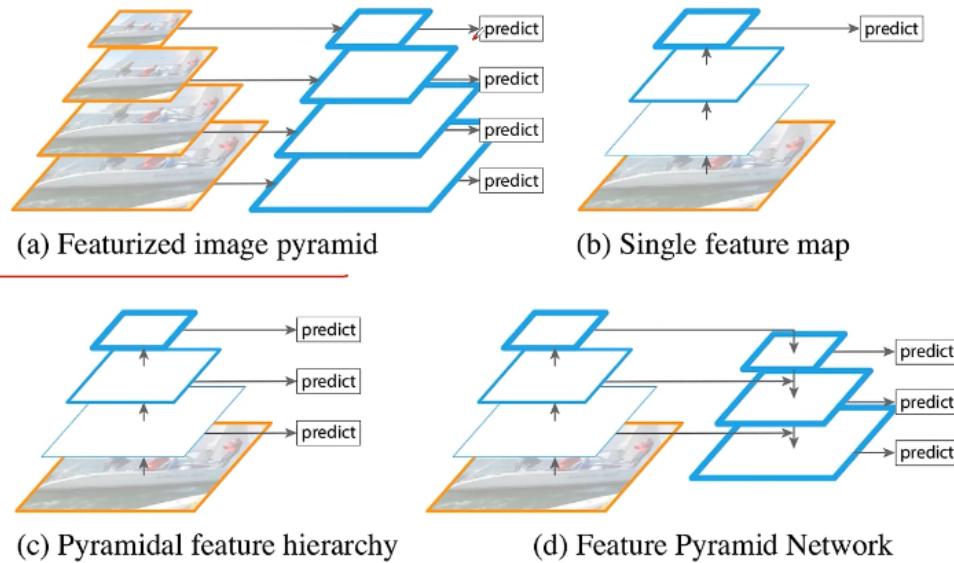


Others

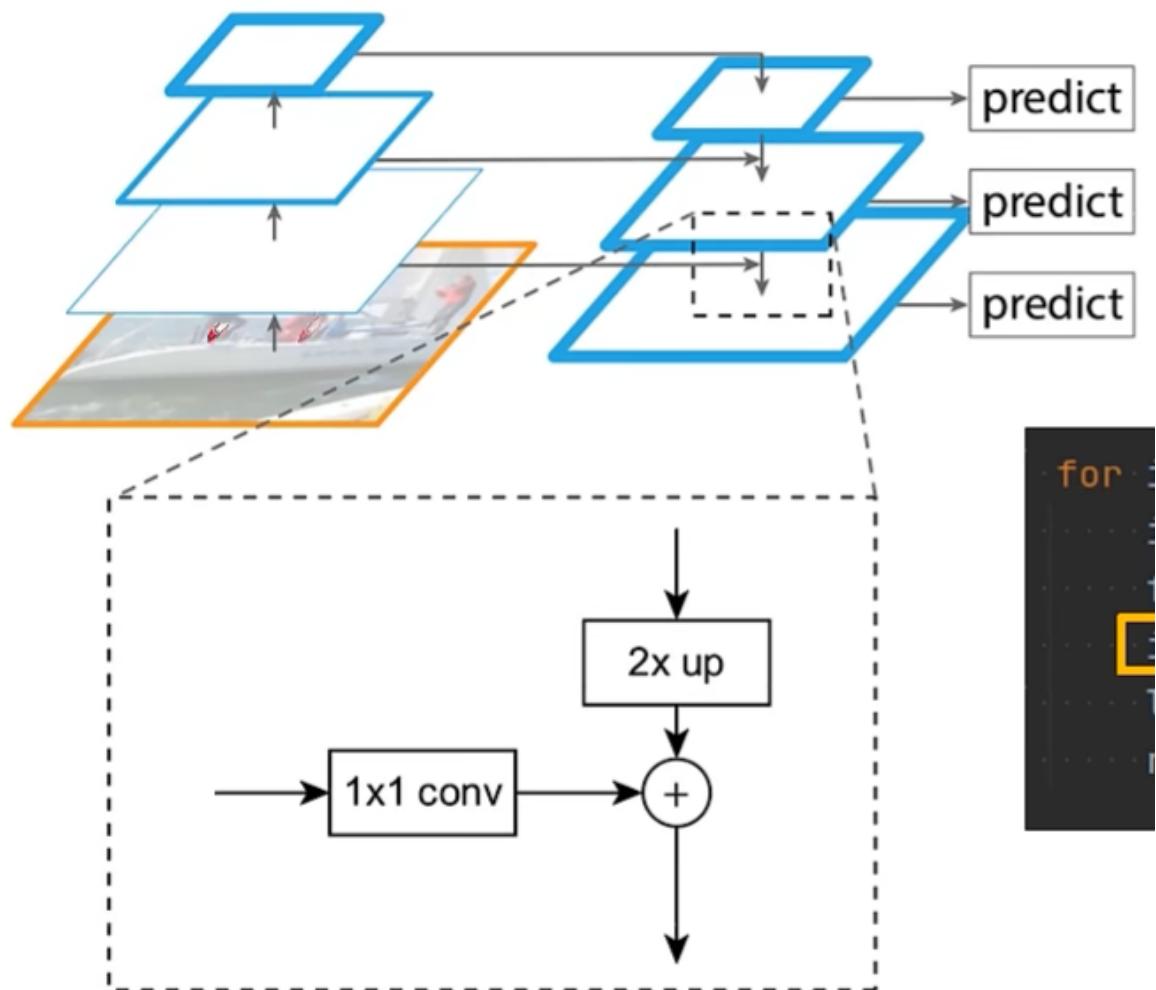
BN(Batch Normalization)

FPN(Feature Pyramid Networks)



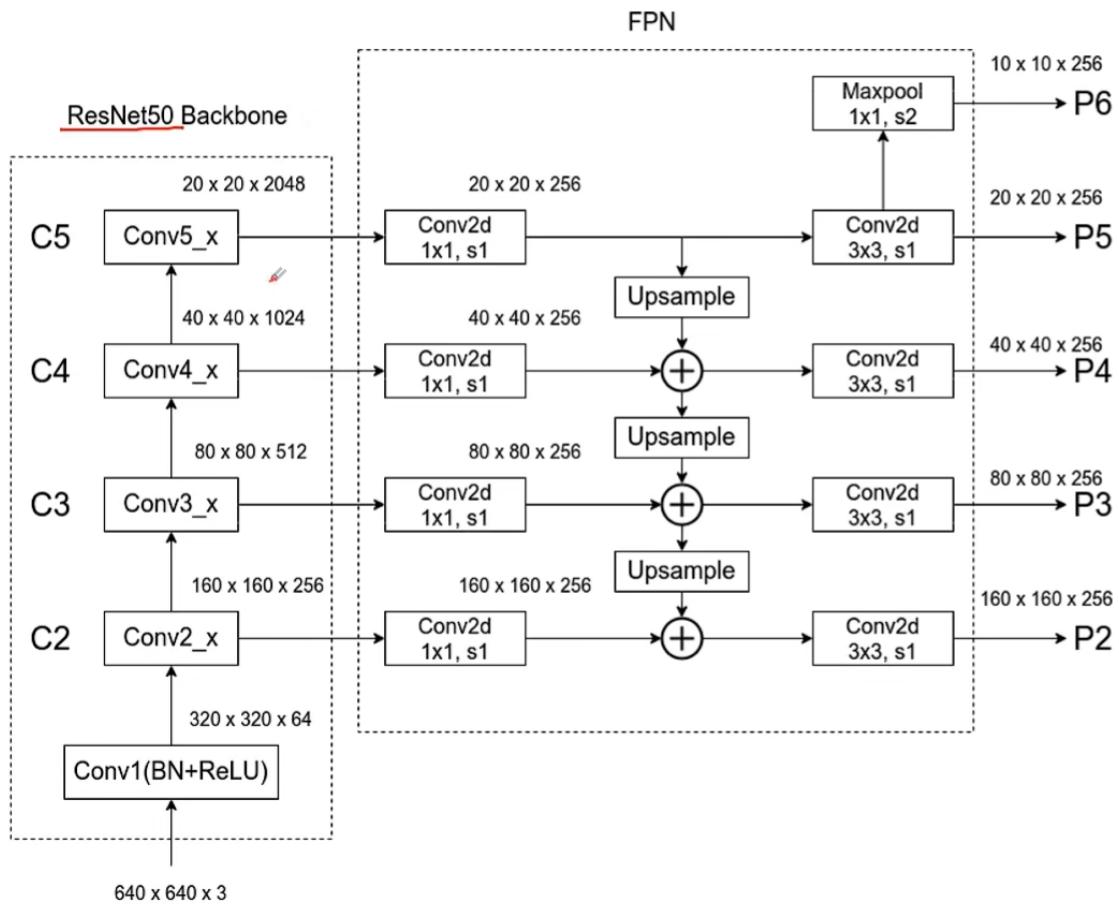
a):传统图像金字塔

others: 基于backbone提取的feature map



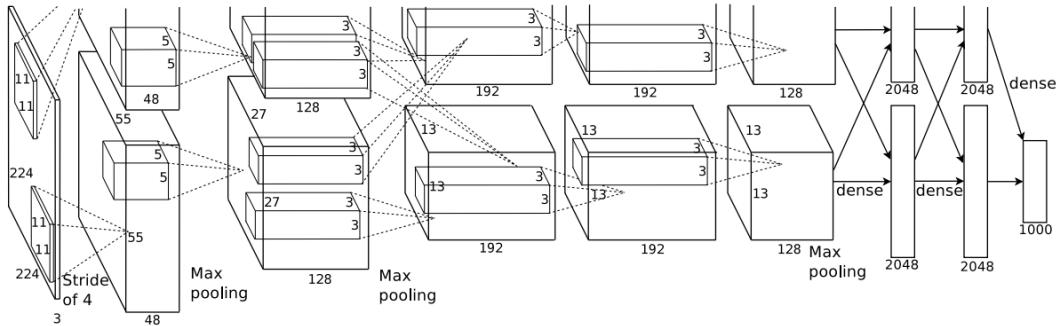
1*1conv的作用：主要是使每一层channel一致

上采样：临近插值算法



- 在FasterRcnn中，RPN网络是通过P2-P6生成，生成的RP再映射到P2-P5的feature map中进行Fast rcnn网络部分的预测
- 可在不同的预测特征层上（P2-P6）预测不同尺度的目标，如P2为底层特征层，会保留更加底层的细节信息，更适合预测小目标。所以，在生成anchor时，在P2-P6层分别生成 $\{32^2, 64^2, 128^2, 256^2, 512^2\}$ 的三种比例的anchor{1: 1, 1: 2, 2: 1}

AlexNet



1. ReLU
2. Training on Multiple GPUs
3. Local Response Normalization (LRN层，现已不使用)
4. Overlapping Pooling (重叠的池化，现已不使用)
5. Data Augmentation (image translations and horizontal reflections，颜色变化)
6. Dropout

训练阶段 每一个 batch
随机掐死一半的神经元
(将神经元输出设为0)
阻断该神经元的前向-反向传播
预测阶段 保留所有神经元
预测结果乘 0.5

Dropout 为什么能减少过拟合?

