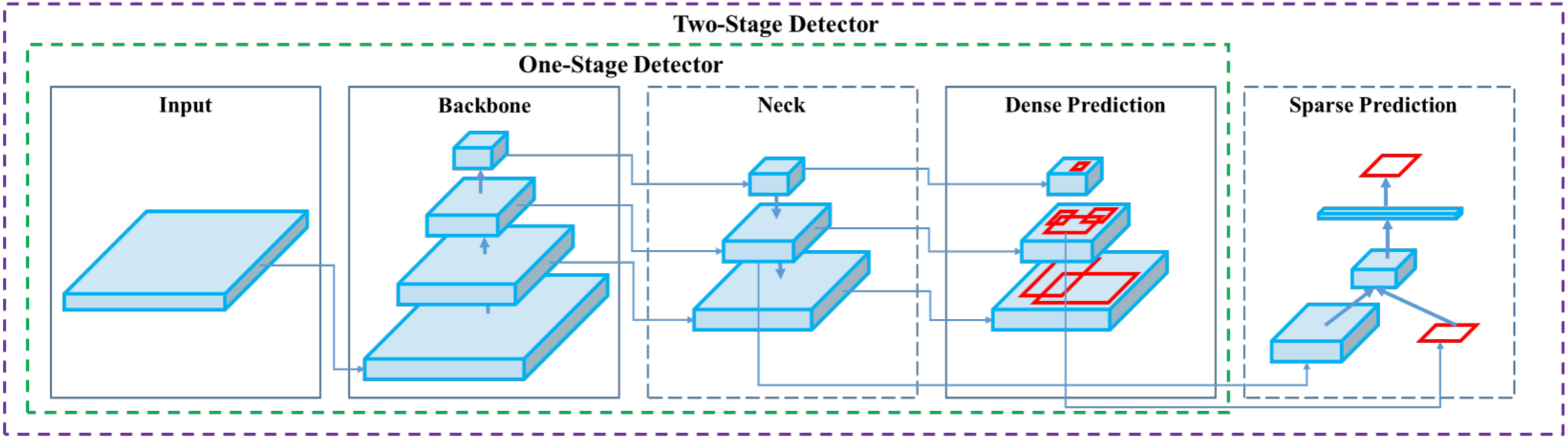
YOLOv5网络架构与组件

CSDN



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, SPP
- Head: PANet + Detect (YOLOv3/v4 Head)

网络可视化工具: netron

Netron

Netron is a visualizer for neural network, deep learning and machine learning models. It runs on macOS, Windows, Linux and in the browser.

