目标检测算法学习

> Tensorflow 官方目标检测API：https://github.com/tensorflow/models/tree/master/research/object_detection

一、PASCAL VOC2012 数据集介绍及使用

目前主要用COCO数据集

PASCAL VOC挑战赛 （The PASCAL Visual Object Classes ）是一个世界级的计算机视觉挑战赛，PASCAL全称：Pattern Analysis, Statical Modeling and Computational Learning，是一个由欧盟资助的网络组织。PASCAL VOC挑战赛主要包括以下几类：图像分类(Object Classification)，目标检测(Object Detection)，目标分割(Object Segmentation)，动作识别(Action Classification)等。

Table 1 The VOC classes

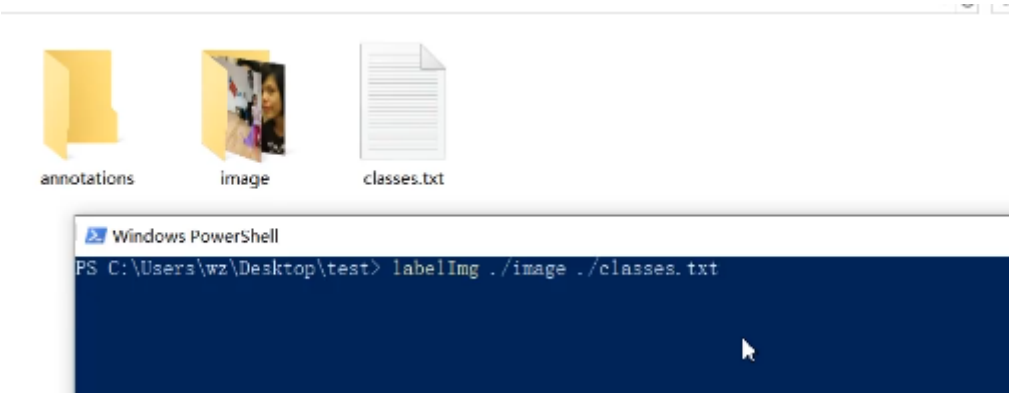
Vehicles	Household	Animals	Other
Aeroplane	Bottle	Bird	Person
Bicycle	Chair	Cat	
Boat	Dining table	Cow	
Bus	Potted plant	Dog	
Car	Sofa	Horse	
Motorbike	TV/Monitor	Sheep	
Train			

```
.
├── Annotations                所有的图像标注信息(XML文件)
├── ImageSets
│   ├── Action                 人的行为动作图像信息
│   ├── Layout                 人的各个部位图像信息
│   ├── Main                   目标检测分类图像信息
│   │   ├── train.txt          训练集 5717
│   │   ├── val.txt            验证集 5823
│   │   └── trainval.txt       训练集+验证集 11540
│   └── Segmentation           目标分割图像信息
├── JPEGImages                 所有图像文件
├── SegmentationClass          图像分割png图（基于类别）
└── SegmentationObject         图像分割png图（基于目标）
```

注意：2012年的 test数据集不公开

标注自己的数据集：

- 下载 labelImg （github直接搜索即可）
- 创建一个空文件夹，设置需要标注的标签，准备如下信息，在当前文件夹内打开该软件即可
- 进入界面，首先选择保存的位置（即annotations），之后即可框选图像来进行标注



使用自己标注的数据集（或直接修改API路径）：

- 将图像放入JPEGImages文件夹（API目录里）中
- 将标注文件放入对应的API目录里（即Annotations中）
- 自己生成对应的train.txt和val.txt文件放入对应位置

二、目标检测中的常见指标

TP(True Positive): IoU>0.5的检测框数量（同一Ground Truth只计算一次）
FP(False Positive): IoU<=0.5的检测框（或者是检测到同一个GT的多余检测框的数量）
FN(False Negative): 没有检测到的GT的数量

Precision: TP / (TP + FP) 模型预测的所有目标中，预测正确的比例
Recall: TP / (TP + FN) 所有真实目标中，模型预测正确的目标比例

查准率
查全率

AP: P-R曲线下面积
P-R曲线: Precision-Recall曲线

mAP: mean Average Precision, 即各类别AP的平均值

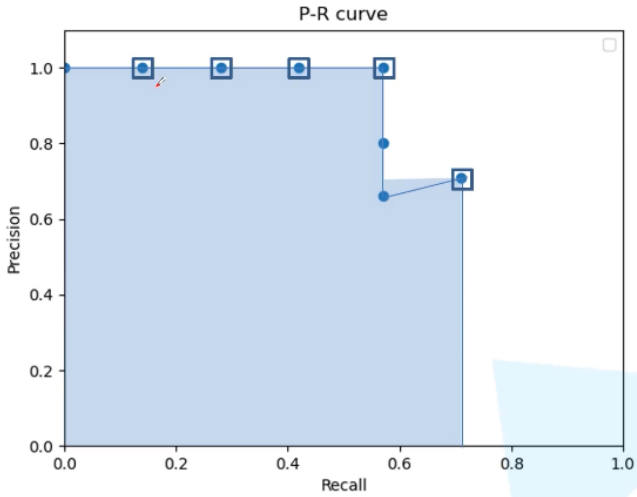
只使用查准率：如果有多个目标只检测到了一个且正确，那么查准率为1，很明显不准确。

只使用查全率（召回率）：检测到了所有存在的目标，但是还有很多多余的检测框，这也是不准确的。

P-R曲线：

- 计算查准率P和查全率R（一张图像一张图像计算）
- 将R相等时，P最大的保留，其他删除。然后以P为纵坐标，R为横坐标画图
- 如图所求即为一个类别的AP值（0.6694），mAP即为所有类别的AP平均值
- （所有参与计算的边界框均为通过极大值抑制后的边界框）

Rank	Precision	Recall
1	1.0	0.14
2	1.0	0.28
3	1.0	0.42
4	1.0	0.57
~~5~~	~~0.80~~	~~0.57~~
~~6~~	~~0.66~~	~~0.57~~
7	0.71	0.71



$$(0.14-0)\times1.0+(0.28-0.14)\times1.0+(0.42-0.28)\times1.0+(0.57-0.42)\times1.0+(0.71-0.57)\times0.71=0.6694$$

COCO评价指标：

COCO官网：https://cocodataset.org/#detection-eval

根据检测目标重点关注某几个指标，如前两类中的某两个等。

COCO Evaluation Result

```
Average Precision (AP):
  AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75          % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small            % AP for small objects: area < 32^2
  AP^medium           % AP for medium objects: 32^2 < area < 96^2
  AP^large            % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1            % AR given 1 detection per image
  AR^max=10           % AR given 10 detections per image
  AR^max=100          % AR given 100 detections per image
AR Across Scales:
  AR^small            % AR for small objects: area < 32^2
  AR^medium           % AR for medium objects: 32^2 < area < 96^2
  AR^large            % AR for large objects: area > 96^2
```

三、R-CNN（Rich feature hierarchies for accurate object detection and semantic segmentation）

RCNN算法流程可分为4个步骤

- 一张图像生成1K~2K个候选区域(使用Selective Search方法)
- 对每个候选区域，使用深度网络提取特征
- 特征送入每一类的SVM 分类器，判别是否属于该类
- 使用回归器精细修正候选框位置

R-CNN框架

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

论文阅读（Rich feature hierarchies for accurate object detection and semantic segmentation）

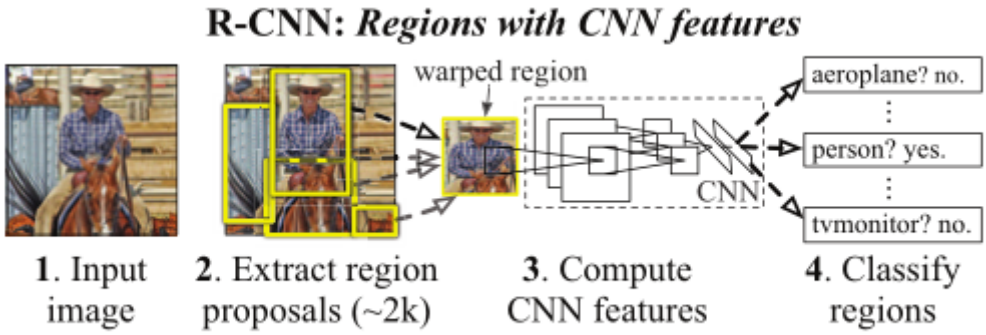


Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [34] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%.

