TensorFlow

PYTORCH

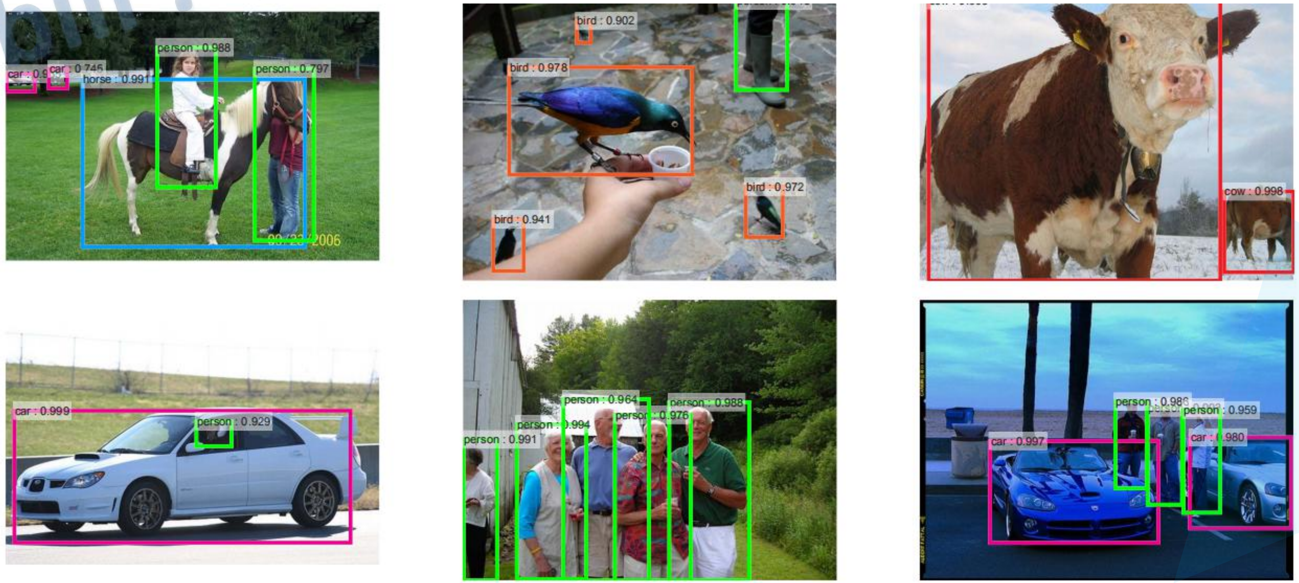
# 深度学习-目标检测篇

作者：神秘的wz

## Faster R-CNN

### Faster R-CNN

Faster R-CNN是作者Ross Girshick继Fast R-CNN后的又一力作。同样使用VGG16作为网络的backbone，推理速度在GPU上达到5fps(包括候选区域的生成)，准确率也有进一步的提升。在2015年的ILSVRC以及COCO竞赛中获得多个项目的第一名。

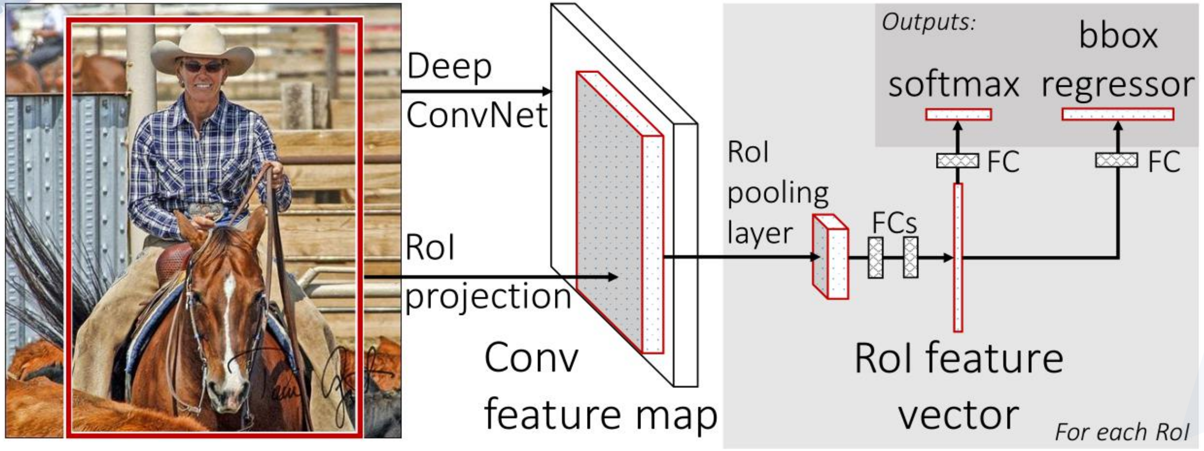


# Fast R-CNN

Fast R-CNN算法流程可分为3个步骤

- 一张图像生成1K~2K个**候选区域**(使用Selective Search方法)
- 将图像输入网络得到相应的**特征图**，将SS算法生成的候选框投影到特征图上获得相应的**特征矩阵**
- 将每个特征矩阵通过ROI pooling层缩放到**7x7大小的特征图**，接着将特征图展平通过一系列全连接层得到预测结果

