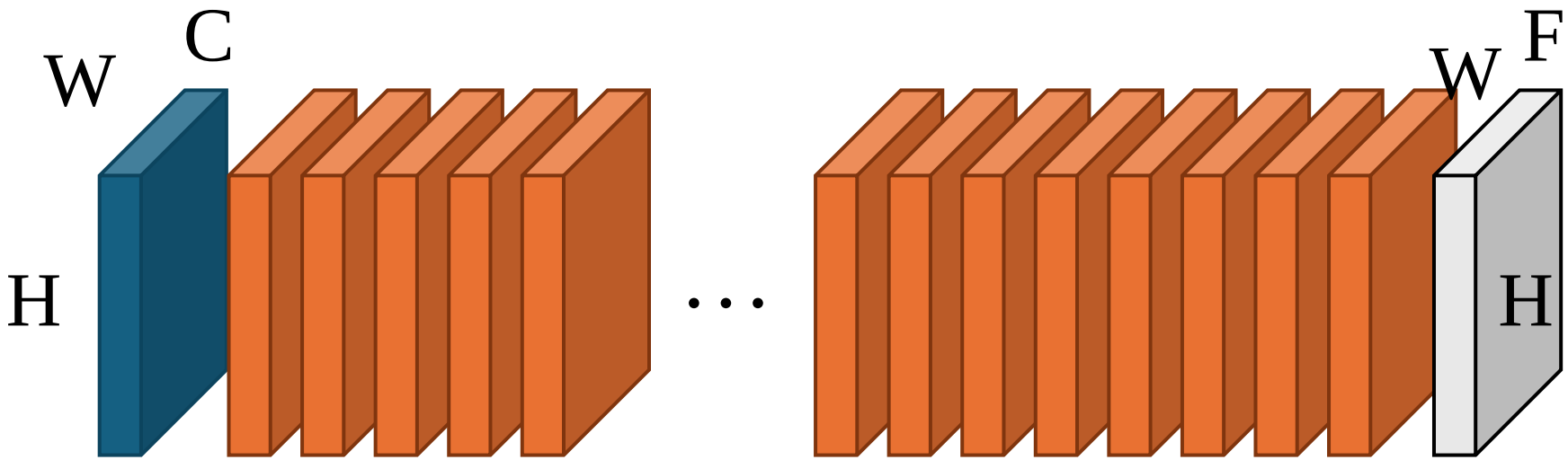
回顾——语义分割

怎么做？堆叠卷积？



N 个 3x3 卷积的感受野是 2n+1 大小的方形区域

如果想让感受野>=200 需要多少卷积?

100

滑动窗口：开销依然过大

把每个窗口的局部拿出来过CNN，
预测窗口中心的像素标签

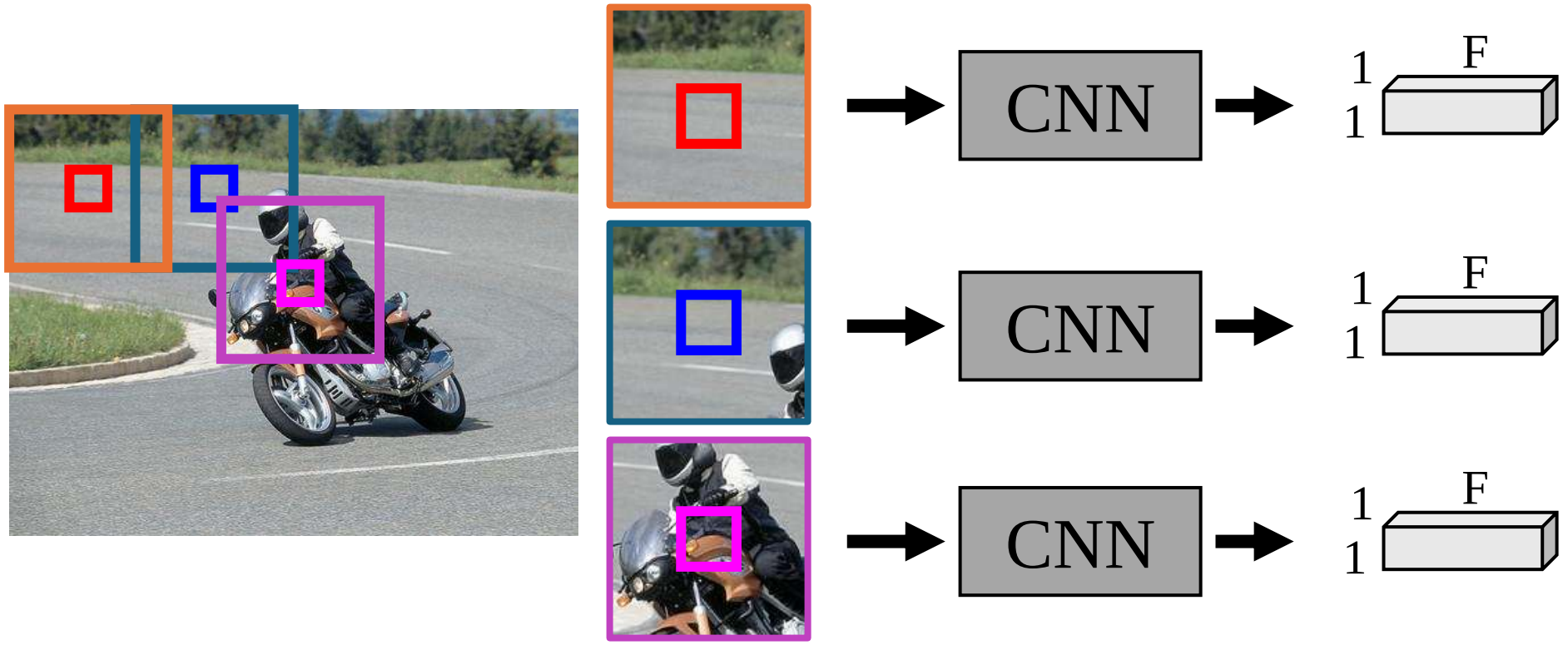
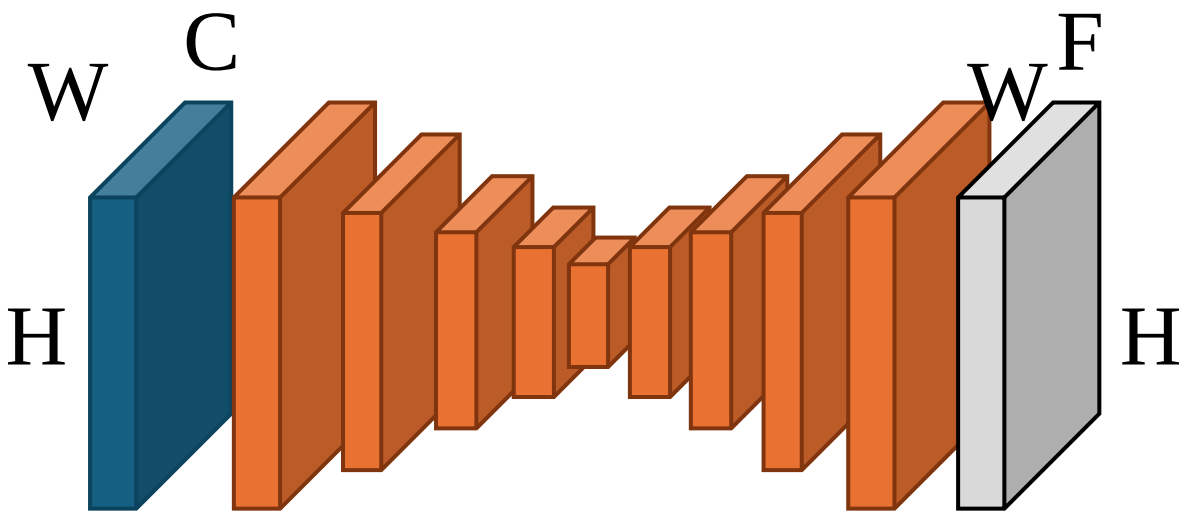


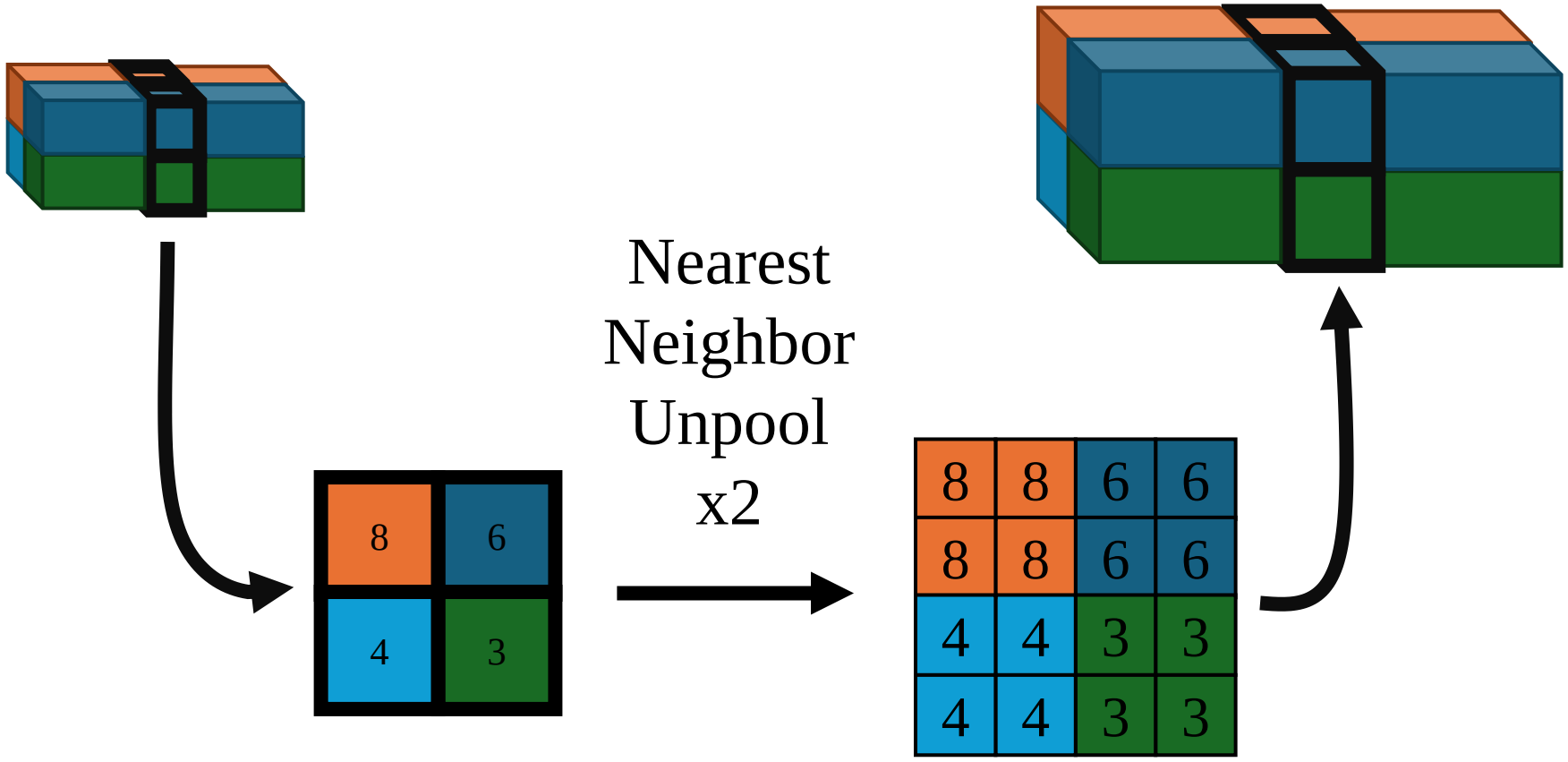
Image credit: PASCAL VOC, Everingham et al.

编码器-解码器

核心思想: 先 **downsample** 下采样
再 **upsample** 上采样.
怎么做下采样?
卷积, 池化

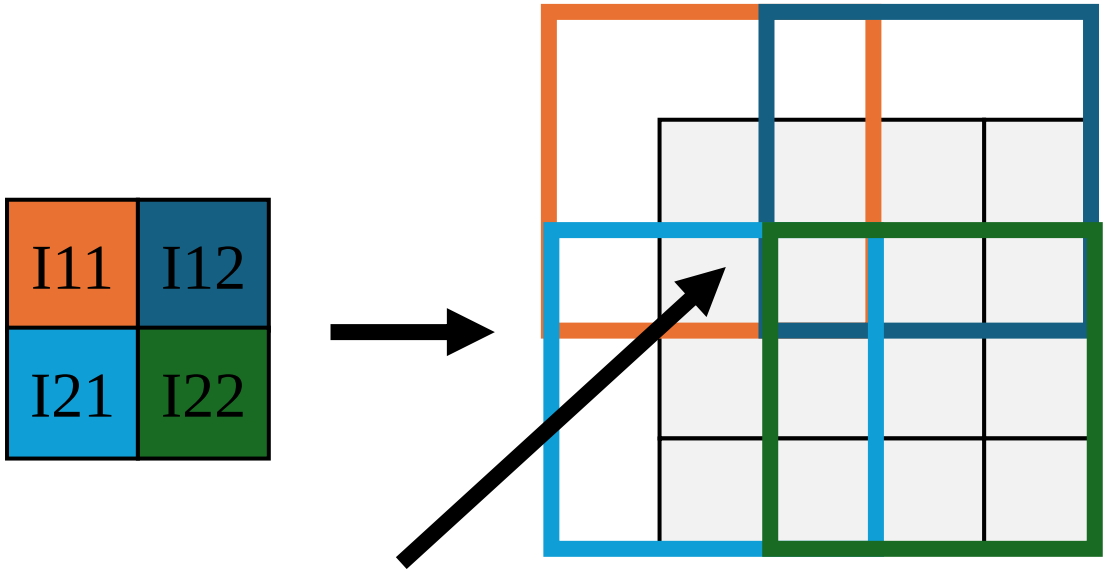


Unpooling——反池化



Transpose Convolution 、DeConvolution——反卷积

3x3 Transpose Convolution, Stride 2, Pad 1



输出为 滤波 F 乘上输入 I

相交出加和(e.g., $I_{11}F$ $I_{21}F$)

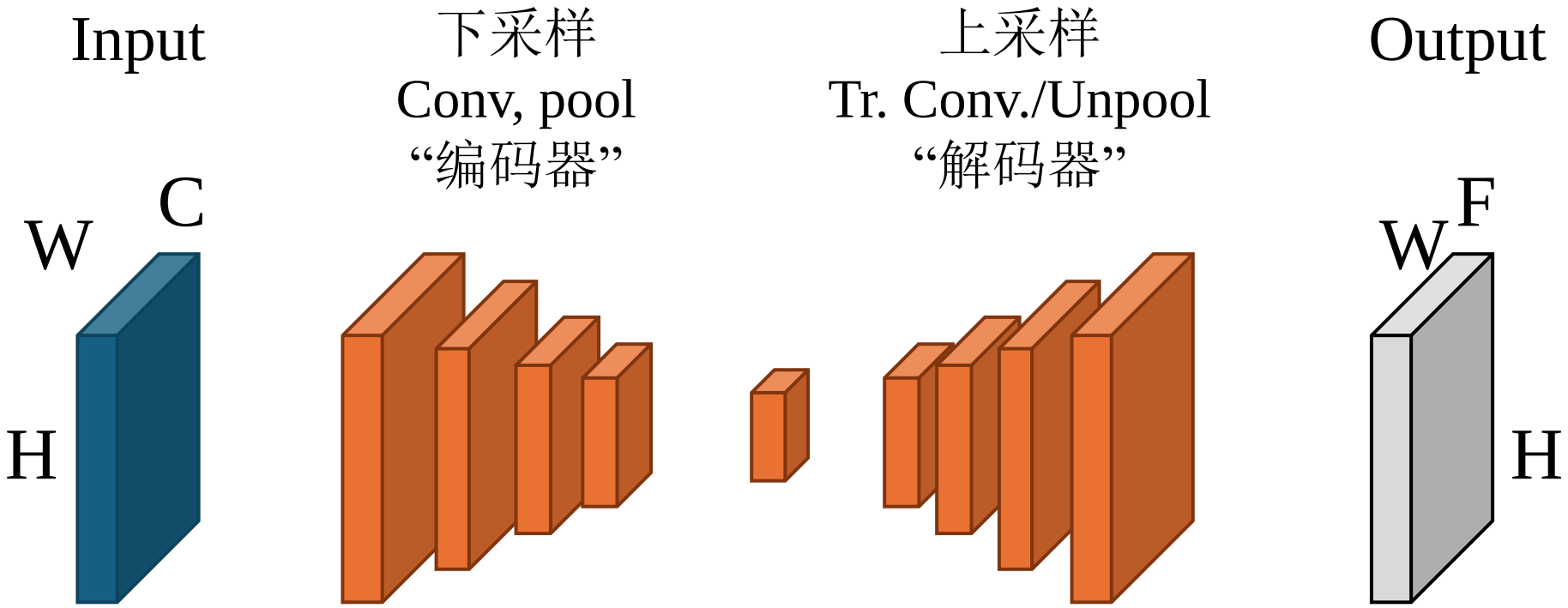
$I_{11}F$ $I_{12}F$

$I_{21}F$ $I_{22}F$

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung

汇总起来

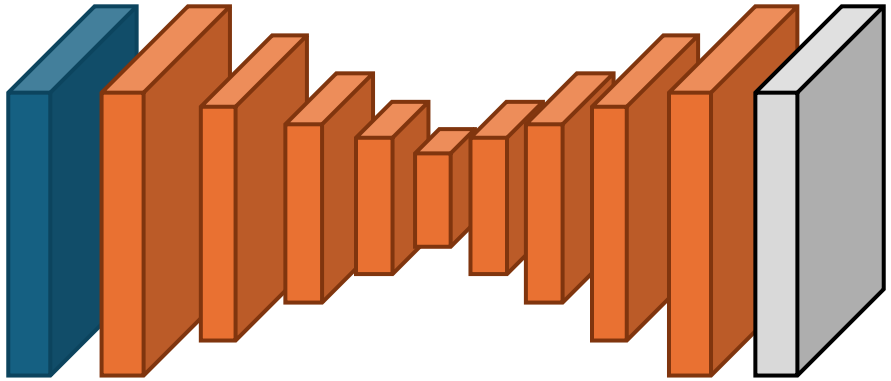
卷积 + 池化 来做 下采样/压缩/编码
反卷积/反池化 来做 上采样/解压缩/解码

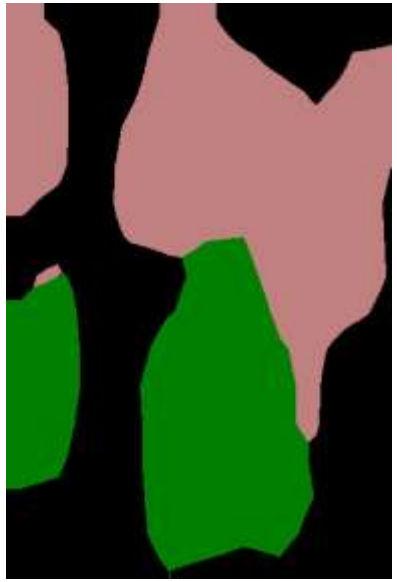


缺失的细节内容

如果输出尺寸时 HxW, 直接上采样一个经过下采样的图像，
我们经常会得到一个缺失细节的结果.
为什么?

下采样时信息会丢失!

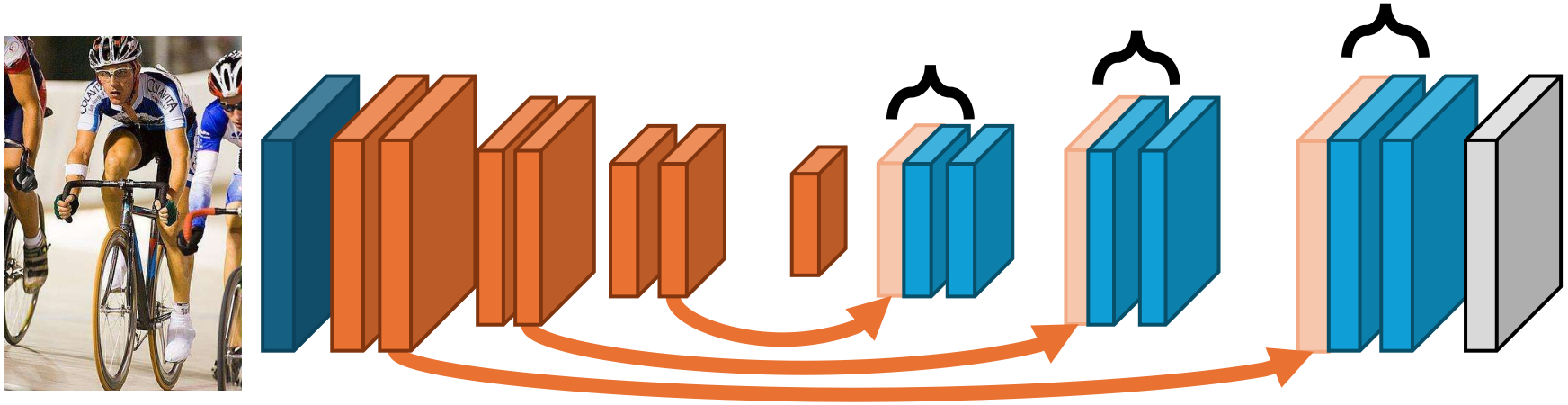




Result from Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014

UNet

怎么把缺失的细节补回来？
复制这些层的输出.
把他们拷贝到相同分辨率的、缺失细节的地方去.



Copy

Result from Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014

实例分割与物体检测

分类

CAT

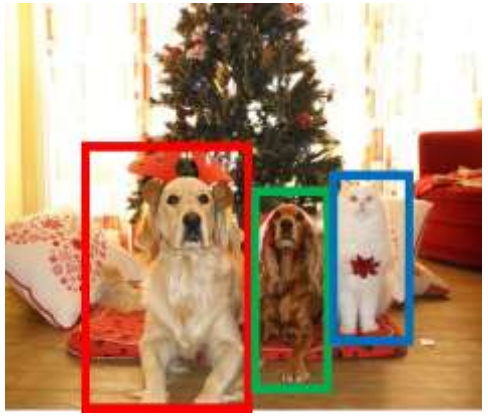
无空间限定

语义分割

GRASS, **CAT**,
TREE, **SKY**

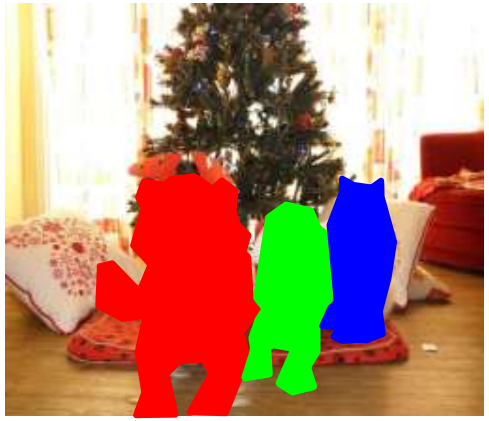
像素级预测

物体检测



DOG, **DOG**, **CAT**

实例分割



DOG, **DOG**, **CAT**

多物体

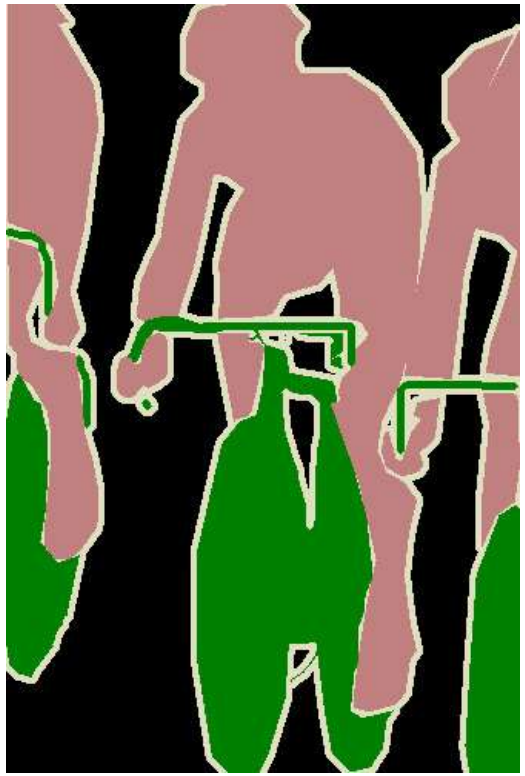
回顾

“语义分割”: 标注每个像素点其对应的类别.

Input

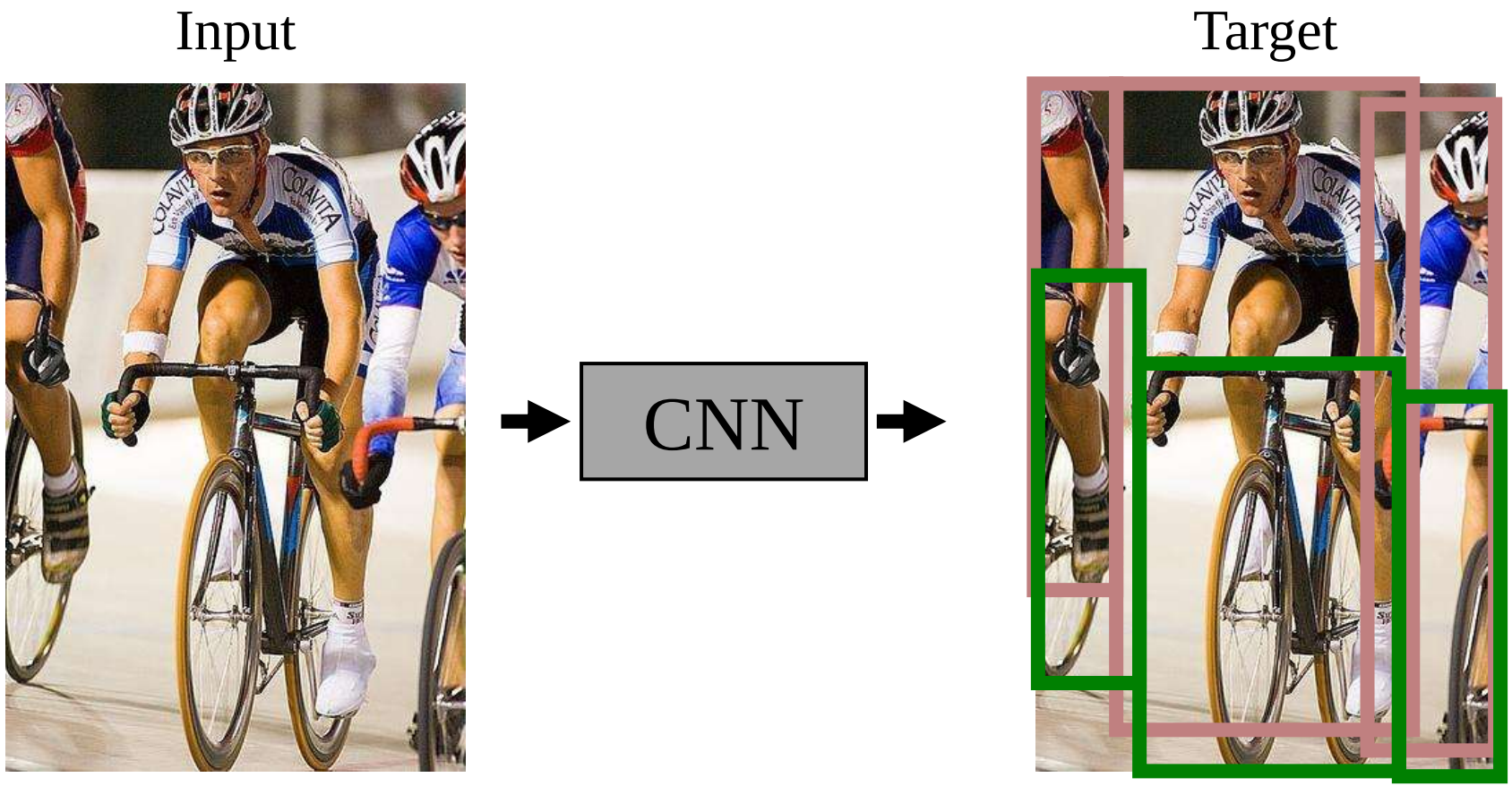
CNN

Target

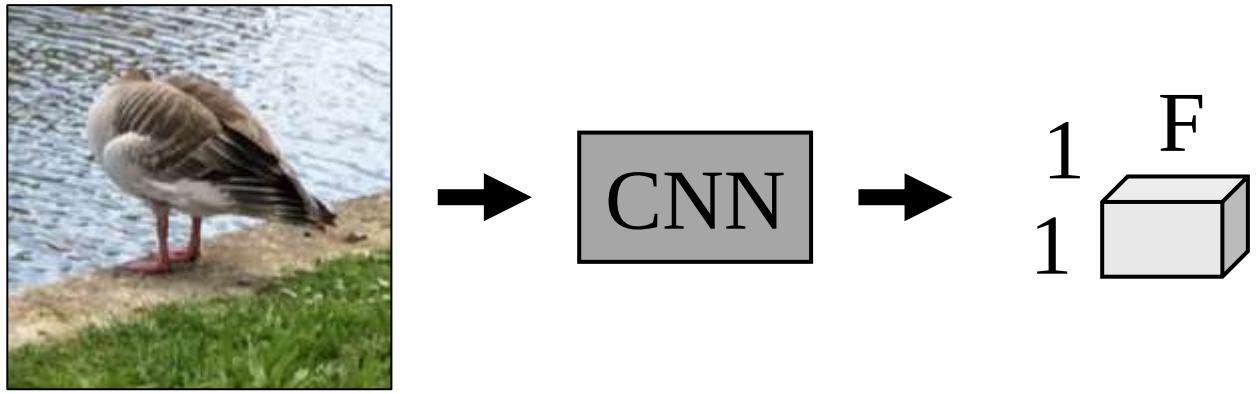


Object Detection：物体检测

“Object Detection”: 对某个类别的所有实体画一个边界框



如何错误地做物体检测

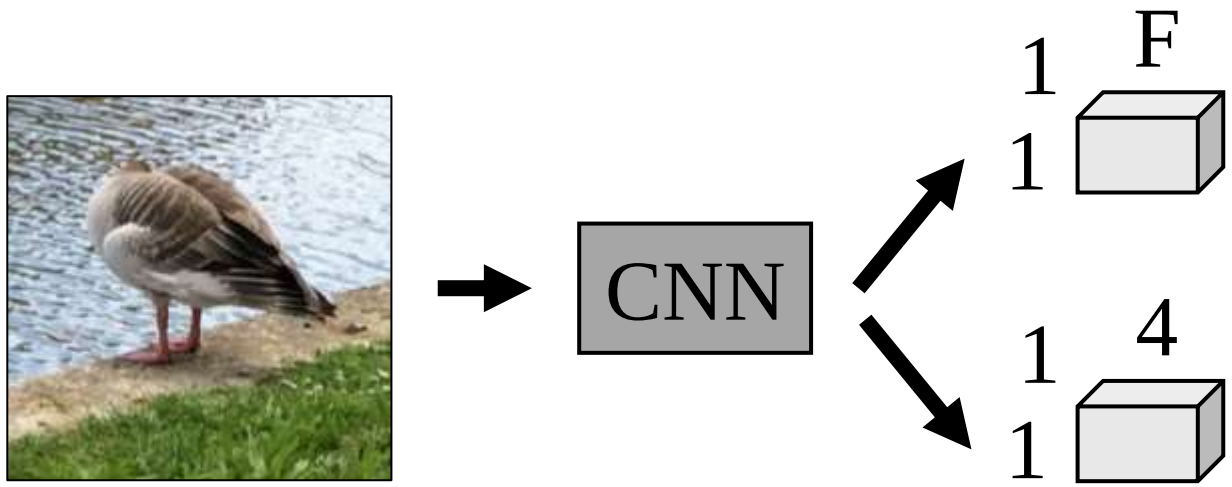


让我们从图像分类开始：

可以从F个类别中进行预测

P(cat), P(goose), … P(tractor)

