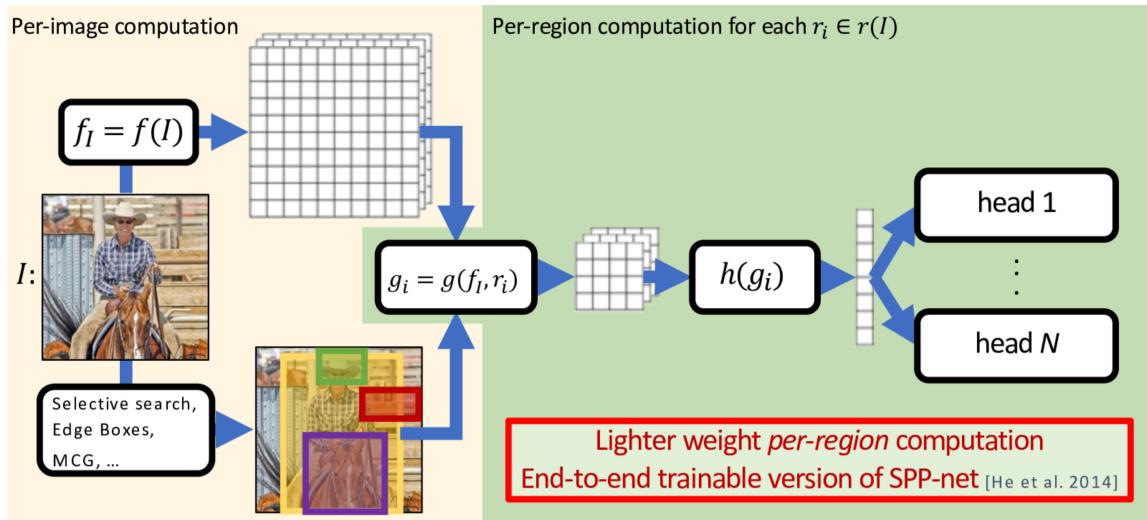
- ■ 黄色区域表示对每张图片的计算
  - 在这里，我们对图片计算其特征，以及 bounding-box proposals，然后利用特征表示对应的框选区域
- 绿色区域表示对每个区域的计算
  - 我们将黄色区域的输出输入到一个神经网络  $h(g_i)$  中，并且根据需求，添加若干 head，使其可以输出所需要的数据
- 我们所说的 R-CNN 代入这个框架的情况也就如下图所示



- R-CNN 有哪些问题呢？
  - 对于每一张图，有非常大的计算量（比如 2000 次全卷积网络的评估）
  - 没有计算结果或特征可以共享（也即没有做任何优化）
  - 传统的 region proposal 方法使得整个运行时间更长了
  - 传统的 region proposal 技术并不是很精确，召回率很低

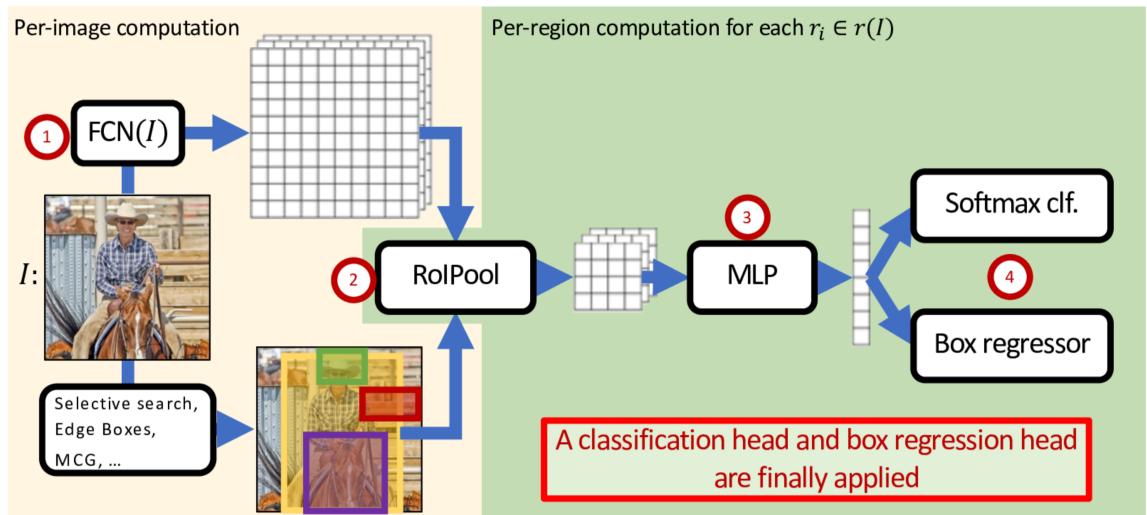
在 2015 年，Girshick 等人又提出了这个网络的下一个版本——Fast R-CNN

- Fast R-CNN [Girshick: Fast R-CNN. ICCV, 2015](#)

