一些经典的目标检测方法

目录

在此篇文章中，我们将回顾以下经典的目标检测方法：

1. RCNN 系列：RCNN[1], Fast RCNN[2], Faster RCNN[3]
2. YOLO 系列：YOLO v1[4], YOLO v2[5], YOLO v3[6]
3. 其他方法：SSD[7], FPN[8], RetinaNet[9], FCOS[10]

1 RCNN 系列

RCNN 系列目标检测器是经典的两阶段目标检测方法。两阶段目标检测方法包含**提出候选区域**和**对候选区域进行目标判别和检测框修正**两个阶段。由于两阶段目标检测方法在两个阶段都需要进行大量的计算，因此其运行速度较一阶段目标检测方法较慢但精度略高。

1.1 RCNN

RCNN 的工作原理可如图1.1所示，共可分为以下几个步骤：**首先**对于输入图片使用 Selective Search 算法提出约 2000 个候选框。**随后**将候选框中的图片转换到特定的大小。**其次**将固定大小的图片输入一个卷积神经网络得到该候选框中的图片的特征表示。**最后**将特征表示通过若干个 SVM 对候选框进行分类并通过回归方法对检测框进行修正。

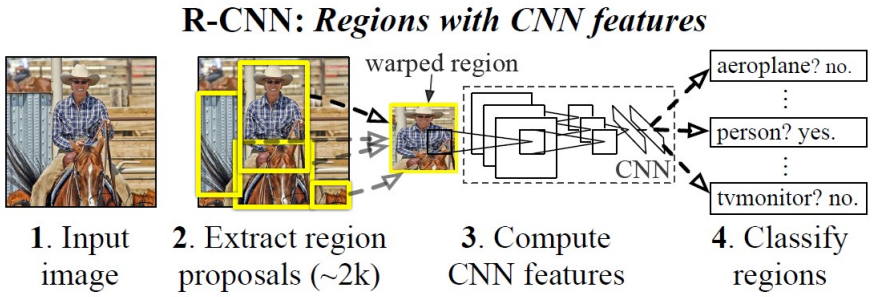

图 1.1: RCNN 工作原理：子图 2 对输入的图片生成了约 2000 个候选框。子图 3 对子图 2 生成的候选框进行裁剪后输入一个卷积神经网络。子图 4 对子图 3 生成的特征使用线性 SVM 对每一个类别进行分类。

候选框生成 (Reigon Proposals)

RCNN 使用 Selective Search 算法生成候选框或候选区域。Selective Search 算法是一套独立于 RCNN 后续流程的方法。Selective Search 对一张图片进行过分割，并逐步将过分割的区域按一定算法进行合并，生成一系列候选框。在这些生成的候选框中，物体有较大的可能在其中出现。RCNN 使用 Selective Search 算法生成约 2000 个候选框。

候选框转换 (Proposal Transformation)

使用 Selective Search 算法得到的候选框大小各异，由于之后需将其统一输入卷积神经网络，而神经网络要求输入的图片大小相同。因此，我们首先要对候选框进行大小转换。RCNN 原文中使用的卷积神经网络为 Alexnet，其输入维度为 227×227。在 RCNN 原文中，作者讨论了 4 种不同的候选框转换方法，最后使用了以下方法对候选框进行转换：首先对候选框周边的 16 个像素距离内的像素点也进行采集 (即若 SS 算法给出候选框为 $[w_1, w_2] \times [h_1, h_2]$，则

我们将候选框采集为 $[w_1-16, w_2+16]\times[h_1-16, h_2+16]$)。若将候选框向外扩充 16 个像素时遇到图片的边界无法继续扩充时，则使用图像的平均值进行填补，由于之后对图片预处理时，会将图片减去平均值，此处进行填补的部分最后会变为 0。实验证实，向外采样 16 个像素对 mAP 有 3-5 个点的提升。在采样后，RCNN 直接将候选框直接缩放至 227×227。

卷积神经网络

RCNN 将转换过后的 227×227 的图片直接输入卷积神经网络，神经网络的输出是一个 4096 维的特征向量。RCNN 原文中使用 Alexnet 作为卷积神经网络 (但是实际上任一个神经网络都可以直接取代该网络)。该神经网络的训练过程如下：

1. 首先将该网络后连接一个 1000 维的全连接层，将其在 ImageNet-1000 数据集上进行分类问题预训练。从现在的角度看，在 ImageNet 上进行预训练是十分自然且直接的选择，但是 RCNN 是第一个证实在大规模数据集上进行域无关任务 (non domain-specific task) 的预训练可以大大减少训练的时间和效率。

2. 其次将 1 中训练完成的网络的最后一个全连接层去除，并在目标检测的数据集上继续进行域相关的微调。例如在 VOC 数据集上进行微调时，由于 VOC 数据集共有 20 个类别，因此，需在网络后连接一个 21 维的全连接层 (20 个类别 + 背景) 继续进行训练。训练的数据是 Selective Search 算法所得到的候选框。每张图片会得到 2000 个候选框，对其中的每个候选框做如下标记：若该候选框和某个真实的目标框 (ground-truth box) 的 IoU 大于等于 0.5，则将该候选框的标签标记为目标框的标签，否则将其标签标记为背景。在训练过程中，为了控制正例 (20 个类别) 和负例 (背景) 的比例，在随机梯度下降的每一个批次中，RCNN 在大小为 128 的批次中使用 32 个正例和 96 个负例。

目标分类 (Object Category Classifier)

RCNN 将卷积神经网络输出的 4096 维特征向量输入线性 SVM 进行分类。对于此多分类问题，RCNN 使用多个线性 SVM 进行分类。例如对于 VOC 数据集，RCNN 使用 20 个 SVM 进行分类，对于每一类别，该类别的 SVM 会对特征向量进行一个二分类的判断，判断其是该类别的物体还是背景。

SVM 训练时使用以下方法。对于某一类别的 SVM，其正例为真实的目标框 (GT box)，负例为所有候选框中与该类别目标框 IoU 小于 0.3 的候选框。注意，此处 IoU 的阈值为 0.3，与卷积神经网络中 IoU 阈值为 0.5 不同。此处的 0.3 是由格点搜索法得出的，采用其他阈值会导致 mAP 大幅度下降。在 RCNN 论文中，作者在附录中专门讨论了两处 IoU 阈值的设置原因。此外，在 RCNN 中，目标分类是使用 SVM 而不是 softmax 进行分类的原因是作者发现使用 softmax 时，模型的 mAP 会下降 4 个点左右。这样的设置导致 RCNN 是一个有层次的目标检测器，即必须先通过卷积神经网络，再通过 SVM，两个步骤是先后进行的。

检测框回归 (Bounding Box Regression)

在 RCNN 的第一版中，是没有检测框回归这一步骤的。因此，在测试阶段，若某一类别的 SVM 将候选框判断为正例，则 RCNN 直接将该候选框作为预测结果。但是显而易见的，由 SS 算法得来的候选框不一定是最准确的，如果可以对候选框继续进行微调，则可以预见地提升预测框的准确程度。

检测框回归的训练数据为成对的候选框和目标框 $\{P_i, G_i\}_{i=1,2,\ldots,N}$，其中 P_i 是所有候选框中与 G_i 有最大 IoU(且该 IoU 需大于 0.6) 的候选框。假设 $P=(P_x, P_y, P_w, P_h), G=$

(G_x, G_y, G_w, G_h),x, y, w, h 分别表示左上角的坐标和检测框的长宽。假设卷积神经网络输出的特征向量为 P，检测框回归方法使用以下函数拟合 G_x, G_y, G_w, G_h：

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp\left(d_w(P)\right)$$
$$\hat{G}_h = P_h \exp\left(d_h(P)\right)$$

其中 $d_*(.)$ 是可学习的参数，RCNN 使用线性函数刻画这些函数，即 $d_*(P) = w_*^T P$，并使用岭回归求得 $w_* = \text{argmin}_w \sum_{i=1}^{N}(G_* - \hat{G}_*)^2 + \lambda\|w\|^2$。其中，在 RCNN 中 λ 的取值为 1000。使用检测框回归可以提升 VOC2010 数据集的 mAP3.5 个点左右。

测试

测试时的步骤较为简单，首先使用 SS 算法对图片提出约 2000 个候选框，随后将候选框按训练时的步骤转换为 227×227 大小的图片，之后将这 2000 个候选框依次通过训练好的卷积神经网络变为特征向量，最后通过 SVM 得到每个候选框为正例的置信度 (仅取置信度最高的类别) 并使用检测框回归法校正候选框的位置。在对 2000 个候选框完成此套流程后，可通过划定阈值及 NMS 方法留下最终的预测框。