GitHub

在线版本： 浏览器中输入链接：https://lutzroeder.github.io/netron/

netron官方的Github链接: https://github.com/lutzroeder/netron

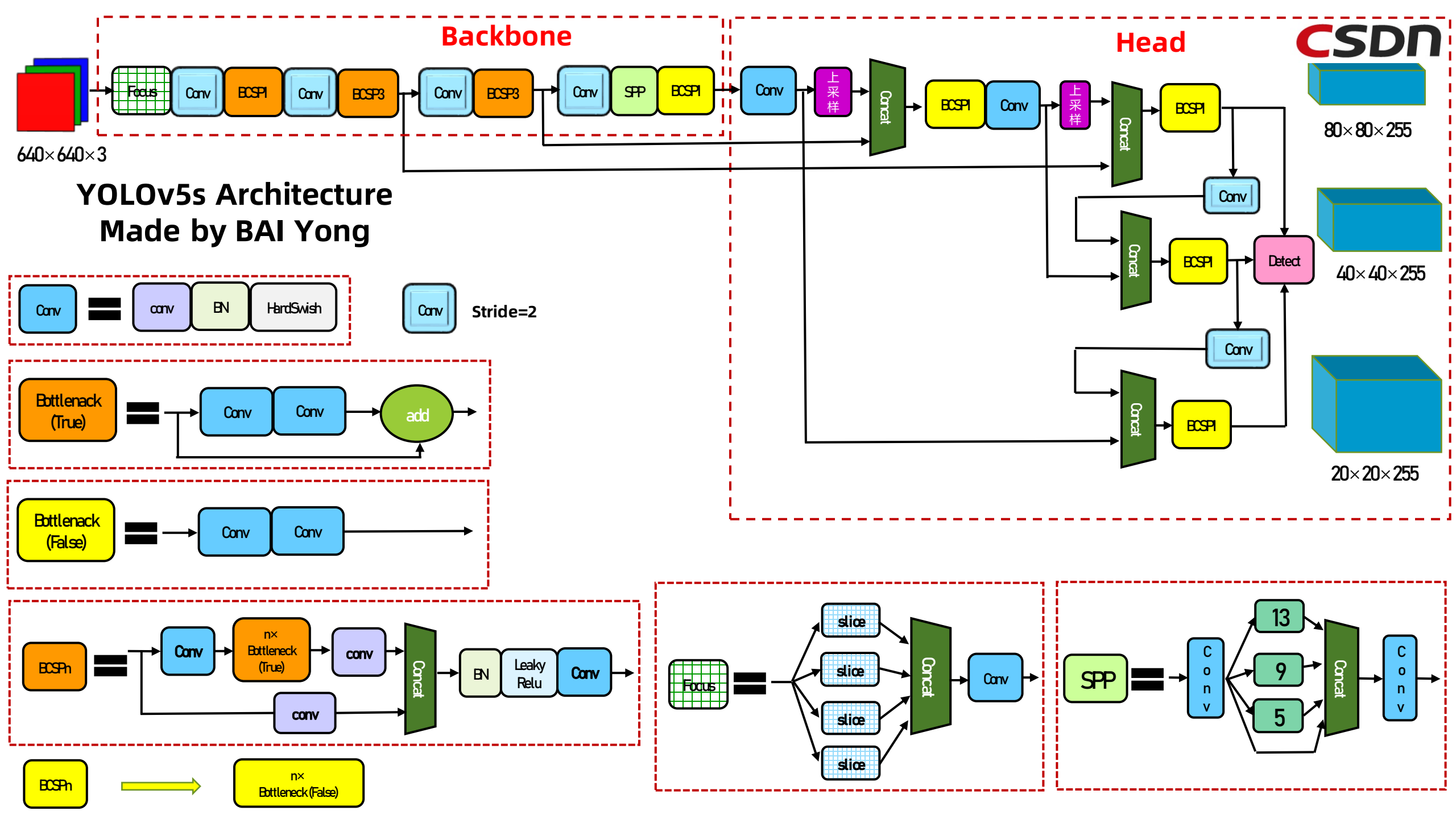
netron对pt格式的权重文件兼容性不好，直接使用netron工具打开， 无法显示整个网络。

可使用YOLOv5代码中**models/export.py**脚本将pt权重文件转换为onnx格式，再用netron工具打开，就可以看YOLOv5网络的整体架构。

导出ONNX文件

```
pip install onnx>=1.7.0  -i https://pypi.tuna.tsinghua.edu.cn/simple  # for ONNX export
pip install coremltools==4.0 -i https://pypi.tuna.tsinghua.edu.cn/simple  # for CoreML export
```

```
python models/export.py --weights weights/yolov5s.pt --img 640 --batch 1
```


Backbone
Head
CSDN
640×640×3
Focus
Conv
BCSP1
BCSP3
SPP
上采样
Concat
Detect
80×80×255
40×40×255
20×20×255
YOLOv5s Architecture
Made by BAI Yong
conv
BN
HardSwish
Stride=2
Bottlenack (True)
add
Bottlenack (False)
BCSPn
n× Bottleneck (True)
Leaky Relu
n× Bottleneck (False)
slice
SPP
13
9
5