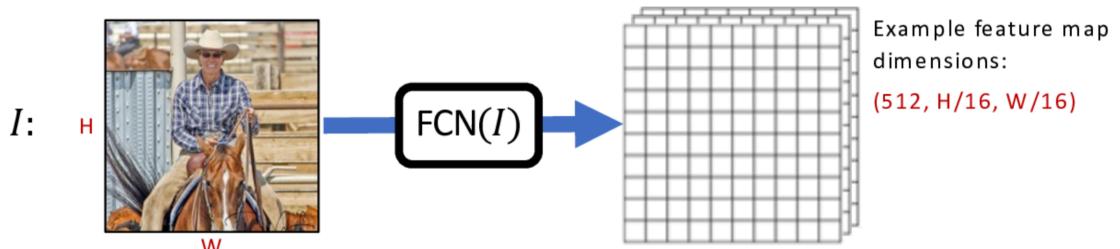


- ■ 主要思想

- 用一个更加轻量级的模型替换掉原来的 FCN，将复杂的计算转移到黄色区域的部分，从而加快运行速度

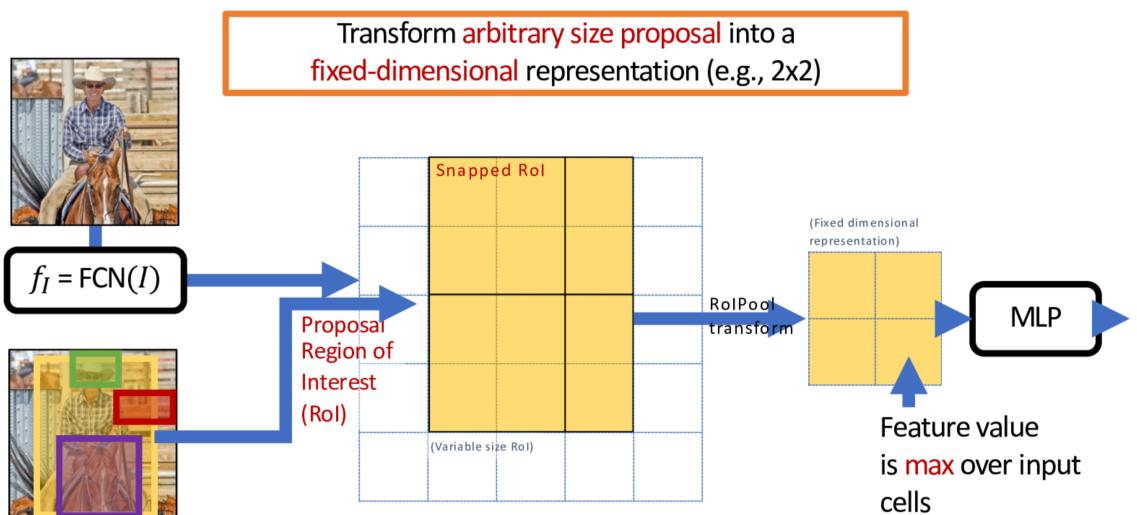


- ■ 具体来说，我们用一个 FCN 来计算每张图片的特征，得到一个较低分辨率的特征图（图中白色区域所示），然后利用一个 RoI (Region of Interest) 池来结合 bounding-box 和对应特征，并将它们转化为固定大小，准备输入绿色区域
- 绿色区域中，我们接收黄色区域的结果，输入一个轻量级的 MLP，然后输出 box-regressor 和 classifier 的结果
- 然后我们分别具体说说每个部分的构成
- 首先是 FCN

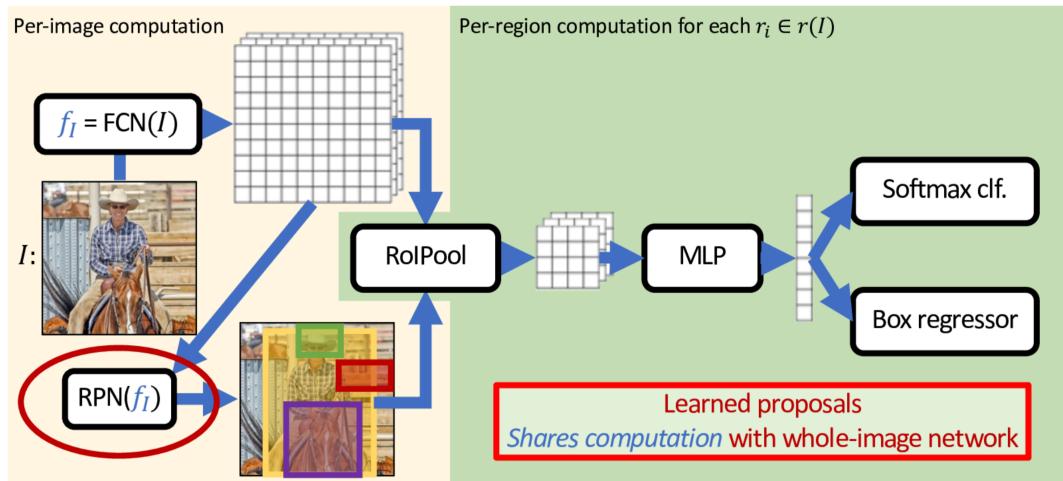


- 我们可以使用任何标准的卷积网络作为 backbone 结构

- 例如 AlexNet、VGG、ResNet、ResNeXt、DenseNet 等
- 不过我们需要移除全局池化的部分，也即输出的维度与输入的维度直接成正比
- 根据实验结果来说，特征非常重要，而强大的 backbone 能够有更好的结果
- 然后说说 RoI Pool