- 选择性搜索（SS）算法提取约2k个建议框，并优先筛选（非最大值抑制）出一定数量的正样本，同时选定相同数量的负样本即背景。训练中为每一个类优化一个线性SVM
- 消融实验证明，仅使用卷积层提取特征的效果比加上全连接层效果要好

四、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

Fast R-CNN框架

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

Cross Entropy Loss 交叉熵损失

1. 针对多分类问题（softmax输出，所有输出概率和为1）

$$H = -\sum_i o_i^* \log(o_i)$$

2. 针对二分类问题（sigmoid输出，每个输出节点之间互不相干）

$$H = -\frac{1}{N}\sum_{i=1}^{N}[o_i^* \log o_i + (1-o_i^*)\log(1-o_i)]$$

其中o_i^*为真实标签值，o_i为预测值，默认log以e为底等于ln

论文阅读（Fast R-CNN）

网络：使用VGG16网络训练，代码开源（Caffe）：https://github.com/rbgirshick/fast-rcnn

数据集：PASCAL VOC 2012和COCO

R-CNN缺点：

- 训练是多级管道（SPPnet也是）
- 边界框回归训练和特征提取取存储耗费大量时间和空间
- 检测慢（为每个对象候选框进入网络，不共享计算）

Fast R-CNN优点：

- 检测质量（mAP）更高
- 训练是一体的，使用多任务损失
- 训练更新所有网络层
- 特征缓存不需要存储

网络结构：

- 整个图像和一组对象候选作为输入
- 首先用卷积层和最大池化层（最后一个池化层为RoI池化层）处理整个图像，产生卷积特征图
- 然后对于每个对象候选，RoI池化层从特征图中提取固定长度的特征向量
- 每个特征向量被馈送到全连接层，之后分支为两个输出层
- 其中一个输出层是softmax输出K+1类概率估计（1表示背景），针对每个RoI区域，下同
- 另一个输出层是为K个对象类回归输出4个实数值（即边界框参数）

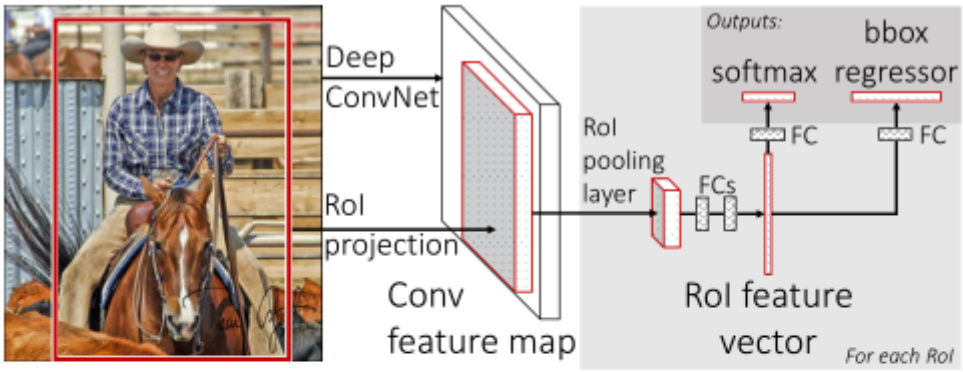


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

RoI池化层：

- RoI池化层使用最大池化将感兴趣区域（即经过卷积特征图的矩形窗口）特征转换为固定的HxW特征图（其中H、W是超参数）
- 每个RoI由四元组（r, c, h, w）定义，参数为左上角坐标及高和宽。

初始化模型及训练损失：

- 预训练的模型（如VGG16）中最后一个全连接层和softmax被替换为本网络的两个并列输出层；网络被修改为接受两个数据输入（如上所述输入）
- 微调使用小批量随机梯度下降（SGD）对图像分层采样并提取RoI区域，注意同一图像的感兴趣区域在前向和后向传播中共享计算和内存。——加快计算速度，减慢收敛速度。但是减少迭代次数总体达到了训练快和准确率高的目标
- 采用多任务联合训练模型，loss公式如下。【回归边界框优化方法参考论文：R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014. 1, 3, 4, 8】

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

in which $L_{\text{cls}}(p, u) = -\log p_u$ is log loss for true class u.

其中`[u>=1]`表示预测为对象时为1，否则为0（即非对象时无回归损失），λ=1

- 对于边界框损失，公式如下。

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i), \qquad (2)$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \qquad (3)$$

- 在检测中全连接层的计算将花费一半左右的时间（分类任务则很少），采用截断的SVD来压缩全连接层，最后使用两个全连接层分别放入两个输出端，加速计算。大幅提高速度，mAP下降很少。

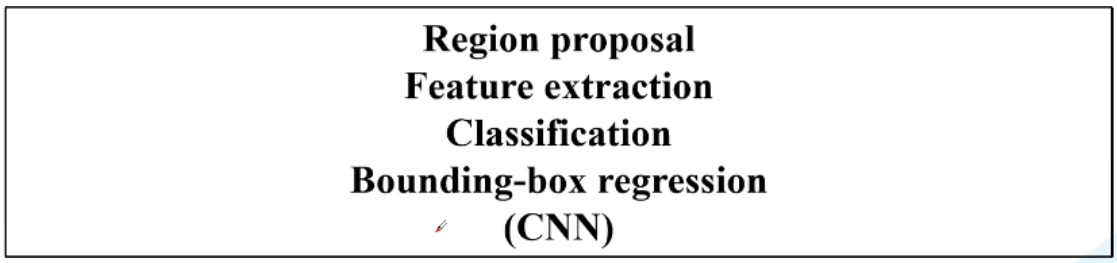
- 微调训练时冻结conv3以下的层，后面的层参与学习训练
- 实验证实多任务训练比分开训练效果要好；单尺度训练和图像金字塔（多尺度）训练效果差不多，但是单尺度训练较快
- 一次微调的softmax略好于SVM，但是softmax分类引入了"类竞争"，而非SVM的一对多

五、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

Faster R-CNN框架



论文阅读（Faster R-CNN: T owards Real-Time Object Detection with Region Proposal Networks)