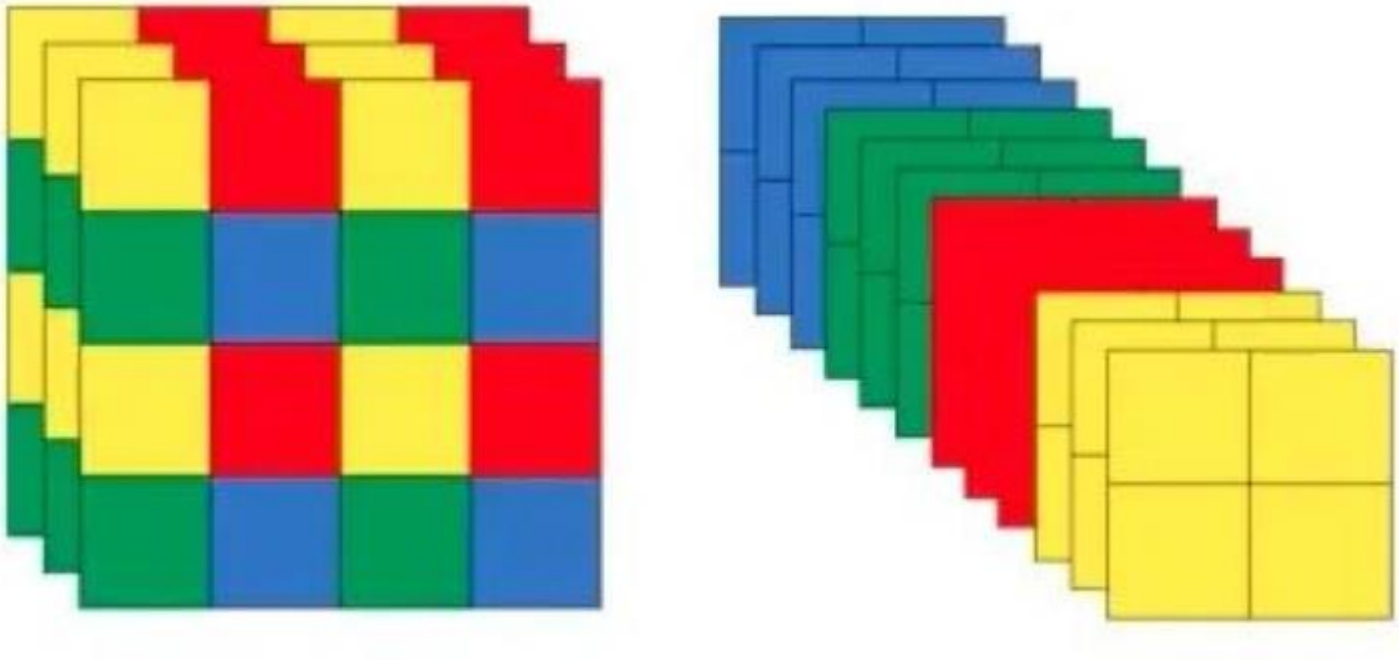
灵活配置不同复杂度的模型

应用类似EfficientNet的channel和layer控制因子

	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
depth_multiple	0.33	0.67	1.0	1.33
width_multiple	0.50	0.75	1.0	1.25
BottleneckCSP数 BCSPn(True)	1,3,3	2,6,6	3,9,9	4,12,12
BottleneckCSP数 BCSPn(Flase)	1	2	3	4
Conv卷积核数量	32,64,128,256,512	48,96,192,384,768	64,128,256,512,1024	80,160,320,640,1280

YOLOv5的四种网络结构是depth_multiple和width_multiple两个参数，来进行控制网络的**深度**和**宽度**。其中**depth_multiple**控制网络的**深度**（BottleneckCSP数），**width_multiple**控制网络的**宽度**（卷积核数量）。

Focus



把数据切分为4份，每份数据都是相当于2倍下采样得到的，然后在channel维度进行拼接，最后进行卷积操作。

You can do a small experiment like this:

Create a toy tensor x

```
tensor([[[[11, 12, 13, 14],
          [21, 22, 23, 24],
          [31, 32, 33, 34],
          [41, 42, 43, 44]]]])
```

Pass x to Focus layer and the result is

```
tensor([[[[11, 13],
          [31, 33]],

         [[21, 23],
          [41, 43]],

         [[12, 14],
          [32, 34]],

         [[22, 24],
          [42, 44]]]])
```

Focus() module is designed for FLOPS reduction and speed increase, not mAP increase.