1.2 Fast RCNN

RCNN 在测试时的主要计算时间在于对约 2000 个候选区域在神经网络中计算特征向量。要注意到这 2000 个候选区域的计算是不共享计算的，因此 RCNN 在测试时速度较慢。Fast RCNN 对于 RCNN 的主要改进在于将一张图片在神经网络中的传播共享，即对一张图片仅进行一次神经网络的计算，并将候选区域的分类与校正放于特征图 (feature map) 中进行计算。图1.2显示了 Fast RCNN 的工作步骤。**首先**，Fast RCNN 同样基于 Selective Search 算法得到一张图片的约 2000 个候选区域。**其次**，与 RCNN 分别将 2000 个候选区域稍作处理送入卷积神经网络不同，Fast RCNN 直接将整张图片送入卷积神经网络得到该图片的特征图，并将 2000 个候选框映射到特征图中。因此 Fast RCNN 的 2000 个候选区域可以近似看做是由 SS 算法在图片的特征图层面上得到的。**随后**，对于每个候选区域，Fast RCNN 将其从特征图上裁剪下来，并使用候选区域池化方法将其变为固定大小。**最后**，再将固定大小的候选区域通过 2 个并行的分支，分别对候选区域的类别进行分类和检测框回归。

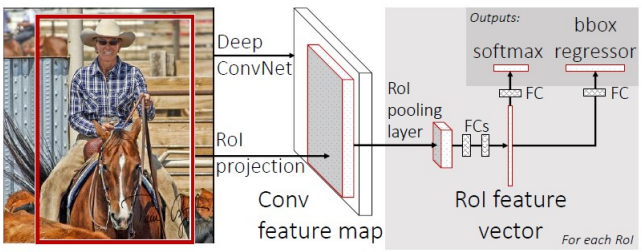

图 1.2: Fast RCNN 工作原理：对于一张图片和其对应的约 2000 个候选框，均通过神经网络一次性地将候选框转换到特征图上。随后使用池化方法将候选区域转换成固定的大小，再将其通过全连接层并分为两个分支，在两个分支上分别进行类别分类和检测框校正。

候选区域池化 (RoI Pooling)

Fast RCNN 通过卷积神经网络将候选框从原始图片转换到特征图中。首先需要说明此处的卷积神经网络仅仅包括卷积层和部分池化层，并不包括全连接层。其次由于原始的候选框大小各异，因此将其映射到特征图后，其在特征图中的大小也各不相同。与 RCNN 相同，Fast RCNN 也需要将其转换为相同大小，以便后续同一操作。由于原始的候选框是一个矩形，根据卷积层和池化层的特性，候选框在特征图上也为一个矩形。假设用 (r, c, h, w) 表示特征图中的候选框，r, c 表示候选框的左上角的坐标，h, w 表示候选框在特征图上的长和宽。**候选区域池化层**是一个最大池化层，其将大小为 $h \times w$ 的候选框转换为较小且大小固定为 $H \times W$ 的特征表示 (Fast RCNN 中 $H = W = 7$)。该最大池化层将 $h \times w$ 大小的候选框分为大小接近的 $H \times W$ 个子区域，每个子区域的大小约为 $h/H \times w/W$。随后在每个子区域上使用最大池化方法，得到大小为 $H \times W$ 的特征表示。注意，候选区域池化层是最大池化层的一种特殊形式，因此在网络训练时，其也可反向传播梯度。

卷积神经网络

Fast RCNN 的网络结构可分为：特征提取器 (由若干个卷积层和池化层组成)，候选区域池化层 (由一个最大池化层组成)，全连接层 (将候选区域池化层转化为一个特征向量) 和两个并行的分支 (均由全连接层构成，分别进行类别分类和检测框回归)。因此，Fast RCNN 的网络是一个端到端的网络，可以在训练中从头至尾进行优化。与一般网络不同，Fast RCNN 的输入不再是一张张图片，而是**图片和该图片上的若干候选区域**。对于一张图片和该图片上的一个候选区域，Fast RCNN 都有 2 个输出。

Fast RCNN 使用的网络为 VGG Net。同样的，该 VGG 网络需先在 ImageNet-1000 分类数据集上进行预训练。不过为了满足 Fast RCNN 中候选区域池化层的要求，VGG 网络中的最后一个池化层将被更改为输出为 $H \times W$ 的池化层。相应的，原始 VGG 网络中的后续全连接层的维度也需要相应变化。在预训练过后，VGG 网络将被移除最后的全连接层，并将前部的卷积层和池化层分别变为 Fast RCNN 中的特征提取器和候选区域池化层。

随后，Fast RCNN 在之后加上若干个全连接层先将特征图变为一个特征向量。随后将这一特征向量通过两个并行分支。第一个分支对于该候选区域输出其在 $K + 1$ 个类别上的概率分布。其中 K 为物体的类别数，1 表示背景。该分支的损失函数由 softmax 函数计算。第二个分支对于该候选区域输出其检测框的偏移量 $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$，其中 t_*^k 均采用了与 RCNN 一样的标准化转换方法。该分支的损失函数由 smooth-L1 函数计算。smooth-L1 函数较为稳定且不易受不易受异常值影响，定义如下：

$$\underset{L_1}{\text{smooth}}(x) = \begin{cases} 0.5x^2 & , |x| < 1 \\ |x| - 0.5 & , |x| \geq 1 \end{cases}$$

训练时，对于每一个候选区域，Fast RCNN 都会输出两部分的损失，我们可以使用多任务损失函数将两部分的损失融合起来。假设候选区域的类别标签为 u，其检测框回归任务的目标为 v。假设 Fast RCNN 两个分支的输出分别为 p 和 t^u，则该候选框的损失为：

$$\begin{aligned} L(p, u, t^u, v) =& L_{cls}(p, u) + \lambda I(pos) L_{loc}(t^u, v) \\ =& \begin{cases} -\log p_u & , \text{background} \\ -\log p_u + \lambda \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i^u - v_i) & , \text{otherwise} \end{cases} \end{aligned}$$

从损失函数中可以看出，如果候选框内包含的是背景，则仅计算其分类损失；如果候选框内包含的是物体 (正例)，则计算其分类损失和检测框损失。在 Fast RCNN 中，超参数 $\lambda = 1$。

同样地，Fast RCNN 也使用批次进行训练。由于每次网络所需的输入为图片和图片上的一个候选区域的组合对。因此，为加速训练过程，减少训练图片在每次训练时的前向传播所用时间，Fast RCNN 层次化地构造批次数据。首先，Fast RCNN 从所有图片中随机抽取出 N 张图片，其次从这 N 张图片的每张图片上随机抽取出 R/N 个候选框，共得到 R 个图片-候选框的组合对，构成一个训练批次的数据。注意到，这样的批次构成方式仅需网络的卷积层和池化层向前传播 N 次，比起较为简单的随机抽取图片-候选框的方法所需传播 R 次的方法快上 R/N 倍。在 Fast RCNN 中，$N = 2, R = 128$，因此使用这样的批次构成法使得 Fast RCNN 的训练速度可以快上 64 倍左右。通过实验可以发现，这样的批次构成法不会导致网络收敛慢的情况发生。

此外，为加速训练速度，Fast RCNN 使用了和 RCNN 一样的批次的采样方法。每一个批次包括 2 张图片，这两张图片是随机从所有图片中抽取的。从这 2 张图片上各选取 64 个候选框，选取方法如下：其中 25% 的候选区域为正例，其与某个真实的目标框的 IoU 大于 0.5；剩余的 75% 候选区域为负例，Fast RCNN 使用了难例挖掘的方法，从 IoU 位于 $[0.1, 0.5)$ 的候选框中随机抽取。其中 0.1 是一个可以调节的超参数。

测试

测试时，网络需要输入一张图片和这张图片上的约 2000 个候选区域 (使用 Selective Search 算法得来)。对于这 2000 个候选区域，网络都会输出其类别的概率密度和检测框的偏移量。注意，在输出的过程中，底部的特征提取器只运行一次，而其他部分则会运行 2000 次。最后，Fast RCNN 使用 NMS 得到最终的检测框。相比于 RCNN，使用相同的 VGG 网络作为特征提取器，Fast RCNN 将网络的训练时间提升 9 倍，测试时间提升 213 倍。

1.3 Faster RCNN

Fast RCNN 在测试时的主要时间用于约 2000 个候选区域的生成，剩余的部分的运行速度接近于实时。Faster RCNN 相比 Fast RCNN 在候选区域的生成上做出了重大的改变，其不再使用独立于网络的 Selective Search 算法，而是使用区域提出网络 (Region Proposal Network, RPN)，使用神经网络来给出候选区域。在仅仅给出 300 个候选区域的情况下，Faster RCNN 可以每秒处理 5 张图片并获得很高的 mAP。图1.3展现了网络的结构，其中最重要的结构是 RPN，Faster RCNN 可以看做是 Fast RCNN 和 RPN 的结合。若在图1.3中去除 RPN，则剩余结构和 Fast RCNN 并无差别，因此 Faster RCNN 可以看做将 Fast RCNN 中的候选区域提出方法由 Selective Search 算法更换为 RPN。更进一步的，Faster RCNN 的网络可以分为三个部分，**第一部分**是特征提取器，由卷积层和池化层构成，将原始图片转换成特征图；**第二部分**是 RPN，其以特征图为输入，通过 anchor box 为载体，输出若干个候选区域；**第三部分**是 Fast RCNN 检测器，其包括一个 RoI 池化层将候选区域转换为特定大小，并通过几个全连接层得到两个并行的输出，得到特征区域的类别和偏移量。第二部分和第三部分统一在特征图上进行操作，共用特征图，由 RPN 提出候选区域，并由 Fast RCNN 对候选区域进行判断和校正。