- A. 模型集成 $p=0.5$ 意味着 2^n 个共享权重的潜在网络
- B. 记忆随机抹去 不再死记硬背
- C. Dropout 减少神经元之间的联合依赖性
每个神经元都被逼着独当一面
- D. 有性繁殖 每个基因片段都要与来自另一个随机个体的
基因片段协同工作
- E. 数据增强 总可以找到一个图片 使神经网络中间层结果
与 Dropout 后相同 相当于增加了这张图片到数据集中
- F. 稀疏性

YOLO家族

YOLOV1

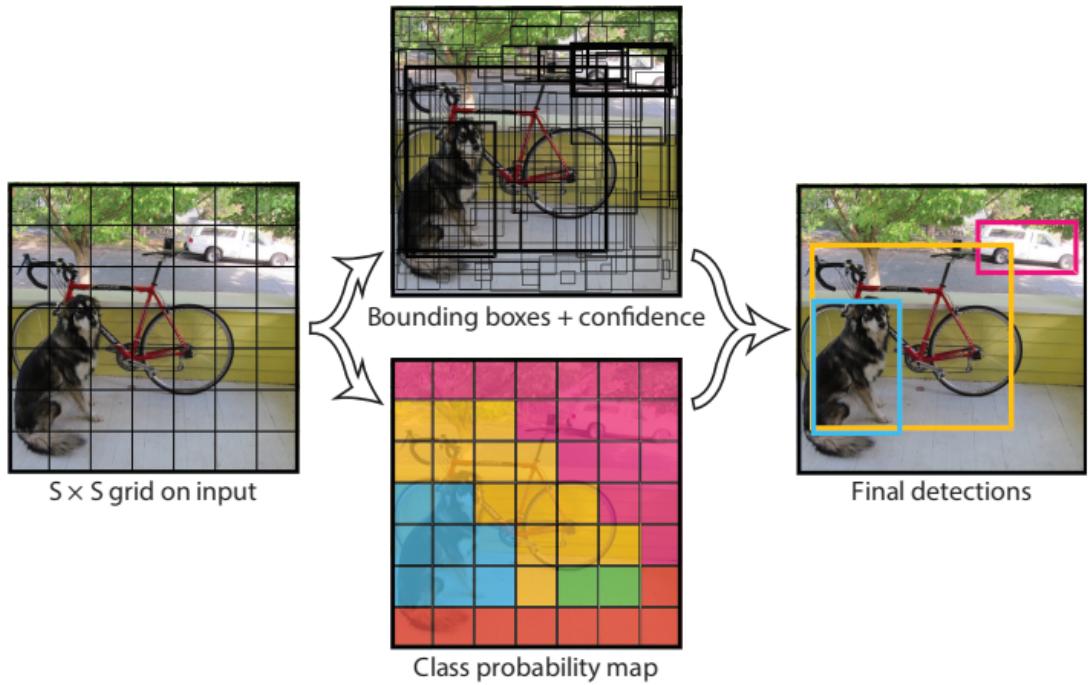
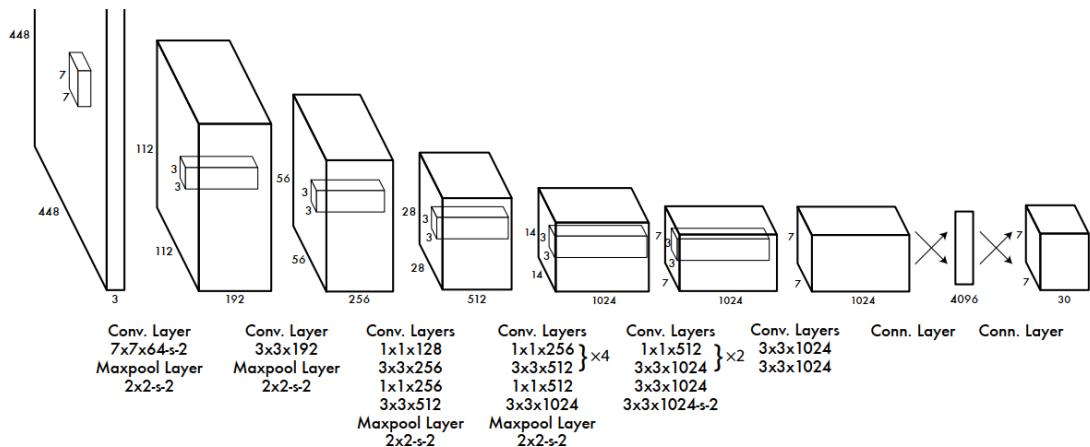


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.



预测阶段

Input: $448 \times 448 \times 3$

将图像大小分为 $7 \times 7 = 49$ 个部分 (grid cell)

每个 grid cell 包含两个 bounding box

所以最后的 output 为 $7 \times 7 \times 30$ 的张量

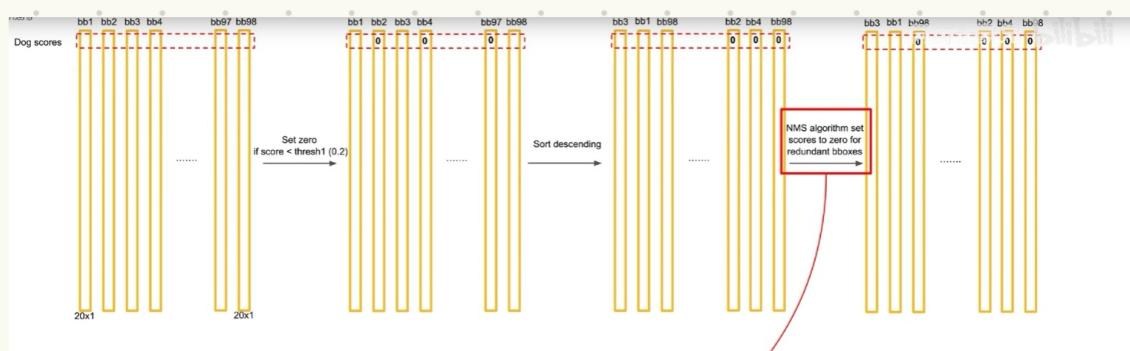
其中 30: 1-5 为第一个 bbox 的 P(置信度), X, Y, Width, Height

6-10 为第二个 bbox 的 P(置信度), X, Y, Width, Height

11-30 为某一个类别别的条件概率 (Pascal VOC 有 20 个类别, 所以此处为 20 维)

所以每一个 bbox 对应一个 20×1 的概率向量 (20×1 的条件概率向量 $\times P = 全概率$)

共有 $7 \times 7 \times 2 = 98$ 个 20×1 的概率向量



选择 20 维中的第一维数 (即该为 dog 的概率)

1. 设定一个概率阈值, 低于的都置 0

2. 不重叠放在前面, 为重的放在后面 (从高到低排序)

3. NMS

NMS: 非极大值抑制 (可检测多目标)

1. 将所有框与最大的框计算 Iou, 大于阈值的, 从重叠到一个维度, 直接剔除

2. 将所有框与第二大的

3. 以此类推, 最后剩下的框为检测框

YOLO V1 损失函数

每一项都是平方和误差 将目标检测问题当作回归问题

批注: 同济子豪兄

loss function:

$$\begin{aligned}
 & \text{负责检测物体的bbox} \quad \text{中心点定位误差} \\
 & \lambda_{\text{coord}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
 & \text{负责检测物体的bbox} \quad \text{宽高定位误差} \\
 & \text{宽高号能使小框对误差更敏感} \\
 & + \lambda_{\text{coord}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
 & \text{负责检测物体的bbox} \quad \text{Confidence 误差} \\
 & \text{Confidence 回归误差} \\
 & + \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
 & \text{不负责检测物体的bbox} \quad \text{Confidence 误差} \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
 & \text{负责检测物体的grid cell} \quad \text{分类误差} \\
 & \text{类别预测误差} + \sum_{i=0}^S \mathbb{1}_{ij}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (3)
 \end{aligned}$$

注释:

- $\mathbb{1}_{ij}^{\text{obj}}$: 第 i 个 grid cell 是否包含物体，即是否有 ground truth 的中心点落在该 grid cell 中。若有则为 1，否则为 0。
- $\mathbb{1}_{ij}^{\text{noobj}}$: 第 i 个 grid cell 的第 j 个 bounding box 若负责预测物体则为 1，否则为 0。
- C_i : Predicted confidence score
- \hat{C}_i : Ground truth confidence score
- $p_i(c)$: Predicted probability of class c
- $\hat{p}_i(c)$: Ground truth probability of class c