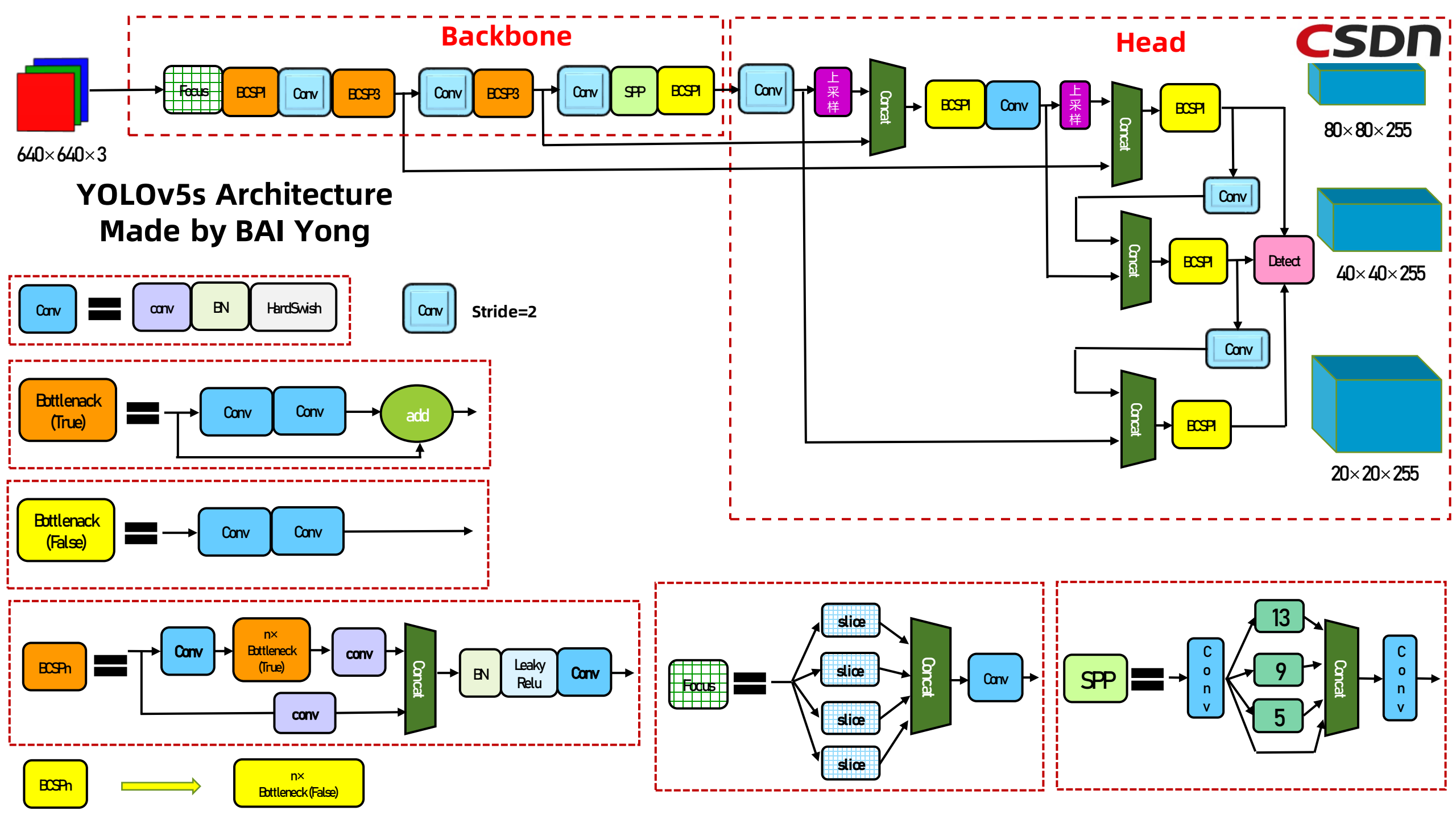
In YOLOv5 the author wants to reduce the cost of Conv2d computation and actually using tensor reshaping to reduce space (resolution) and increase the depth (number of channels).

Input will be transformed likes this [b, c, h, w] -> [b, c*4, h//2, w//2]

把宽度w和高度h的信息整合到c空间中

以Yolov5s的结构为例，原始640*640*3的图像输入Focus结构，采用切片操作，先变成320*320*12的特征图，再经过一次32个卷积核的卷积操作，最终变成320*320*32的特征图。

而yolov5m的Focus结构中的卷积操作使用了48个卷积核，因此Focus结构后的特征图变成320*320*48。yolov5l，yolov5x也是同样的原理


Backbone
Head
CSDN
640×640×3
Focus
BCSP1
Conv
BCSP3
SPP
上采样
Concat
Detect
80×80×255
40×40×255
20×20×255
YOLOv5s Architecture
Made by BAI Yong
conv
BN
HardSwish
Stride=2
Bottlenack (True)
add
Bottlenack (False)
BCSPn
n× Bottleneck (True)
Leaky Relu
n× Bottleneck (False)
slice
SPP
13
9
5