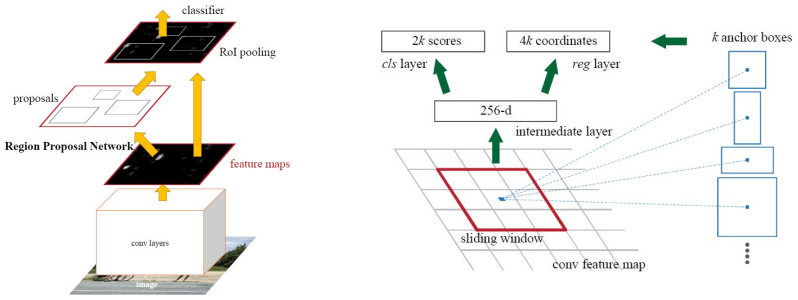

图 1.3: **左**：Faster RCNN 网络结构，在通过卷积层得到特征后，其通过 RPN 得到候选区域，并使用 RoI 池化得到候选框的特征，最后通过分类器决定其类别并使用检测框回归对检测框进行校正。**右**：RPN 的结构，RPN 是一个类似于全卷积神经网络 (FCN) 的结构，使用锚 (anchor) 对提出的区域判断其可否成为候选区域，并对区域进行校正。

区域提出网络 (Region Proposal Network, RPN)

Faster RCNN 的特征提取器由 VGG-16 网络的卷积层和池化层构成。输入的图片的维度为 1000×600，经过特征提取器后的特征图大小为 60×40。RPN 即以该 60×40 的特征图作为输入，输出的若干个候选区域，可以用四元组 (x, y, w, h) 表示。RPN 生成候选区域的方式是：在特征图上的每一个像素点上都拿出若干个固定大小的锚 (anchor)，通过与真实的目标框进行比较训练神经网络，并在测试时得到候选框。

Anchor 是一组给定大小且拥有固定大小的矩形框，在 Faster RCNN 中共使用了 9 个 anchor，9 个 anchor 有着 3 个不同的大小参数和 3 个不同长宽比例参数。其在原始图像上的大小分别为 $128^2, 256^2$ 和 512^2，长宽比例分别为 $1:2, 1:1, 2:1$。因此 anchor 的大小分别为 $64 \times 128, 128 \times 128, 128 \times 64, 128 \times 256, 256 \times 256, 256 \times 128, 256 \times 512, 512 \times 512, 512 \times 256$。9 个不同大小、不同尺寸的 anchor 可以满足不同物体的大小和尺寸。

如前所述，RPN 的输入是特征图，其维度为 $C \times 60 \times 40$，其中 C 为通道个数。RPN 在 60×40 的每个格点上都使用 9 个 anchor 进行判断，anchor 的中心位于该格点的中心，anchor 所处位置即 RPN 所提出的潜在的候选区域。注意到此处会一共提出 $60 \times 40 \times 9 \approx 20000$ 个潜在的候选区域。不过这些候选区域仅仅是潜在的，之后它们会通过 RPN 网络。RPN 网络的输出有两个，可如图1.3所示，其一是一个二分类的分数，分别表示其可以成为真正的候选区域的分数和不能成为真正的候选区域的分数，用来判断该 anchor 所表示的潜在的候选区域可否成为真正的候选区域；其二是候选区域的偏移量，用来控制改变 anchor 的大小和位置，其对每个 anchor 输出 4 个值，分别表示其中心的偏移量和长宽的变化量，其定义和 RCNN 中检测框回归的 4 个参数一致。所以可以看到，虽然 anchor 的大小是固定的 (对同一 anchor 而言) 且在同一格点上的 anchor 的中心相同，但其经过 RPN 后，RPN 会对其位置和大小进行调整，得到的候选区域的位置和大小便存在差异。

在训练时，RPN 会对特征图的每个格点都将所有 anchor 进行判断。因此，对于一个格点，需判断 9 个 anchor 是否可以成为候选区域，RPN 会对该格点输出 $2 \times 9 = 18$ 个分数以判断其是否可成为候选区域和 $4 \times 9 = 36$ 个偏移量以对 anchor 进行调整。所以，若输入的特征图的维度为 $C \times H \times W$，RPN 使用的 anchor 数量为 k，则输出的维度应为 $(2k + 4k) \times H \times W$。由于其输入和输出的内容仅在通道个数上不同，长宽一样，与全卷积神经网络 (FCN) 结构类似，因此 RPN 可视为 FCN 的一种特例。RPN 的训练数据是按以下规则进行划定。对于判断其是否可以成为候选框的标签，若该 anchor 和某个目标框拥有最大的 IoU **或**其与任意一个目标框的 IoU 大于 0.7，则该 anchor 将被视作正例；若该 anchor 和所有目标框的 IoU 小

于 0.3，则该 anchor 将被视作负例；剩余的 anchor 在训练时将不被使用。这一部分的损失函数为 softmax 函数。对于候选框的偏移量，在训练过程中只使用被标记为正例的 anchor，假设 anchor 和其对应的目标框的表示分别为 x_a, y_a, h_a, w_a 和 x, y, h, w，则网络的回归目标 (与 RCNN 一致) 为：

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

RPN 训练时也使用批次，每个批次中的 anchor 来自于同一张图片。每个批次由 256 个 anchor 构成，其中正例和负例各 128 个。若正例数不足 128 个，则用负例补充。RPN 将两个损失函数进行融合，形式如下：

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

损失函数中的两项分别被 $N_{cls} = 256$ 和 $N_{reg} \approx 2400$ 规范化，其中 N_{reg} 是特征图的格点数。超参数 $\lambda = 10$，在这种情况下，两个分支的损失的权重相差不大。

训练过程

Faster RCNN 共有 3 个部分组成：特征提取器是 VGG16 的卷积和池化层，RPN 和 Fast RCNN 均使用 VGG16 得到的特征图作为输入，完成对应模块的任务。在训练时需要经历 5 个过程：**第一**，在 ImageNet-1000 数据集上预训练 VGG16 模型。**第二**，在 1 的预训练模型上训练 RPN。**第三**，在 1 的预训练模型上，使用 2 训练得到的 RPN 训练 Fast RCNN。注意到在 2 中训练 RPN 时，VGG16 部分已经得到微调，但在此时训练 Fast RCNN 时使用的仍是 1 中预训练的模型。因此，此时 RPN 和 Fast RCNN 还没有共享特征图。在训练 Fast RCNN 时，RPN 的参数是固定的。**第四**，使用 3 中微调得到的 VGG16 网络作为预训练模型继续训练 RPN。在此步的训练中，VGG 的参数是固定的，仅有 RPN 的参数得到微调。**第五**，使用 3 中微调得到的 VGG16 网络作为预训练模型，使用 4 中训练得到的 RPN 继续训练 Fast RCNN。在此步的训练过程中，VGG 和 RPN 的参数均是固定的，仅有 Fast RCNN 的参数得到微调。

测试

Faster RCNN 在测试时的思路较为简单。首先将图片通过特征提取器得到特征图。随后先使用 RPN 得到各个 anchor 在特征图上的每个格点上成为候选区域的可能性。Faster RCNN 仅选取分数最高的 300 个 anchor 成为候选区域，并按输出的偏移量移动和缩放 anchor 得到最终的候选区域。这里需要注意，在训练 RPN 时，网络仍然使用了 2000 个 anchor 左右，但在测试时只使用 300 个 anchor 便已足够。随后，将得到的候选区域依次通过 Fast RCNN 部分，得到该候选区域的类别概率分布和偏移量，得到最终的目标框。最后，再使用 NMS 得到最终的测试结果。

2 YOLO 系列

YOLO 系列是目标检测领域的另一种流行的算法，自其提出以来，其已从 v1 一路更新至 v5。与 RCNN 算法属于两阶段检测方法不同，YOLO 系列没有候选区域提出这一步骤，其直

接通过网络将原始图片作为输入，直接输出目标框。YOLO 系列的运行速度较快，在 YOLO v1 时，其测试速度已超过 Faster RCNN，但 YOLO 系列的准确率相比对应的两阶段方法较低。

