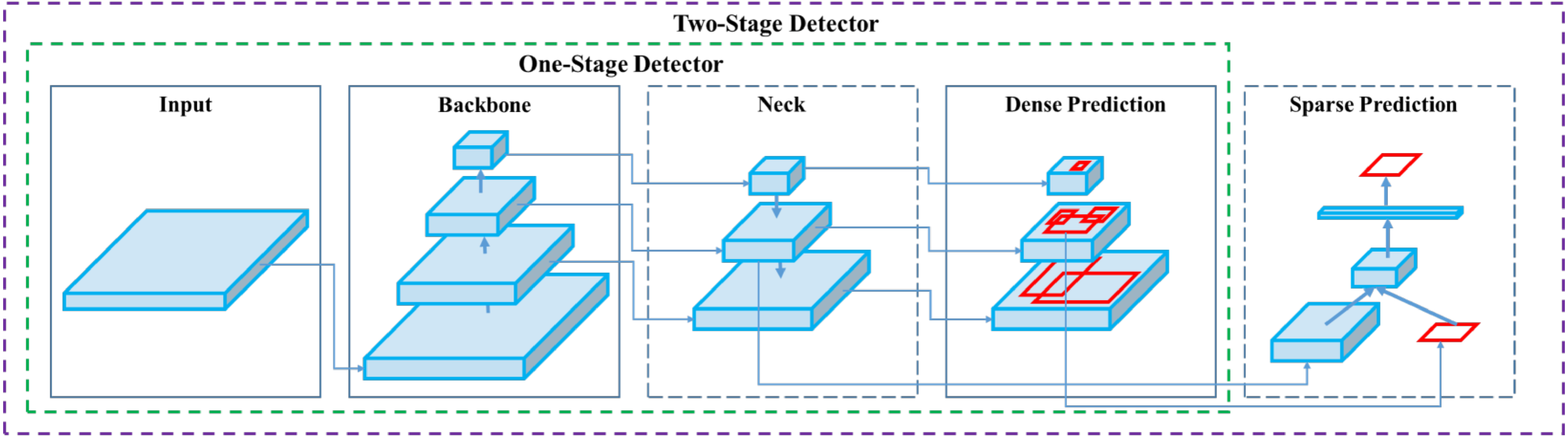
YOLOv5网络架构、组件和Loss函数



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

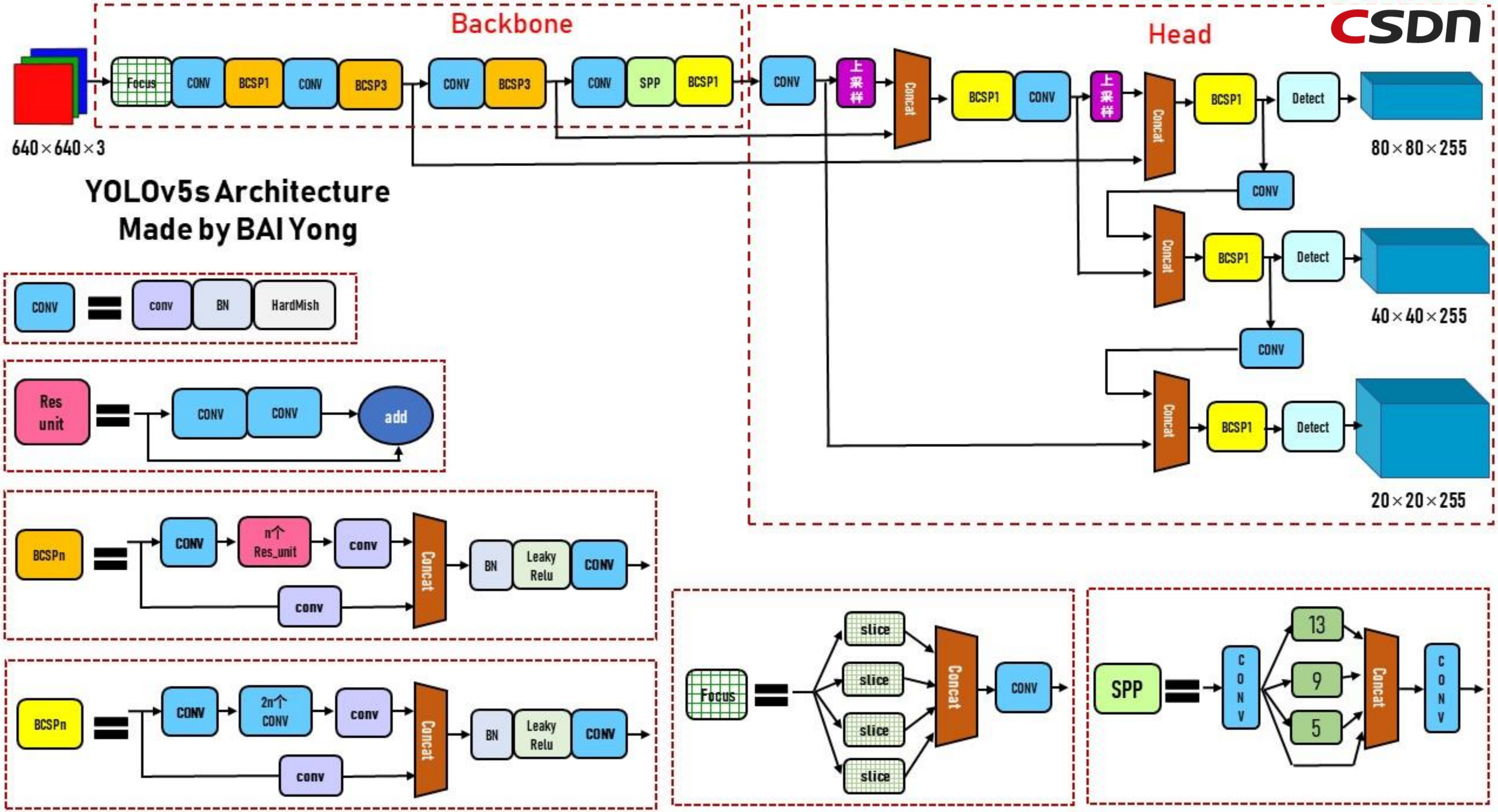
Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

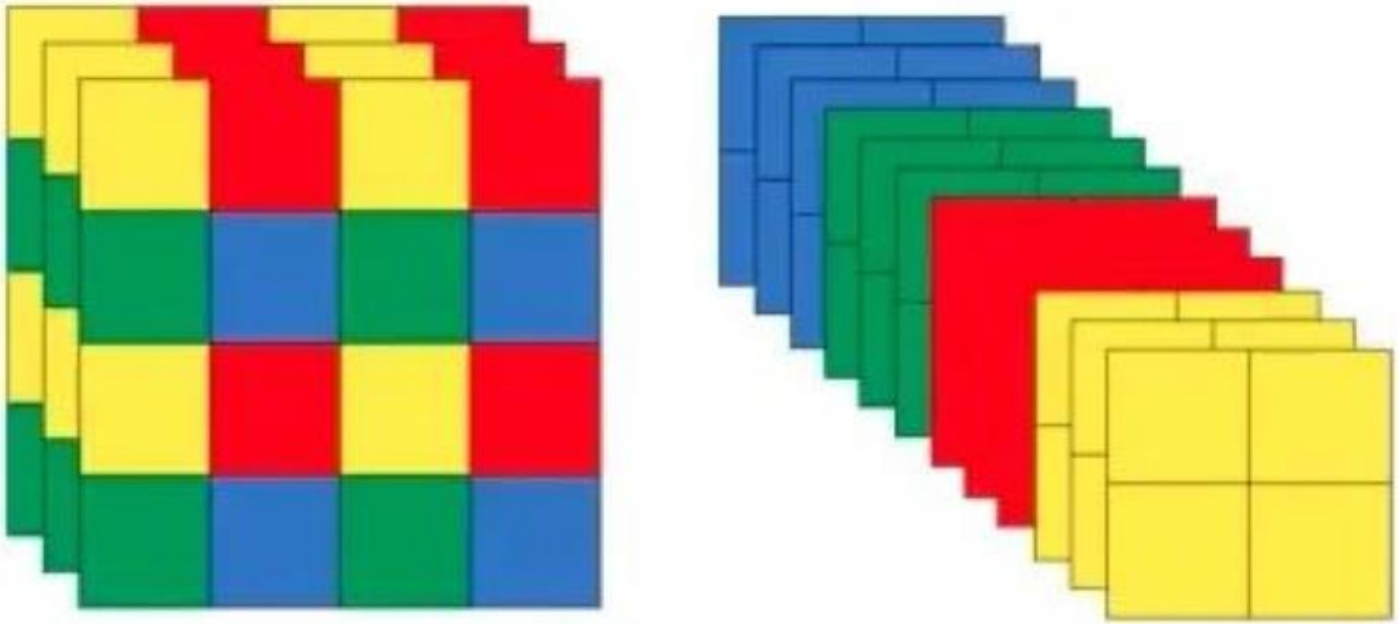
Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