如何错误地做物体检测

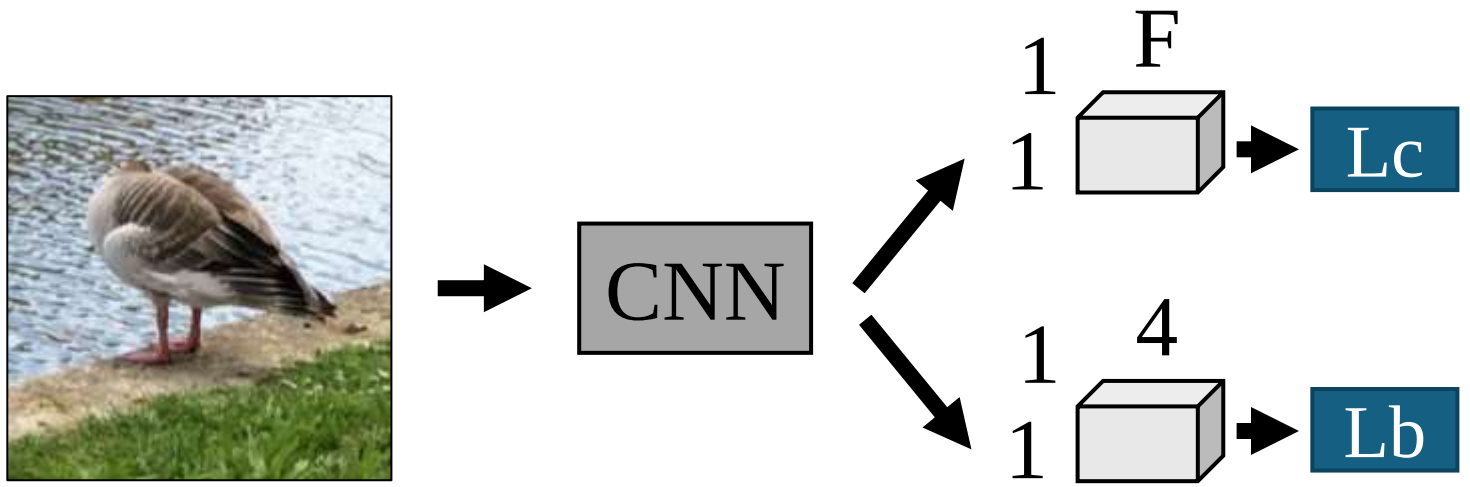


加一个额外的输出：

预测物体所在的位置

[x,y,width,height] or [minX,minY,maxX,maxY]

如何错误地做物体检测



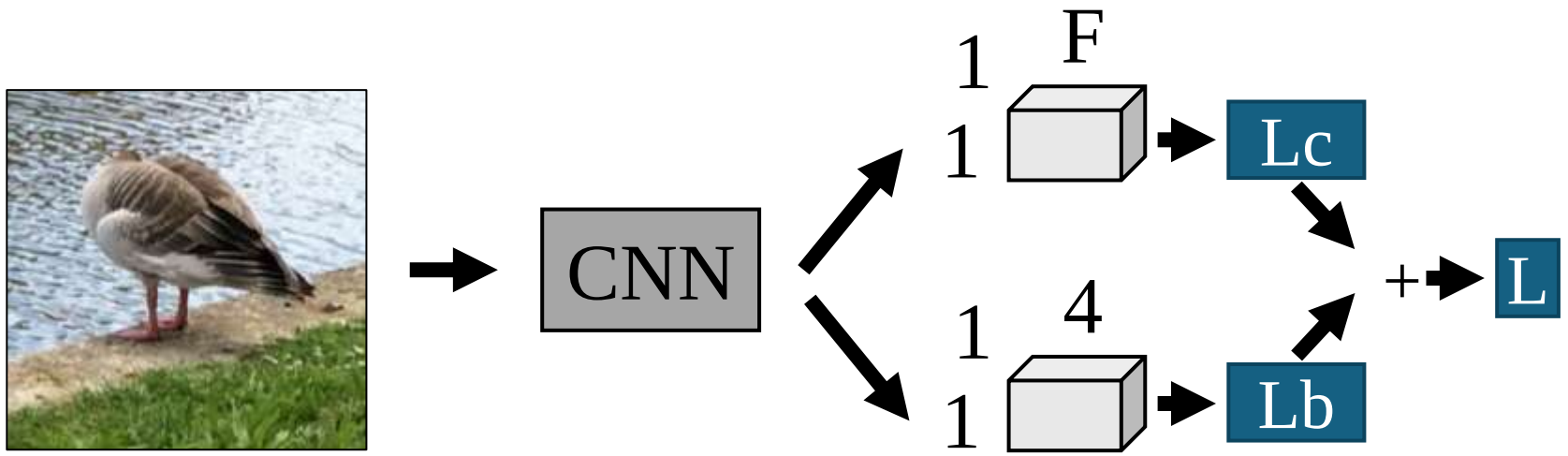
加上损失函数:

惩罚分类错误/定位错误

Lc = negative log-likelihood

Lb = L2 loss

如何错误地做物体检测

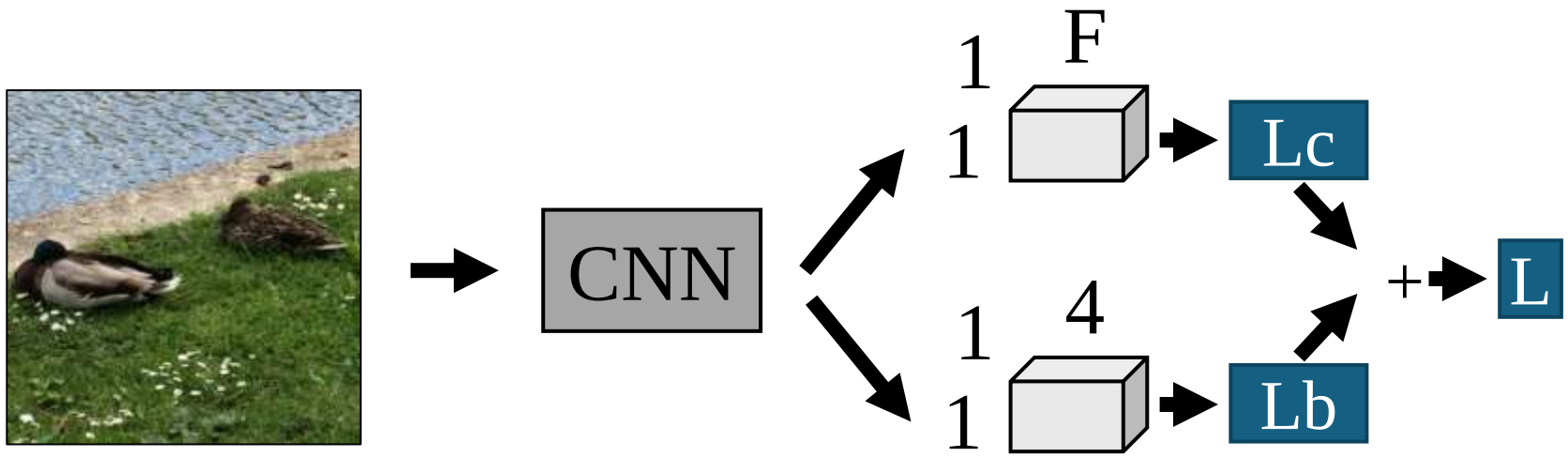


计算误差，梯度反传

最终的误差: L = Lc + λLb

为什么会有λ?

如何错误地做物体检测

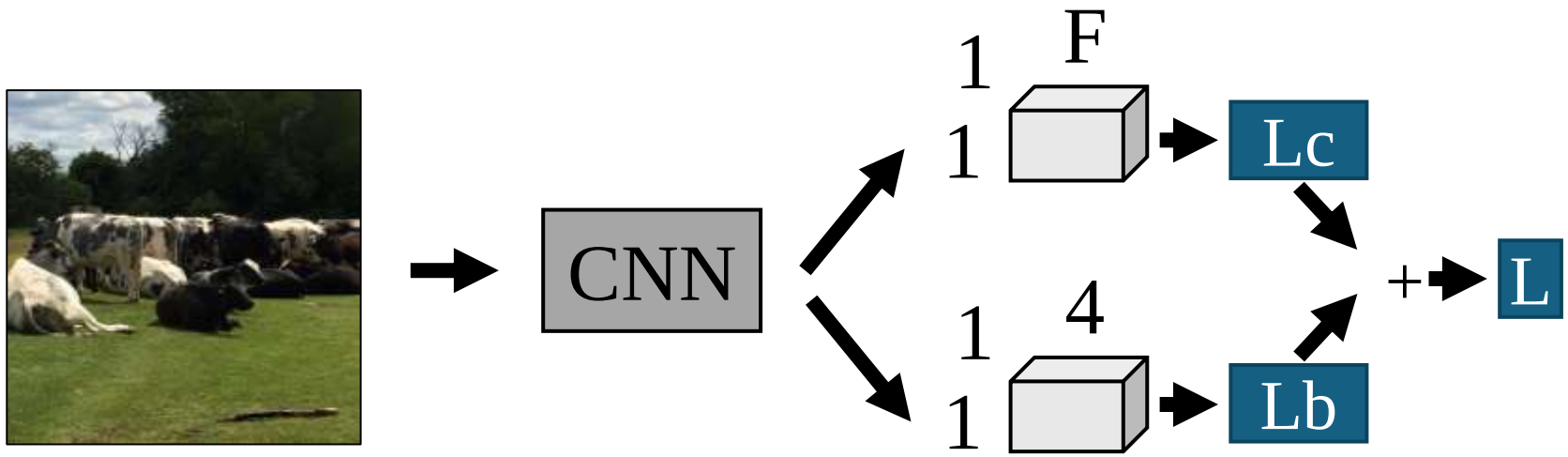


现在游过两只鸭了.

我们需要多少输出?

F, 4, F, 4 = 2*(F+4)

如何错误地做物体检测



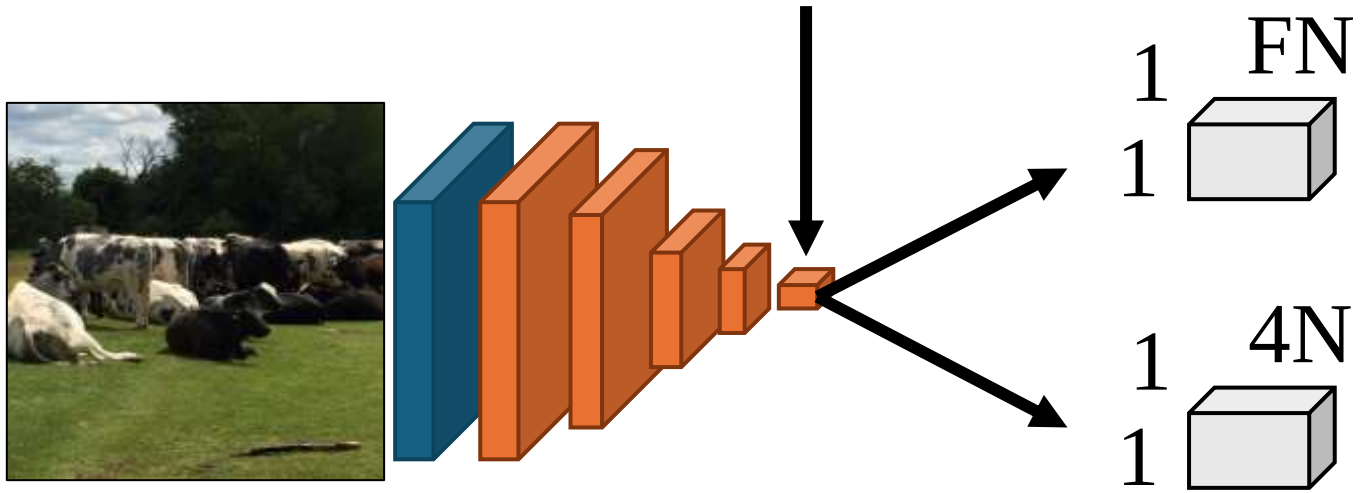
现在牛群来了.
我们需要更多的输出
(甚至根本不知道到底需要多少输出).

总结问题——物体检测

- 我们不知道输出有多少个 因为它经常是在变化的
  - 就算我们可以，我们怎么让网络来预测呢.

关键在于怎么让网络知道这N个物体

在哪



换个思路

检查所有窗口，看看这个窗口是否“紧紧包含”某一个物体

Yes

No?

记住这个窗口！

No

滑动窗口检测

假设我们在固定尺寸的窗口寻找行人

Slide credit: J. Hays

滑动窗口检测

在每个窗口上寻找...

Slide credit: J. Hays

滑动窗口检测

在每个**尺度**每个窗口上寻找…

Note – 固定尺寸的窗口

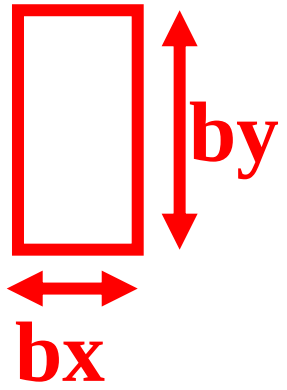
Slide credit: J. Hays

所有的窗口

Slide credit: J. Hays

一共有多少窗口?

给定大小 HxW 的图像和 一个 “template 模板” 大小为 by, bx.



Q. 有多少框大小为(by,bx)?

A. (H-by)*(W-bx)

这是在考虑以下问题以前的答案:

- *scales* (by*s,bx*s)
- *aspect ratios* (by*sy,bx*sx)

物体检测的挑战

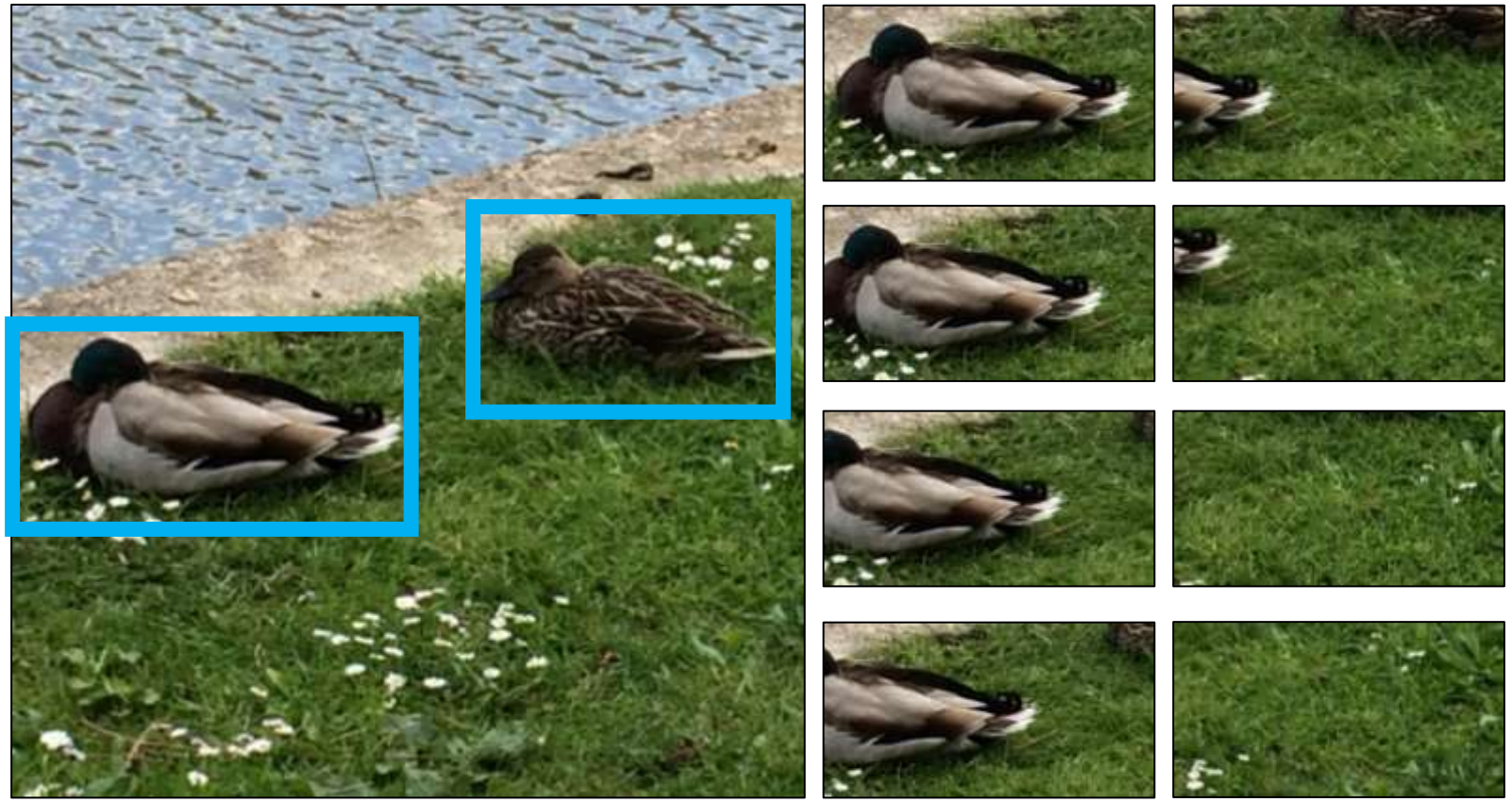
- 成千上万的框
- 却没几个有用的，包括物体的框极少

怎么整一个错误的框?

1. Wrong left x
2. Wrong right x
3. Wrong top y
4. Wrong bottom y

评估——边界框

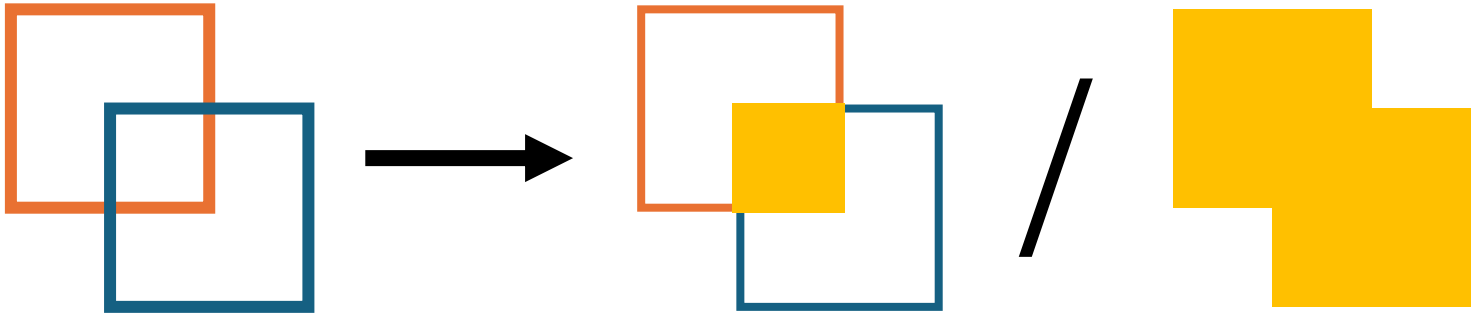
告诉我什么时候你觉得检测不再正确了.

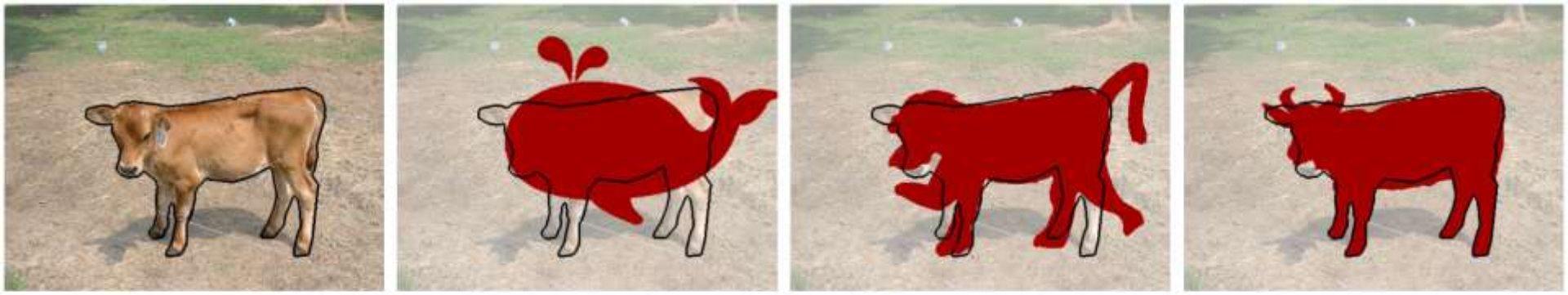


评估——边界框

标准的评价方法:

Intersection over union/IoU/Jaccard coefficient





(a) Ground truth (b) $\mathcal{J} = 0.554$ (c) $\mathcal{J} = 0.703$ (d) $\mathcal{J} = 0.910$

Jaccard example credit: P. Kraehenbuehl et al. ECCV 2014

知识问答

你比五年级小学生聪明吗?

成年人跟五年级小学生比小学知识竞赛.

成年人不见得能赢，因为你学的知识没有用！

CV 知识问答

你的模型比随机数生成器厉害吗?

训练后的模型与随机模型相比.

如果你的评价指标不够真实，其实并不一定哪个模型谁更好lol.

你真的比随机数生成器更智能?

- **求概率：1000类分类正确?**
  - 1/1,000
- **求概率：猜边界框的四个角，且误差小于10% 图像大小?**
  - (1/10)*(1/10)*(1/10)*(1/10)=1/10,000
- 两者概率叠加: 1/10,000,000
- 不能用最常出现的标签（没有物体）来直接预测

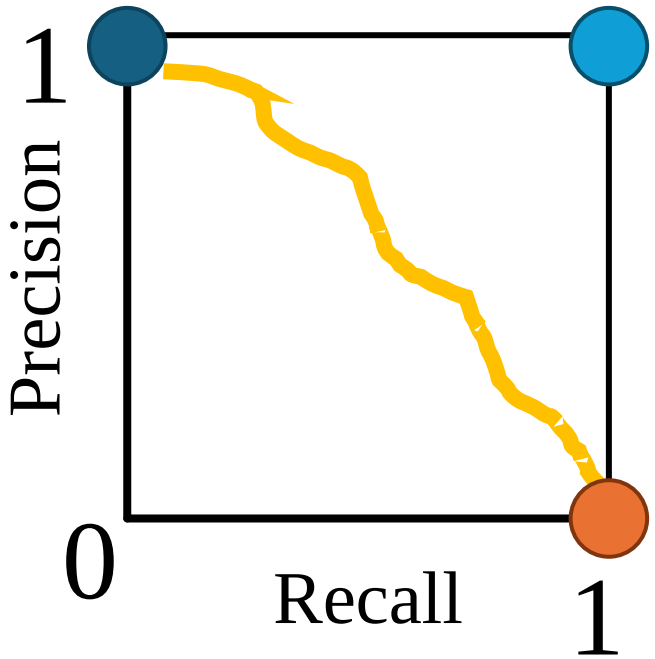
性能评估

- 准确率 = 预测正确的概率
- 如果我声称我的行人检测99%正确.
- **这代表我的检测做的很好吗?**
- 大街上没有多少人!

性能评估

- True detection 真实检测 （true positive）: 高 IoU
- Precision 精确度: #真实检测/ #检测总数
- Recall召回率: #真实检测/ #真实的物体数量

如果拒绝一切，那么就不会有错误

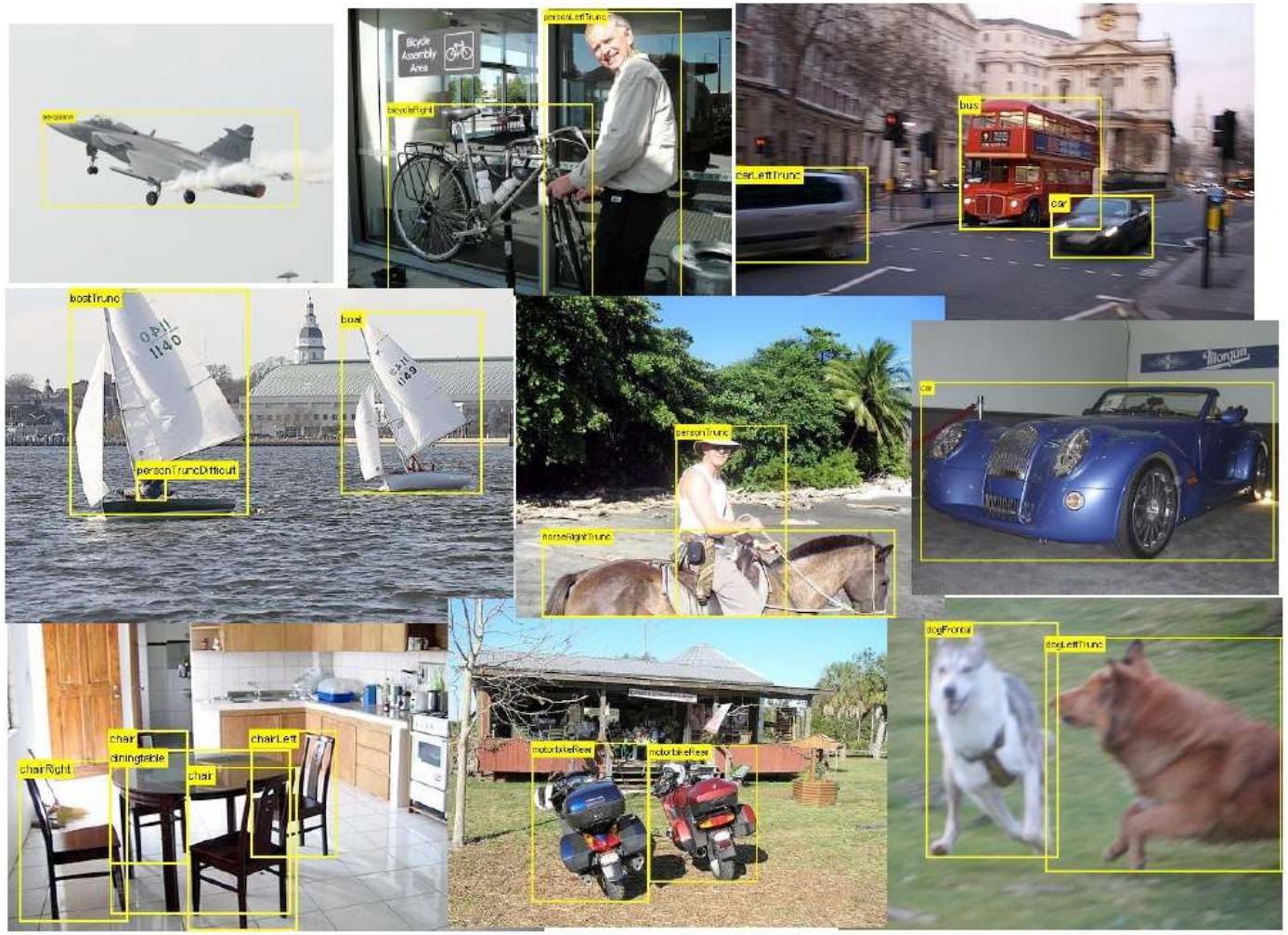


理想情况是精确度和召回率都很高!

AUC area under curve (avg. precision)

如果接受一切，那么就不会错过任何东西

各式物体检测



Slide Credit: S. Lazebnik

传统方法：Histograms of oriented gradients (HOG)

将图像分割成多个块，并在每个块中计算梯度方向的直方图

HxWx3 Image

H’xW’xC’ Image

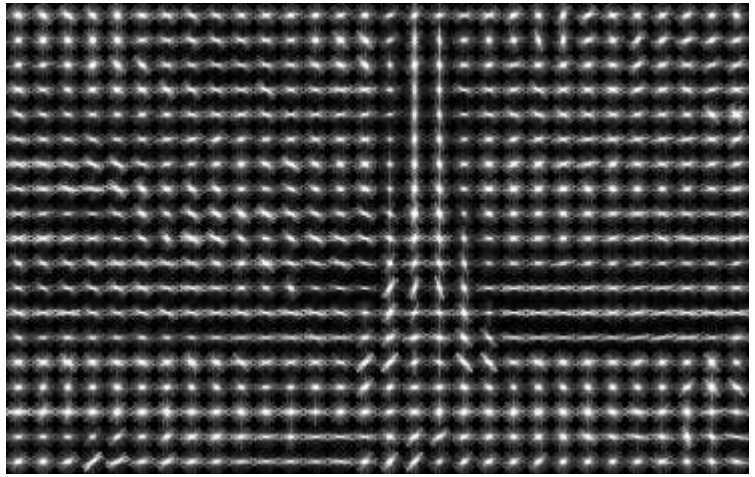


Image credit: N. Snavely

N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Slide Credit: S. Lazebnik

使用HOG进行行人检测

- 使用线性支持向量机（SVM）来训练一个行人模板

正训练样本：包含行人

负训练样本：没有行人

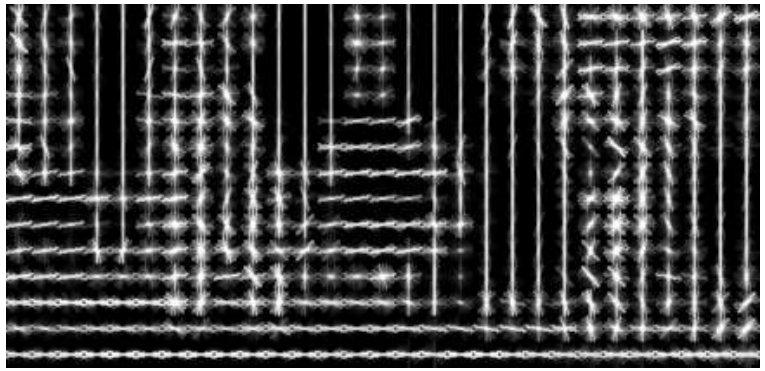
N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Slide Credit: S. Lazebnik

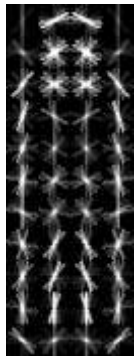
使用HOG进行行人检测

- 用线性SVM训练行人“模板”
- 测试时对HOG特征图用模板做卷积
- 找到局部最大响应值
- 不要忘了应对多尺度： HOG *pyramid 金字塔*

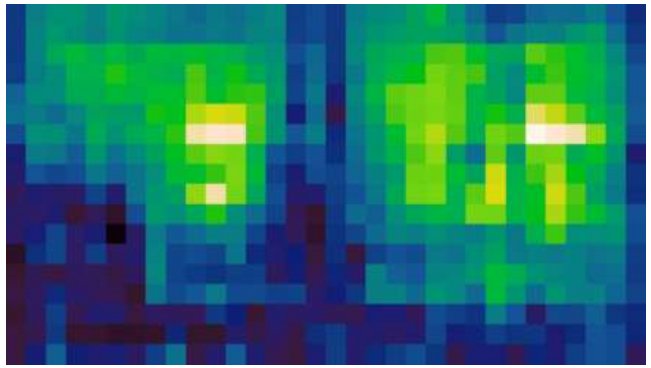
HOG 特征图



模板



检测器响应图



N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Slide Credit: S. Lazebnik

检测效果

[Dalal and Triggs, CVPR 2005]

Slide Credit: S. Lazebnik

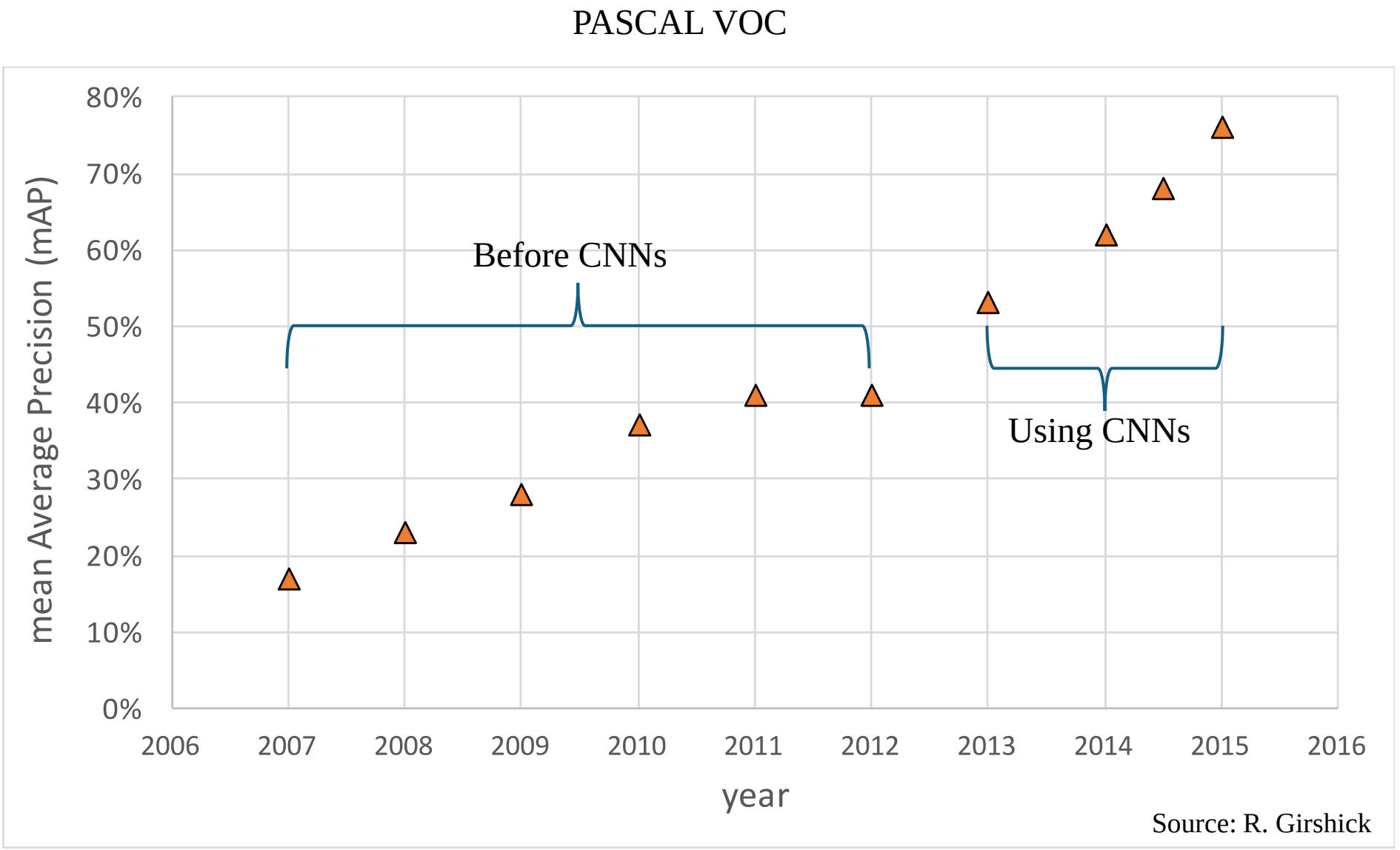
PASCAL VOC 数据集 (2005-2012)