2.1 YOLO v1

如前所述，YOLO 系列的目标检测器是一阶段的目标检测器，因此在目标检测时，图片将只通过一个网络，直接得到其目标框。YOLO v1 的做法是 (如图2.1所示)：将原始图片分为 $S \times S$ 个格点，若某个目标框的中心落入了该格点中，则该格点则需要对该目标框对应的物体进行检测。网络将会对每个格点输出 B 个目标检测框的位置和其对应的置信度。每个目标检测框可以由 5 个参数表示，分别为 x, y, w, h 和置信度，其中 (x, y) 是检测框的中心位置，h, w 是检测框的长和宽，置信度的计算公式为 $C = \Pr(\text{Object}) * \text{IoU}_{\text{truth}}^{\text{pred}}$。当没有物体落入该格点时，置信度为 0，否则置信度等于预测框和目标框之间的 IoU。**注意到预测框和目标框之间的 IoU 在训练的过程中会不断变化，因此置信度的标签需要在训练时不断进行调整**。对于图片上的每个格点，网络还会输出其在所有类别上的概率分布 $\Pr(\text{Class}|\text{Object})$。**对于每个格点，虽然其在预测时会预测 B 个置信度，但是类别概率分布仅会输出一次**。因此，网络的输出维度为 $S \times S \times (5 * B + C)$。

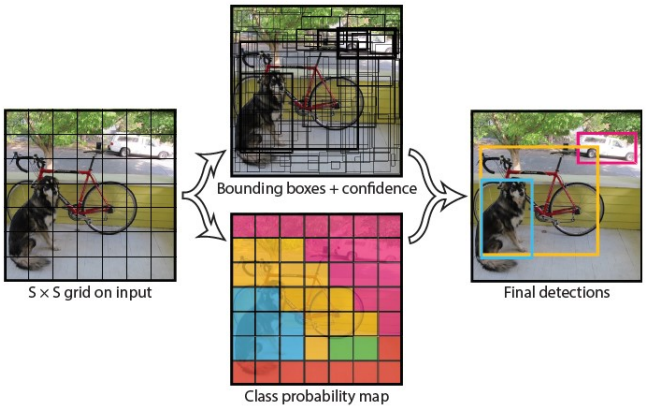

图 2.1: YOLO v1：网络将输入的图片分为 $S \times S$ 格点，通过格点对中心落入格点的物体预测其位置、置信度和类别概率。

网络结构

原始论文中给出了两种不同的网络，较大的网络主要由 24 个卷积层构成，而较小的网络则主要由 9 个卷积层构成。该两个网络的设计思路类似，均由论文的作者提出，名为 Darknet。但是，不论网络的结构是什么形式，其主要形式都相同的。网络先将原始图片通过卷积层和全连接层，转化为一个特征向量。例如，在 YOLO v1 中，网络的输入图片维度为 448×448，其通过多个卷积层和 1 个全连接层后，转变为了长度为 4096 的特征向量。随后，网络再通过一个全连接层或卷积层变为输出的维度 $S \times S \times (5 * B + C)$。例如，在 YOLO v1 中，$S = 7, B = 2$，若将其使用在 VOC 数据集上，则 $C = 20$，其输出维度为 $7 \times 7 \times 30$。YOLO v1 的网络同样先在 ImageNet-1000 上进行预训练。

YOLO v1 的损失函数如下所示，其由 4 部分组成：

$$
\begin{aligned}
l =& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
&+ \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in class} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
$$

这四项分别度量了预测框的位置、大小、置信度和类别概率分布上的损失，下面对其分别进行说明：

1. 预测框的位置和大小是分别在第一项和第二项中进行优化的。这两项在 $S \times S$ 个格点的 B 个预测框上进行求和，且带有系数 λ_{coord}。在 YOLO v1 中，$\lambda_{\text{coord}} = 5$，被给予了较高的权重。这是因为真实框的个数较少，因此在这两项中 0 的个数较多，对于如此稀疏的损失，有必要在不为 0 的位置提高权重，促进收敛。此外 h_i, w_i 在整张图上经过标准化，其取值位于 0 和 1 之间；x_i, y_i 在每个格点中经过了标准化，其取值也位于 0 和 1 之间。在计算预测框大小时，YOLO v1 对 h_i 和 w_i 进行了开根处理。这是因为相比预测框的位置，预测框的大小更为重要 (试想在较小的检测框中，大小的微小变动比位置的微小变动更为重要)。

2. 预测框的置信度是在第三项中进行计算的。置信度也是在 $S \times S$ 个格点的 B 个预测框上进行求和。对于包含物体的格点，其权重为 1；而对于不包含物体的格点，其权重为 $\lambda_{\text{noobj}} = 0.5$。此处权重设置的原因和 λ_{coord} 的设置相同，其降低了不含物体的格点的损失的大小，用来缓解检测框的稀疏性。

3. 最后一项用来优化每个格点的类别概率分布。需要注意两点，首先是这里使用的不再是交叉熵损失函数，而使用了平方损失函数，且同样只在包含物体的格点内计算；其次是这里的损失函数只在 $S \times S$ 个格点计算，即不论 B 的取值为多少，类别概率分布在每个格点处仅计算一次。因此，这里产生了一个十分自然的问题：既然类别概率分布仅计算一次，为什么要预测 B 个预测框呢？YOLO v1 原文中给出的解释如下：*Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.* 我本人对此表达疑问，并且认为对 B 个预测框分别预测一个类别概率分布可能得到一个更好的结果，特别是在格点内存在多个物体的情况下。在 YOLO v1 中，格点仅会对 IoU 最大的物体进行预测，其余物体会被丢弃。

测试

YOLO v1 在测试时，仅需进行一次网络的正向传播。在 $S = 7, B = 2$ 的设置下，网络会输出 $S \times S \times B = 7 \times 7 \times 2 = 98$ 个预测框。同时，网络还会输出这 98 个预测框的置信度和类别分布概率 (2 个预测框共享 1 个类别分布)。置信度和类别分布概率通过乘法变为类别特定的置信度，即：$\Pr(\text{Class}|\text{Object}) \times C = \Pr(\text{Class}|\text{Object}) \times \Pr(\text{Object}) \times \text{IoU}_{\text{truth}}^{\text{pred}} = \Pr(\text{Class}) \times \text{IoU}_{\text{truth}}^{\text{pred}}$。最后，再通过 NMS 将多余的预测框去除，得到预测结果。

2.2 YOLO v2

YOLO v2 对于 YOLO v1 做出多个改进，不仅可以实时完成检测任务，同时还将 VOC 数据集的 mAP 提升了 10 个点左右。YOLO v2 的主要改进包括：

1. 使用 anchor 进行目标框的预测
2. 将检测级的数据和分类级的数据共同进行训练，可以使模型检测出在分类数据而不在检测数据中的类别。YOLO v2 可以通过此方法检测出 9000 个类别的物体，因此 YOLO v2 论文的题目也叫 YOLO9000
3. 其他改进，可小幅度地提升模型的性能

Anchor 的引入

YOLO v1 中仅将原始图片分为 7×7 个格点，并在每个格点上提出 2 个预测框，因此 YOLO v1 最多仅能提出 98 个预测框。提出过少的预测框容易遗漏部分可能存在的物体，引入 anchor 后，提出的预测框的个数会大幅上升。具体地，YOLO v2 按以下方式引入 anchor：首先，YOLO v2 改变了图像的输入维度，由 448×448 降低至了 416×416。这样经过网络的卷积层后得到的特征图的大小为 13×13。输入维度的降低导致了特征图的长和宽都是一个奇数，因而存在一个正中心的格点。正中心的格点对于较大的物体来说十分重要，因为较大的物体的中心往往落在图片的正中心，其在特征图中的中心也在中心格点处。YOLO v2 的 anchor 的大小与 Faster RCNN 中直接通过大小和比例给出得到不同，而是通过聚类方法 (Dimension Clusters) 得到的。该聚类方法以所有标注的检测框为输入，因而 anchor 可以得到较好的先验。

对于 anchor 来说，我们希望提出的 anchor 具有的性质是：其与存在的标注框之间有尽量大的 IoU，这样可以提高 anchor 的召回率。因此，在聚类时距离的定义为 $d(\text{box,centroid}) = 1-\text{IoU(box,centroid)}$，这表示若标注框和聚类中心的 anchor 的 IoU 越高，则两者的距离越近。通过聚类，将标注框聚为 5 类，并将 5 类的中心设置为 anchor。因此，在 YOLO v2 中，提出的预测框的个数为 $13 \times 13 \times 5 \approx 850$。事实上，如果将 anchor 的个数设置为 9，模型的性能可以大幅提升，但此处为了模型的便捷性仅采用 5 个 anchor。

在得到 anchor 后，YOLO v2 在 13×13 的特征图的每个格点上使用 5 个 anchor 进行预测，在每个格点上对每个 anchor 都输出五个指标 t_x, t_y, t_w, t_h 和 t_o，其中假设 anchor 所在格点的左上角距离特征图左上角的偏移量是 (c_x, c_y)，anchor 的大小是 (p_w, p_h)，真实的检测框的标签为 (b_x, b_y, b_w, b_h)。则在预测时遵循如下等式：

$$
\begin{aligned}
b_x &= c_x + \sigma(t_x) \\
b_y &= c_y + \sigma(t_y) \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
\sigma(t_o) &= \Pr(\text{Object}) \times \text{IoU}(\text{Object}, \text{Anchor})
\end{aligned}
$$