Region of Interest

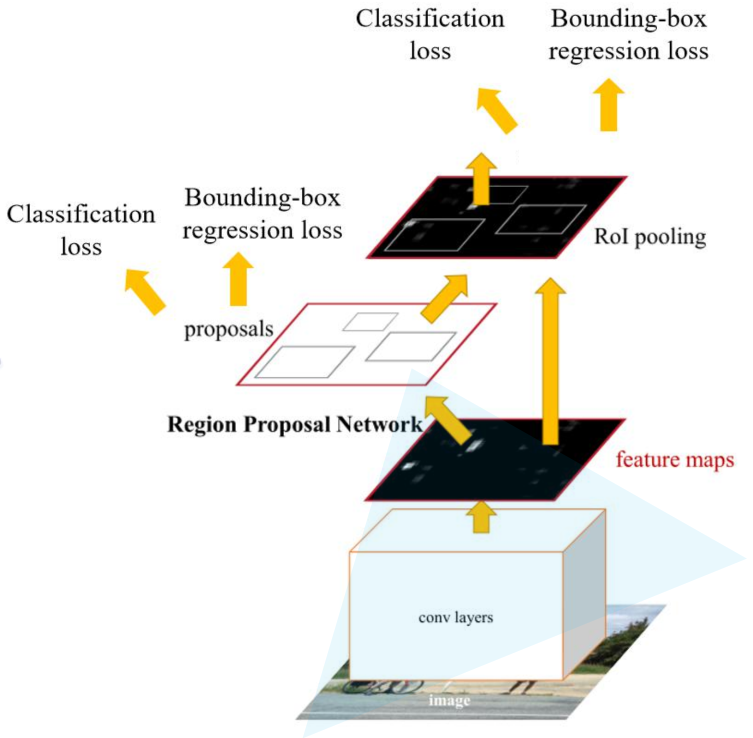


# Faster R-CNN

Faster R-CNN算法流程可分为3个步骤

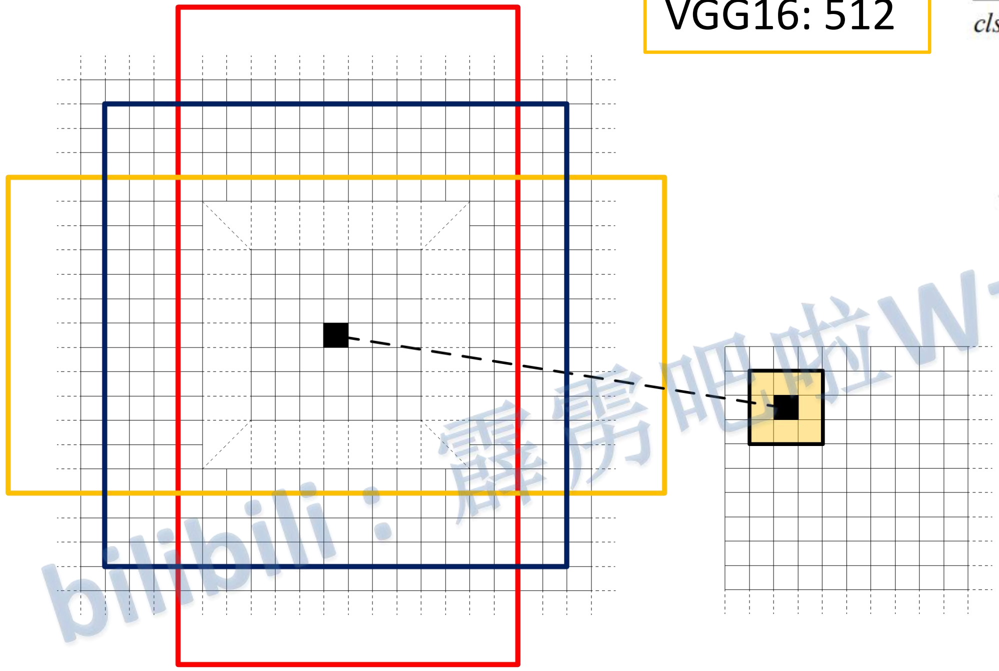
- 将图像输入网络得到相应的**特征图**
- 使用RPN结构生成候选框，将RPN生成的候选框投影到特征图上获得相应的**特征矩阵**
- 将每个特征矩阵通过ROI pooling层缩放到**7x7大小的特征图**，接着将特征图展平通过一系列全连接层得到预测结果