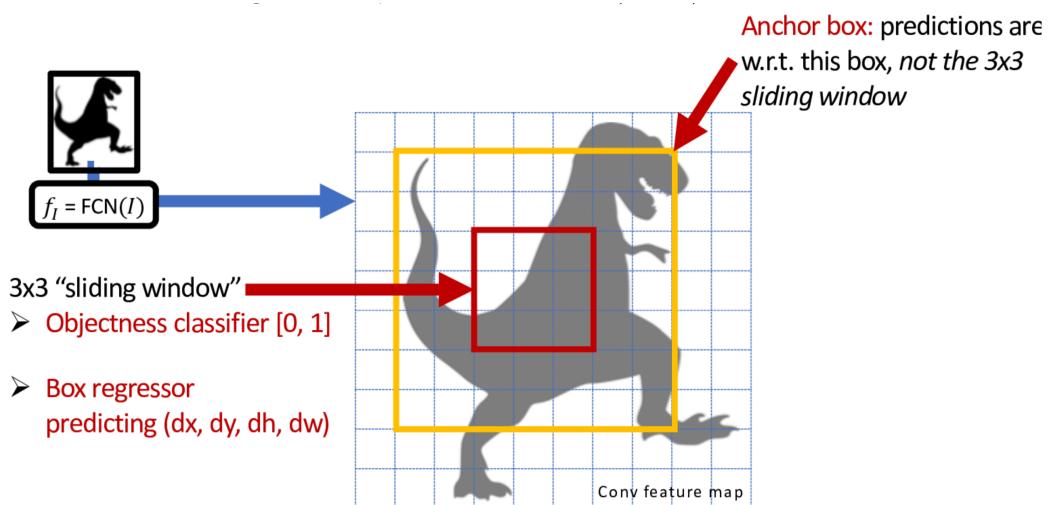


- 首先，根据图像计算 candidates (这里和 proposals 是一个意思，都代表被选上的候选区域) 和低分辨率特征图
- 将每个 candidate 对齐到对应的特征图上，形成一个 RoI 区域
- 然后通过最大池化操作，将 RoI 区域转换到一个固定的维度大小，使其可以输入 MLP
- 由此，RoI Pool 可以处理任意尺寸的 candidate 输入
- Fast R-CNN 有哪些问题呢？
  - 对于每一张图，有非常大的计算量 (比如 2000 次全卷积网络的评估)
  - 没有计算结果或特征可以共享 (也即没有做任何优化)
  - 传统的 region proposal 方法使得整个运行时间更长了
  - 传统的 region proposal 技术并不是很精确，召回率很低
- 现在，我们还需要解决传统的 Region Proposal 问题
  - Faster R-CNN [Ren, He, Girshick and Sun et al.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015](#)

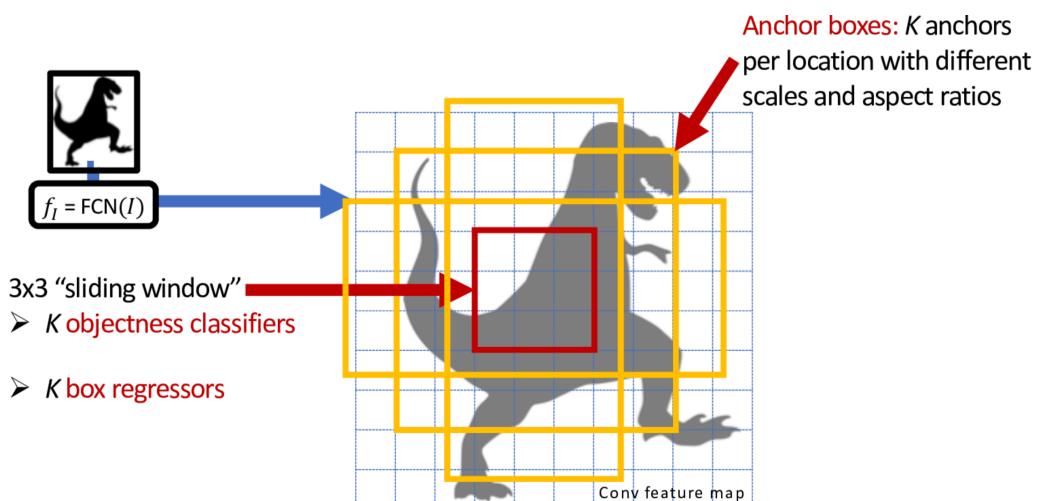


- - 我们不使用传统方法来生成 proposals，取而代之的是一个 end-to-end 的神经网络 RPN，并且我们不用将图像复制 2 份，而是将 FCN 的特征图结果直接共享作为 RPN 的输入

- Region Proposal Network (RPN)



