University of Chinese Academy of Sciences

学士学位论文

基于 FreeAnchor 算法的目标检测研究

作者姓名：马金戈

指导教师：叶齐祥

学位类别：工学学士

专　　业：电子信息工程

学院（系）：电子电气与通信工程学院

2020 年 4 月

FreeAnchor algorithm based SSD detector

A thesis submitted to
University of Chinese Academy of Sciences
in partial fulfillment of the requirement
for the degree of
Bachelor of electrical engineering
in Apr. 2020

By
Jinge Ma

Supervisor: Professor Qixiang Ye

摘 要

在传统的基于CNN的单阶段检测器中，对于锚框和目标的匹配（matching）过程是基于锚框与真实边界框的交并比的，但这种直觉性的锚框匹配方式在处理长条，偏心或者遮挡的物体时具有局限性，因此我们采用了FreeAnchor算法对现有的SSD检测网络进行改进。

FreeAnchor的目标是学习最能分类和定位的特征。FreeAnchor可以和基于CNN的检测器随意融合。SSD检测器是单阶段（one-stage）检测器的代表，其检测速度高于two-stage检测器，但在其完善之前检测精度不及two-stage检测器。本毕业设计将SSD检测器的骨干网络与FPN，Focal Loss，FreeAnchor结合后，其精度和速度均超过了传统的单阶段检测器，并且在检测长条、偏心、密集物体上更具优势。

关键词：单阶段检测器，目标检测，计算机视觉，人工智能

Abstract

Now CNN-based algorithms use IoU(Intersection-over-Union) to allocate anchor s to objects. We propose a method which breaks the IoU limitations and allows free all ocation of anchors. Our method, called FreeAnchor, upgrades manual anchor assignm ent to "free" anchor matching.

The goal of FreeAnchor is to learn the best features which can be classified and l ocated. FreeAnchor can be combined with any CNN-based detectors.

SSD detector is the representative of single-stage detector. Its speed is higher than the two-stage detector, but the accuracy is lower than the two-stage detector before it is improved.In this thesis, combined with FreeAnchor, accuracy and speed of SSD both exceed single-stage detector, and it performs better on slender, non-central and densely-placed objects.

Key Words: single-stage detector, object detection, computer vision, artificial intelligence

目 录

1 第一章 绪 论

1.1 研究背景

计算机视觉作为一门新兴学科，主要使用计算机来代替人眼对特定图像中的目标进行识别，从而大大提升工作效率并解放劳动力。目前已经有很多学术界的成果走向了应用端，例如人脸识别，自动驾驶和交通监控等。此外未来工业产品检测、智能导航、视频监控等方面技术的推进也需要计算机视觉的发展，故对于计算机视觉的研究对于未来各项技术的发展显得尤为重要。

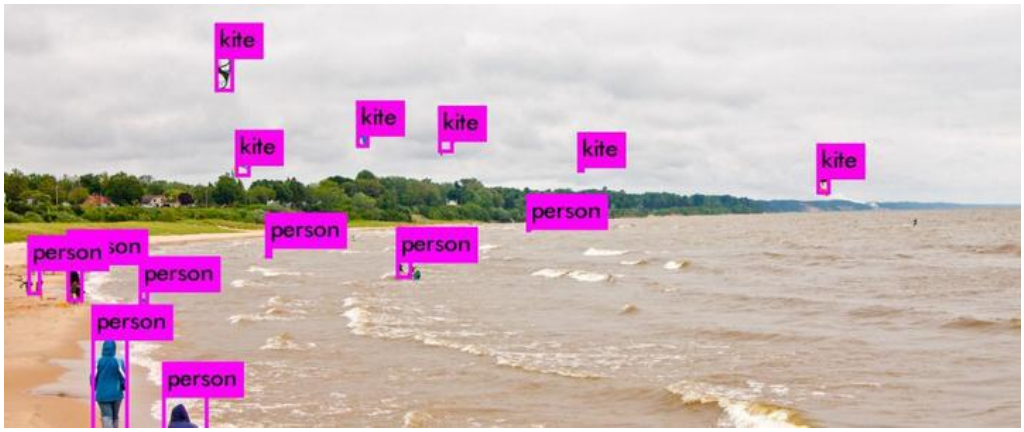

图 1：目标检测效果示意图

如图 1，目标检测则为很多计算机视觉任务的基础，在学术界已有十多年的研究历史。目标检测的任务是给出图片中的物体的具体类别和精细位置。通过神经网络提取图片的特征，使用具有特殊结构和功能的网络组件对特征图进行处理，最终得到网络能够识别的物体的分类与预测框的坐标。随着 RCNN 的提出，目标检测算法逐渐转向基于 CNN 的深度神经网络检测器的构建，目标检测算法也从最初提出的 SIFT,HOG 算法，转向 RCNN、Fast RCNN 算法，再到后面的 YOLO、SSD 系列算法，其最先进算法的检测速度和精度都在不断地提升。但是在提出之初，单阶段检测器的检测速率虽然快于双阶段检测器，但双阶段检测器的候选区域法却在精度上高于单阶段检测器。随着 Focal Loss 解决了正负样本和难易区分对象在误差函数中的优化，单阶段检测器无论是在速度还是精度上均超过了双

阶段检测器。但是，现有算法在实现部分任务，例如识别长条，偏心，密集遮挡的物体时，仍然存在困难。

1.2 研究现状

目标检测的研究发展历大致按照双阶段检测器→单阶段检测器的顺序，具体来说是 RCNN， fast RCNN，faster RCNN 再到 YOLO 和 SSD。

1.2.1 RCNN[1]

介绍 RCNN 之前，先介绍一下早期的计算机视觉算法。早在 1989 年，LeCun 已经提出利用卷积神经网络（Convolutional Neural Network，简称 CNN）对小尺度图片进行分类，在之后的时间里，激活函数、池化、归一化、正态化的处理在神经网络中进一步成熟。但 RCNN 之前的最好的目标检测算法并没有使用神经网络，而归属于传统的数字图像处理，例如 SIFT 算法和 HOG 特征提取。

2012 年世人重新将目光投在卷积神经网络之上——Krizhevsky 等人提出的 AlexNet 在 ImageNet 举办的 ILSVRC 目标识别挑战赛中夺得第一名，而 RCNN 的想法就在此基础之上建立：RCNN 的算法结构如图 2 所示，利用候选区域提取算法（论文中使用的是 Selective Search 算法）从图片中得到 2000 个候选区域，再将每个区域输入 CNN 网络，并提取特征向量，之后对特征向量使用支撑向量机（SVM）得到分类结果，最后使用 NMS 抑制提取局部极值，得到最终输出结果[1]。

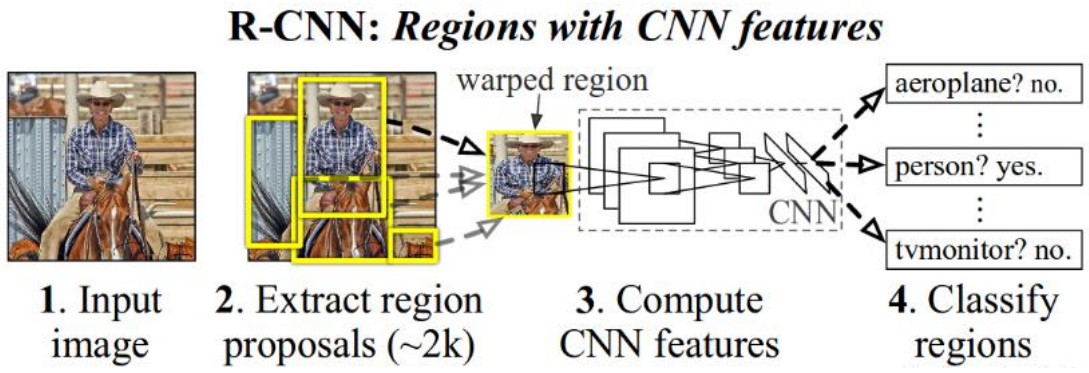

图 2：RCNN 检测网络结构图[1]

1.2.2 Fast RCNN[2]

2015 年，Fast RCNN 被提出以改善 RCNN 中存在的一些问题：测试速度慢、训练速度慢和训练所需空间大。Fast RCNN 的网络结构如图 3 所示。

测试、训练速度慢：RCNN 提取特征时，候选框重合的部分很多，会造成计算的繁琐重复。Fast RCNN 对单张图像进行 Normalization 后输入 CNN，在输出层之前的的几层加入候选框信息，处理每个候选框[2]。

训练所需空间大：RCNN 中的 classification 和 regression 需要提取很多特征用以训练。在 Fast RCNN 中，classification 和 regression 的偏移通过神经网络实现，减少了所需要的存储空间。

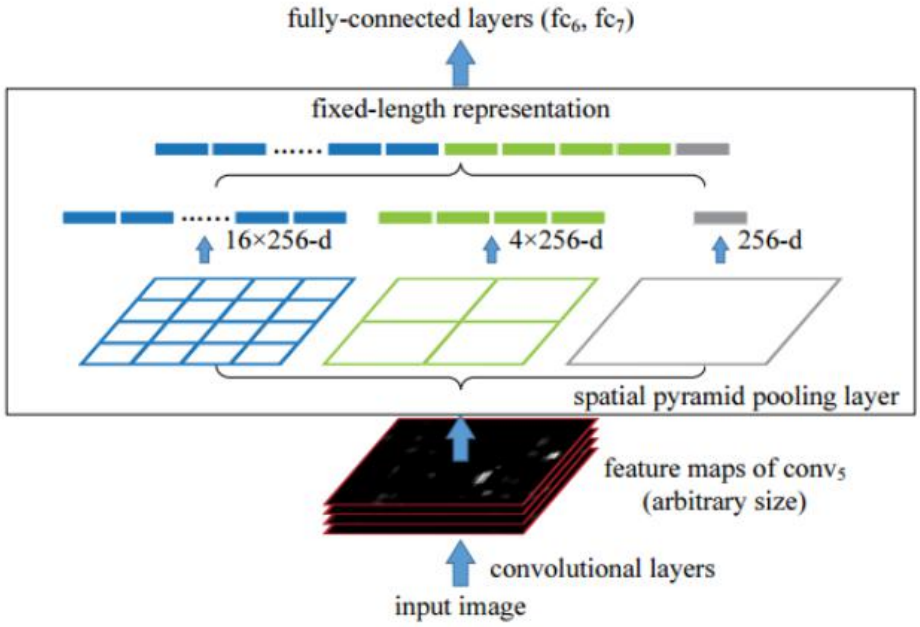

Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

图 3：Fast RCNN 网络结构图[2]

1.2.3 Faster RCNN[3]

经过 R-CNN 和 Fast RCNN 的发展，在 2016 年 Girshick 提出了新的 Faster RCNN，目标检测的几个步骤：特征抽取(feature extraction)，候选框提取（proposal)，预测框回归(bounding box regression)，分类)(classification)被 Faster RCNN 整合在了一个网络中，不仅检测速度大为提升，综合能力也得到进一步增强[3]。Faster RCNN 的网络结构如图 4 所示。