RPN + Fast R-CNN



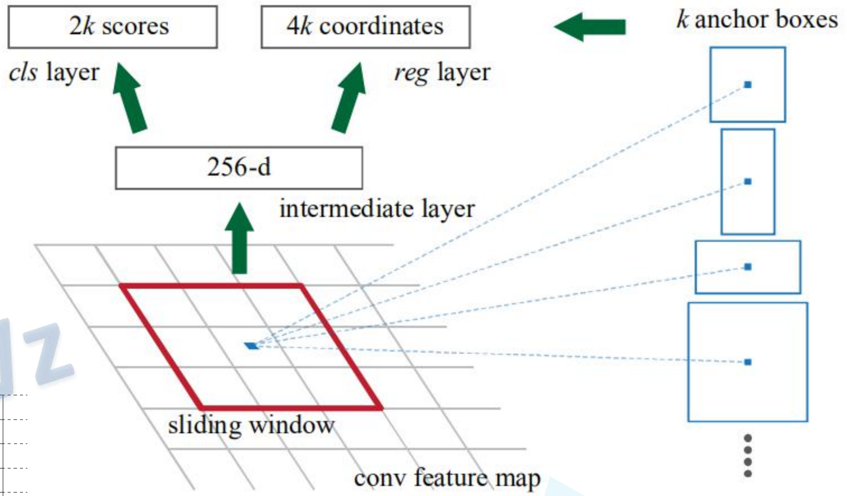
# Faster R-CNN

ZF: 256
VGG16: 512



原图

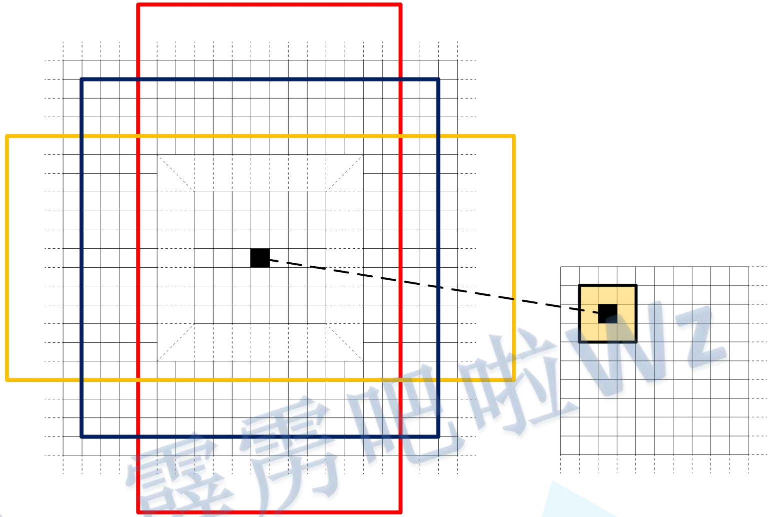
特征图



对于特征图上的每个3x3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)。

# Faster R-CNN

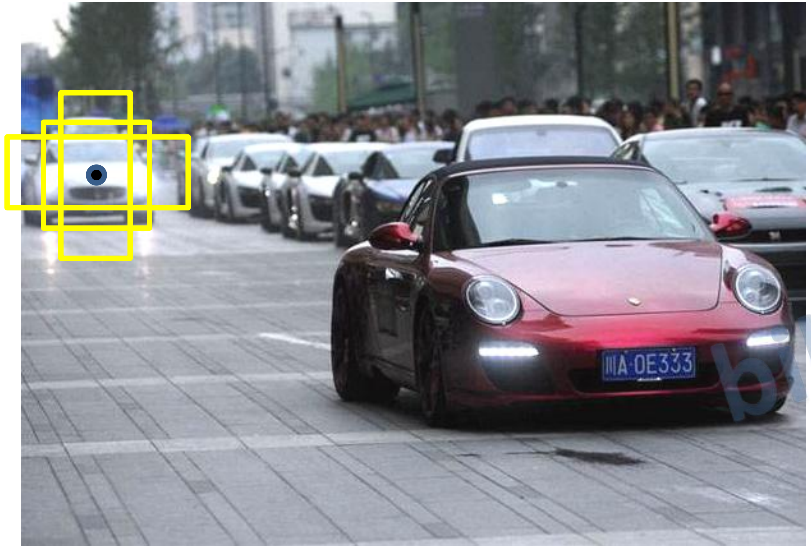
原图

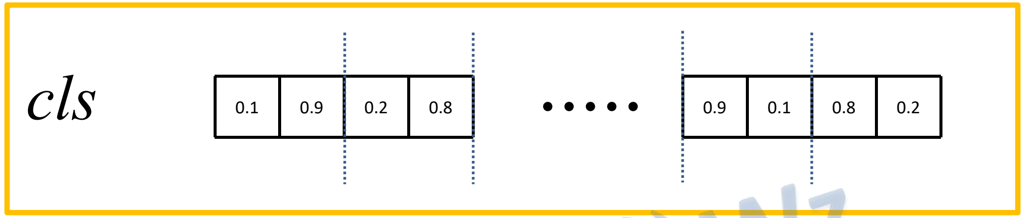


原图

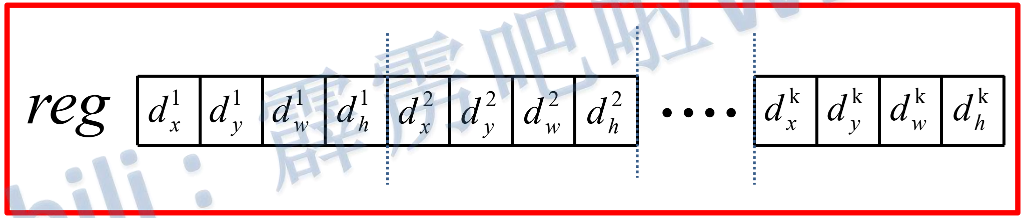
特征图

# Faster R-CNN

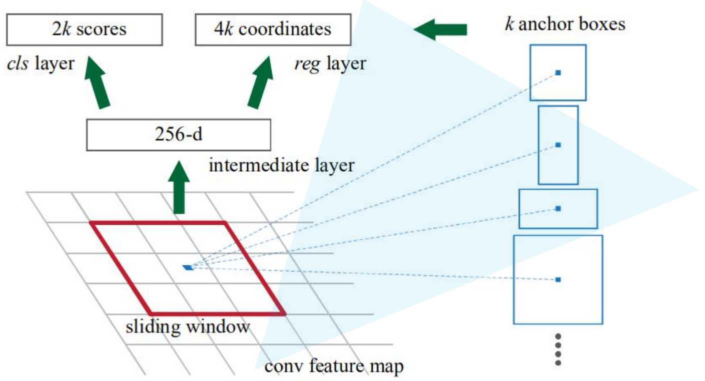




cls
0.1
0.9
0.2
0.8
• • • • •
0.9
0.1
0.8
0.2

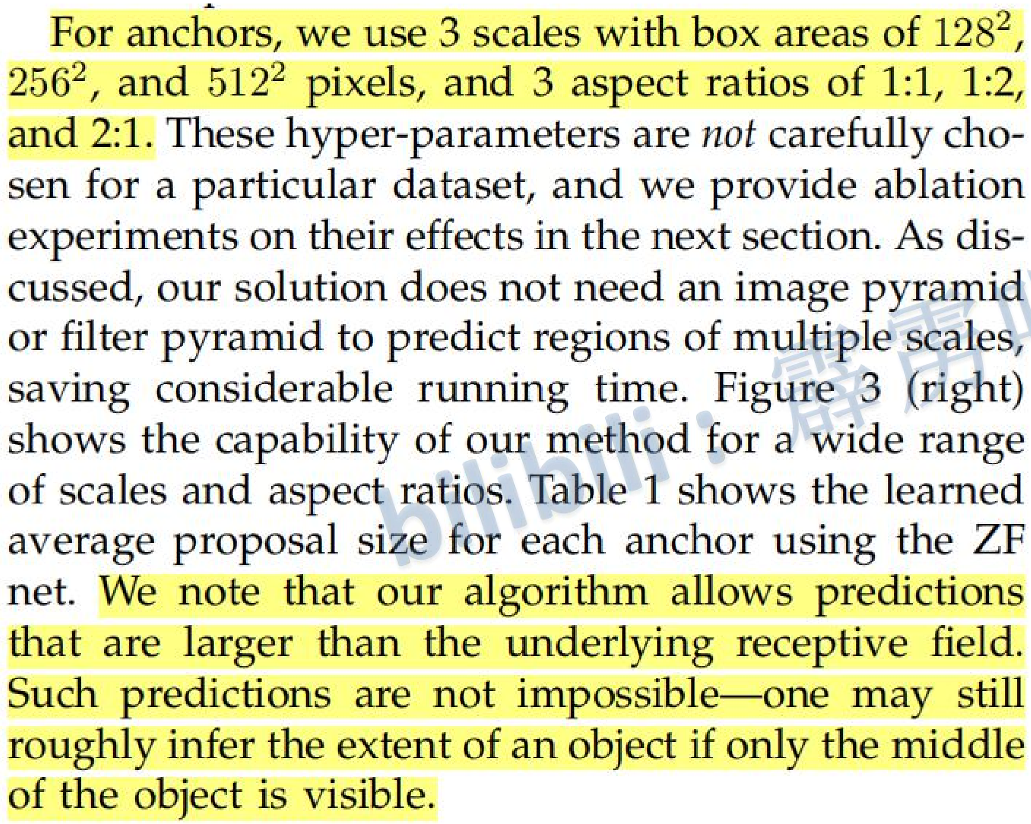


reg
$d_x^1$ $d_y^1$ $d_w^1$ $d_h^1$ $d_x^2$ $d_y^2$ $d_w^2$ $d_h^2$ • • • • $d_x^k$ $d_y^k$ $d_w^k$ $d_h^k$

