视觉目标分类检测分割及PyTorch实现

彭小江 副教授

深圳技术大学 大数据与互联网学院

邮箱: pengxiaojiang@sztu.edu.cn
个人主页: pengxj.github.io

计算机视觉基础任务

- 分类
- 检测
- 分割、语义分割
- 其他 (tracking in videos, camera pose estimation, body pose estimation,

3d reconstruction, denoising, super-resolution, auto-captioning, synthesis, etc.)

分类任务

- 解决是什么问题

"Small scale" classification data-sets.

MNIST and Fashion-MNIST: 10 classes (digits or pieces of clothing) 50,000 train images, 10,000 test images, 28×28 grayscale.

(leCun et al., 1998; Xiao et al., 2017)

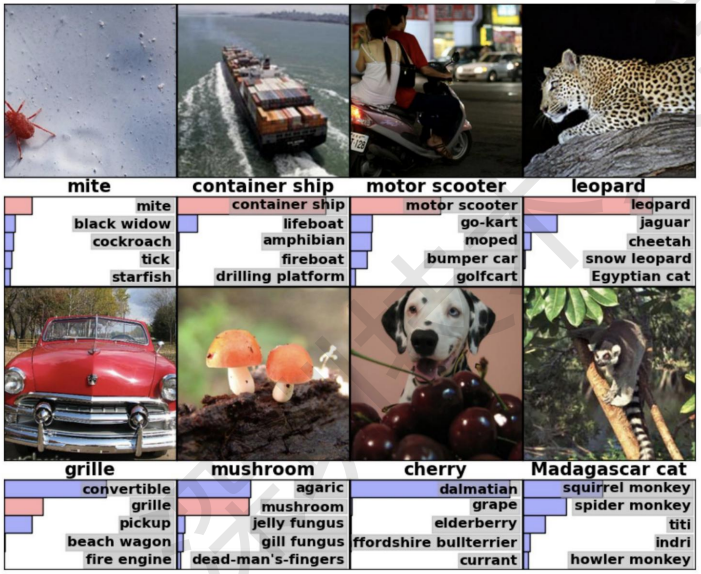
CIFAR10 and CIFAR100 (10 classes and 5×20 "super classes"), 50,000 train images, 10,000 test images, 32×32 RGB

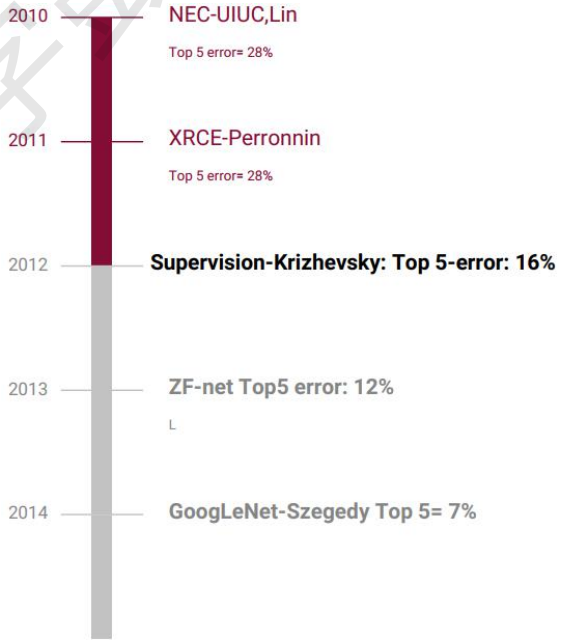
ImageNet（大规模视觉目标分类）

- ImageNet is an image database organized according to the WordNet hierarchy
- About 15M Labeled High resolution Images
- Roughly 22K Categories
- Collected from the web,labeled by Amazon Mechanical Turk

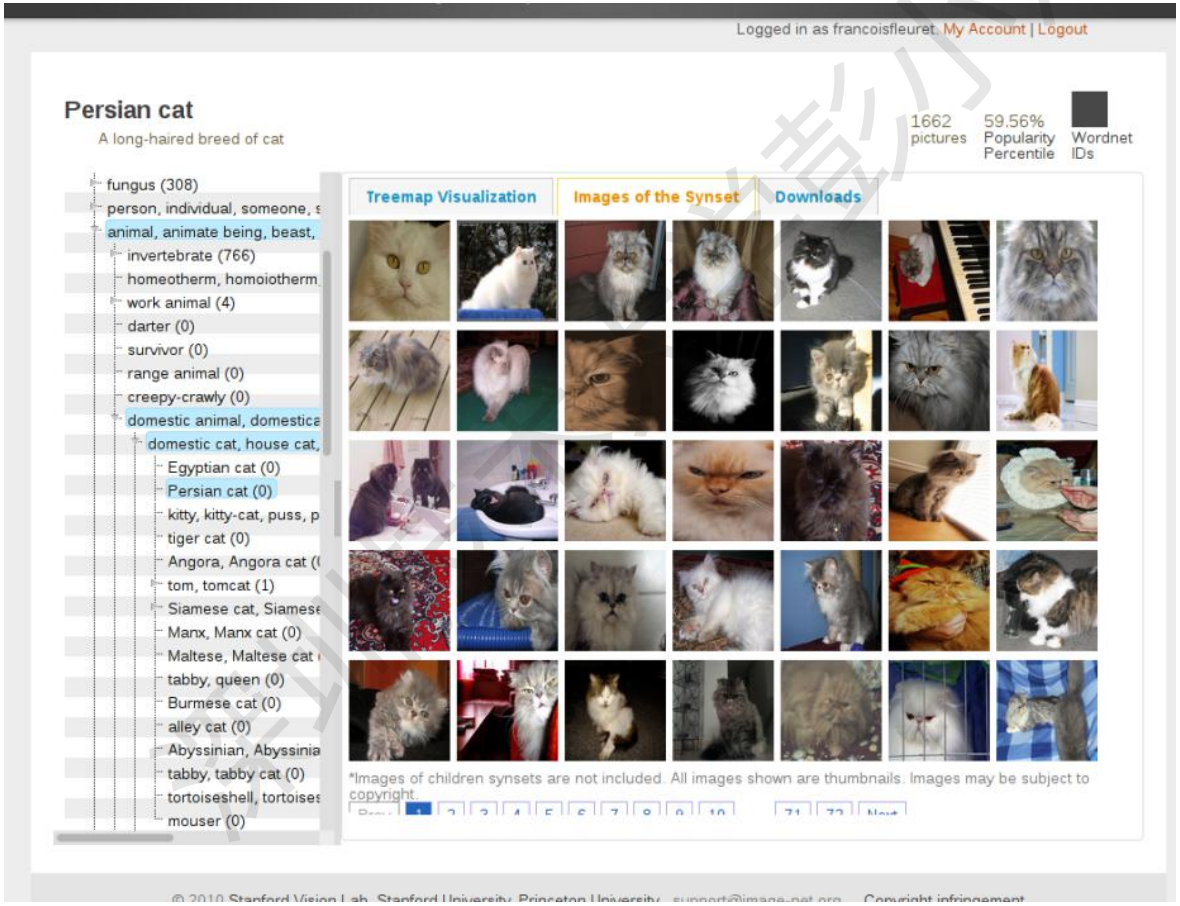
ILSVRC

- ImageNet Large Scale Visual Recognition Challenge
- Task: 1.2M train, 50K Validation, 100K test,1000 categories
- Goal: Top-5 error





ImageNet


Logged in as francoisfleuret. My Account | Logout
Persian cat
A long-haired breed of cat
1662 pictures
59.56% Popularity Percentile
Wordnet IDs
fungus (308)
person, individual, someone,
animal, animate being, beast,
invertebrate (766)
homeotherm, homoiotherm
work animal (4)
darter (0)
survivor (0)
range animal (0)
creepy-crawly (0)
domestic animal, domestica
domestic cat, house cat,
Egyptian cat (0)
Persian cat (0)
kitty, kitty-cat, puss, p
tiger cat (0)
Angora, Angora cat (
tom, tomcat (1)
Siamese cat, Siamese
Manx, Manx cat (0)
Maltese, Maltese cat
tabby, queen (0)
Burmese cat (0)
alley cat (0)
Abyssinian, Abyssinia
tabby, tabby cat (0)
tortoiseshell, tortoises
mouser (0)
Treemap Visualization
Images of the Synset
Downloads
*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.
© 2010 Stanford Vision Lab, Stanford University, Princeton University
Copyright infringement

检测任务

- 解决是什么、在哪里

Parsing at fixed scale

Final list of detections

（语义）分割任务

- 解决每个像素点属于什么类

Cityscapes fine annotations (5,000 images)

Cityscapes coarse annotations (20,000 images)