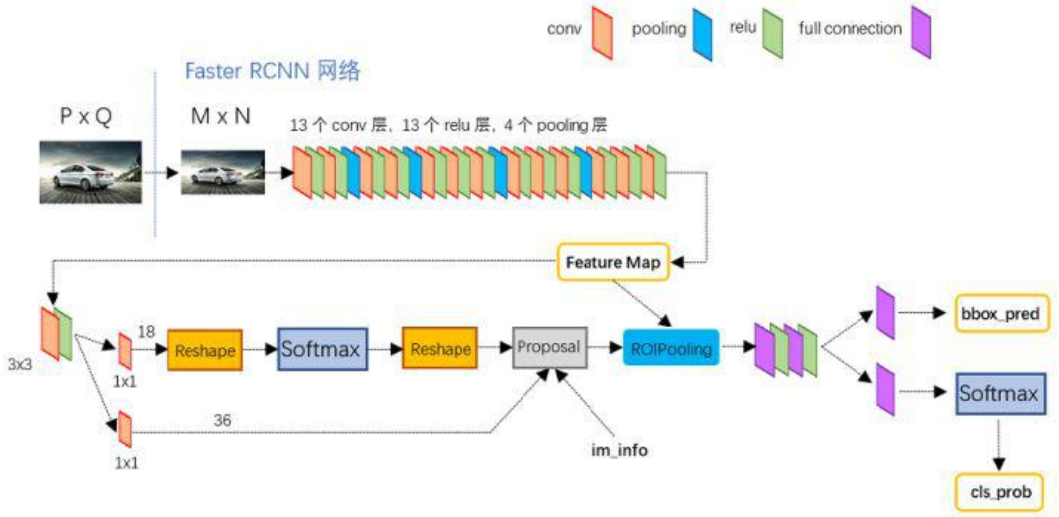

图 4 ：Faster RCNN 网络结构图[3]

图中展示了 faster_rcnn 的网络结构，faster_rcnn 对于输入尺寸为 P×Q 的图像，进行放大或者缩小，至固定大小 M×N，图像需要通过卷积池化激活等网络提取特征图；而卷积层中包含了 13 个二维卷积层和 13 个 ReLU 非线性层和 4 个 pooling 层；feature map 经过 RPN 网络，分别给出 positive anchor 和 predicted bounding box 的偏移值，由此得出候选区域；然后将候选区域经过 RoiPooling 层，从特征图中提取候选区域的 feature map，输入全连接层和 softmax 网络进行 classification。

我们系统分析一下该双阶段检测网络的设计思想，因为该模型的部分网络功能结构在之后的单阶段目标检测网络中也被借鉴很多。第一，原始图像通过深度卷积网络提取不同尺度大小的 feature map（也被称为 backbone)；第二，采用锚框对物体进行检测，网络的输出为锚框的 classification 与 regression 的值，

第三，特征图经过一系列卷积网络（下文称之为“检测头”）得到最终预测结果；第四，使用 softmax 对 classification 处理，说明 Fast RCNN 认为物体只能属于类别中的一类，且各类没有相似之处。

1.2.4 单阶段检测器[8]

是否有候选区域算法，是单阶段检测器和双阶段检测器的最大区别。双阶段检测器中的候选区域算法(selective search,RPN 等）提高了检测的 accuracy，却也降低了检测的速度。

2015 年，Joseph Redmon 提出了最早的单阶段检测器 YOLO。YOLO 的基本思想是将整体的图片划分为更小的区域：针对图片先设定一个划分尺寸 7×7，之后针对图片划分后的每个小块进行判断预测，分别去预测这一小块的类别的概率、这个块所属的检测框的回归偏移量[8]。YOLO 的网络结构如图 5 所示。可以看出 YOLO 试图用机械拆分小块区域代替 two-stage 的检测器中的候选区域算法，所以 YOLO 的速度更快了，但也是因此，YOLO 的检测精度低于 faster RCNN（各种算法的性能比较附后）。YOLO 还在不断的完善和发展中，YOLO v2[9]，YOLO v3[10]，YOLO v4 也逐渐被设计出来。

同年的 SSD[7]也找到了替代区域候选算法的方法，因为本设计使用了 SSD，因此具体的内容将在下章详细介绍。

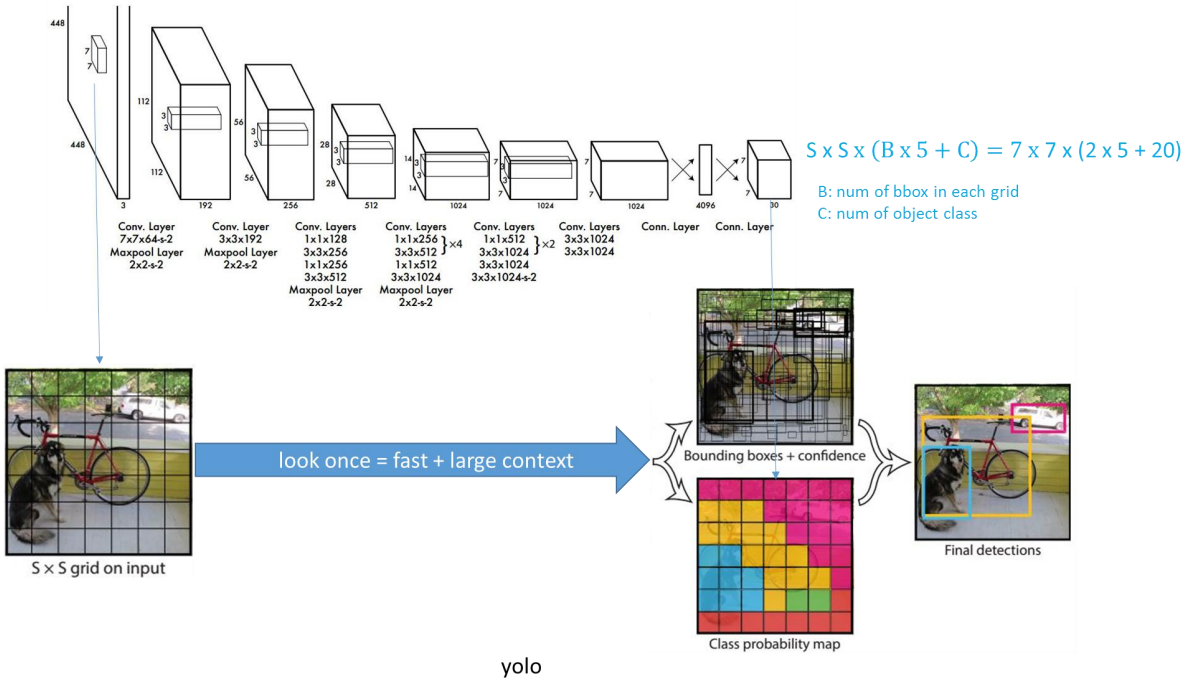

图 5：YOLO 网络结构图

2019 年，国科大与其他单位合作提出了一种新的算法——FreeAnchor，FreeAnchor 算法在 COCO 数据集上超过了 two-stage 检测器，该论文在 NIPS 上发布。

该论文作者张小松认为，此前简单根据交并比的大小的锚框匹配机制是不合理的。而应该通过构建锚框匹配的极大似然估计来优化这一过程。

1.3 目标检测数据集简介

1.3.1 PASCAL VOC 数据集简介

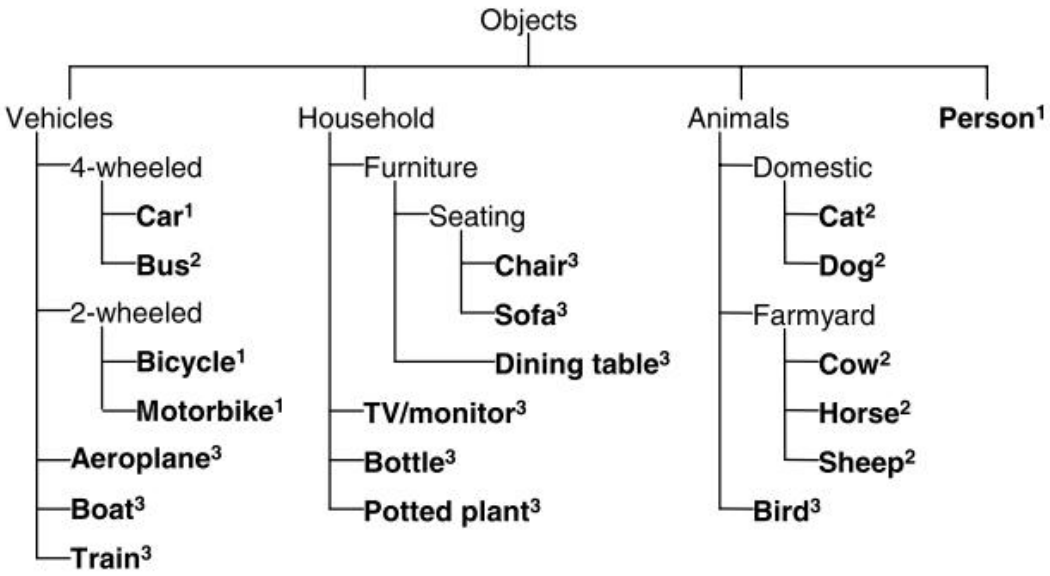

图 6　VOC2012 数据集中的种类

PASCAL VOC 提供用于对象类别识别的标准化图像数据集，提供用于访问数据集和注释的制造数据集的通用格式，可以评估和比较不同的目标检测算法，挑战评估识别的性能（从 2005 年至 2012 年，现已结束）

以 PASCAL VOC2012 数据集作为例子，VOC 数据集的目录结构如图 6 所示。数据集下载后中间包含 5 个文件夹，Annotations, ImageSets, JPEGImages, SegmentationClass, SegmentationObject。其中图像分为 20 类，数据集大小为 1.9GB 左右。

因为 PASCAL VOC 数据集小巧完备，训练测试省时省力，所以一般用于模型的实验和测试。本毕业设计的实验部分主要在 PASCAL VOC 上完成。

1.3.2 COCO 数据集简介

COCO(Common Object in Context)是微软提供的一个可以用于进行图像识别的数据集，主要针对目标检测、目标上下文关系和目标在 2 维上的定位三个问题。

相比与 PASCAL VOC 数据集，COCO 数据集有 80 类，超过 20 万个图像，并且每一类的图像多，其中 80 个类有超过 50 万个标注，而且平均每个图像的目标有 7.2 个。其他被广泛使用的计算机视觉数据集还有 MNIST 手写数字数据集，ImageNet，Open Images Dataset 等。COCO，ImageNet，Open Images Dataset 主要用于已完善的模型的大规模训练。

1.4 本文章节安排

本毕业设计论文将按照网络的结构顺序，依次说明特征提取骨架网络（SSD），特征金字塔网络（FPN），正负样本平衡误差（Focal Loss）和锚框自由匹配（FreeAnchor）的算法原理，并给出 FreeAnchorSSD300 的网络结构图，最后对该检测器的整体性能进行评估和展示。