- 20 个类别:
- *Person*
- *Animals:* bird, cat, cow, dog, horse, sheep
- *Vehicles:* aeroplane, bicycle, boat, bus, car, motorbike, train
- *Indoor:* bottle, chair, dining table, potted plant, sofa, tv/monitor
- 数据集大小 (2012年): 11.5K 训练/校验图像, 27K边界框, 7K分割图

Slide Credit: S. Lazebnik

http://host.robots.ox.ac.uk/pascal/VOC/

物体检测的性能发展

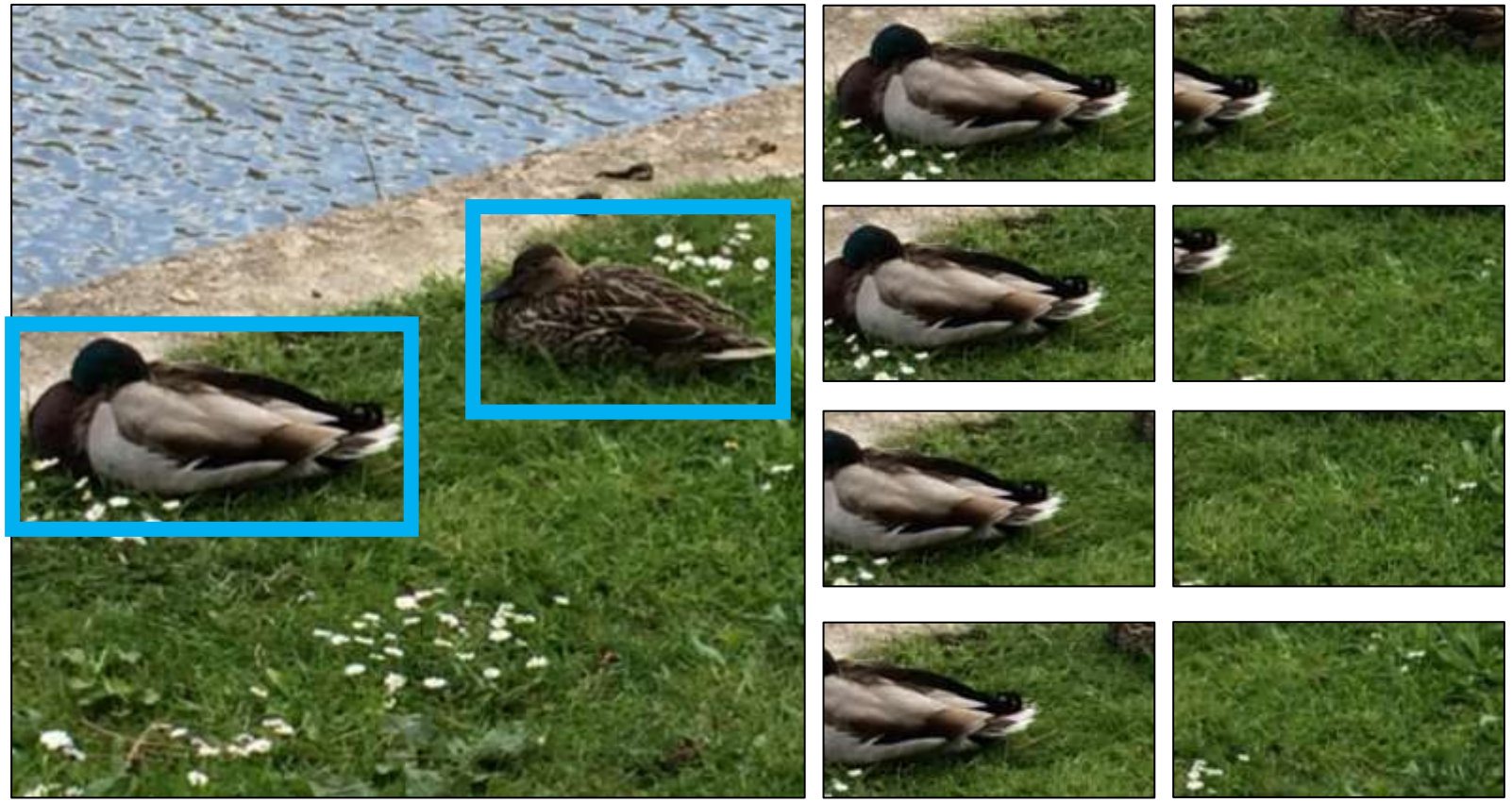

PASCAL VOC
mean Average Precision (mAP)
80%
70%
60%
50%
40%
30%
20%
10%
0%
Before CNNs
Using CNNs
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
year
Source: R. Girshick

回顾：所有的窗口—— 只有少量窗口有目标物体

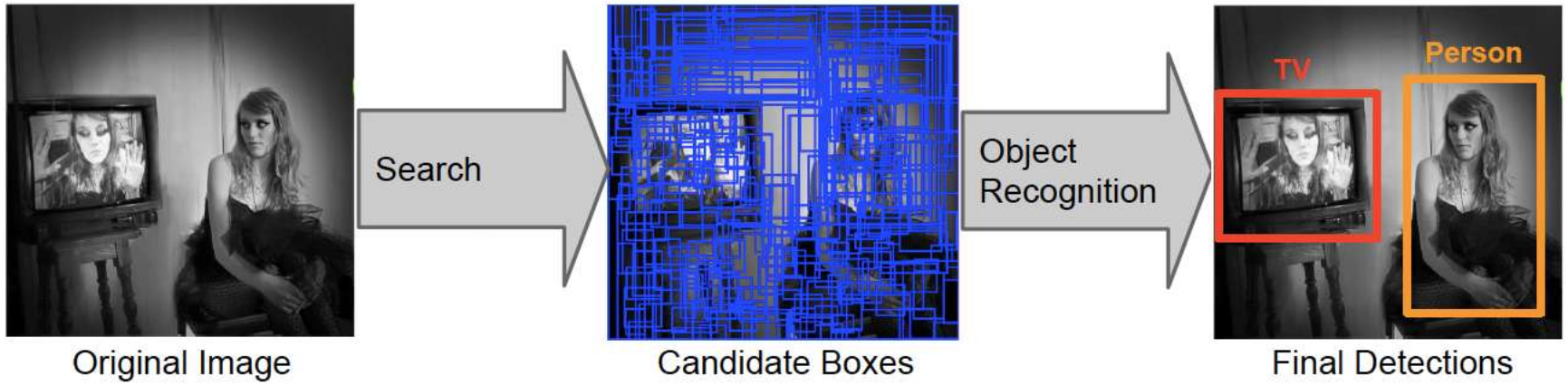
Slide credit: J. Hays

Region Proposals 区域提议

我们需要花时间对草地的所有框进行筛选吗？



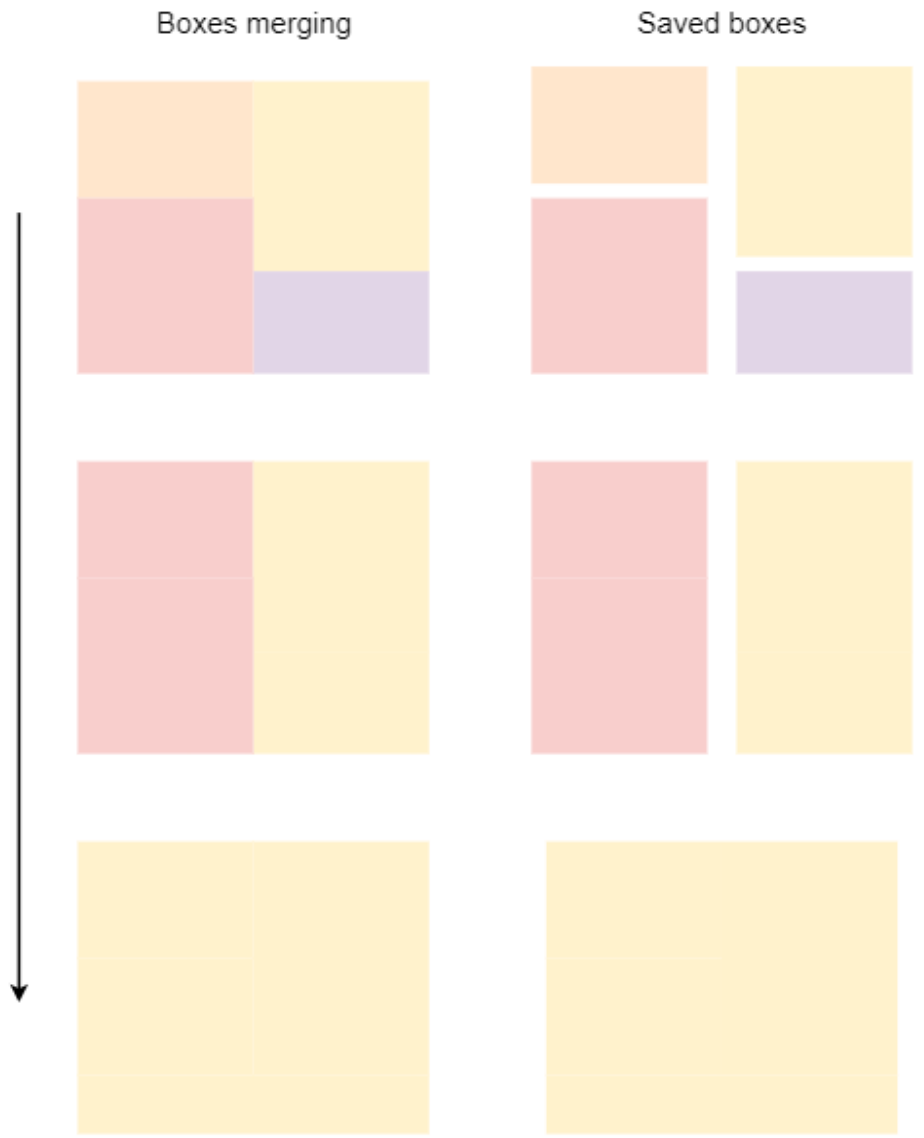
Region Proposals区域提议



- 替代滑动窗口，只选出部分候选框作为*region proposals 区域提议*
  - 谨记：特征提取和分类器速度较慢，区域提议可以减少候选区域，提升运行效率
  - 一般候选区域类别无关
  - 这个机制可以被训练（后面会讲）

Slide Credit: S. Lazebnik

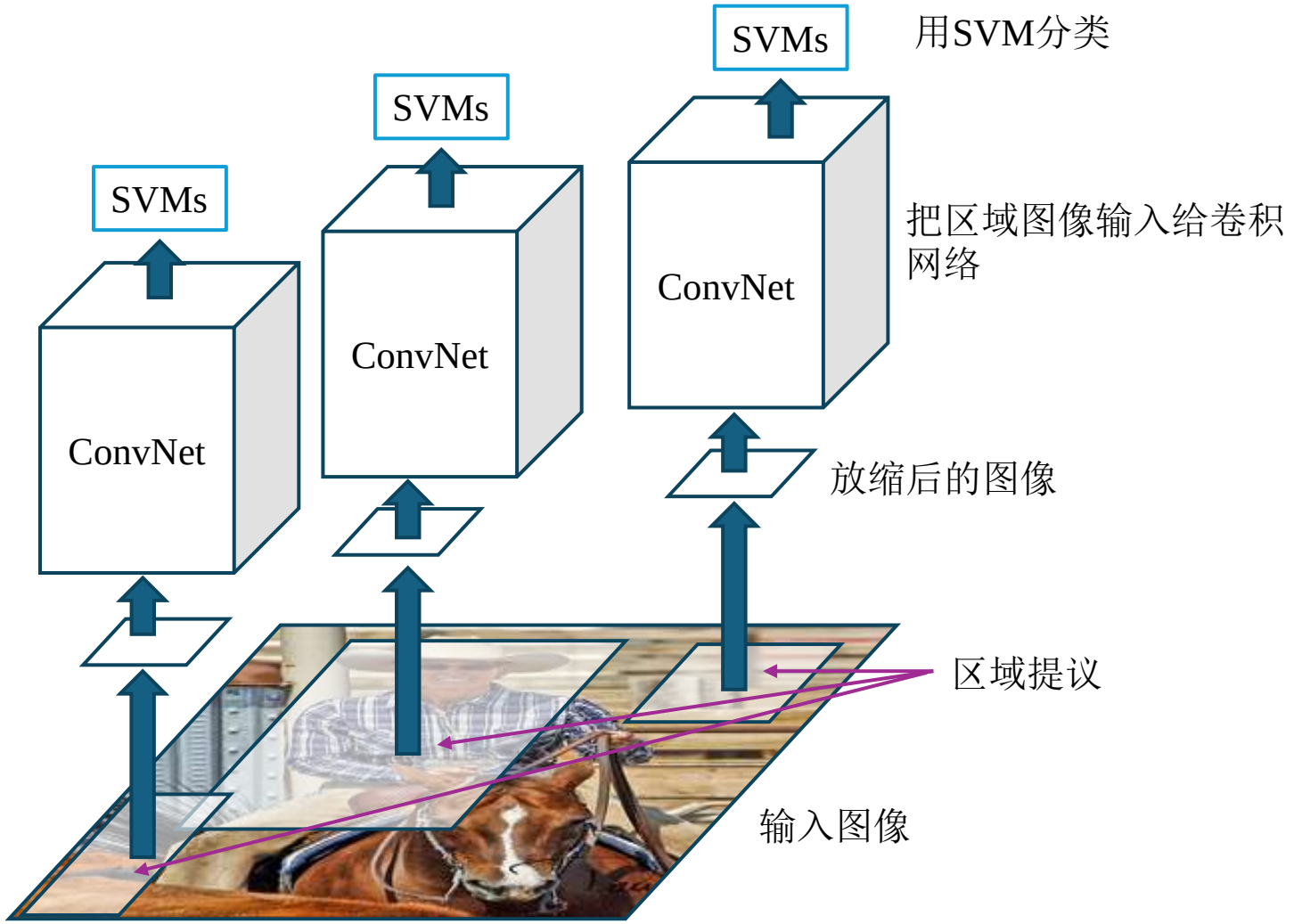
Region Proposals: Selective Search



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014 Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

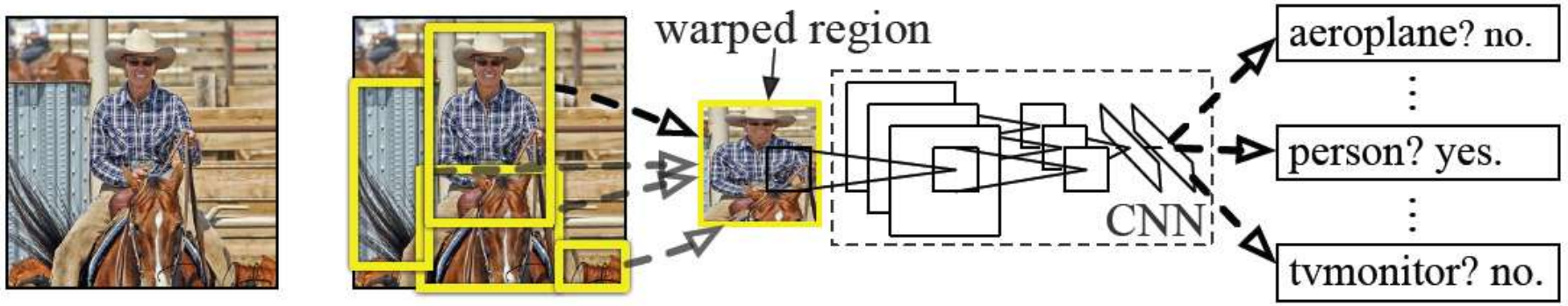
R-CNN: Region proposals + CNN features

Source: R. Girshick



R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014.

R-CNN



- 候选区域: ~2000 Selective Search 候选区域
- 网络: AlexNet 预训练模型，ImageNet上预训练(1000类), PASCAL 上微调 (21类：20+背景)
- 检测器: 放缩图像输入卷积网络, 使用 fc7 层特征 (4096维), 使用SVM分类
- 同时使用SVM回归候选框
- **性能:** mAP of 53.7% on PASCAL 2010 (vs. 35.1% for Selective Search and 33.4% for DPM).

R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014.

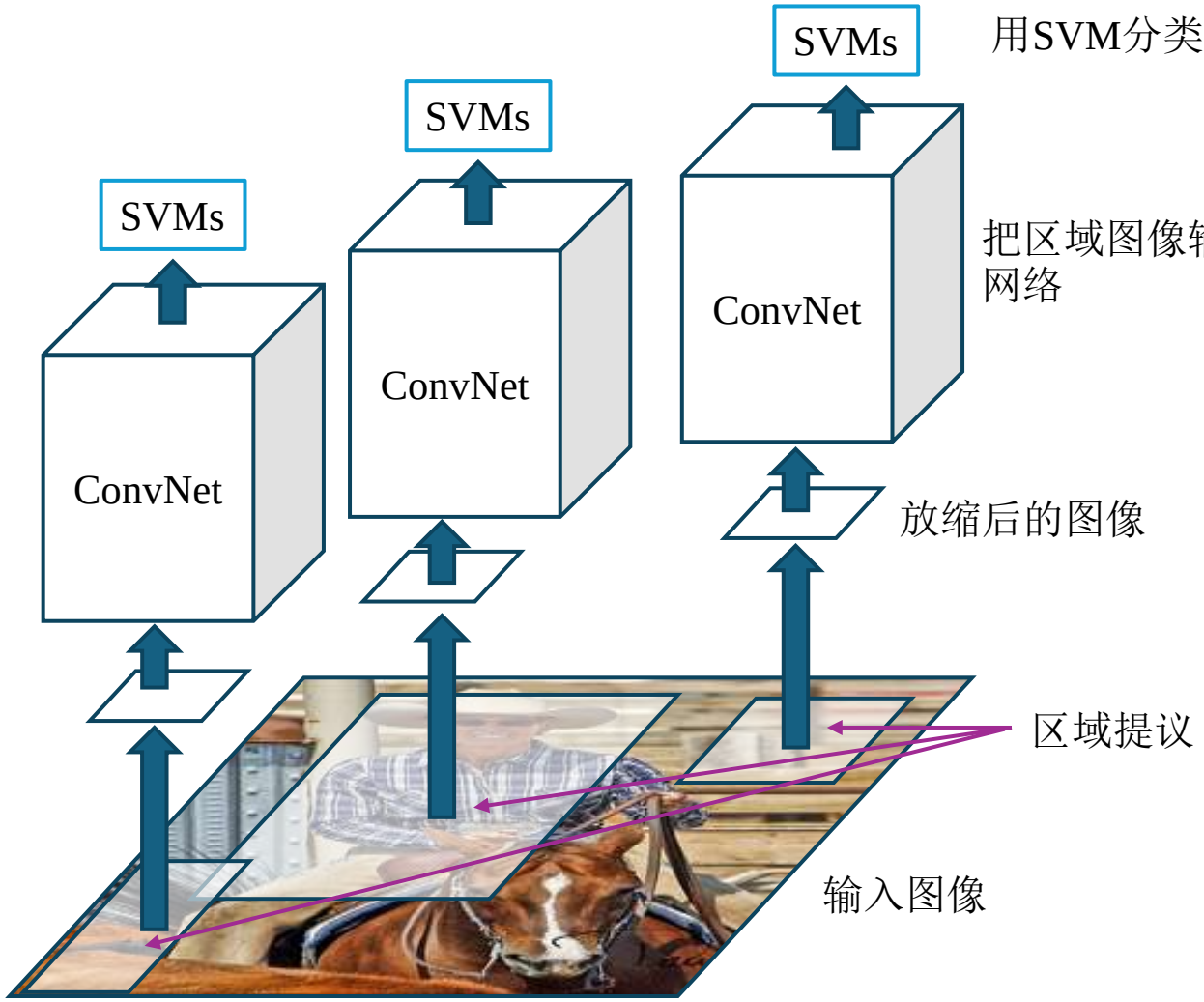
R-CNN pros and cons

- Pros
  - 准!（相较以前的模型）
  - 任何深度学习架构都可以即插即用。
- Cons
  - Ad hoc：没有统一的端到端训练，而是基于不同的需求独立设置。
    - 网络需要用softmax分类器进行微调，这一步使用的是对数损失 (log loss)
    - 训练后处理的线性支持向量机，使用截断损失(hinge loss)
    - 训练后处理的边界框回归使用的是回归损失(least squares)
  - 训练过程慢，需要84小时，且占用大量磁盘空间
    - 每张图像需要2000次CNN
  - 推理（检测）过程慢，使用VGG16模型时，每张图像需要47秒

Slide Credit: S. Lazebnik

“Slow” R-CNN

Source: R. Girshick



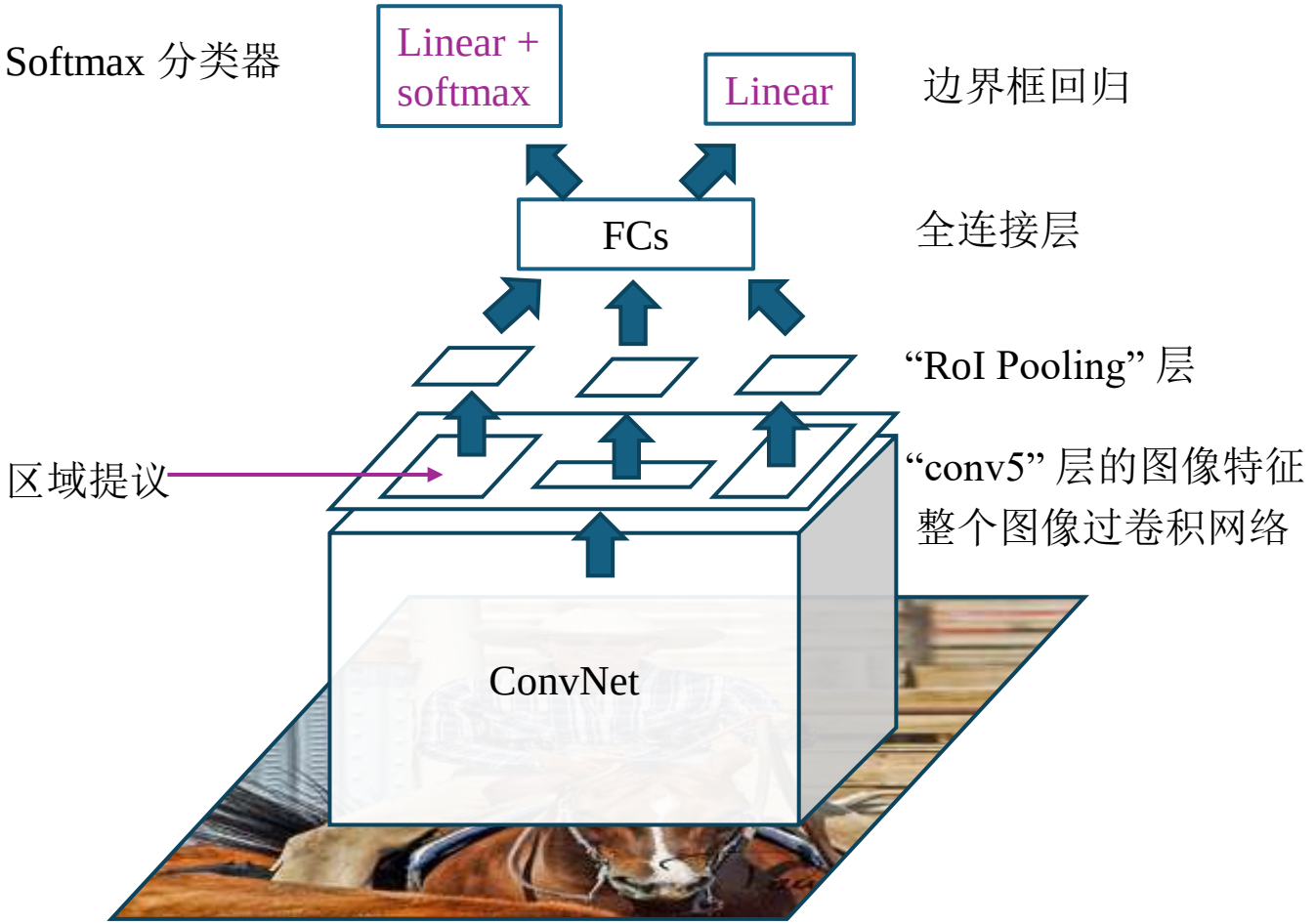
问题: 很慢!每个区域需要重新跑卷积网络

(RoI) SS需要(~2k)个框，都要过卷积

想法: 只跑一次卷积网络，每次提取特征基于这次网络的结果

R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014.