源码（语言及框架同上）：https://github.com/rbgirshick/py-faster-rcnn

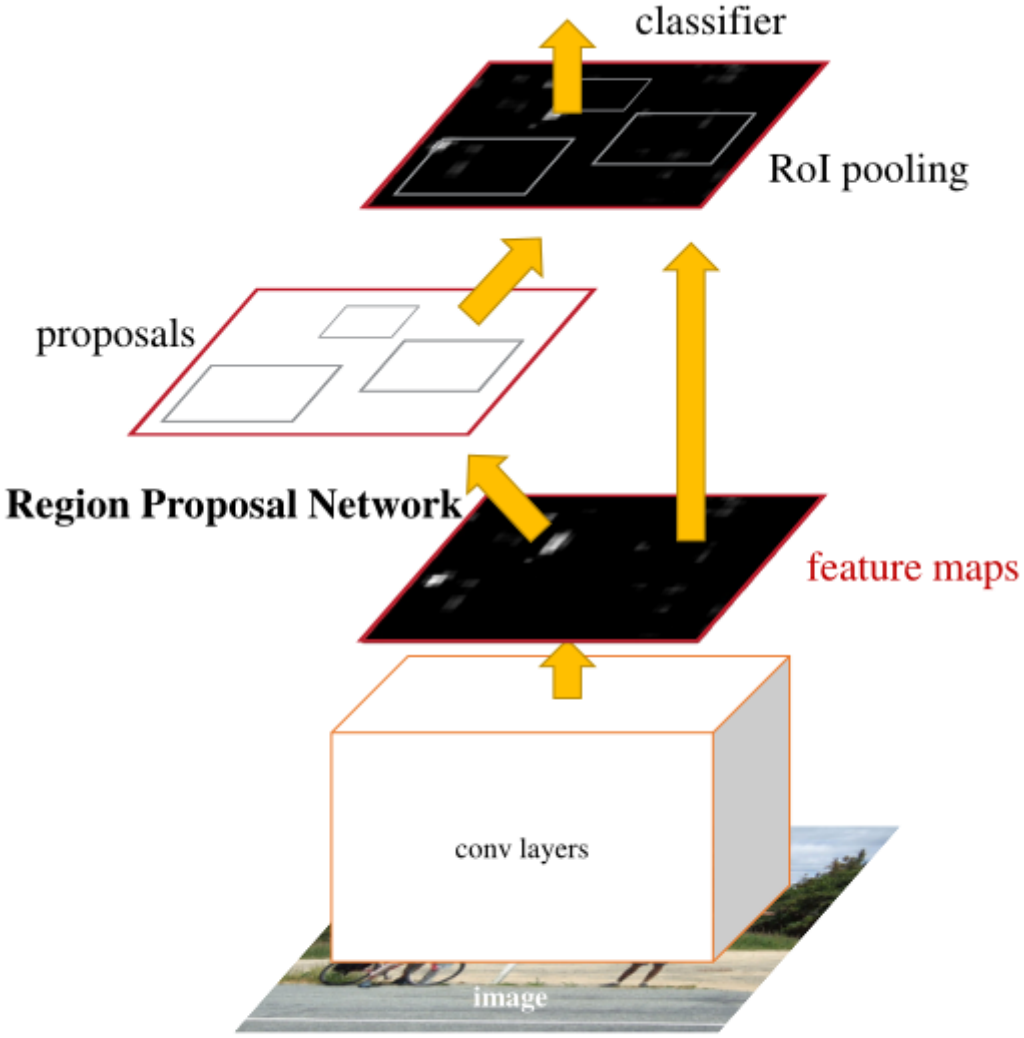


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

模型:

- 两个模块组成，一个是生成建议区域的深度全卷积网络（如ZFnet，即改进版AlexNet）；一个是使用建议区域的Fast R-CNN检测器
- 损失函数:

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

$$L_{reg}(t_i,t_i^*) = R(t_i - t_i^*)$$

其中 i 是小批量锚的索引； p_i 是锚 i 成为对象的预测概率； p_i^* 只有当为正样本时为1，否则为0； t_i 表示预测边界框的4个参数化坐标的向量； t_i^* 是真实边界框的坐标向量；分类层和回归层的输出分别由$\{p_i\},\{t_i\}$组成；设置 λ=10，因此 *cls* 和 *reg* 的权重大致相等，实验也证明最终结果对 λ 不敏感

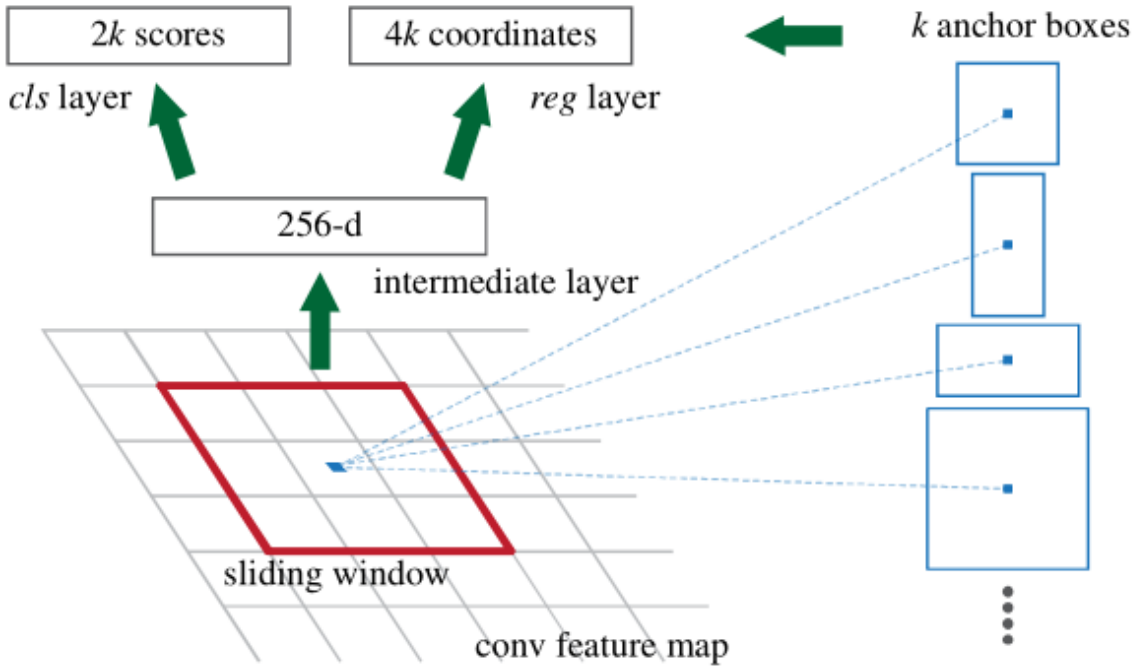
- 对于上述中的边界框回归损失：

$$
\begin{aligned}
t_{\mathrm{x}} &= (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t_{\mathrm{w}} &= \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}), \\
t^*_{\mathrm{x}} &= (x^* - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t^*_{\mathrm{y}} = (y^* - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t^*_{\mathrm{w}} &= \log(w^*/w_{\mathrm{a}}), \quad t^*_{\mathrm{h}} = \log(h^*/h_{\mathrm{a}}),
\end{aligned} \tag{2}
$$

  其中 x、y、w 和 h 表示窗口的中心坐标及其宽度和高度，而 x，x_{a}和 x^* 分别表示预测框、锚框和真实框

RPN（Region Proposal Networks）：

- RPN将任意大小的特征图作为输入，并输出一组矩阵对象建议，每个建议都有一个对象分数。因为最终目标是与Fast R-CNN对象检测网络共享计算，因此令两个网络共享一组公共的卷积层。
- 为了生成区域建议，在公共卷积层的最后一个共享卷积层输出的卷积特征映射上滑动一个网络（窗口），每个滑动窗口被映射到一个低维特征，并采用非最大值抑制减少冗余（NMS），其IoU设置为0.7，这样每幅图像大概剩下2k个建议框
- 每个滑动窗口位置，同时预测多个区域建议，每个位置的最大可能建议数表示为k。则建议生成层有4k个输出来编码k个矩阵的坐标，分类层输出2k个分数来估计每个建议是否是对象的概率。这k个矩阵框，成为锚。
- 一个锚点位于滑动窗口中心，并且边框面积（1282、2562、5122）和纵横比（1:1、1:2、2:1）相关。默认情况下，使用3个边框面积和3个纵横比，即每个滑动窗口产生k=9个锚框
- 对于正样本锚：1、IoU得分最大；2、IoU大于0.7（只有2没有时才使用1）
- 对于负样本锚：IoU低于0.3。最后使得抽样中正负样本1:1（即负样本补全与正样本一样多）
- 可通过反向传播和随机梯度下降（SGD）进行端到端训练



训练：

- 第一步：训练RPN，使用预训练模型进行初始化，并针对该任务进行端到端微调
- 第二步：使用上述生成的建议，送入检测网络（同样已被初始化）进行训练。而上述两步训练两个网络不共享卷积层
- 第三步：使用检测器网络来初始化RPN训练，但是固定共享卷积层，只微调RPN持有的层。这一步两个网络共享卷积层

- 最后：保持共享卷积层不变，仅仅微调检测器网络。两个网络共享相同的卷积层，形成一个统一的网络
- 网络具有预测比潜在感受野更大的对象，原因是如果只有物体中间是可见的，人们仍然可以粗略地推断出物体的范围
- 忽略了所有跨境锚，在测试中对于跨境建议框，将其裁减到图像边界
- 评估标准采用mAP，而不关注对象建议度量