2 第二章 FreeAnchorSSD 算法

2.1 SSD 算法[7]

2.1.1 SSD 算法思想

大部分卷积神经网络有一个特点，随着网络的深度变深，特征图的尺度会逐渐变小，而通道数会增加，因此，大尺度的特征图适合检测小目标以及局部的特征，比如纹理，小尺度的特征图适合检测大目标以及整体的特征，比如轮廓。

SSD 就采用了这种多级特征图预测的思想：通过预训练好的骨架网络得到尺度逐渐减小的 feature map，然后在不同尺度的 feature map 上设置尺度不同的 anchor，用以预测图片中大小不同的目标，最后根据 anchor 得到预测框的 classification 和 bounding box 的值（并不直接给出，而是相对于 anchor 的偏移）。

这样，与 YOLO 相比，SSD 的锚框更加具有系统性和层次性，SSD 的网络检测流程如图 7 所示。同时，SSD 输出中，classification 和 regression 是分别由不同的 subnet 给出的，而 YOLO 是通过一个全连接层给出，SSD 的网络结构设计更加具有逻辑性。SSD 网络结合了 YOLO 中关于 anchor 给出预测框位置的偏移值的回归思想和 Faster RCNN 中由特征图映射回原图的 anchor 机制，在尺度不同的特征图上给出不同尺度的物体的分类和位置。

因为 SSD 网络的上述特点，其速度和 YOLO 一样较快，同时也保证了目标检测的精度较高。但是因为 SSD 的 Anchor 是密集预测，无论是 SSD300 还是 SSD512（大于 8000），其 Anchor 数量均大于双阶段检测器中的候选区域数量（约 2000）所以仍然存在匹配到背景的 Anchor 过多而造成训练过程中正负样本不平衡的问题，这一点将会在 Focal Loss 中解决。

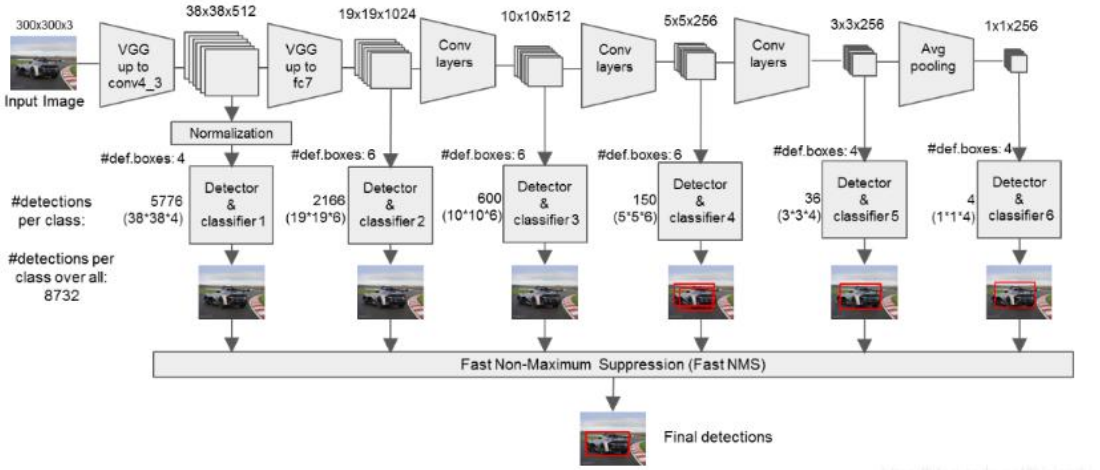

图 7：SSD 检测流程图[7]

2.1.2 SSD 网络结构

以 SSD300 为例，SSD 网络结构示意图如图 8：

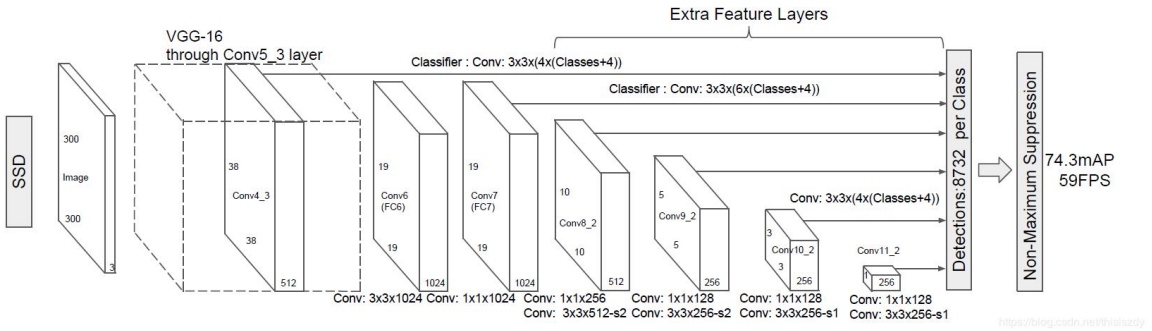

图 8：SSD 网络结构图[7]

结构说明：

1. backbone：输入一幅图片（300×300），将其输入到预训练好的 VGG16 特征提取网络中来获得不同大小的特征图[7]。SSD 将传统的 VGG16 网络的输入大小改为 300×300，第六全连接层（FC6，14×14×512）和第七全连接层（FC7，7×7×512）转化为二维卷积层 Conv6 和 Conv7；并添加了一些卷积层（Conv8，Conv9，Conv10，Conv11），这些层的大小逐渐减小，可以进行多尺度预测。

2. anchor generate：选取上图中 Conv4_3、Conv7、Conv8_2、Conv9_2、Conv10_2、Conv11_2 层得到的 feature map，然后分别在这些 feature map 层上面的每一个点构造 6 个不同尺度大小的 bounding box，然后分别进行检测和分类，生成多个 bounding box 的分类值和回归值，如上图所示；

3. anchor head: 检测目标时，将不同 feature map 通过 anchor head 获得的 bounding box 结合起来，经过 NMS（非极大值抑制）方法来抑制掉同类别中一部分重叠的 bounding box，生成最终的 bounding box 集合（即检测结果）

4. anchor matching&loss fuction: 训练模型时，通过 ground truth box(真实框，下面简称 gt_box）与每个 anchor 的交并比（下简称 IoU，具体的意义见第三章），为每个 anchor 分配一个类别以及回归值，而计算误差函数时，分别计算分类的总误差与回归的总误差，最终得到整体网络的误差。

2.1.3 SSD 代码分析

下列分析虽然称作代码分析，却不涉及具体的代码，而是以公式和矩阵的形式，用伪代码更加清晰地描述 SSD300 网络整个训练和预测的过程。

2.1.3.1 backbone

输入为三通道图像 3×300×300

3×300×300 经过 卷积层 得到 64×300×300

64×300×300 经过 ReLU 层 得到 64×300×300

64×300×300 经过 卷积层 得到 64×300×300

64×300×300 经过 ReLU 层 得到 64×300×300

64×300×300 经过 Maxpooling 层 得到 64×150×150

64×150×150 经过 卷积层 得到 128×150×150

128×150×150 经过 ReLU 层 得到 128×150×150

128×150×150 经过 卷积层 得到 128×150×150

128×150×150 经过 ReLU 层 得到 128×150×150

128×150×150 经过 Maxpooling 层 得到 128×75×75

128×75×75 经过 卷积层 得到 256×75×75

256×75×75 经过 ReLU 层 得到 256×75×75

256×75×75 经过 卷积层 得到 256×75×75

256×75×75 经过 ReLU 层 得到 256×75×75

256×75×75 经过 Maxpooling 层 得到 256×38×38

256×38×38 经过 卷积层 得到 512×38×38

512×38×38 经过 ReLU 层 得到 512×38×38

512×38×38 经过 卷积层 得到 512×38×38

512×38×38 经过 ReLU 层 得到 512×38×38——①

512×38×38 经过 Maxpooling 层 得到 512×19×19

512×19×19 经过 卷积层 得到 512×19×19

512×19×19 经过 ReLU 层 得到 512×19×19

512×19×19 经过 卷积层 得到 512×19×19

512×19×19 经过 ReLU 层 得到 512×19×19

512×19×19 经过 卷积层 得到 1024×19×19

1024×19×19 经过 ReLU 层 得到 1024×19×19

1024×19×19 经过 卷积层 得到 1024×19×19——②

1024×19×19 经过 ReLU 层 得到 1024×19×19

1024×19×19 经过 卷积层 得到 256×19×19

256×19×19 经过 卷积层 得到 512×10×10——③

512×10×10 经过 卷积层 得到 128×10×10

128×10×10 经过 卷积层 得到 256×5×5——④

256×5×5 经过 卷积层 得到 128×5×5

128×5×5 经过 卷积层 得到 256×3×3——⑤

256×3×3 经过 卷积层 得到 128×3×3

128×3×3 经过 卷积层 得到 256×1×1——⑥

对①进行 L2 Normalization，对②③④⑤⑥进行 ReLU，得到六张 feature maps，其尺寸分别为 512×38×38，1024×19×19，512×10×10，256×5×5，256×3×3 和 256×1×1。

2.1.3.2 anchor generate

①anchor 的数量：对每张特征图，例如 512×38×38，看成 38×38 个 512 通道的像素的组合，而每个像素对应原图上一定数量的，固定形状大小位置的 anchor。具体地说：

38×38 特征图上的每个像素对应 4 个 anchor，

19×19 特征图上的每个像素对应 6 个 anchor，

10×10 特征图上的每个像素对应 6 个 anchor，

5×5 特征图上的每个像素对应 6 个 anchor，

3×3 特征图上的每个像素对应 4 个 anchor，

1×1 特征图上的每个像素对应 4 个 anchor，

共计：38×38×4 + 19×19×6 + 10×10×6 + 5×5×6 + 3×3×4 + 1×1×4 = 8732 个 anchor，而 SSD512 的 anchor 数量更多。

②anchor 中心：每个像素生成的锚框以上每个像素特征图为中心。之后 feature map 上锚框中心的坐标会映射回原图对应的位置。

③anchor 大小（scale）：用面积来形容尺度大小更形象：6 个 feature map 中，38×38 的 feature map 上的 anchor 的 scale 值为 $s_{\min} = 0.2$，1×1 的 feature map 上的 anchor 的 scale 值为 $s_{max} = 0.95$，其他特征图的 anchor 的 scale 值通

过下面的公式计算得到，以第 k 个特征图的 anchor 的 scale 值 s_k 为例：

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

④anchor 的形状（ratio）：anchor 都是一系列具有不同长宽比例的矩形，因此我们使用一系列长宽比值（ratio）：1, 2, 3, 1/2, 1/3，通过下面的公式计算 anchor 的宽度 width 和高度 height

$$width(w_k^a = s_k\sqrt{a_r})$$
$$height(h_k^a = s_k / \sqrt{a_r}) \quad (2)$$

2.1.3.3 anchor head

anchor head 的作用是从不同尺度的 feature map 上得到每个 anchor 所预测的类别的 classification vector 和 regression vector。对于一个 anchor head 来说，输入为一张 feature map，经过 reg_convs 和 cls_convs（本质为二维卷积层）得到 cls_scores 和 bbox_pred（即上文所述的两个 vector）。

cls_scores 进行 softmax 操作后为物体分类的 one-hot 编码，特别 d 是 SSD 将 background 也作为了类别的第一个，举例来说，Pascal voc 数据集共有 20 个类别，网络需要的 one-hot 编码维度为 21，代码中的 num_class = 21 其实包括了背景。给出的 cls_scores 需要经过 softmax。最后得到的向量中数值最大的维度对应就是 SSD 网络的预测框中的物体所属的预测类别。