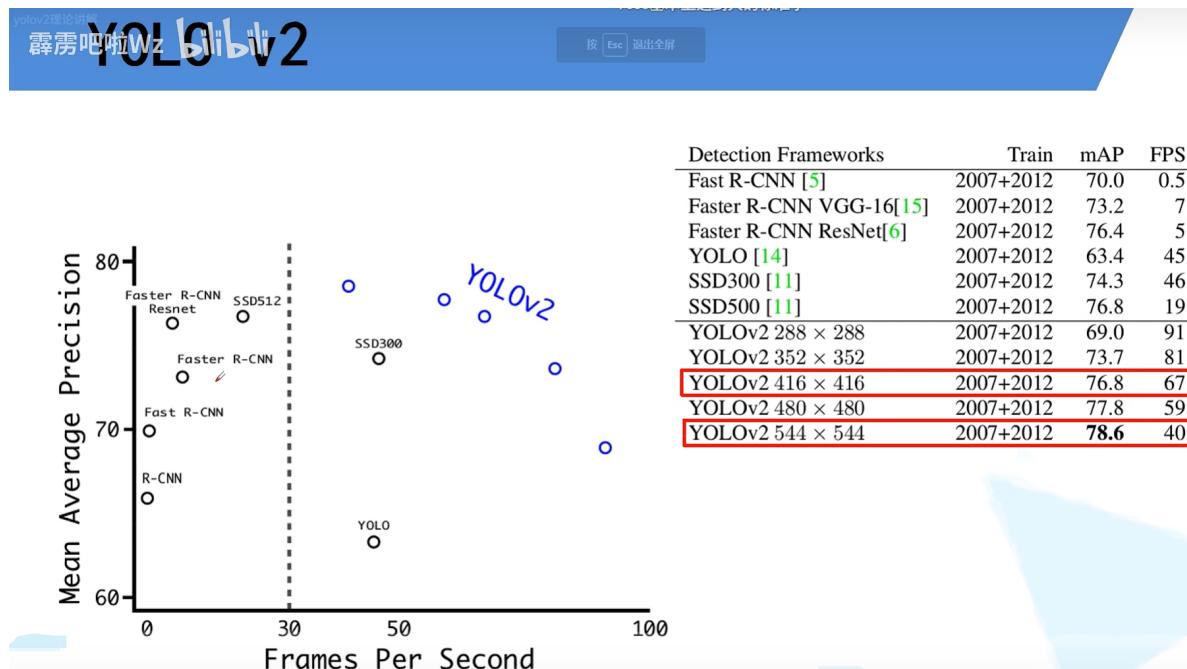
1. Leaky ReLU

2. Grid Cell

模型泛化迁移能力较强，小物体检测性能较差（由于grid cell的限制导致）

定位误差较大，但区分背景能力强（对比RCNN，可以看到整图信息，而不是region proposal）

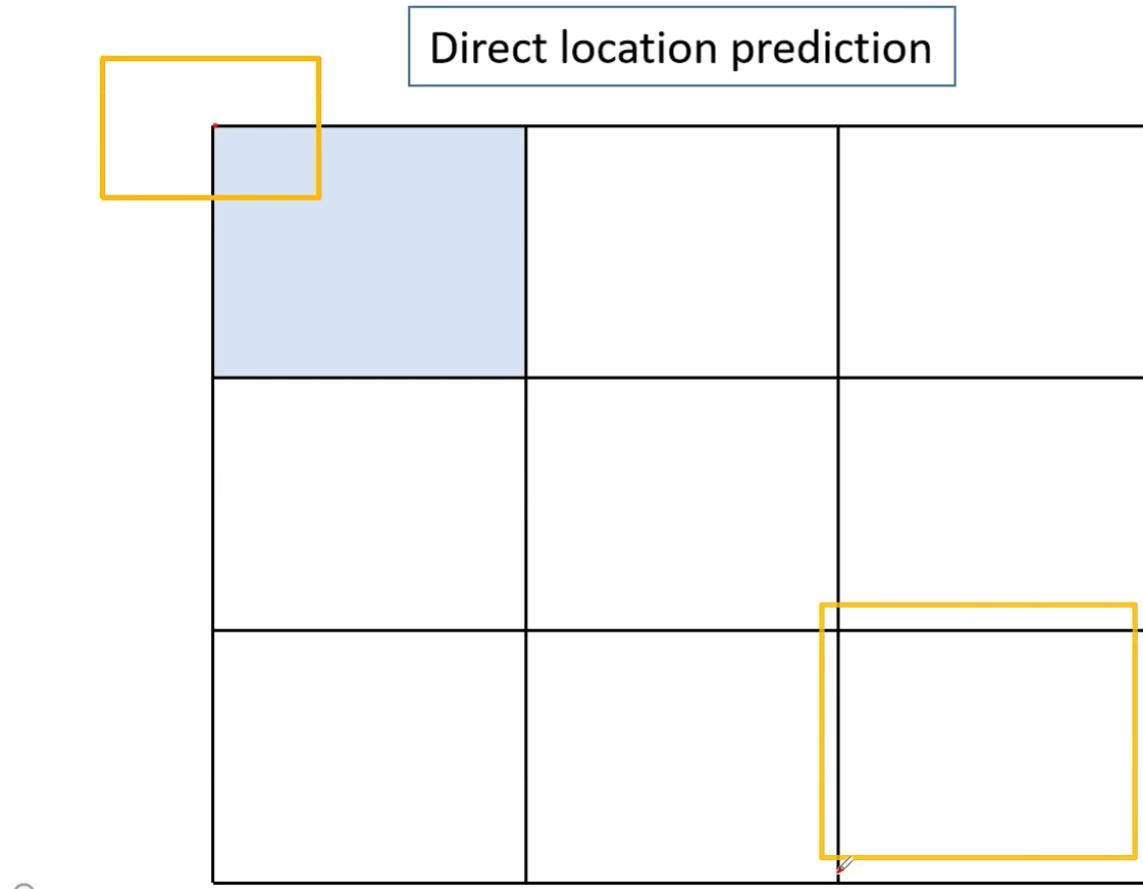
YOLOV2



Better

- 引入BN层 (batch normalization)，加在每一个卷积层之后，不仅改善了网络性能，还有一定的正则化的作用，因此文章移除了yolo中使用的dropout操作

2. 高分辨率输入
3. 引入anchor机制，与FasterRcnn不同，anchor的大小与比例不为预设，通过k-means来聚类出k类 anchor类，文章最终k=5
4. 修改了bbox regression的计算方式，FasterRcnn中tx, ty没有规定范围，在训练初期不稳定，如下图所示，左上角为原bbox，回归后可能到右下角



fasterrcn预测方式如下：

$$x = (t_x * w_a) + x_a$$

$$y = (t_y * h_a) + y_a$$

上式中， x 、 y 是预测的框中心， x_a 、 y_a 是anchor框的中心点坐标， w_a 、 h_a 是anchor框的宽和高， t_x 、 t_y 是网络的输出。

注意：这里与yolo9000原文不同的是，上式中是‘-’号，但是按照fasterrcnn的公式推导其实应该为‘+’，这样更好理解

fasterrcnn这种训练方式对于 t_x 和 t_y 没有约束，使得训练早期坐标不容易稳定。

所以yolov2的预测方式如下：

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

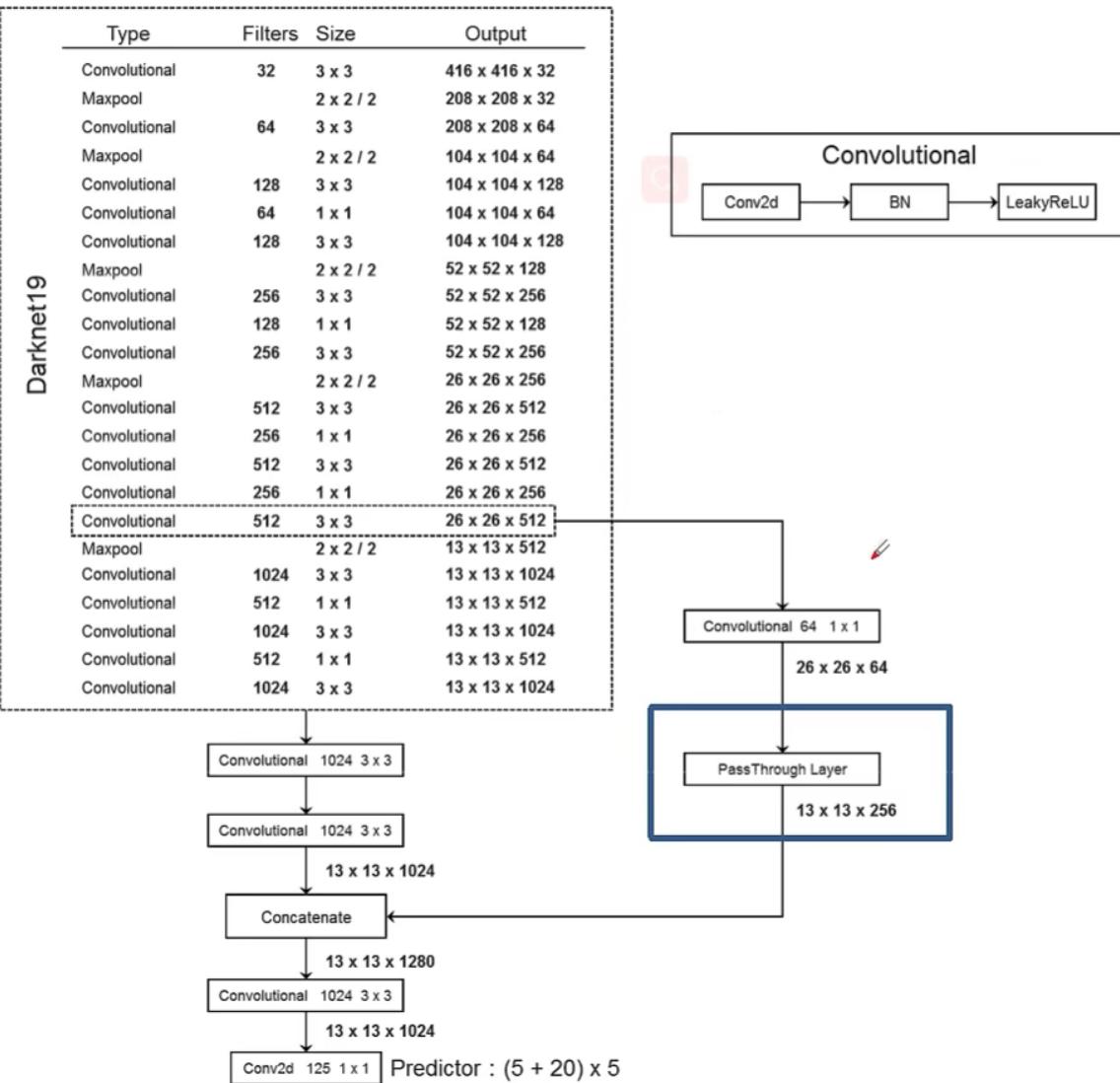
$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

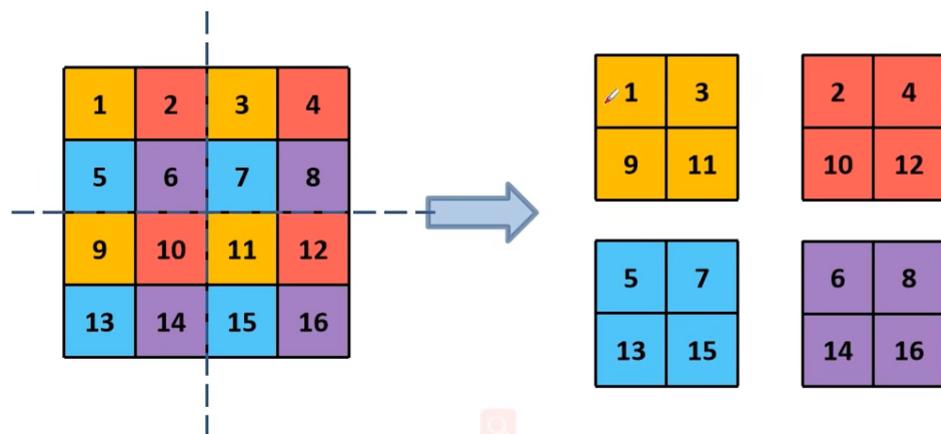
$$Pr(object) * IOU(b, object) = \sigma(t_o)$$

上式中， t_x, t_y, t_w, t_h, t_o 分别为预测参数， b_x, b_y, b_w, b_h 为归一化的预测框的中心坐标和长宽， c_x, c_y 是当前网格距离左上角的距离，该距离也为规划化后的距离， p_w, p_h 表示anchor的长宽， σ 表示sigmoid函数。上述中的归一化指的是每个网格的长宽为1。这里可以看出因为加了sigmoid函数，使得预测出的 $\sigma(t_x), \sigma(t_y)$ 始终为0-1的，就不容易造成训练早期的坐标不稳定了。

5. 细粒度的特征 (Fine-Grained Features)，引入了passthrough layer，将高层特征和低层特征融合（低层特征保留的信息较多，可用来检测小目标），对小目标检测更精确



PassThrough Layer ($W/2, H/2, C \times 4$)



由上图可以看到，经过passthrough layer后，宽高变成原来的一半，channel变为四倍

6. 多尺度训练：

7. yolov1中对于检测任务采用的是448*448来训练网络。yolov2为了适应多尺度的物体检测，网络的训练时采用多种图片尺寸，这些尺寸为32的倍数，有320, 352, 608。训练过程中，每10个 batches后随机选择这些尺度中的一个输入网络进行训练。

Faster

Darknet-19

霹雳吧啦Wz bili bili

YOLOv2

BackBone: Darknet-19

Faster章节

Darknet-19(224x224) only requires **5.58 billion operations** to process an image yet achieves **72.9% top-1 accuracy** and **91.2% top-5 accuracy** on ImageNet.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