代码实现 - 使用API接口

1、官方源码测试demo

Tensorflow目标检测API使用（三大步7小步）

2、训练自己的数据集

> 包括如何制作自己的数据集、如何使用官方目标检测API接口进行训练等

- 制作数据集：参考上述第一部分制作方法（或直接使用公开数据集）
- 使用接口训练：使用Tensorflow目标检测API训练自己的数据集

代码实现 - 使用keras模块

参考：TF2实现Faster R-CNN（不能对不同的目标进行区分，仅能将所有的待检测目标进行标框）

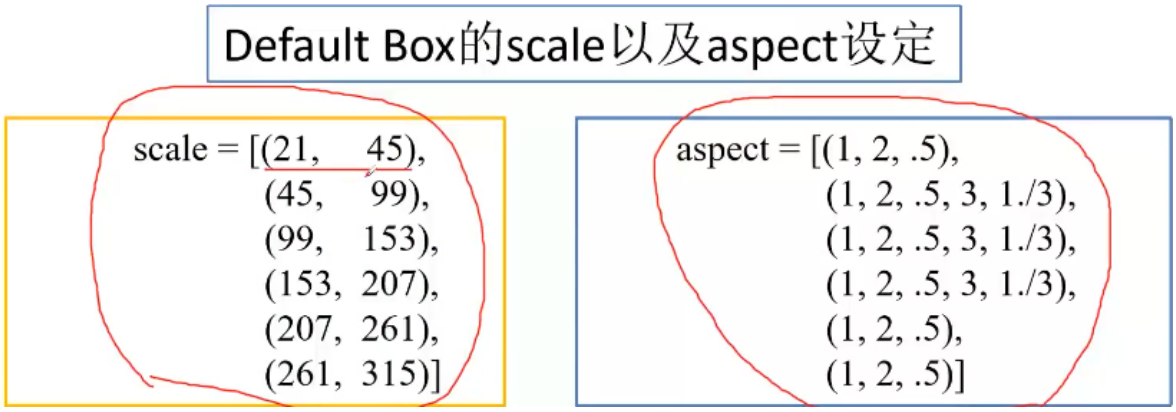
- 数据生成
  - 每张图像中的目标整理为图像（images，shape=[batch_size, setting.image_height, setting.image_width, 3]）、置信度（target_scores，shape=[batch_size, setting.grid_h, setting.grid_w, 9, 2]）、锚框（target_bboxes， [batch_size, setting.grid_h, setting.grid_w, 9, 4]）和掩码（ target_masks，[batch_size, setting.grid_h, setting.grid_w, 9]）。这里的锚框不是数据集中提供的，而是通过给定的9个锚框比例和真实标注框计算而得到的最符合该图像对象所对应的新生成锚框，之后预测的目标都是以该锚框为准。这里的掩码主要用来区分是否是前景/背景
- 模型创建
  - VGG+RPN网络的创建，输入一张图像，返回该图像中所有预测对象的置信度cls_scores和锚框cls_bboxes信息 *(rpn.py)*
- 训练过程
  - 每次迭代首先获取图像以及数据生成的共4个变量image_data, target_scores, target_bboxes, target_masks
  - 之后模型预测得到置信度和锚框信息pred_scores, pred_bboxes
  - 计算置信度损失（只关心前景损失）和锚框回归损失（只关心正预测框损失）
  - 梯度下降/优化训练
- 日志记录/模型保存

实现代码：Faster R-CNN实现链接

六、SSD（Single Shot MultiBox Detector）

Faster RCNN存在的问题：

- 对小目标检测效果很差（特征图只有一层，层次较高很难保留小物体的特征）
- 模型大，检测速度较慢



1、其中 scale 每个默认框有两个值分别表示 s_k 和 s_k+1（合起来就是为了求根号那个值，下面论文有具体介绍。同时当前的第二个等于下一个的第一个 scale）

2、其中第 1 和后两个 aspect 是4个（含比例为1的情况），而其他3个是6个默认框。数据由原论文给出

特征图层	特征图层的宽和高	默认框尺寸	默认框数量
特征图层①	38×38	21{1/2, 1, 2}; $\sqrt{21\times45}$ {1}	38×38×4
特征图层②	19×19	45{1/3, 1/2, 1, 2, 3}; $\sqrt{45\times99}$ {1}	19×19×6
特征图层③	10×10	99{1/3, 1/2, 1, 2, 3}; $\sqrt{99\times153}$ {1}	10×10×6
特征图层④	5×5	153{1/3, 1/2, 1, 2, 3}; $\sqrt{153\times207}$ {1}	5×5×6
特征图层⑤	3×3	207{1/2, 1, 2}; $\sqrt{207\times261}$ {1}	3×3×4
特征图层⑥	1×1	261{1/2, 1, 2}; $\sqrt{261\times315}$ {1}	1×1×4

论文阅读（SSD: Single Shot MultiBox Detector）

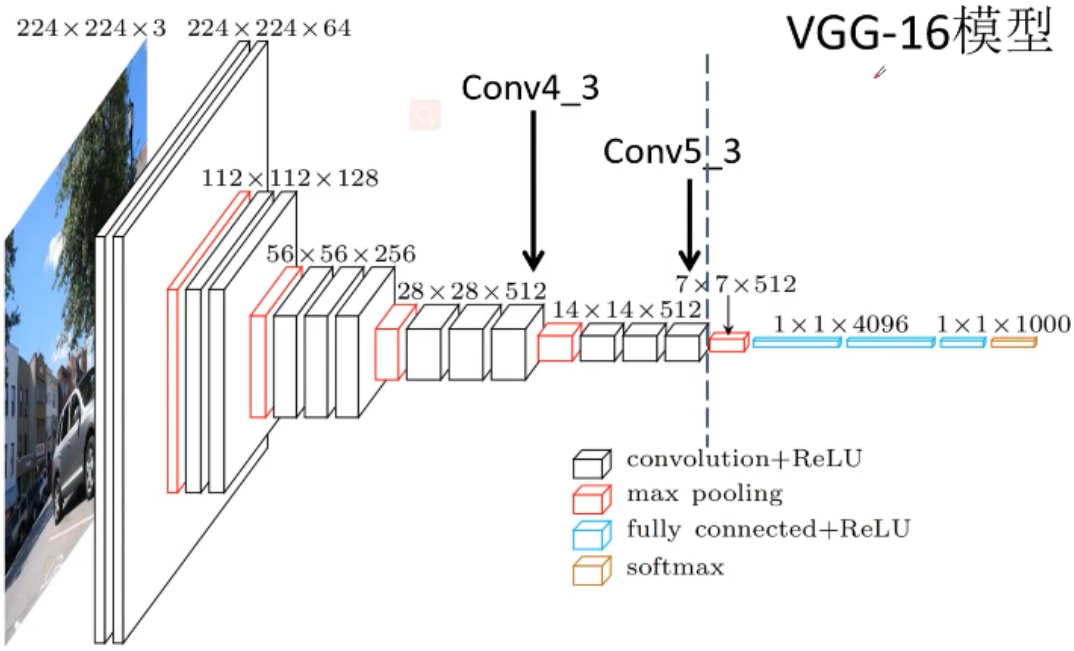
源码（Caffe）：https://github.com/weiliu89/caffe/tree/ssd

特点与贡献：

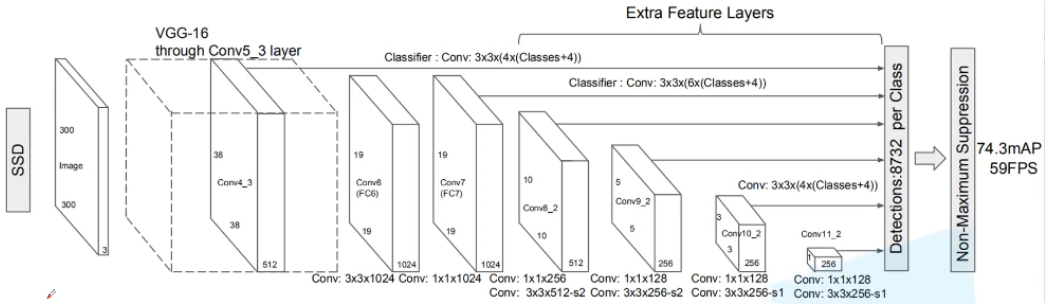
- 速度快，实时目标检测器，适用于多种类的单次检测器
- 消除边界框建议和对像素和特征的重采样
- SSD的核心是使用应用于特征图的小卷积滤波器来预测一组固定的默认边界框的类别分数和框偏移
- 为了实现高检测精度（同时对低分辨率输入图像具有较高的检测精度），从不同比例的特征图中生成不同比例的预测，并通过纵横比明确地分离预测