而 bbox_pred 的值为预测框相对于 anchor 的中心位置偏移，以及长和宽的形变比，最终的预测框用 4 个参数 $(l_{cx}, l_{cy}, l_w, l_h)$ 描述，分别表示预测框的中心坐标以及宽高。gt_box 绝对位置（中心点 x 坐标，中心点 y 坐标，长，宽）用 $d = (d_{cx}, d_{cy}, d_w, d_h)$ 表示，anchor 的绝对位置（含义同上）用 $b = (b_{cx}, b_{cy}, b_w, b_h)$ 表示，bbox_pred 应该逼近的正确值是 b 相对于 d 的转换值：

$$l_{cx} = (b_{cx} - d_{cx}) / d_w, l_{cy} = (b_{cy} - d_{cy}) / d_h \quad (3)$$
$$l_w = \log(b_w / d_w), l_h = \log(b_h / d_h) \quad (4)$$

上述过程为边界框的编码（encode），根据 bbox_pred $(l_{cx}, l_{cy}, l_w, l_h)$ 和 anchor 坐标 $(d_{cx}, d_{cy}, d_w, d_h)$ 得到预测框的真实坐标和大小时需要反向进行解码（decode）：

$$b_{cx} = d_w \cdot l_{cx} + d_{cx}, b_{cy} = d_h \cdot l_{cy} + d_{cy} \quad (5)$$
$$b_w = d_w \cdot \exp(l_w), b_h = d_h \cdot \exp(l_h) \quad (6)$$

2.1.3.4 anchor 匹配 & 误差函数设计

对于 SSD300 网络，其误差函数由分类误差与位置误差组成，论文中称做 confidence loss(classification loss)和 location loss(bouding box regression loss)。

在计算误差函数之前，必须给出网络需要逼近的正确输出值，这一过程是通过 IoU matching 机制实现：首先，寻找与每一个 gt_box 有最大的 IoU 的 anchor，这样就能保证每一个 gt_box 与唯一的一个 anchor 对应起来。SSD 之后又将还没有配对的 anchor 与任意一个 gt_box 尝试配对，只要两者之间的 IoU 大于阈值，就认为 match（SSD 300 阈值为 0.5）。match 到 gt_box 的 anchor 就是 positive，没有配对到 gt_box 的 default box 就是 negative。

根据 matching 结果，得到误差函数中的正确输出值：对于 positive 的 anchor，其 classification vector 中对应的物体类别应当逼近 1，其余类别应该逼近 0；其 regression vector 应当逼近 anchor 与 gt_box 之间的转换值（即上文提到的 encode）。对于 negative 的 anchor，其 classification vector 中对应的背景维度应当逼近 1，其余类别应该逼近 0；其 regression vector 应当逼近首位为 1 其余位为全 0 的向量。基于这种思想我们计算网络的输出与应该逼近的正确结果之间的误差函数：

损失函数定义为位置误差（locatization loss，简称 loc）与置信度误差（confidence loss, 简称 conf）的加权和，N 是 positive anchor 的总数：

$$L(x,c,l,g)=\frac{1}{N}[L_{conf}(x,c)+\alpha L_{loc}(x,l,g)](7)$$

总误差函数中分类的误差 confidence loss（L_{conf}）是这样给出的：

$$L_{conf}(x,c)=-\sum_{i\in Pos}^{N}x_{ij}^{p}\log(\hat{c}_i^{p})-\sum_{i\in Neg}^{N}\log(\hat{c}_i^{0})\ where\ \hat{c}_i^{p}=\frac{\exp(c_i^{p})}{\sum_{p}\exp(c_i^{p})}(8)$$

其含义为，对属于 positive 的 anchor，其 cls_scores 经过 softmax 其对 confidence loss 的贡献为第 p+1 维度（即 anchor 所被匹配到的 gt_box 所属类别对应的维度）与真实标注（即 x_{ij}^{p}，当 i（anchor）与 j（gt）的匹配结果为 positive 时，值为 1，否则为 0）的 p+1 维度的交叉熵；对于属于 negative 的 anchor，其对 confidence loss 的贡献为第 0 维度（即 anchor 所被匹配到的背景对应的维度）与首位为 1 其余位为全 0 的背景向量的交叉熵误差（cross entropy loss）[7]。

对于位置误差（L_{loc}），其采用 Smooth L1 loss，定义如下：

$$L_{loc}(x,l,g)=\sum_{i\in Pos}^{N}\sum_{m\in\{cx,cy,w,h\}}x_{ij}^{k}\,\mathrm{smooth}_{\mathrm{L1}}(l_i^m-\hat{g}_j^m)(9)$$

$$\hat{g}_j^{cx}=\frac{g_j^{cx}-d_i^{cx}}{d_i^{w}},\ \hat{g}_j^{cy}=\frac{g_j^{cy}-d_i^{cy}}{d_i^{h}}\ (10)$$

$$\hat{g}_j^{w}=\log\left(\frac{g_j^{w}}{d_i^{w}}\right),\ \hat{g}_j^{h}=\log\left(\frac{g_j^{h}}{d_i^{h}}\right)\quad（11）$$

其中：

$$\mathrm{smooth}_{\mathrm{L1}}(x)=\begin{cases}0.5x^2 & if|x|<1\\ |x|-0.5 & otherwise\end{cases}\quad）（12）$$

由于 x_{ij}^{p} 的存在，可以看出位置误差仅针对 positive 的 anchor 进行计算。当 bbox_pred 越接近 anchor 相对于 gt_box 的 encode 时，其 location loss 越小。

权重系数 α 通过 cross-validation 设置为 1[7]。

分析比较:

SSD 中的 anchor（论文中称作 default box）与 Faster-RCNN 中的 RPN 是一样的，不过 RPN 是预测 Box 里面有 Object 或者没有，没有分类，SSD 直接使用 Softmax 分类并将背景作为类别的一种包含在内。而 SSD 中位置的损失与 Faster R-CNN 相同，都是用“预测框和 anchor 的 encode”与“gt_box 和 ancho 的 encode”作差，得到 location loss。与同年提出的 YOLO 相比，因为同为 one-stage 检测器，SSD 检测速度与 YOLO 都有要快于 two-stage 的检测器，而且因为 SSD 的 anchor 的设置比 YOLO 的小区域划分更加科学，其预测精度比 YOLO 更好，但仍然略低于 two-stage 的检测器。

2.2 FPN[11]

在 SSD 的介绍中，我们可以领会到使用多尺度特征图提取特征（feature extraction）的思想，这一网络结构，在上文中被称作 backbone——即提取图像特征的骨架。这一结构可以固定下来，事先预训练好，然后再做 fine-tuning，常用的 backbone 有 vgg，resnet，darknet 等，顺带一提，其中 resnet 的作者何恺明也是 Focal Loss、retinanet 和 mask RCNN 的作者，我个人十分尊敬和崇拜。

FPN（Feature Pyramid Network）是对 backbone 网络对图片信息 feature extraction 的一种改善。其目的是构建一种 top-down 的特征整合方式，从而使得特征输出可以更好地表达各个 scale 的特征图的信息。因此 FPN 在代码中被称为 neck。FPN 的网络结构如图 9 所示，FPN 改善特征提取的步骤包括以下两步：通过 backbone 网络 bottom-up 的通路提取不同 scale 的特征；通过 neck 网络 top-down 的通路进行特征补充和增强。

bottom-up：作为 backbone 的大部分卷积神经网络有一个特点，随着网络的深度变深，特征图的尺度会逐渐变小，而通道数会增加，因此，浅层的特征图适合检测小目标，深层的特征图适合检测大目标。或者说浅层的特征用于检测纹理和局部，深层的特征图用于检测轮廓和整体。

top-down：上层的特征输出一般其 feature map 尺寸较小，能表征更大范围的图片特征。而整体的特征在对局部特征提取中也有关键的作用[11]。因此将上层特征图放大后再与下一层的特征图相加。

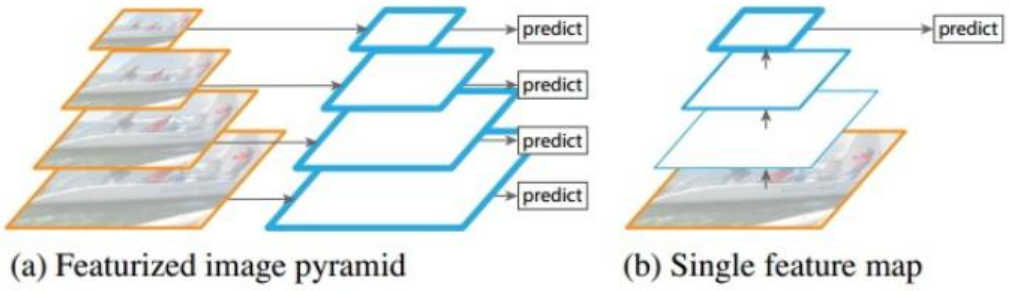

图 9：FPN 网络示意图[11]

图 10 为 Focal Loss 的论文中对 ResNet 进行 FPN 网络提取特征的示意图，可以看出存在自上而下和自下而上两个特征金字塔，最后 neck 网络的输出作为 anchor head 的输入，进行回归和分类。

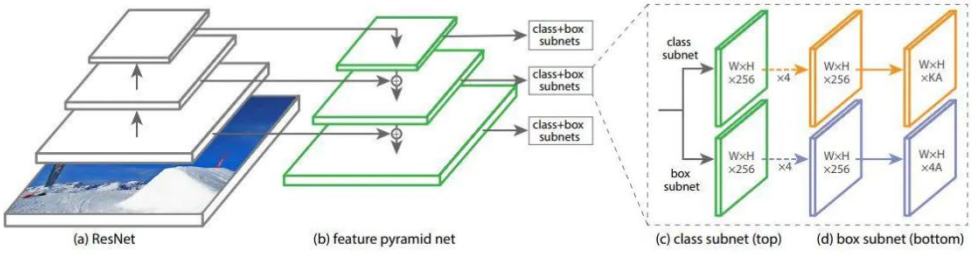

Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [20] backbone on top of a feedforward ResNet architecture [16] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN [20] while running at faster speeds.