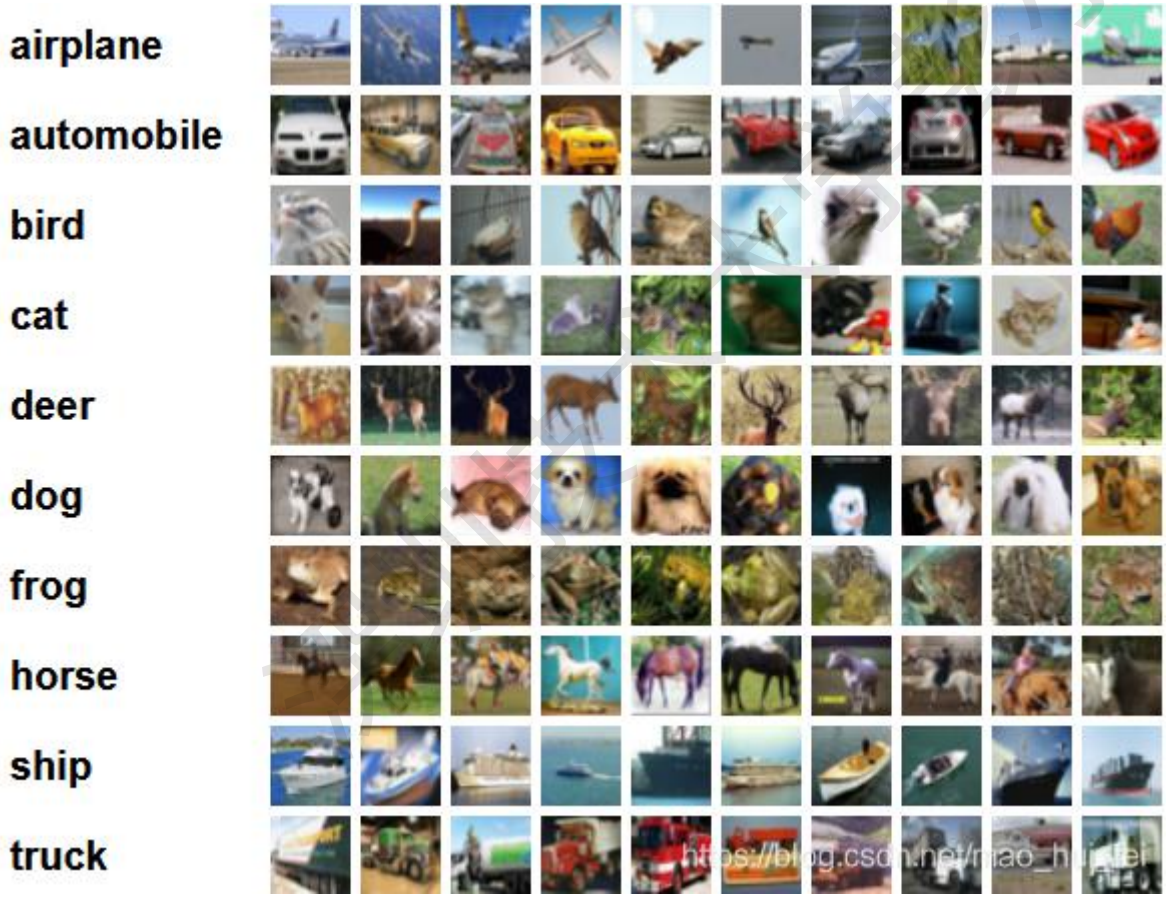
Pytorch自定义CNN模型进行分类

- training from scratch（从零开始训练）
- smoke　non-smoke

代码见project-1

Pytorch自带CNN模型进行分类

- finetuning（微调）CIFAR10物体分类



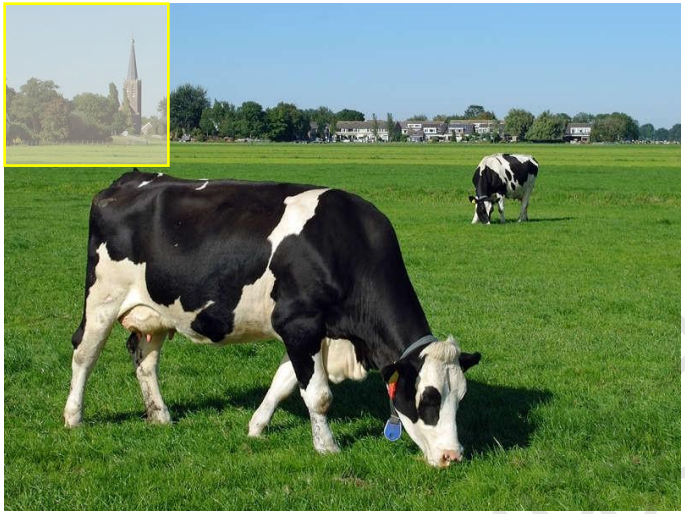
lecture4-cvtasks-classification.py

深度学习目标检测

R-CNN, Fast R-CNN, Faster R-CNN, YOLO, SSD, FPN, YOLOV3, YOLOV5，FCOS

最直接的检测方法

滑动窗口并分类



检测结果

最直接的检测方法

滑动窗口并分类

检测结果

最直接的检测方法

滑动窗口并分类

检测结果

最直接的检测方法

滑动窗口并分类

检测结果

最直接的检测方法

滑动窗口并分类

检测结果

最直接的检测方法

滑动窗口并分类

检测结果

最直接的检测方法

滑动窗口并分类

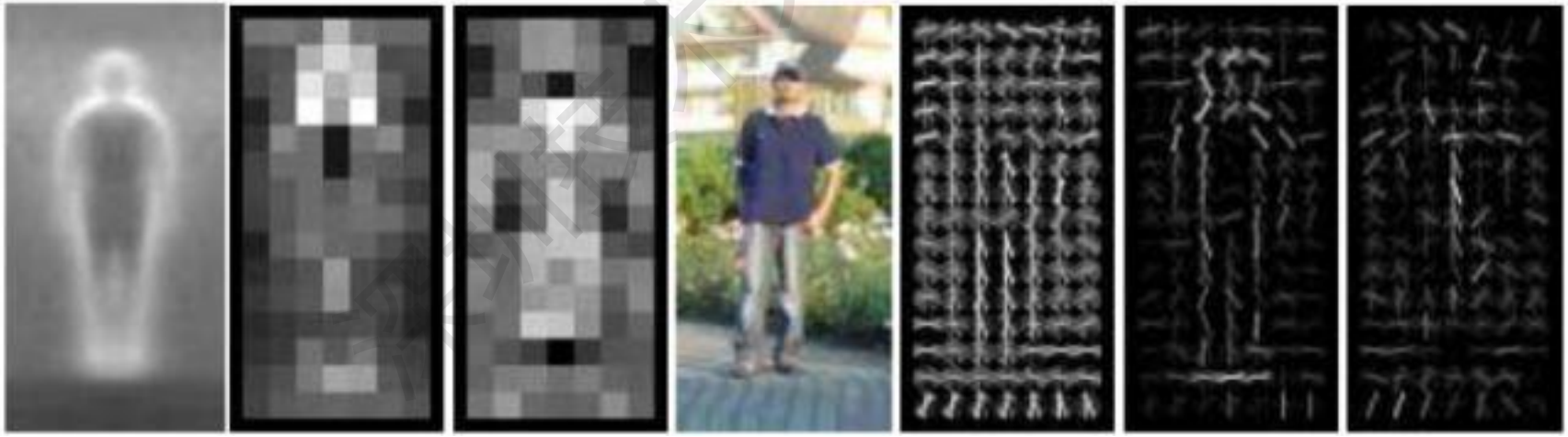
检测结果

这种粗暴的策略需要图片金字塔，很多次计算窗口特征以及分类，非常耗时。

传统特征工程+分类器

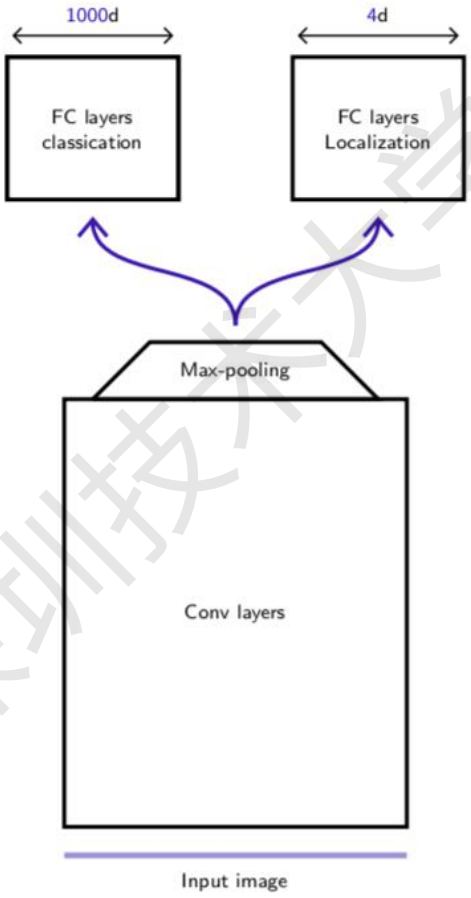
- Haar特征，LBP特征，SIFT特征，HOG特征，BoW特征

- SVM分类器



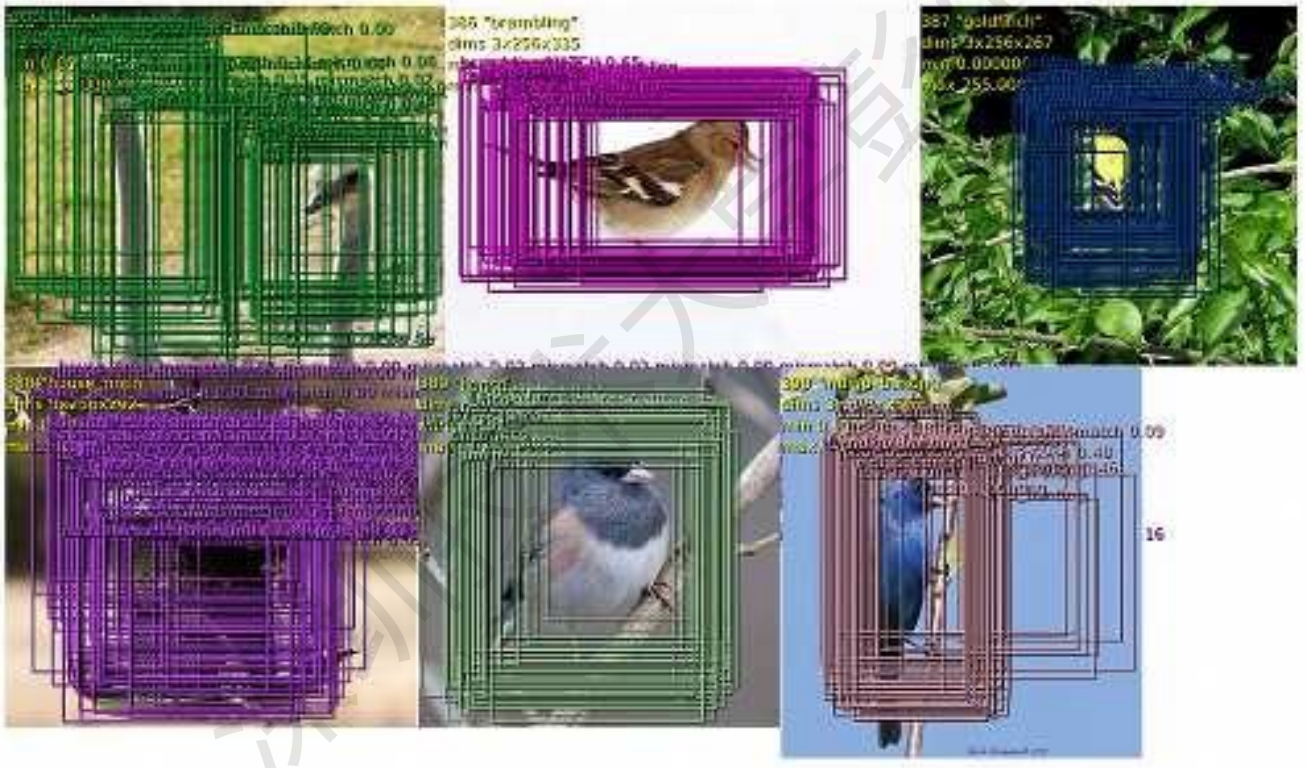
深度学习方法

- **Overfeat:** Sliding window + box回归（减少金字塔图片层数）



深度学习方法

- 使用非极大值抑制去除冗余box



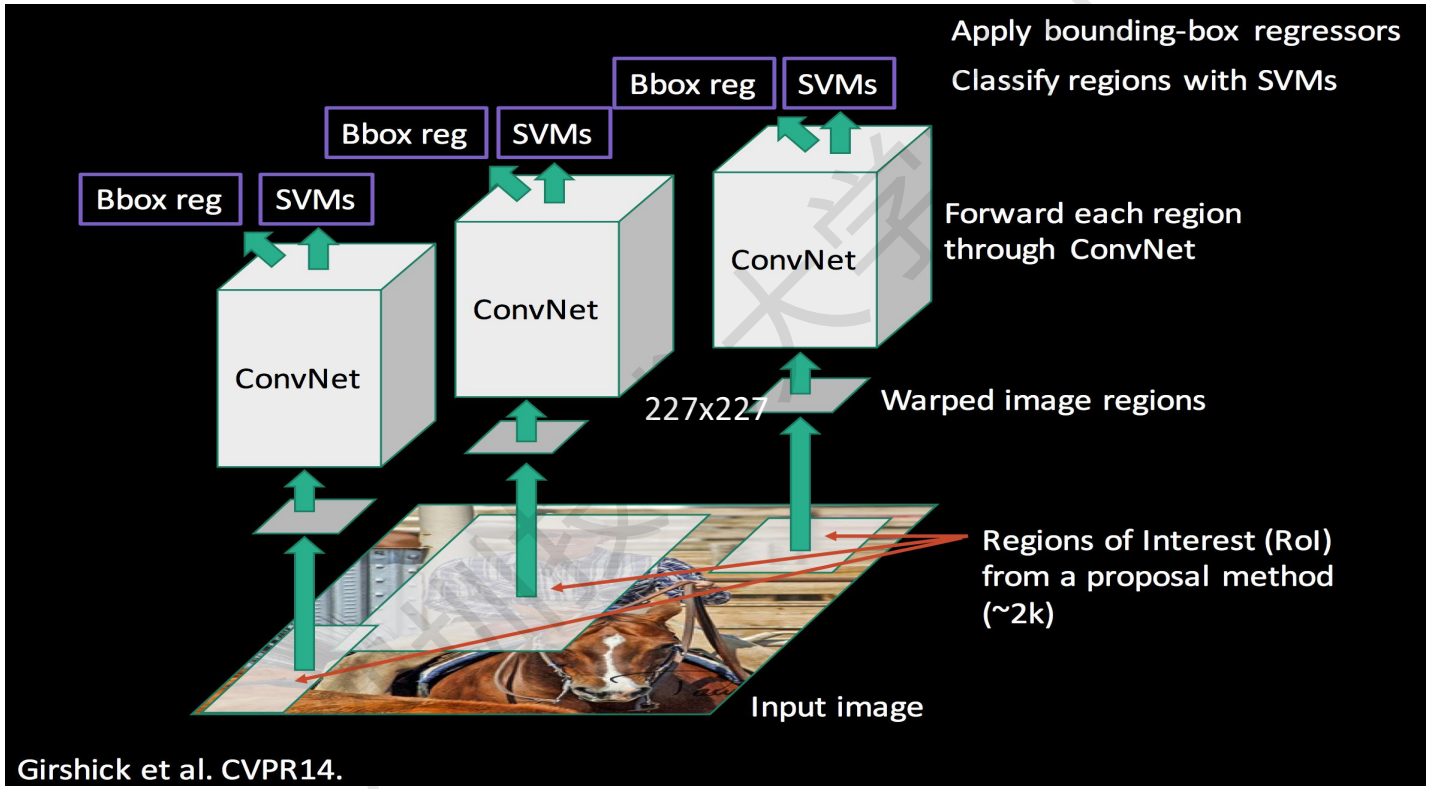
R-CNN

- R-CNN：对每个box做一次CNN前向非常耗时

=>Generate thousands of proposal bounding boxes with a non-CNN "objectness" approach such as Selective search

(b)

R-CNN


Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
227x227
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image
Girshick et al. CVPR14.

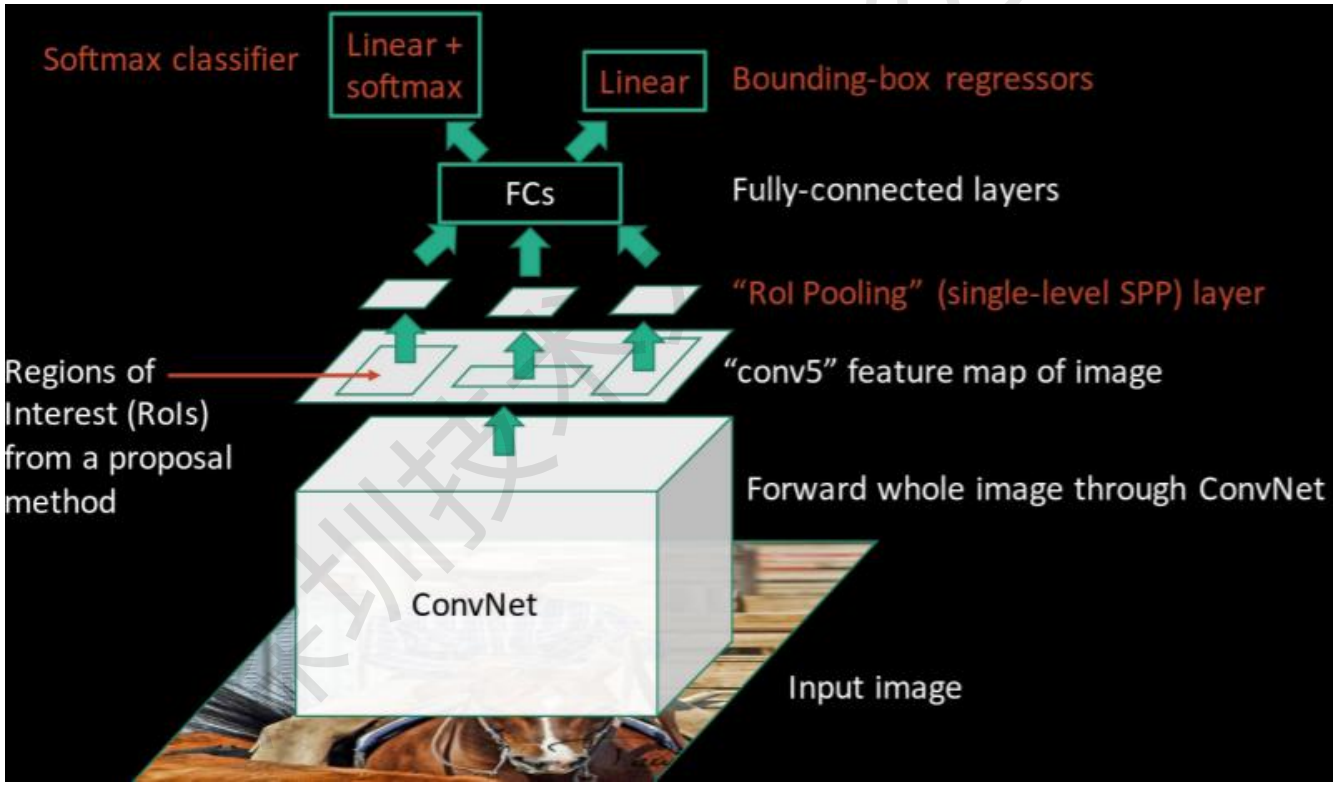
Fast R-CNN

- R-CNN虽然只在2000左右的window进行分类回归，但是2000次CNN前向依然无法实用，10mins/img

- 能否只一次CNN前向运算就好？

Fast R-CNN

- RoI Pooling layer进行区域特征采集



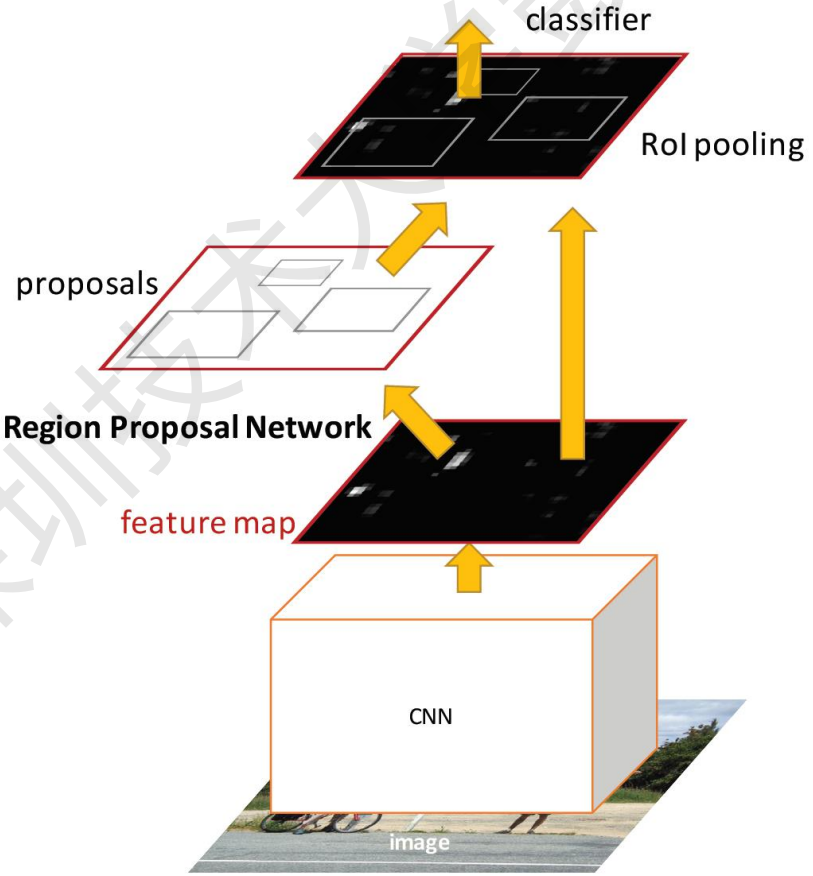
Faster R-CNN

- Fast R-CNN虽然只一次CNN前向,减少了运算，但是selective search进行box选择过程也很耗时，2s/img

- 能否不需要这一步呢?