PS: 图中Convolutional = conv2d+BN+LeakyRelu

YOLOv3

backbone: DarkNet-53

霹雳吧啦Wz bili bili

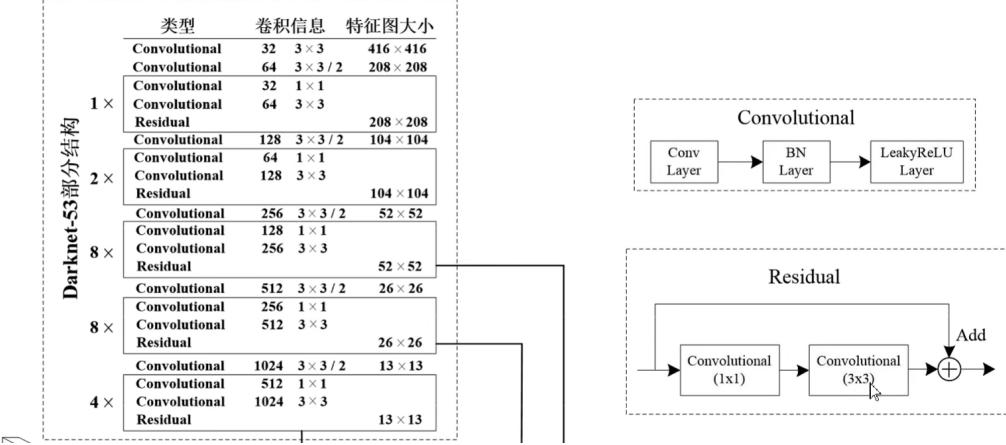
YOLOv3

Darknet-53

Type	Filters	Size	Output
Convolutional	32	3×3	256×256
Convolutional	64	$3 \times 3 / 2$	128×128
1x			
Convolutional	32	1×1	
Convolutional	64	3×3	
Residual			128×128
Convolutional	128	$3 \times 3 / 2$	64×64
2x			
Convolutional	64	1×1	
Convolutional	128	3×3	
Residual			64×64
Convolutional	256	$3 \times 3 / 2$	32×32
8x			
Convolutional	128	1×1	
Convolutional	256	3×3	
Residual			32×32
Convolutional	512	$3 \times 3 / 2$	16×16
8x			
Convolutional	256	1×1	
Convolutional	512	3×3	
Residual			16×16
Convolutional	1024	$3 \times 3 / 2$	8×8
4x			
Convolutional	512	1×1	
Convolutional	1024	3×3	
Residual			8×8
Avgpool		Global	
Connected		1000	
Softmax			

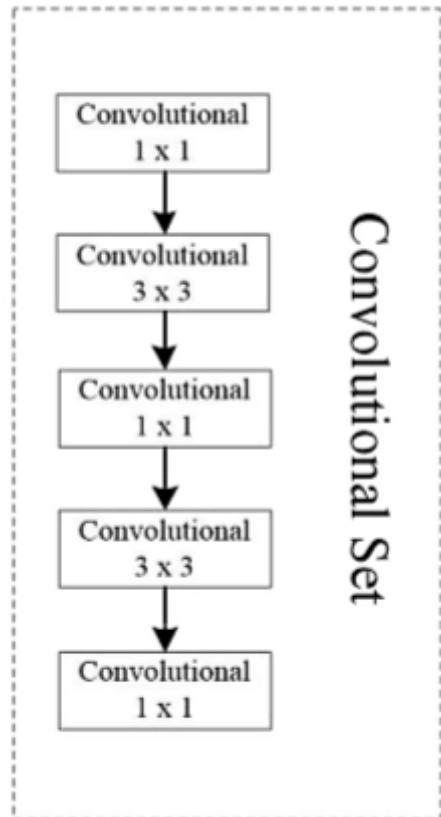
Table 2. Comparison of backbones. Accuracy, billions of operations, billion floating point operations per second, and FPS for various networks.

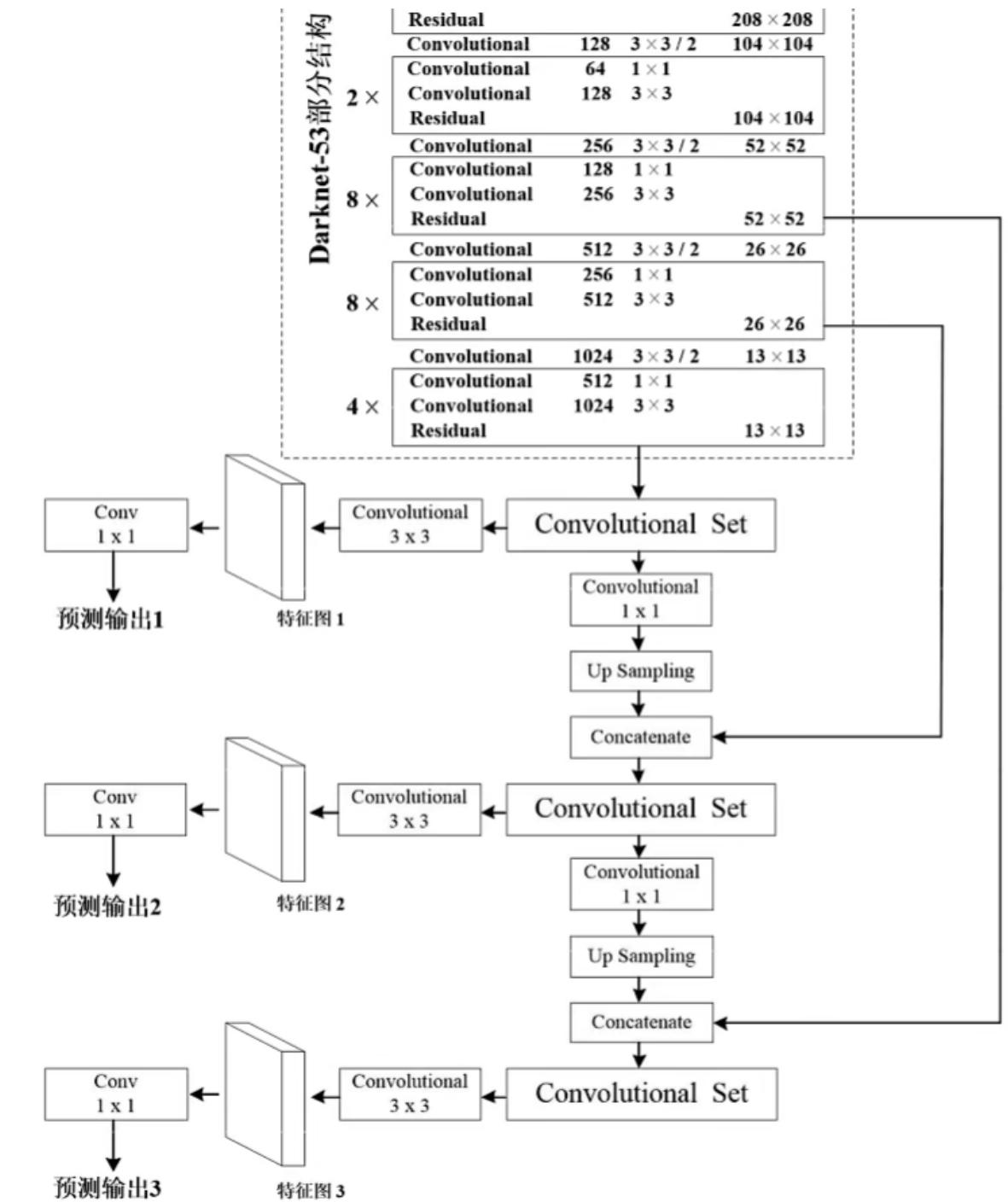
2+
 (1x2)+1+
 (2x2)+1+
 (8x2)+1+
 (8x2)+1+
 (4x2)+1=53



没有Max Pooling层，下采样用Conv层进行

多尺度输出，不同的feature大小负责不同大小的框的检测





此处Concatenate前的多尺度融合是在通道维度的相加（通道数相加），对比FPN中是通道数相同，各通道分别相加（通道数不变）

损失计算

损失的计算

置信度损失 分类损失 定位损失

$$L(o, c, O, C, l, g) = \lambda_1 L_{conf}(o, c) + \lambda_2 L_{cla}(O, C) + \lambda_3 L_{loc}(l, g)$$

$\lambda_1, \lambda_2, \lambda_3$ 为平衡系数

置信度损失

Binary Cross Entropy

$$L_{conf}(o, c) = -\frac{\sum_i(o_i \ln(\hat{c}_i) + (1-o_i) \ln(1-\hat{c}_i))}{N}$$

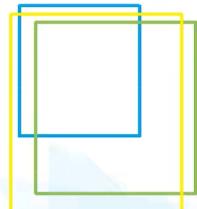
$$\hat{c}_i = Sigmoid(c_i)$$

其中 $o_i \in [0, 1]$, 表示预测目标边界框与真实目标边界框的IOU,

c 为预测值, \hat{c}_i 为 c 通过Sigmoid函数得到的预测置信度,

N 为正负样本个数

YOLOv3 predicts an objectness score for each bounding box using logistic regression. This should be 1 if the bounding box prior overlaps a ground truth object by more than any other bounding box prior. If the bounding box prior



类别损失

Binary Cross Entropy

$$L_{cla}(O, C) = -\frac{\sum_{i \in pos} \sum_{j \in cla} (O_{ij} \ln(\hat{C}_{ij}) + (1 - O_{ij}) \ln(1 - \hat{C}_{ij}))}{N_{pos}}$$

$$\hat{C}_{ii} = Sigmoid(C_{ii})$$

其中 $O_{ij} \in \{0, 1\}$, 表示预测目标边界框*i*中是否存在第*j*类目标,

$C_{\bar{u}}$ 为预测值, $\hat{C}_{\bar{u}}$ 为 $C_{\bar{u}}$ 通过Sigmoid函数得到的目标概率

N_{pos} 为正样本个数

Each box predicts the classes the bounding box may contain using multilabel classification. We do not use a softmax as we have found it is unnecessary for good performance, instead we simply use independent logistic classifiers. During training we use binary cross-entropy loss for the class predictions.