图 10：RetinaNet 网络示意图[5]

2.3 Focal Loss

在 Focal Loss 提出之前，前面分析 SSD 算法的时候已经说明，单阶段检测器虽然在检测速度上优于双阶段检测器，但是其精度却因为没有候选区域推荐算法，一直低于双阶段检测器。何恺明认为，候选区域算法推荐的区域更少，而基于密集预测的单阶段检测器的 anchor 更多，negative（即匹配到 background）的 anchor 远远多于 positive（即匹配到 object）的 anchor，这导致其误差函数中，背景的 loss 淹没了前景的 loss，但前景往往更难分类，因此有必要改变 loss 的结构，使得 loss 能更好地针对前景，尤其是难以分类的前景类别。Focal loss 解决了 one-stage 目标检测中正负样本比例严重失衡的问题，降低了大量 negative anchor 或者容易分类的 anchor 在总的 loss 中所占的 weight[5]。Focal Loss 是一种难例挖掘。

Focal loss 是在 cross entropy loss 的基础上进行的修改，首先看一下二分类的 CE Loss 函数，其中 y'为网络预测出的二分类的分布在某一类上的概率，y 为二分类的真实概率（某一类的概率为 0 或者 1）

$$\mathrm{L} = -y\log y' - (1-y)\log\left(1-y'\right) = \begin{cases} -\log y' & y=1 \\ -\log\left(1-y'\right) & y=0 \end{cases} \quad (1)$$

对于 negative anchor，其全部的-log(1-y')相加，因为 negative 的数量众多，会导致其在 loss 中占很大一部分比例，而 positive anchor 的 loss 因为数量太少而被淹没，这导致了最终测试时的精度较低。

而二分类情况下，Focal Loss 的设计如下：

$$\mathrm{L}_{fl} = \begin{cases} -\left(1-y'\right)^{\gamma}\log y' & y=1 \\ -y'^{\gamma}\log\left(1-y'\right) & y=0 \end{cases} \quad (2)$$

Focal loss 的改进方法是：在交叉熵函数前增加了一个权重项：与正确预测值之间的差的γ次方，这使得分类效果越好，输出的概率越接近期望值的 anchor 在 loss 中占据的比例也会更小，其中γ>0 使得减少易分类样本的损失。使得更

关注于困难的、错分的样本：前景被重视，背景被轻视，难分类被重视，易分类被轻视。

假设γ为 2，对于 positive anchor 而言，假定一个容易分类的输出概率为 0.9，其交叉熵误差就是 0.045，而其 Focal Loss 只有 0.01×0.045，这说明容易分类的样本在总 loss 中占比会更少。而预测概率为 0.3 的难以分类的 positive anchor 其交叉熵损失为 0.523，其 Focal Loss 为 0.256，很显然在 Focal Loss 里，难以分类的样本在误差函数中占比更大。对于负类样本而言同样。

预测 0.1 的结果应当远比预测 0.7 的样本损失值要小得多。对于预测概率为 0.5 时，损失只减少了 0.25 倍，所以更加关注于这种难以区分的样本，权重项减少了简单样本和负样本的比重。

此外，加入正负样本均衡系数α，用来平衡 positive 和 negative 本身的比例不平衡：α一般取得比 1 小，意味着 positive anchor 要比 negative anchor 数量更少，这是因为匹配到物体*的 anchor 在所有 anchor 中只占少数。*

$$L_{fl}=\begin{cases}-\alpha\left(1-y^{'}\right)^{\gamma}logy^{'} & y=1\\ -(1-\alpha)y^{'\gamma}\,log\left(1-y^{'}\right) & y=0\end{cases} \quad (3)$$

Focal Loss 极大地改善了单阶段检测器中正负 anchor 不平衡导致精度不如候选区域算法的问题，从此单阶段检测器无论是在精度还是速度上都超过了原来的双阶段检测器，下图是目前的各类检测器的性能比较图，可以看出单阶段监测器表现强于双阶段检测器。

2.4 评价指标

2.4.1 Recall(召回率)和 Precision(准确率)

假设数据集为 $\mathcal{D}$，模型为 $\mathcal{M}$，模型 $\mathcal{M}$ 的错误率 $\mathcal{E}$ 为其分类错误的样本数占总样本的比例，公式如下：

$$\mathcal{E}(\mathcal{M};\mathcal{D}) = \frac{1}{c}\sum_{i=1}^{c} I(\mathcal{M}(x_i) \neq y_i) \quad （1）$$

与错误率 $\mathcal{E}$ 对应的则为模型的精度 accuracy，即为模型分类正确的样本数占总样本数比例，公式如下：

$$acc(\mathcal{M};\mathcal{D}) = \frac{1}{c}\sum_{i=1}^{c} I(\mathcal{M}(x_i) = y_i) = 1 - \mathcal{E}(\mathcal{M};\mathcal{D}) \quad （2）$$

而实际的目标检测中，包含正样本和负样本，需要模型尽量检测出正样本并且检测不出负样本，此时样本测试的情况就有如下表所示情况：

数据真实情况	模型输出为正	模型输出为负
正样本	TP(真正)	FN(假反)
负样本	FP(假正)	TN(真反)

表 3.1 样本测试情况表

准确率 accuracy 计算公式为：

$$P = \frac{TP}{TP+FP} \quad （3）$$

而召回率统计的为数据集中所有正样本，模型预测对了多少，计算公式为：

$$R = \frac{TP}{TP+FN} \quad （4）$$

2.4.2 IoU(Intersection over Union,交并比)

IoU 为两个集合的交集与并集的比值。在目标检测中常用于锚框匹配——根据模型产生的候选框(anchor)与真实的标记框(gt_box)的 IoU，即两个框交集与并集的比值决定是否将此 anchor 匹配到物体上。公式如下：

$$IoU = \frac{area(C) \cap area(G)}{area(C) \cup area(G)} \quad (5)$$

2.4.3 mAP(Mean Average Precision, 平均准确度均值)

假设模型输出某张图像某个类别 C 有 N 个候选框，我们选取候选框与真实的标记框的 IoU 的阈值为 0.5，当 anchor 与真实的 gt_box 的 IoU 高于阈值时，就认定这个候选框为正确检测，否则就认为其为错误检测，这样我们就能得到有N'个正确检测，然后我们知道图像实际上有 M 个真实目标，这样我们就能得到模型对该图片的类别 C 的准确度为N'/M，通过对数据集中所有图像中的类别 C 进行上述计算即可得到模型对于类别 C 的平均准确度(Average Precision)，公式如下：

$$Average\ Precision_C = \frac{\sum Precision_C}{N(Total\ Images)_C} \quad (6)$$

即为某个类别C的平均准确度等于数据验证集中C类的准确度的和除以含有该类别的图片的数量。

将计算类别 C 的平均准确度算法推广到数据集中所有的类别中，我们就能得到每个类别的平均准确度，通过将每个类别的平均准确度求平均我们即可得到 mAP(平均准确度均值)，公式为：

$$Mean\ Average\ Precision = \frac{\sum Average\ Precision_C}{N(Classes)} \quad (7)$$

即为平均准确度均值等于所有类别的平均准确度的和除以类别个数。

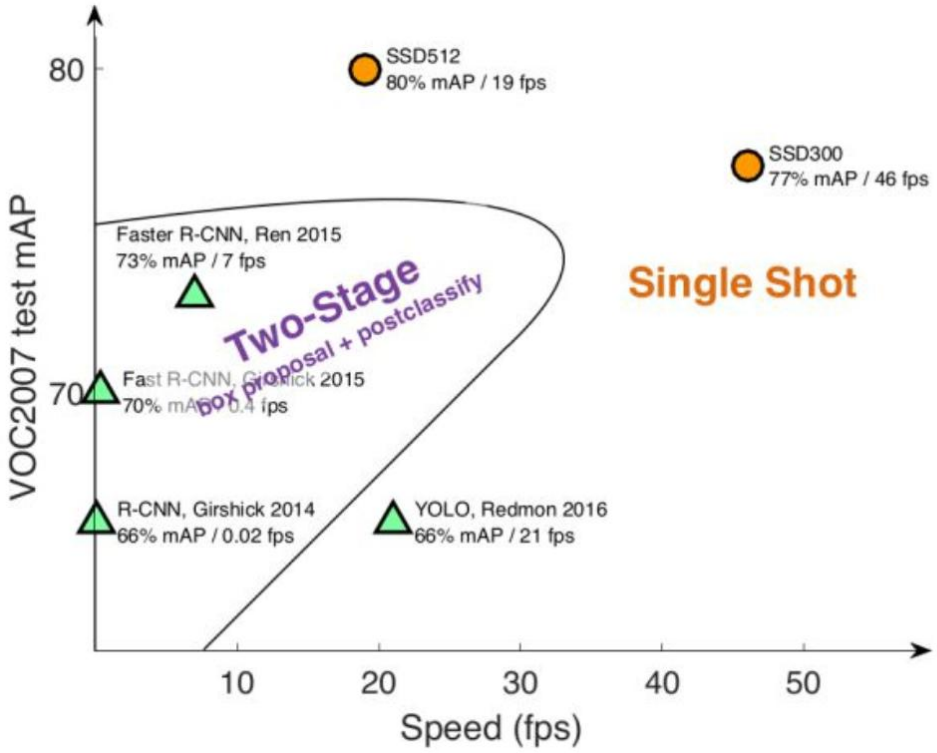

图 11：各类检测器性能比较

如图 11 所示为各类检测器（单阶段检测器使用了 Focal Loss）的性能对比，其中横轴检测速度 Speed 的单位是 fps（frame per second）即每秒检测的帧数，纵轴是其在 Pascal VOC2007（注：本文在 Pascal VOC2012 上测试）上的 mAP 值，可见单阶段检测器在精度和速度上均优于双阶段检测器。

2.5 FreeAnchorSSD

2.5.1 任务概述

经过 Focal Loss 的完善，one-stage detector 的性能已经得到很大提升，但是对于非中心的物体，遮挡，以及密集物体的预测，则需要引入 FreeAnchor 算法，

这里的 free 是自由锚框匹配的意思，而不是 anchor-free 中不需要锚框的意思。

基于锚的检测器利用空间对齐（即 gt_box 与锚之间的单位相交（IoU））作为锚分配的标准。 每个 anchor 的 positive 或 negative 的判定都基于与物体在空间上对齐的锚最适合分类和定位的假设，并以此监督网络学习以进行物体预测。然而，FreeAnchor 算法认为这样的假设是不合理的，IoU 标准不是锚框匹配的最佳选择[6]。

举个例子，如果一台笔记本电脑中间站了一只猫，那么这只猫对应的在特征图上的 anchor 的几何中心虽然和轮胎的 gt_box 非常重合，从而初始化后生成的预测框也和 gt_box 的 location loss 很小，自然，原始的匹配机制会将这代表猫的特征图的区域的理论输出结构匹配为笔记本电脑——但很显然这个中心区域的像素不能够也不应该做出笔记本电脑的预测，这就是传统 IoU 的匹配机制存在的弊端。