Faster R-CNN

- 通过设定anchor boxes，然后使用RPN网络对anchor进行回归出“objectness”proposals



低端GPU 5fps!

Faster R-CNN的Pytorch实现

torchvision.models.detection.faster_rcnn

```
def __init__(
    self,
    backbone,
    num_classes=None,
    # transform parameters
    min_size=800,
    max_size=1333,
    image_mean=None,
    image_std=None,
    # RPN parameters
    rpn_anchor_generator=None,
    rpn_head=None,
    rpn_pre_nms_top_n_train=2000,
    rpn_pre_nms_top_n_test=1000,
    rpn_post_nms_top_n_train=2000,
    rpn_post_nms_top_n_test=1000,
    rpn_nms_thresh=0.7,
    rpn_fg_iou_thresh=0.7,
    rpn_bg_iou_thresh=0.3,
    rpn_batch_size_per_image=256,
    rpn_positive_fraction=0.5,
    rpn_score_thresh=0.0,
    # Box parameters
    box_roi_pool=None,
    box_head=None,
    box_predictor=None,
    box_score_thresh=0.05,
    box_nms_thresh=0.5,
    box_detections_per_img=100,
    box_fg_iou_thresh=0.5,
    box_bg_iou_thresh=0.5,
    box_batch_size_per_image=512,
    box_positive_fraction=0.25,
    bbox_reg_weights=None,
):
```

```
if rpn_anchor_generator is None:
    anchor_sizes = ((32,), (64,), (128,), (256,), (512,))
    aspect_ratios = ((0.5, 1.0, 2.0),) * len(anchor_sizes)
    rpn_anchor_generator = AnchorGenerator(anchor_sizes, aspect_ratios)
if rpn_head is None:
    rpn_head = RPNHead(out_channels, rpn_anchor_generator.num_anchors_per_location()[0])

rpn_pre_nms_top_n = dict(training=rpn_pre_nms_top_n_train, testing=rpn_pre_nms_top_n_test)
rpn_post_nms_top_n = dict(training=rpn_post_nms_top_n_train, testing=rpn_post_nms_top_n_test)

rpn = RegionProposalNetwork(
    rpn_anchor_generator,
    rpn_head,
    rpn_fg_iou_thresh,
    rpn_bg_iou_thresh,
    rpn_batch_size_per_image,
    rpn_positive_fraction,
    rpn_pre_nms_top_n,
    rpn_post_nms_top_n,
    rpn_nms_thresh,
    score_thresh=rpn_score_thresh,
)
```

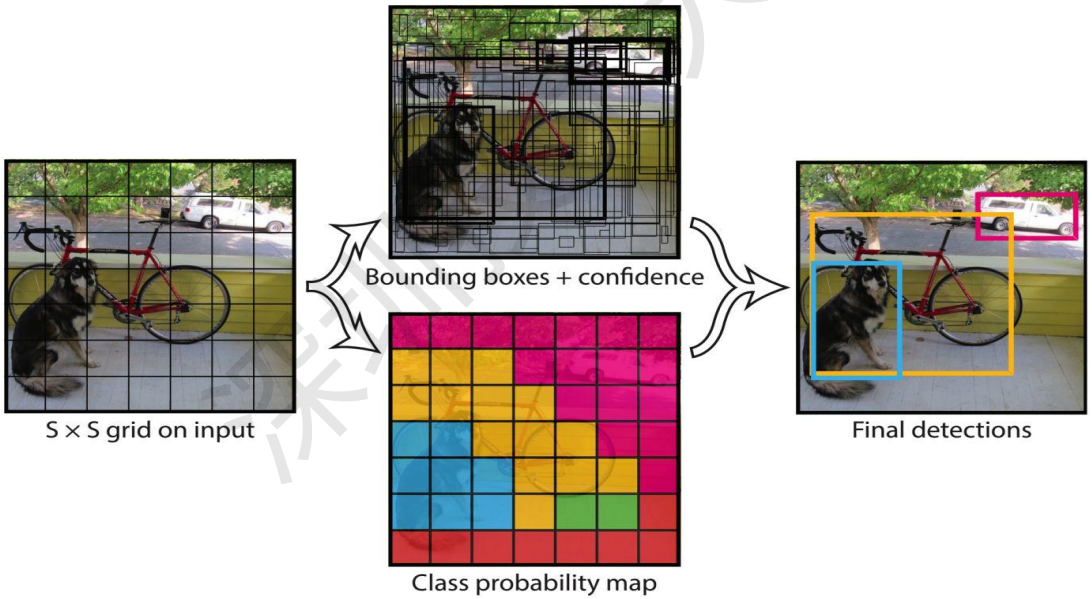
Two-stage vs. one-stage/shot

- Two-stage方案：需要先得到疑似region（Proposals），再进行特征提取和分类

- One-stage: 无需Proposals,直接出检测结果
  - YOLO, SSD

YOLO: You only look once

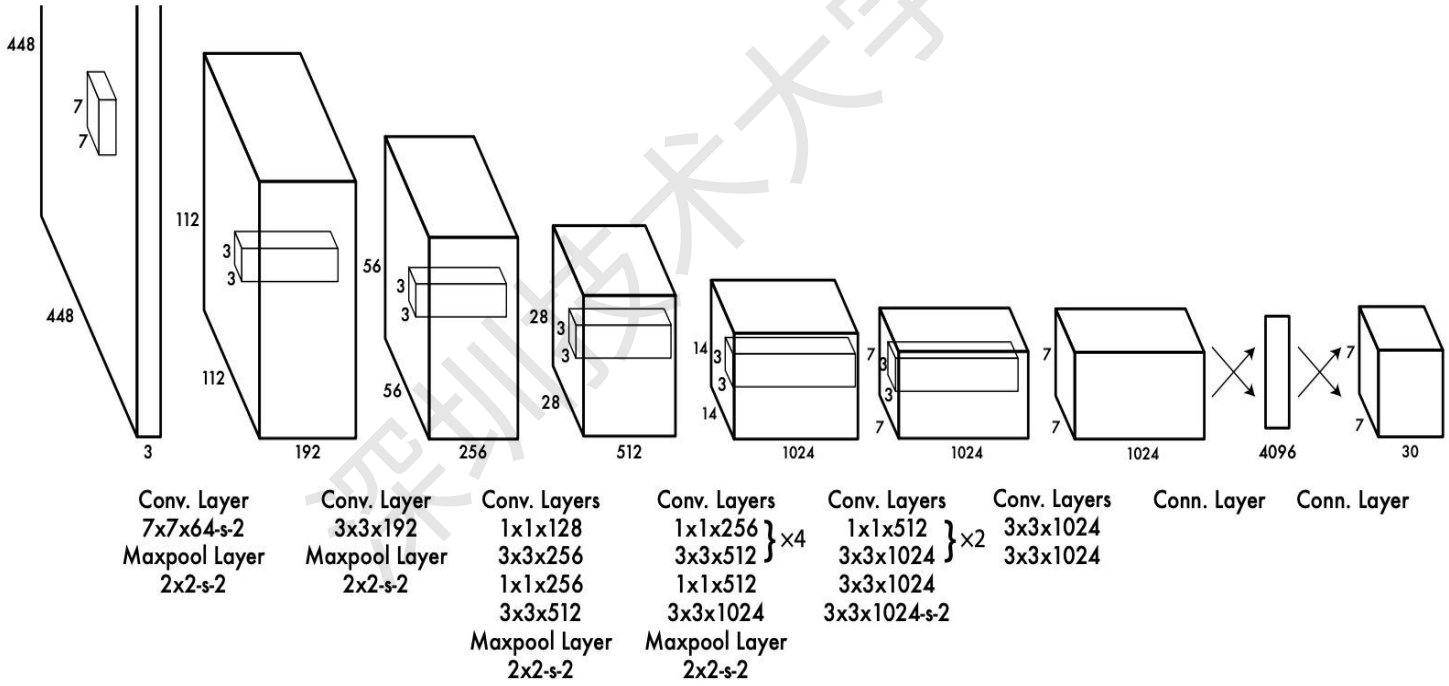
- 特征图每个点设置9个不同大小和比例的anchor box太麻烦
- 能否直接对原图网格进行分类和几个box回归呢?



100+ fps on GPU

YOLO: You only look once

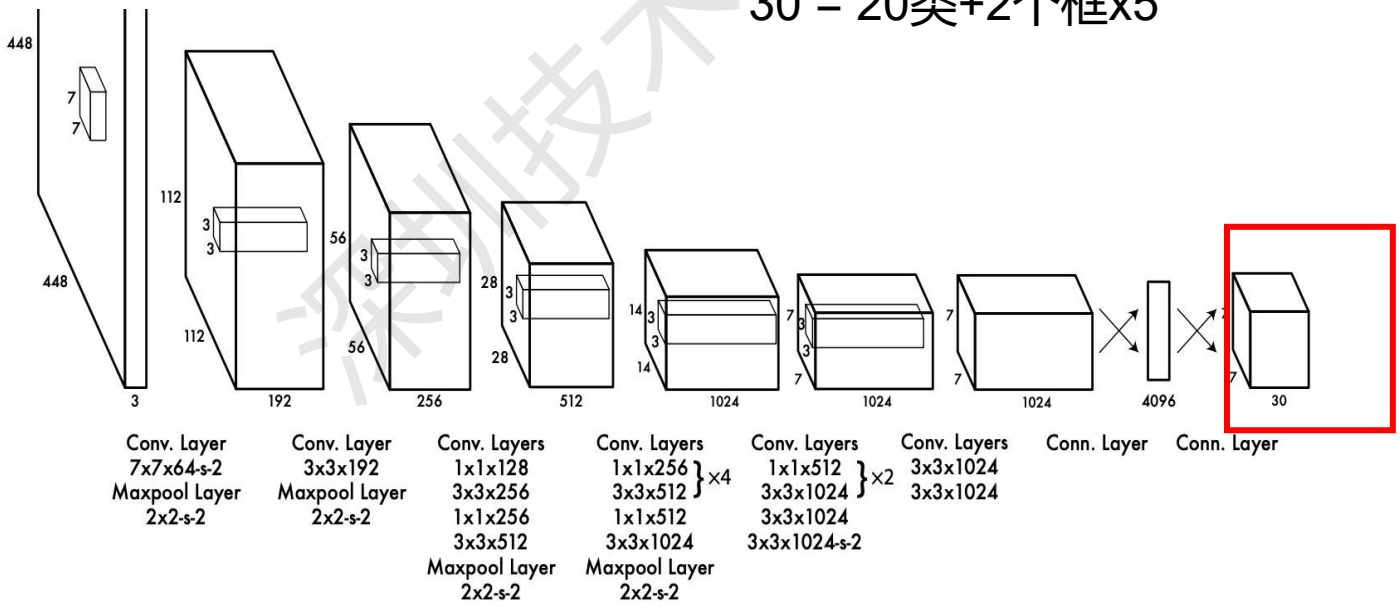
- 首先将输入图片划分为S*S个珊格
- 然后每个格子回归出B个检测框



YOLO: You only look once

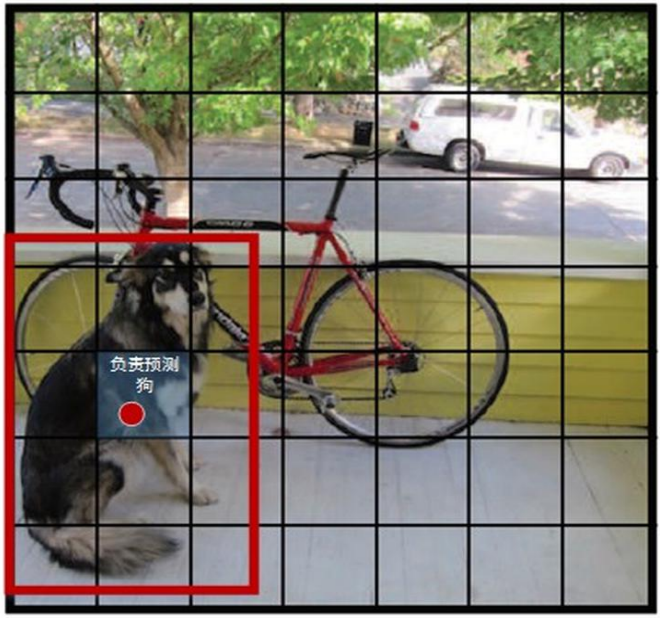
- 每个检测框包含5个值（x,y,w,h,conf），x,y为目标中心点相对于珊格边界的偏移比例，w,h为目标宽高相对于输入图像的比例，conf为预测目标与所有GT的IOU。同时每个珊格回归出其包含类别的概率。