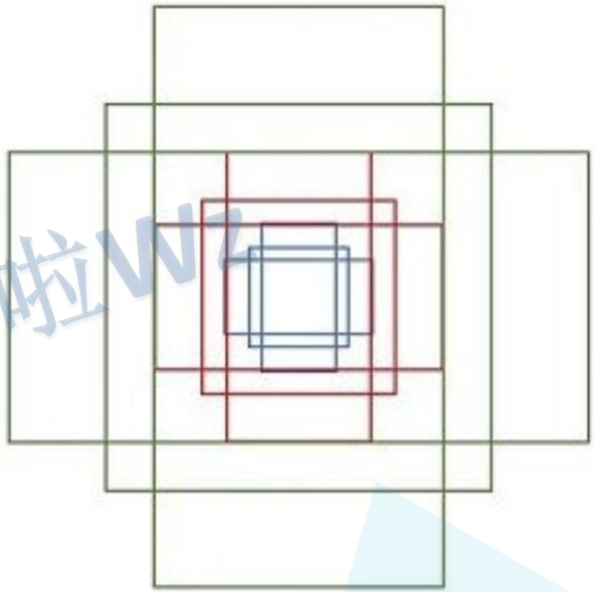


2k scores
4k coordinates
k anchor boxes
cls layer
reg layer
256-d
intermediate layer
sliding window
conv feature map

# Faster R-CNN



For anchors, we use 3 scales with box areas of $128^2$, $256^2$, and $512^2$ pixels, and 3 aspect ratios of 1:1, 1:2, and 2:1. These hyper-parameters are *not* carefully chosen for a particular dataset, and we provide ablation experiments on their effects in the next section. As discussed, our solution does not need an image pyramid or filter pyramid to predict regions of multiple scales, saving considerable running time. Figure 3 (right) shows the capability of our method for a wide range of scales and aspect ratios. Table 1 shows the learned average proposal size for each anchor using the ZF net. We note that our algorithm allows predictions that are larger than the underlying receptive field. Such predictions are not impossible—one may still roughly infer the extent of an object if only the middle of the object is visible.

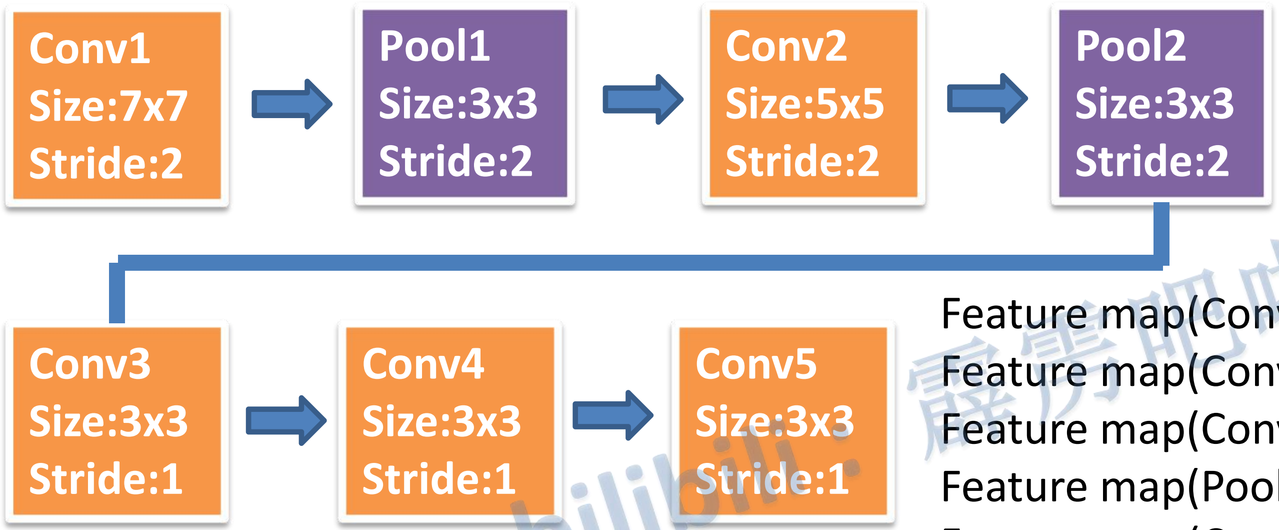


chor

对于ZF感受野: 171
对于VGG感受野: 228

# 基本概念拓展—CNN感受野

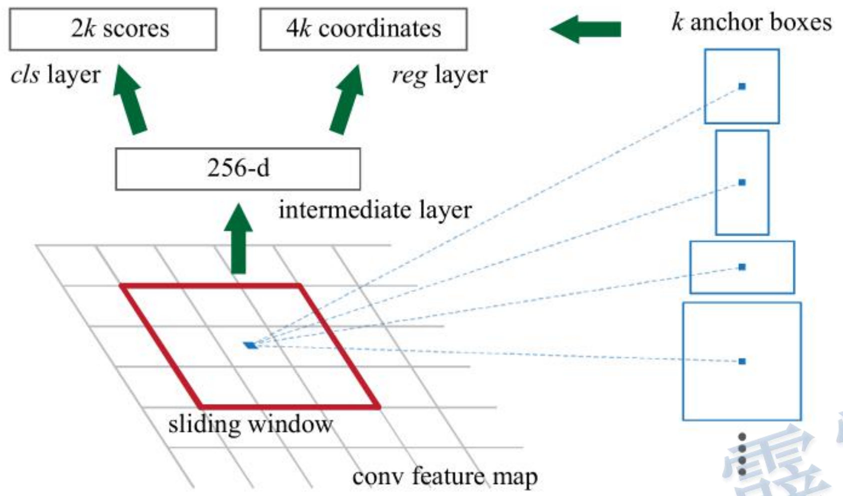
计算Faster RCNN中ZF网络feature map 中3x3滑动窗口在原图中感受野的大小。



$$F(i) = (F(i+1) - 1) \times Stride + Ksize$$

Feature map(Conv5): F = 3
Feature map(Conv4): F = (3 - 1) x 1 + 3 = 5
Feature map(Conv3): F = (5 - 1) x 1 + 3 = 7
Feature map(Pool2): F = (7 - 1) x 1 + 3 = 9
Feature map(Conv2): F = (9 - 1) x 2 + 3 = 19
Feature map(Pool1): F = (19 - 1) x 2 + 5 = 41
Feature map(Conv1): F = (41 - 1) x 2 + 3 = 83
Feature map(Image): F = (83 - 1) x 2 + 7 = 171