YOLOv5 包括

- **Backbone**: Focus, BottleneckCSP
- **Head:** Neck(SPP, PANet) + YOLOv3/v4 Head


Backbone
Head
CSDN
640×640×3
Focus
CONV
BCSP1
CONV
BCSP3
CONV
BCSP3
CONV
SPP
BCSP1
CONV
上采样
Concat
BCSP1
CONV
上采样
Concat
BCSP1
Detect
80×80×255
CONV
Concat
BCSP1
Detect
40×40×255
CONV
Concat
BCSP1
Detect
20×20×255
YOLOv5s Architecture
Made by BAI Yong
CONV = conv BN HardMish
Res unit = CONV CONV add
BCSPn = CONV n个 Res_unit conv conv Concat BN Leaky Relu CONV
BCSPn = CONV 2n个 CONV conv conv Concat BN Leaky Relu CONV
Focus = slice slice slice slice Concat CONV
SPP = CONV 13 9 5 Concat CONV

Focus



CSPNet

CSP （Cross Stage Partial Network） 跨阶段局部网络

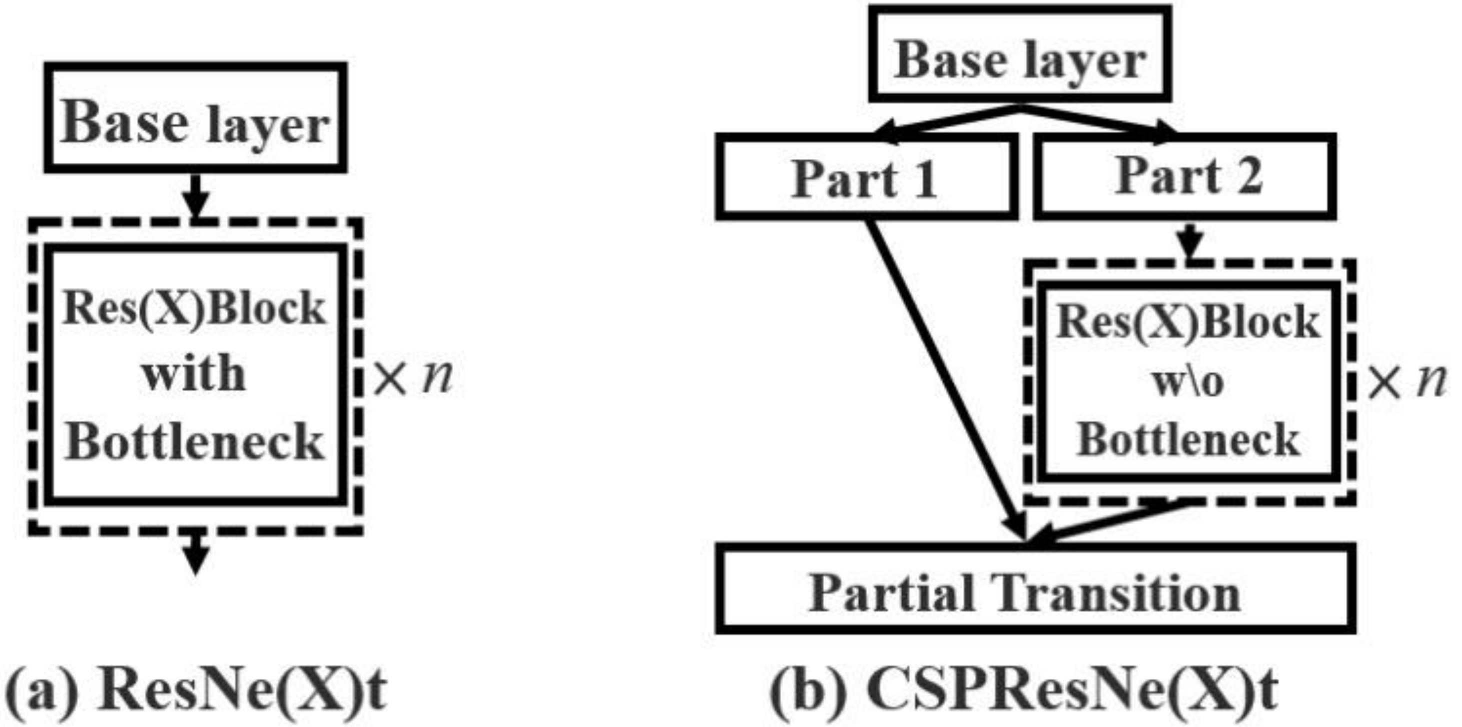
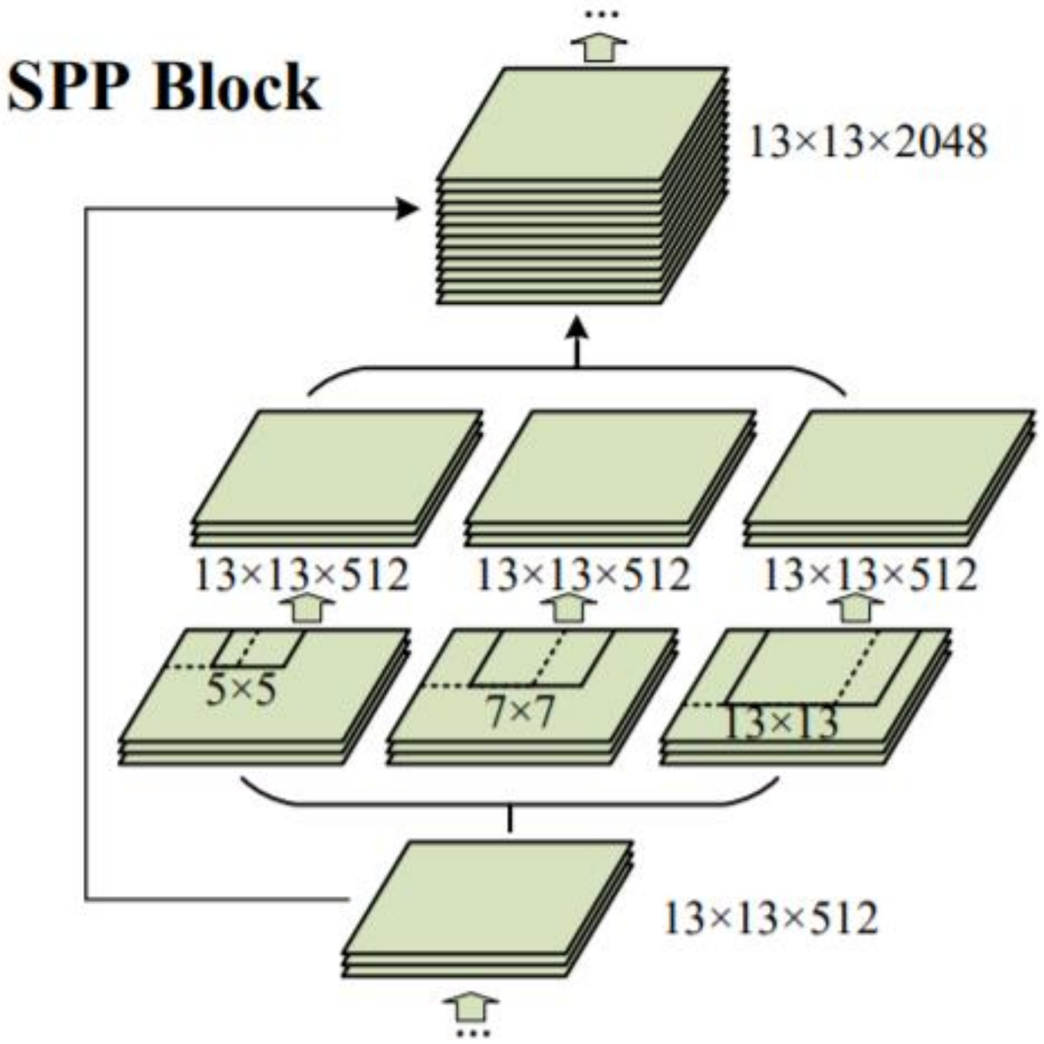