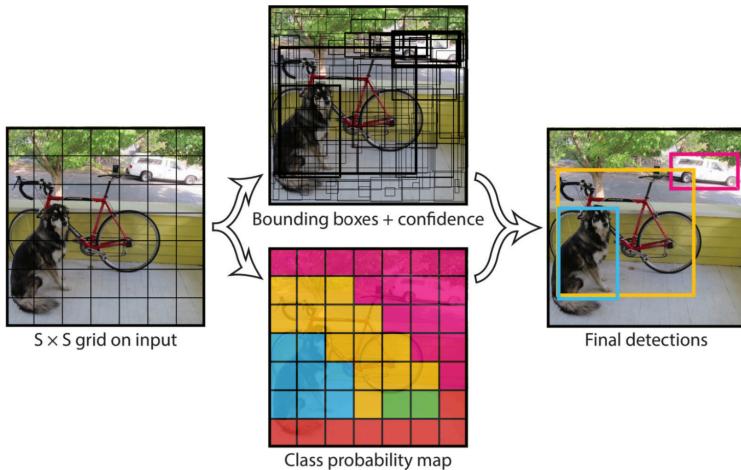
- - 在特征图上，我们滑动一个窗口，在每个窗口中尝试判断区域内是否有物体
  - 然后在四个方向的 delta 范围内，尝试做 box regression，确定精确的位置



- 由于窗口只能变换长宽，而不能任意改变形状，所以只使用1个锚框往往是不准确的
- 在实践中，我们实际会使用K个不同长宽的锚框，通过多个锚框覆盖的范围，作为最后输出的结果

在这里粗略地拓展一下，关于另一种该领域其它工作的策略

- Single Stage Detection

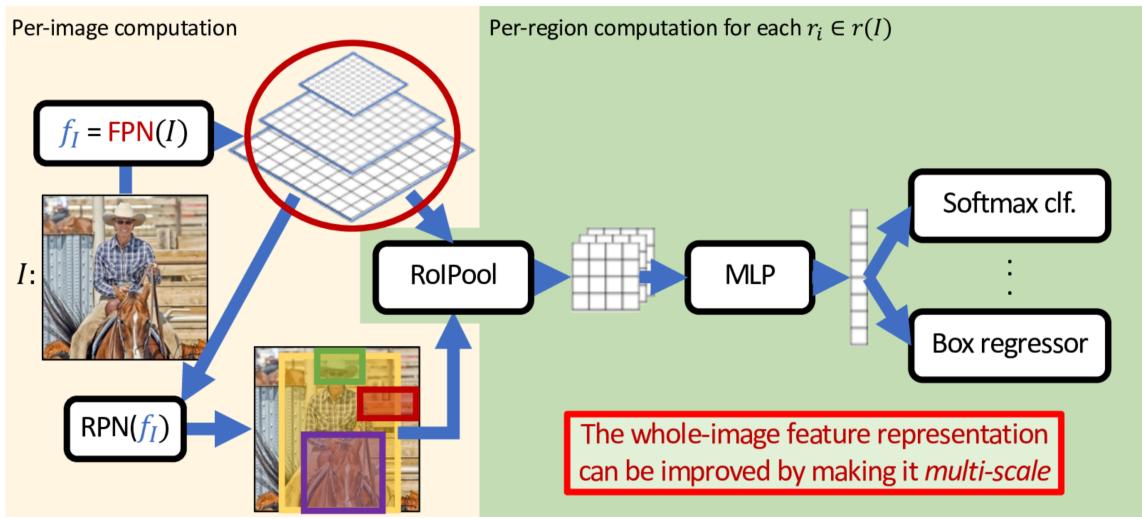


- YOLO 和 SSD [Joseph, Divvala, Girshick and Farhadi: You Only Look Once: Unified, Real-Time Object Detection. CVPR, 2016](#)

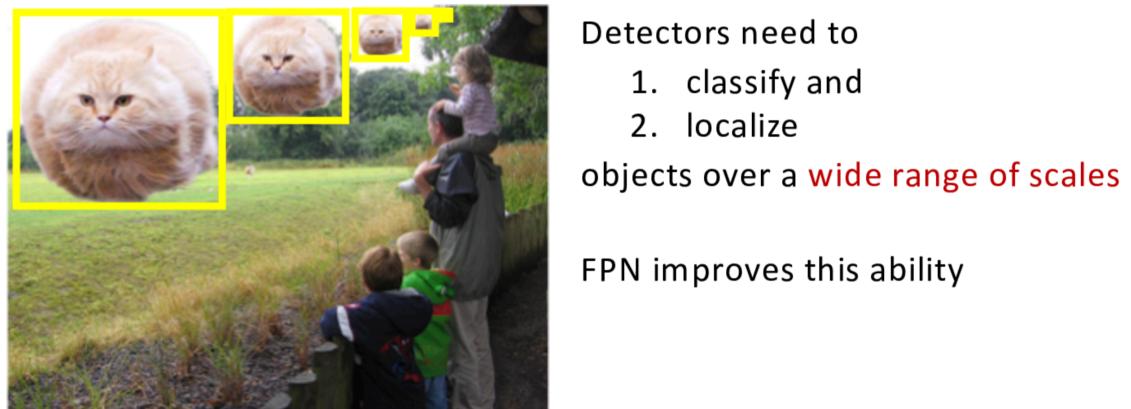
- 它们不单独使用一个网络来提取 proposal，而是将这些过程全部融合在一个阶段中
- 然而，只使用一个阶段来完成这些工作是一个非常困难的任务——这些工作有更快的速度，但是准确度往往不如两阶段的 State-of-the-Art 的工作

然后继续回到两阶段的工作的介绍

- Feature Pyramid Network [Lin, Dollár, Girshick, He, Hariharan and Belongie: Feature Pyramid Networks for Object Detection. CVPR, 2017](#)