# Faster R-CNN



三种尺度(面积){ $128^2, 256^2, 512^2$ }

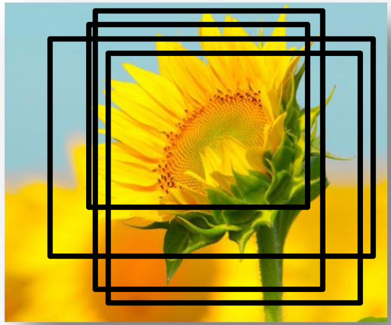
三种比例{ 1:1, 1:2, 2:1 }

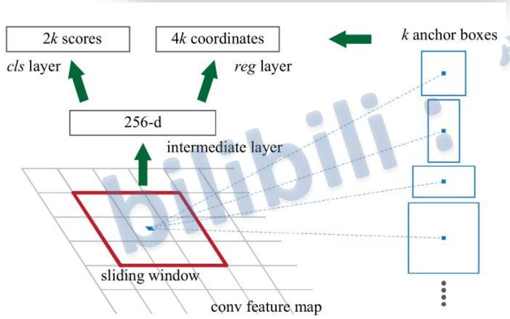
每个位置在原图上都对应有3x3=9 anchor

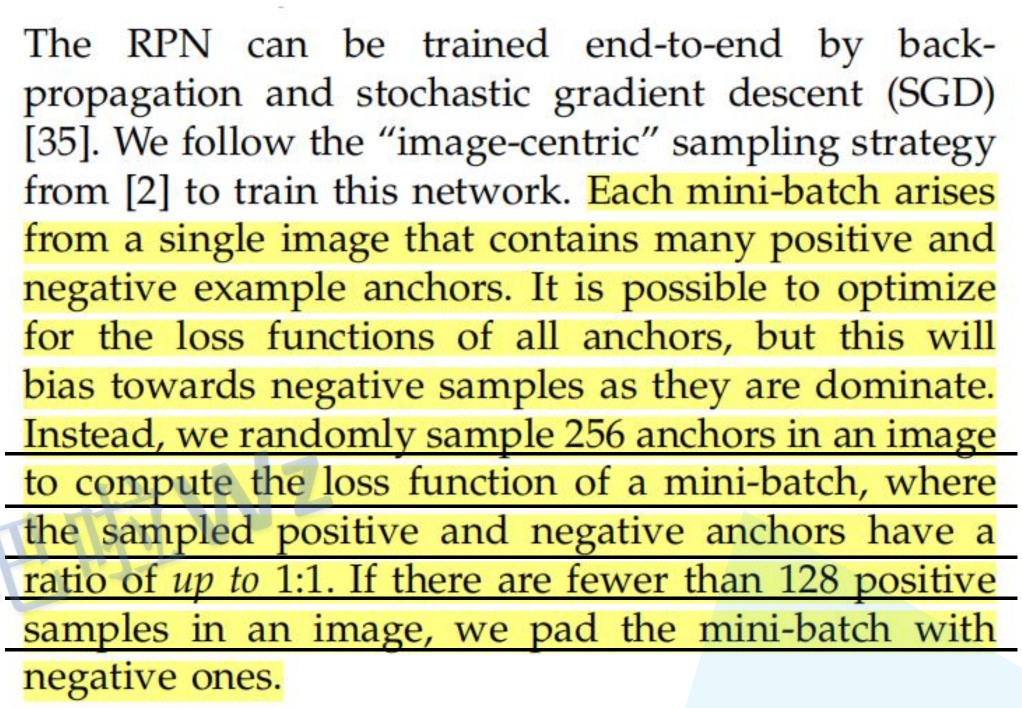
对于一张1000x600x3的图像，大约有60x40x9(20k)个anchor，忽略跨越边界的anchor以后，剩下约6k个anchor。对于RPN生成的候选框之间存在大量重叠，基于候选框的*cls*得分，采用非极大值抑制，IoU设为0.7，这样每张图片只剩2k个候选框。

# Faster R-CNN

训练数据的采样
(正样本，负样本)