CSPNet

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

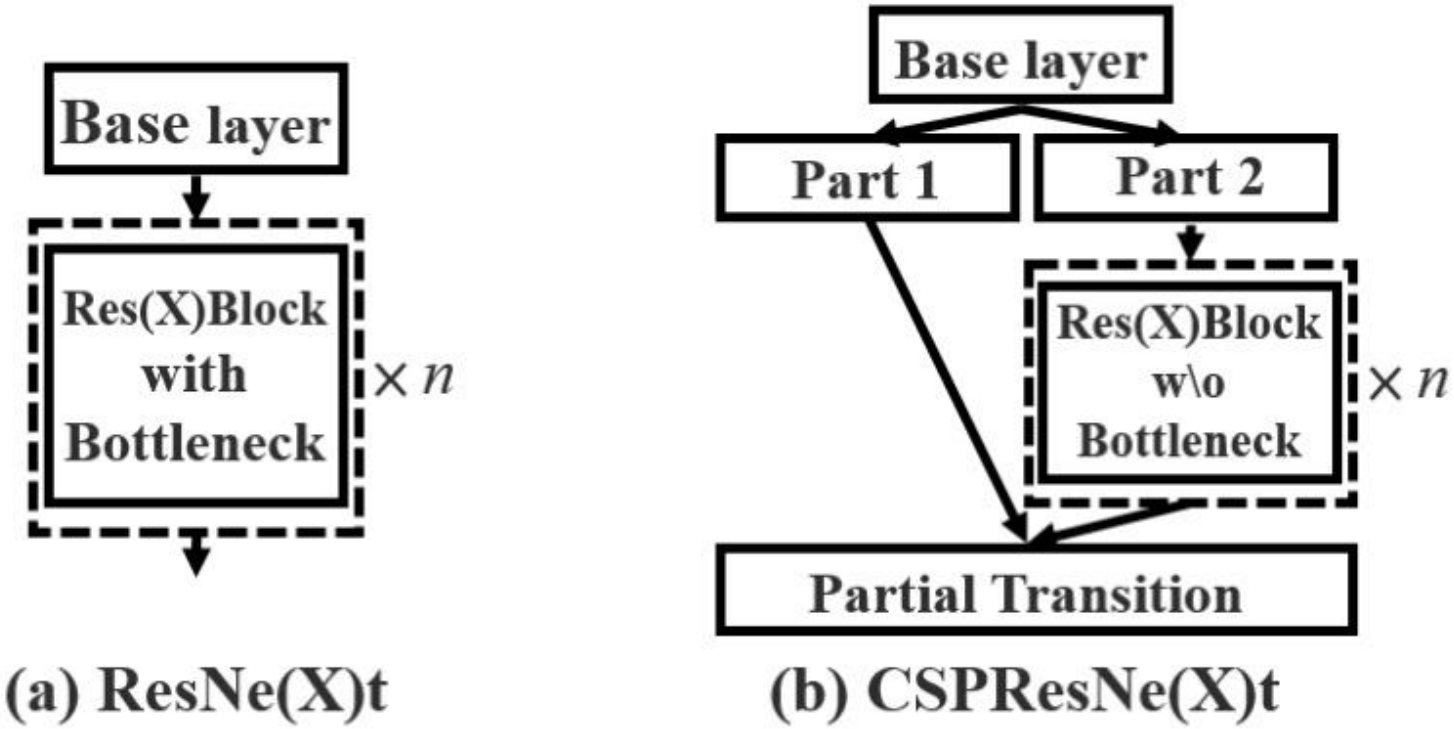
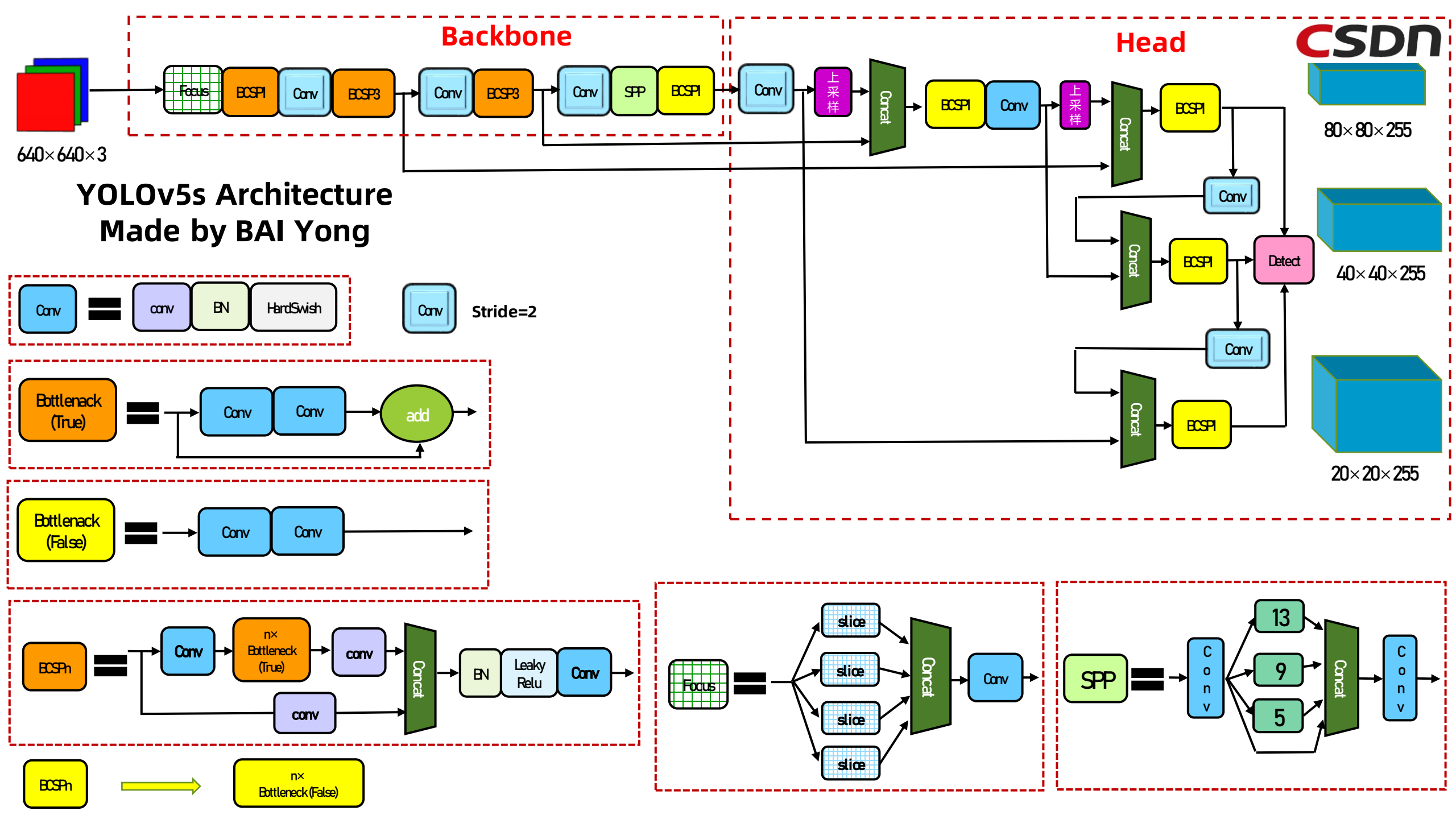