其中 $\sigma(.)$ 函数为 sigmoid 函数，起作用在 t_x, t_y 上将其转换到 $[0, 1]$ 之间 (此处表示我们已对 anchor 相对于格点左上角的距离做了标准化)。对 t_o 使用 sigmoid 函数后，输出 $[0, 1]$ 之间的值，表示置信度，其计算方式仍然和 YOLO v1 的计算方式相同。

同时进行分类和检测任务

在现实生活中，分类数据的标注是较为容易的，而检测任务数据的标注则是成本较高的。YOLO v2 将两种类型的数据同时进行训练。其专门使用检测任务的数据进行检测框位置、大小和置信度的学习，而对于物体的类别，则使用分类数据和检测数据共同进行学习。由于检测数据标注较为昂贵，其包含的类别数较少，额外使用分类数据后，当模型在测试时，便可以检测出原本仅属于分类数据的类别了。

将两种类型的数据同时进行训练时，首先需要解决类别冲突问题。例如，在 COCO 数据集中，存在一个类别为狗；但相对应地，在 ImageNet 中，关于狗的类别有 100 多种。我们需要找到一个较好的方式将两个数据集的标签融合起来。ImageNet 使用 WordNet 的数据结构来存储类别之间的关系。WordNet 是一个有向图，若一个类别包含若干个子类别，则将该类别指向子类别。例如，狗类别会指向猎犬，猎犬类别会指向梗犬，梗犬类别会指向约克夏梗犬。但是要注意，WordNet 不是一个树的结构，例如类别狗既会指向犬类，也会指向家畜。但是，WordNet 中的节点的关系较为复杂，直接使用 WordNet 可能导致问题难度过高。ImageNet 将其简化为一个树的结构，简化方式为：若从根节点 (根节点为物体) 出发，有两条路可以到达同一个节点，则仅保留路径较短的一条。通过这种方式，可以将 WordNet 简化为一个有层次的树结构。例如，将 ImageNet-1000 分类数据集按这种规则进行改进，加入中间节点后，可以将标签的个数从 1000 增加到 1369。

YOLO v2 将 ImageNet 分类数据集和 COCO 目标检测数据进行融合。其中，从 ImageNet 数据集中选取出现频次最高的 9000 个类别，将其转变为树的结构，再将 COCO 数据集的 80 个类别添加到对应的层级中，最后构成一个类别共 9418 个的树结构。由于 ImageNet 数据集较大，为保证分类数据和监测数据的相对平衡，原文对检测数据进行过采样，使分类数据和检测数据的比例大约为 4：1。由于当前的类别是以树的结构存储的，因此当前进行类别预测时，其预测需在每一个节点都进行一次，每一层都使用 softmax 函数进行分类，直至类别所在层次停止。例如，树的根节点为物体，其子节点包括动物，人工制品和自然物体，则分类器先在这 3 个类别上进行分类，随后继续到对应的子节点继续进行分类。因此，通过这种方式，一个物体属于某个类别的概率可以通过不断的使用条件概率相乘得到。例如 $P($ 香蕉 $) = P($ 香蕉 $|$ 水果 $) \times P($ 水果 $|$ 自然物体 $) \times P($ 自然物体 $|$ 物体 $)$。在对分类数据和检测数据同时进行训练时，网络仅使用 3 个 anchor，anchor 还是由聚类得来的。当网络遇到检测数据时，网络会像往常一样反向传播所有梯度；但当网络遇到分类数据时，网络仅会回传分类损失一支的梯度。

其他改进

YOLO v2 还对网络做出一些其他改进，其可小幅度地提升网络的性能。首先，网络使用了 **Batch Normalization** 技术，在每个卷积层后加上一个 BN 层，对网络做出一些正则化。其次，YOLO v2 **加大图像的输入维度**，将输入的图像的维度从 224×224 增加到 448×448。由于网络需要首先在 ImageNet 数据集上做预训练，而 ImageNet 训练时的维度为 224×224(这是因为，ImageNet 数据集较大，直接使用 448×448 的输入维度会使网络优化地非常慢)。因此，在模型转向检测任务进行微调之前，网络多加了一个微调过程，其以 448×448 的输入维度继续在 ImageNet 数据集的分类任务上训练 10 个回合。此外，网络还**加入了一个直通层**。YOLO v2 输出的特征图为 13×13，在该特征图下采样之前的维度是 26×26，网络的特征图将 26×26 的特征图按 2 的步长下采样后，与 13×13 的特征图直接相连接。该操作与 ResNet 中的恒等映射操作类似。**多尺度训练**也是 YOLO v2 的改进方式之一。YOLO v2 的输入维度为

448×448，网络会对图像进行 32 倍的下采样，最后构成 13×13 的特征图。多尺度训练指每 10 个批次，模型使用不同的输入维度进行目标检测 (这样仅仅需要将 anchor 作用到大小不同的特征图上，不改变模型的整体结构和训练过程)。YOLO v2 使用的尺度为 $\{320, 352, ..., 608\}$。多尺度训练会增加训练时间，但可以提升检测器的性能。最后，YOLO v2 对其**主干网络进行了调整**。DarkNet-19 为其所用模型，其较为轻量级。

2.3 YOLO v3

YOLO v3 相比 YOLO v2 所添加的变化较少。v3 模型相比 v2 模型大小仅有小幅度的上升，但性能有较大幅度提升。YOLO v3 的改进主要包括：主干网络的调整，在 3 个不同特征图上使用 anchor 进行检测框的预测和置信度、类别预测时的小调整。

YOLO v3 将 DarkNet-19 升级到了 DarkNet-53，DarkNet-53 连续使用 BottleNeck 结构，即先通过一个 3×3 的卷积层，随后通过一个 1×1 的卷积层，再通过一个 3×3 的卷积层，最后使用 ResNet 中的恒等映射操作。其中，网络会在第一个 3×3 的卷积层处做下采样操作，将特征图的长和宽均缩小为原来的一半。网络的输入维度为 256×256，经过 5 次下采样，最后输出 8×8 的特征图。

此外，YOLO v3 在三个不同大小的特征图上均输出预测框。YOLO v3 在 8×8，16×16，32×32 三个特征图上都使用 anchor 在每个格点处输出预测结果。次方法是由 SSD 所提出的，具体可见第3.1节，此处不进行详细讨论。YOLO v3 使用了 9 个 anchor，anchor 的大小仍然是由聚类方法得到的。

YOLO v3 还对置信度和类别的预测方式提出了改进。对于类别的分类方法，其不再使用 softmax 损失函数直接进行多类别的分类，而是对每个类别进行一个二分类问题。这可以提高具有重合的检测框的格点的预测准确性。对于置信度，YOLO v3 也不再使用 IoU 进行计算，而是使用逻辑回归将置信度预测转化为一个二分类问题。一个 anchor 的置信度为 1 当且仅当该 anchor 和某个标注框有最大的 IoU，且该 IoU 大于 0.5；否则该 anchor 的置信度即为 0。在训练时，置信度标注为 1 的 anchor 将反传标注框的位置、大小和置信度的梯度，而置信度标注为 0 的 anchor 将只回传置信度的梯度。

3 其他

RCNN 系列和 YOLO 系列是两个较为经典的目标检测器。此外还存在部分非常重要且非常有趣目标检测器，其对 RCNN 系列和 YOLO 系列的改进可以分为以下几类：**第一**、将预测框的提出从一个特征图上变换到多个特征图上，以确保在各个不同的感受视野上可以检测到不同尺度的物体。使用多个特征图提出预测框的代表性方法包括 SSD 和 FPN。**第二**、使用不同的损失函数。由于在目标检测任务中，大部分提出的候选区域或 anchor 都是不带有物体的，其均为负例，而包含物体的仅为很少一部分。因此，目标检测问题实质上是类别不平衡问题，Focal Loss 是解决该问题的一种代表性方法。**第三**、anchor 自从被提出以来，一直都是目标检测任务的首选和关键性技术，其相当于预测框的先验信息，可以从不同的大小和长宽比例适应不同的物体，其对于网络的性能的提升也非常明显。但是同时，anchor 给网络带来了参数量较大的缺点，这导致网络在测试时的速度有所下降。因此，使用一些 anchor-free 的目标检测结构也成为了重要的研究方向，其中代表性的方法为 FCOS。

3.1 SSD

SSD 是一种一阶段的目标检测方法，其与使用 anchor 后的 YOLO v1 非常相似，区别在于 YOLO 仅在一个特征图上提出预测框，而 SSD 会在多个特征图上提出预测框。如图3.1所示，SSD 在 6 个大小不同特征图上对检测框进行预测。同时，SSD 在 6 个不同的特征图上都使用 anchor，因此相比于 YOLO v1 仅提出 98 个预测框，SSD 可以提出 8000 多个预测框。提出数量较多的预测框不仅可以提高包含物体的检测框的召回率，还可以通过不同大小和长宽比例适应不同的物体。