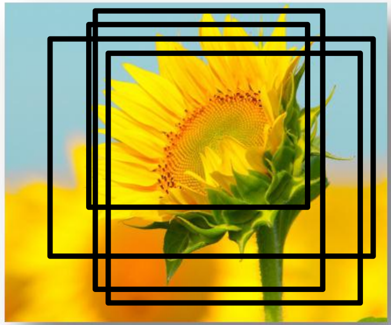


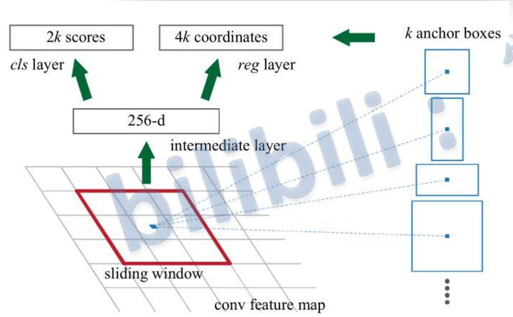


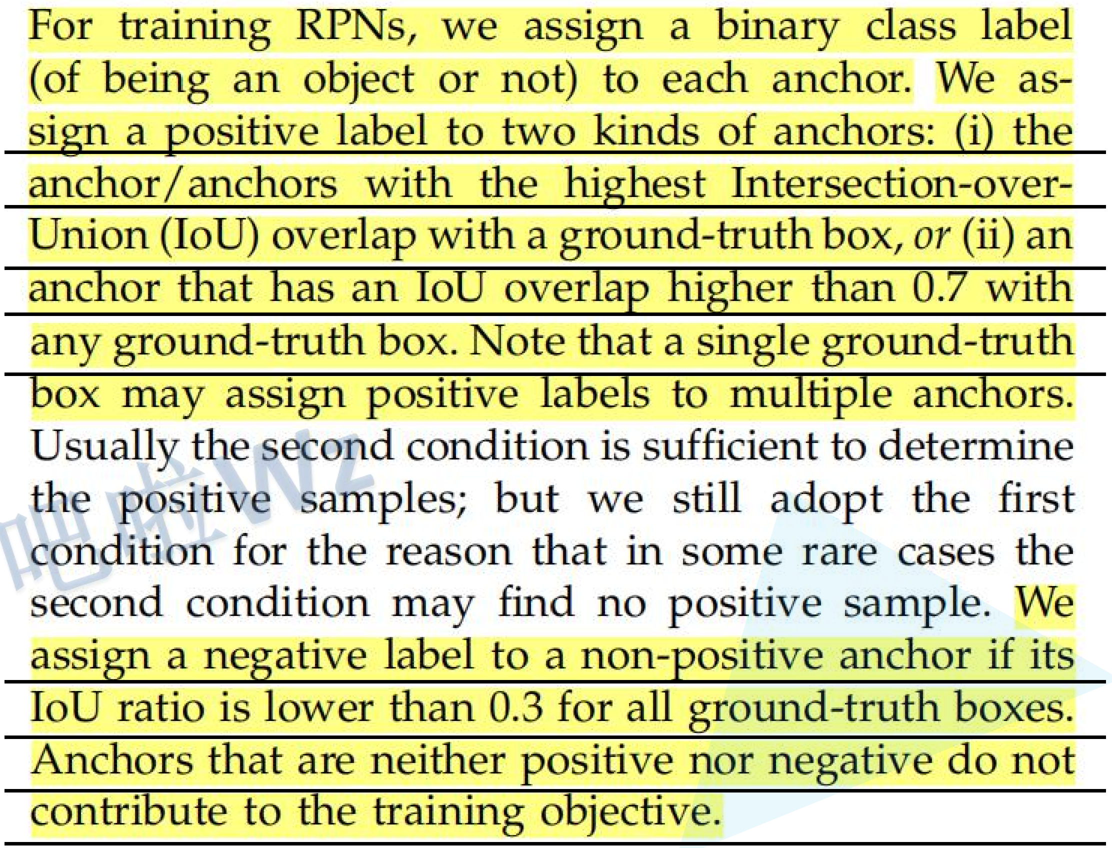

The RPN can be trained end-to-end by back-propagation and stochastic gradient descent (SGD) [35]. We follow the "image-centric" sampling strategy from [2] to train this network. Each mini-batch arises from a single image that contains many positive and negative example anchors. It is possible to optimize for the loss functions of all anchors, but this will bias towards negative samples as they are dominate. Instead, we randomly sample 256 anchors in an image to compute the loss function of a mini-batch, where the sampled positive and negative anchors have a ratio of *up to* 1:1. If there are fewer than 128 positive samples in an image, we pad the mini-batch with negative ones.

# Faster R-CNN

训练数据的采样
(正样本，负样本)







For training RPNs, we assign a binary class label (of being an object or not) to each anchor. We assign a positive label to two kinds of anchors: (i) the anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box, *or* (ii) an anchor that has an IoU overlap higher than 0.7 with any ground-truth box. Note that a single ground-truth box may assign positive labels to multiple anchors. Usually the second condition is sufficient to determine the positive samples; but we still adopt the first condition for the reason that in some rare cases the second condition may find no positive sample. We assign a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes. Anchors that are neither positive nor negative do not contribute to the training objective.

# Faster R-CNN

## RPN Multi-task loss

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

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

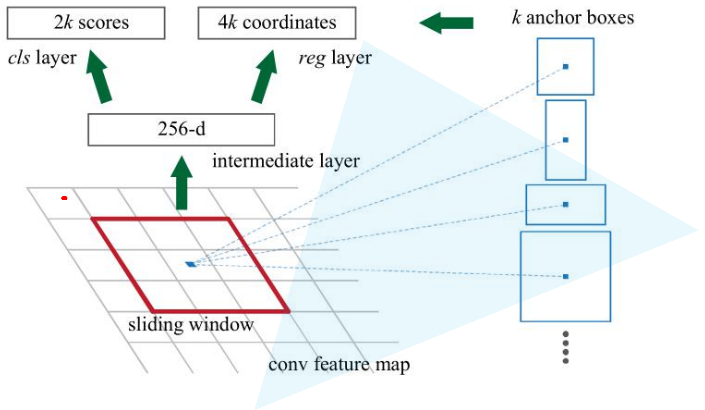
$p_i^*$当为正样本时为1,当为负样本时为0

$t_i$ 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$GT\,Box$的边界框回归参数

$N_{cls}$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

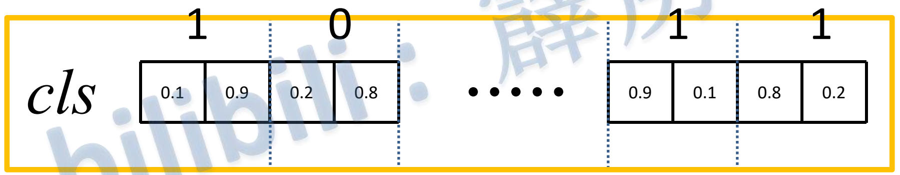


# Faster R-CNN

## RPN Multi-task loss

Softmax Cross Entropy

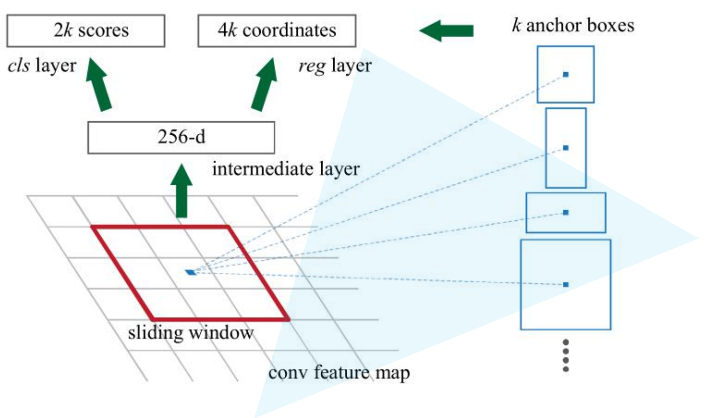
$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \overset{\text{分类损失}}{\boxed{\sum_i L_{cls}(p_i, p_i^*)}} + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$