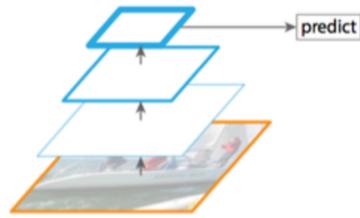
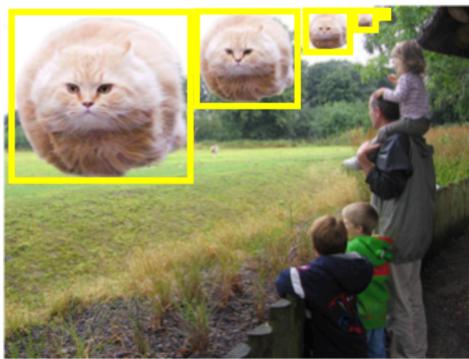
- 另一项该领域的创新，考虑到将多尺度特征检测引入到 FPN 中，从而使得模型可以捕捉图像中的多尺度特征对象
- 因此，我们需要一种特征表示可以表示多尺度特征



- 我们希望可以检测到任意尺寸、任意位置的目标
- 策略 1：Image Pyramid



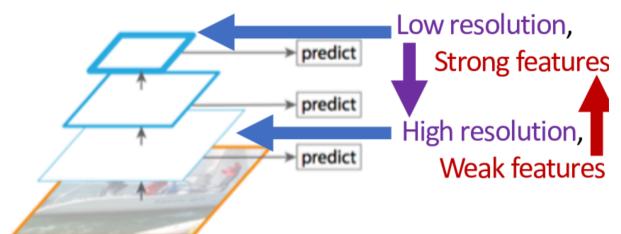
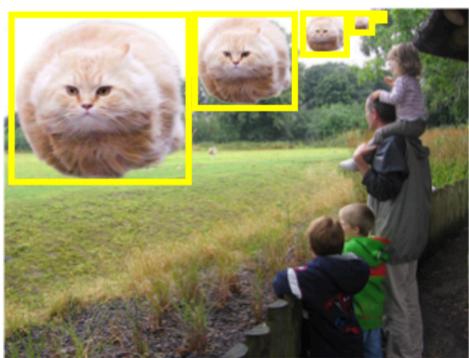
- 我们将图像缩放到多个尺寸，每个尺寸单独进行一次 predict
- 这是一种最标准的解决方案，但是代价是运行极其的慢
- 策略 2：Multi-scale Features (Single-scale Map)



(b) Single feature map

*Leave it all to the features – fast, suboptimal  
(E.g., Fast/er R-CNN, YOLO, ...)*

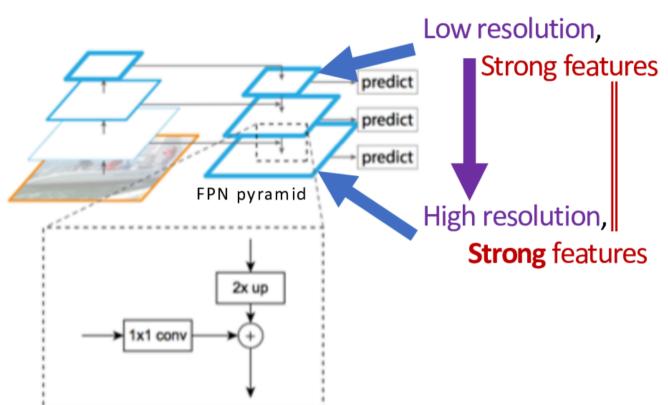
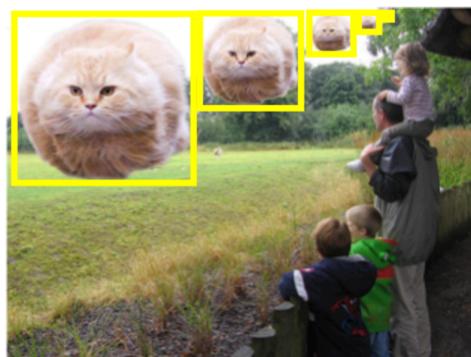
- 我们将图像不断提取特征，只在最小尺寸的特征上进行一次 predict
- 这种方案的运行速度最快，但是效果也是次优的，因为它实际只能捕捉到一种尺度特征
- 策略 3：Naive In-network Pyramid



(c) Pyramidal feature hierarchy

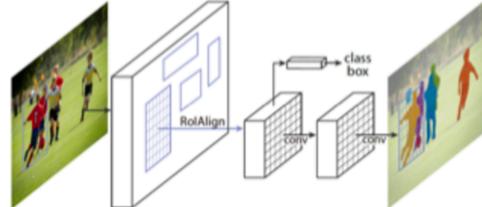
*Use the internal pyramid – fast, suboptimal  
(E.g., ≈ SSD, ...)*

- 这种策略中，我们单纯的使用内部结构的金字塔，在图像卷积过程生成的多个尺度的特征图上分别进行 predict
- 这种方案运行的速度仍然很快，但是效果仍然是次优的，因为在高分辨率的情况下，特征十分模糊，能够捕捉到的特征非常稀少，对于小物体的预测结果会非常糟糕
- 策略 4：Feature Pyramid Network