模型：

- 使用VGG-16作为主干网络



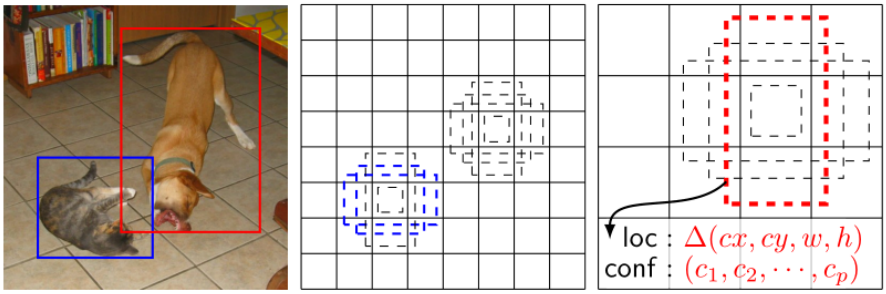
- 基于前馈卷积网络，该网络生成固定大小的边界框集合，并对这些框中存在的对象类实例进行评分，之后是非最大值抑制，以生成最终预测
- 将用于高质量图像分类的标准架构（不含分类层，即分类层之前截断的）称为基础网络，SSD网络在其中加入一些辅助结构：
  - 用于检测的多尺度特征图：即将卷积特征层添加到截断的基础网络的末端，而这些层的尺寸逐渐减小，允许在多个尺度上预测检测，并且对于每个特征层的卷积模型是不同的
  - 卷积预测器进行检测：每个特征层可以使用一组卷积滤波器产生一组固定的检测预测



对原VGG-16网络FC6和FC7进行卷积，并将pool5从 2x2xs_2 更改为 3x3xs_1 ，使用了padding填充；移除了所有的dropout层和FC8层

  - 默认框和纵横比：对于网络中的多个特征图，分别将一组默认边界框*（类似锚框，但是应用于多个尺寸的特征图）*与每个特征图单元相关联（以卷积方式对应）。在每个特征图单元中，预测相对于单元中默认框*（形状）*的偏移，以及指示每个框中存在类实例的每个类的分数。即对于给定位置的 k 个框中的每个框，计算 c 类分数和相对于原始默认框*（形状）*的4个偏移量，那么每个位置存在过滤器$(c+4)k$个，对于 mxn 的特征图一共有

$(c+4)kmn$个输出。



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

训练：

- 与基于区域建议的检测器训练相比，SSD中ground truth信息需要分配给固定检测器输出集合中的特定输出
- 匹配策略：最优交并比（即*IoU*或*jaccard*）。将 IoU>0.5 的进行匹配，这允许网络预测多个重叠默认框的高分，而不是要求它只选择重叠最大的一个
- 损失函数：定位损失与置信度损失的加权和。令 $x_{ij}^p = \{1, 0\}$作为一个指示器，即指示第 i 个默认框与 p 类的第 j 个ground truth框相匹配（或不匹配），则易得 $\sum_i x_{ij}^p \geq 1$。损失函数如下：

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{1}$$

其中N是匹配的默认框的数量，若 N=0 则设置 loss=0。定位损失是预测框 *l* 和真实框 *g* 之间的平滑L1损失，类似于Faster R-CNN，回归到默认框 d 的中心及宽 w 高 h 的偏移。置信度损失是多个类别置信度的最大软损失（通过交叉验证将 α 设置为 1）。

  - 定位损失（与Faster R-CNN几乎一样）：

$$\begin{aligned} L_{loc}(x, l, g) &= \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m) \\ \hat{g}_j^{cx} &= (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \\ \hat{g}_j^w &= \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right) \end{aligned} \tag{2}$$

  - 置信度损失：

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \tag{3}$$

选择默认框比例与纵横比：

- 通过利用来自单个网络中几个不同层的特征图进行预测，不仅可以达到与将图像变换尺度处理相同的效果，还可以在所有对象比例上共享参数

- 已知使用来自较低层的特征图可以提高语义分割质量，因为较低层捕捉输入对象的更多细节；从特征图添加全局上下文池化可以帮助平滑分割结果。受此启发，使用底层和高层特征图进行提取 *（实例展示见图1）*
- 通过设计默认框，可以达到特定的特征图能够响应对象的特定比例。假设使用 m 个特征图进行预测，则每个特征图的默认框比例计算如下 **（github上的大多数源码中 scale 非该公式计算得出，如上开始所列）**：

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m-1}(k-1), \quad k \in [1, m] \tag{4}$$

  其中最底层比例为0.2，最高层比例为0.9，中间每层存在间隔。

- 默认框的长宽比：$a_r \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$，由此可以计算每个默认框的宽度 $(w_k^a = s_k\sqrt{a_r})$和高度$(h_k^a = s_k/\sqrt{a_r})$。对于纵横比为 1 的默认框，其比例为 $s_k' = \sqrt{s_k s_{k+1}}$，因此每个特征图位置有 6 个默认框。
- 每个默认框的中心位置设置为 $\left(\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|}\right)$，其中 $|f_k|$是第 k 个正方形特征图的大小 $i, j \in [0, |f_k|)$

硬负面采集：

大多数默认框是负数，这时候对每个默认框的置信度损失进行排序，之后选择最高的多个置信度损失默认框作为负样例，使得负样例与正样例的比例最大为3:1

数据增强：

为了使模型对各种输入对象大小和形状更加稳健，每个训练图像都通过以下选项之一进行随机取样：

- 使用整个原始图像
- 选取一个部分，使其最小 IoU 为0.1、0.3、0.5、0.7或0.9
- 随机裁减一部分（可被认为是放大操作，增强了对小目标的检测性能）

每个样本的大小是原始图像大小的[0.1, 1]，纵横比在 0.5 到 2 之间，如果 ground truth 的中心在采样的块中，则保留它的重叠部分。之后每个采样被重新调整到固定的大小，并且以0.5的概率水平翻转

实验细节：

- 使用Xavier初始化方法，即为了使得网络中信息更好的流动，每一层输出的方差应该尽量相等。
- 不同层默认框数量不一样，如本节第一部分图所示

模型分析：

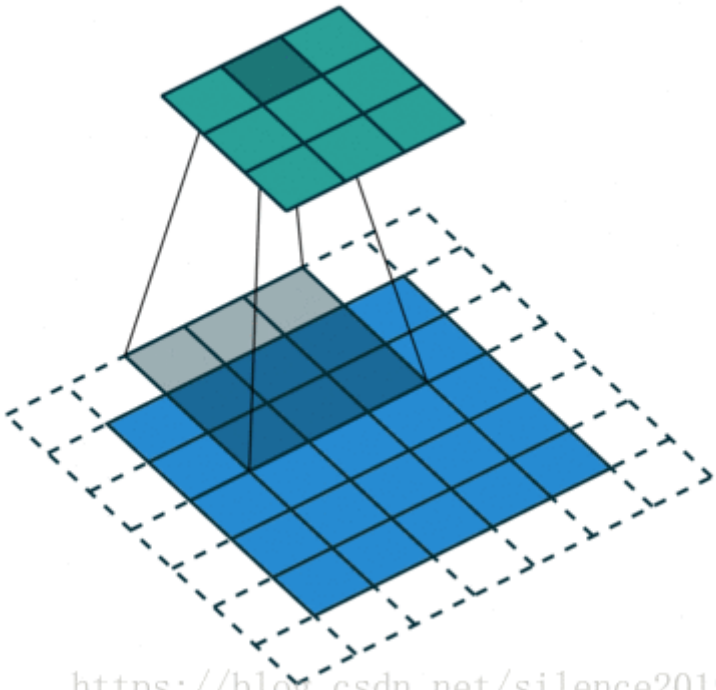
- 数据增强至关重要：使用了更广泛的抽样策略

	SSD300				
more data augmentation?		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔			✔	✔
use atrous?	✔	✔	✔		✔
VOC2007 `test` mAP	65.5	71.6	73.7	74.2	**74.3**

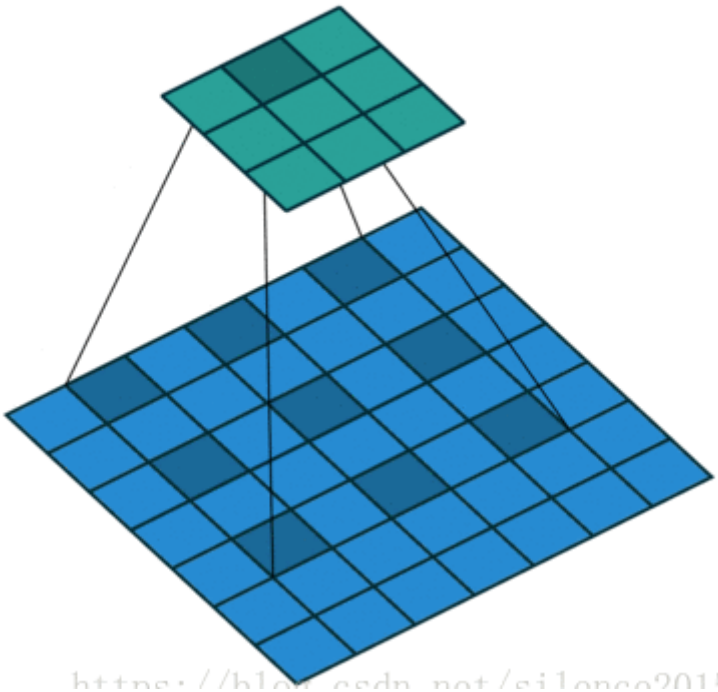
Table 2: **Effects of various design choices and components on SSD performance.**