Figure 5: Applying CSPNet to ResNe(X)t.

doubts regarding the CSPNet implementation：

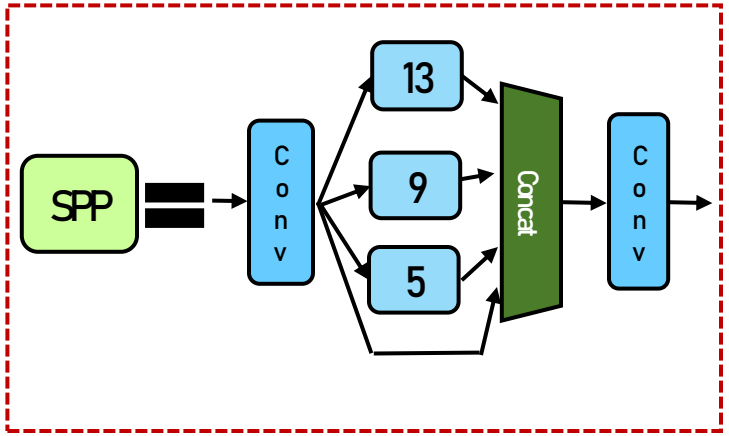
问题： As per the paper the https://openaccess.thecvf.com/content_CVPRW_2020/papers/w28/Wang_CSPNet_A_New_Backbone_That_Can_Enhance_Learning_Capability_of_CVPRW_2020_paper.pdf , the input should be split into 2 parts which should be processed through 2 branches independently. But in your implementation, Both branches are taking the same input and without any spliting.

回答：yes. that is correct. The inputs are not split, they are used in two places here, which I believe aligns with the actual CSPNet implementation.