FreeAnchor 提出了一种锚框匹配方式，目的是将 IoU 分配改进为可学习的锚框匹配，如图 1 所示。FreeAnchor 主要从三个方面改进了原来的训练方式：第一，为了达到较高的召回率，网络需要保证至少一个 anchor 的预测值接近真实物体的分类和回归值。 第二，为了获得高检测精度，网络需要将定位不良（即较大的边界框回归误差）的 anchor 分类为 background。第三，锚框的预测应该减小非极大抑制（NMS）带来的负面影响，即对于分类得分低，但定位准确的情况。因为在之前使用局部非极大值抑制的过程时，如果有分类得分较低但与 gt_box 偏移小的预测框，也不应该直接作为非局部极值被抑制[6]。

2.5.2 算法思想

为了达成上述的任务目标，FreeAnchor 建立了锚框和物体匹配的概率矩阵，根据锚框和物体匹配的概率构建物体的锚框集合，并把根据匹配概率分配在 positive 误差函数中的比重。通过匹配概率引入 positive 误差函数和 negative 误差函数，将锚的匹配概率锚框集合的匹配联系起来。

另外，FreeAnchor 优化匹配功能以 Mean-Max 函数构建 positive 误差函数，以此从每个锚框集合中选取最佳的锚框[6]。 同时，大多数具有较大分类或定位误差的锚点被分类为背景，按照匹配为背景的概率计算 negative 误差函数。 训练时，将锚框匹配的似然概率设计在误差函数中，然后锚框匹配和分类、回归等预测值一起得到了训练和学习。

2.5.3 流程方法

在设计 FreeAnchorSSD 时，采用了原始 SSD 网络的多尺度特征提取的网络骨架，结合了 FPN 网络用于整体的特征在对局部特征提取中的作用，采用了 Focal Loss 对锚框的正负样本进行平衡和难分类物体重点关注，最后使用了 FreeAnchor 的锚框匹配似然以及优化匹配 Mean-Max 函数（2019，张小松）设计了整个模型，其检测流程如图 12 所示。

FreeAnchorSSD 的 backbone 网络提取特征的步骤与 SSD300 相同，在第二章已经系统论述，此处不在赘述，同样我们得到了 38×38，19×19，10×10，5×5，3×3，1×1 的特征图，再使用 FPN 网络从 SSD300 的骨架网络得到的特征图，使用 1×1 的卷积改变上层特征图的通道数，以向下层特征图吻合，再上采样上层特征图成为与下层特征图相同长宽，再进行相加，实际操作时可以采用线性插值、transpose convolution 进行上采样。得到最终的 38×38，19×19，10×10，5×5，3×3，1×1 六层的特征图。anchor 的生成方式也与 SSD300 网络相同，共

生成 8732 个 anchor。再使用不同的四层 conv2d 网络从特征图上得到每层所有 anchor 的 classification 和 localization 的预测值。

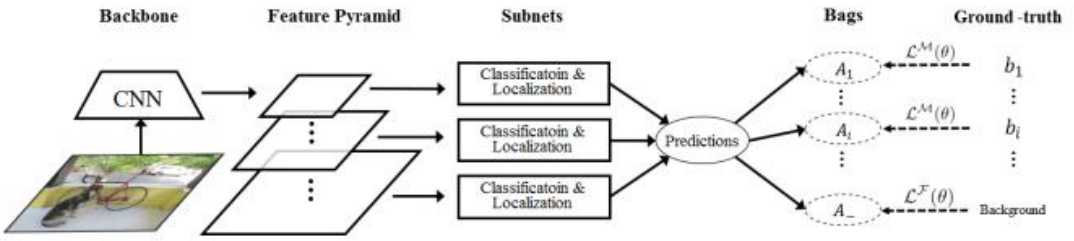

图 12：以 FreeAnchor 算法为基础的 CNN 检测器流程[6]

训练时还需要对网络输出进行误差计算，下面以一张图片为例进行说明：给定一个包含 I 个物体的训练图像 X，将 gt_box 的集合表示为 B，将第 i 个物体的 gt_box 表示为 bi∈B.在 X 的卷积特征图上，SSD 定义了一组锚框的集合 A 作为多尺度和形状的初始框。每个锚框对应于所有特征通道上的一维特征向量，即传统意义上的像素。在原始 SSD 网络里，网络基于 IoU 为每个物体分配锚框：当 bi 和 aj 之间的 IoU 大于阈值时，bi 匹配 aj。如果 aj 对多个物体的 IoU 大于阈值，则最大 IoU 的物体将成功匹配此锚框，从而确保每个锚框最多被单个物体匹配。在 FreeAnchor 训练开始时，我们仍然根据 IoU 准则，将 A 中的锚分为多个正锚框集合 Ai∈A +和负锚框集合 A-，而 A＝A＋∪A -（2019，张小松）。但意义不同于原始 SSD 的“属于”：在计算 positive 误差函数时，positive anchor 的误差并不会因为许多 anchor 同属于 positive 就占据相同的权重，而是要根据某个 anchor 给出的预测框与 gt_box 在位置上的相近程度（即 IoU)与**分类上的相近程度**分配该 anchor 在 positive loss 中的权重：例如某个 anchor 给出的预测框虽然与 gt_box 上吻合很好，但却在分类相差很大，原始的 SSD 网络会将其在 positive loss 中分配与其他 anchor 相同的权重，但在 FreeAnchor 中就会给出更低的 loss 的权重。而这种 anchor 的位置可能就是对应的真实物体中被遮挡的部分：理论上网络无法根据这一局部区域得到整个物体的分类或位置信息，故我们也不应该期望网络的预测逼近理想值。而 negative loss 更为简单，根据每个 anchor 不属于任何一个物体的概率，分配其在 negative loss 中的比重。

2.5.4 基本误差函数

首先介绍未完善的基本的误差函数。对于锚框物体进行匹配时，将正类锚框的分类损失定义为：

$$\mathcal{L}_{ij}^{cls}(\theta) = -log\big(a_j^{cls}(\theta)\big), \quad for \quad a_j \in A_i,$$

而将负类锚框的分类损失定义为：

$$\mathcal{L}_{j}^{cls}(\theta) = -log\big(1 - a_j^{cls}(\theta)\big), \quad for \quad a_j \in A_{-},$$

其中 $a_j^{cls}(\theta)$ 表示在给定网络参数θ的情况下锚 aj 的分类置信度。

正锚的定位损失定义为：

$$\mathcal{L}_{ij}^{loc}(\theta) = \ell_{reg}\big(a_j^{loc}, b_i, \theta\big),$$

其中 $l_{reg}(\cdot)$ 表示前面已经介绍过的 SmoothL1 Loss 回归误差：网络位置预测 a_j^{loc} 和 gt_box bi 之间的回归误差。根据定义的误差函数，通过优化以下总误差函数函数来进行检测器训练[6]：

$$\arg\min_{\theta} \mathcal{L}(\theta) = \sum_{i} \sum_{a_{ij} \in A_i} \big(\mathcal{L}_{ij}^{cls}(\theta) + \beta \mathcal{L}_{ij}^{loc}(\theta)\big) + \sum_{a_j \in A_{-}} \mathcal{L}_{j}^{cls}(\theta), \tag{1}$$

为了将误差转化为锚框匹配的概率。引入β作为平衡回归误差和分类误差的正则化因子。 从 likelihood 的角度来看，关于正类锚框 aj 正确预测第 i 个对象的概率定义为：

$$\begin{aligned} \mathcal{P}_{ij}(\theta) &= \exp\big(-\mathcal{L}_{ij}^{cls}(\theta) - \beta \mathcal{L}_{ij}^{loc}(\theta)\big) \\ &= a_j^{cls}(\theta) \exp\big(-\beta \ell_{reg}(a_j^{loc}(\theta), b_i)\big), \end{aligned} \tag{2}$$

该式将分类置信度 $a_j^{cls}(\theta)$ 与锚框 aj 的位置置信度结合起来，共同构成了锚框和物体匹配的概率。而为了使 anchor 的误差最小，就等同于使上述锚框匹配的概率最大：

$$\arg\max_{\theta} \mathcal{P}(\theta) = \prod_{i} \prod_{a_j \in A_i} \mathcal{P}_{ij}(\theta). \tag{3}$$

2.5.5 NMS 兼容

等式 3 从 likelihood 的角度统一了正类锚框的分类和定位误差。但是，上式计算得到的 likelihood 只表征单个 anchor 的检测效果。没有明确反映整体网络的检测性能。 因此，我们在 MLE 框架中引入了新的方法，旨在兼顾实现整体网络的高召回率和准确率的目标，同时确保锚框与 NMS 的兼容性（2019，张小松）。

要实现高召回率。对于每个对象 bi∈B，需要保证至少存在一个锚 aj∈Ai，其预测（ $a_j^{cls}(\theta)$ 和 $a_j^{loc}(\theta)$ ）接近于 gt_box。 锚框集合的召回率定义为：

$$\mathcal{P}_i^{rec}(\theta) = p(Y_i = 1|A_i;\theta),$$

其中 Yi∈{1，0}是一个二值变量，指示锚框集合 Ai 是否可以很好地预测物体 bi。aj∈Ai 能够预测的可能性遵循等式 2 中的定义，即：

$$p(y_{ij};\theta) = a_{ij}^{cls}(\theta)\exp\big(-\beta\ell_{reg}(a_{ij}^{loc}(\theta), b_i)\big), \qquad (4)$$

其中 yij∈{1，0}是一个二进制变量，用于指示是否锚 aj 可以很好地预测物体 bi。yij = 1 表示 aj 是正类锚框而 yij = 0 表示 aj 不是正类锚框（2019，张小松）。

为了达到较高的检测精度，检测器需要将定位不良的锚框归类于背景：

$$\mathcal{P}_i^{pre}(\theta) = p(Y_i = 0|A_i;\theta),$$

一个负类锚框没有定位物体的 likelihood 定义为：

$$p(y_{ij}^{-};\theta) = 1 - \max_i \frac{IoU_{ij}}{\max_j(IoU_{ij})}. \qquad (5)$$

其意在表征一个负类锚框不属于任何一个物体的概率，定义为该 anchor 不与任何物体重叠的程度[6]。其中 yij∈{1，0}是一个二进制变量，用于指示是否锚 aj 可以很好地预测物体 bi。y_{ij}^{-} = 1 表示 aj 是负类锚框而 y_{ij}^{-} = 0 表示 aj 不是负类锚框（2019，张小松）。

通过最大化上面定义的召回率 $P_i^{rec}(\theta)$ 和精度 $P_i^{pre}(\theta)$， 等式 3 体现为目标检测的自定义似然函数，如下：

$$\begin{aligned}\mathcal{P}^{\mathcal{M}}(\theta) &= \prod_i \mathcal{P}_i^{rec}(\theta) \times \prod_i \mathcal{P}_i^{pre}(\theta) \\ &= \prod_i p(Y_i = 1|A_i;\theta) \times \prod_i p(Y_i = 0|A_i;\theta),\end{aligned} \tag{6}$$

等式 6 结合了物体（与 NMS 的兼容性）的召回率，精度和 IoU 的目标。通过优化这种 likelihood，旨在同时使锚点与对象匹配并训练最佳检测器[6]。