- 默认框的形状越多越好：每个位置使用6个默认框
- 使用Dilated/Atrous卷积：可以保证在尺度大小变换后进行池化仍然维持相同的感受野，且速度更快。

标准卷积方式



空洞卷积方式



- 不同分辨率的多个输出层更好：在不同的输出层使用不同比例的默认框（依次减少输出层则精度会下降），实验也证明在不同的层上展开不同比例的默认框是至关重要的

代码实现 - 使用slim模块（暂未学习）

源码：https://github.com/balancap/SSD-Tensorflow

讲解：https://blog.csdn.net/qq1483661204/article/details/79776065

暂未学习

七、YOLO v1

论文阅读（You Only Look Once:Unified, Real-Time Object Detection）

不足：

- 定位误差大，但是在背景下预测假阳性可能性小
- 检测精度不是最好的
- 难以精确定位小目标

优点：

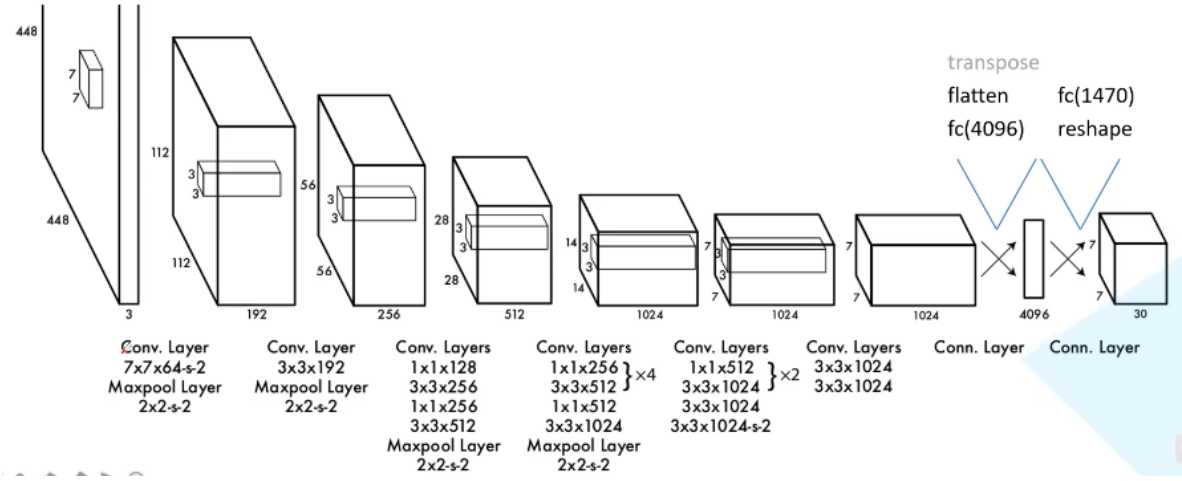
- 速度快，将目标检测作为一个回归问题，回归到空间分离的边界框和相关的类概率。
- 与滑动窗口和基于区域建议不同，在整个训练和测试期间看到整个图像，因此它隐含地编码了关于类及其外观的上下文信息
- 学习目标的一般表现，当应用于新领域或意外输入时，不太可能崩溃

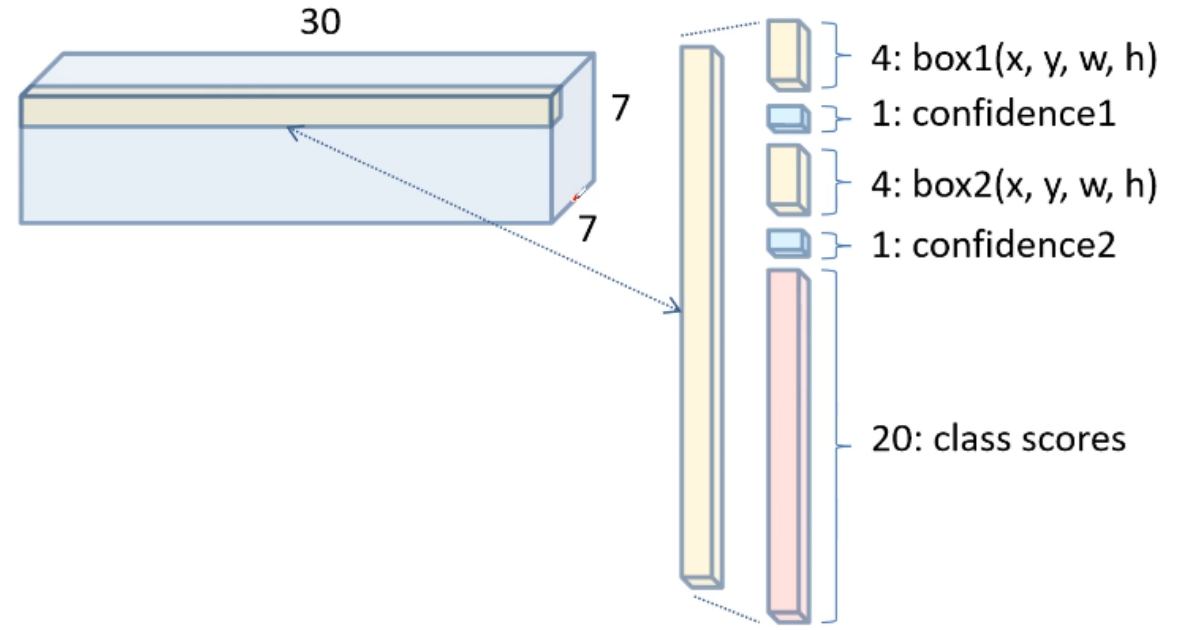
模型思想：

- 使用整个图像的特征来预测每个边界框，还可以同时预测图像所有类别的所有边界框
- 将一幅图像分成SxS个网格，如果某个目标的中心落在这个网格中，则这个网格就负责预测这个目标
- 每个网格要预测B（一般为2）个bounding box，每个bbox除了要预测位置之外，还要附带预测一个confidence值（表示认为预测的准确度 $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$.）。即每个边界框由5个预测组成（x，y，w，h+置信度）
- 每个网格还要预测C个类别概率 $\Pr(\text{Class}_i|\text{Object})$，最后测试时，将条件类概率和单个网格置信度预测相乘（表示每个网格的特定类别的置信度得分）：

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \quad (1)$$

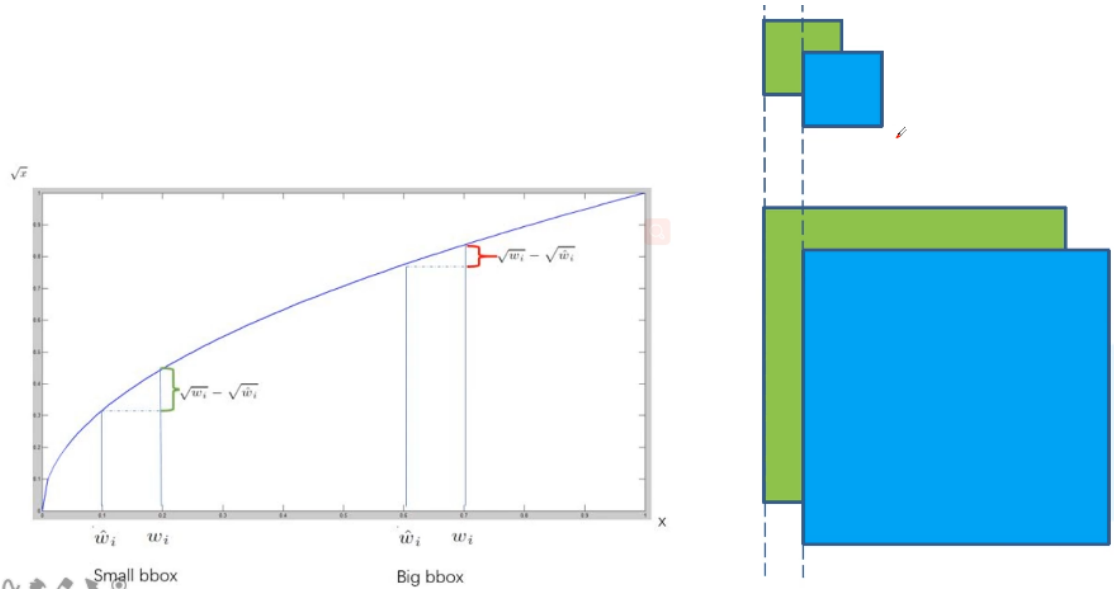
（最后输出向量为 S x S x B (B x 5 + C) 的大小）





训练：

- 其中边界框的宽高和坐标偏移进行规范化处理，值在0~1直接，这样使得中心点坐标不偏离当前网格区域
- 预测边界框宽度和高度的平方根，而不是直接预测宽度和高度，为了使得误差度量应该反映出大边界框里的小偏差比小边界框的小偏差更重要