30 = 20类+2个框x5



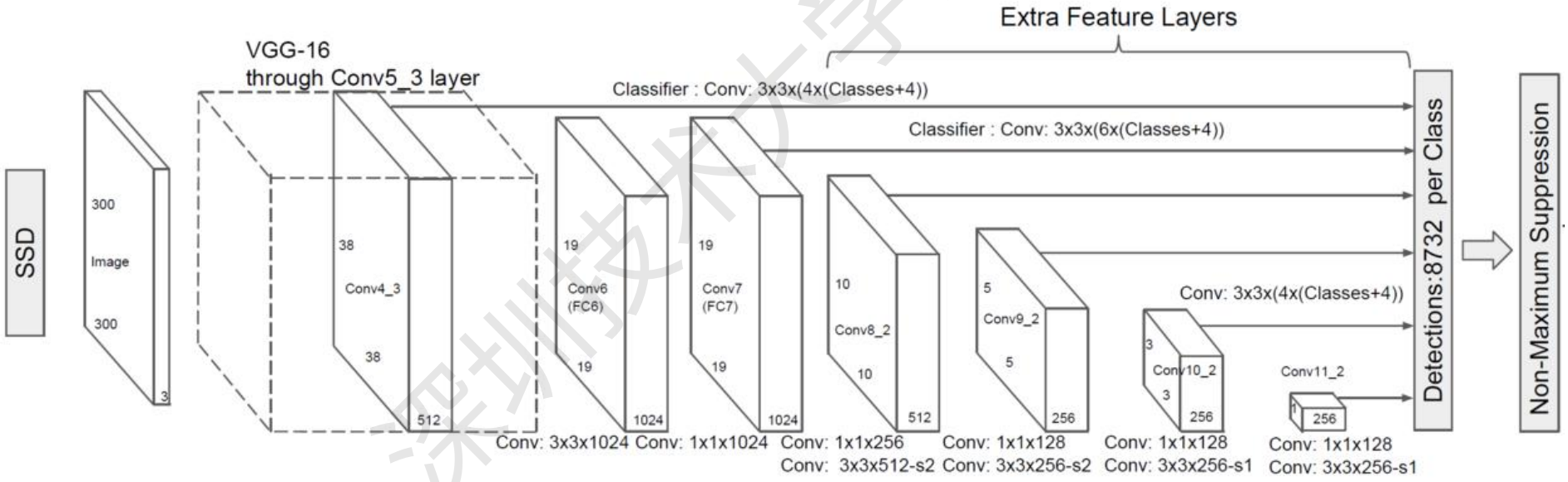
YOLO

- YOLO训练假设目标（中心）总会处于某个网格（框）中
- 训练时候如果网格内有目标则为正样本，会有回归损失和分类损失



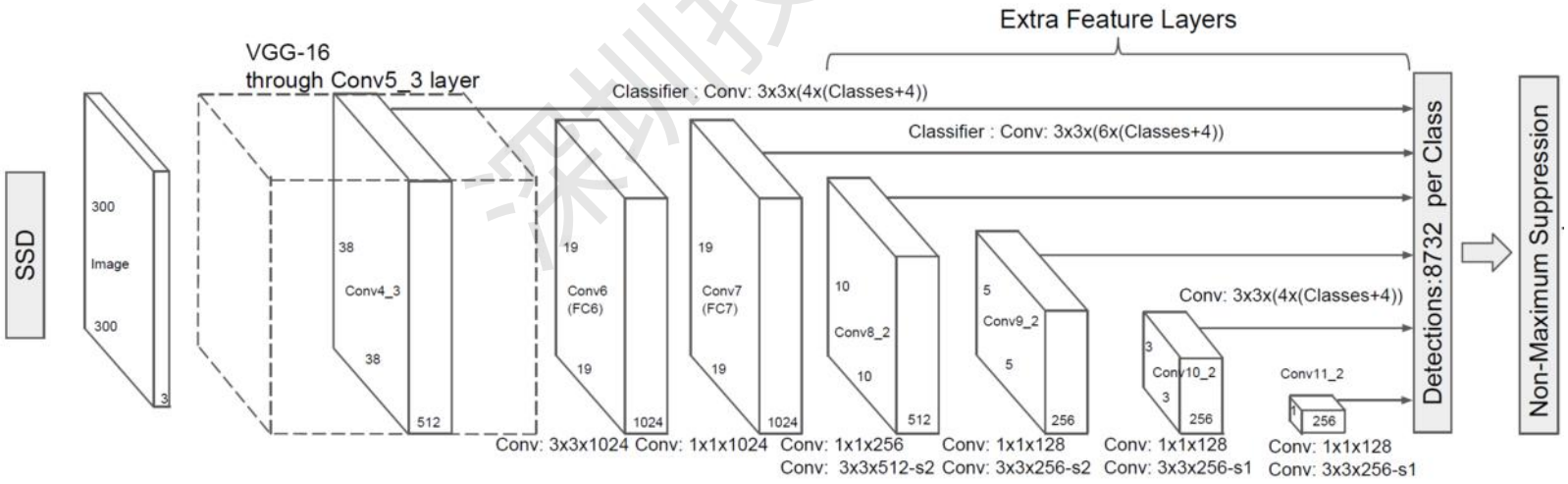
SSD: Single Shot Multi-box Detector

- YOLO只在最后一层预测几个box太粗糙了
- 基于不同层的anchors,多尺度box预测



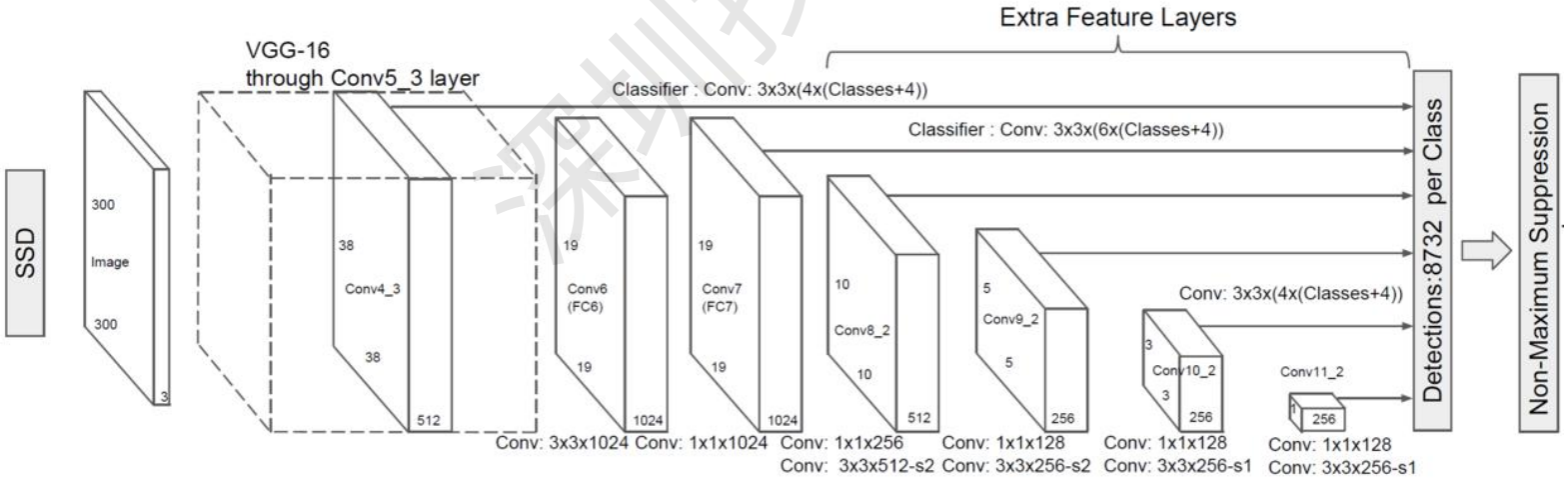
SSD: Single Shot Multi-box Detector

- SSD300: VGG16 conv4 作为 backbone
- Head: conv4_3, conv7,conv8_2,conv9_2,conv10_2, conv11_2, 6个特征图作为输出层，anchor的基本大小设置为30,60,111,162,213,315（相对图像的size）
- 输出层预测的步长为：8,16,32,64,100,300（图像而言）



SSD: Single Shot Multi-box Detector

- 基本大小的anchor按照1:2,2:1,1:3,3:1设定更多anchor
- 用第2层输出为例：30，**60**,111. 为了连续，该层和下一层anchor大小中间还会插入一个sqrt(60x111)=82的基础size. 因此，该层有2个正方形anchor,然后对它们进行1:2,2:1（sxsqrt(2),s/sqrt(2)）变换，得到4个anchor
- 8732=38x38x4+19x19x6+10x10x6+5x5x6+3x3x4+1x4



SSD的Pytorch实现

```python
class SSDHead(nn.Module):
    def __init__(self, in_channels: List[int], num_anchors:
        super().__init__()
        self.classification_head = SSDClassificationHead(in
        self.regression_head = SSDRegressionHead(in_channel

    def forward(self, x: List[Tensor]) -> Dict[str, Tensor]
        return {
            "bbox_regression": self.regression_head(x),
            "cls_logits": self.classification_head(x),
        }
```

```python
# get the features from the backbone
features = self.backbone(images.tensors)
if isinstance(features, torch.Tensor):
    features = OrderedDict([("0", features)])

features = list(features.values())

# compute the ssd heads outputs using the features
head_outputs = self.head(features)

# create the set of anchors
anchors = self.anchor_generator(images, features)

losses = {}
detections: List[Dict[str, Tensor]] = []
if self.training:
    assert targets is not None

    matched_idxs = []
    for anchors_per_image, targets_per_image in zip(anchors, targets):
        if targets_per_image["boxes"].numel() == 0:
            matched_idxs.append(
                torch.full((anchors_per_image.size(0),), -1, dtype=torch.int64, device=anchors_per_image.device)
            )
            continue

        match_quality_matrix = box_ops.box_iou(targets_per_image["boxes"], anchors_per_image)
        matched_idxs.append(self.proposal_matcher(match_quality_matrix))

    losses = self.compute_loss(targets, head_outputs, anchors, matched_idxs)
else:
    detections = self.postprocess_detections(head_outputs, anchors, images.image_sizes)
    detections = self.transform.postprocess(detections, images.image_sizes, original_image_sizes)
```

lecture4-cvtasks-detection.py

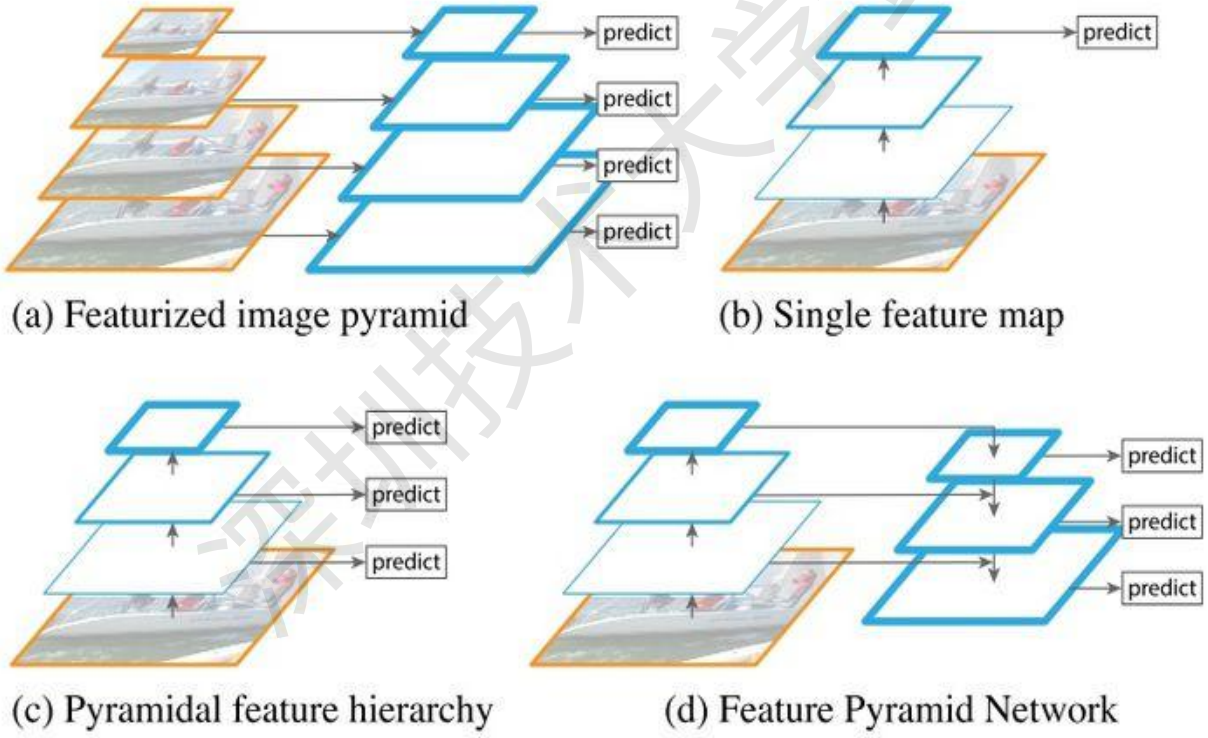
YoloV2

- hi-res classifier: 448*448 的高分辨率样本对分类模型进行微调再用于检测模型,缓解分辨率切换造成的影响
- location prediction: 将预测边框的中心约束在特定gird网格内,设定先验框的宽和高. yolov1直接预测绝对位置训练很不稳定
- Multi-Scale Training（多尺度图像训练）

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

FPN:Feature Pyramid Networks for Object Detection

- 想法：将语义丰富的高层特征融合到细节丰富的底层特征，对小物体检测能从宏观特征进行帮助

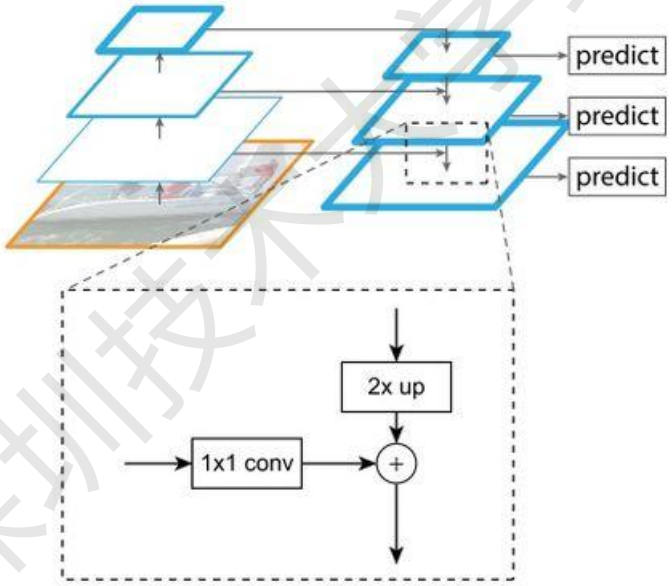


(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

FPN:Feature Pyramid Networks for Object Detection

- 具体做法



YoloV2 (DarkNet-19)

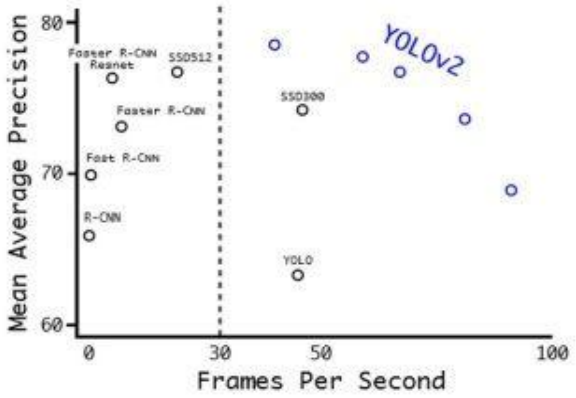


Figure 4: Accuracy and speed on VOC 2007.

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

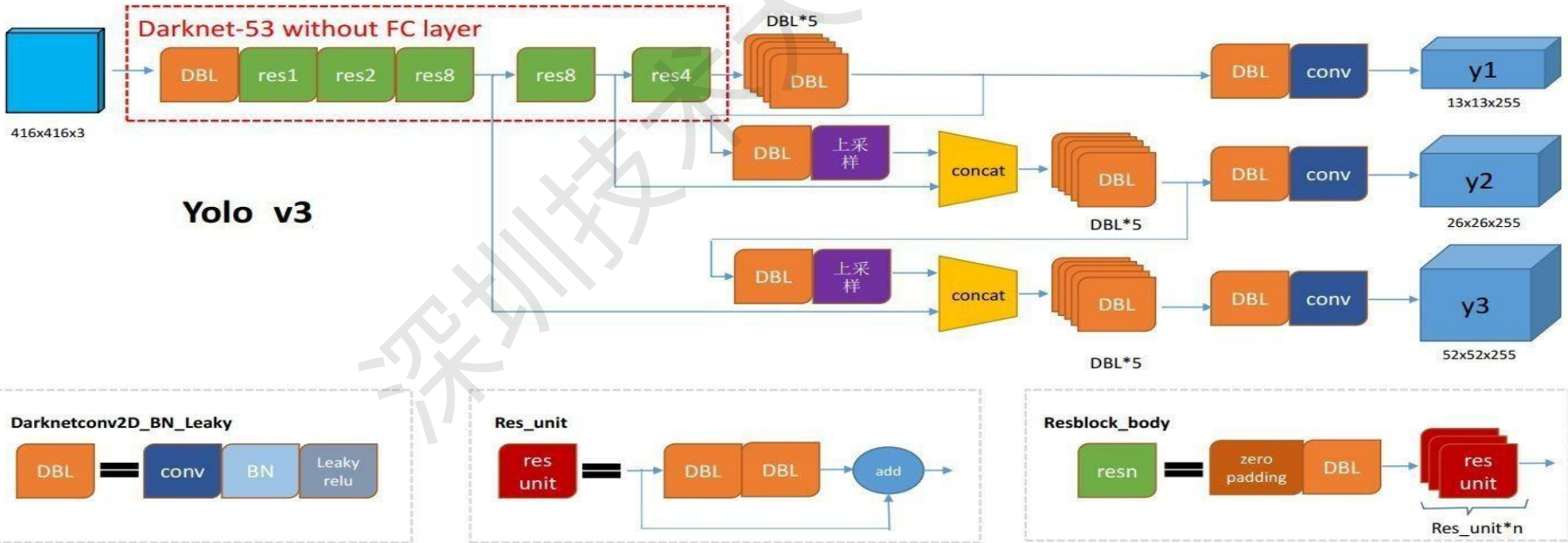
Table 3: Detection frameworks on PASCAL VOC 2007. YOLOv2 is faster and more accurate than prior detection methods. It can also run at different resolutions for an easy tradeoff between speed and accuracy. Each YOLOv2 entry is actually the same trained model with the same weights, just evaluated at a different size. All timing information is on a Geforce GTX Titan X (original, not Pascal model).