定位损失

$$L_{loc}(t, g) = \frac{\sum_{i \in pos} (\sigma(t_x^i) - \hat{g}_x^i)^2 + (\sigma(t_y^i) - \hat{g}_y^i)^2 + (t_w^i - \hat{g}_w^i)^2 + (t_h^i - \hat{g}_h^i)^2}{N_{pos}}$$

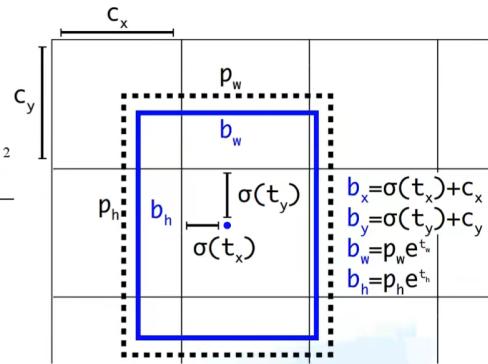
$$\hat{g}_x^i = g_x^i - c_x^i$$

$$\hat{g}_y^i = g_y^i - c_y^i$$

$$\hat{g}_w^i = \ln(g_w^i / p_w^i)$$

$$\hat{g}_h^i = \ln(g_h^i / p_h^i)$$

t_x, t_y, t_w, t_h : 为网络预测的回归参数
 g_x, g_y, g_w, g_h : 为GT 中心点的坐标
 x, y 以及宽度和高度
(映射在Grid网格中的)



During training we use sum of squared error loss. If the ground truth for some coordinate prediction is \hat{t}_* our gradient is the ground truth value (computed from the ground truth box) minus our prediction: $\hat{t}_* - t_*$. This ground truth value can be easily computed by inverting the equations

YOLOV3SPP

Mosaic图像增强：将多张图片拼接在一起（默认四张）进行训练

1. 增加了一张图片中的目标个数
2. 增加了图片的多样性（随机组合）
3. 变相增大batchsize（因为多张图片拼接在一起变成一张图片），有利于BN层对均值和方差的统计（样本变大）

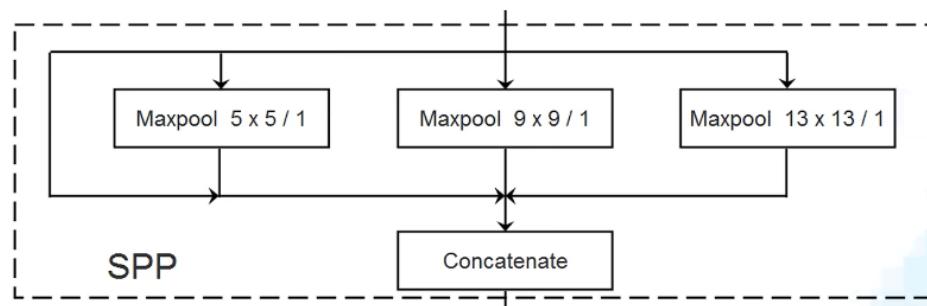
SPP结构：因为会padding，所以四个部分的whc都是一样的，可以Concatenate拼接

实现了不同尺度的特征融合

SPP模块

实现了不同尺度的特征融合

注意：这里的SPP和SPPnet中的SPP结构不一样
Spatial Pyramid Pooling



IOU LOSS:

一般 = 1-IOU

优点：相比L2 LOSS能够更好的反应重合程度，具有尺度不变性

缺点：没有重合时，IOU为0（无法优化）

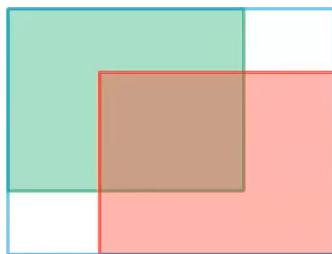
GIOU LOSS:

GIoU Loss

Generalized IoU

$$GIoU = IoU - \frac{A^c - u}{A^c}$$

$$-1 \leq GIoU \leq 1$$



$$L_{GIoU} = 1 - GIoU$$

$$0 \leq L_{GIoU} \leq 2$$

Table 1. Comparison between the performance of YOLO v3 [21] trained using its own loss (MSE) as well as \mathcal{L}_{IoU} and \mathcal{L}_{GIoU} losses. The results are reported on the test set of PASCAL VOC 2007.

Loss / Evaluation	AP		AP75	
	IoU	GIoU	IoU	GIoU
MSE [21]	.461	.451	.486	.467
\mathcal{L}_{IoU}	.466	.460	.504	.493
Relative improv %	1.08%	2.02%	3.70%	6.64%
\mathcal{L}_{GIoU}	.477	.469	.513	.499
Relative improv %	3.45%	4.08%	5.56%	6.85%

A^c 为蓝框面积, u 为并集面积

与IOU共同缺点：收敛较慢，还不够精确

DIOU LOSS:

是等于IoU了，GIoU等于0了

DIoU Loss
Distance-IoU

L_{IoU} Slow Convergence
 L_{GIoU} Inaccurate Regression

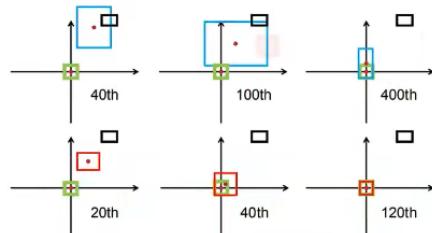


Figure 1: Bounding box regression steps by GIoU loss (first row) and DIoU loss (second row). Green and black denote target box and anchor box, respectively. Blue and red denote predicted boxes for GIoU loss and DIoU loss, respectively. GIoU loss generally increases the size of predicted box to overlap with target box, while DIoU loss directly minimizes normalized distance of central points.

如何更快的收敛?
如何达到更高的定位精度?

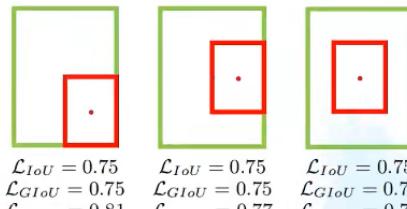


Figure 2: GIoU loss degrades to IoU loss for these cases, while our DIoU loss is still distinguishable. Green and red denote target box and predicted box respectively.

DIoU Loss

$$DIoU = IoU - \frac{\rho^2(b, b^{gt})}{c^2} = IoU - \frac{d^2}{c^2}$$

$$-1 \leq DIoU \leq 1$$



Figure 5: DIoU loss for bounding box regression, where the normalized distance between central points can be directly minimized. c is the diagonal length of the smallest enclosing box covering two boxes.

DIoU损失能够直接最小化两个boxes之间的距离，因此收敛速度更快。

$$L_{DIoU} = 1 - DIoU$$

$$0 \leq L_{DIoU} \leq 2$$

Loss / Evaluation	AP	
	IoU	GIoU
\mathcal{L}_{IoU}	46.57	45.82
\mathcal{L}_{GIoU}	47.73	46.88
Relative improv. %	2.49%	2.31%
\mathcal{L}_{DIoU}	48.10	47.38
Relative improv. %	3.29%	3.40%

CIOU LOSS:

CloU Loss

Complete-IoU

$$CIoU = IoU - \left(\frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \right)$$

$$v = \frac{4}{\pi^2} \left(\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h} \right)^2$$

$$\alpha = \frac{v}{(1 - IoU) + v}$$

一个优秀的回归定位损失应该考虑到3种几何参数：
重叠面积 中心点距离 长宽比

$$L_{CIoU} = 1 - CIoU$$

Loss / Evaluation	AP		AP75	
	IoU	GIoU	IoU	GIoU
\mathcal{L}_{IoU}	46.57	45.82	49.82	48.76
\mathcal{L}_{GIoU}	47.73	46.88	52.20	51.05
Relative improv. %	2.49%	2.31%	4.78%	4.70%
\mathcal{L}_{DIoU}	48.10	47.38	52.82	51.88
Relative improv. %	3.29%	3.40%	6.02%	6.40%
\mathcal{L}_{CIoU}	49.21	48.42	54.28	52.87
Relative improv. %	5.67%	5.67%	8.95%	8.43%
$\mathcal{L}_{CIoU}(D)$	49.32	48.54	54.74	53.30
Relative improv. %	5.91%	5.94%	9.88%	9.31%

Focal LOSS:

2、**Balanced Cross Entropy**: 常见的解决类不平衡方法。引入了一个权重因子 $\alpha \in [0, 1]$ ，当为正样本时，权重因子就是 α ，当为负样本时，权重因子为 $1-\alpha$ 。所以，损失函数也可以改写为：

$$\text{CE}(p_t) = -\alpha_t \log(p_t).$$

这里给出一张图：

α	AP	AP ₅₀	AP ₇₅
.10	0.0	0.0	0.0
.25	10.8	16.0	11.7
.50	30.2	46.7	32.8
.75	31.1	49.4	33.0
.90	30.8	49.7	32.3
.99	28.7	47.4	29.9
.999	25.1	41.7	26.1

CSDN用户:lao688

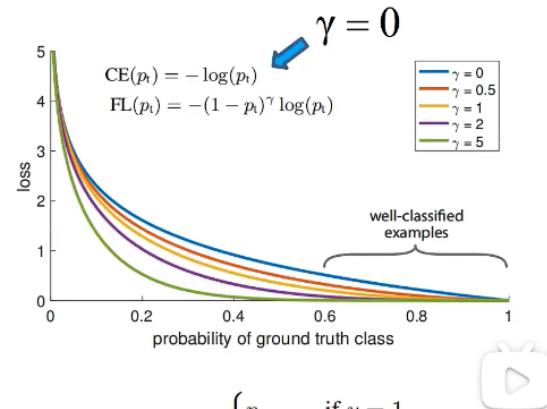
3.2. Focal Loss Definition

As our experiments will show, the large class imbalance encountered during training of dense detectors overwhelms the cross entropy loss. Easily classified negatives comprise the majority of the loss and dominate the gradient. While α balances the importance of positive/negative examples, it does not differentiate between easy/hard examples. Instead, we propose to reshape the loss function to down-weight easy examples and thus focus training on hard negatives.

More formally, we propose to add a modulating factor $(1 - p_t)^\gamma$ to the cross entropy loss, with tunable *focusing* parameter $\gamma \geq 0$. We define the focal loss as:

$$\text{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t). \quad (4)$$

$(1 - p_t)^\gamma$ 能够降低易分样本的损失贡献



$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise,} \end{cases}$$



区分容易的样本和困难的样本，将训练中心放在困难样本上（即是正样本但是IOU较小和负样本但是IOU较大的样本）

1. 是正样本但是IOU较小， $p_t = p$ 但是较小，则 $1-p_t$ 较大， γ 之后更大
2. 是负样本但是IOU较大， $p_t = 1 - p$ ， $1-p_t = p$ ，但是 p 较大， γ 之后更大

Focal loss

$$FL(p) = \begin{cases} -\alpha(1-p)^\gamma \log(p) & \text{if } y=1 \\ -(1-\alpha)p^\gamma \log(1-p) & \text{otherwise,} \end{cases}$$

In practice we use an α -balanced variant of the focal loss:

$$\underline{FL(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t)}. \quad (5)$$

We adopt this form in our experiments as it yields slightly improved accuracy over the non- α -balanced form. Finally, we note that the implementation of the loss layer combines the sigmoid operation for computing p with the loss computation, resulting in greater numerical stability.

$$p_t = \begin{cases} p & \text{if } y=1 \\ 1-p & \text{otherwise,} \end{cases}$$

γ	α	AP	AP ₅₀	AP ₇₅
0	.75	31.1	49.4	33.0
0.1	.75	31.4	49.9	33.1
0.2	.75	31.9	50.7	33.4
0.5	.50	32.9	51.7	35.2
1.0	.25	33.7	52.0	36.2
2.0	.25	34.0	52.5	36.5
5.0	.25	32.2	49.6	34.8

(b) Varying γ for FL (w. optimal α)