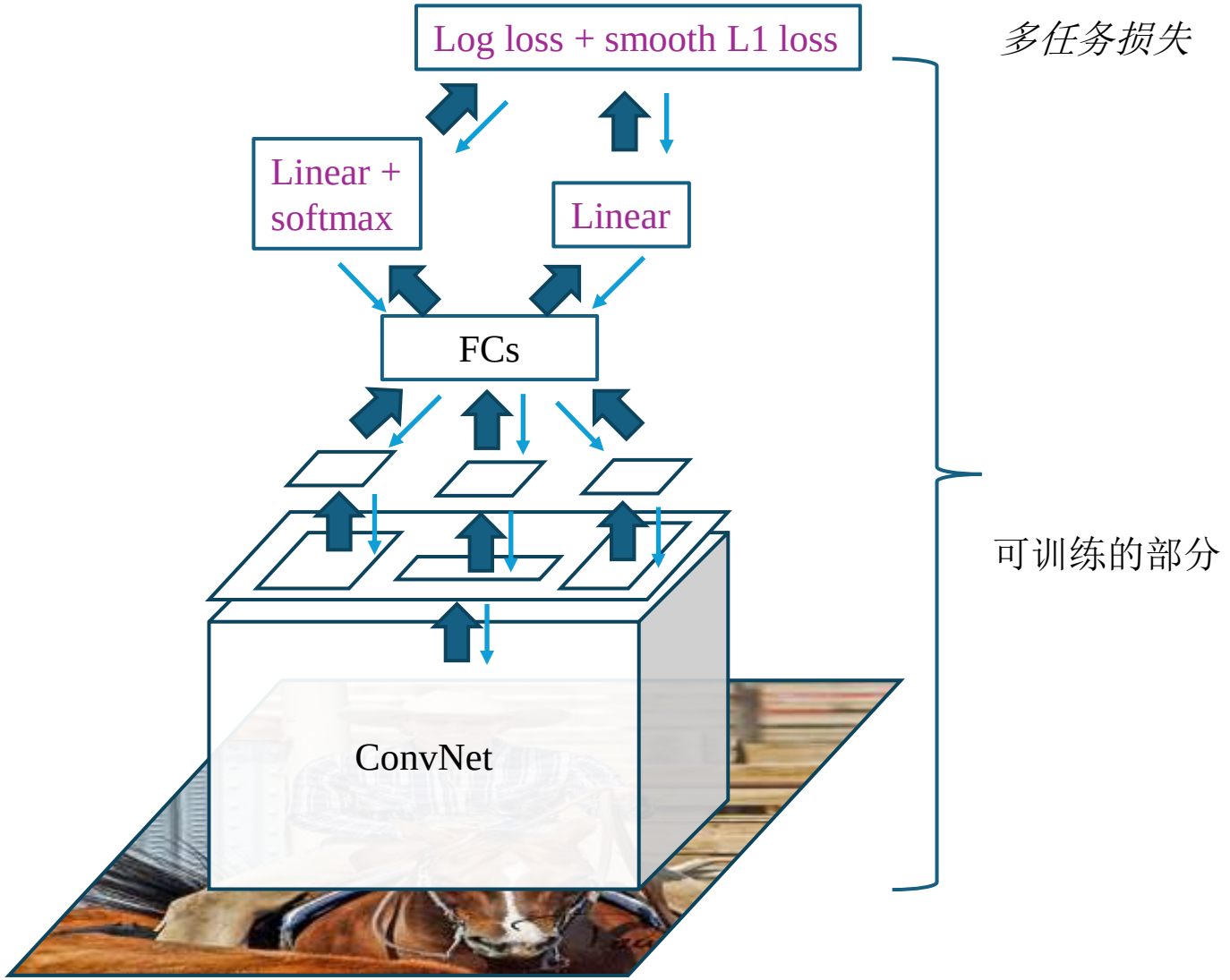
Fast R-CNN



Source: R. Girshick

R. Girshick, Fast R-CNN, ICCV 2015

Fast R-CNN 的训练

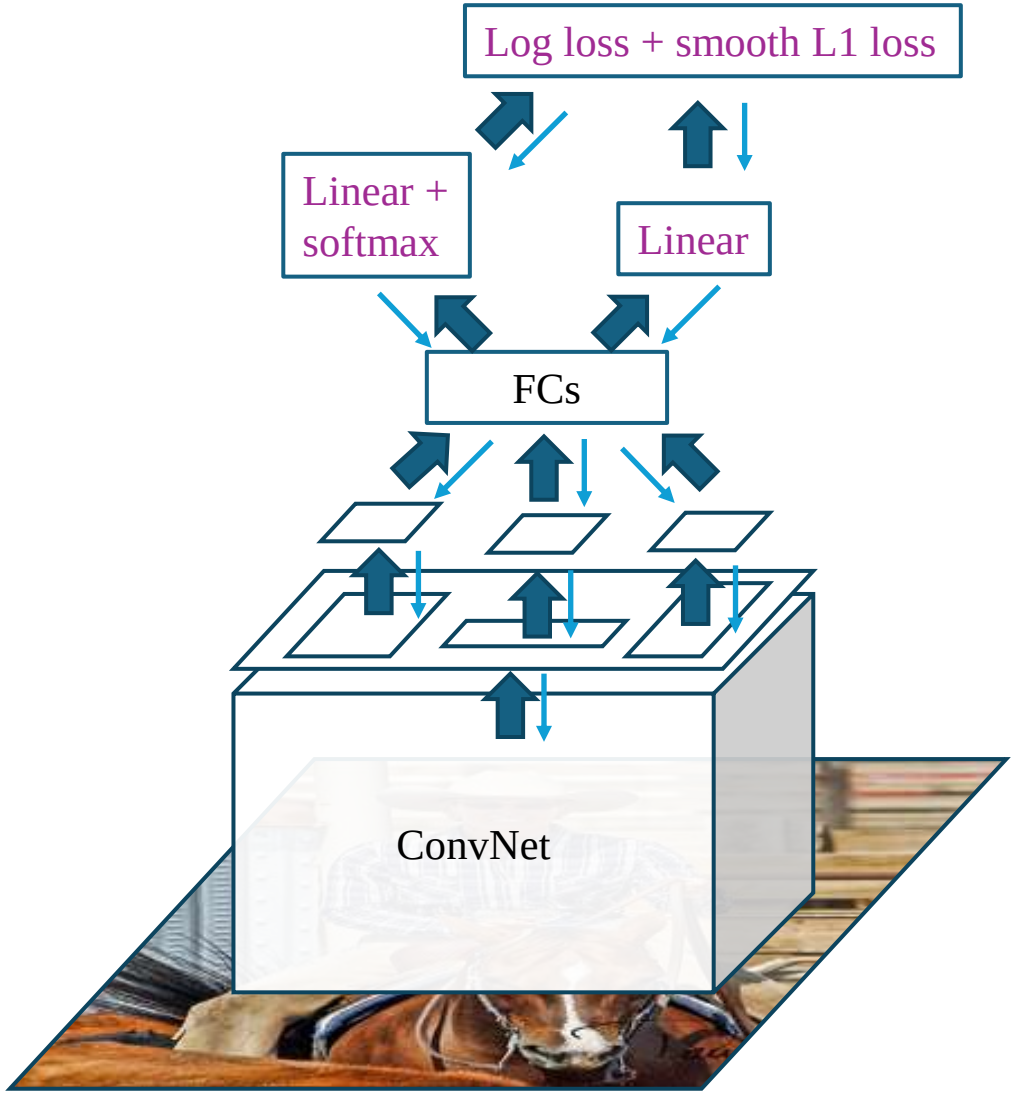


Source: R. Girshick

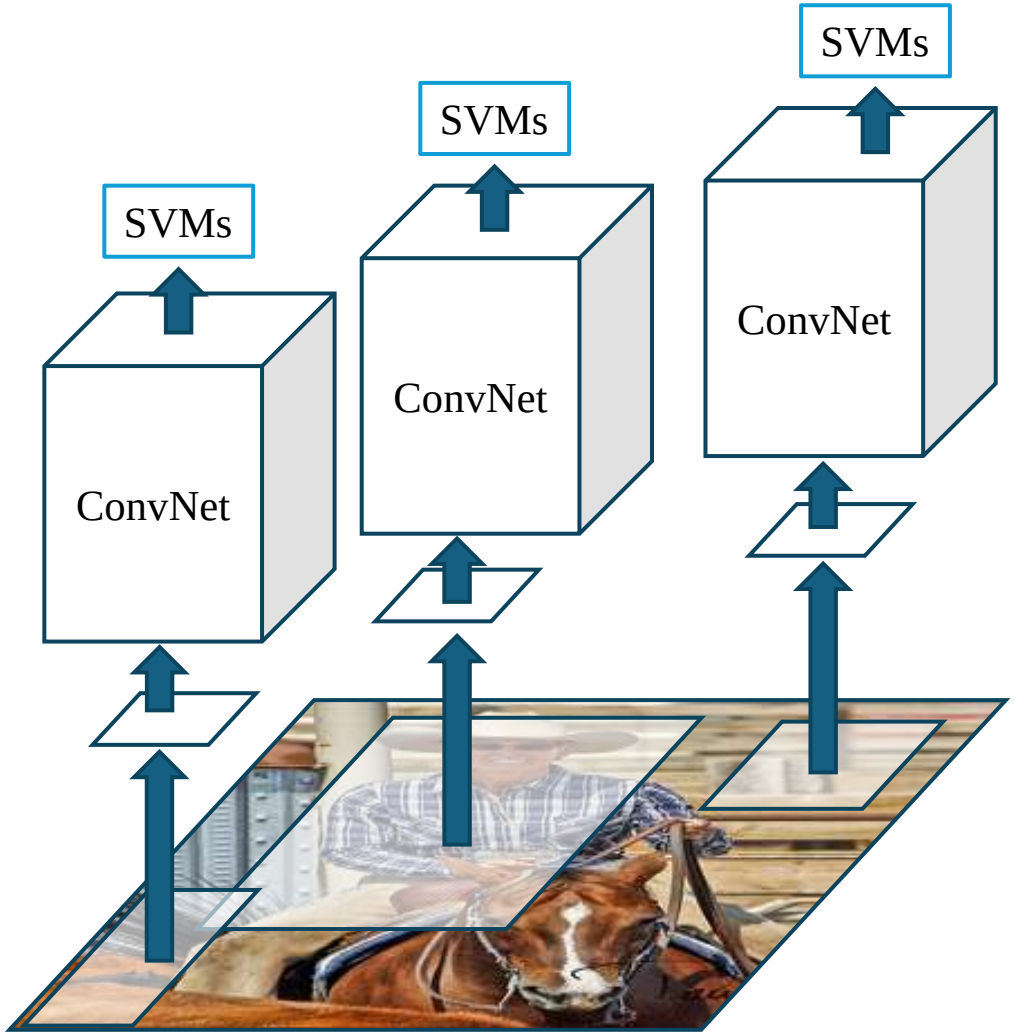
R. Girshick, Fast R-CNN, ICCV 2015

Fast R-CNN VS “Slow” R-CNN

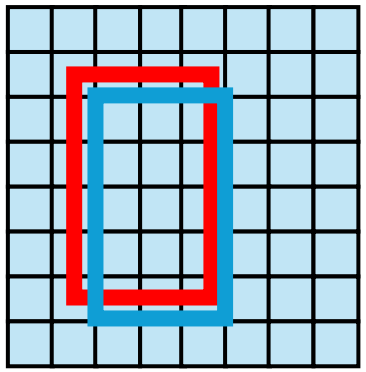
Fast R-CNN



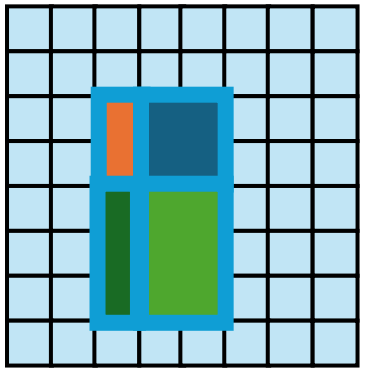
“Slow” R-CNN



Fast R-CNN – ROI-Pool

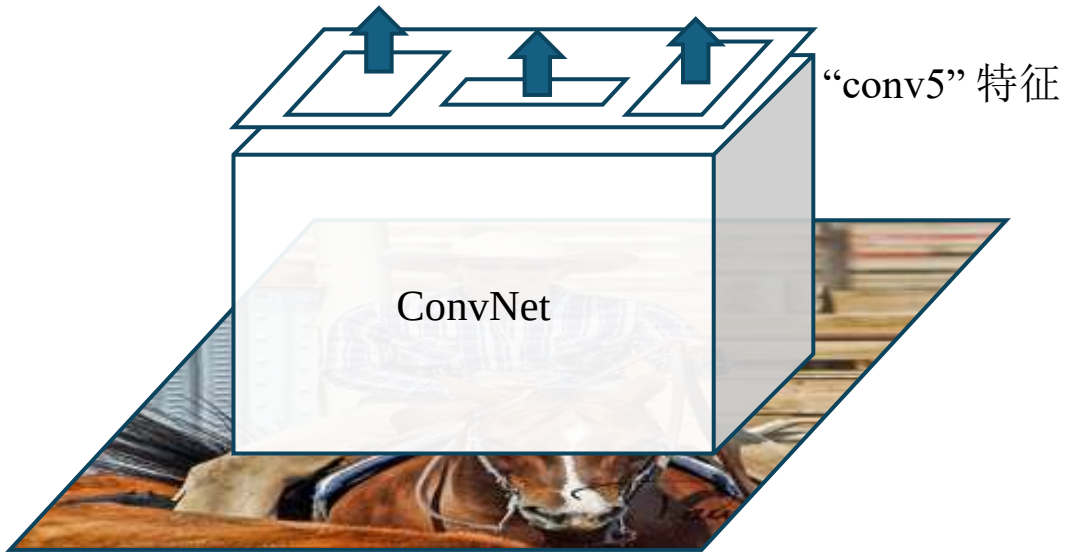


找到网格点



划分

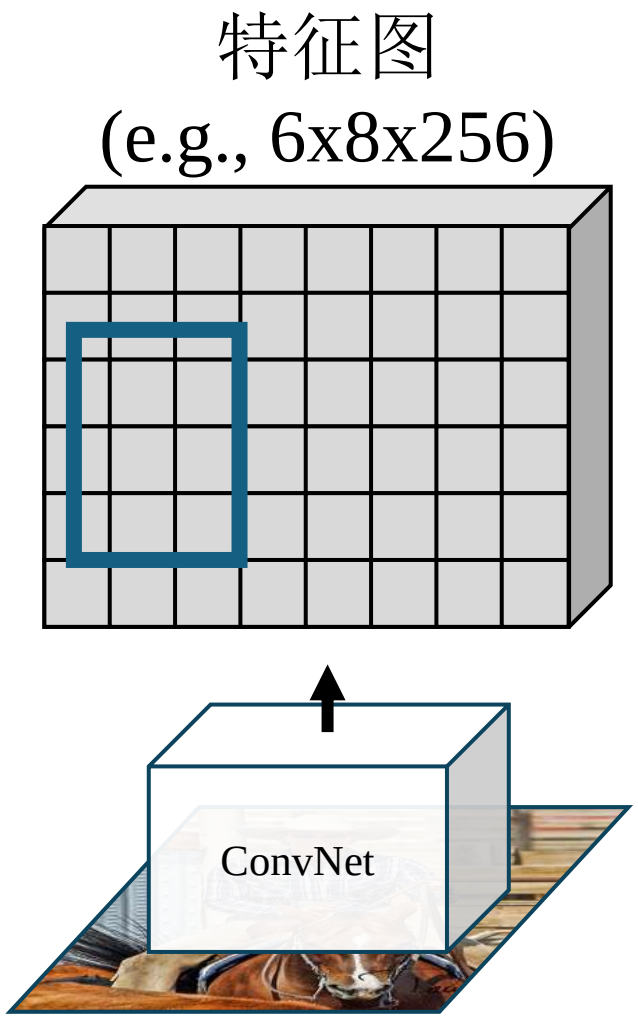
池化



Source: R. Girshick

R. Girshick, Fast R-CNN, ICCV 2015

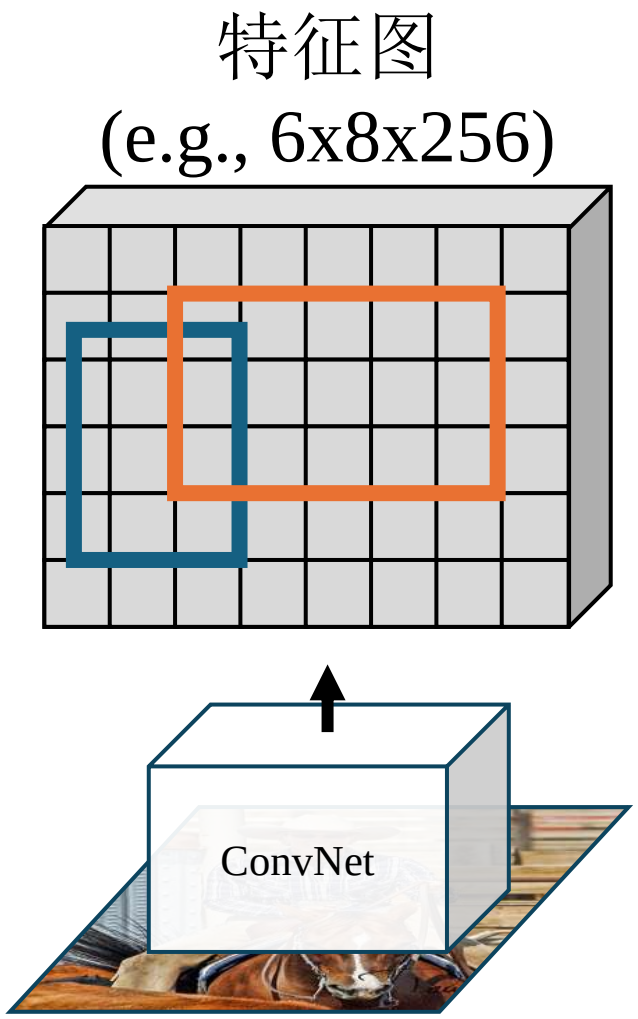
ROI Pooling/Align



对于原图上的边界框，找到在特征图上的位置

- 例如：H=600,W=800
- 特征图大小：H’=6, W’=8
- 边界框： left x=50, top y=150, width=250, height=350
- 特征图对应的框：left x=0.5, y=1.5, width=2.5, height=3.5

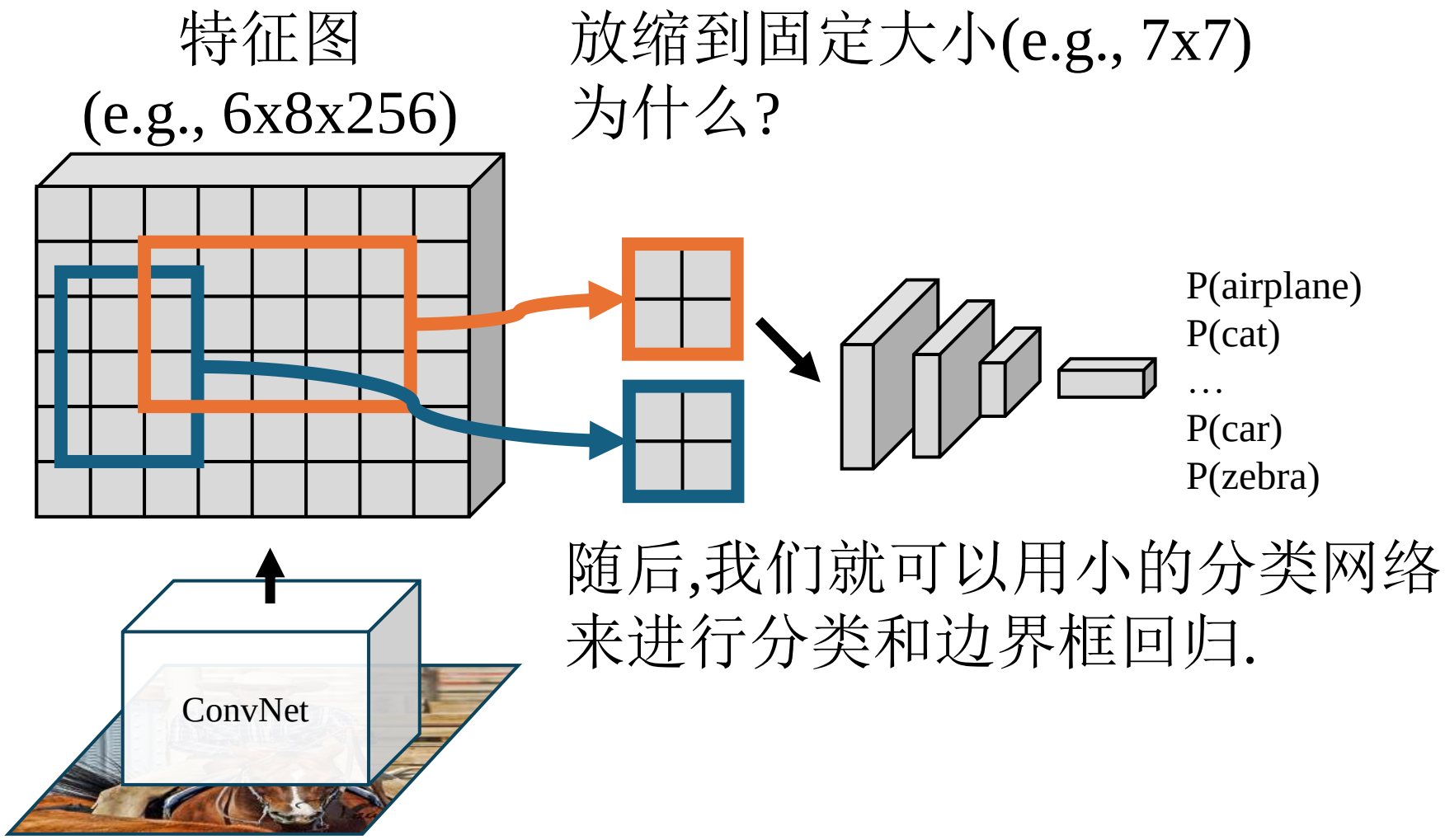
ROI Pooling/Align



对于原图上的边界框，找到在特征图上的位置

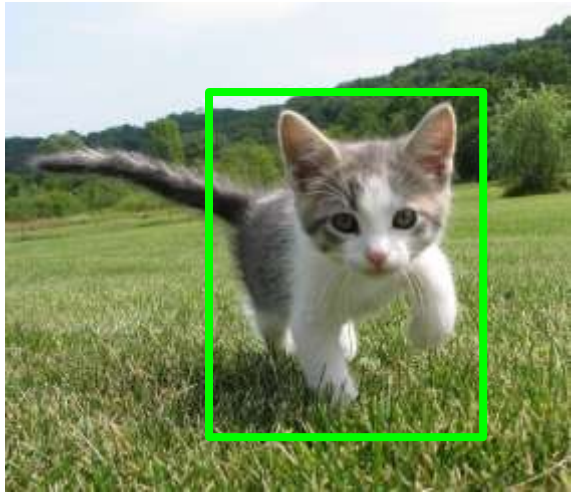
- 例如：H=600,W=800
- 特征图大小：H’=6, W’=8
- 边界框： left x=50, top y=150, width=250, height=350
- 特征图对应的框：left x=0.5, y =1.5, width=2.5, height=3.5
- 另外一个例子: left x=2, top y = 1, width=5, height=3
- 对应原图边界框是？

ROI Pooling/Align

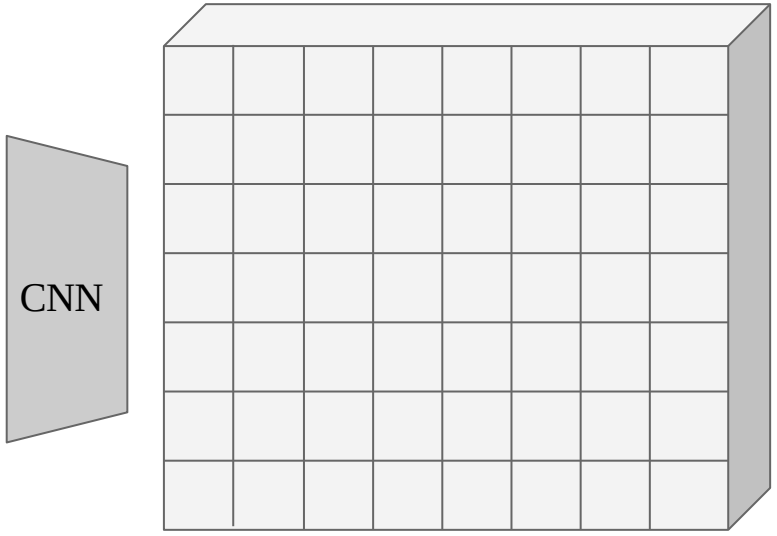


随后,我们就可以用小的分类网络来进行分类和边界框回归.

Fast R-CNN 裁剪 特征: RoI Pool



输入图像 (e.g. 3 x 640 x 480)

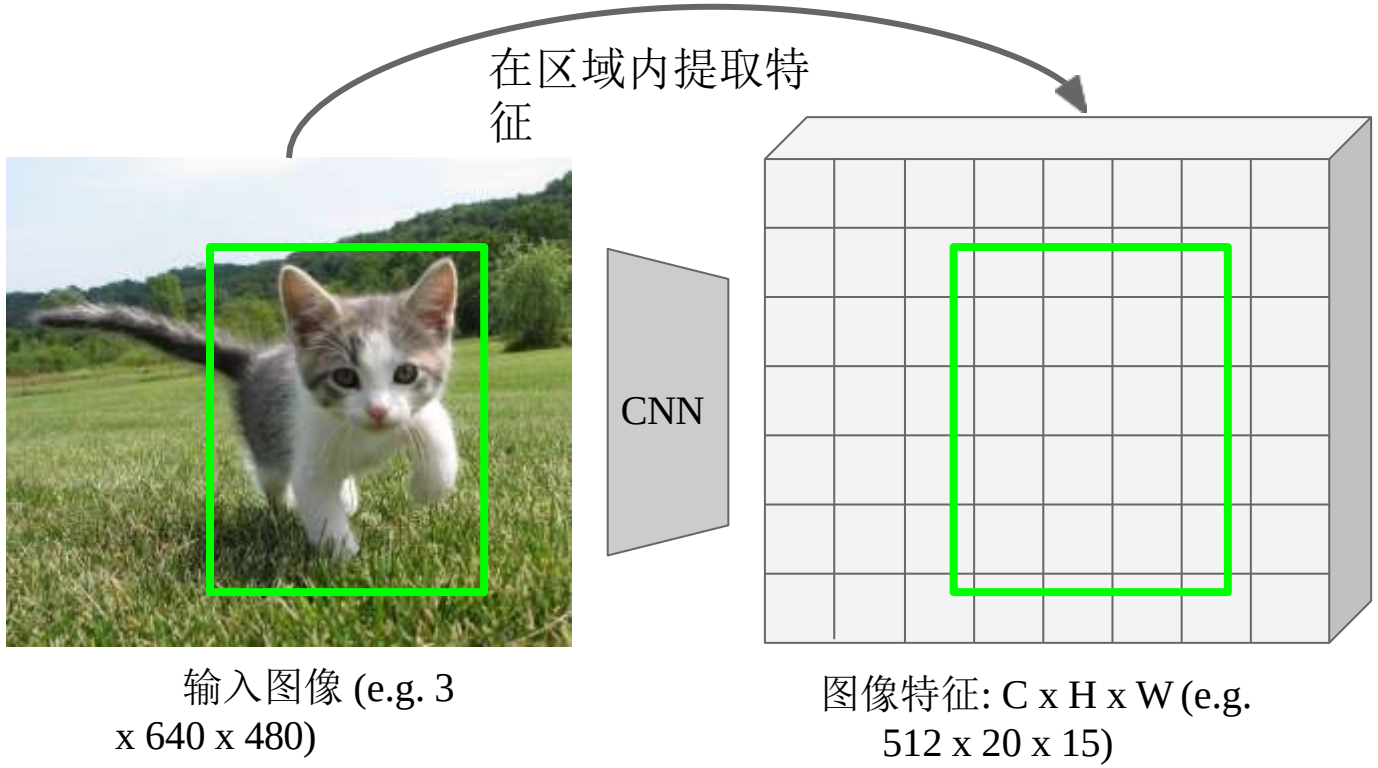


图像特征: C x H x W (e.g. 512 x 20 x 15)

Girshick, “Fast R-CNN”, ICCV 2015.

Girshick, “Fast R-CNN”, ICCV 2015.

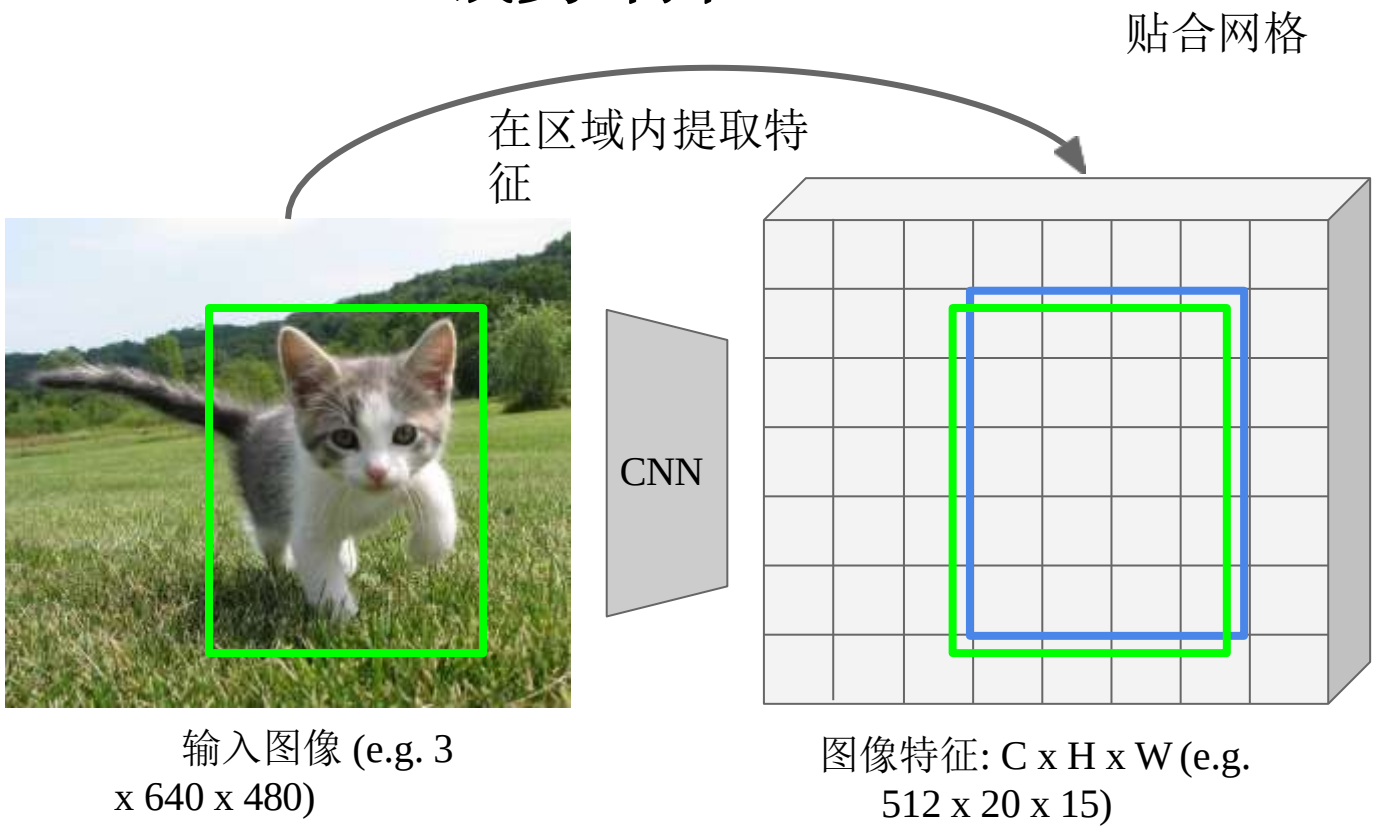
Fast R-CNN 裁剪 特征: RoI Pool



Girshick, “Fast R-CNN”, ICCV 2015.

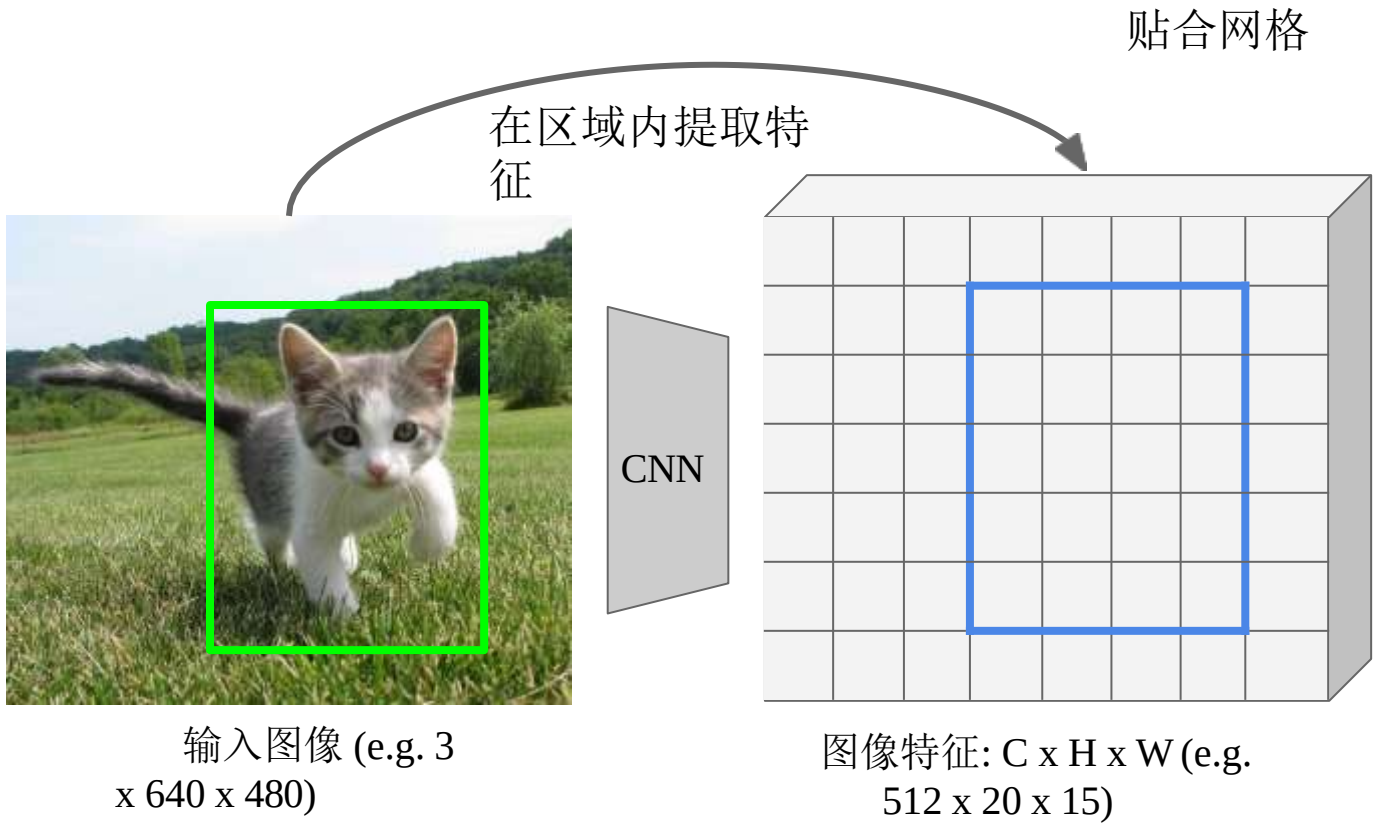
Girshick, “Fast R-CNN”, ICCV 2015.

Fast R-CNN 裁剪 特征: RoI Pool



Girshick, “Fast R-CNN”, ICCV 2015.

Fast R-CNN 裁剪 特征: RoI Pool



怎么把 512 x 5 x 4 的区域特征 放缩到512x 2 x 2?

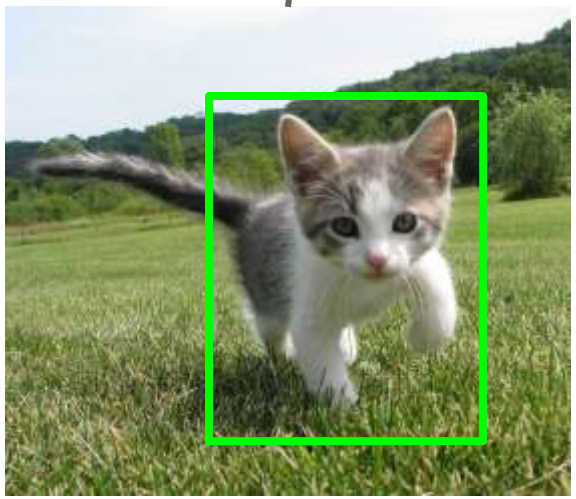
Girshick, “Fast R-CNN”, ICCV 2015.

Fast R-CNN裁剪 特征: RoI Pool

贴合网格

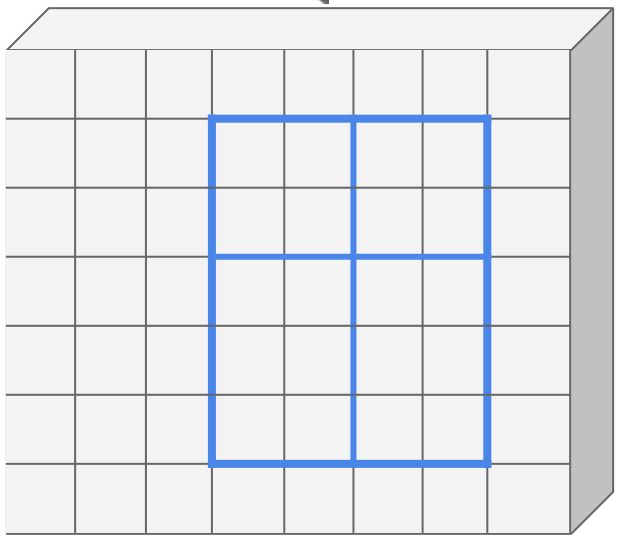
分成2x2的近似相同大小的区域

在区域内提取特征



输入图像 (e.g. 3 x 640 x 480)

CNN

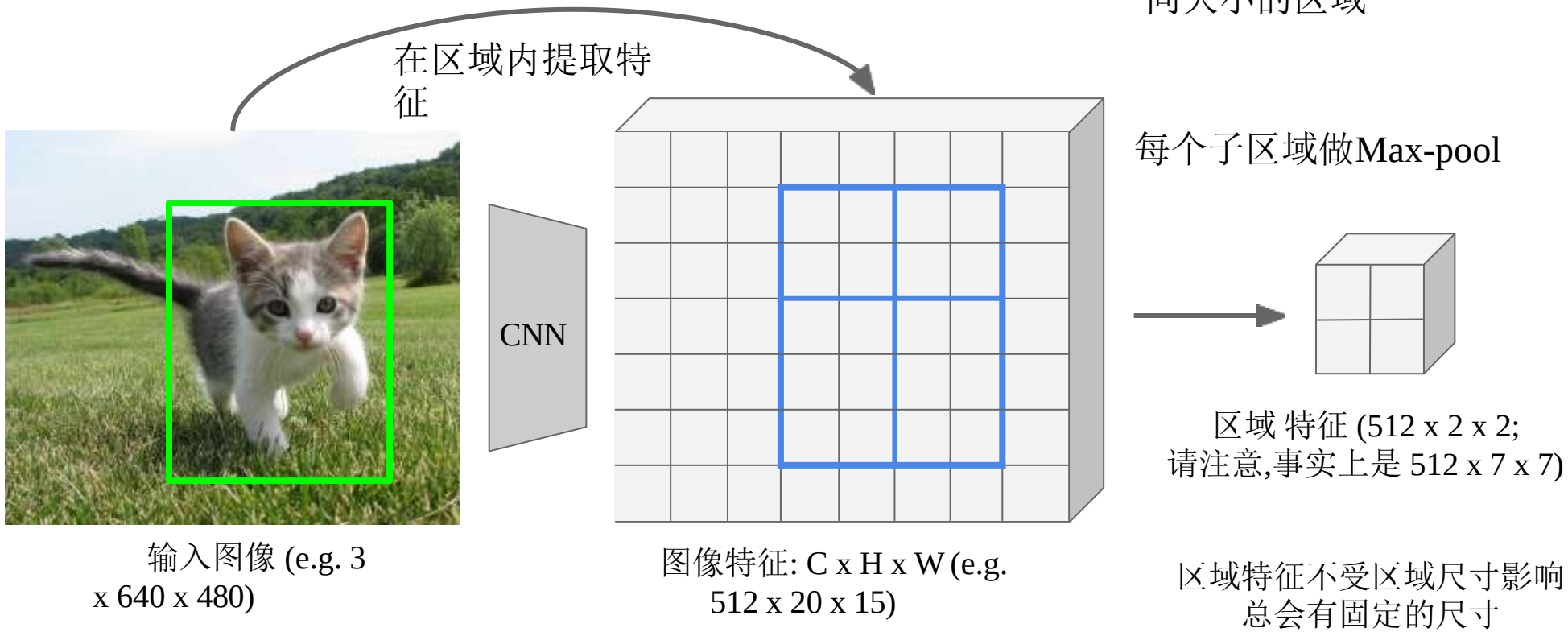


图像特征: C x H x W (e.g. 512 x 20 x 15)

怎么把 512 x 5 x 4 的区域特征 放缩到512x 2 x 2?