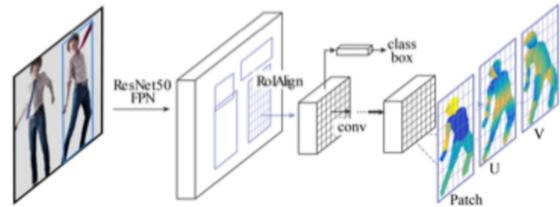


- 这种策略类似于 U-Net 结构，我们将图像卷积缩放到多个尺度特征，并且用一个反向金字塔来逐步扩大，对应尺度之间用跳跃层补充信息
- 从而在低分辨率和高分辨率下都能够有很强大的特征

- Feature Pyramid Network 在实际使用中有很多好的表现结果
- 对于这个框架体系，我们可以很容易地添加 heads 从而实现别的功能



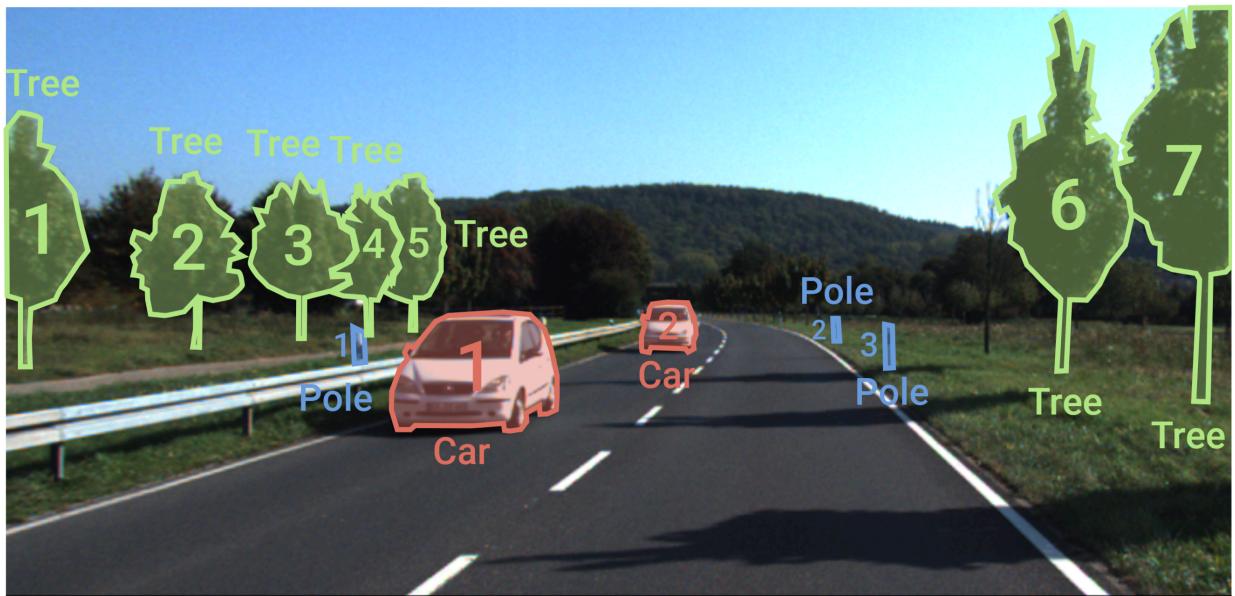
**Mask R-CNN**  
[He, Gkioxari, Dollár, Girshick]



**DensePose**  
[Güler, Neverova, Kokkinos]

- Mask R-CNN：对每个检测，预测实例分割的遮罩
- DensePose：预测对象的坐标（纹理贴图坐标）

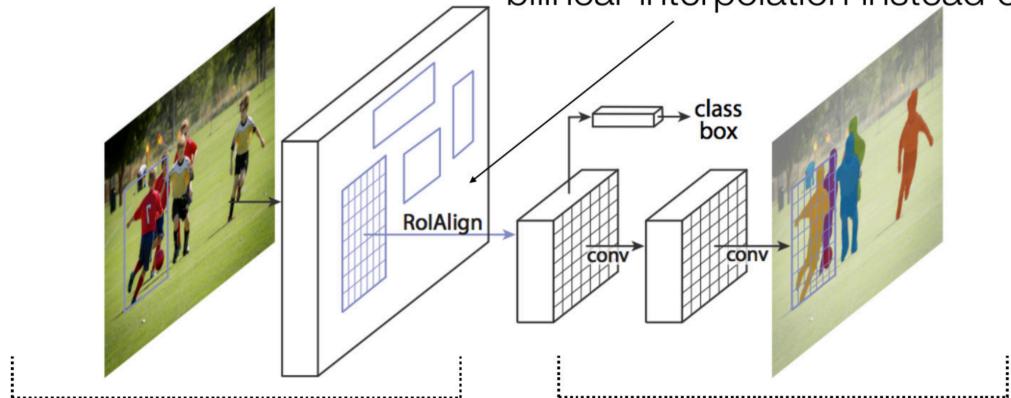
- Instance Segmentation



**Goal:** Assign a semantic and an instance label to every pixel of an object

- 下面我们再来看看一下实例分割任务，与目标检测不同，它希望能够给出对象具体的形状边界，而不是一个大致的锚框
- Mask R-CNN: Faster R-CNN for Instance Segmentation [He, Gkioxari, Dollár and Girshick: Mask R-CNN. ICCV, 2017.](#)