最终：

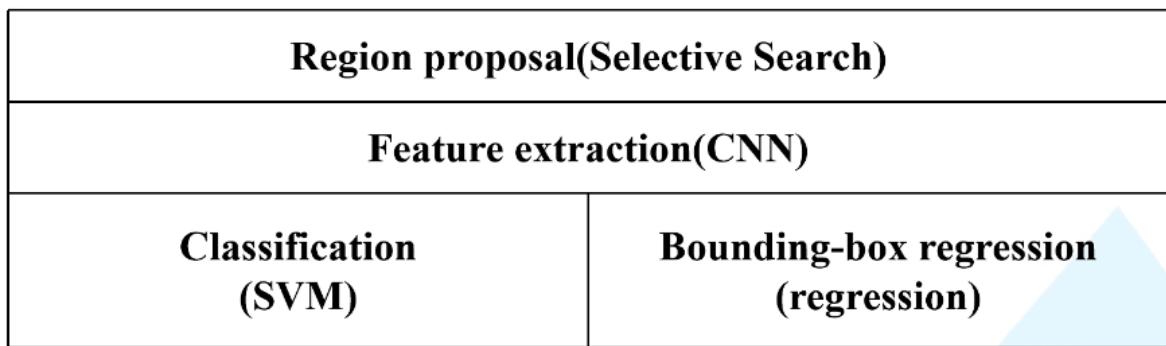
alpha: 平衡正负样本

gamma: 聚焦难分样本

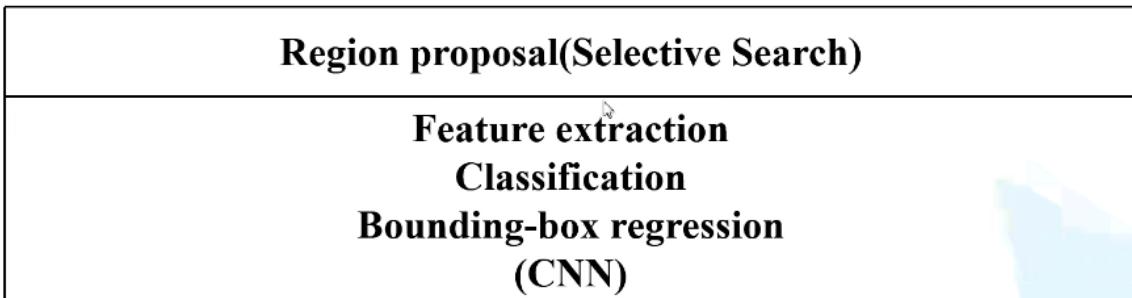
R-CNN家族

框架对比

R-CNN框架



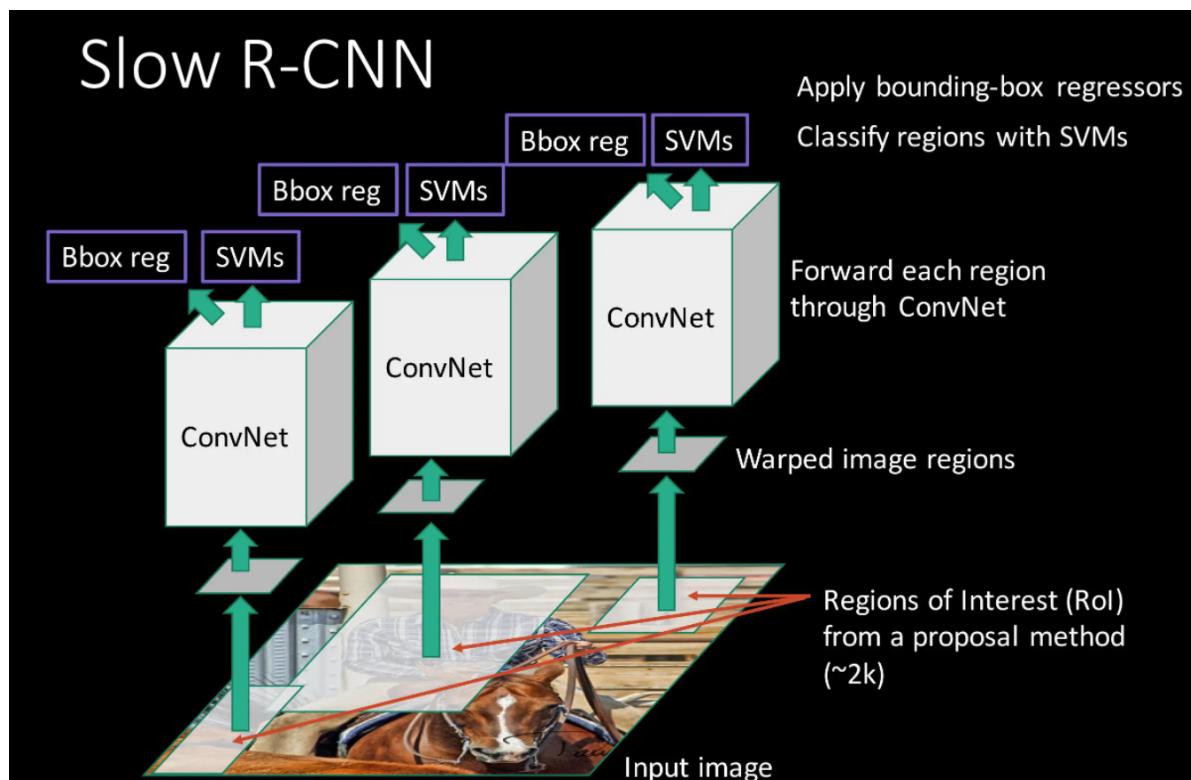
Fast R-CNN框架



Faster R-CNN框架

Region proposal
Feature extraction
Classification
Bounding-box regression
(CNN)

R-CNN



1. pre-trained + fine-tuning (应对小数据集时，用在大数据集上训练的预训练模型，再在小数据集上微调)
2. pre-trained模型的主要信息来自于conv层而不是fc层
3. VGG16作为backbone
4. region proposal
5. selective search
6. bounding box regression
7. SVM

SPP-Net

SPP-net

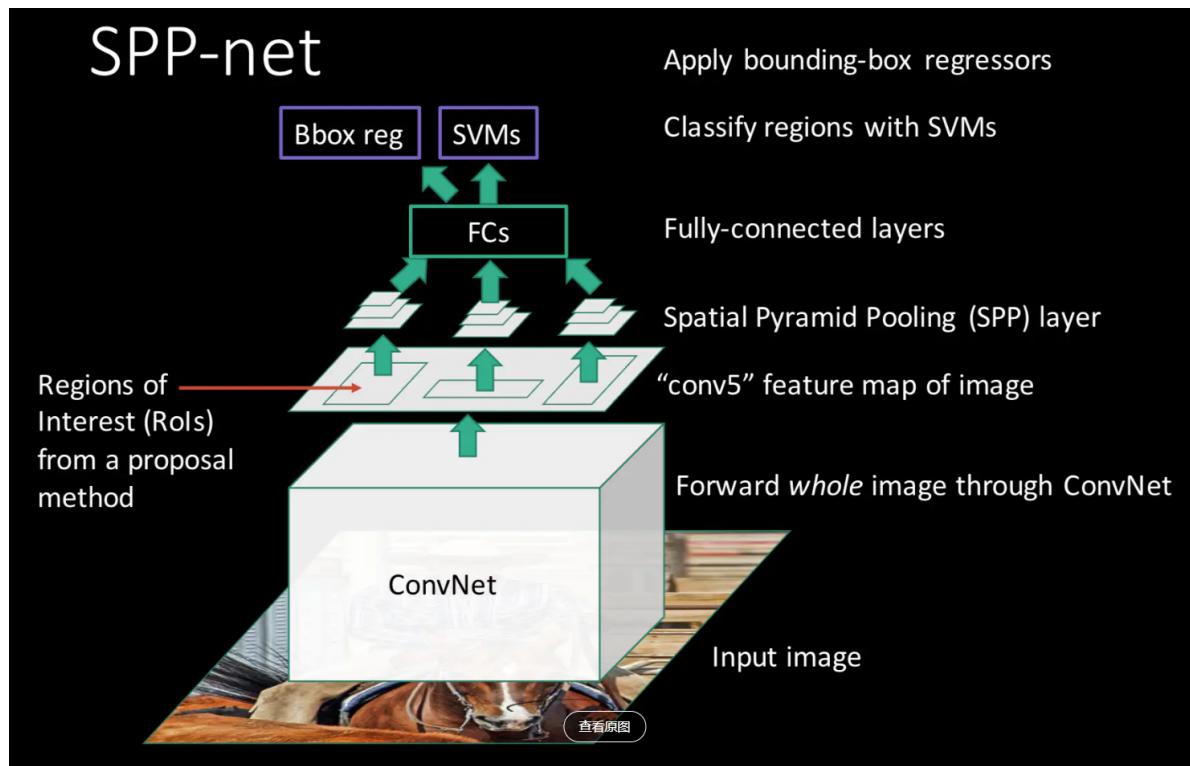
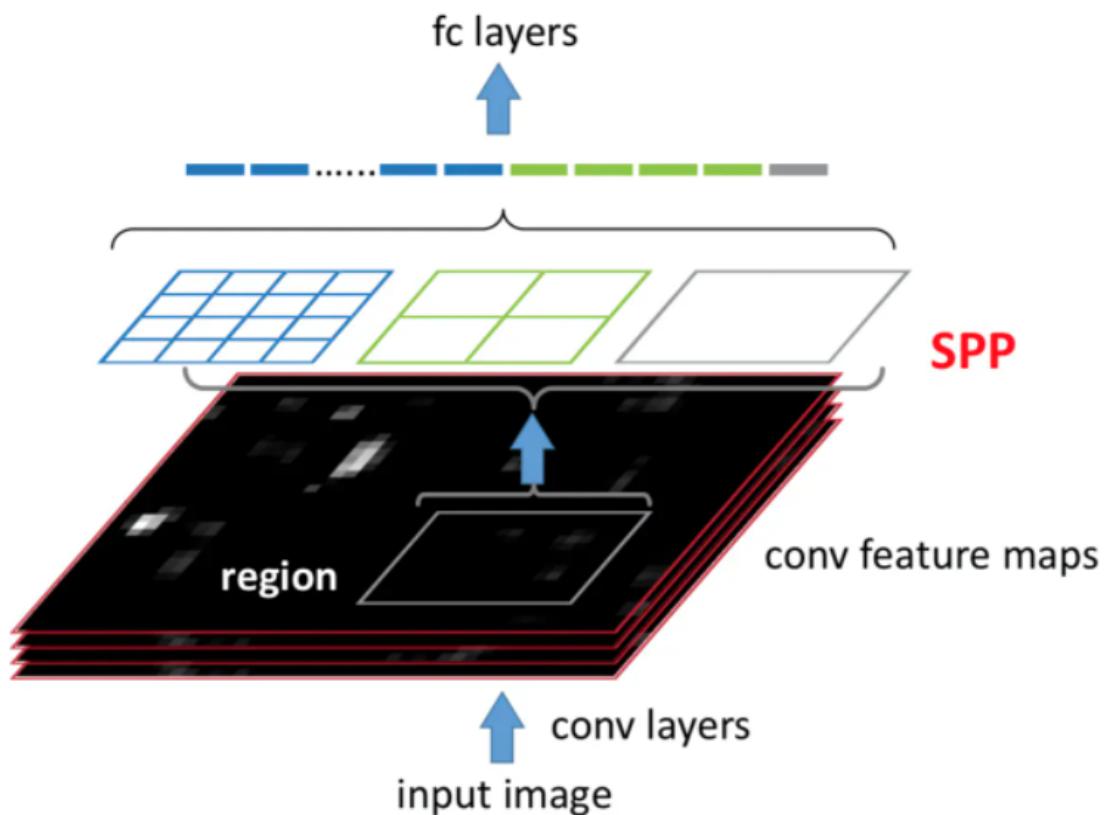


image → conv layers → spatial pyramid pooling → fc layers → output

1. spp层



输入：最后一层卷积输出的特征(我们称为feature map), feature map为上图的黑色部分表示, SPP层的输入为与候选区域对应的在feature map上的一块区域

输出：一共有256个feature map, 每个feature map通过 $4 * 4, 2 * 2, 1 * 1$ 的max pooling的之后得到一个 $(16+4+1) * 256$ 的固定输出的特征向量