（损失加根号可以在偏移同样的情况下，小目标的损失更大，事实也是如此）

- 每个网格单元有多个边界框，只需要一个边界框预测器来负责每个对象。指定一个预测器来“负责”预测一个对象，基于该对象的预测具有最高的IoU

局限性：

- 每个网格单元只能预测两个框，并且只能有一个类。对于多个小物体聚集预测不好，如一群鸟
- 模型从数据中预测边界框，很难推广到新的或不常见纵横比的对象
- 定位不准确（由于不同大小边界框对错误的影响不一样）

第四章后面的实验与对比没有看

八、YOLO v2

主干网络：Darknet-19（使用448x448尺度图像）

相对YOLO v1所做的尝试：

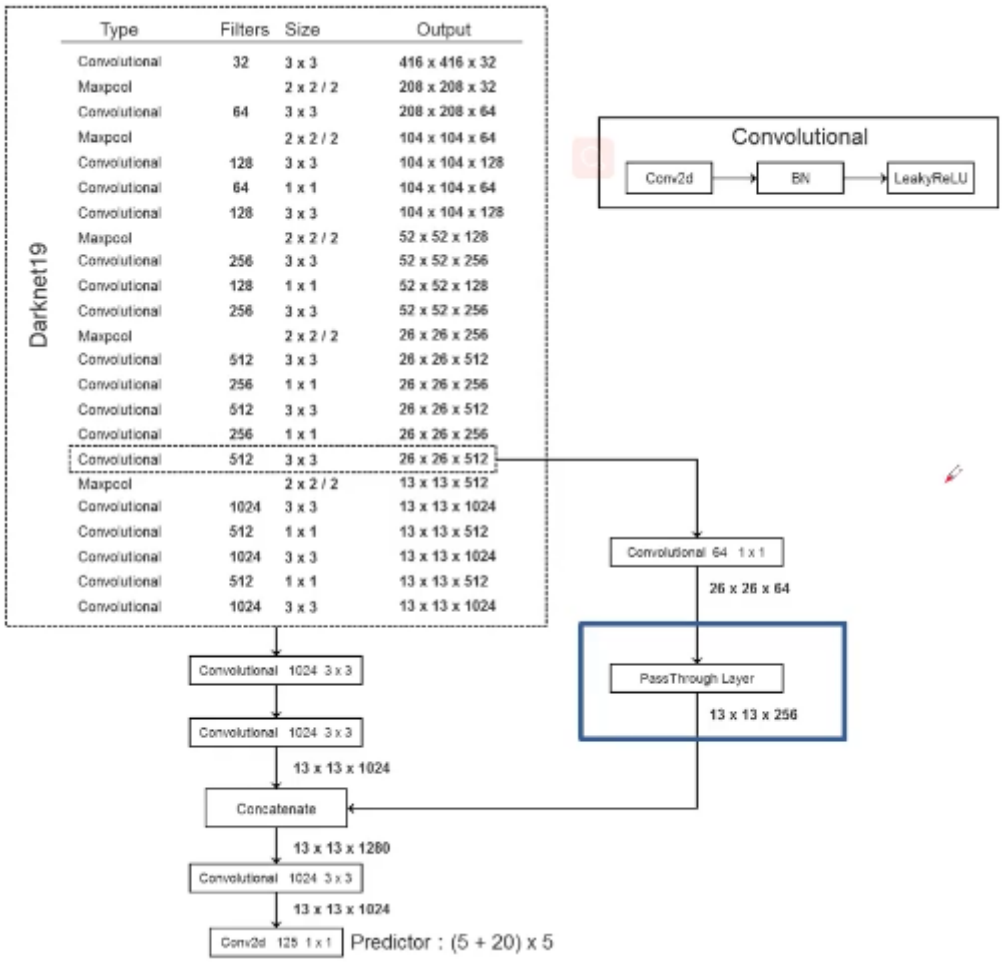
- Batch Normalization：可以消除其他形式的正则化需要，在不过度拟合情况下可以移除dropout
- High Resolution Classifier：采用更大分辨率图像，提高4% mAP
- Convolutional With Anchor Boxes：预测偏移而不是坐标，将类预测从空间位置中分离出来，取而代之的是预测每个锚框的类和对象。使用锚框降低了mAP，但是召回率有了大的提升
- Dimension Clusters：采用 k-means 聚类获得bbox，其中不使用欧几里得距离，因为不同尺度的边界框度量不准确，采用GIoU加快收敛：

$$d(\text{box}, \text{centroid}) = 1 - \text{IOU}(\text{box}, \text{centroid})$$

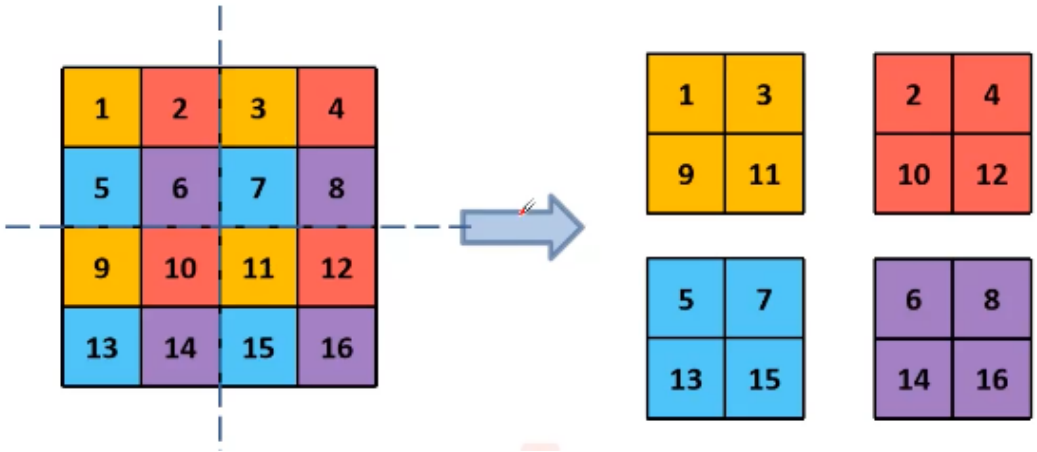
- Direct location prediction：论文公式有误（$\begin{aligned} x &= (t_x * w_a) - x_a \\ y &= (t_y * h_a) - y_a \end{aligned}$ 应该是加号）。该方法的目的是让每个anchor (prior)去负责预测目标中心落在某个grid cell区域内的目标，不发生过多的偏移。如下公式，由于限制了位置预测，提高了模型的准确性。

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \\ Pr(\text{object}) * IOU(b, \text{object}) &= \sigma(t_o) \end{aligned}$$

- Fine-Grained（细粒度） Features：将高层特征图与底层特征图进行融合（即通过PassThrough层进行深度拼接），提高检测小目标效果



PassThrough Layer (W/2, H/2, Cx4)