$$L_{cls} = -\log(p_i)$$

$p_i$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



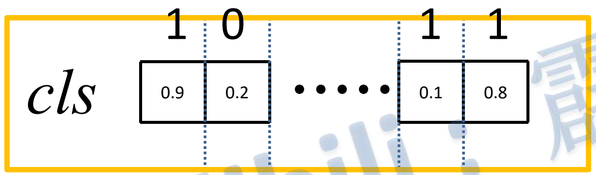
# Faster R-CNN

## RPN Multi-task loss

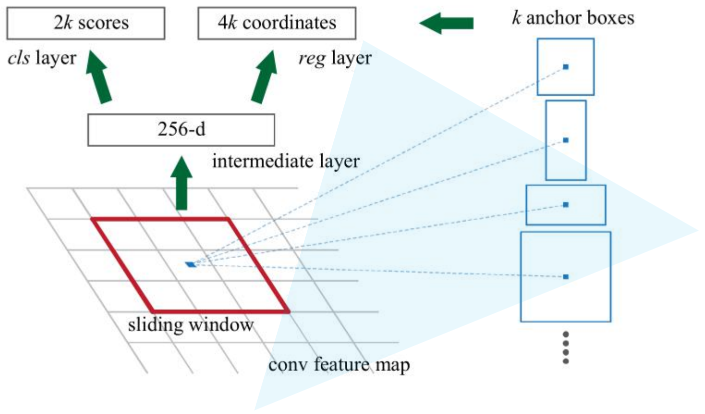
Binary Cross Entropy

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

分类损失



**注意：使用二值交叉熵损失，*cls* layer只预测k scores**



$$L_{cls} = -[p_i^* \log(p_i) + (1 - p_i^*)\log(1 - p_i)]$$

$p_i$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

# Faster R-CNN

## RPN Multi-task loss

边界框回归损失

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

$$L_{reg}(t_i, t_i^*) = \sum_i smooth_{L_1}(t_i - t_i^*)$$

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

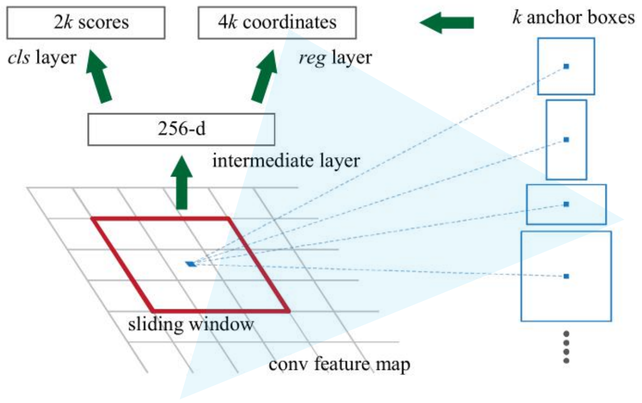
$$t_i = [t_x, t_y, t_w, t_h] \quad t_i^* = [t_x^*, t_y^*, t_w^*, t_h^*]$$

$p_i^*$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$GT\,Box$的回归参数

$$t_x = (x - x_a)/w_a,\quad t_y = (y - y_a)/h_a,$$
$$t_w = \log(w/w_a),\quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, t_y^* = (y^* - y_a)/h_a,$$
$$t_w^* = \log(w^*/w_a),\quad t_h^* = \log(h^*/h_a)$$



# Faster R-CNN

## Fast R-CNN Multi-task loss

分类损失　　边界框回归损失

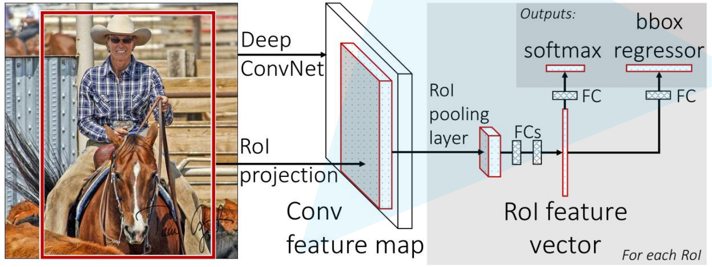
$$L(p,u,t^u,v) = \boxed{L_{cls}(p,u)} + \boxed{\lambda[u \geq 1]L_{loc}(t^u,v)}$$

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

$u$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

## Fast R-CNN Multi-task loss

分类损失

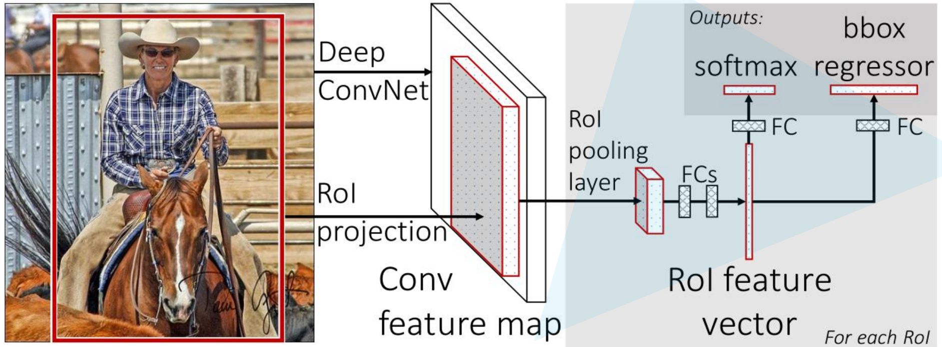
$$L(p,u,t^u,v) = \boxed{L_{cls}(p,u)} + \lambda[u \geq 1]L_{loc}(t^u,v)$$