2. 候选区域在原图与feature map之间的映射关系

类型	大小
第 l 层的输入尺寸	$W^{[l-1]} * H^{[l-1]}$
第 l 层的输出尺寸	$W^{[l]} * H^{[l]}$
第 l 层的卷积核大小	$f^{[l]} * f^{[l]}$
第 l 层的卷积步长	$S^{[l]}$
第 l 层的填充大小	$p^{[l]}$

输入的尺寸大小与输出的尺寸大小有如下关系：

$$W^{[l]} = (W^{[l-1]} + 2p^{[l]} - f^{[l]})/S^{[l]} + 1$$

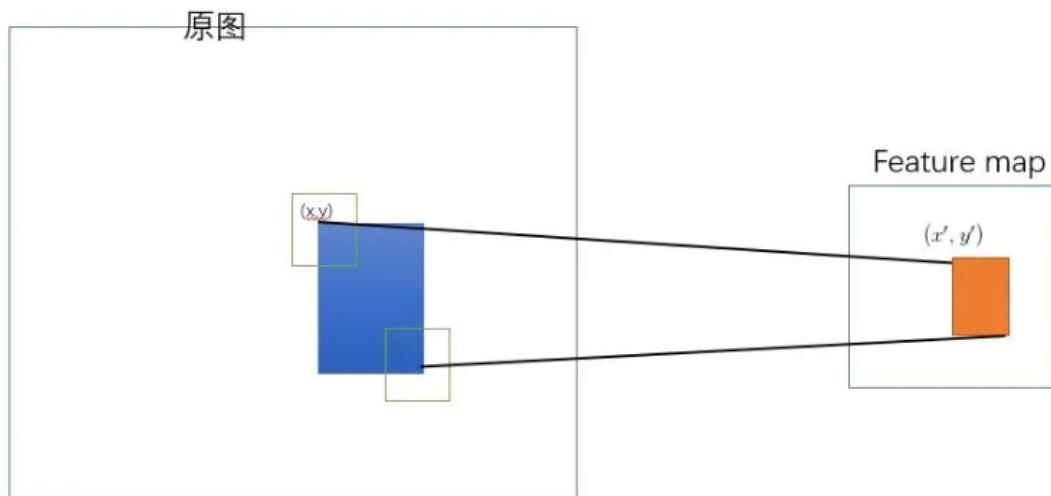
$$H^{[l]} = (H^{[l-1]} + 2p^{[l]} - f^{[l]})/S^{[l]} + 1$$

上面是区域尺寸大小的对应关系，下面看一下坐标点之间的对应关系

$$p_i = s_i \cdot p_{i+1} + ((k_i - 1)/2 - padding)$$

含义	符号
在 i 层的坐标值	p_i
i 层的步长	s_i
i 层的卷积核大小	k_i
i 层填充的大小	padding

spp-net设置每层的padding = $k_i/2$,这样 $p_i = s_i * p_{i+1}$

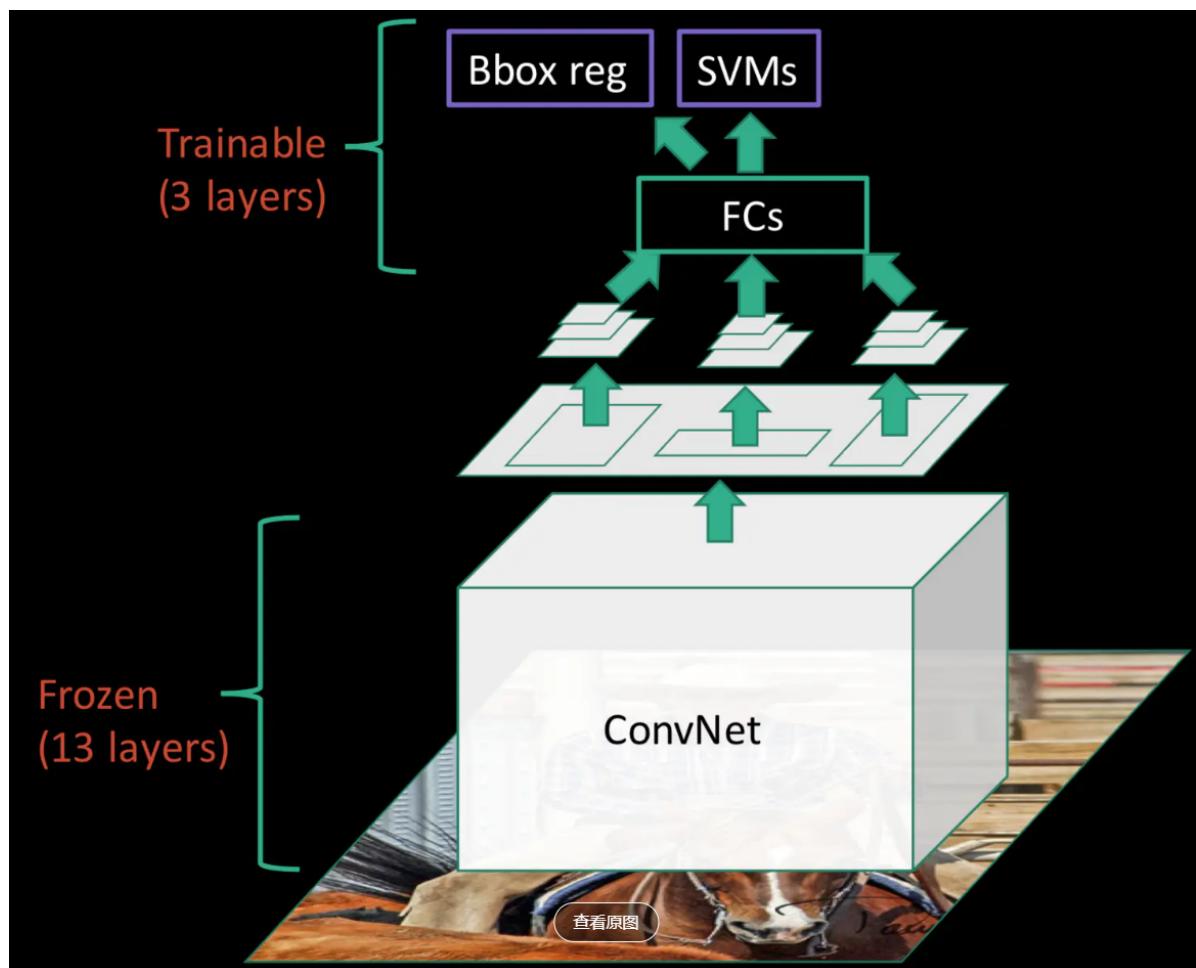


从原图坐标 (x, y) 到特征图中坐标 (x', y') 的映射关系为

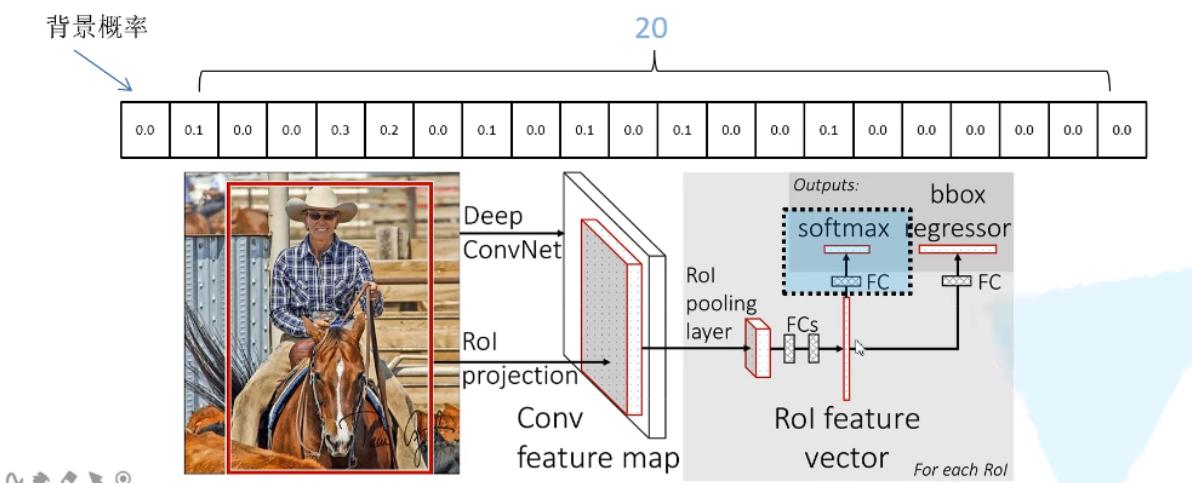
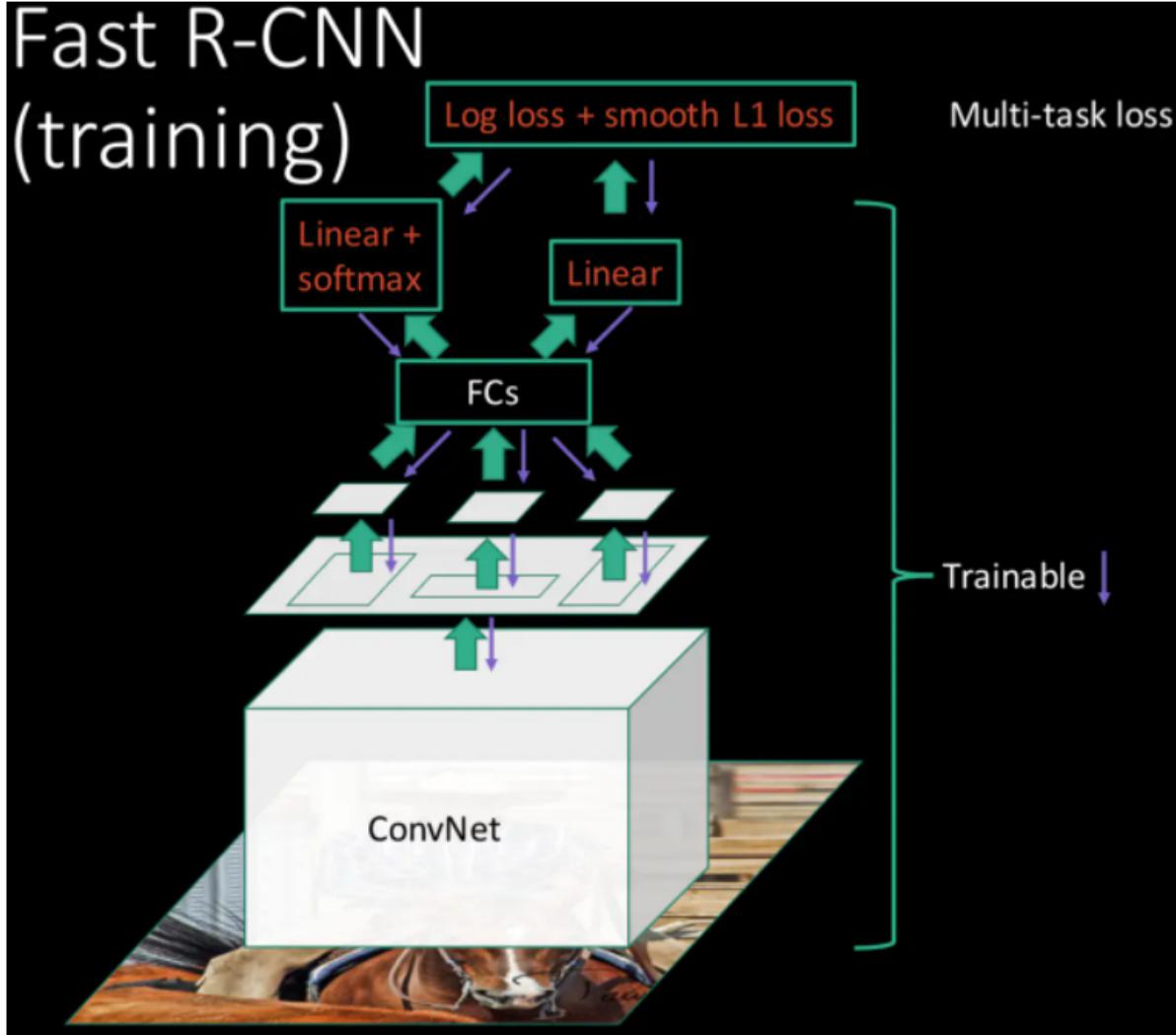
- 前面每层都填充padding/2 得到的简化公式 : $p_i = s_i \cdot p_{i+1}$
- 需要把上面公式进行级联得到 $p_0 = S \cdot p_{i+1}$ 其中 $(S = \prod_0^i s_i)$
- 对于feature map 上的 (x', y') 它在原始图的对应点为 $(x, y) = (Sx', Sy')$
- 论文中的最后做法: 把原始图片中的ROI映射为 feature map中的映射区域(上图橙色区域)其中左上角取: $x' = \lfloor x/S \rfloor + 1, y' = \lfloor y/S \rfloor + 1$; 右下角的点取: $x' = \lceil x/S \rceil - 1, y' = \lceil y/S \rceil - 1$ 。