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN [15]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [14]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

Table 4: PASCAL VOC2012 `test` detection results. YOLOv2 performs on par with state-of-the-art detectors like Faster R-CNN with ResNet and SSD512 and is $2-10\times$ faster.

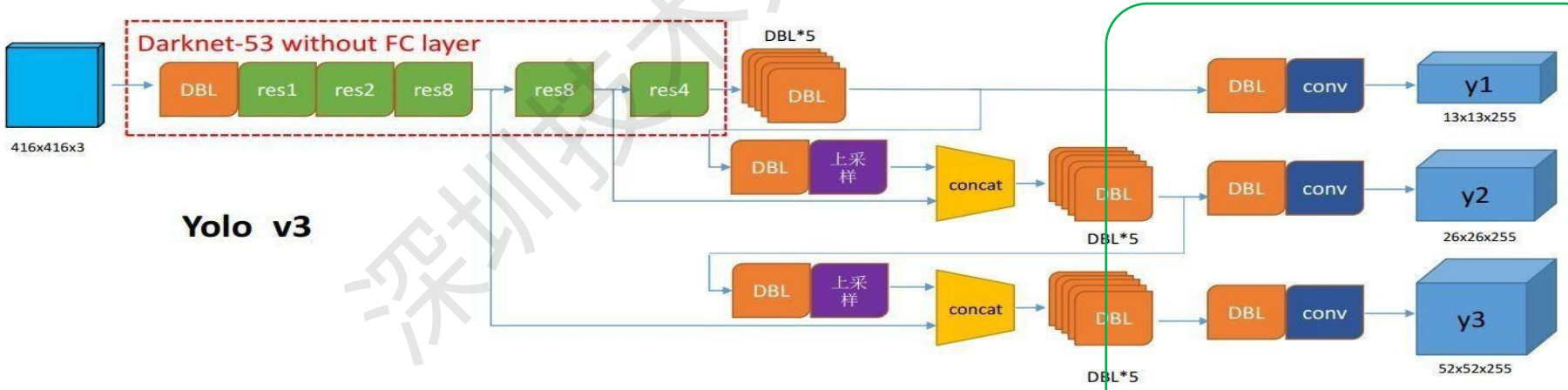
YoloV3 (Darknet-53)

- DBL: yolo_v3的基本组件，卷积+BN+Leaky relu
- resn：n代表数字，有res1，res2，…，res8等等，表示这个res_block里含有多少个res_unit

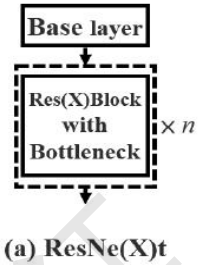


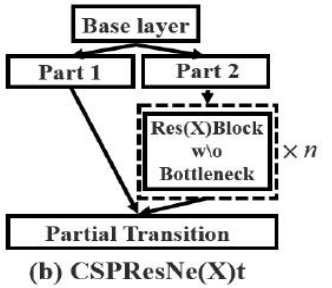
YoloV3 (Darknet-53)

- FPN的机制
- 多尺度预测(类似SSD): 13x13, 26x26,52x52 feature map
  - （13x13）适合检测大的目标，每个cell的三个anchor boxes为(116 ,90),(156 ,198)，(373 ,326)
  - （26x26）适合一般大小的物体，anchor boxes为(30,61)， (62,45)，(59,119)

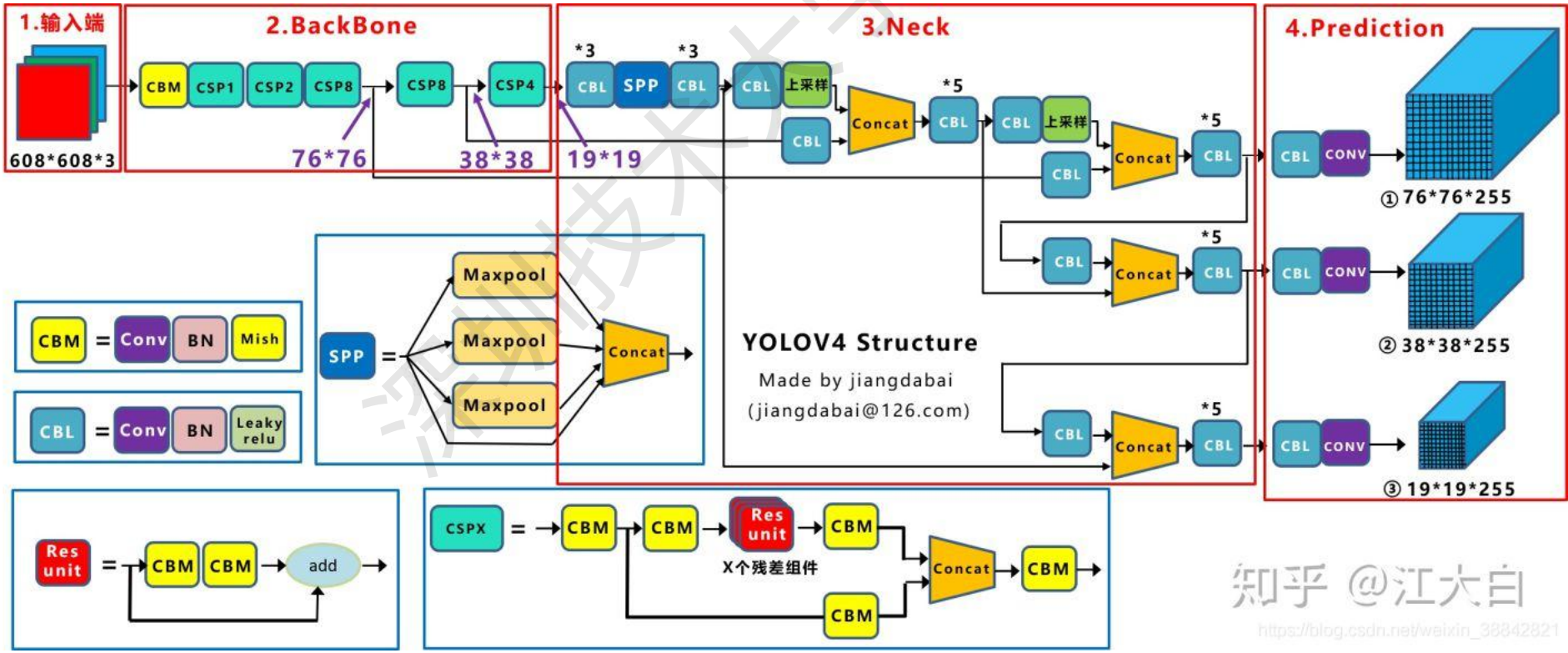


YoloV4





- CSPNet结构（Cross Stage Partial Network）
- 减少了计算量的同时可以保证准确率



YoloV4

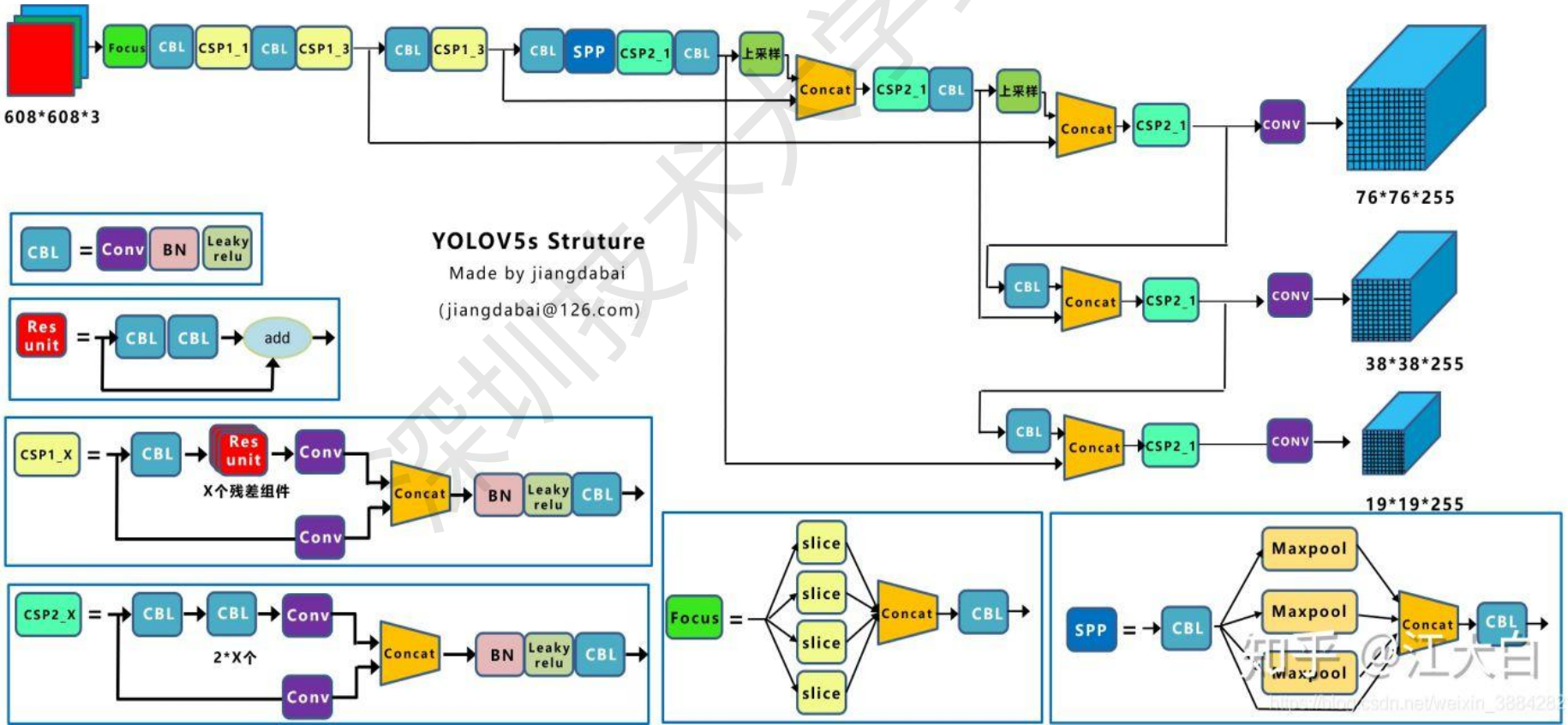
- 输入端采用mosaic数据增强：多张图截切后拼起来
- Backbone上采用了CSPDarknet53、Mish激活函数、Dropblock等方式。Mish函数为

$$Mish=x*\ tanh(ln(1+e^x))$$

YoloV5

https://github.com/ultralytics/yolov5

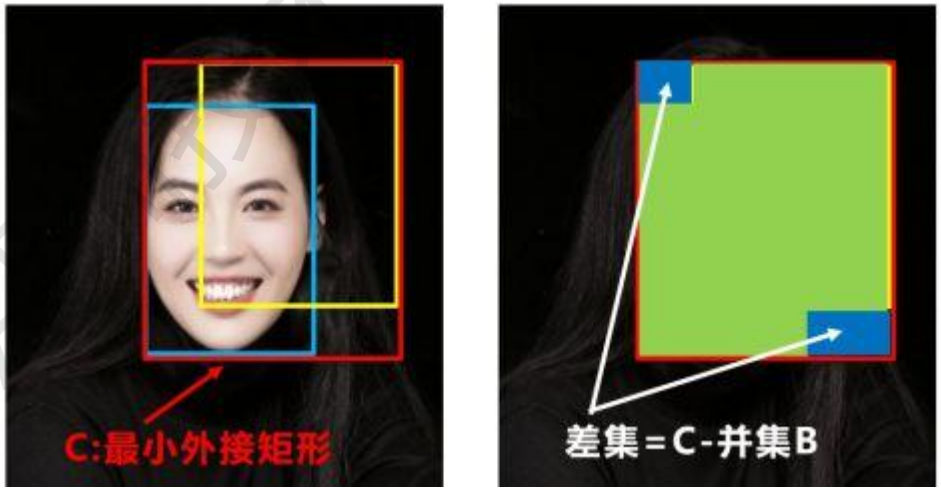
- mosaic数据增强，自适应锚框计算
- 自适应图片缩放,Focus结构



YoloV5

- 采用GIOU_Loss做Bounding box的损失函数,Ac代表最小外接矩形面积，U代表并集区域

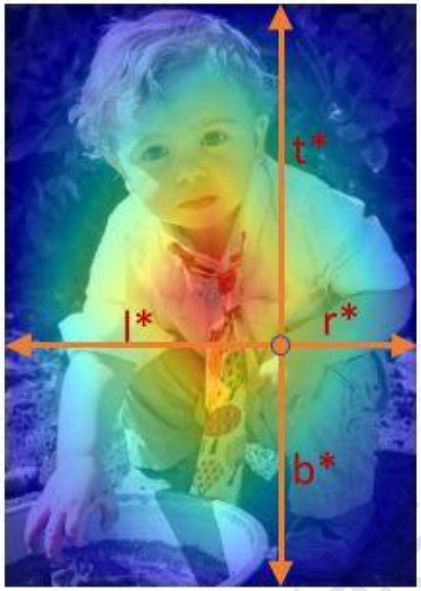
$$GIoU = IoU - \frac{|A_c - U|}{|A_c|}$$



Anchor based vs. anchor free

- R-CNN系列，yolo系列，SSD系列都是预设定了anchor box作为与标注的box进行loss计算
- 设定anchor需要“神一样的调参技巧”
- 是否可以anchor free?懒得调