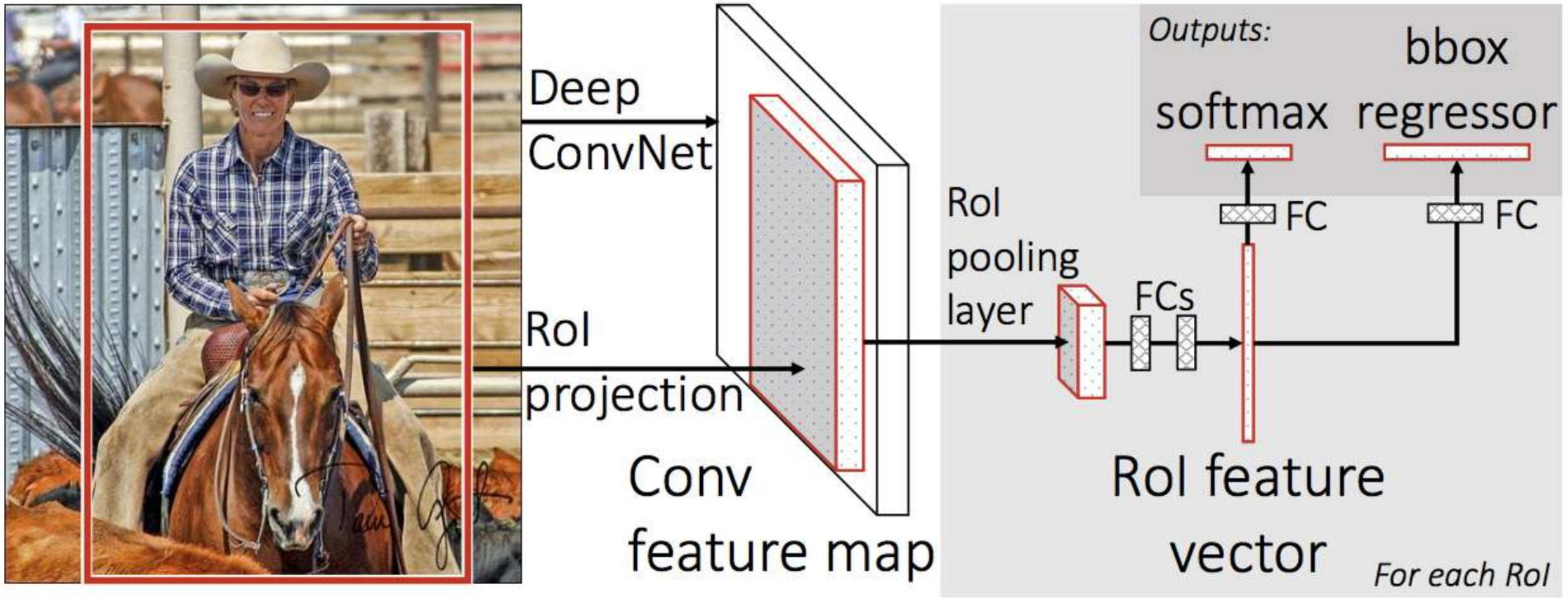
Girshick, “Fast R-CNN”, ICCV 2015.

Fast R-CNN裁剪 特征: RoI Pool



Girshick, “Fast R-CNN”, ICCV 2015.

Fast R-CNN: 在回顾一遍

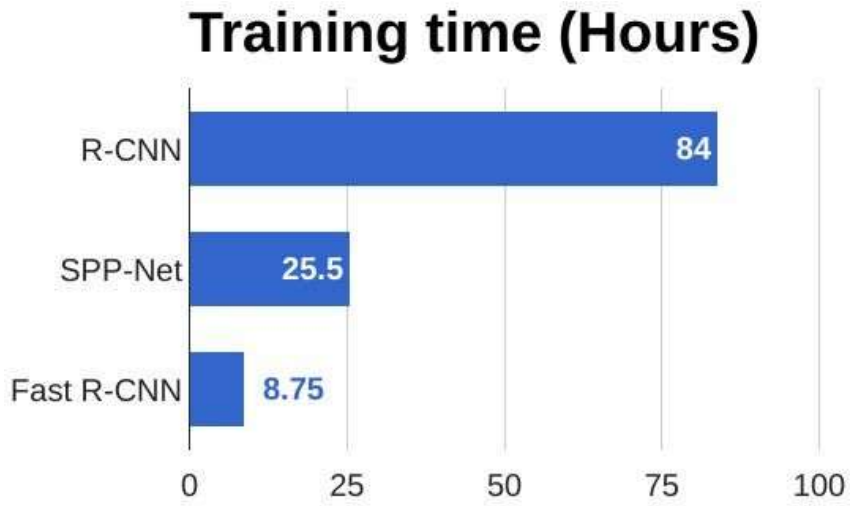


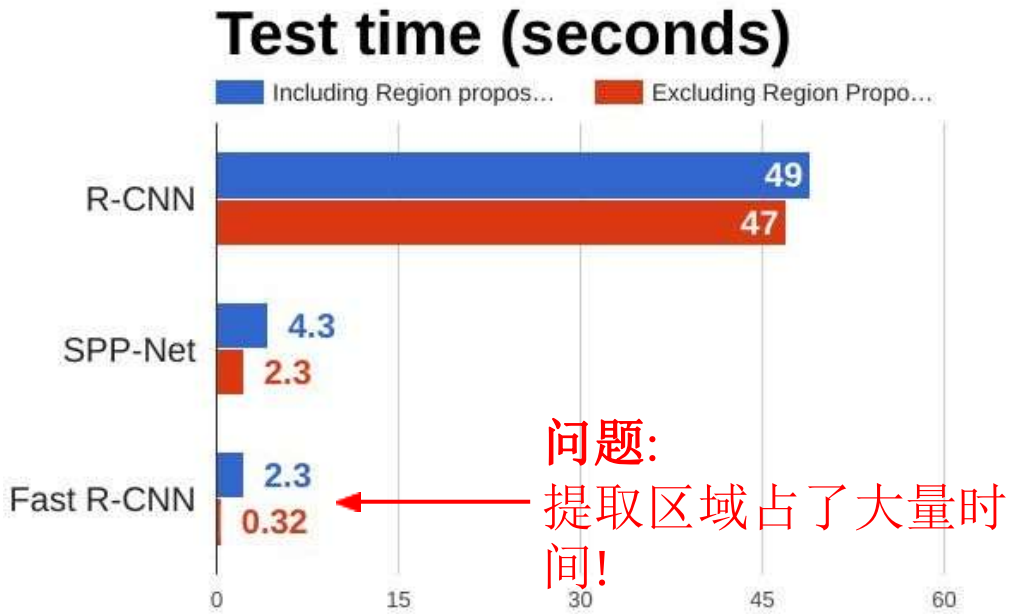
Source: R. Girschick

R. Girshick, Fast R-CNN, ICCV 2015

Fast R-CNN 性能

	Fast R-CNN	R-CNN
Train time (h)	**9.5**	84
- Speedup	**8.8x**	1x
Test time / image	**0.32s**	47.0s
Test speedup	**146x**	1x
mAP	**66.9%**	66.0%



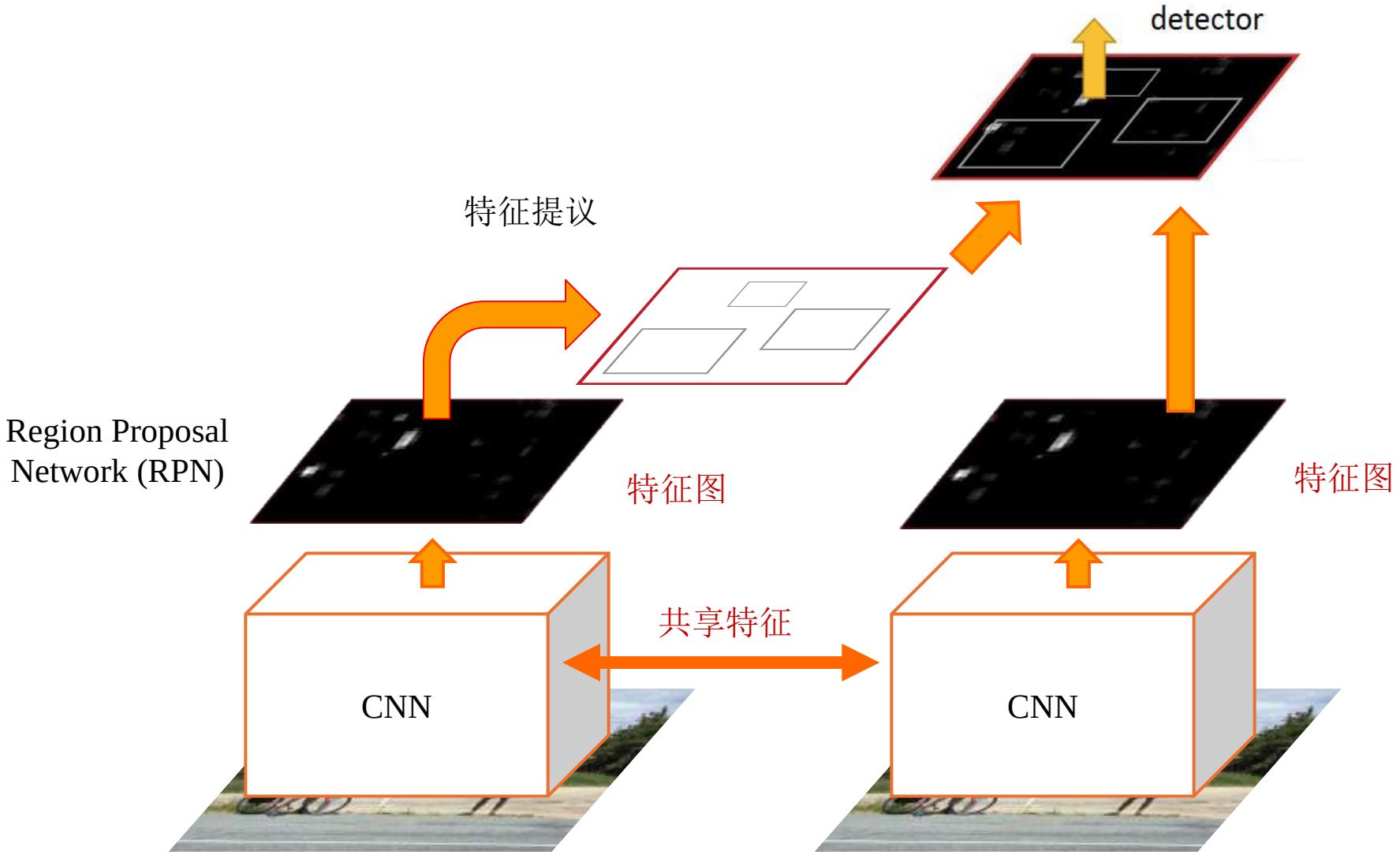


问题:
提取区域占了大量时间!

左边的表里没有计算区域提议的时间.
使用 VGG16 网络

Source: R. Girshick

Faster R-CNN

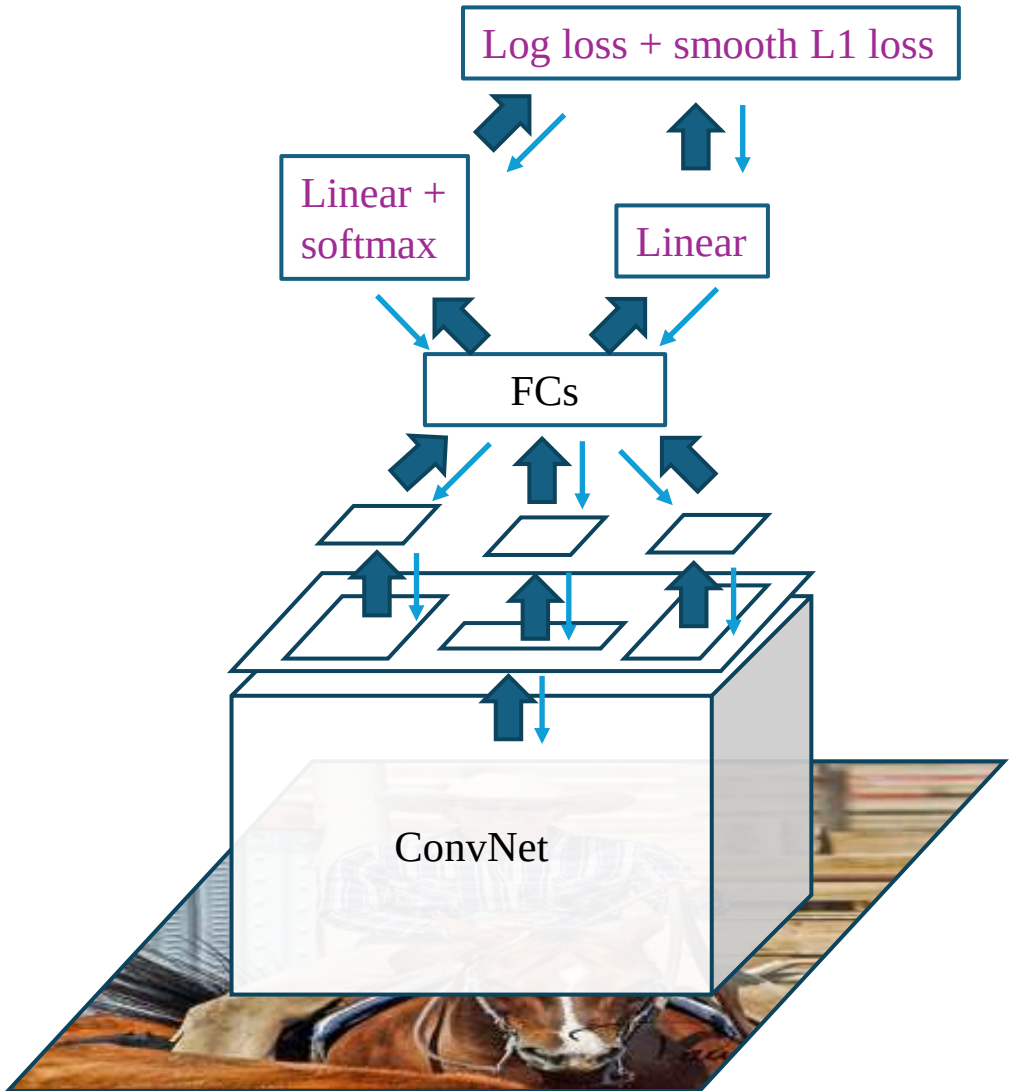


S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS 2015

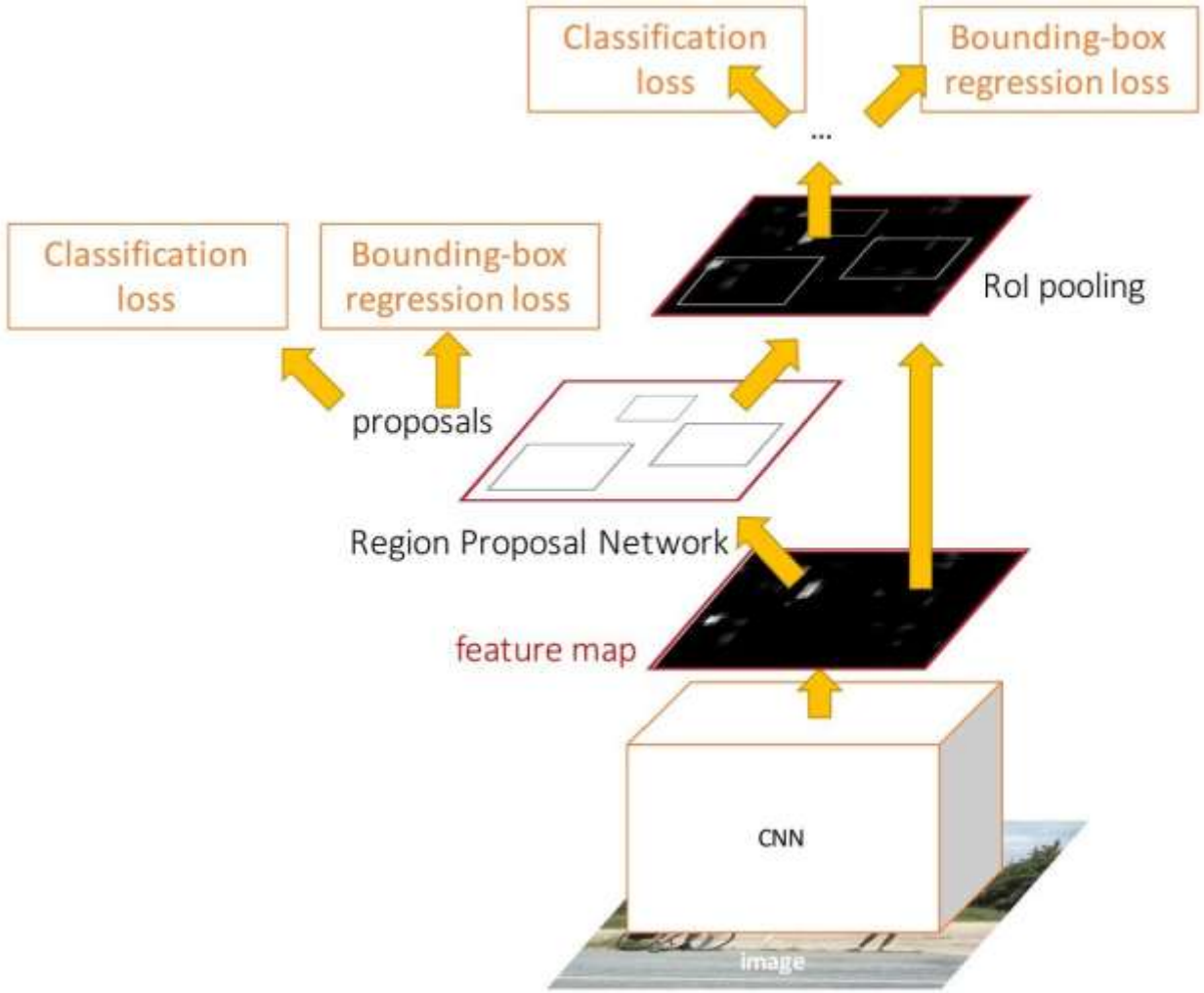
Faster R-CNN:
让CNN提Proposals！

Region Proposal Network (RPN) 预测区域proposal
其余的与Fast R-CNN一致

Fast R-CNN

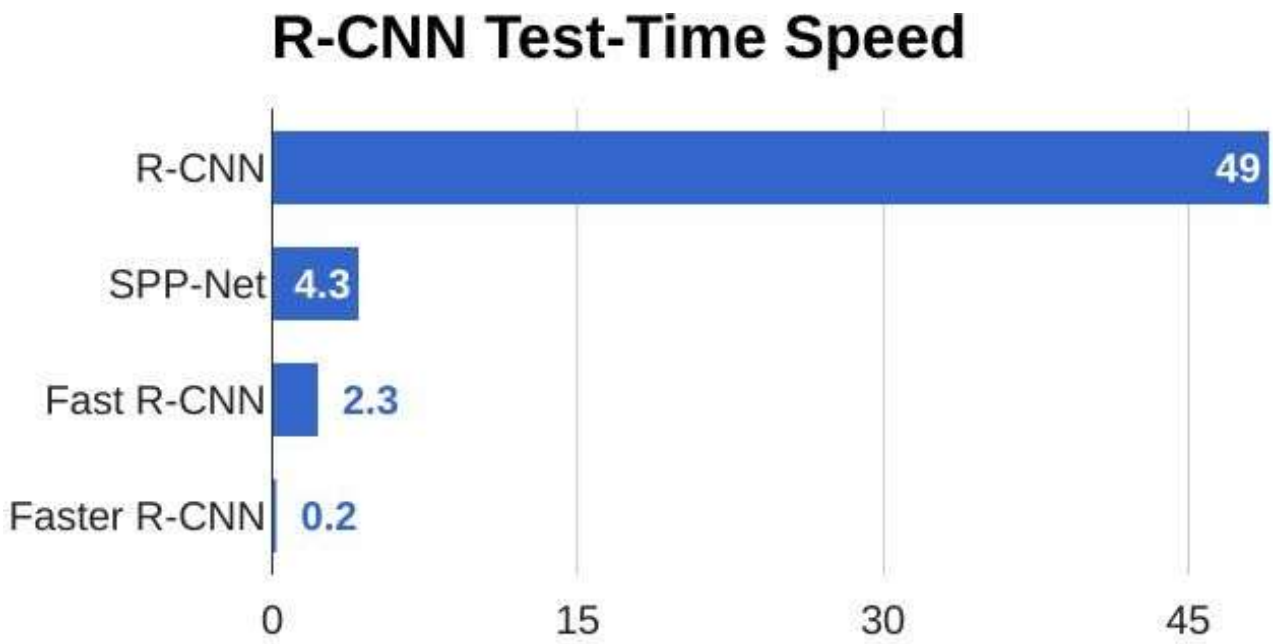


Faster R-CNN

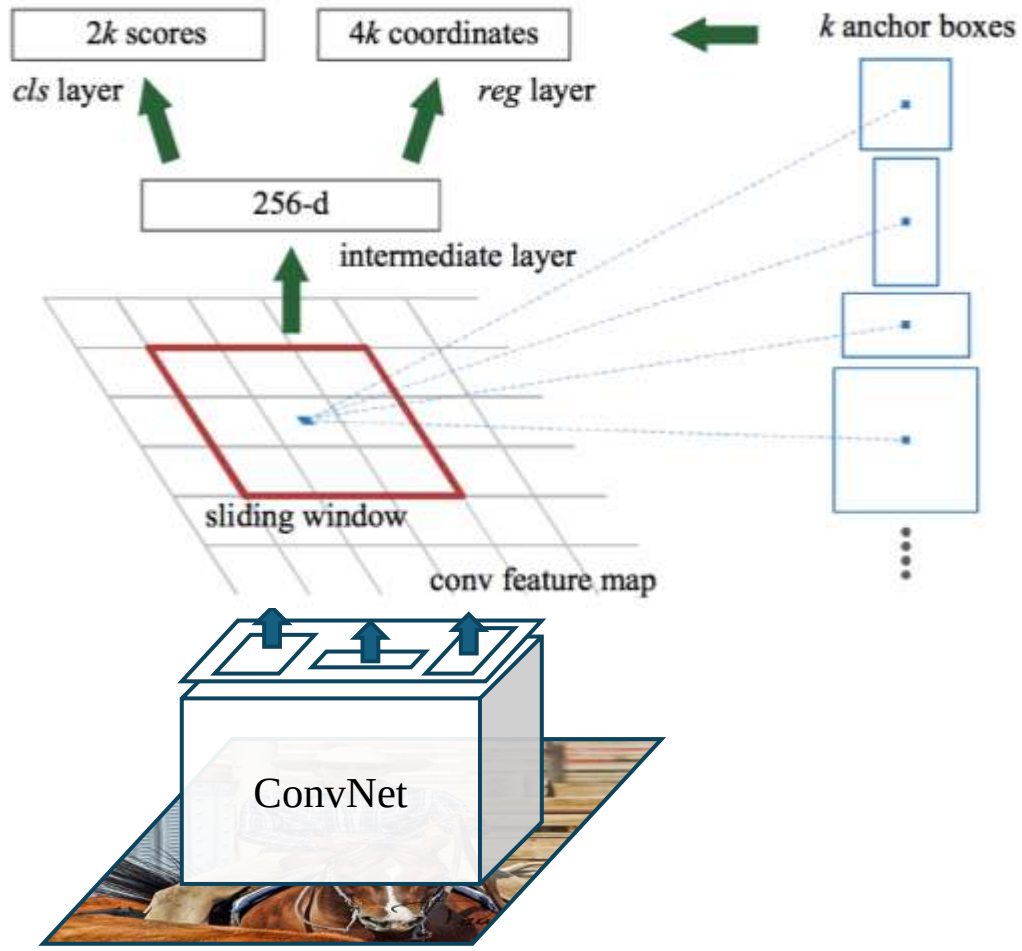


Fast**er** R-CNN:

让CNN提Proposals！



Region Proposal Network (RPN) 区域提议网络



Source: R. Girshick

基于conv5 特征的小网络模块.

预测:

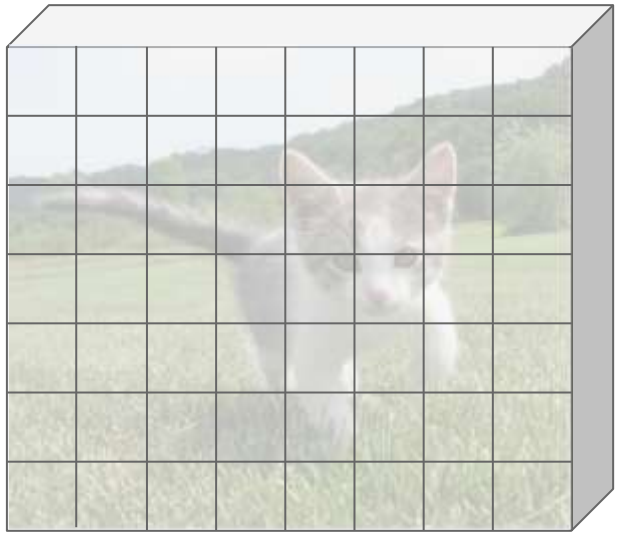
- 是否包括物体 (classification),
  - IoU>0.7: 包含物体
  - IoU<0.3: 不包含
  - 其他: 忽略
- 区域是否需要校正 (regression)

k 个 根据 特征图提取的 “anchors”.

Region Proposal Network

输入图像 (e.g. 3 x 640 x 480)

CNN

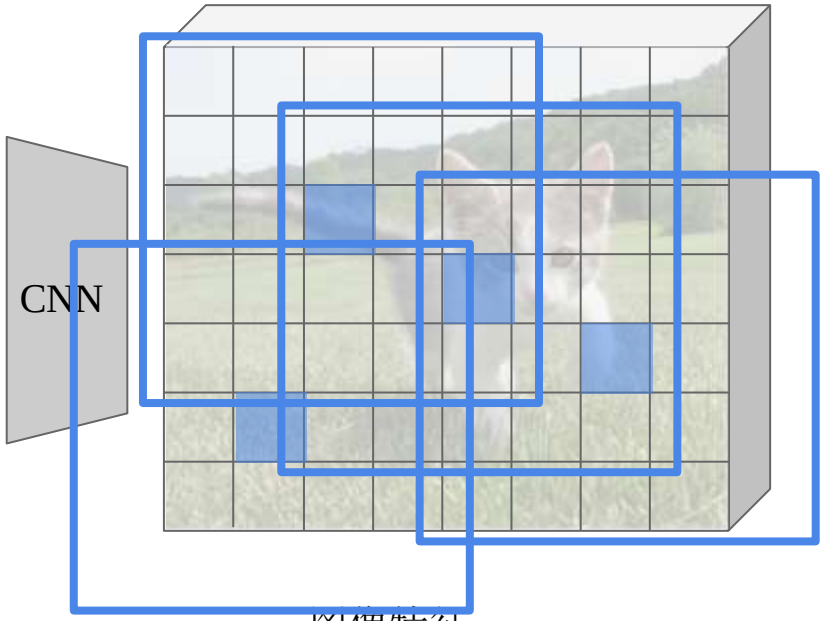


图像特征 (e.g. 512 x 20 x 15)

Region Proposal Network

想象一个有固定尺寸
anchor框

输入图像
(e.g. 3 x 640 x 480)

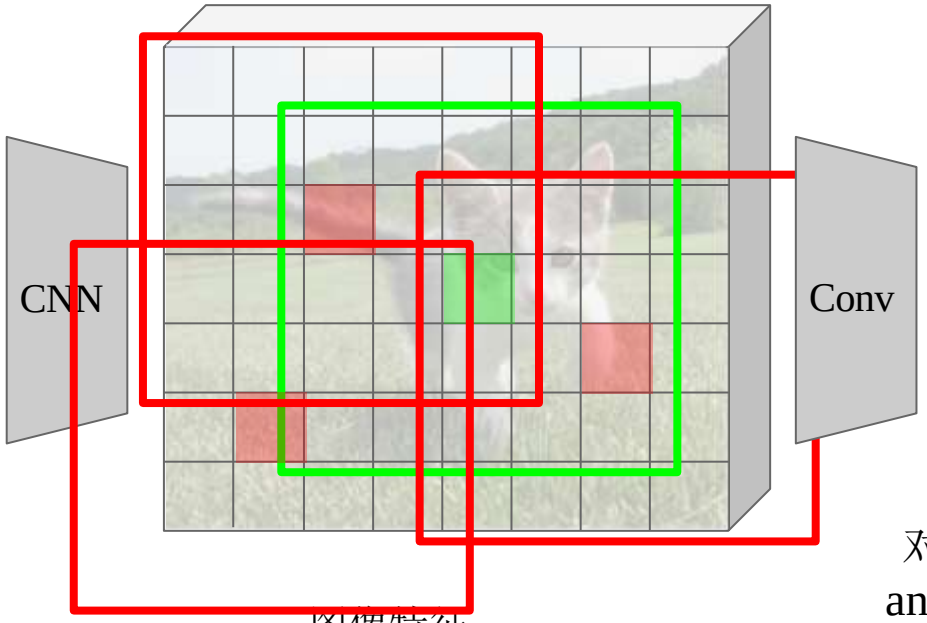


图像特征
(e.g. 512 x 20 x 15)

Region Proposal Network

想象一个有固定尺寸
anchor框

输入图像
(e.g. 3 x 640 x 480)



图像特征
(e.g. 512 x 20 x 15)