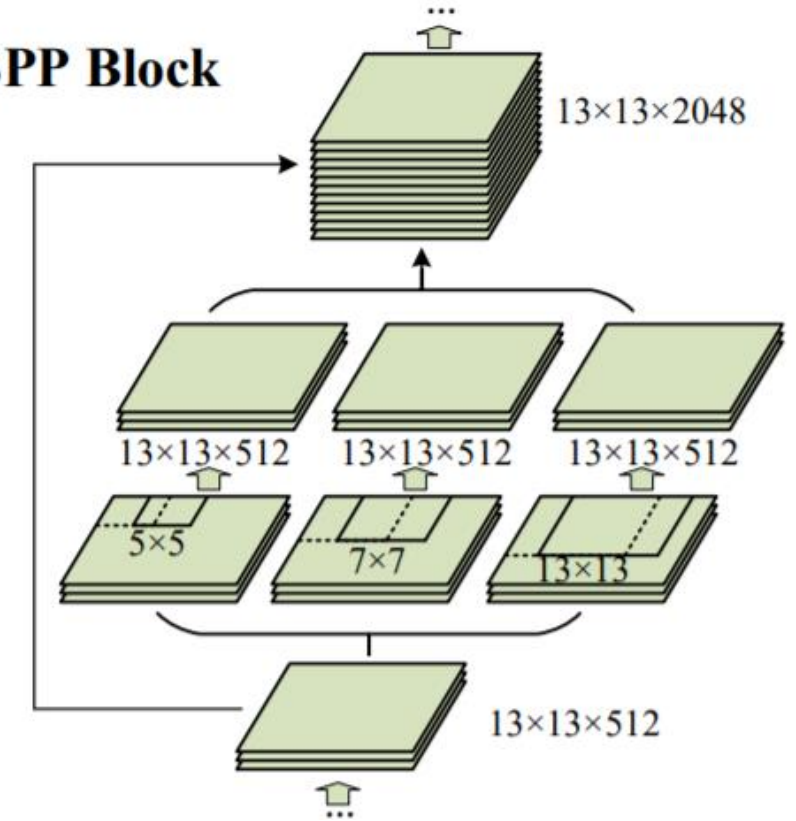

Backbone
Head
CSDN
640×640×3
Focus
BCSP1
Conv
BCSP3
SPP
上采样
Concat
Detect
80×80×255
40×40×255
20×20×255
YOLOv5s Architecture
Made by BAI Yong
conv
BN
HardSwish
Stride=2
Bottlenack (True)
add
Bottlenack (False)
BCSPn
n× Bottleneck (True)
Leaky Relu
n× Bottleneck (False)
slice
SPP
13
9
5

SPP (Spatial Pyramid Pooling)