FCOS: Fully Convolutional One-Stage Object Detection

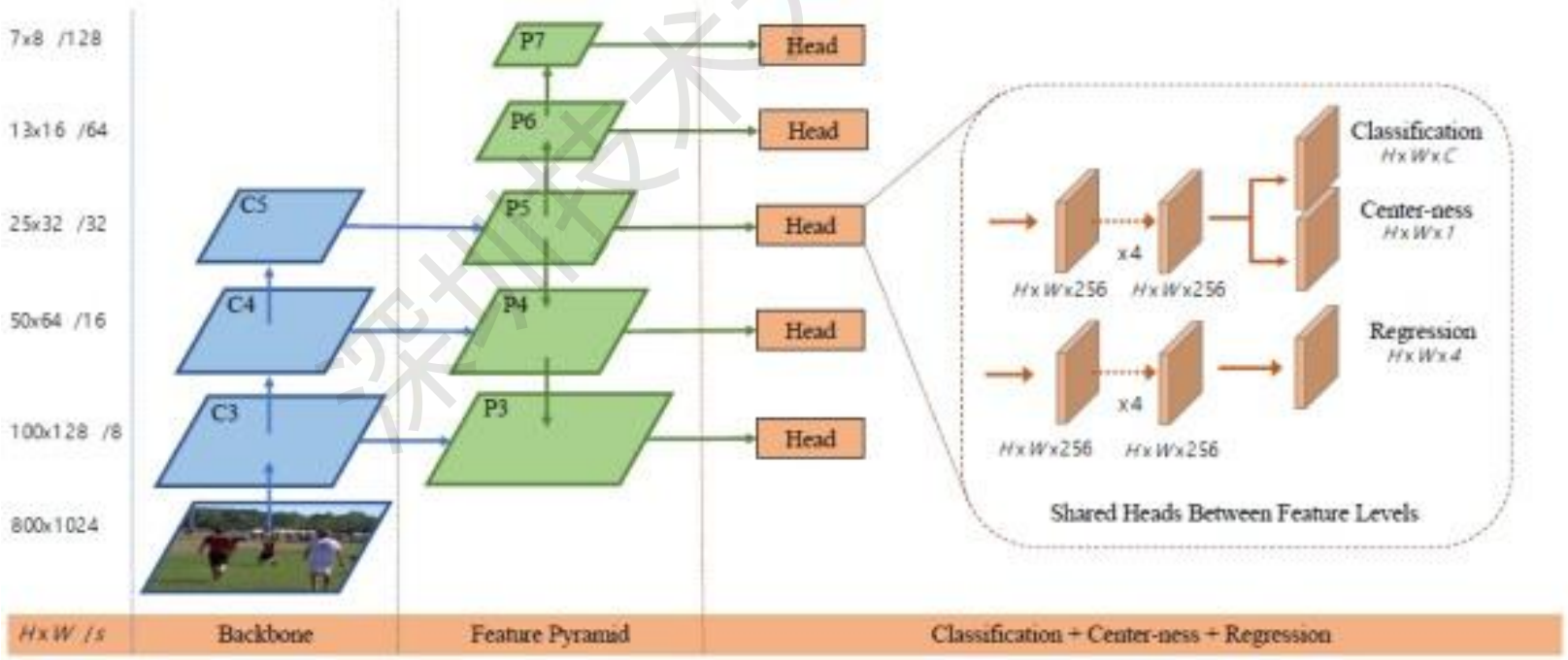


$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}$$

Figure 3 – Center-ness. Red, blue, and other colors denote 1, 0 and the values between them, respectively. Center-ness is computed by Eq. (3) and decays from 1 to 0 as the location deviates from the center of the object. When testing, the center-ness predicted by the network is multiplied with the classification score thus can down-weight the low-quality bounding boxes predicted by a location far from the center of an object.

FCOS: Fully Convolutional One-Stage Object Detection

- 不算IOU，如果某个位置落入了任何gt中，那么该位置就被认为是正样本，并且类别为该gt的类别，小目标更容易检测



SSD或者YoloV5训练

深度学习之分割

深度学习下的分割

- 语义分割
- 实例分割

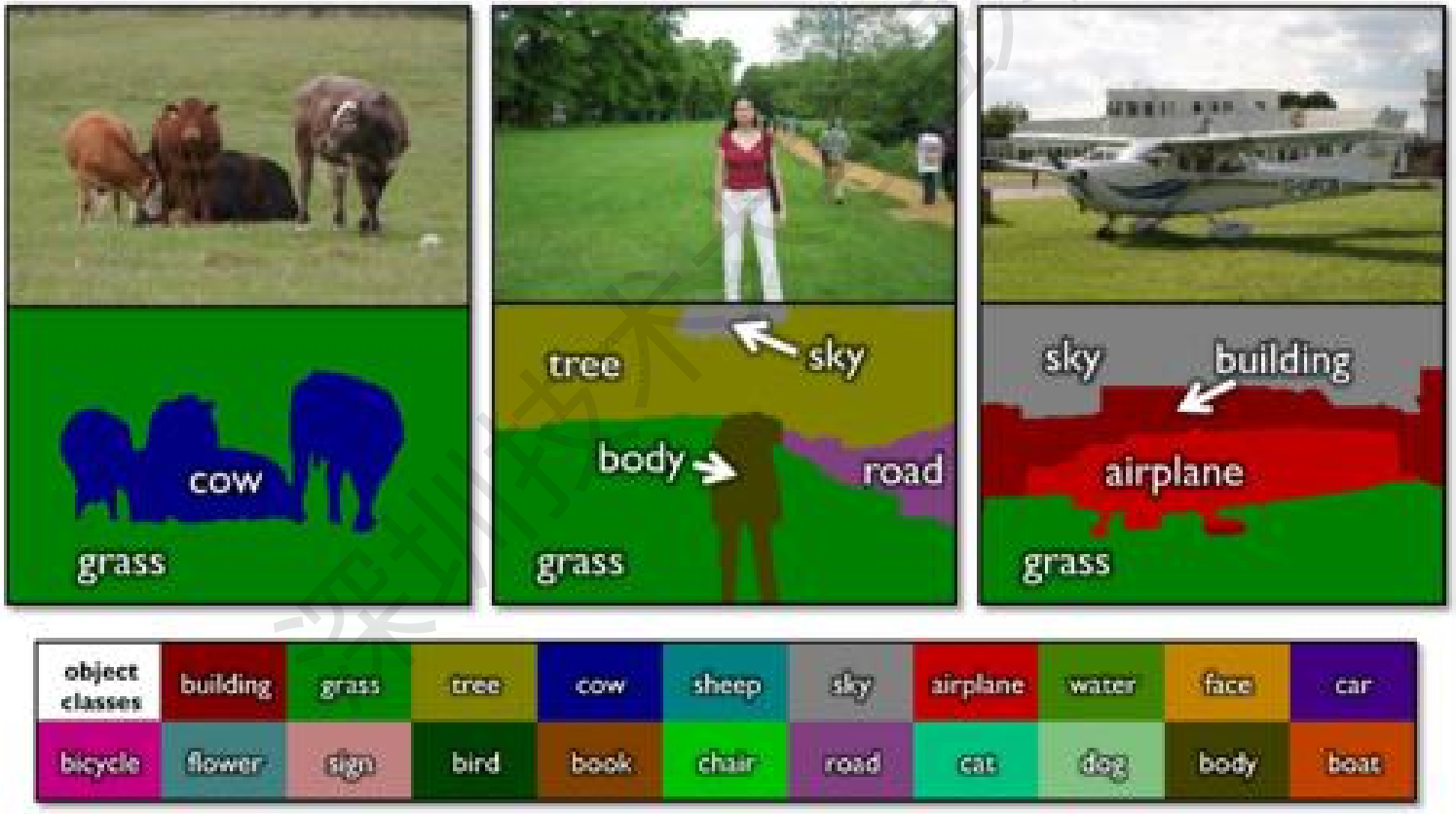
语义分割

- 语义分割（全像素语义分割）作为经典的计算机视觉问题（图像分类，物体检测识别，语义分割）。其涉及将一些原始数据（例如：平面图像）作为输入并将它们转换为具有突出显示的感兴趣区域的掩膜，其中图像中的每个像素根据其所属的对象被分配类别ID

传统基于像素颜色分割方法会把地面分割成很多ID

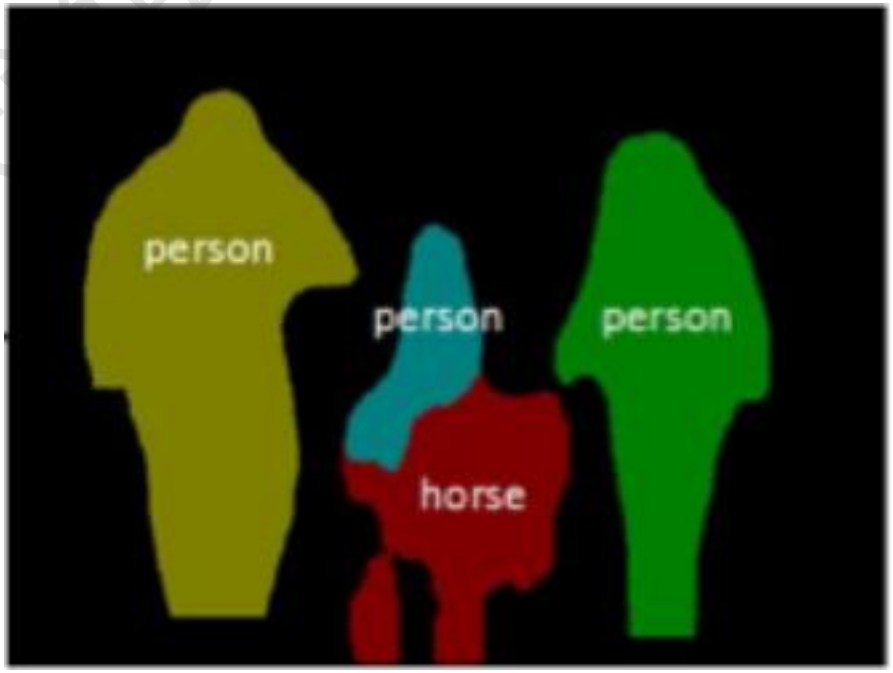
语义分割

- 不区分实例：如奶牛像素



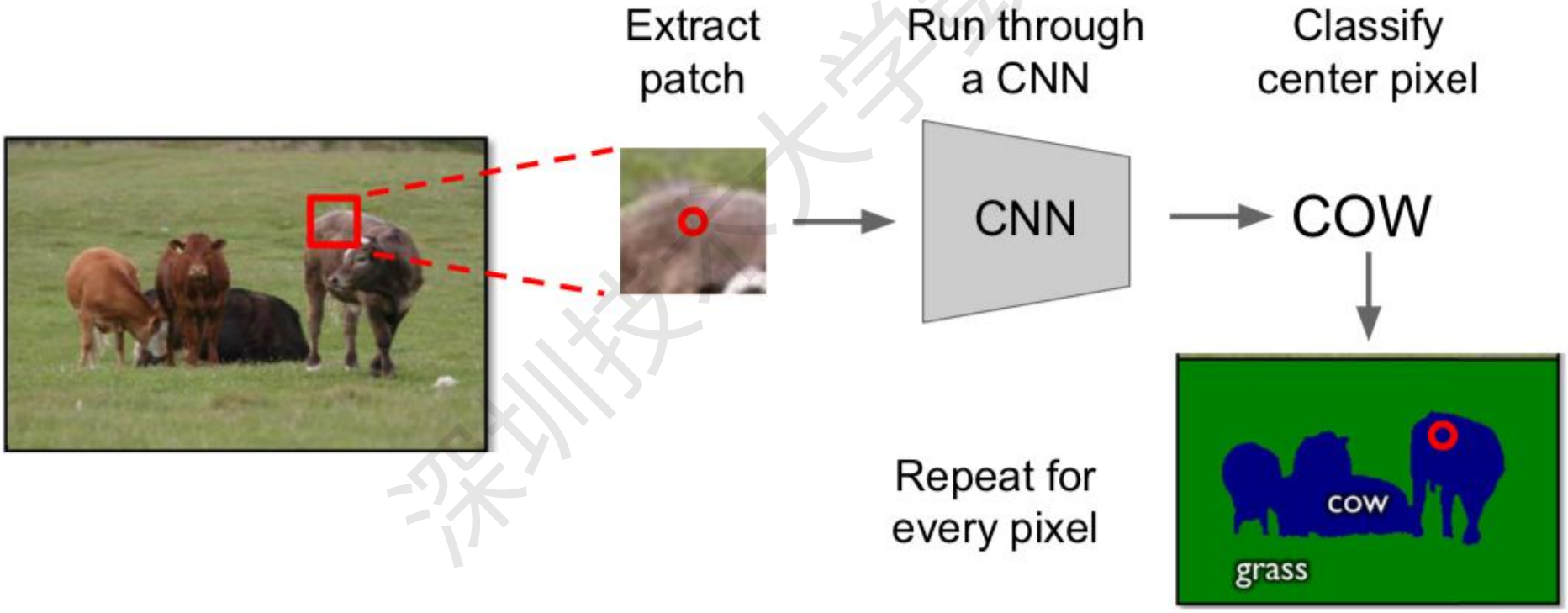
实例分割

- 自带检测，区分每个实例



Semantic segmentation (语义分割)

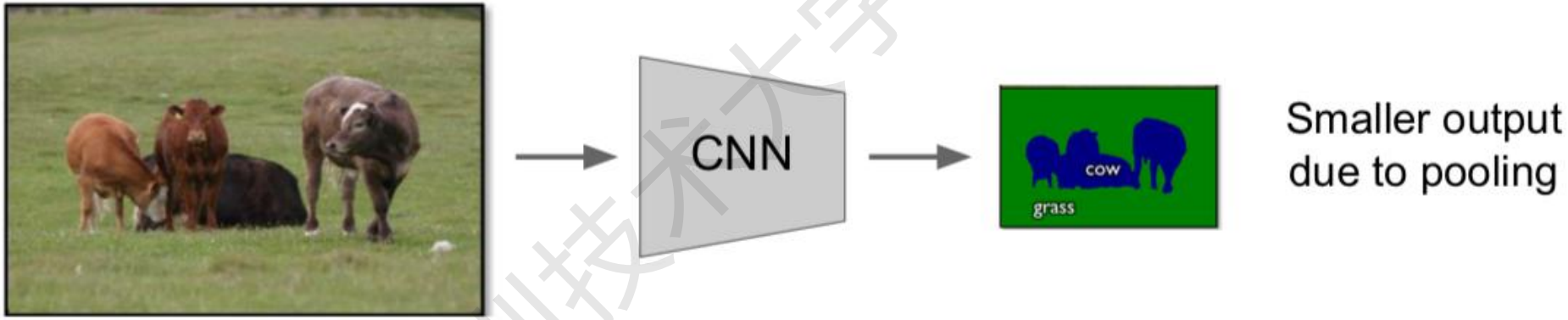
- CNN-based Naïve method



Semantic segmentation (语义分割)