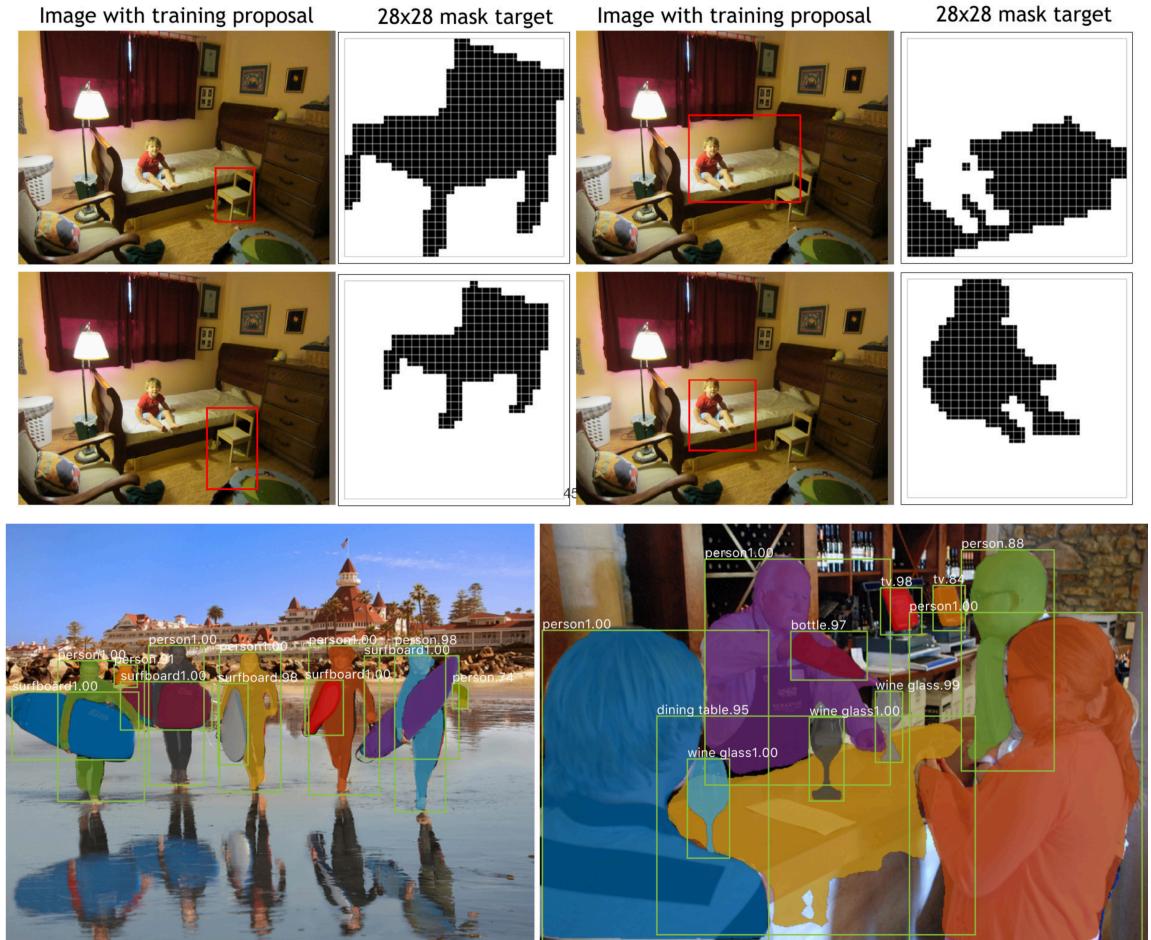
ROI pooling with tiny change:  
bilinear interpolation instead of max



Faster R-CNN

Extra “head” on network  
predicts binary mask

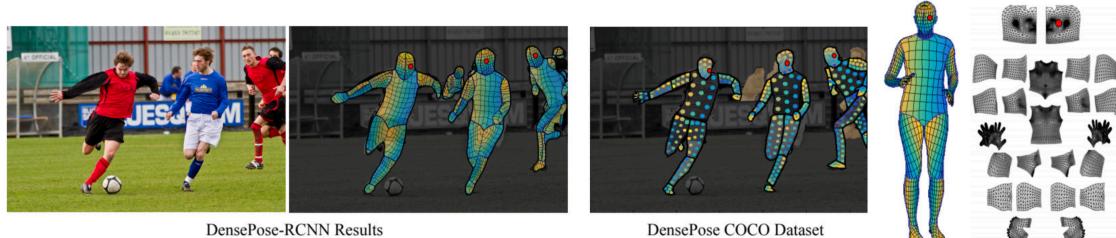
- ■ 它在 Faster R-CNN 的基础上进行修改，将原来的 RoI Pool 修改为一个称为 RoI Align 的部件
- 我们仍然是使用 RoI 结合对应特征，但是不再使用最大池化来缩放，而是使用双线性插值使其可以恢复到原来分辨率，并进一步细化物体的边缘
- 从而最后预测出一个二进制的实例分割遮罩（前景 vs 背景）
- 预测结果