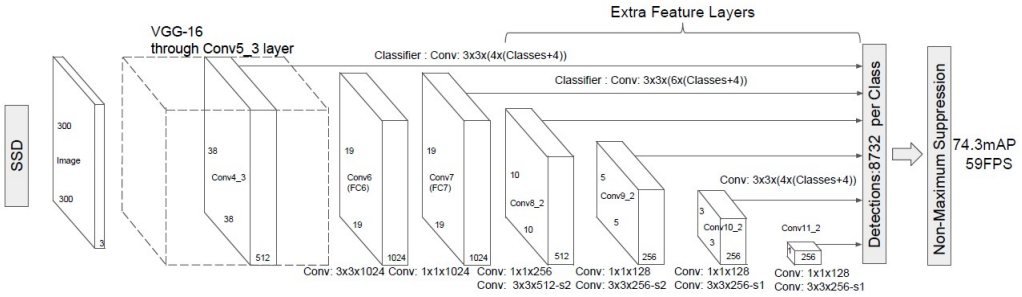

图 3.1: SSD 工作原理：SSD 使用 VGG 网络作为特征提取器，得到 38×38 的特征图。随后，其不断对特征图进行下采样，分别得到大小为 19×19、10×10、5×5、3×3 和 1×1 的特征图。在 6 个大小不同的特征图，SSD 均使用 anchor 提出预测，对于特征图上每个格点的 anchor，网络均对其类别分布和相对 anchor 的偏移量进行预测。通过 anchor 在多个特征图上的预测，SSD 相比 YOLO 大幅度增加了预测框的个数，从近 100 个上升到了 8000 多个。

模型结构

SSD 使用 VGG-16 作为特征提取器，其图片的输入维度 300×300。在主干网络阶段，其经历了经历了 8 倍的下采样，得到的特征图为 38×38。注意到，在原始的 VGG 网络中，之后将连接一个池化层和三个全连接层，继续进行分类问题，在检测问题中，这些层会被丢弃。之后 SSD 会在此 38×38 的特征图上使用 anchor 提出一批预测框进行判断。同时，SSD 会对该 38×38 的特征图不断继续向前通过卷积层进行下采样得到新的特征图。新的特征图的长宽依次为 19，10，5，3，1；其通道数依次为 1024，512，256，256，256。这表明，SSD 每次的下采样的倍数为 2，即不断将特征图的长宽变为原来的一半。由于，这些特征图的大小从低到高不断变小，其形象类似金字塔，因此这些特征图构成的层也被称为特征金字塔。此外原始的 38×38 的通道数为 512，之后的特征图的通道数会先增加到 1024，再一半一半的递减到 256 后保持不变，最后保持在 256 的原因是防止特征图信息损失过多。在这些特征图上，SSD 都会使用 anchor 提出预测框，并进行下一步的预测。

在这些特征图上，anchor 的使用方式和 Faster RCNN 中 anchor 的使用方式完全一致。假设特征图的维度为 $m \times n \times p$，在该特征图上使用一个较小的 3×3 的卷积核，卷积核的输入通道数为 p，输出通道数为 $k(c+4)$。其中，k 是 anchor 的个数，c 是类别的个数，4 表示真实的标注框和对应的 anchor 之间的偏移量，该偏移量的 4 个的计算方式与 Faster RCNN 一致。因此，对于输入维度为 $m \times n \times p$ 的特征图，经过 anchor 预测的卷积层，其输出维度为 $m \times n \times k(c+4)$。此外，需要指出，不同的特征图使用的预测卷积层是共享参数的，即所有特征图使用同一个卷积层对 anchor 进行分类和位置大小校正。

由于不同的特征图大小不同，在网络中所处的位置也不同，其拥有不同的感受野，因此在不同的特征图上使用不同大小的 anchor 是更为合适的。假设我们共有 m 个特征图，则第

$k(1 \le k \le m)$ 个特征图的大小为 $s_k = s_{\min} + \frac{s_{\max}-s_{\min}}{m-1}(k-1)$。其中 $s_{\min}, s_{\max}$ 是需要设定的超参数，其分别代表最小的特征图和最大的特征图的大小参数，在 SSD 中分别设置为 0.2 和 0.9。SSD 还对 anchor 的长宽比例设定了 5 个选项：$a_r \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$。Anchor 的长和宽可分别按 $w_k^a = s_k\sqrt{a_r}$ 和 $h_k^a = s_k/\sqrt{a_r}$ 求得。此外，对于 $a_r = 1$，SSD 还添加了一个大小为 $s'_k = \sqrt{s_k s_{k+1}}$ 的 anchor。因此，对于每个特征图的每个格点上，SSD 均使用 6 个 anchor。

测试

SSD 的测试过程与之前介绍的目标检测器较为类似，将一张图片输入网络，通过网络便可得到一系列预测框，通过 NMS 算法即可得到最终的预测结果。

3.2 FPN

FPN 同样是使用多尺度特征图提出预测框的方法。SSD 的特征图是网络前向传播中的不同层数的特征图，这些特征图拥有不同的感受野和语义信息，感受野较小的特征图拥有较高语义信息而感受野较大的特征图所拥有的语义信息则较低。SSD 同等对待这些特征图，因此感受野较大的特征图没有能够得到复用，而这对于检测较小的物体帮助较大。FPN 通过将**具有较大感受野和较低语义信息的特征图**和**具有较小感受野和较高语义信息的特征图**结合起来，使得特征金字塔中的各个特征图都具有较高的语义信息。图3.2的左图展现了各种不同的网络的预测框提出方式，其中 (d) 为 FPN 的结构。图3.2的右图具体展现了 FPN 的结构，其将网络中的特征图通过 1×1 的卷积将网络中的特征图转换为新的特征图，并将新的特征图从上往下进行累加。预测框将在新的特征图上进行提出。

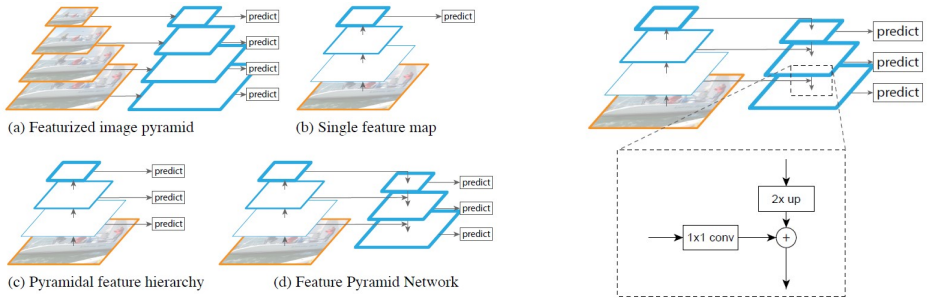

图 3.2: **左：**(a) 是最原始的图片金字塔，将图片变换为不同大小的特片进行下一步处理。(b) 是单特征图的目标检测器，其仅使用最高层的特征图进行预测。YOLO v1 是这类目标检测器。(c) 是多特征图的目标检测器，其使用不同层次的特征图进行预测。SSD 是这类目标检测器。(d) 是 FPN，其将原始向上传播的特征图通过卷积变换和向下累加操作变换为新的特征图。**右：**。FPN 新特征图的计算方式，其通过 1×1 的卷积将同层的原特征图和上采样的更高层的新特征图进行求和依次得到新的特征图。

模型结构

FPN 的模型结构由三个部分组成：一个自下而上的通路、一个自上而下的通路和连接两个特征金字塔的连接结构。其中**自下而上的通路**是主干网络的一部分。FPN 使用的主干网络为 ResNet，ResNet 由残差块构成，每个残差块由 3×3 的卷积和恒等映射构成。在特定的残差块中，其会对特征图进行下采样，使特征图的长和宽变为原来的一半。在 FPN 中，FPN 将具有相同大小的特征图所在的层称为同一阶段。对于每一个阶段，将其定义为特征金字塔的一个层次，每个阶段的最后一层即为 FPN 的特征图。这里之所以选取阶段的最后一层作为特

征图，是因为每个阶段的最后一层具有最多的信息。FPN 使用了 ResNet 的第 2、3、4、5 阶段的最后一层构成特征图，这些特征图分别被记为 $\{C_2, C_3, C_4, C_5\}$，其下采样倍数分别为 4，8，16 和 32(下采样倍数为 N 表示特征图的长宽为原图片的 $\frac{1}{N}$)。注意到 C_1(下采样倍数为 2) 的最后一层没有纳入特征金字塔.