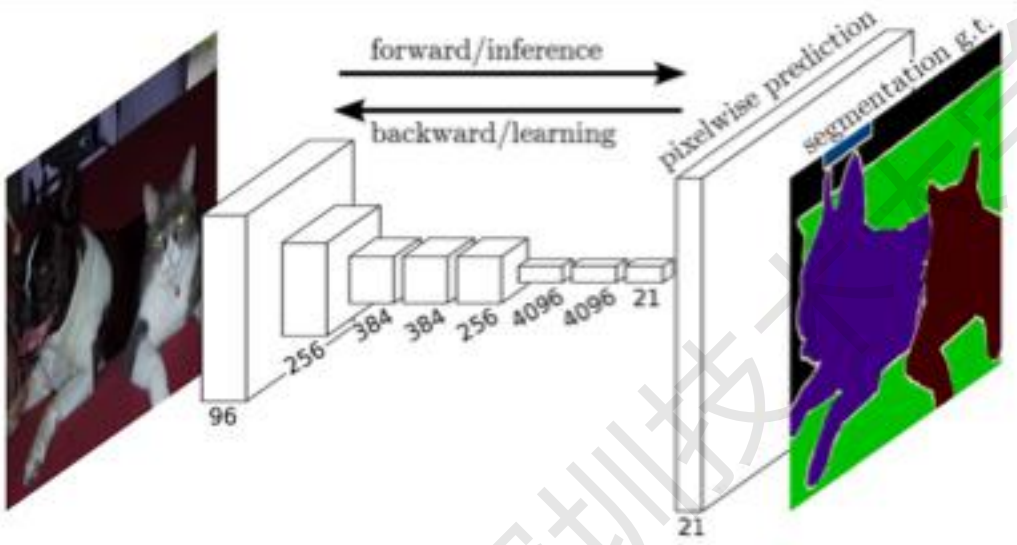
- 深度学习分割开山之作：全卷积网络

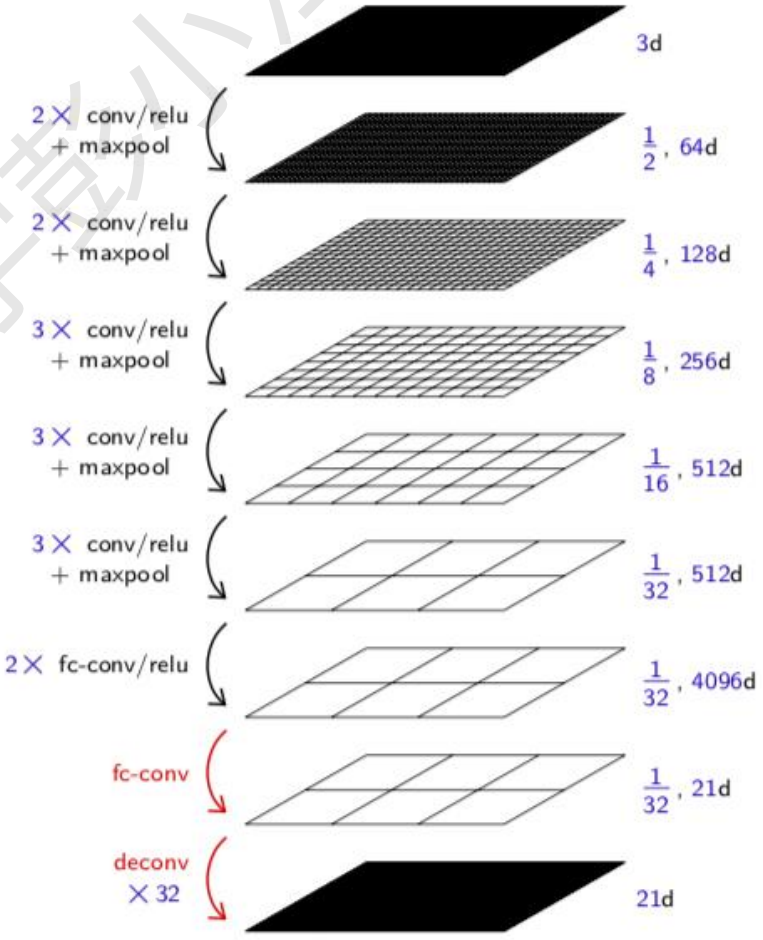
Run “fully convolutional” network
to get all pixels at once



Semantic segmentation (语义分割)

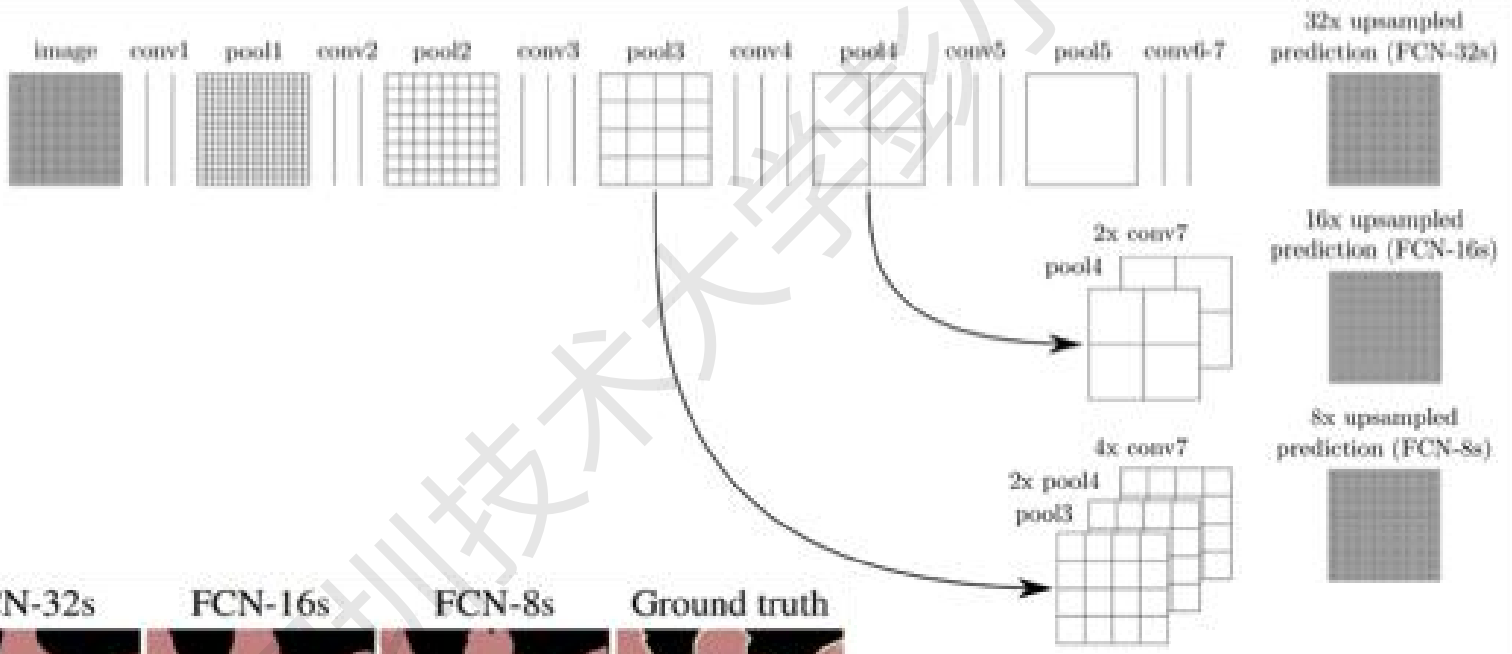
- 全卷积网络

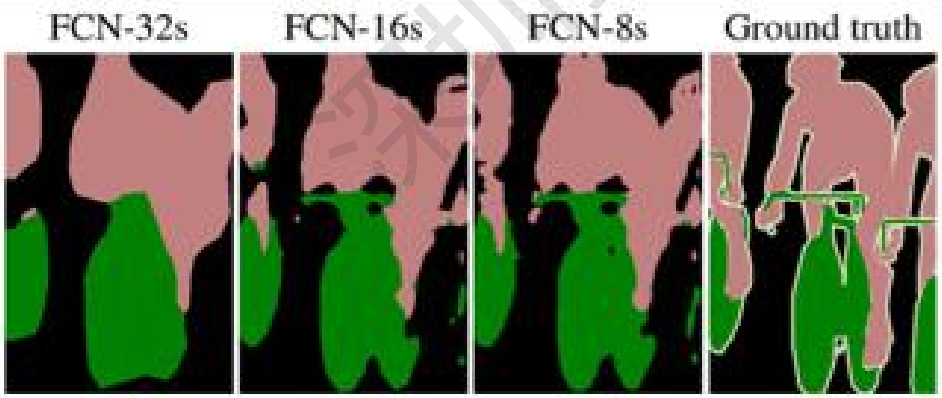




Semantic segmentation (语义分割)

- Skip-connection





常用的语义分割数据库

- VOC2012：有 20 类目标，这些目标包括人类、机动车类以及其他类，可用于目标类别或背景的分割

- MSCOCO：是一个新的图像识别、分割和图像语义数据集，是一个大规模的图像识别、分割、标注数据集。它可以用于多种竞赛，与本领域最相关的是检测部分，因为其一部分是致力于解决分割问题的。该竞赛包含了超过80个物体类别

- Cityscapes ： 50 个城市的城市场景语义理解数据集

常用的语义分割评价指标

- VOC2012 & MSCOCO

Pascal Visual Object Classes
20 Classes
~ 5.000 images

Microsoft COCO
80 Classes
~ 300.000 images

基于类进行计算的IoU就是将每一类的IoU计算之后累加，再进行平均，得到的就是基于全局的评价，所以我们求的IoU其实是取了均值的IoU，也就是均交并比（mean IoU）

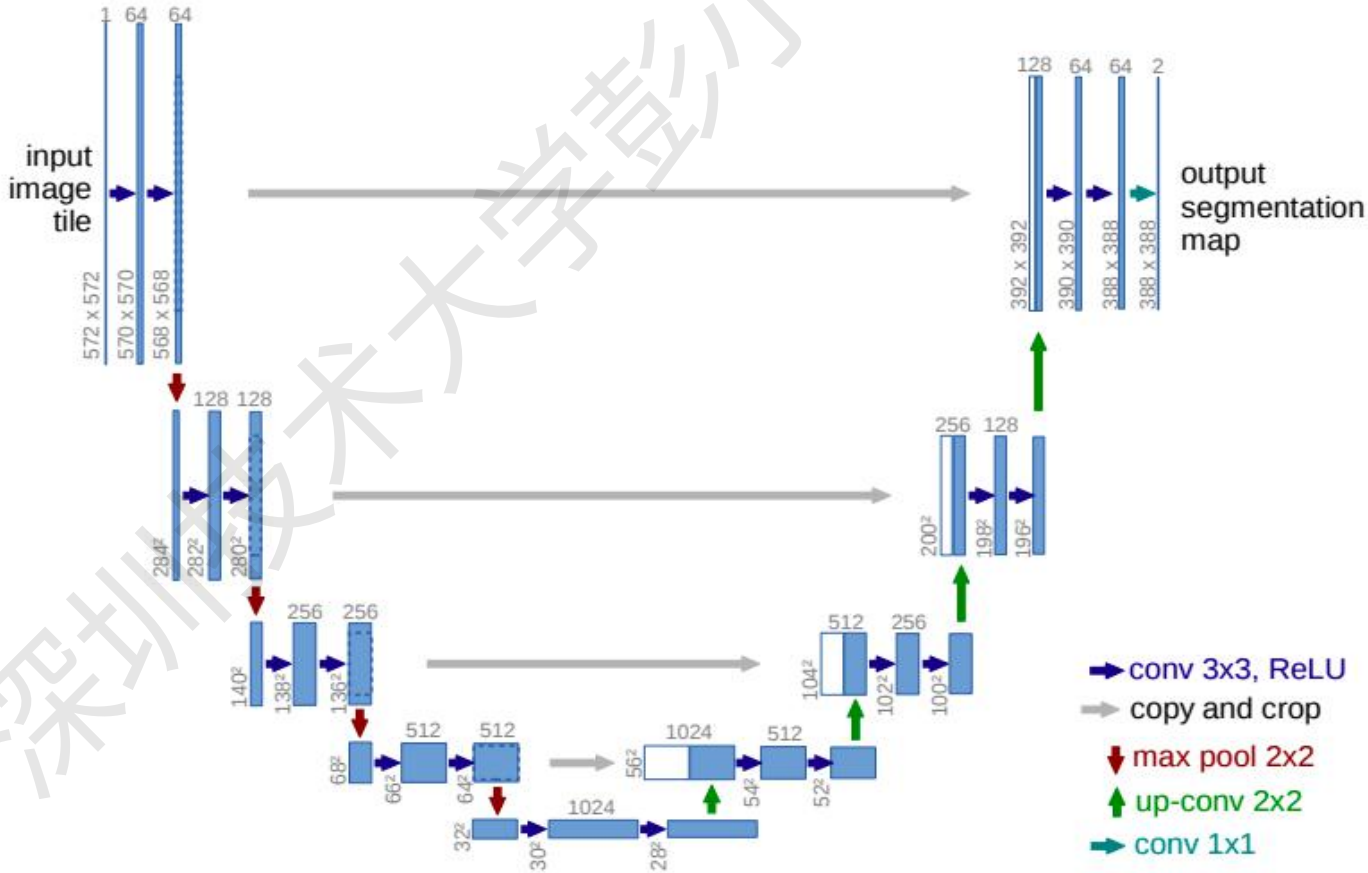
pixcal-accuracy （PA，像素精度）：基于像素的精度计算是评估指标中最为基本也最为简单的指标，从字面上理解就可以知道，PA是指预测正确的像素占总像素的比例.