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

$u$对应目标真实类别标签

分类损失

$$L_{cls}(p,u) = -\log p_u$$



# Faster R-CNN

## Fast R-CNN Multi-task loss

边界框回归损失

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

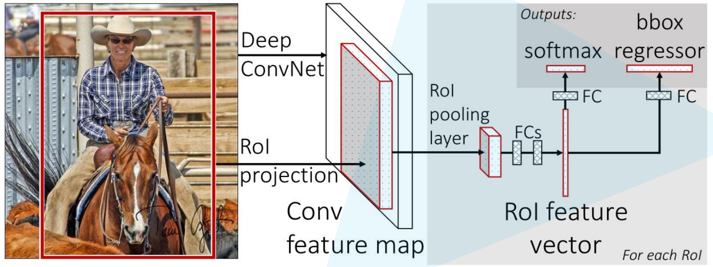
$[u \geq 1]$是艾弗森括号

$t^u$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$(v_x, v_y, v_w, v_h)$

$$L_{loc}(t^u,v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i)$$

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$



https://www.cnblogs.com/wangguchangqing/p/12021638.html

# 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

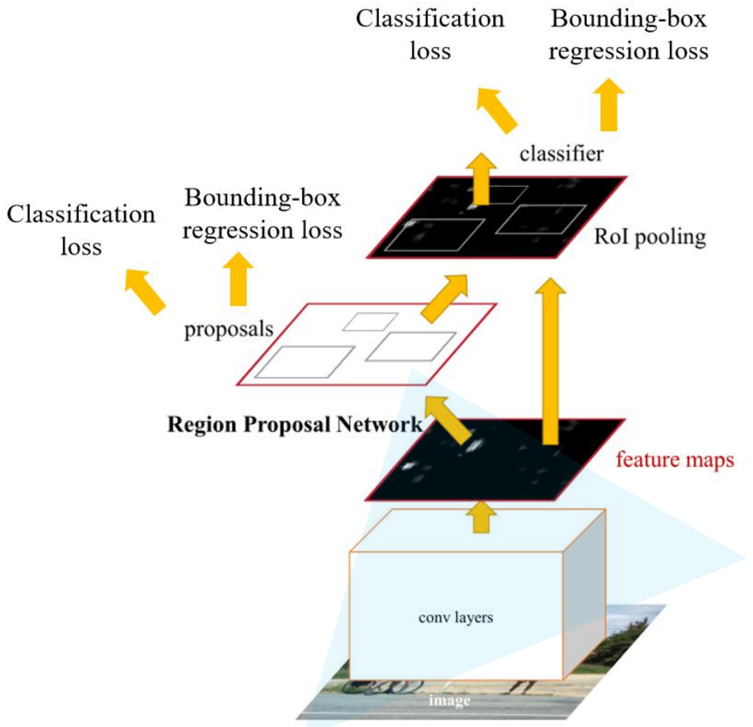
原论文中采用分别训练RPN以及Fast R-CNN的方法

(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。

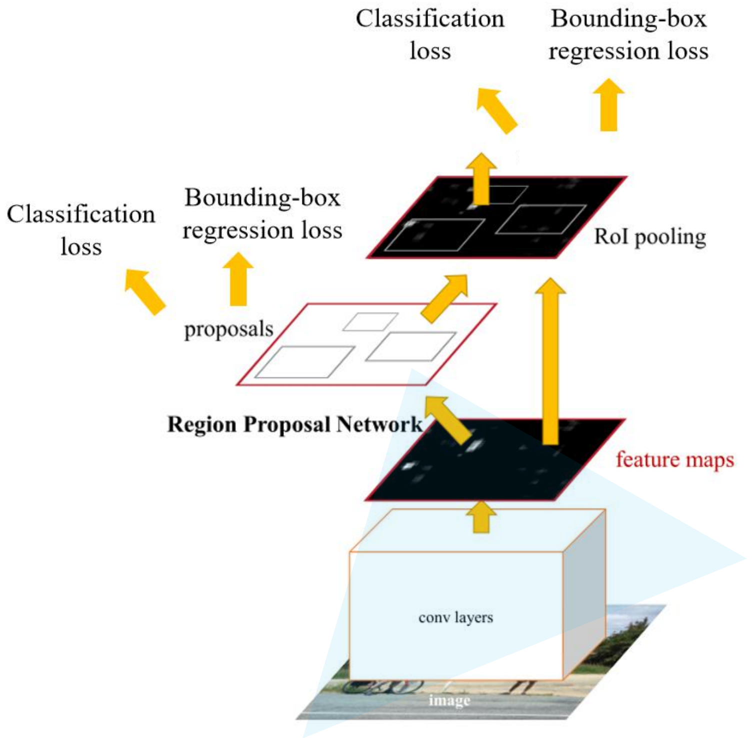


# Faster R-CNN

Faster R-CNN算法流程可分为3个步骤

- 将图像输入网络得到相应的**特征图**
- 使用RPN结构生成候选框，将RPN生成的候选框投影到特征图上获得相应的**特征矩阵**
- 将每个特征矩阵通过ROI pooling层缩放到**7x7大小的特征图**，接着将特征图展平通过一系列全连接层得到预测结果

RPN + Fast R-CNN



# Faster R-CNN

## Faster R-CNN框架

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

# Faster R-CNN

## Fast R-CNN框架

| |
|---|
| Region proposal(Selective Search) |
| Feature extraction<br>Classification<br>Bounding-box regression<br>(CNN) |

## Faster R-CNN

### R-CNN框架

| | |
|---|---|
| Region proposal(Selective Search) | |
| Feature extraction(CNN) | |
| Classification (SVM) | Bounding-box regression (regression) |

# Faster R-CNN

## R-CNN框架

| Region proposal(Selective Search) | |
|---|---|
| Feature extraction(CNN) | |
| Classification<br>(SVM) | Bounding-box regression<br>(regression) |

## Fast R-CNN框架

| Region proposal(Selective Search) |
|---|
| Feature extraction<br>Classification<br>Bounding-box regression<br>(CNN) |

## Faster R-CNN框架

| Region proposal<br>Feature extraction<br>Classification<br>Bounding-box regression<br>(CNN) |
|---|

## 沟通方式

**1.github**

https://github.com/WZMIAOMIAO/deep-learning-for-image-processing

**2.CSDN**

https://blog.csdn.net/qq_37541097/article/details/103482003

**3.bilibili**

https://space.bilibili.com/18161609/channel/index

尽可能每周更新