空间金字塔池化



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

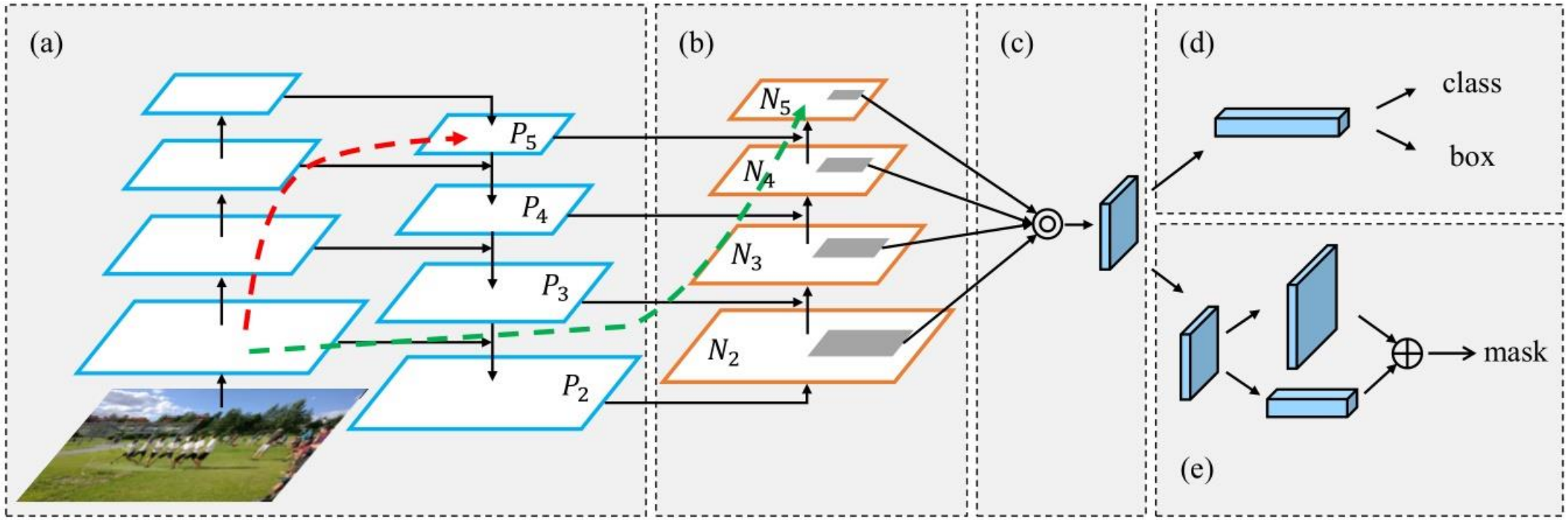
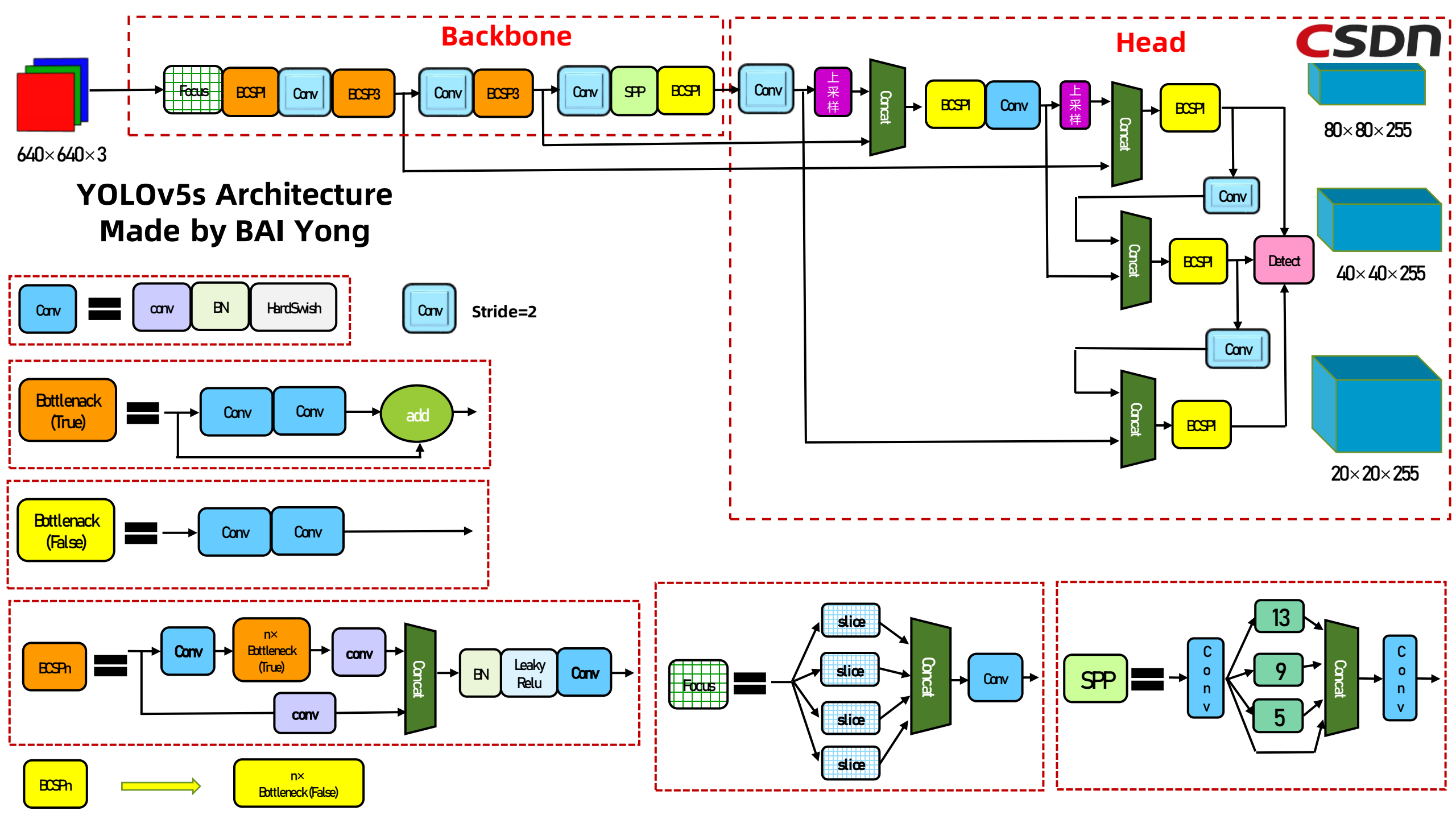


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.


Backbone
Head
CSDN
640×640×3
Focus
BCSP1
Conv
BCSP3
SPP
上采样
Concat
Detect
80×80×255
40×40×255
20×20×255
YOLOv5s Architecture
Made by BAI Yong
conv
BN
HardSwish
Stride=2
Bottlenack (True)
add
Bottlenack (False)
BCSPn
n× Bottleneck (True)
Leaky Relu
n× Bottleneck (False)
slice
SPP
13
9
5