经典深度学习语义分割方法

- U-Net
- DeepLab
- PSPNet

U-Net

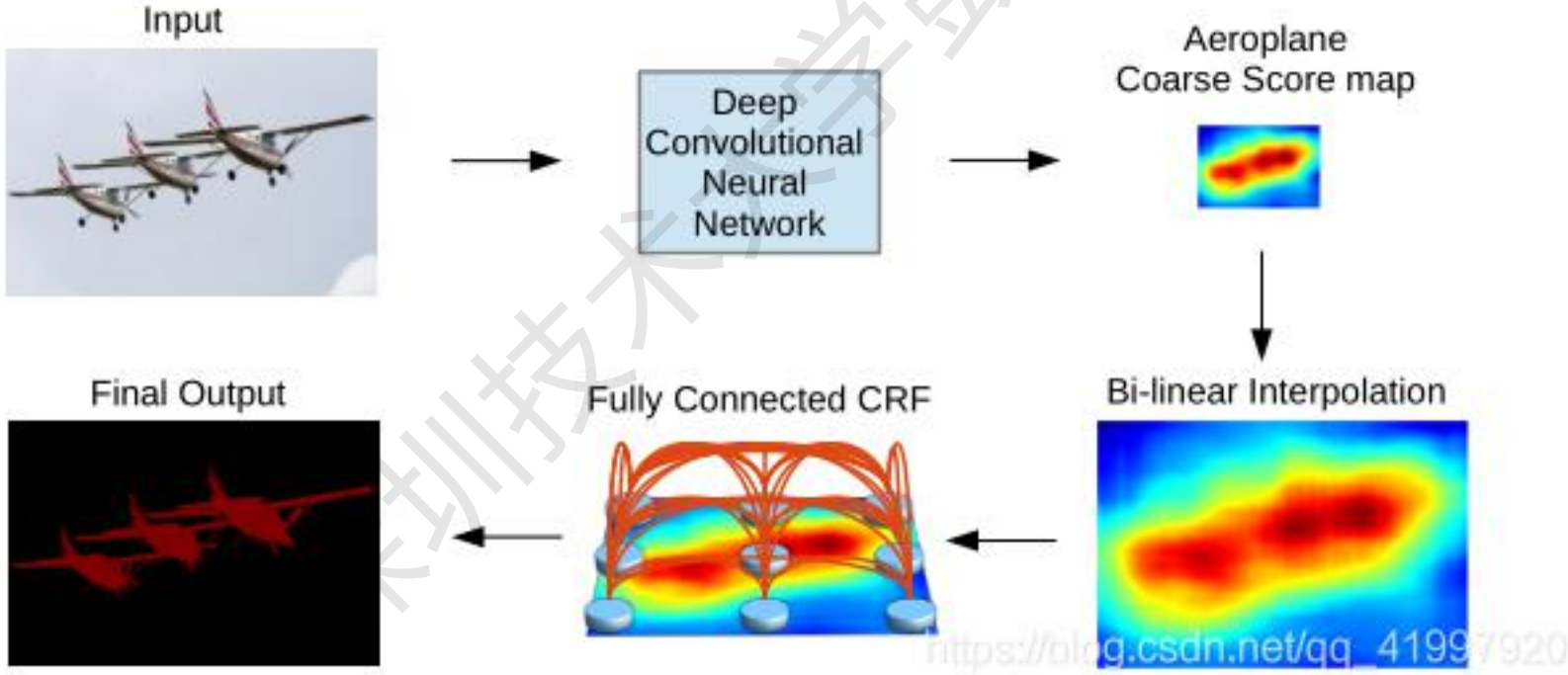
- 左边编码器，右边解码器，skip为channel 串联



U-Net: Convolutional Networks for Biomedical Image Segmentation （2015）

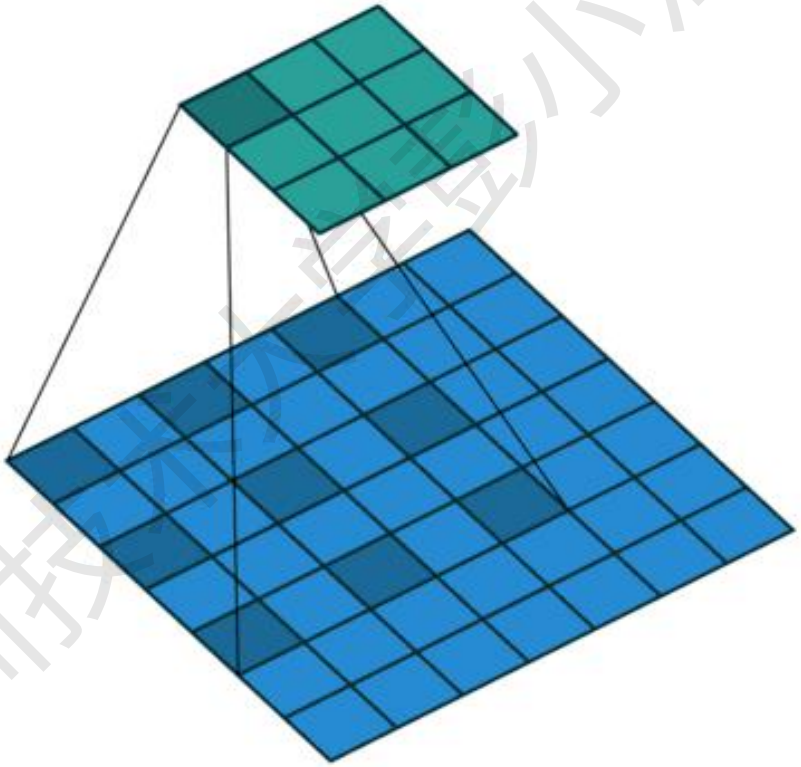
DeepLab(v2)

- 空洞/多孔卷积，CRF考虑所有像素关系



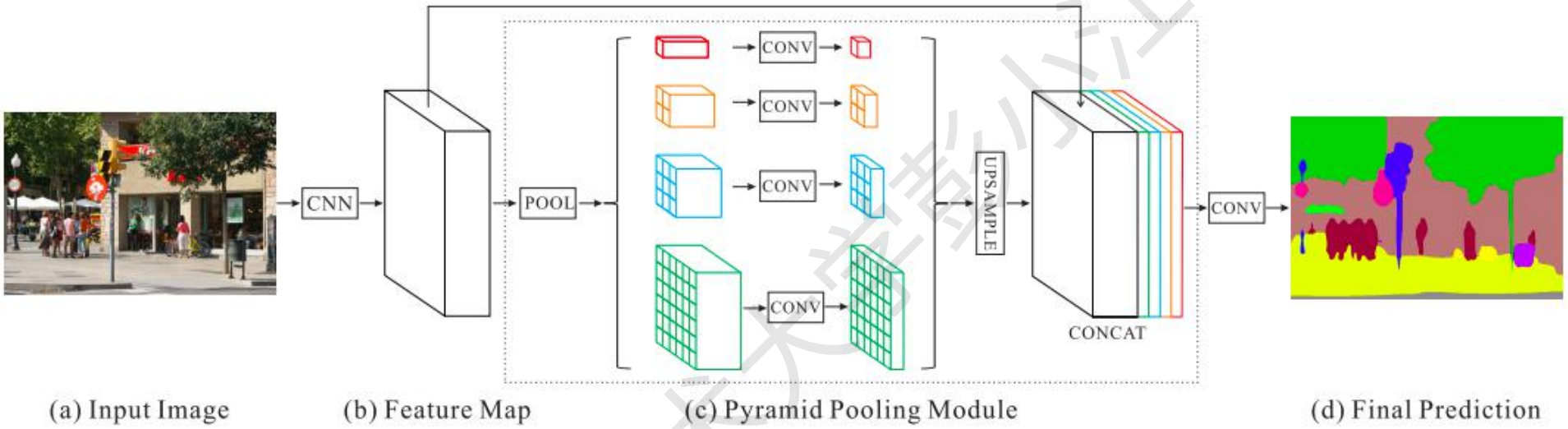
DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs（2016/6/2）

- 多孔卷积



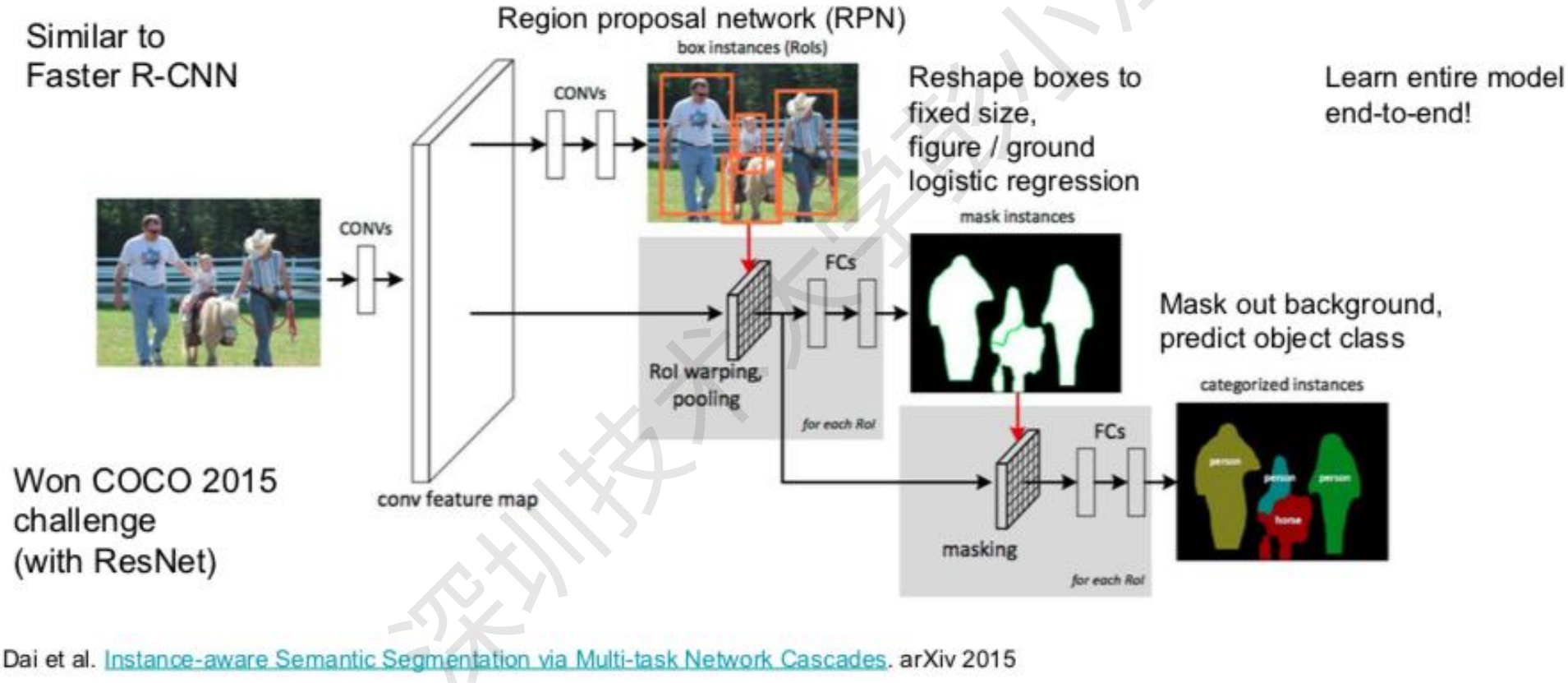
Multi-Scale Context Aggregation by Dilated Convolutions（2015/11/23）

PSPNet

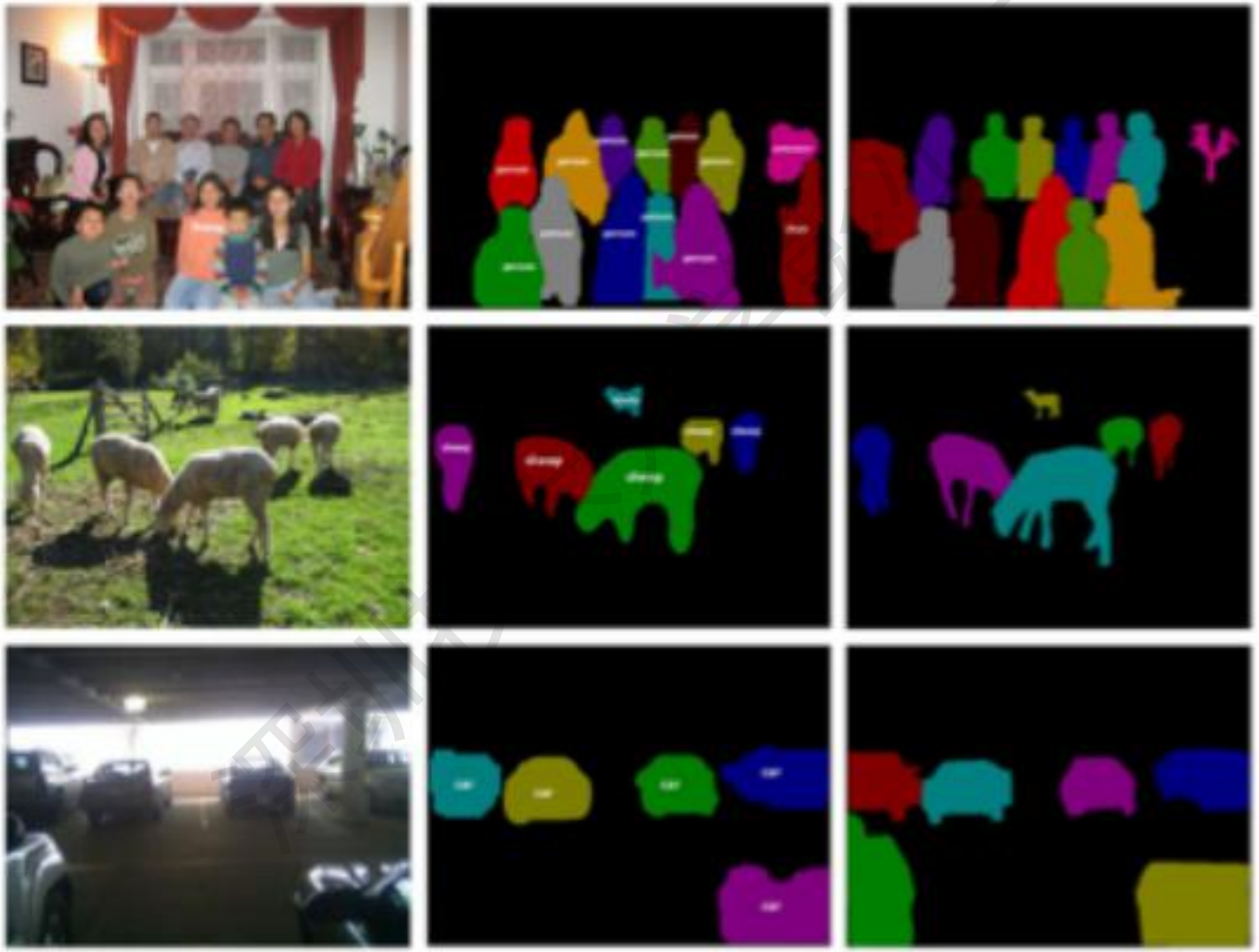


Pyramid Scene Parsing Network（2017）

Instance segmentation(实例分割)



Dai et al. Instance-aware Semantic Segmentation via Multi-task Network Cascades. arXiv 2015



Mask RCNN

- 在Faster R-CNN基础上加了一个分支

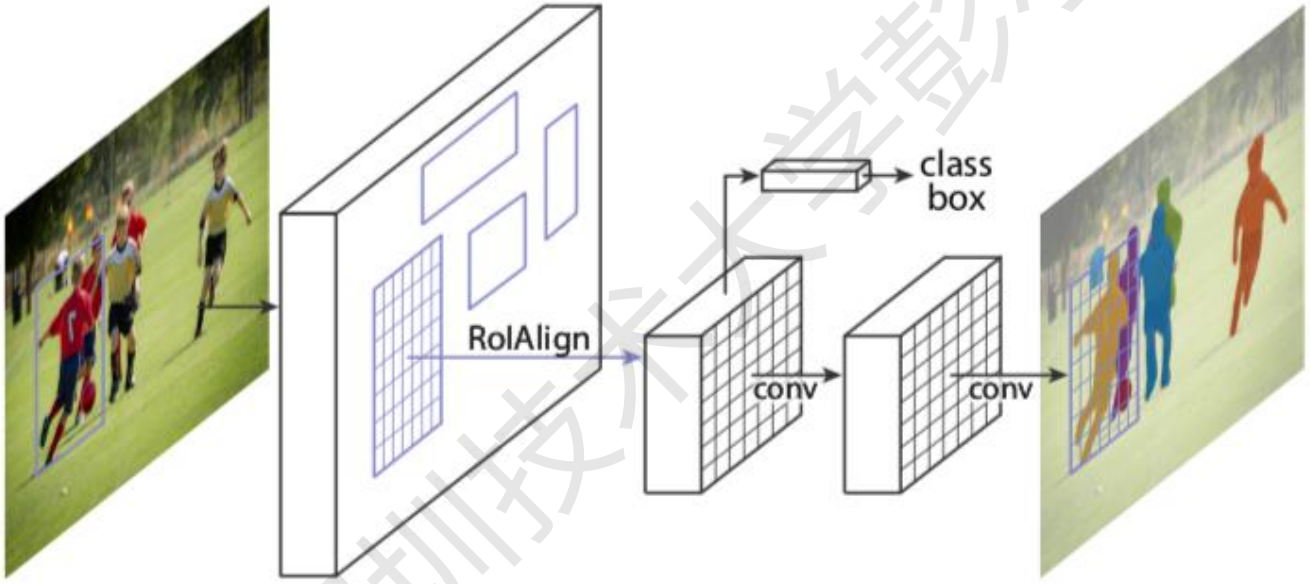


Figure 1. The **Mask R-CNN** framework for instance segmentation.

Mask R-CNN (He et al., 2017)

参考文献

1. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. CoRR, abs/1312.6229, 2013.
2. J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. International Journal of Computer Vision, 104(2):154–171, 2013.
3. Long et al. Fully Convolutional Networks for Semantic Segmentation. CVPR 2015