- Multi-Scale Training：为提升鲁棒性，将图像进行多尺度训练（每10个批次则随机选择预定图像尺度）

论文阅读（YOLO9000: Better, Faster, Stronger）

源码：https://pjreddie.com/darknet/yolo/

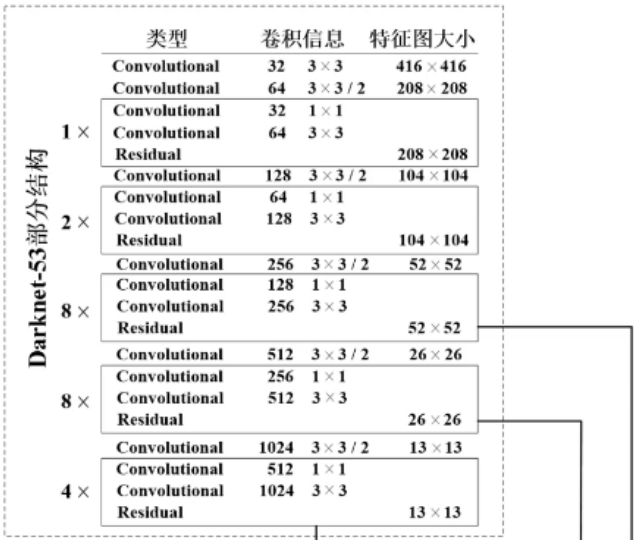
特点和主干网络如上介绍

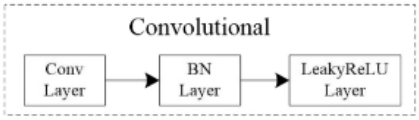
数据增强：包括随机裁减、旋转、色调、饱和度和曝光偏移等

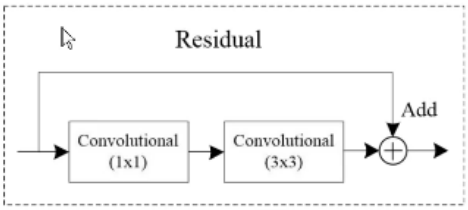
九、YOLO v3

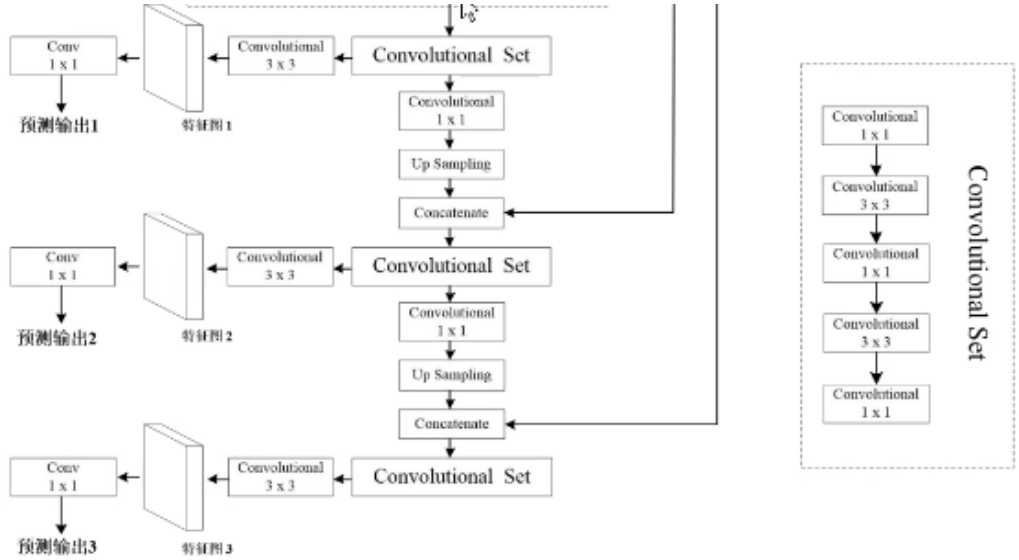
主干网络：Darknet-53

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









论文阅读（YOLOv3: An Incremental Improvement）

训练：

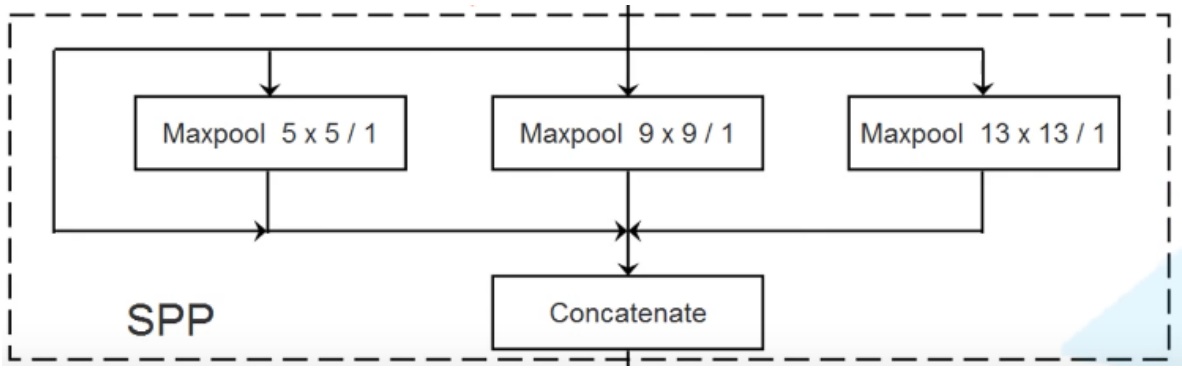
- 使用二元交叉熵损失进行类预测，对于多标签时很有帮助（如一个对象属于两个类别）

使用过的无用的改进（对于该网络）：

- 锚框偏移预测，稳定性降低
- 线性而非逻辑激活预测，mAP下降
- Focal loss：mAP下降。含义如下章。
- 双IoU阈值：如Faster R-CNN中用到过。

十、YOLO v3 SPP

网络结构：将YOLO v3中第一个Convolutional Set（上节图）替换为SPP结构，非SPPnet，实现了不同尺度的特征融合



Mosaic图像增强：图像拼接，增加数据的多样性，增加目标个数，BN能一次性统计多张图像的参数

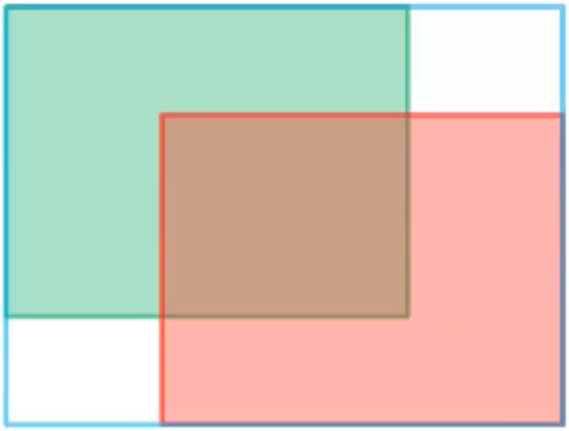
IOU Loss：

- IoU Loss：能够很好的反应重合程度，具有尺度不变形。但是当不相交时，loss为0。收敛慢
- GIoU Loss：当两个边界框水平或垂直相交时，退化为IoU。定位不准确 *(A^c表示下图中蓝框的面积，u表示预测框和真实框的并集)*

Generalized IoU

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

$$-1 \le GIoU \le 1$$



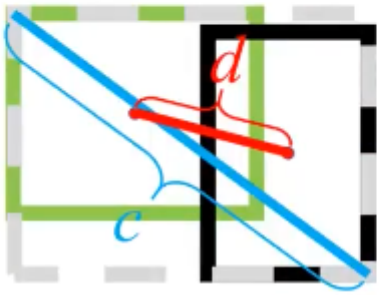
$$L_{GIoU} = 1 - GIoU$$

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

- DIoU Loss：能够直接最小化两个boxes之间的距离，因此收敛速度更快，定位更准确

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

$$-1 \le DIoU \le 1$$



- CIoU Loss：一个优秀的回归定位损失应该考虑重叠面积、中心点距离和长宽比。DIoU Loss没有考虑长宽比

Complete-IoU

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

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

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

Focal loss：主要适合单阶段目标检测模型中正负样本不平衡问题。对于公式$(1-p)^{\gamma}$或p^{γ}，能够降低易分样本的损失贡献（y=1表示正样本，p_t越大表示越容易区分），但是易受噪音干扰，即易受标注错误的样本

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

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

视频参考（其他博客及源码链接在对应出现位置）：

https://www.bilibili.com/video/BV1fT4y1L7Gi