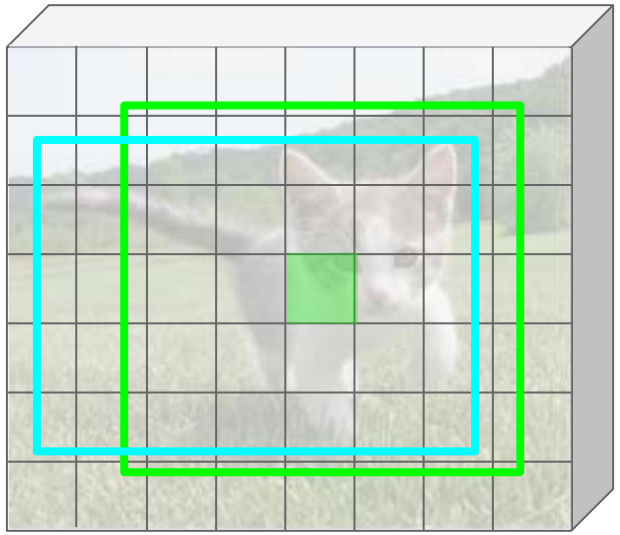
Anchor包含物体吗?
1 x 20 x 15

对于每一个点，预测该
anchor是否包含一个物体
（二分类）

Region Proposal Network

想象一个有固定尺寸
anchor框

CNN



Conv

Anchor包含物体吗?
1 x 20 x 15

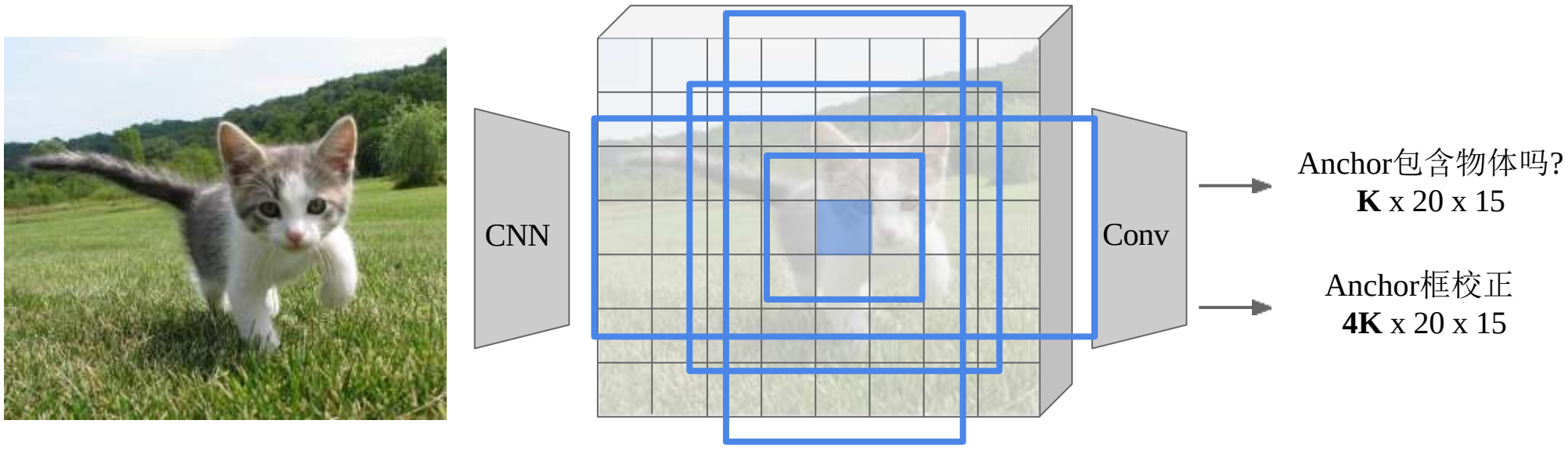
Anchor框校正
4 x 20 x 15

对于预测有物体的框，预测一个
对anchor框的校正

图像特征
(e.g. 512 x 20 x 15)

Region Proposal Network

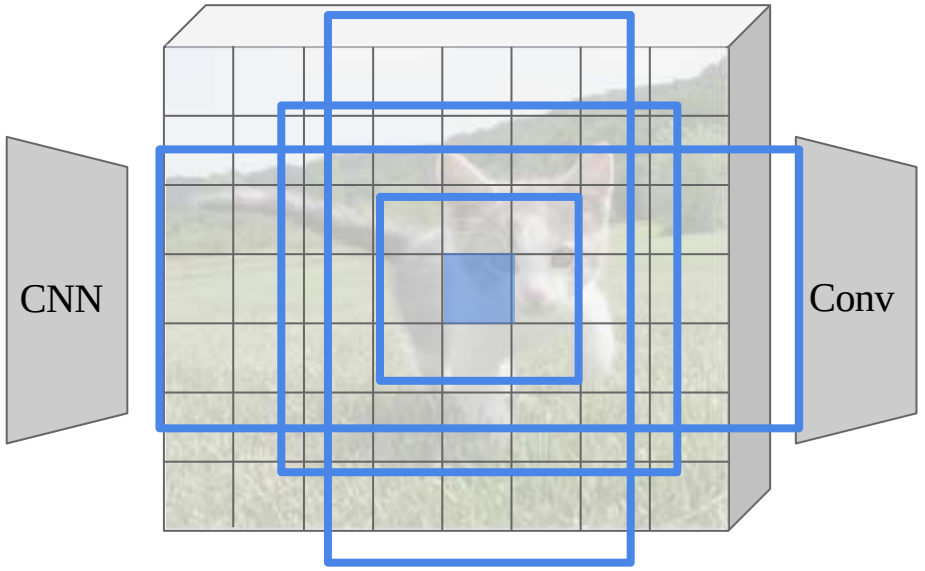
K个anchor对应不同尺寸\、尺度\形状的物体



Region Proposal Network

K个anchor对应不同尺寸\、尺度\形状的物体

输入图像
(e.g. 3 x 640 x 480)



图像特征
(e.g. 512 x 20 x 15)

Anchor包含物体吗?
K x 20 x 15

Anchor框校正
4K x 20 x 15

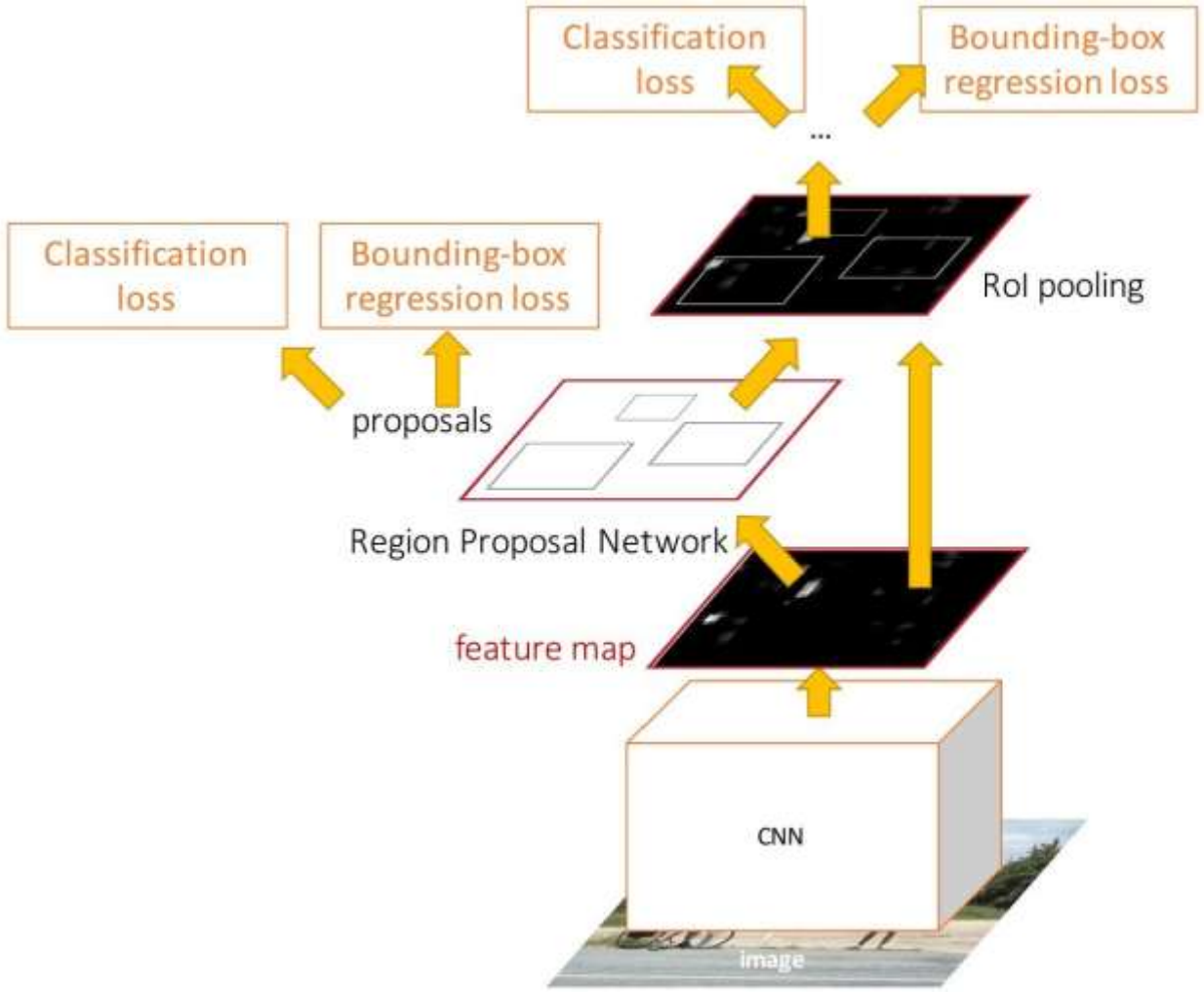
对K*20*15 个框“物体存在的概率” 排序，取~300个进行预测,映射回原图

Faster R-CNN:

让CNN提Proposals！

训练4 个 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classify score (object classes)
4. Final box coordinates

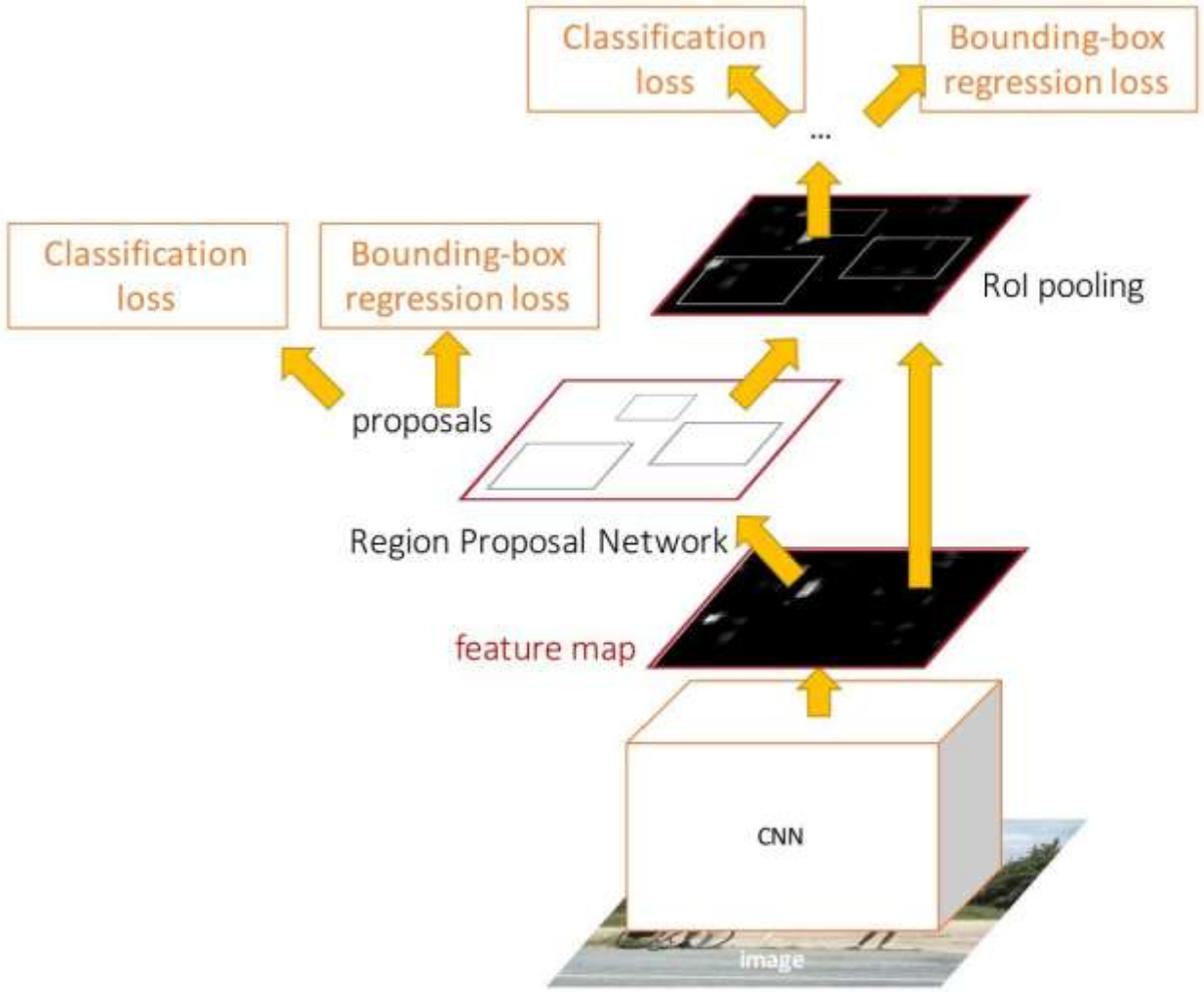


Faster R-CNN:

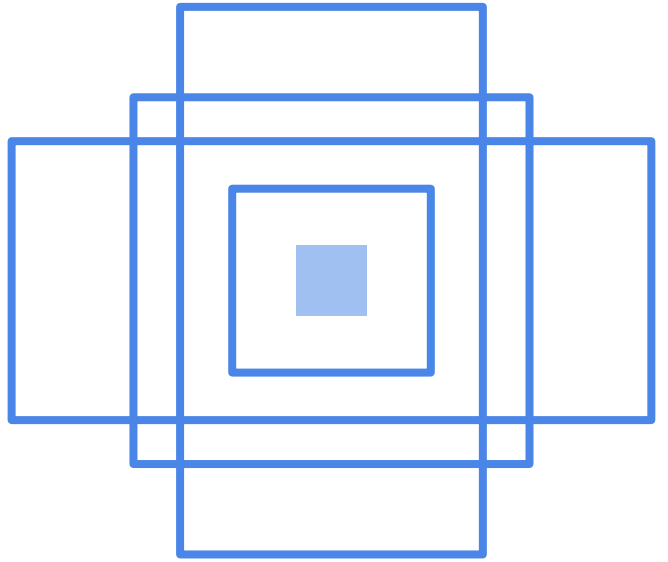
让CNN提Proposals！

大量的代码细节:

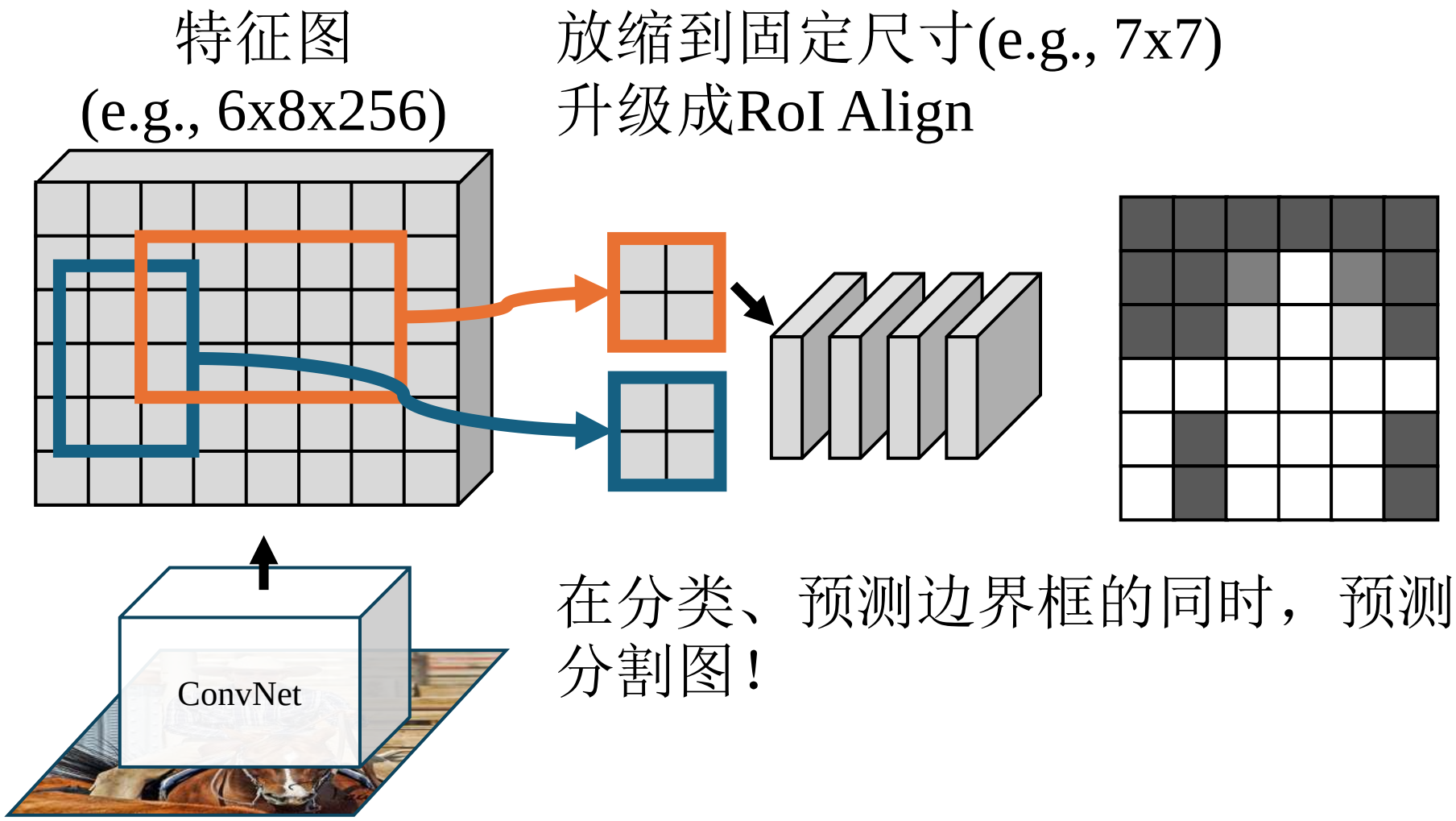
- 去掉重叠的proposal：**non-max suppression**
- 多少个anchor?
- 采样多少个 物体/没有物体 的proposal来训练?
- 怎么做物体框的回归?



回顾：Selective Search 与 NMS

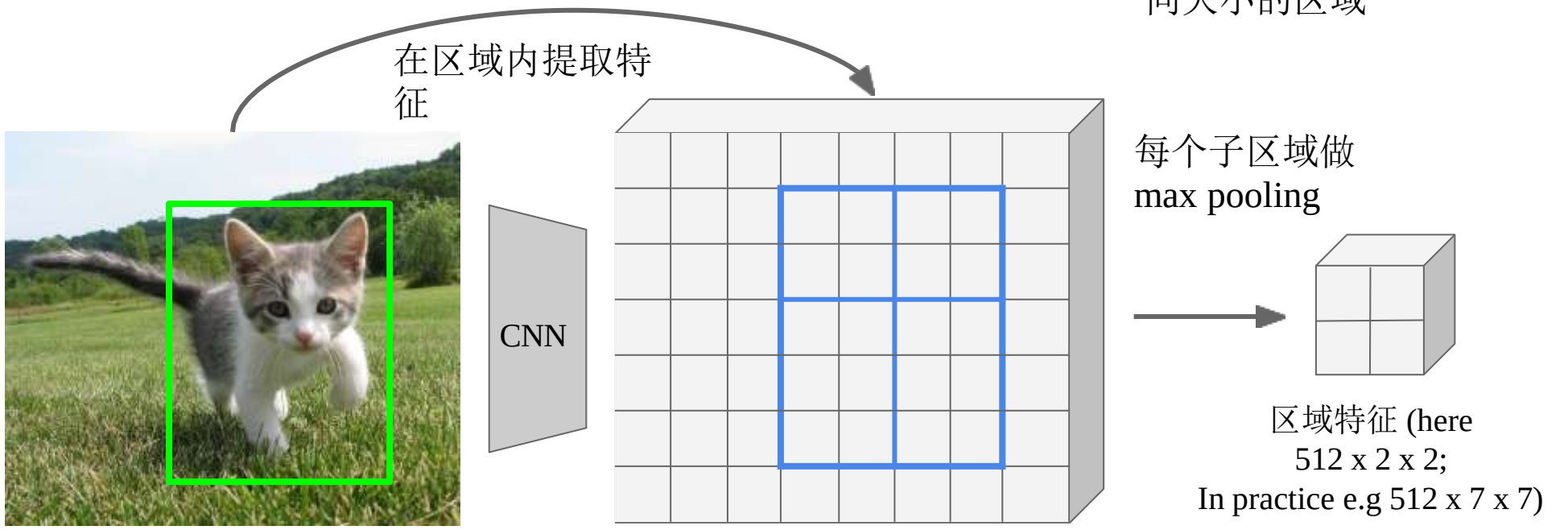


Mask RCNN



Mask RCNN: He, Gkioxari, Dollar, Girshick . ICCV 2017.

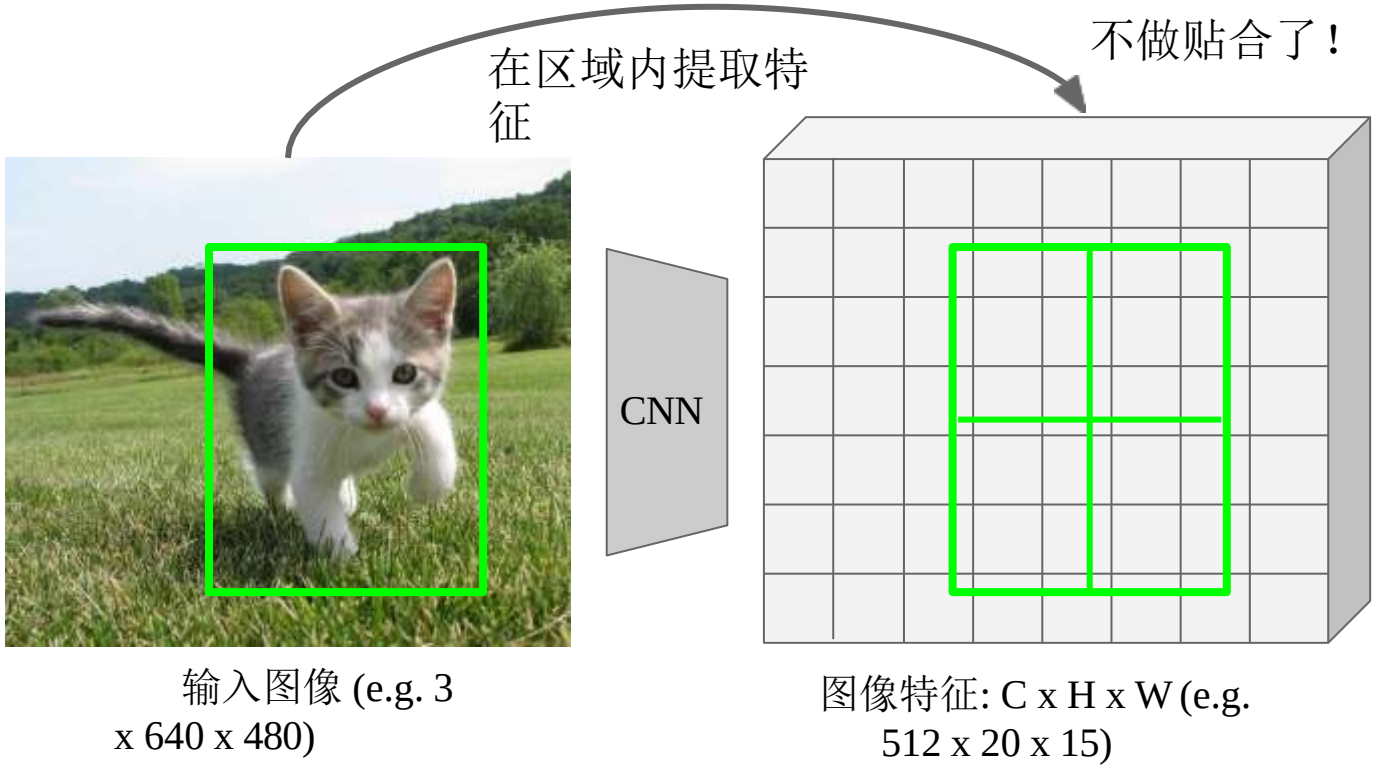
Fast R-CNN裁剪 特征: RoI Pool



问题: 每个区域的特征格点数量上不均匀

Girshick, “Fast R-CNN”, ICCV 2015.

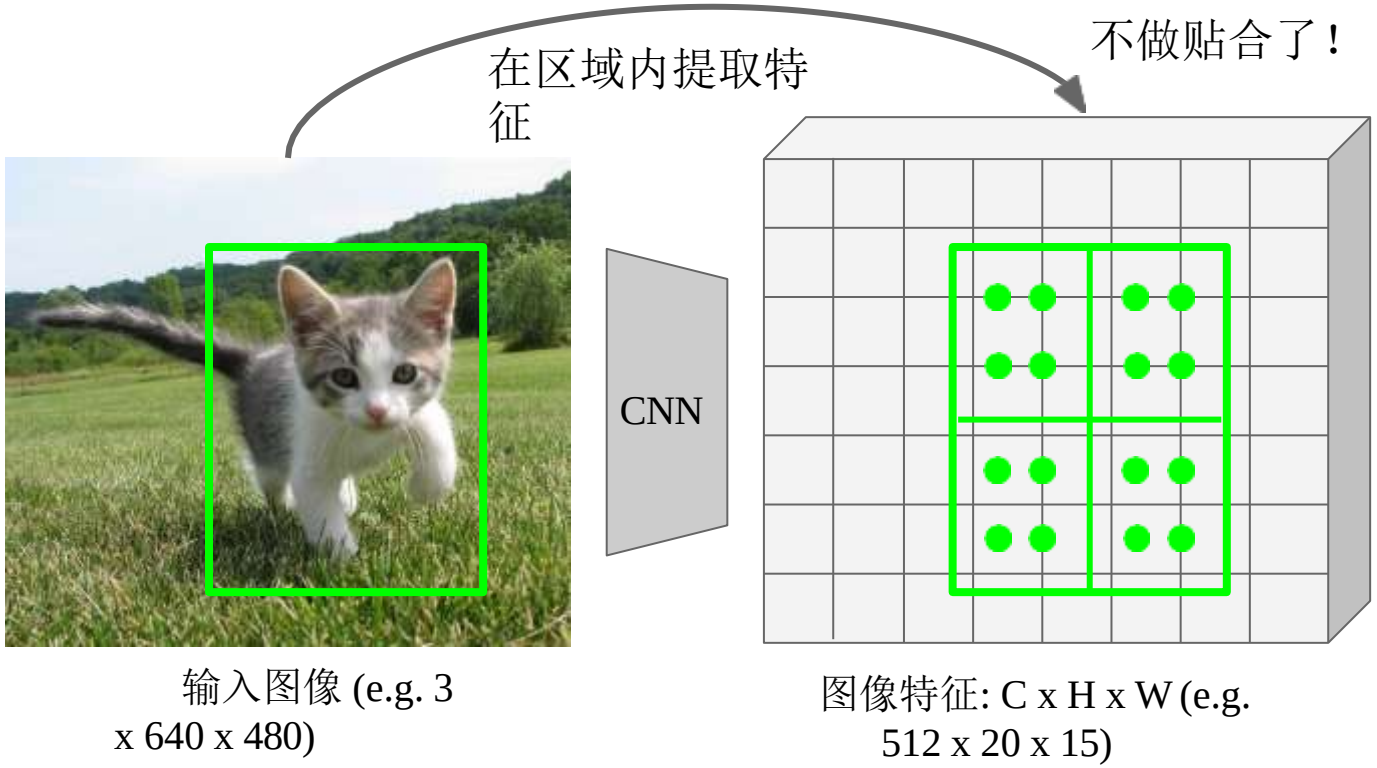
RoI Pool升级版: RoI Align



He et al, “Mask R-CNN”, ICCV 2017

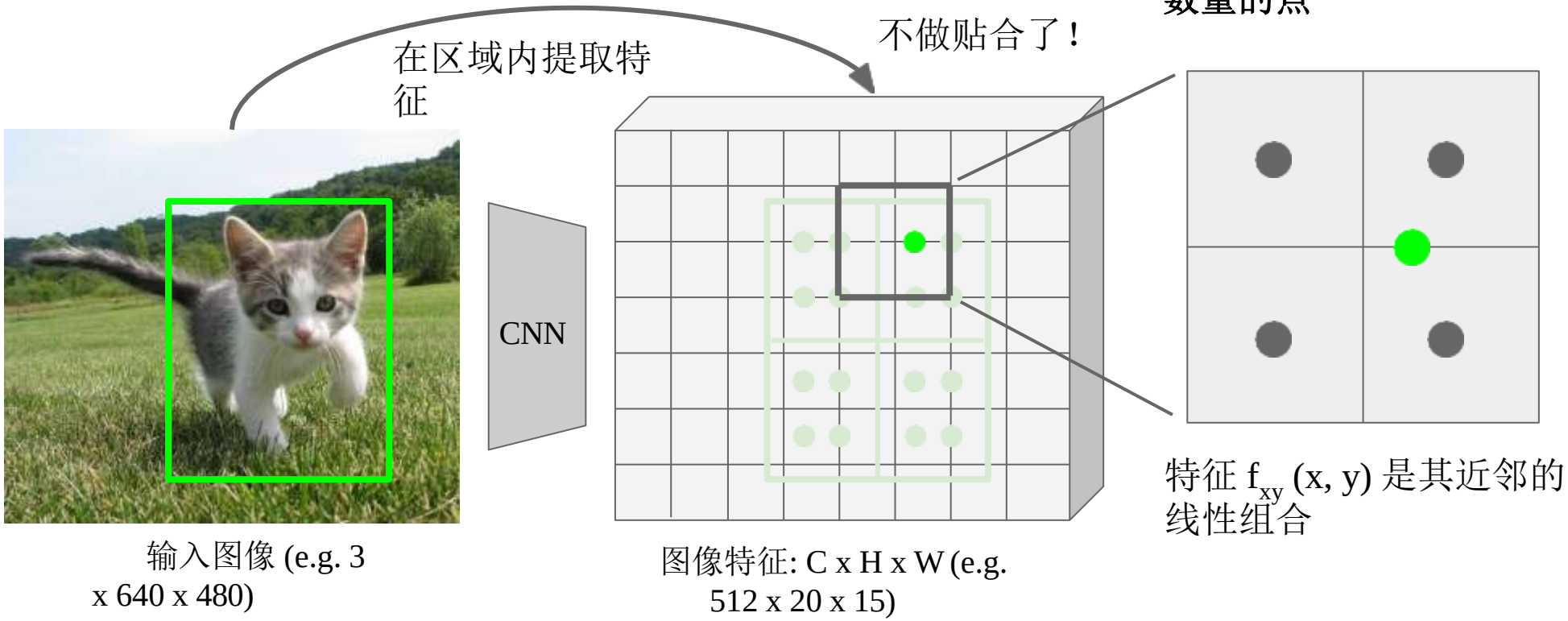
RoI Pool升级版: RoI Align

在每个子区域采样固定数量的点



He et al, “Mask R-CNN”, ICCV 2017

RoI Pool升级版: RoI Align



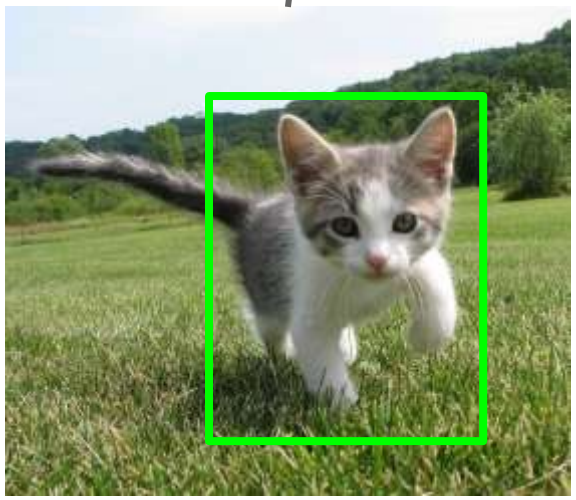
He et al, “Mask R-CNN”, ICCV 2017

RoI Pool升级版: RoI Align

在每个子区域采样固定数量的点

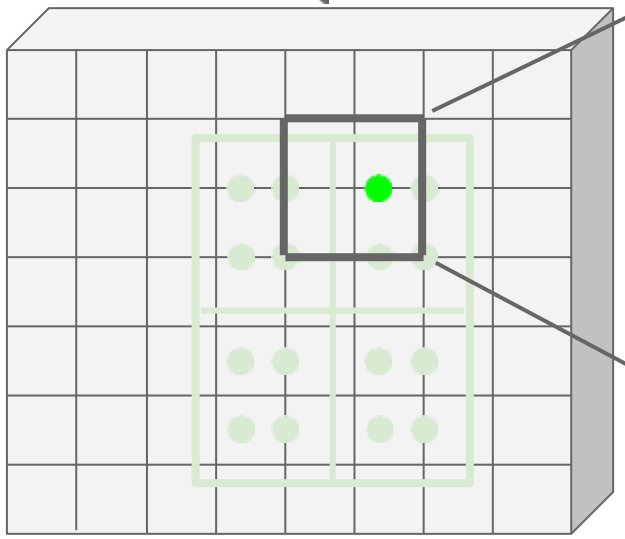
不做贴合了！

在区域内提取特征

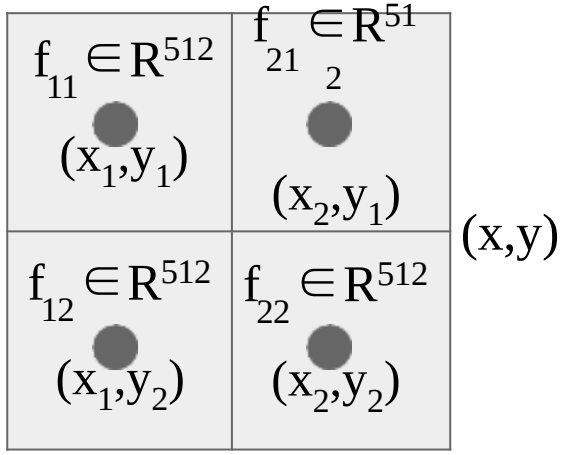


输入图像 (e.g. 3 x 640 x 480)