- ■ 上图为遮罩级别的 IOU 精确度评估结果，可以看到结果出乎意料得好，很震惊居然预测的遮罩居然能和对应的实例有如此高

## 的契合度

- DensePose: Faster R-CNN for Dense Human Pose Estimation [Guler, Neverova and Kokkinos: DensePose: Dense Human Pose Estimation in the Wild. CVPR, 2018](#)



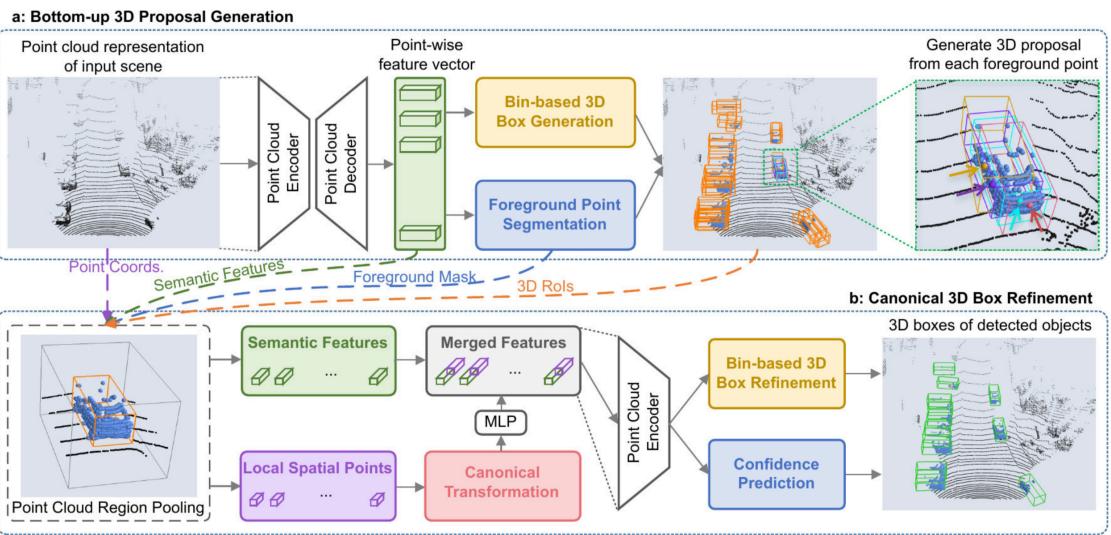
- ■ 将 RGB 中的所有人类像素映射到 3D 人体表面  
■ 在每个代表人类的区域内密集地回归部分特定地 UV 坐标  
■ 主要贡献还包括一个数据集——DensePose-COCO，一个手动标注图像与对应 3D 表面关系的大型数据集

- Mesh R-CNN: Faster R-CNN for Meshes [Gkioxari, Malik and Johnson: Mesh R-CNN. ICCV, 2019](#)

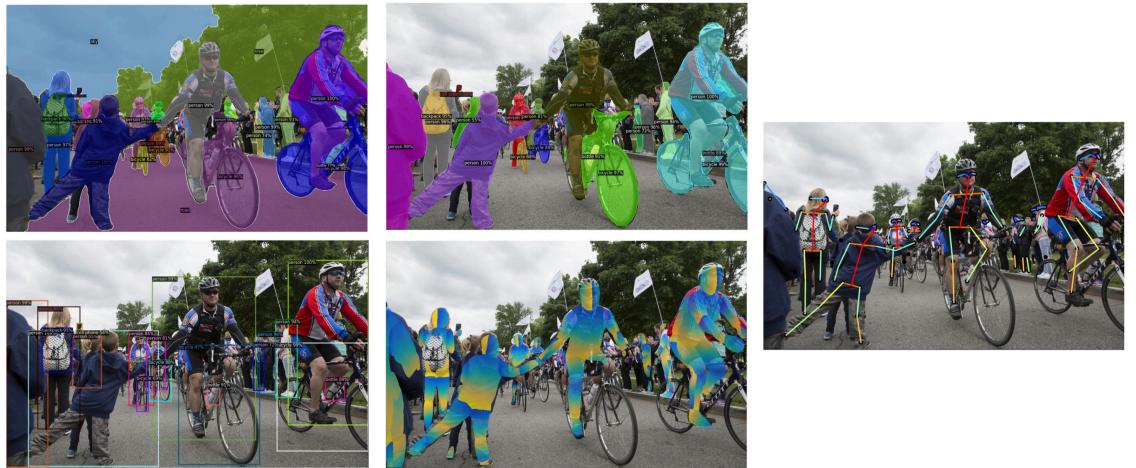


- ■ 从 2D 数据中预测 3D 网格，在 Faster R-CNN 的基础拓展而来  
■ 这一类工作也可以通过在 Faster R-CNN 上添加 head 并做部分修改实现

- PointRCNN: Faster R-CNN for Point Clouds [Shi, Wang and Li: PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. CVPR, 2019](#)



- ■ 输入为点云，输出为 3D 的识别框
- 该工作使用和 Faster R-CNN 的类似结构，进行 3D 层面的目标识别
- Detectron2 for PyTorch <https://github.com/facebookresearch/detectron2>



- ■ 最后介绍一项 Facebook 发布的工具箱，给出许多预先定义好的类似的权重，基于 Pytorch 实现，便利于我们自己来实现很多不同输出目标的模型