2.5.6 锚框匹配函数

等式 6 定义了锚框集合的 likelihood。 但是，仍然缺乏一种将锚框集合的 likelihood 与锚框的 likelihood 联系起来的机制。 为了实现此目的，设置了一种锚框匹配函数 M +和 M-：

$$p(Y_i = 1|A_i;\theta) = \mathcal{M}_+\big(p(y_{ij};\theta)\big), \tag{7}$$

$$p(Y_i = 0|A_i,\theta) = \mathcal{M}_-\big(p(y_{ij}^-;\theta)\big). \tag{8}$$

基于锚框匹配函数，等式 6 的改写成：

$$\begin{aligned}\mathcal{P}^{\mathcal{M}}(\theta) &= \prod_i \mathcal{P}_i^{rec}(\theta) \times \prod_i \mathcal{P}_i^{pre}(\theta) \\ &= \prod_i p(Y_i = 1|A_i;\theta) \times \prod_i p(Y_{i|} = 0|A_i;\theta) \\ &= \prod \mathcal{M}_+\big(p(y_{ij};\theta)\big) \times \mathcal{M}_-\big(p(y_{ij}^-;\theta)\big).\end{aligned} \tag{9}$$

到目前为止，剩余的问题是实现锚框匹配函数 M +和 M-。 解决方法是多实例学习。例如 M+可以从一个正类锚框集合中选择匹配物体可能性最大的锚，即：

$$p(Y_i = 1;\theta) = p(\max_j y_{ij} = 1;\theta),$$

这样设计的原因是因为不能简单的把正类锚框集合中的锚框全部占据相同的比重来计算 positive loss/likelihood，因为有些 gt_box 会重合，或者物体特征不在 gt_box 的中心，理论上网络无法根据这一局部区域得到整个物体的分类或位置信息，故我们也不应该期望网络的预测逼近理想值。因此要有选择性地将综合效果（分类+回归）较好乃至**最好**预测框在 positive loss/likelihood 中的权重逐渐增大，将综合效果不好的预测框在 positive loss 中的权重逐渐减小，体现出“让好的更好”的针对性。

但是，在早期训练时期，对于随机初始化的网络参数，每个正类锚框的置信度都很小，而且相差不大[6]。 因此，针对置信度最高的锚进行优化不是检测器训练的最佳选择。 为了解决此问题，为正类锚框匹配指定了 Mean-Max 函数如下（2019，张小松）：

$$\mathcal{M}_+(\phi_i) = \frac{\sum_j \frac{\phi_{ij}}{1-\phi_{ij}}}{\sum_j \frac{1}{1-\phi_{ij}}},$$

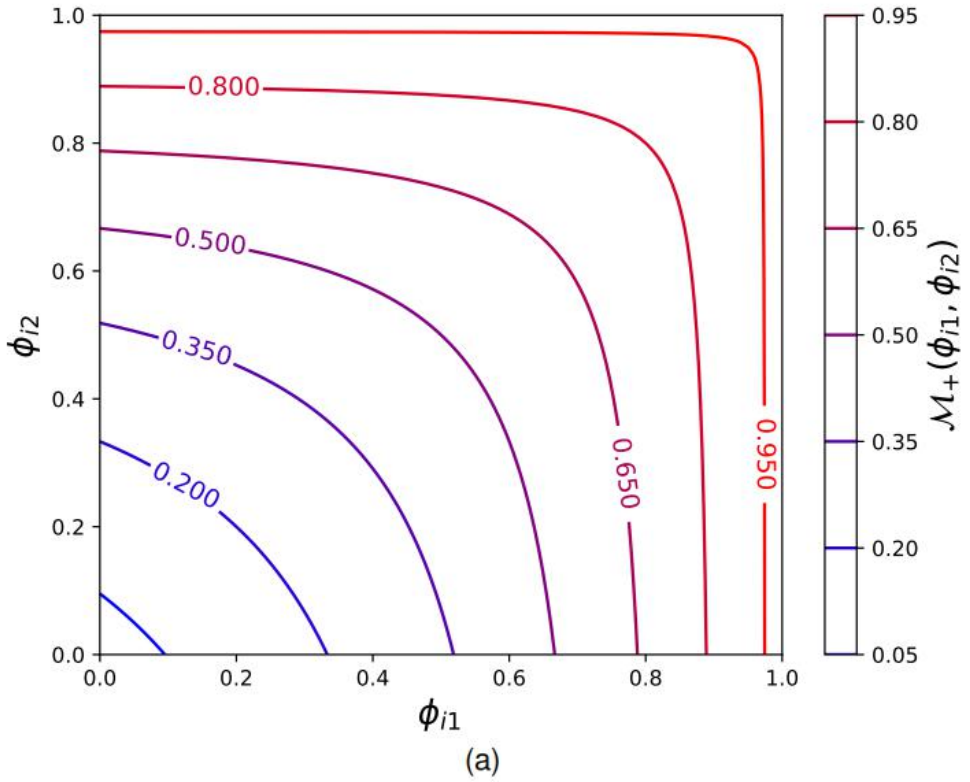

图 13：M+正锚框匹配函数简化示意图

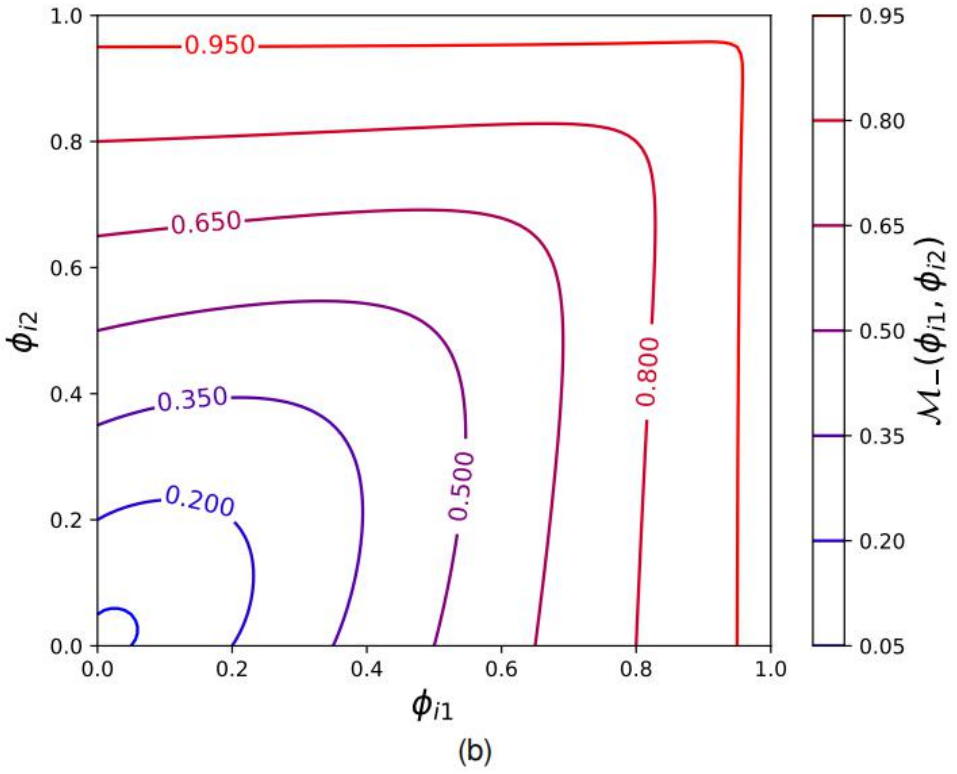

图 14：M-负锚框匹配函数简化示意图

正锚匹配（a）和负锚匹配（b）的二维 Mean-max 函数的如图 13 和 14 所示。“二维”表示一个锚框集合中有两个锚。 每个锚对应一个维度。 当两个维度的值都接近于零时，Mean-max 函数近似于 Mean 函数，并且函数值由两个维度（锚）确定。 当两个维度中的任何一个值较大时，均值最大值函数将近似最大值函数，并且函数值由较大值的维度确定。

当训练不足时，Mean-Max 函数接近均值函数，因此每个锚框集合中的所有锚框均用于训练。 随着训练的进行，一些锚框的置信度增加，并且 Mean-max 函数逐渐接近 max 函数。 经过足够的训练后，将从每个袋子中选择一个最佳锚框以匹配每个物体[6]。

类似地，引入负匹配函数 M-来选择负类锚框（2019，张小松）：

$$\mathcal{M}_{-}(\phi_i) = \frac{\sum_j \frac{\phi_{ij}}{1-\phi_{ij}}}{\sum_j \frac{1}{(1-\phi_{ij})^2}}.$$

如图 13 和图 14 所示，M-从均值到最大值的变化速度快于 M +。 原因在于，M +在使用 max 函数可以匹配最佳锚框之前，需要使用 Mean 函数充分考虑所有锚框。 相反，M-很快可以从负类锚框集合中大量的负类锚框选择高分的负锚框。从 M +到 M-的更快变化会更早地确定选定的锚框。

2.5.7 FreeAnchorSSD 误差函数

对于 FreeAnchorSSD 检测器的训练，最大化由等式 9 定义 likelihood，就可以同时优化网络参数和锚框匹配。 对于基于 CNN 的物体检测器，最大化自定义的 likelihood 就等同于减小锚框匹配的误差，即：

$$\begin{aligned}\mathcal{L}^{\mathcal{M}}(\theta) &= -\log \mathcal{P}^{\mathcal{M}}(\theta) \\ &= -\sum_i \log \mathcal{M}_+\big(p(y_{ij};\theta)\big) - \sum_i \log \mathcal{M}_-\big(p(y_{ij}^-;\theta)\big),\end{aligned} \tag{12}$$

对等式 9 取-log()即可得到等式 12。

为了平衡正锚和负锚对误差函数的影响，$L^M(\theta)$ 通过正锚和负锚的数量（K 和 N）来进行平衡，如下所示：

$$\mathcal{L}^{\mathcal{M}}(\theta) = -\frac{1}{K}\sum_i \log \mathcal{M}_+\big(p(y_{ij};\theta)\big) - \frac{1}{N}\sum_i \log \mathcal{M}_-\big(p(y_{ij}^-;\theta)\big), \tag{13}$$

其中，$p(y_{ij};\theta)$ 和 $p(y_{ij}^-;\theta)$ 分别由等式 4 和等式 5 计算得出，并且锚框匹配函数 M +（·）和 M-（·）分别由等式 10 和等式 11 给出。

在 FreeAnchorSSD 检测器训练期间，我们将由等式定义的锚框匹配损失降至最低。 考虑到前景-背景样本的极度失衡，采用 Focal Loss 以防止大量容易分类的负类锚框淹没了正类锚框的 loss，将如果将 loss 设计为 Focal Loss[6]：

$$\mathcal{L}^{\mathcal{F}}(\theta) = -\frac{1}{N}\sum_{a_j \in A_-} a_j^{cls}(\theta)^{\gamma} \log\big(1 - a_j^{cls}(\theta)\big),$$

其中，γ表示 Focal Loss 的指数参数，已经在前文中介绍。因此 FreeAnchorSSD 的最终损失函数是通过将 Focal Loss 与锚框匹配损失进行结合来定义的，即：

$$\mathcal{L}(\theta) = \mathcal{L}^{\mathcal{M}}(\theta) + \mathcal{L}^{\mathcal{F}}(\theta), \tag{14}$$