Fast R-CNN

SPP-Net的缺点: 在fune-tuning阶段不能对SPP层下面所有的卷积层进行后向传播



Fast R-CNN (training)



Fast R-CNN结合了R-CNN与SPP，并作出改进

1. VGG16作为backbone
2. selective search生成约2000候选框，在候选框中随机取N个正样本和N个负样本训练，其中正样本为与GT IOU>0.5,负样本为与GT 0.1<IOU<0.5
3. SPP层，仅用一个尺度的pooling (SPP为多尺度)
4. 引入多任务损失函数，用于同时计算bbox回归和分类的损失

Multi-task loss

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

分类损失 边界框回归损失

p 是分类器预测的softmax概率分布 $p = (p_0, \dots, p_k)$

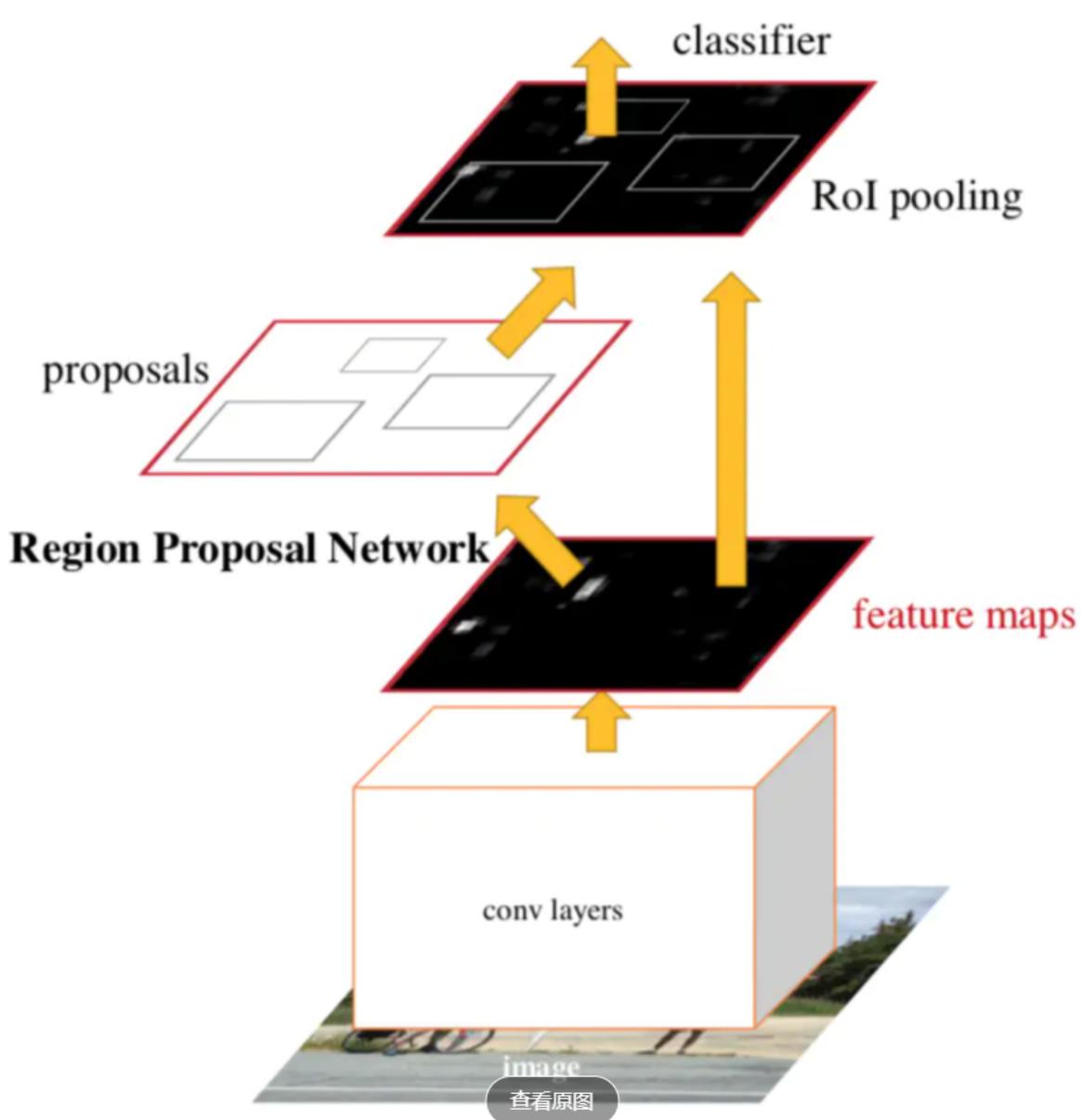
u 对应目标真实类别标签

t^u 对应边界框回归器预测的对应类别 u 的回归参数 $(t_x^u, t_y^u, t_w^u, t_h^u)$

v 对应真实目标的边界框回归参数 (v_x, v_y, v_w, v_h)



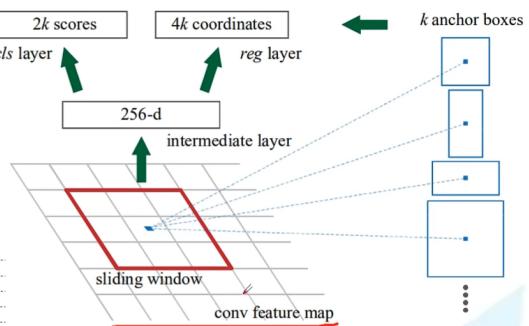
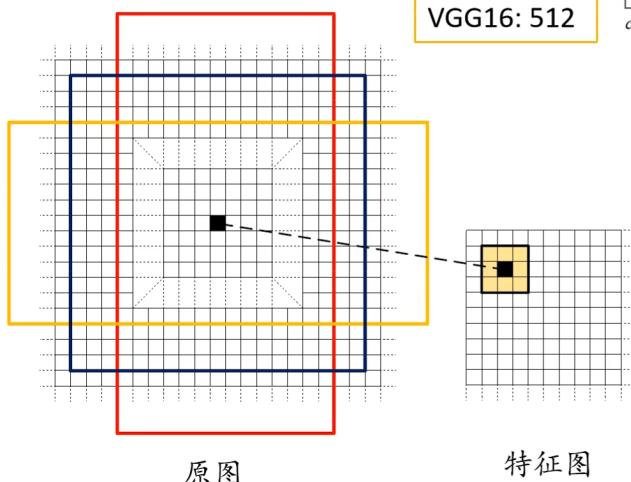
Faster R-CNN



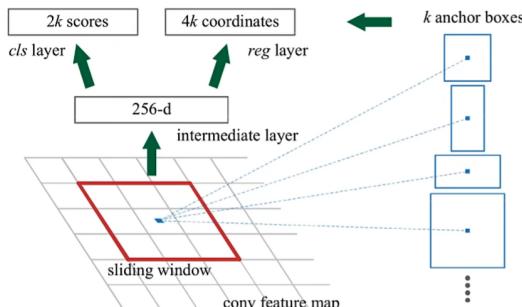
可以看作RPN (region proposal network) +Fast R-CNN

Faster R-CNN

RPN的原论文名字是啥呀



对于特征图上的每个 3×3 的滑动窗口，计算出滑动窗口中心点对应原始图像上的中心点，并计算出 k 个anchor boxes(注意和proposal的差异)。



三种尺度(面积){ $128^2, 256^2, 512^2$ }

三种比例{ 1:1, 1:2, 2:1 }

每个位置在原图上都对应有 $3 \times 3 = 9$ anchor

对于一张 $1000 \times 600 \times 3$ 的图像，大约有 $60 \times 40 \times 9 (20k)$ 个anchor，忽略跨越边界的anchor以后，剩下约 $6k$ 个anchor。对于RPN生成的候选框之间存在大量重叠，基于候选框的cls得分，采用非极大值抑制，IoU设为0.7，这样每张图片只剩 $2k$ 个候选框。



RPN Multi-task loss

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

分类损失 边界框回归损失

p_i 表示第*i*个anchor预测为真实标签的概率

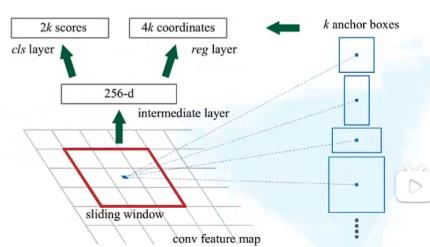
p_i^* 当为正样本时为1,当为负样本时为0

t_i 表示预测第*i*个anchor的边界框回归参数

t_i^* 表示第*i*个anchor对应的GT Box

N_{cls} 表示一个mini-batch中的所有样本数量256

N_{reg} 表示anchor位置的个数(不是anchor个数)约2400



Faster R-CNN

Faster R-CNN训练

直接采用RPN Loss+ Fast R-CNN Loss的联合训练方法

原论文中采用分别训练RPN以及Fast R-CNN的方法

(1)利用ImageNet预训练分类模型初始化前置卷积网络层参数，并开始单独训练RPN网络参数；

(2)固定RPN网络独有的卷积层以及全连接层参数，再利用ImageNet预训练分类模型初始化前置卷积网络参数，并利用RPN网络生成的目标建议框去训练Fast RCNN网络参数。

(3)固定利用Fast RCNN训练好的前置卷积网络层参数，去微调RPN网络独有的卷积层以及全连接层参数。

(4)同样保持固定前置卷积网络层参数，去微调Fast RCNN网络的全连接层参数。最后RPN网络与Fast RCNN网络共享前置卷积网络层参数，构成一个统一网络。

现在直接联合训练，原论文中采用分步训练的方法