CNN



图像特征: C x H x W (e.g. 512 x 20 x 15)



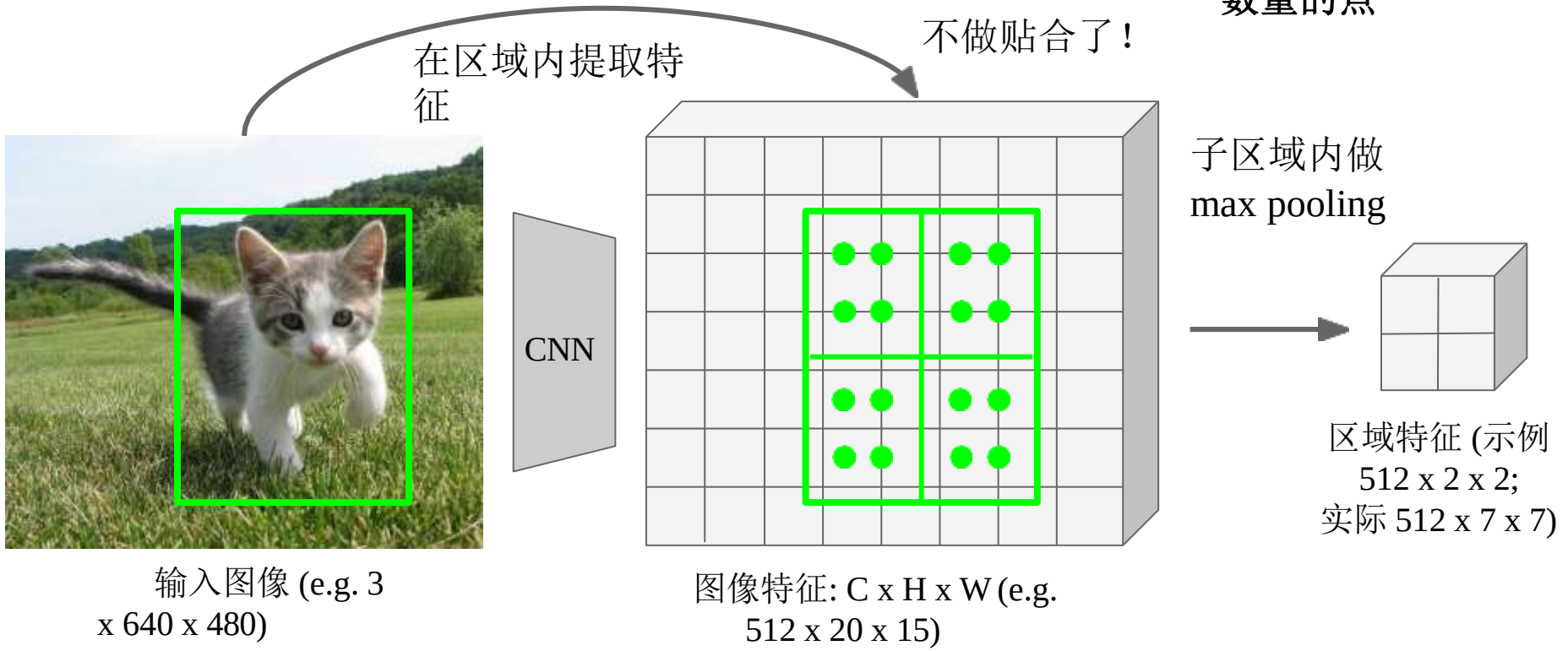
特征 f_{xy} (x, y) 是其近邻的线性组合

$$f_{xy} = \sum_{i,j=1}^{2} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

He et al, “Mask R-CNN”, ICCV 2017

RoI Pool升级版: RoI Align

在每个子区域采样固定数量的点



He et al, “Mask R-CNN”, ICCV 2017

MaskRCNN – 效果


traffic light.99
traffic light1.00
car.95
car.81
car.89
car.91
car.97
car.96
car.94
car.97
car.96
car.99
car.97
car.86
car1.00
parking meter


donut.86
donut.95
donut.89
donut.90
donut.89
donut.89
donut.93
donut.99
donut.96
donut.86
donut.95
donut.98
donut.89
donut1.00
donut.96
donut.98
donut.98
donut.99
donut.98
donut.89
donut.97
donut.95
donut.94
donut.98
donut1.00
donut.95
donut.99
donut.96
donut.90
donut.86
donut1.00

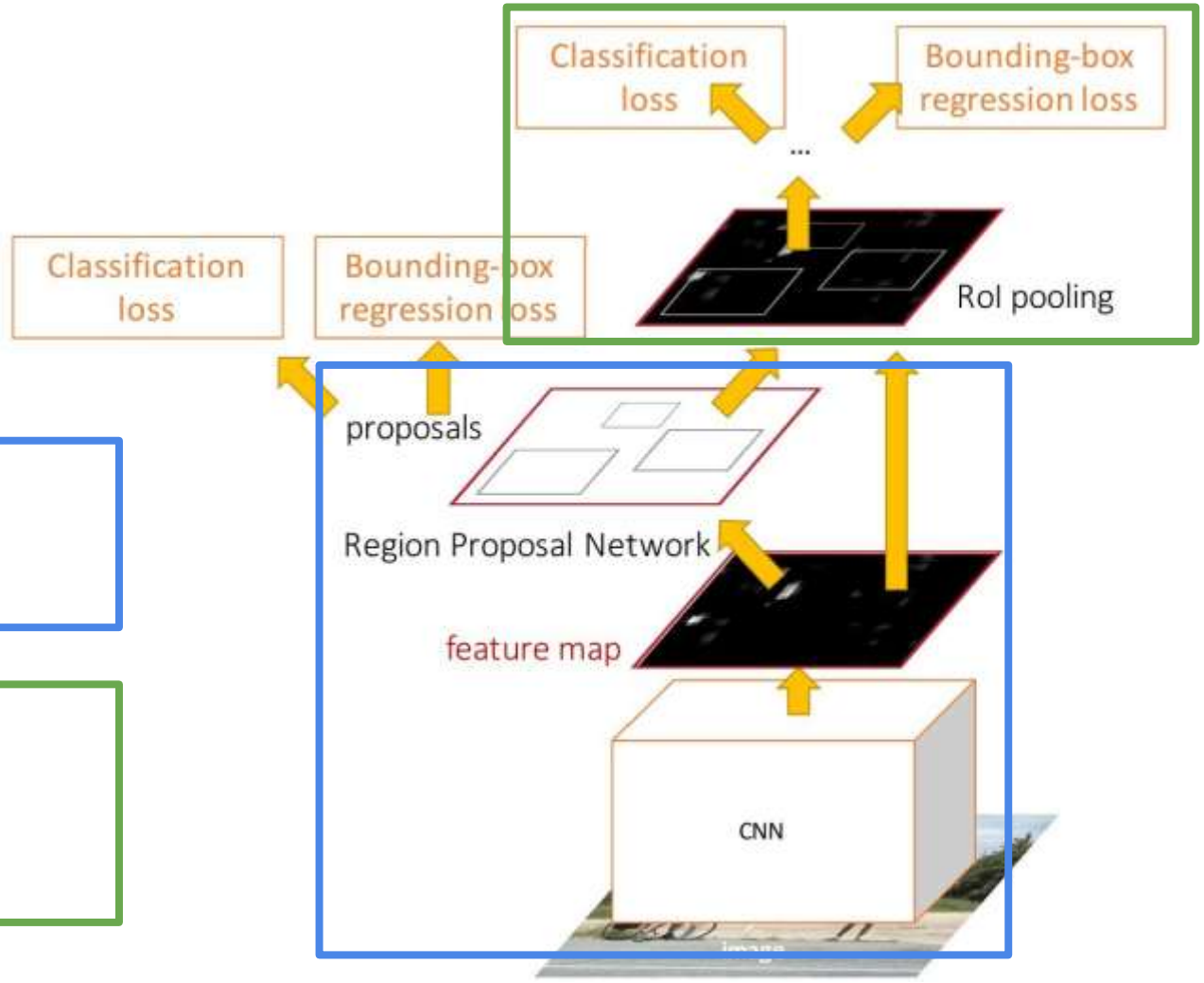
Faster R-CNN:
让CNN提Proposals！

Faster R-CNN 是一个
两阶段物体检测器

第一个阶段: 在每张图上
- Backbone network
- Region proposal network

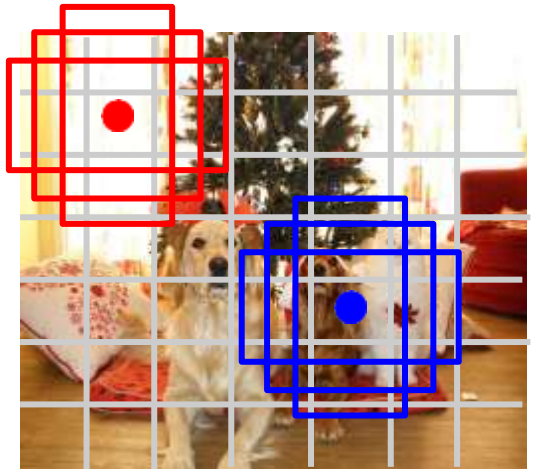
第二个阶段: 在每个区域上
- 裁剪特征: RoI pool / align
- 预测物体类别
- 预测框回归

我们真的需要第二个阶段吗？



单阶段物体检测方法: YOLO / SSD / RetinaNet

输入图像 3
x H x W



把图像分成7 x 7的格点

每张图会包含一些以格点为中心的**基础框**

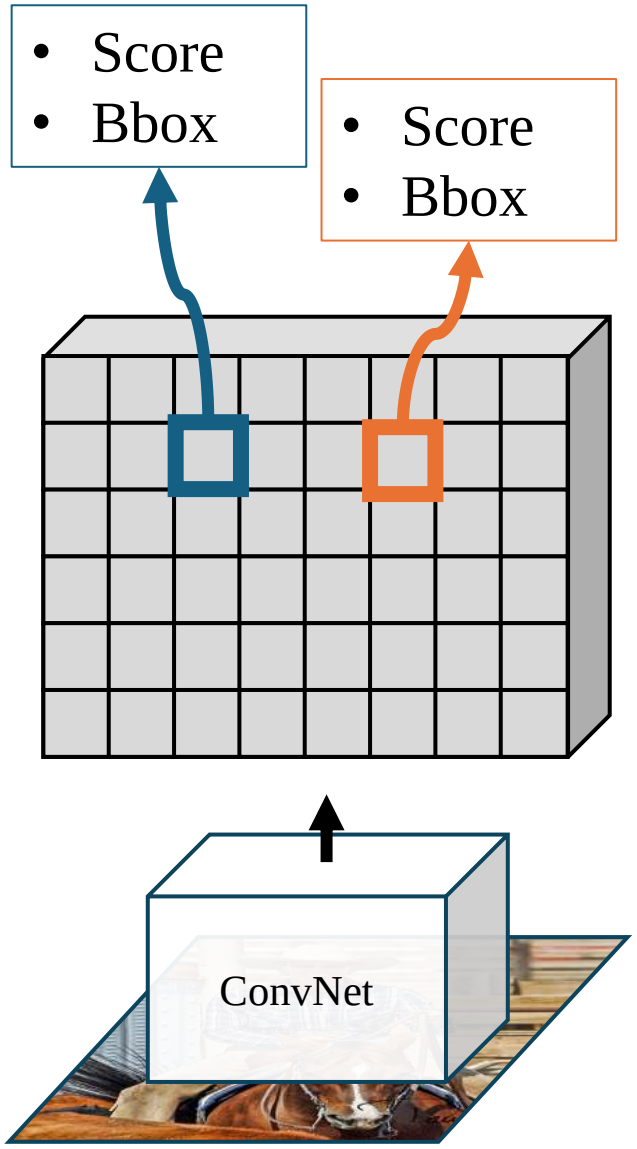
Here B = 3

对于每一个格点:

- 从B个 基础框（Anchor）来回归出最终的物体框:
  (dx, dy, dh, dw, confidence)
- 每个框对每个类都预测一个值 (包括背景)
- 类似于我们有7x7个RPN

Output:
7 x 7 x (5 * B + C)

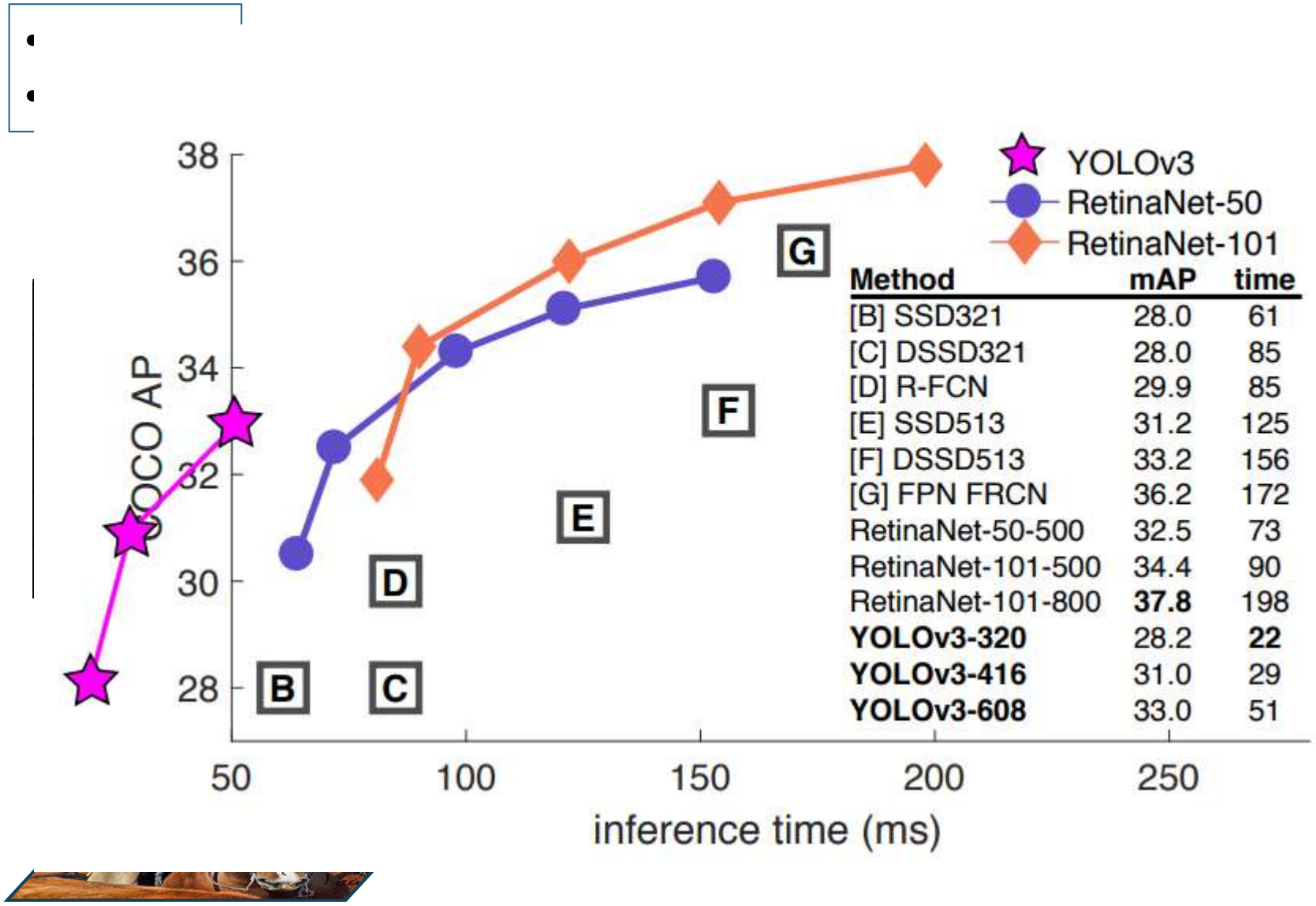
YOLO



- 没有区域提议
- 对7x7的特征图上的每个点，预测分类评分+边界框校正
- 比 Faster-RCNN 快7倍，但精确度较低
- 由于其速度较快，在实际应用中，如机器人等，较常使用
- 众多版本

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

YOLO

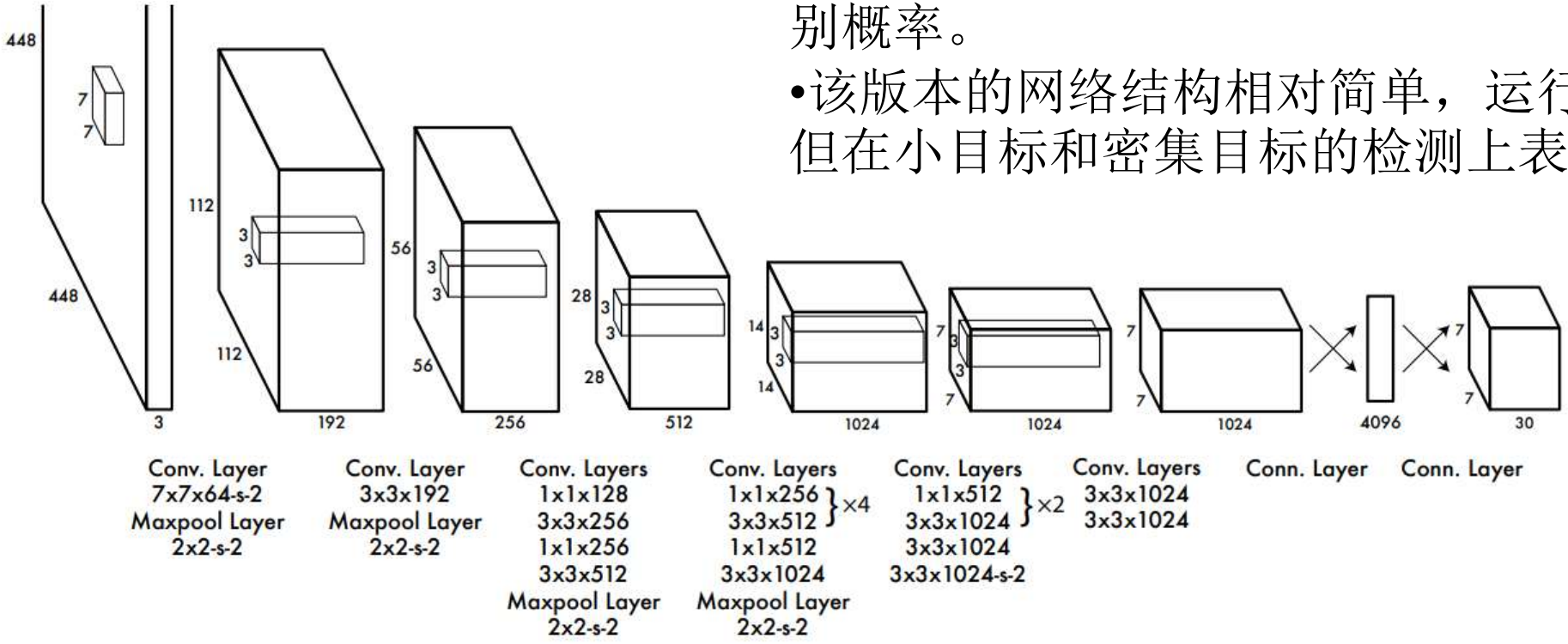


Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

J. Redmon, A. Farhadi, *YOLOv3: An Incremental Improvement* .

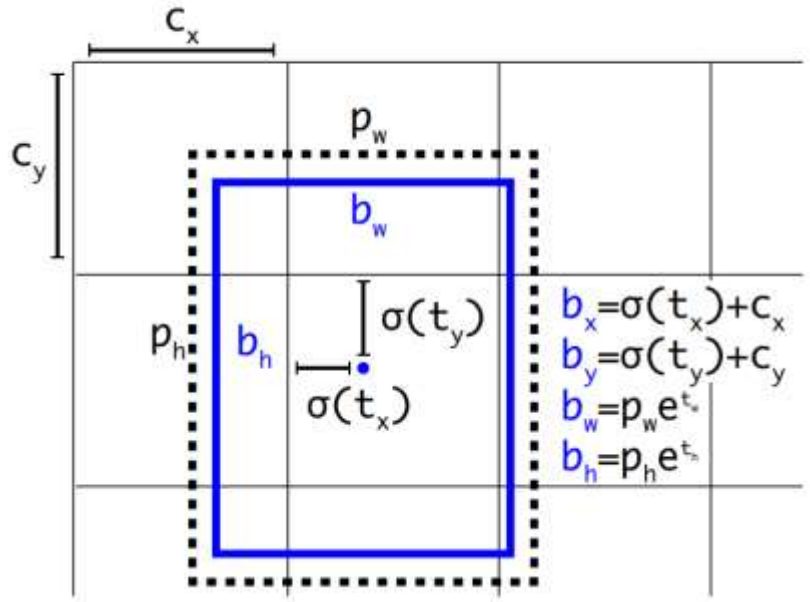
YOLO Zoo

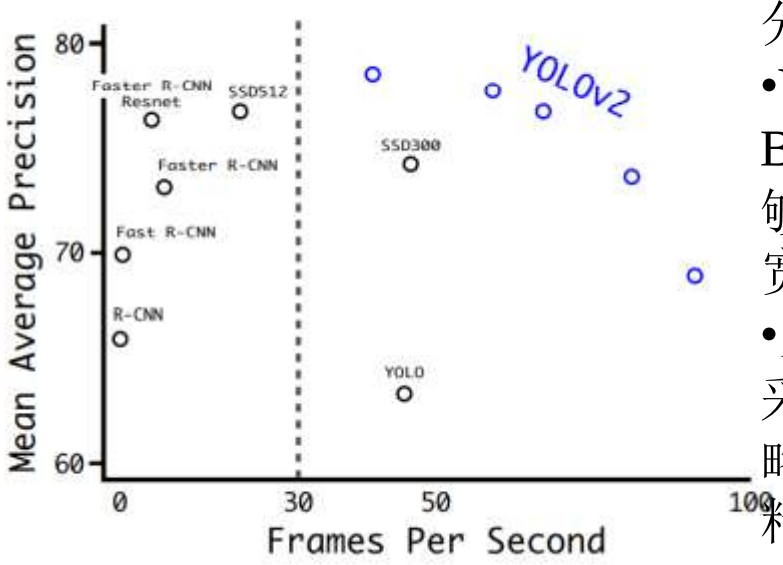
•发布于2015年，是YOLO系列的首个版本。
•YOLO v1采用单一卷积神经网络来进行实时目标检测，将检测问题视为回归问题，并在一次前向传递中直接预测边界框的坐标和类别概率。
•该版本的网络结构相对简单，运行速度很快，但在小目标和密集目标的检测上表现不佳



https://arxiv.org/pdf/1506.02640.pdf YOLO v1

YOLO Zoo

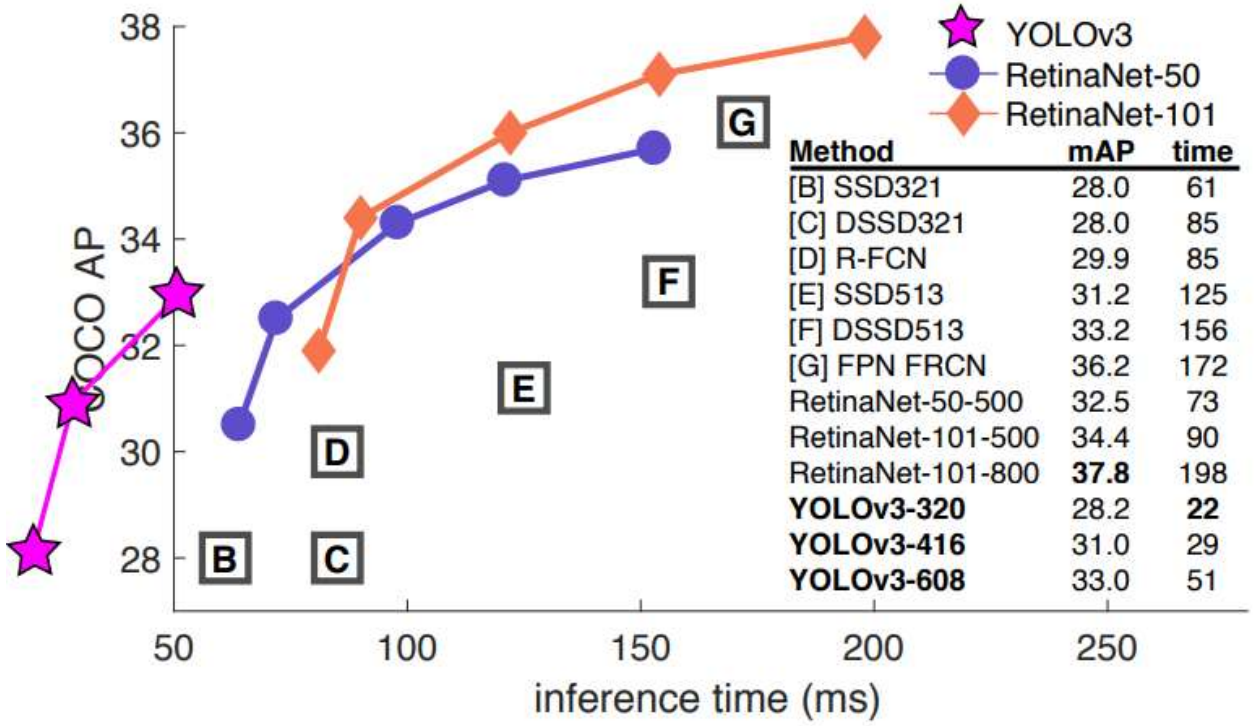




•发布于2016年，是对YOLO v1的改进和升级。
•引入了Darknet-19和Darknet-53两种网络结构，分别具有19和53层。
•YOLO v2采用了Anchor Boxes的概念，使得模型能够更好地预测不同尺寸和长宽比的目标。
•另外，YOLO v2在训练时采用了多尺度训练和预测策略，提高了对小目标的检测精度。

https://arxiv.org/pdf/1612.08242.pdf YOLO v2 Introduced anchor boxes to help predict different object sizes and aspect ratios.

YOLO Zoo

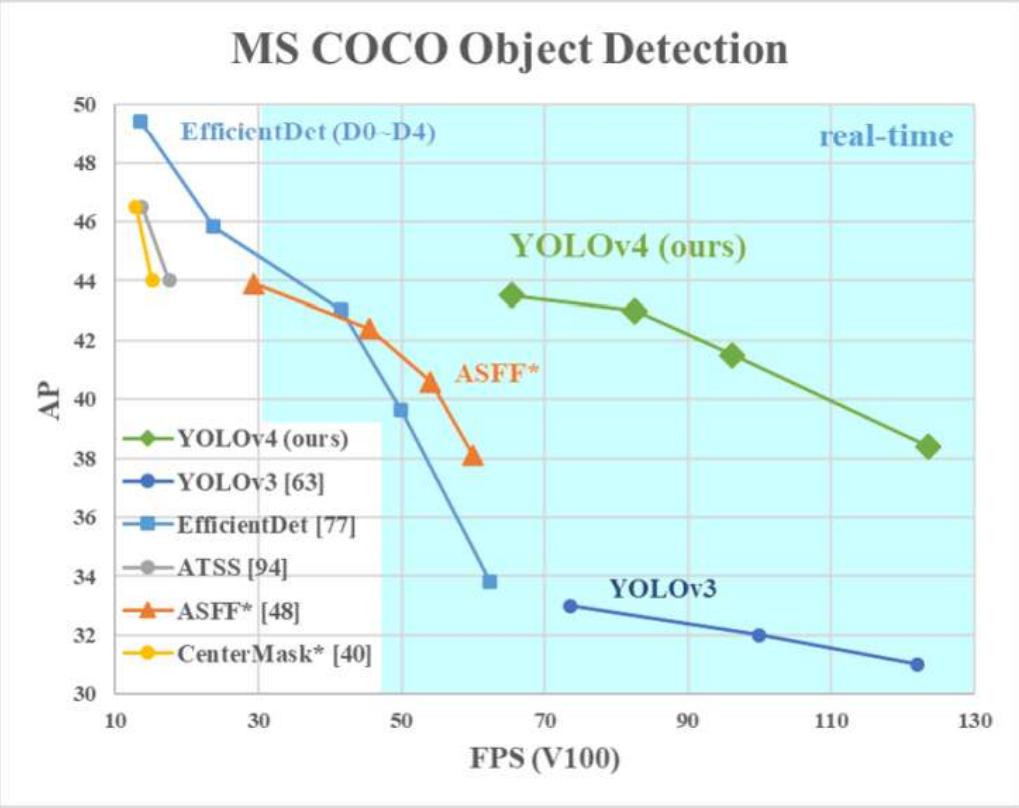


Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

•发布于2018年，是YOLO系列的第三个版本。
•YOLO v3进一步增加了网络的深度和复杂性，引入了更多的卷积层。
•使用了3种不同尺度的特征图来检测不同大小的目标，每个尺度都有自己的检测器和Anchor Boxes。
•YOLO v3采用了多尺度预测和Darknet-53网络结构，使得模型在准确性和速度上都有较好的表现。

https://arxiv.org/pdf/1804.02767.pdf YOLO v3 was able to detect smaller objects more effectively and achieved higher accuracy compared to its predecessors.

YOLO Zoo



- 发布于2020年，是YOLO系列的最新版本。
- YOLO v4引入了一系列技术改进，包括CSPDarknet53、SAM（空间注意力）、PANet（多尺度融合）等，使得网络更加高效和准确。
- 采用更大的模型和更多的卷积层，进一步提升了检测精度。
- YOLO v4还引入了诸如YOLOv4-tiny和YOLOX等变体，以满足不同场景下的需求

https://arxiv.org/pdf/2004.10934.pdf YOLO v4 Implemented several advanced techniques such as PANet, SAM, and PAN, to enhance feature representation and object detection.

物体检测: 各种变种...