根据公式，锚框匹配损失和 Focal Loss 被设置为同等重要。

根据等式 14，使用随机梯度下降（SGD）算法即可端到端（end to end)地同时优化锚框匹配过程和 FreeAnchorSSD 检测器的训练。

2.6 FreeAnchorSSD 整体网络结构设计图

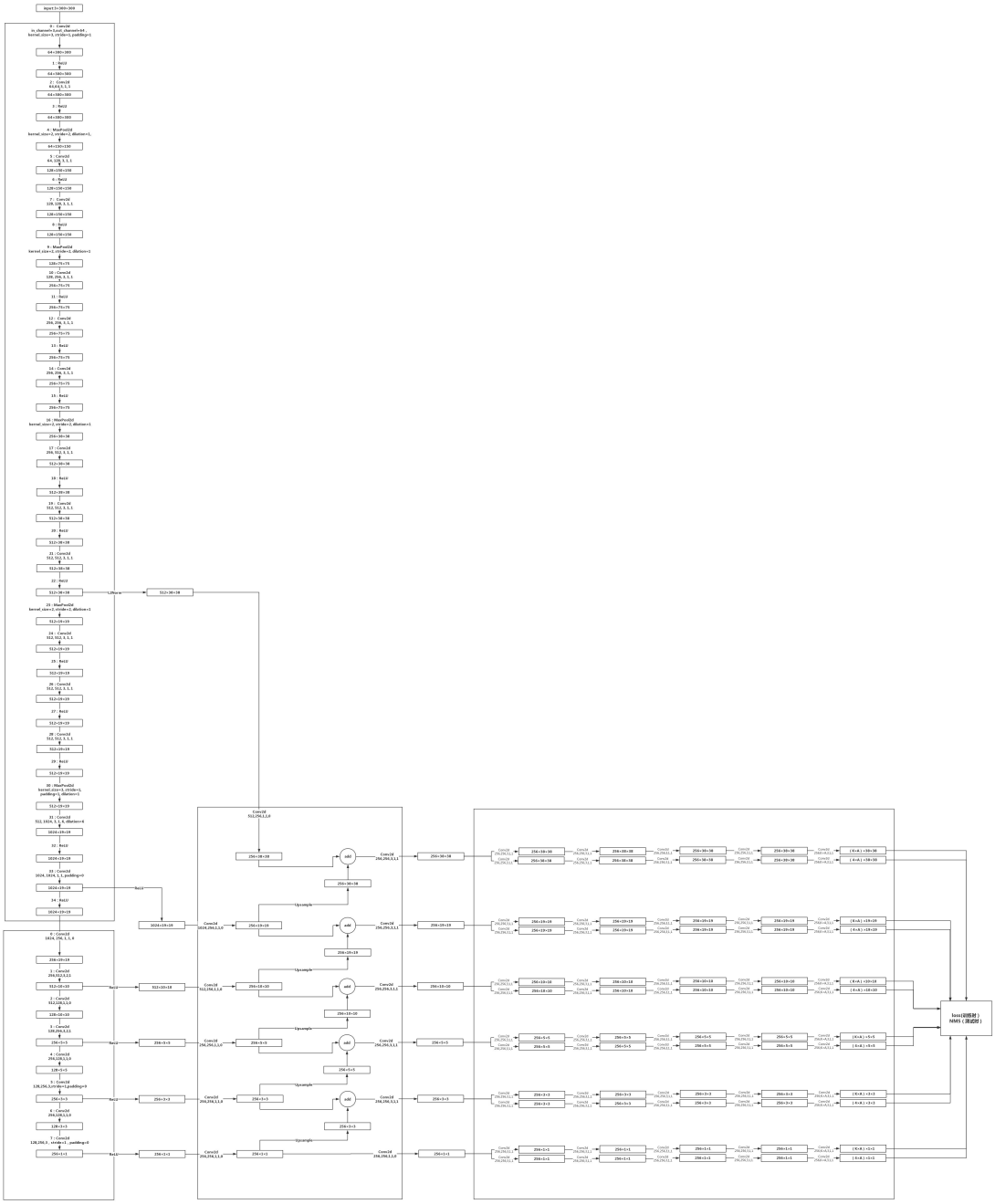

图 15：FreeAnchorSSD300 网络结构图（请放大）

3 第三章 检测评估

3.1 评估和对比

如图我们使用 SSD300 作为基础目标检测器。并且将 FreeAnchorSSD300 模型与基础模型进行比较，来对比显示出我们的模型在目标检测上更加出色，如图 16。

图显示了在 Pascal VOC2012 的测试数据集上 FreeAnchorSSD 的性能指标，可以看出召回率和 mAP 都比较高，如图 15。

class	recall	AP
aeroplane	0.935	0.855
bicycle	0.888	0.781
bird	0.857	0.72
boat	0.776	0.555
bottle	0.644	0.257
bus	0.914	0.826
car	0.785	0.537
cat	0.984	0.966
chair	0.782	0.377
cow	0.953	0.761
diningtable	0.839	0.604
dog	0.98	0.929
horse	0.95	0.878
motobike	0.902	0.816
person	0.876	0.705
pottedplant	0.613	0.299
sheep	0.838	0.548
sofa	0.912	0.78
train	0.971	0.94
tvmonitor	0.832	0.696
map		0.692

图 16：VOC2012 的测试集上，FreeAnchorSSD300 的性能评估图

与 baseline 模型 SSD300 相比，FreeAnchorSSD300 的 mAP 提升了五个点以上，可以证明本设计的网络结构的有效性。

	SSD300	FreeAnchorSSD300	提升
mAP	0.636	0.692	0.056

图 17：baseline 模型和 FreeAnchorSSD300 比较

3.2 FreeAnchorSSD 模型表现展示

图 18

图 19

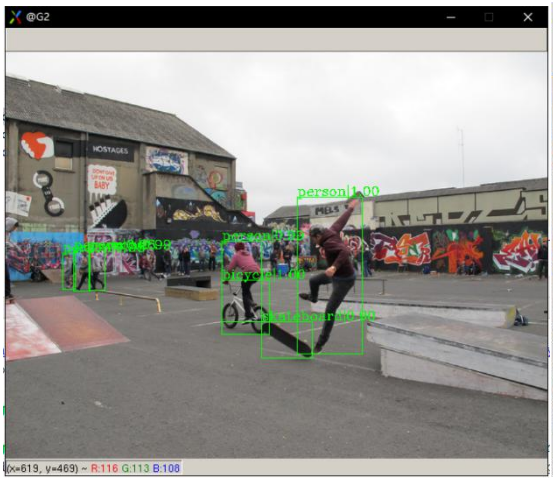

图 20

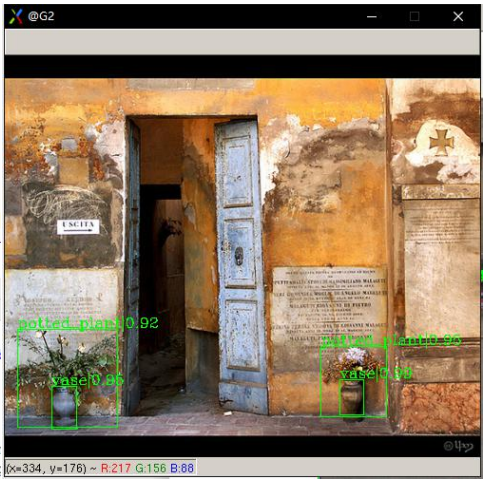

图 21

图 22

图 23

图 24

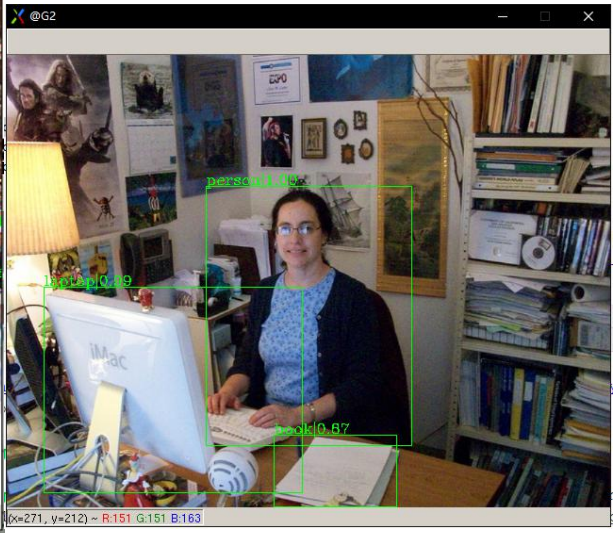

图 25

图 18 至图 25：FreeAnchorSSD300 模型表现展示

4 第四章 总 结

本论文在国科大张小松提出的 FreeAnchor 算法和 SSD 网络的基础上设计了行之有效的 FreeAnchorSSD 检测器，使用深度卷积网络进行目标检测。 通过将检测器的锚框匹配过程公式化为最大似然估计（MLE），将基于 IoU 的锚框分配改进为“自由”的锚框-物体匹配。在此基础上，FreeAnchorSSD 与原始的 SSD 形成鲜明对比，从而提高了对象检测的性能，进一步提供了有关锚框匹配功能的理论分析。 学习匹配方法和锚点匹配机制为深度学习框架中的目标表示和目标定位问题提供了新的见解。

5 第五章 参考文献

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation[C]. In CVPR, 2014.

[2] Ross Girshick. Fast R-CNN[C]. In CVPR, 2015.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[C]. In CVPR, 2016.

[4] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár. Focal Loss for Dense Object Detection[C]. In ICCV, 2017.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition[C]. In CVPR, 2016.

[6] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, Qixiang Ye. FreeAnchor: Learning to Match Anchors for Visual Object Detection. In NIPS, 2019.

[7] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector[C]. In CVPR, 2016.

[8] Redmon, Joseph , et al. "You Only Look Once: Unified, Real-Time Object Detection." 2016 IEEE Conference on Computer Vision and Pattern Recognition [C] IEEE, 2016.

[9] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger[C]. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),pages6517 - 6525.IEEE,2017. 1, 2, 3, 5

[10] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement[C]. arXiv preprint arXiv:1804.02767, 2018.

[11] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie. Feature Pyramid Networks for Object Detection[C]. In CVPR,2017.

6 第六章 致 谢

本论文是在叶齐祥教授的悉心指导下完成的。他饱满的科研热情与严谨的治学态度，给我留下了深刻的印象，将使我受益终生。疫情期间，只能在家远程连接服务器，叶教授为我提供了非常多的帮助。在此论文完成之际，向我的导师叶齐祥教授表达衷心的感谢。

在本文的研究过程中，我得到了实验室张小松、彭志亮师兄的热情指导，他们的帮助使我受益匪浅，在此向他们表达衷心的感谢。

最后还要感谢我的女朋友潘钊滢，疫情期间不能返校，她为我提供了情感上的支持。

再次感谢所有关心和帮助过我的老师、同学以及朋友。