自上而下的通路作用在新的特征金字塔上，其通过上采样的方式由高感受野的特征图向低感受野的特征图传播。新的特征金字塔中的特征图是由原金字塔中对应的特征图经过 1×1 卷积变换，同时通过**连接结构**进行加强得到的。记新的特征金字塔的特征图分别为 $\{P_2, P_3, P_4, P_5\}$，其中 P_i 和 C_i 为对应特征图，处于同一层，两者的大小相同。P_i 由 C_i 和 P_{i+1} 通过连接结构都到，计算方式为 $P_i = \text{conv}_{1\times1}(C_i) + \text{upsample}_2(P_{i+1})$。连接结构将原特征金字塔对应的特征图通过 1×1 的卷积变换与新的金字塔中更高层的特征图 2 倍上采样通过相加后得到的。其中 P_5 是直接由 C_5 卷积变换得到的。

应用

FPN 原始论文是将 FPN 应用到 Faster RCNN 中，使其变为一个更强大的两阶段目标检测器。但是 FPN 也可以应用到一阶段目标检测器中，例如接下来的 RetinaNet 中，在 FPN 的结构上使用 Focal Loss 作为损失函数，也可以使一阶段的目标检测器拥有两阶段目标检测器的性能。FPN 可以分别用于 RPN 和 Fast RCNN 中。

将 FPN 使用到 RPN 中时，RPN 直接将原始的单个特征图变为 FPN 的多个特征图。在不同的特征图上，RPN 使用大小不同但长宽比例相同的 anchor。在 $\{P_2, P_3, P_4, P_5, P_6\}$ 上，anchor 的大小分别为 $\{32^2, 64^2, 128^2, 256^2\, 512^2\}$，anchor 的长宽比例为 $\{1:2, 1:1, 2:1\}$。因此，在 RPN 中，FPN 在每个特征图上使用 3 个 anchor，共 15 个 anchor。注意到，在原始的 FPN 中，特征图仅到 P_5，但在 RPN 中，FPN 对 P_5 进行了一次比例为 2 的下采样构成 P_6。与原始 FPN 相似，在此处网络对每个特征图也使用 3×3 的卷积和两个并行的 1×1 的卷积层对候选区域进行预测。在所有特征图上，预测候选区域的卷积层的参数是共享的，且通过实验证明，在卷积处使用共享的参数可以达到更好的表现。在此处训练数据也是由 IoU 进行划分的，正例表示 anchor 与某一个标注框的 IoU 大于 0.7，负例表示 anchor 与所有的标注框的 IoU 小于 0.3。若一个 anchor 既不满足正例的判断条件，也不满足负例的判断条件，则在训练时该 anchor 不会被使用。经过实验发现，将 FPN 使用到 RPN 后，这些 anchor 对真实的标注框的召回率有极大的提升。

将 FPN 使用到 Fast RCNN 中时，由于 FPN 含有多个特征图，因此我们要决定，对于输入的图片，我们到底把图片映射到哪一个特征图中。假设输入图片的长和宽分别为 w 和 h，则我们将其映射到特征图 P_k 上，其中 $k = [k_0 + \log_2(\sqrt{wh}/224)]$。公式中，224 是网络的预定输入图片的维度，因此 k_0 是关键的一层，其候选区域的原始大小应为 $w \times h = 224^2$。在此处，$k_0 = 4$，因为在 ResNet 为基础的 Faster RCNN 中，其原本将 C_4 作为特征图。对于不同特征图上的候选区域，其首先通过一个 RoI 池化层将其变为 7×7 固定大小的特征图，并在之后连接两个 1024 为的全连接层 (均添加 ReLU 函数)，最后加上两个并行的分支，分别进行分类和目标框校正。

3.3 RetinaNet

RetinaNet 是一阶段的目标检测器，其与之前一阶段目标检测器的最大不同是其使用了全新的损失函数：**Focal Loss**。Focal Loss 的提出使得一阶段目标检测器在性能上可以接近两阶段目标检测器。Focal Loss 的思想是，由于在一阶段目标检测问题中，在使用 anchor 时，背

景所代表的负例的比例远高于正例，导致该问题在正负例上存在不平衡现象。Focal Loss 认为正负例不平衡是导致一阶段目标检测器性能下降的主要原因，其通过改进交叉熵损失函数得到。Focal Loss 增大了难例的权重并减小了简单例的权重，防止了损失函数被简单的负例 (尤其是背景类) 拖累。通过将 ResNet、RPN 和 Focal Loss 三者结合，可以得到接近两阶段目标检测器性能的一阶段目标检测器 RetinaNet。

Focal Loss

在传统的目标检测器中，对物体进行分类时使用的基本都是交叉熵损失函数。假设该物体属于类别 t，且网络预测其为类别 t 的概率为 p_t，则交叉熵损失函数为 $\mathrm{CE}(p_t) = -\log(p_t)$。Focal Loss 在交叉熵损失前乘上一项系数 $(1-p_t)^\gamma$，其中 γ 是专注程度系数。因此，Focal Loss 为：

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

Focal Loss 具有以下性质：第一、当一个样本被错分且 p_t 非常小时，则 $(1-p_t)^\gamma$ 接近于 1，损失函数在该样本上几乎没有受到影响；而当 $p_t \to 1$ 时，系数 $(1-p_t)^\gamma \to 0$，则损失函数在该样本上的权重被降低。这说明，对于分类正确或比较有信心的样本，Focal Loss 降低其权重；而对于分类错误的样本，Focal Loss 则几乎不改变其损失函数值，由于分类正确样本的权重被降低，相当于这部分的样本的权重被提高了。第二，参数 γ 是用来调整简单样本权重降低的程度。当 $\gamma = 0$ 时，Focal Loss 等价于交叉熵函数，当 γ 上升时，简单样本权重下降的程度越厉害。原始论文中发现 $\gamma = 2$ 在实验中表现最好。注意到，在 $\gamma = 2$ 时，对于 $p_t = 0.9$ 的样本，其权重会下降 100 倍，而对于 $p_t = 0.968$ 的样本，其权重会下降 1000 倍。这相当于提升了错分样本的重要性 (对于 $p_t \le 0.5, \gamma = 2$，其损失值最大下降 4 倍)。

在实际使用时，Focal Loss 还使用系数 α_t 对损失函数进行规范化。研究发现，使用规范化的损失函数可以小幅度地提升模型性能最终的 Focal Loss 为：

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

网络结构

RetinaNet 由三部分组成，可如图3.3所示。第一部分是担任主干网络或特征提取器的 ResNet，第二部分是作用在 ResNet 高层特征图上的 FPN，第三部分是在 FPN 提供的特征图上的两个并行的子网络，第一个子网络进行物体的分类，第二个子网络进行检测框的校正。在 ResNet 的基础上构建 FPN 可以增强网络的特征表示能力并构建了一个丰富，多尺度的特征金字塔。特征金字塔是在 ResNet 的 C_3 至 C_5 上构建的，通过 C_3 至 C_5，我们可以通过 FPN 得到 P_3 至 P_5，P_6 是在 P_5 的基础上通过一个 3×3、步长为 2 的卷积层和 ReLU 函数得到的，P_7 是在 P_6 上通过同样的方法得到的。P_3 至 P_7 是 FPN 输出的特征图，l 是特征图的层数 (P_l 层的特征图的长和宽是原始图片的 $1/2^l$)，P_3 至 P_7 的通道数均为 256。

RetinaNet 同样在 FPN 输出的特征图上使用 anchor，在 P_3 至 P_7 上，该层的 anchor 的基准大小依次为 32^2 至 512^2。在每一层上，anchor 同样拥有 3 个长宽尺度，分别为 $\{1:2, 1:1, 2:1\}$。同时，为了增大 anchor 的多样性，在每一层上 anchor 的长宽会分别变为原始长宽的 $\{2^0, 2^{1/3}, 2^{2/3}\}$。因此，在每一层上，均使用 9 个不同的 anchor。每个 anchor 都会被赋予一个长度为 K 的独热向量 (K 为物体的类别数) 和长度为 4 的与真实框的偏移量。在 RetinaNet 中，如果 anchor 和真实框的 IoU 大小 0.5，则其会被分配为对应的真实框，若 anchor 与所有

真实框的 IoU 均小于 0.4，则其会被认作负例，其余的 anchor 在训练时会被忽视。与真实框的偏移量将仅在 anchor 为正例时才会被计算。分类子网络和回归子网络的结构可如图3.3所示，其均先通过 4 个 3×3、通道数为 256 的卷积层，在卷积层后均使用 ReLU 函数添加非线性。最后两者均使用一个 3×3 的卷积层变换至对应的通道数，其中分类子网络的最终通道数为 $9\times$ 类别个数，回归子网络的最终通道数为 $9 \times 4 = 36$。注意，其中 9 为每一层 anchor 的个数。此外在所有的 anchor 上，两个子网络的参数是共享的。

在分类子网络上，RetinaNet 使用 Focal Loss 作为损失函数。RetinaNet 使用的超参数为 $\gamma = 2, \alpha = 0.25$。Focal Loss 是在所有约 100k 个 anchor 上计算的。此外需要指出，当 γ 变大时，α 需略为变小，已获得更好的表现。

网络的初始化按如下操作：ResNet 是在 ImageNet-1000 上初始化的，FPN 和两个子网络的卷积层均是由正态分布随机初始化的，卷积层的偏置项的初始值为 $b = -\log(\frac{1-\pi}{\pi})$，$\pi$ 为 anchor 被标为正例的比例，在论文中取值为 0.01.