Backbone Network
VGG16
ResNet-101
Inception V2
Inception V3
Inception
ResNet
MobileNet

“Meta-Architecture”
Two-stage: Faster R-CNN
Single-stage: YOLO / SSD
Hybrid: R-FCN

Image Size
 # Region Proposals
…

总结
Faster R-CNN 慢，但是准很多
SSD/YOLO快，但是不准
更深的网络总是表现得更好

实例分割

分类

CAT

无空间限定

语义分割

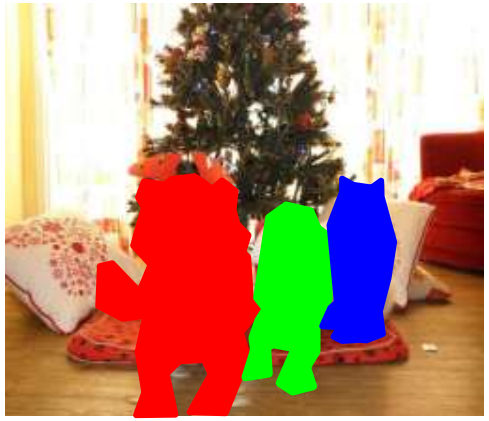
GRASS, CAT, TREE, SKY

像素级预测

物体检测

DOG, DOG, CAT

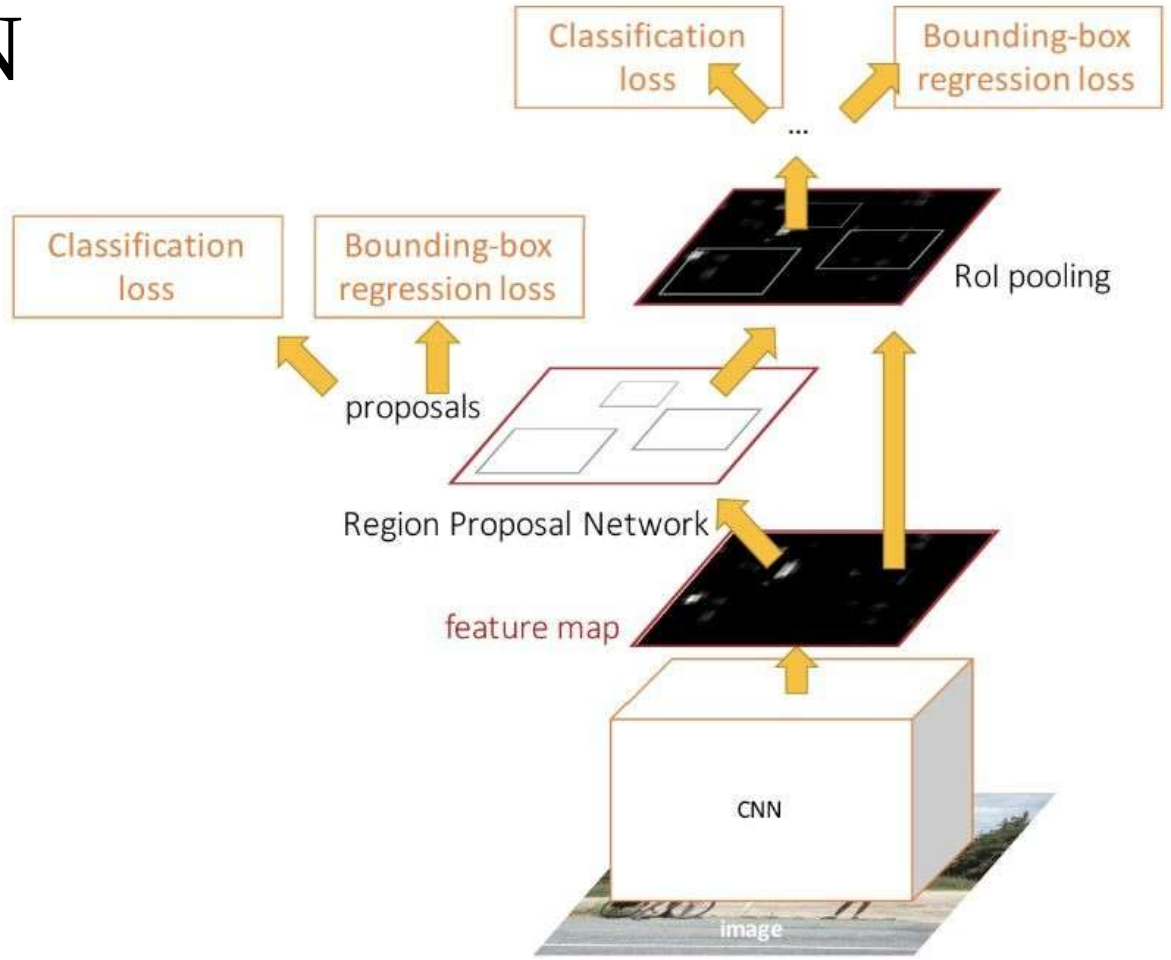
实例分割



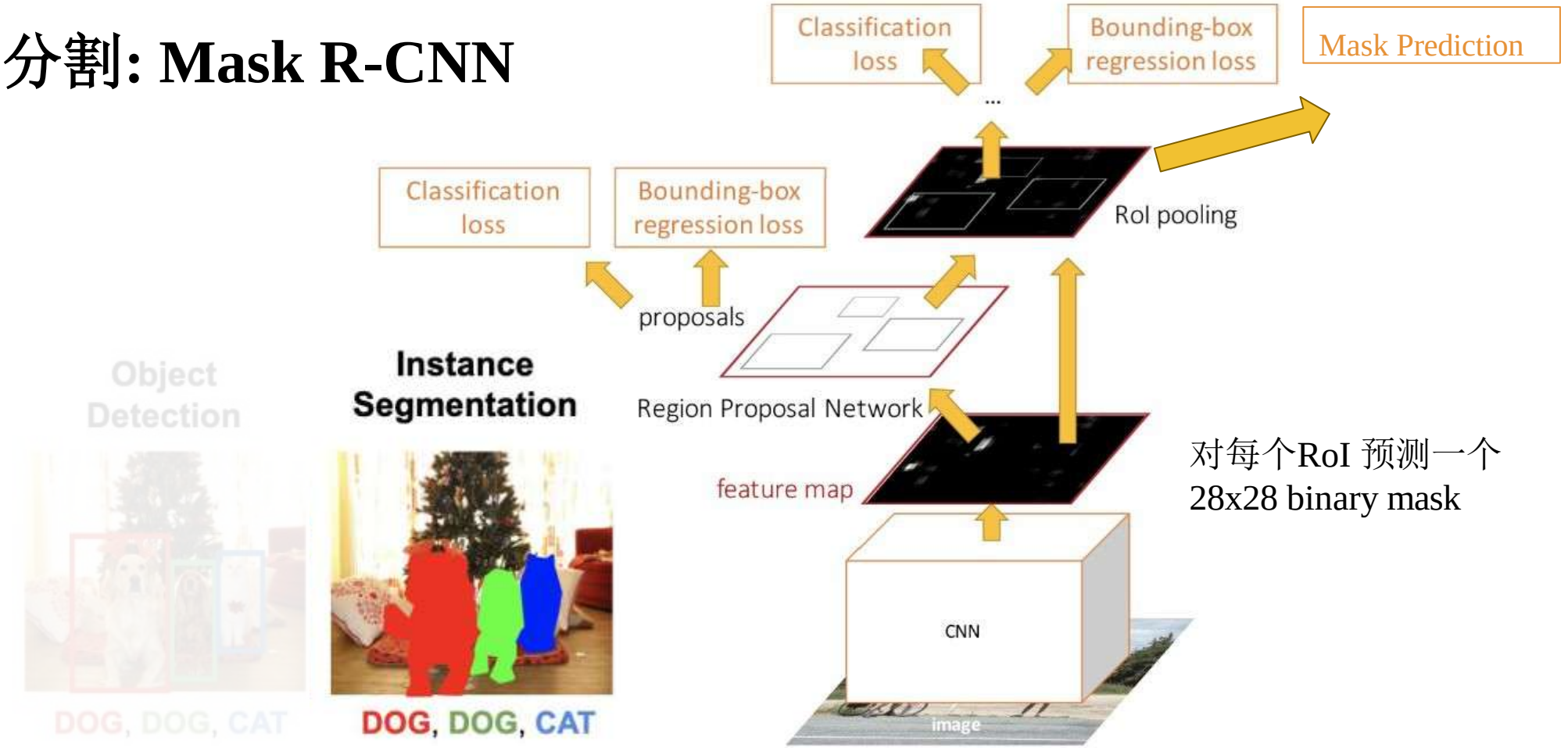
DOG, DOG, CAT

多物体

物体检测: Faster R-CNN



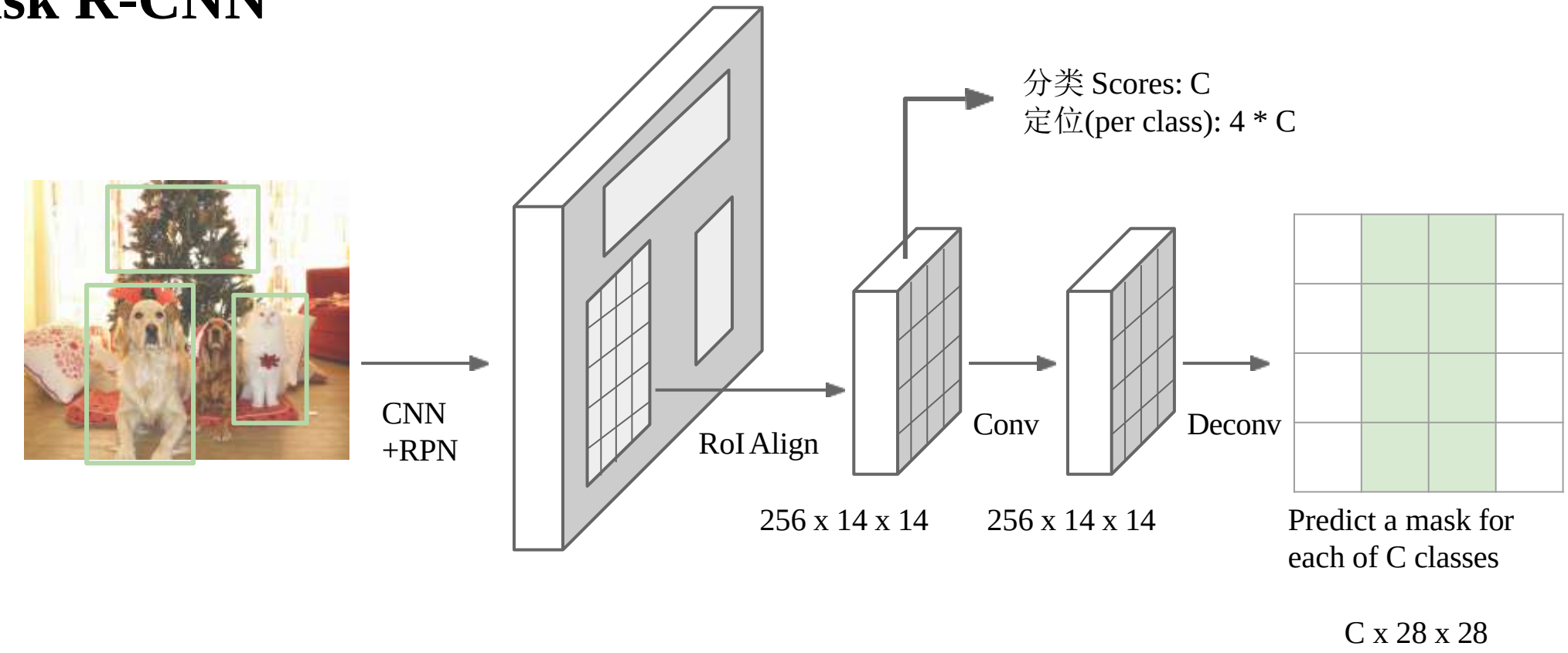
实例分割: Mask R-CNN



对每个RoI 预测一个28x28 binary mask

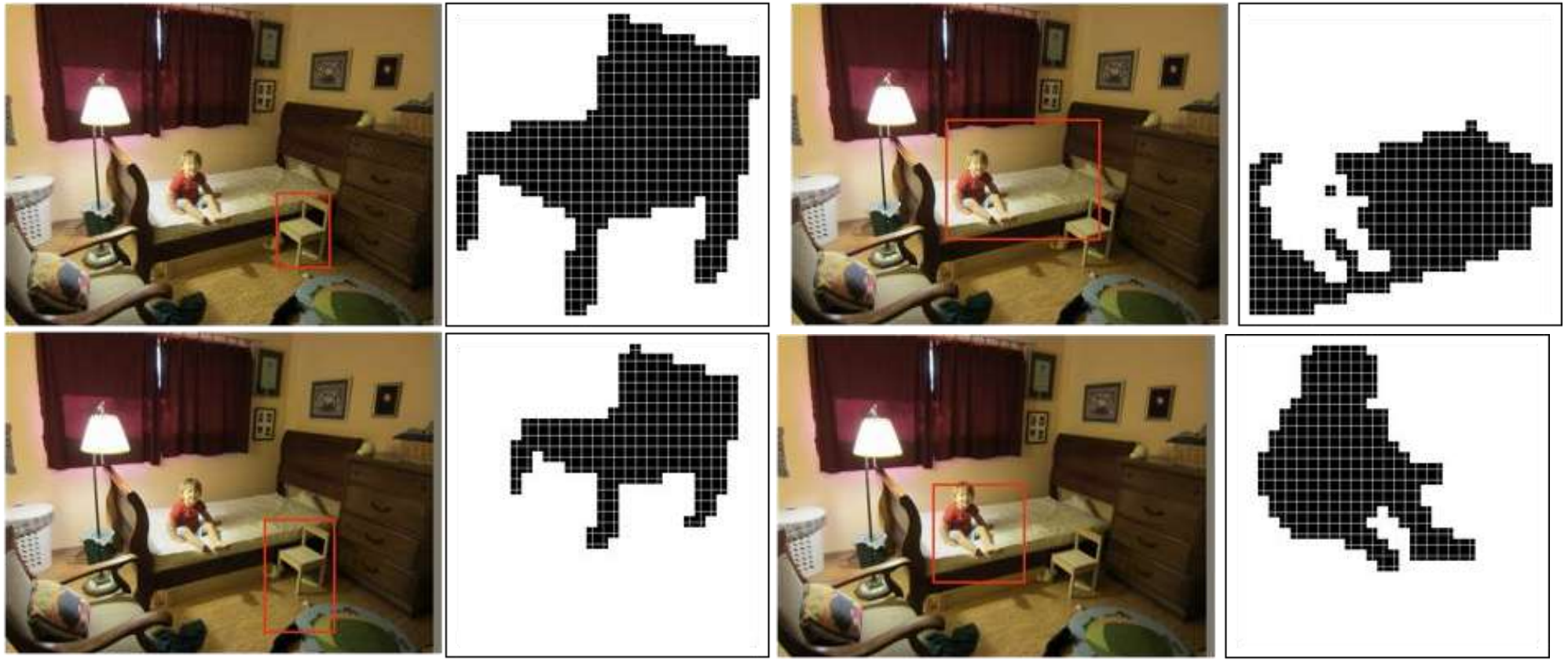
He et al, “Mask R-CNN”, ICCV 2017

Mask R-CNN

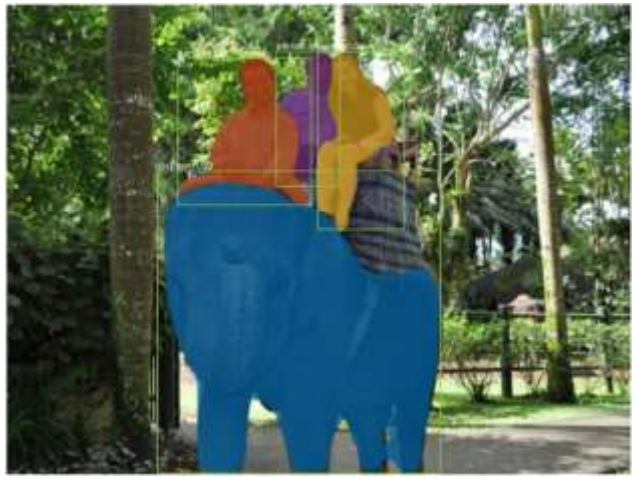
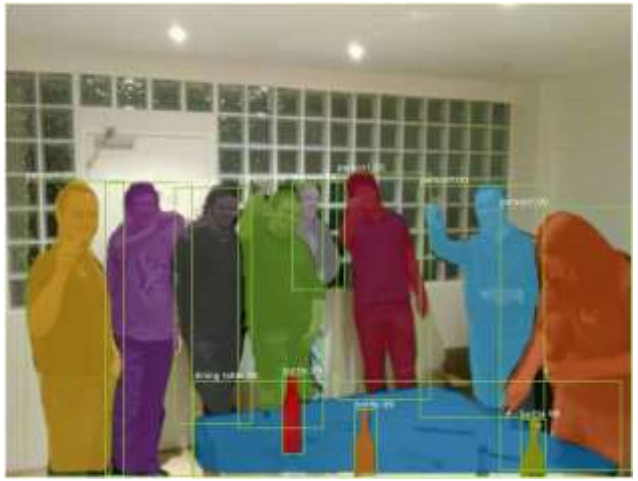


He et al, “Mask R-CNN”, arXiv 2017

Mask R-CNN: 例子



Mask R-CNN

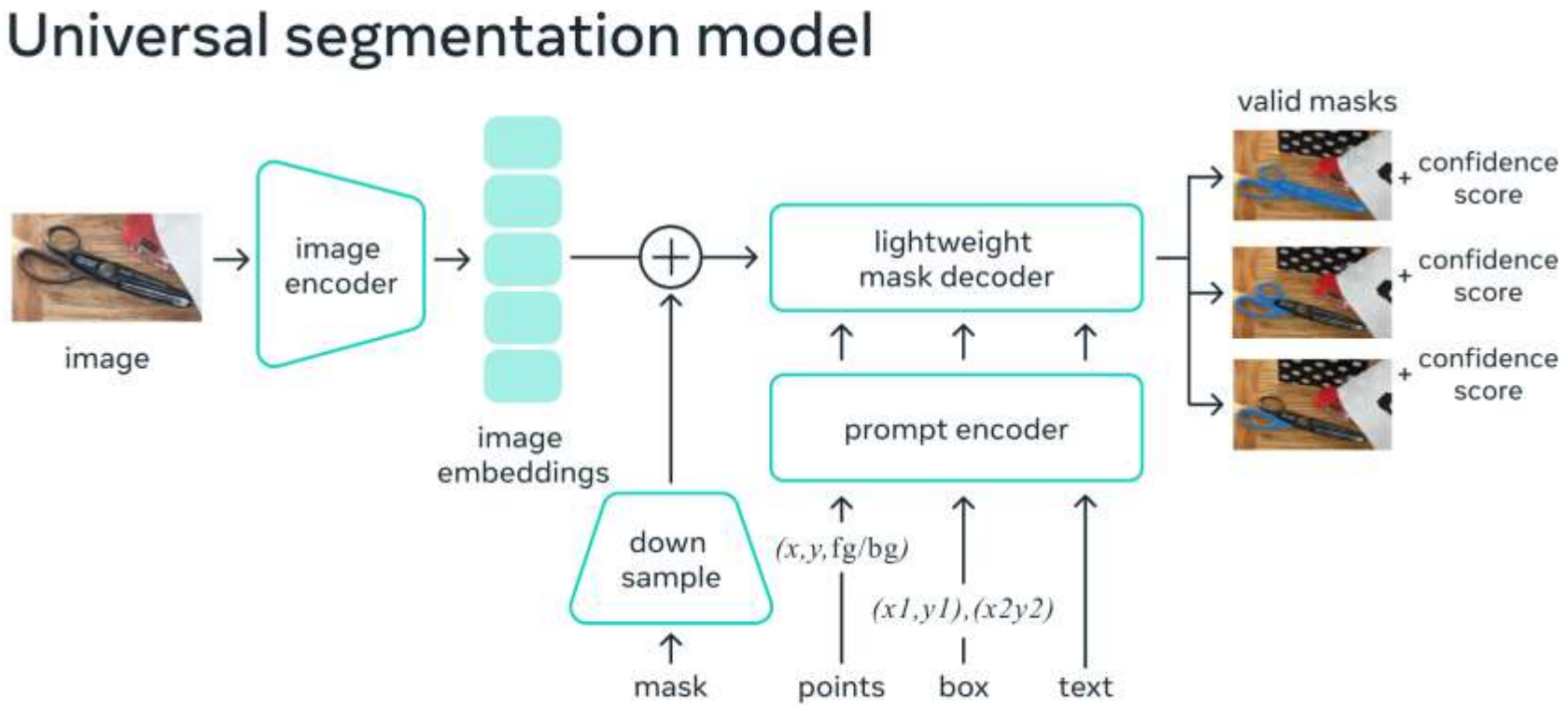
He et al, “Mask R-CNN”, ICCV 2017

Mask R-CNN 也可以做姿态估计

He et al, “Mask R-CNN”, ICCV 2017

Segment Anything

针对任何提示返回有效的分割掩码



Segment Anything

点击作为触发词

Segment Anything

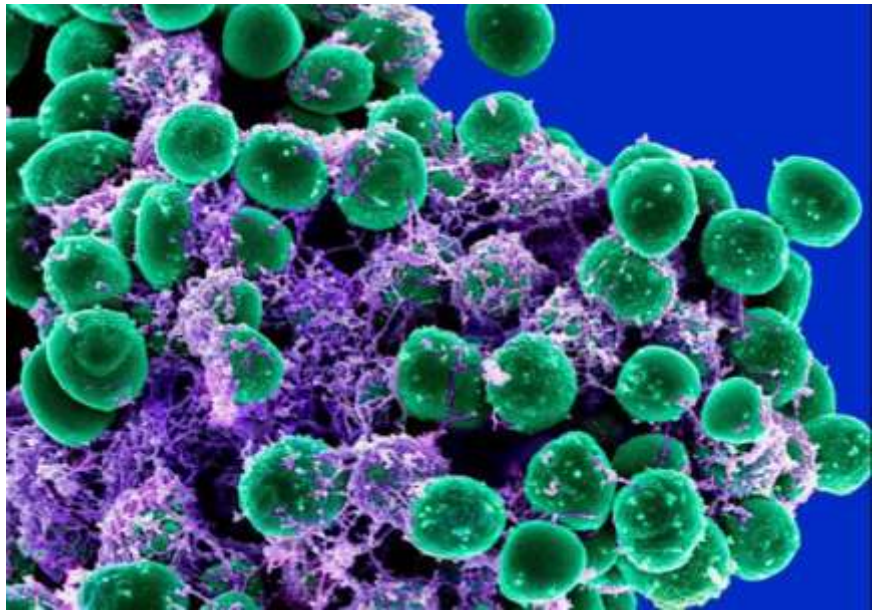
标签作为触发词

Segment Anything

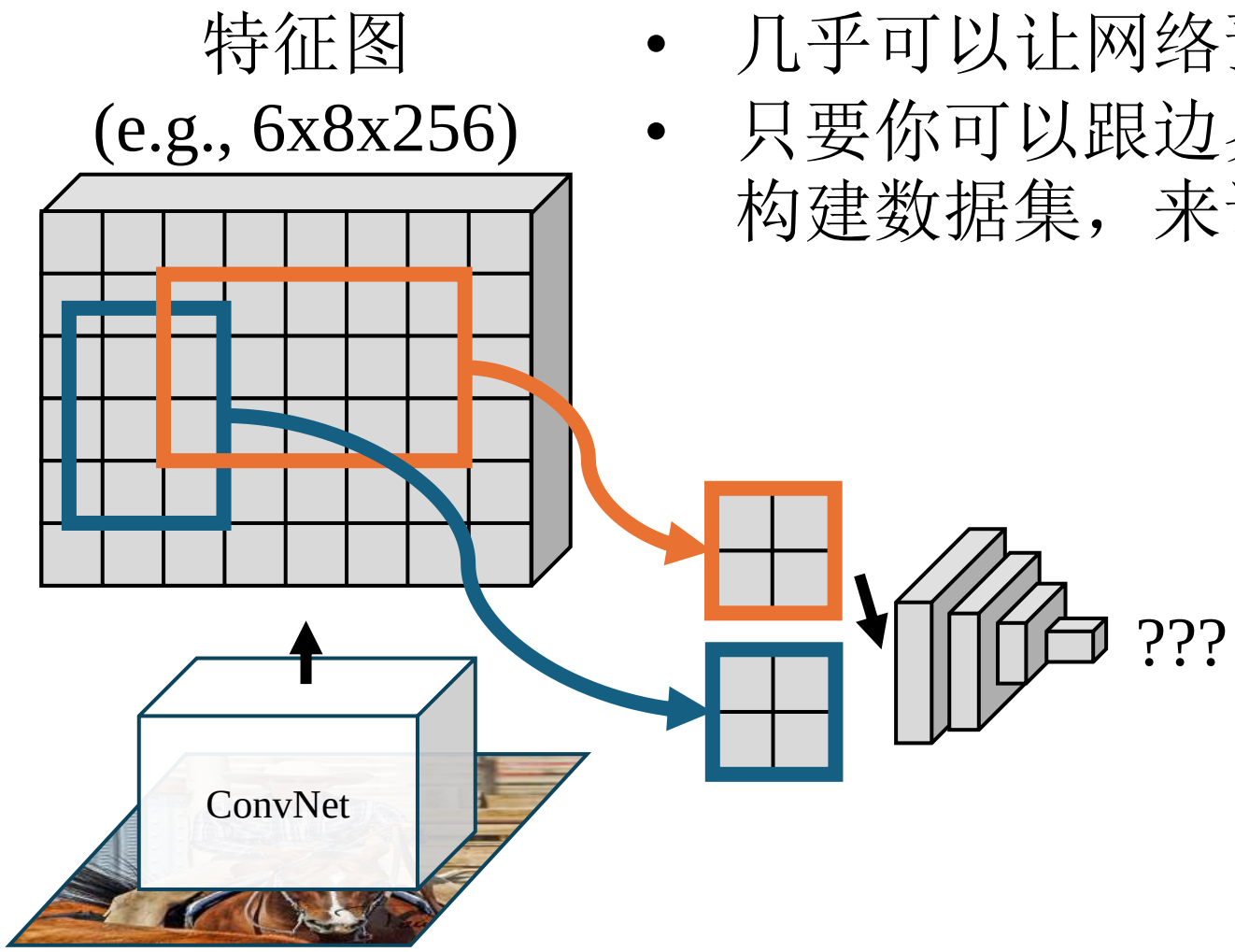
“valid mask” 不同触发词对应的合理掩码

Segment Anything

零知识“zero-shot”预测



物体检测的拓展

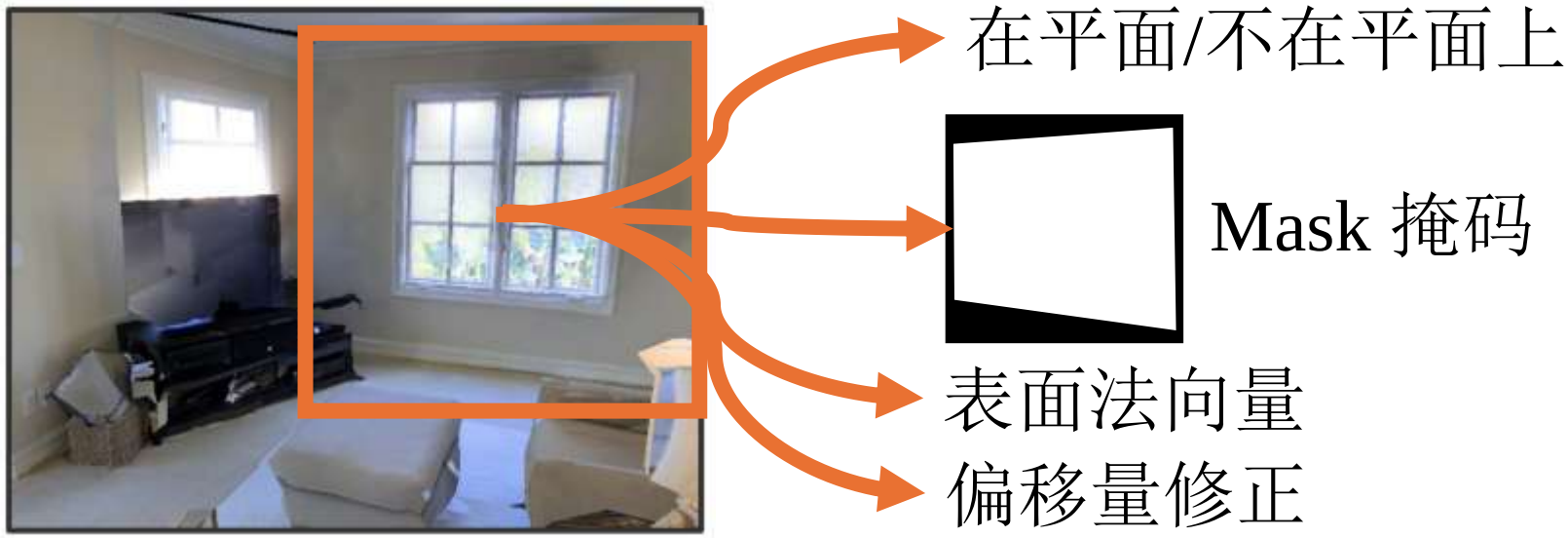


- 几乎可以让网络预测任何东西.
- 只要你可以跟边界框结合起来，构建数据集，来训练网络

Liu et al. *PlaneRCNN: 3D Plane Detection and Reconstruction from a Single Image*. CVPR 2019.

平面检测

示例: RGB 图像输入，预测平面

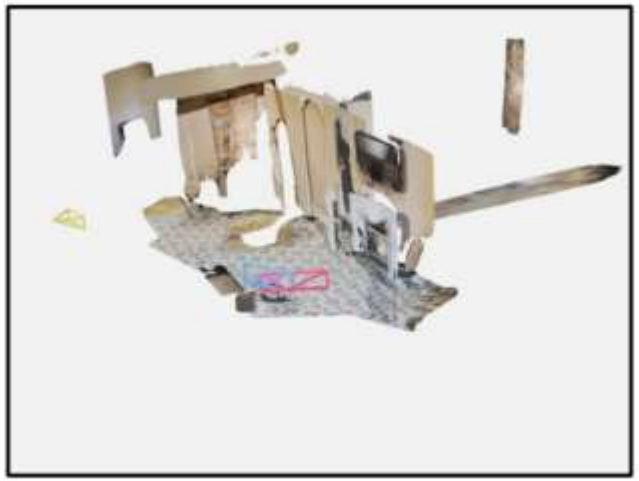


Liu et al. *PlaneRCNN: 3D Plane Detection and Reconstruction from a Single Image*. CVPR 2019.

平面检测

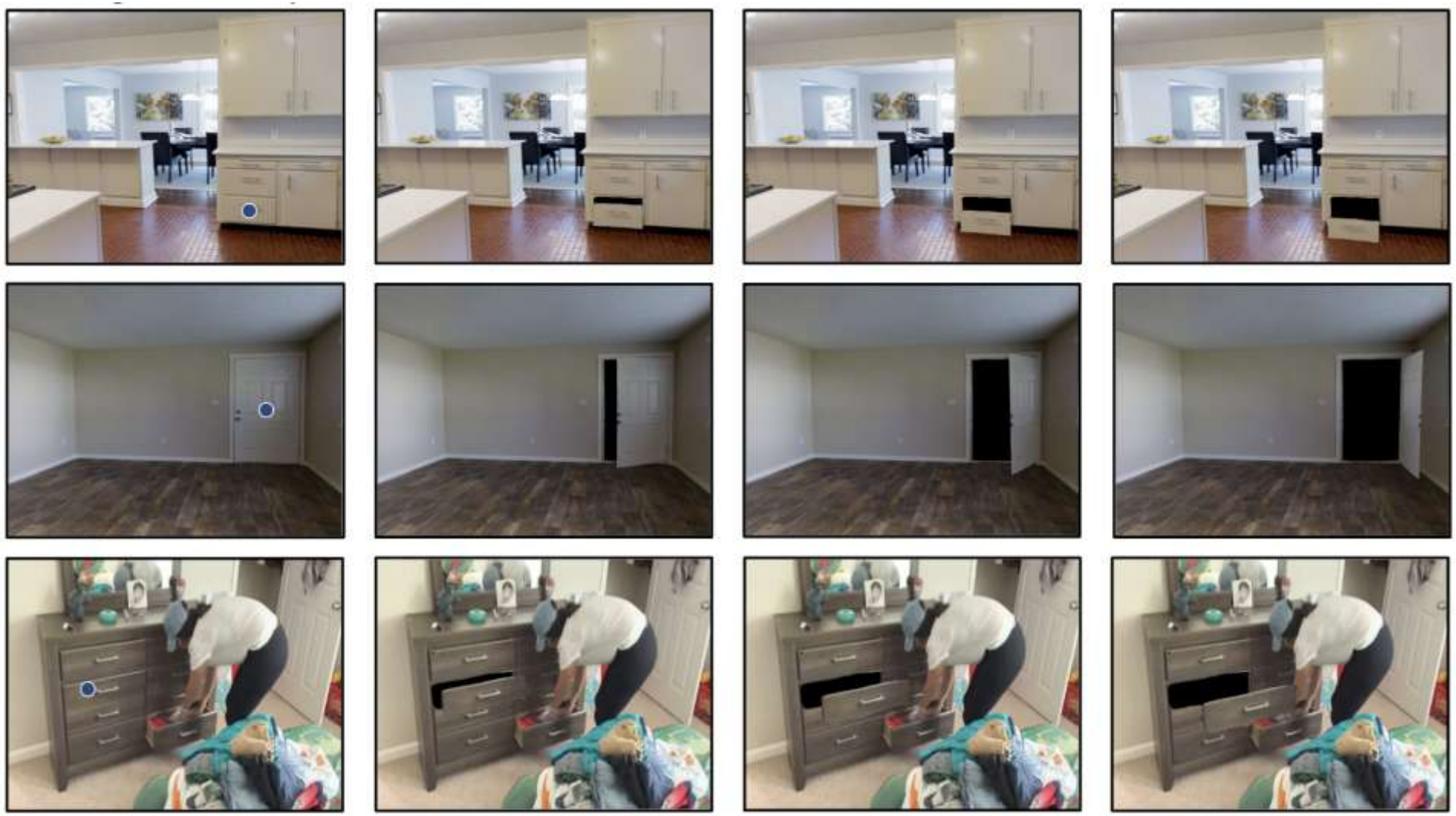


功能：构建3D场景




S. Agarwala, L. Jin, C. Rockwell, D.F. Fouhey. PlaneFormers: From Sparse View Planes to 3D Reconstruction.???, 2022

平面检测



第十周周三（5.6）课程大作业开题汇报

- 每组汇报总时长为 **5–8 分钟**，包含展示、提问与切换时间；**不因组员人数增加而额外延长**。
- 每组汇报结束后，由**助教进行简短提问**；同学们也可就大作业中的问题进行交流。
- 在本次汇报前，各组需**基本确定选题、数据集和组内分工**。
- 选择**自主选题**的组，需**提前与老师或助教沟通确认**，并在本次汇报中说明选题内容、研究目标、数据来源和实施计划。
- 本次汇报为**阶段性汇报**，重点考察**选题思路、方案设计与可行性，不要求已经完成全部实验结果**。