Figure 5: Applying CSPNet to ResNe(X)t.

SPP (Spatial Pyramid Pooling)

416×416输入



Add the SPP block over the CSP, since it significantly increases the receptive field, separates out the most significant context features and causes almost no reduction of the network operation speed.

PANet

Path-Aggregation Network 路径聚合网络

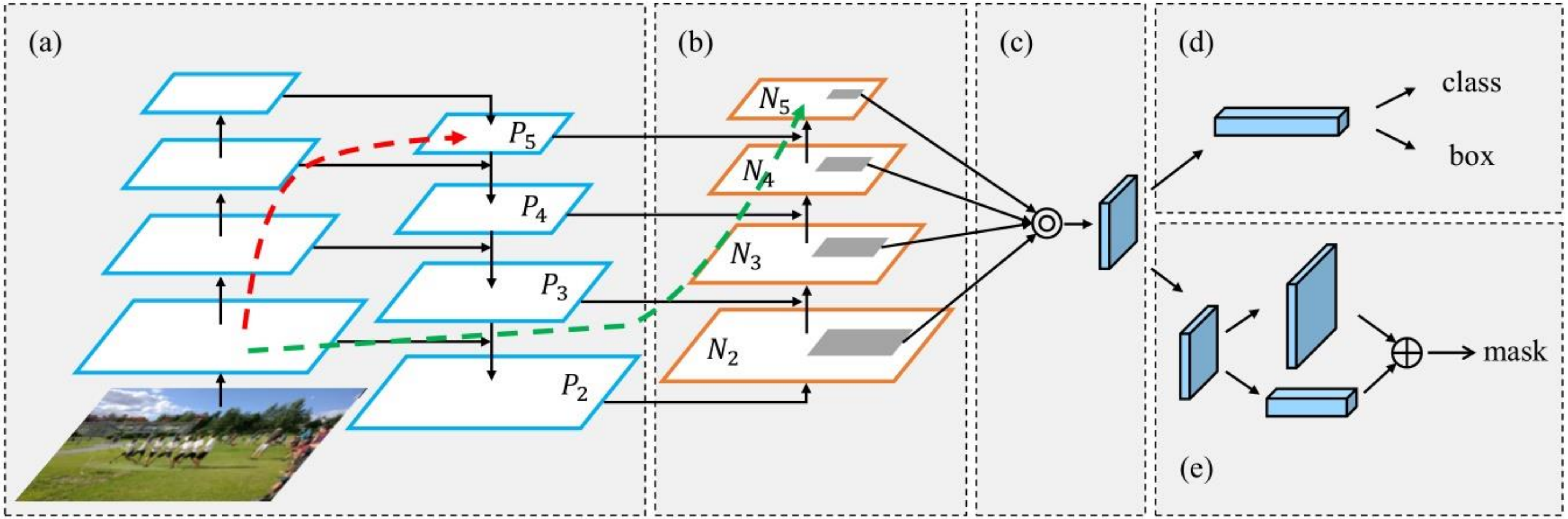


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

Hard Swish激活函数

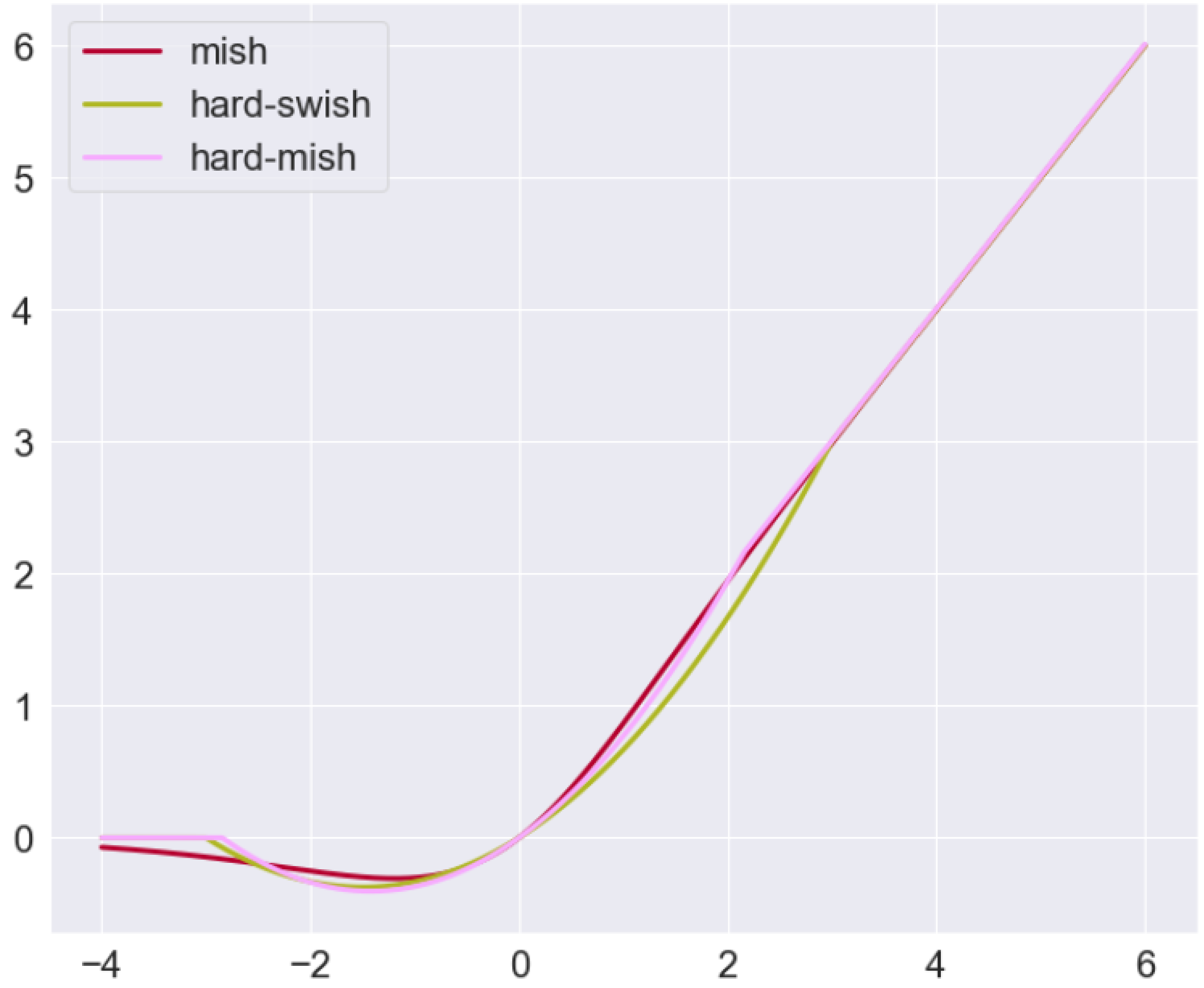
CLASS `torch.nn.Hardswish` [SOURCE]