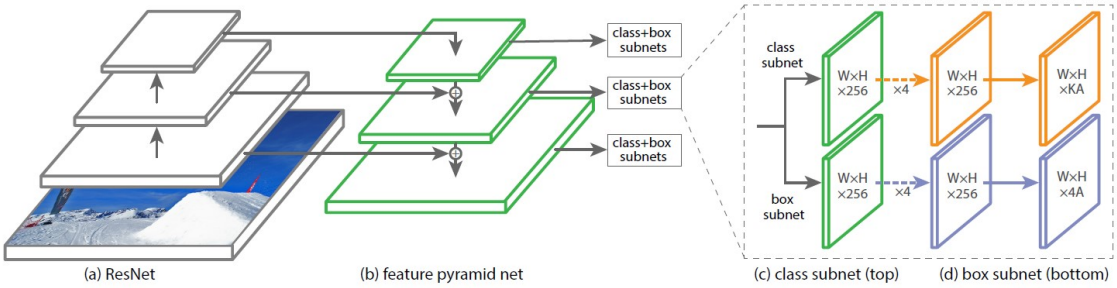

图 3.3: RetinaNet 工作原理：RetinaNet 以 ResNet 作为主干网络，在 ResNet 上添加 FPN 以加强特征的表示能力，随后在 FPN 的特征层上使用 anchor 进行检测框的预测。检测框的生成主要有两个子网络完成，其中分类子网络中使用 Focal Loss 作为损失函数。

3.4 FCOS

在之前介绍的目标检测器中，anchor 几乎是必不可少的组成部分。Anchor 作为一种预测框的先验信息，可以通过定义不同大小不同比例不同位置的大量的 anchor 召回大部分的预测物体，并仅需对 anchor 进行微调即可得到高质量的预测框。但是 anchor 在带来性能提升的同时，也面临着计算量增大的问题。例如，在 RetinaNet 中，网络会提出约 100k 个 anchor，对于每个 anchor 网络都需逐一对其进行分类和修正，这样网络的计算量非常大，在预测时速度会不可避免的下降。FCOS 是一种不使用 anchor 的一阶段目标检测器，通过去除 anchor 不仅避免了复杂的计算，还省去了 anchor 处的超参数设置的问题。FCOS 通过在网络中新增中心度分支，在性能上达到了使用 anchor 的目标检测器的性能。

网络结构

如图3.4所示，FCOS 的主干网络部分与 RetinaNet 非常相似，其使用的特征提取器为 ResNeXt，在 ResNeXt 上同样使用 FPN。FPN 在原始网络的 C_3, C_4, C_5 上构成 P_3, P_4, P_5，并在 P_5 的基础上继续进行下采样构成 P_6, P_7。网络的输入维度为 800×1024，P_3 至 P_7 的下采样倍数分别为 8、16、32、64 和 128。与使用 anchor 的方法在特征图上对 anchor 做调整不同，FCOS 直接对特征图提出特征图。不同的特征图使用相同的参数进行三分支预测，三个分支分别是类别预测分支，中心度分支和回归分支。其中类别预测分支和中心度分支是由同一组变换得来的，回归分支是由另一组变换得来的。变换均是由 4 个连续的卷积变换得来的，

在卷积变换时，特征图的大小不会改变。因此，若特征图的输入维度为 $H \times W$，则三个分支的输出维度均为 $H \times W$。

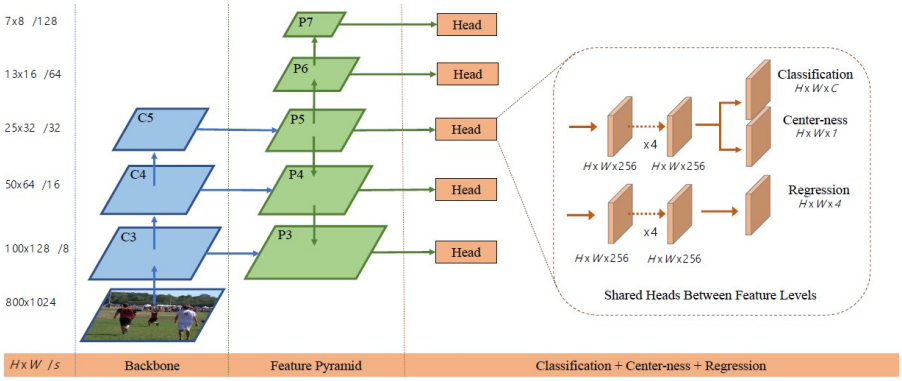

图 3.4: FCOS 的工作原理：FCOS 同样使用 FPN 作为底层网络，并在 FPN 的 5 个不同尺度的特征图进行目标框的预测。FCOS 在预测目标框时不使用 anchor，而是通过三个不同的分支，分类分支预测其类别，回归分支预测特征图上的格点在原图上的点上至物体四条边的距离，中心性分支预测该格点的位于物体的中心程度。

损失函数

对于特征图上的每一个格点 (p_x, p_y)，可通过网络得到其在原始图片上的位置 (x, y)。若原始图片上 (x, y) 落入了某个标注框 $B_i = (x_0^{(i)}, y_0^{(i)}, x_1^{(i)}, y_1^{(i)}, c^{(i)})$，其中 $(x_0^{(i)}, y_0^{(i)})$ 为标注框的左上角坐标，$(x_1^{(i)}, y_1^{(i)})$ 为标注框的右下角坐标，$c^{(i)}$ 为该标注框所代表的物体的类别。对于该特征图上的格点 (p_x, p_y)，若其在原图中的位置 (x, y) 落入了标注框，则需对该特征图上的格点进行损失函数的计算，对于三个分支，其损失函数按如下方法计算：

对于类别预测分支，其预测对象即为 B_i 的类别 $c^{(i)}$，该分支的损失函数使用 Focal Loss 损失函数进行计算。对于回归分支，对于每个格点，其需输出 4 个值 (l, t, r, b)，其目标值分别为 $l^* = x - x_0^{(i)}, t^* = y - y_0^{(i)}, r^* = x_1^{(i)} - x, b^* = y_1^{(i)} - y$。这表示该特征格点 (p_x, p_y) 的预测目标分别为该特征格点在原始图片中所对应的点到其落入的目标框的四条边的距离。回归分支的损失函数为 IoU 损失函数，IoU 损失函数的计算是基于两个检测框的，计算方式如下：

$$\mathrm{IoU}(box_1, box_2) = -\ln \frac{Intersection(box_1, box_2)}{Union(box_1, box_2)}$$

中心性分支是 FCOS 新提出的一个分支。对于 FCOS，只要特征图上的格点对应于原始图像中的格点落入了某个标注框，则该格点就将被视为正例，不管其处于标注框中的哪个位置。因此，对于一些位于标注框边缘的格点，由于其回归目标的 4 个值在数量级上相差较大，其得到的预测框效果较差。FCOS 在分类分支旁新加两个中心性分支来预测该格点的中心性。中心性表示该格点位于对应标注框的中心程度，其计算方法如下：

$$\overset{*}{\mathrm{Centerness}} = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}$$

中心性的取值范围为 $[0, 1]$。在训练时使用交叉熵函数进行损失函数的计算，其标签可视为 $(\mathrm{Centerness}^*, 1 - \mathrm{Centerness}^*)$。因此，损失函数是由三个分支的损失函数相加得到的。同样的，在对不同部分的损失函数求和时，可以对不同的部分赋予不同的权重，在 FCOS 中，三部分的权重均为 1。损失函数仅在正例的特征格点上进行计算。正例的特征格点的意思为，该

格点所对应的原始图片上的格点至少落入了一个标注框。若对应的原始格点落入了多个标注框，那么 FCOS 取面积最小的标注框作为该格点的目标。注意到，RetinaNet 在特征图上的每个格点上使用 9 个 anchor，因此 FCOS 输出的变量个数是 RetinaNet 的九分之一。输出更少的预测框可能会带来真实框低召回率的问题，不过由于 FPN 提出的不同尺度的特征图，低召回率问题在 FCOS 上没有出现。

测试

FCOS 的测试过程与之前的目标检测器有所不同。对于一张图片，网络会输出若干个预测框，这些预测框分别有类别输出、回归输出和中心性分支。根据该预测框对应的特征图的格点，可得到原始图像上对应的点，根据回归输出的四个参数，可以得到预测框的位置和大小。预测框的类别是由类别输出和中心性分支得到的，其类别分数为有类别输出的分数乘以中心性分支的输出。因此，中心性分支的存在可以降低原理目标物体中心的预测框的权重。最后，FCOS 同样使用 NMS 算法得到最终的输出框。

参考文献

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[2] Ross Girshick. Fast r-cnn. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 1440–1448, USA, 2015. IEEE Computer Society.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[5] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[6] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[7] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[8] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020.

[10] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9626–9635, 2019.