Applies the hardswish function, element-wise, as described in the paper:

Searching for MobileNetV3.

$$\text{Hardswish}(x) = \begin{cases} 0 & \text{if } x \le -3, \\ x & \text{if } x \ge +3, \\ x \cdot (x+3)/6 & \text{otherwise} \end{cases}$$

$$\text{h-swish}[x] = x \frac{\text{ReLU6}(x+3)}{6}$$



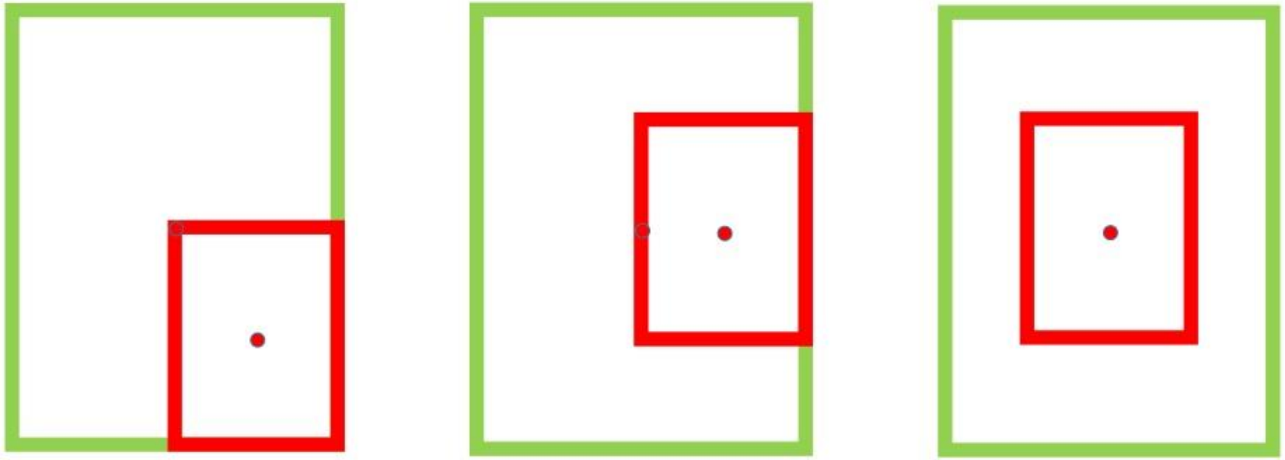
mish
hard-swish
hard-mish
6
5
4
3
2
1
0
−4
−2
0
2
4
6

Mosaic data augmentation

损失函数

- YOLO V5使用二元交叉熵损失函数计算类别概率和目标置信度得分的损失。
- YOLO V5使用 CIOU Loss作为bounding box的损失。

DIoU loss



$\mathcal{L}_{IoU} = 0.75$	$\mathcal{L}_{IoU} = 0.75$	$\mathcal{L}_{IoU} = 0.75$
$\mathcal{L}_{GIoU} = 0.75$	$\mathcal{L}_{GIoU} = 0.75$	$\mathcal{L}_{GIoU} = 0.75$
$\mathcal{L}_{DIoU} = 0.81$	$\mathcal{L}_{DIoU} = 0.77$	$\mathcal{L}_{DIoU} = 0.75$

Figure 2: GIoU loss degrades to IoU loss for these cases, while our DIoU loss is still distinguishable. Green and red denote target box and predicted box respectively.

- A Distance-IoU loss, i.e., DIoU loss, is proposed for bounding box regression, which has faster convergence than IoU and GIoU losses.
- A Complete IoU loss, i.e., CIoU loss, is further proposed by considering three geometric measures, i.e., overlap area, central point distance and aspect ratio, which better describes the regression of rectangular boxes.