YOLO 系列综述-Part 2：

YOLOv1 的伏笔——YOLOv2,YOLOv3 的诞生

目录

1. 简介

实时物体检测已经成为众多应用中的一个重要组成部分，横跨自主车辆、机器人、视频监控和增强现实等各个领域。在各种物体检测算法中，YOLO（You Only Look Once）框架因其在速度和准确性方面的显著平衡而脱颖而出，能够快速、可靠地识别图像中的物体。自成立以来，YOLO 系列已经经历了多次迭代，每次都是在以前的版本基础上解决局限性并提高性能（见表格 1）。

表格 1：历代 YOLO 发布时间、主要改动及作者，由笔者同学、四川大学硕士贺宇劼整理。相关文章链接：
https://blog.csdn.net/qq_54478153/article/details/139477271

时间	版本	主要改动	作者
2015	v1		Joseph Redmon & Ali Farhadi
2016	v2	引入Batch Normalization 和锚框(anchor boxes)	Joseph Redmon & Ali Farhadi
2018	v3	更高效的骨干网络、引入特征金字塔和PAN	Joseph Redmon & Ali Farhadi
2020	v4	引入SPP模块、CSP模块、Mish激活函数、CmBN归一化	Alexey Bochkovskiy
2020	v5	引入Focus结构	Ultralytics
2021	x	引入解耦头 (Decoupled Head)、不再预设锚框	旷视
2022	v6	引入RepConv，量化相关内容，简化解耦头	美团
2022	v7	Efficient Layer Aggregation Network (ELAN)模块，MP模块，Rep	台湾中央研究院
	v8	C2f模块	Ultralytics
2024	v9	Generalized Efficient Layer Aggregation Network (GELAN)	台湾中央研究院
2024	v10	整体高效的网络设计、空间-通道解耦下采样	清华

在本系列文章中，笔者将回顾 YOLO 框架的发展历程，从开创性的 YOLOv1 到最新的 YOLOv10，逐一剖析每个版本的核心创新、主要差异和关键改进。本系列不仅会探讨 YOLO 各版本的技术进步，还将重点讨论在追求检测速度与准确性之间的平衡。

本文为该系列的第二章，笔者将深入分析 YOLOv2 及 YOLOv3 模型。在系列的第一部分，我们对原始 YOLO 模型的基本原理和架构进行了全面的探讨。本章将继续这一工作，详细审视 YOLOv2 和 YOLOv3 的精细调整和关键特性，并探讨它们相较于初代模型的显著改进和性能提升。通过对比分析，本文旨在为读者提供一个清晰的视角，以理解 YOLO 框架的演进及其在物体检测领域的深远影响。

文章攥写时间短，未能仔细校对。可能存在笔误、描述不准确、重要内容缺失等问题。希望大家能指出，笔者会随时修改补充。

注：该系列文章中，所有加粗并配有下划线的关键词，均配有交叉引用。

曹倬瑄

2024/6/15 于惠州

2. YOLOv2：性能提升与技术创新

2.1. 引言&背景

2017 年，作者 Joseph Redmon 和 Ali Farhadi 在 YOLOv1 的基础上，进行了大量改进，提出了 YOLOv2 和 YOLO900。

当时 SSD（Single Shot MultiBox Detector）是 YOLOv1 的强大竞争对手，SSD 在当时展示了更高的实时处理精度。与基于区域的检测器相比，YOLOv1 的坐标定位误差更高，recall（召回率：衡量定位所有目标物体的好坏程度）更低。YOLOv2 作为 YOLO 的第二个版本，它希望重点解决 YOLOv1 召回率和定位精度方面的不足的问题，同时使其更快。

相比于 YOLOv1 是利用全连接层直接预测边界框的坐标，YOLOv2 借鉴了 Faster R-CNN 的思想，引入 Anchor 机制。利用 **K-means 聚类**的方法在训练集中聚类计算出更好的 Anchor 模板，大大提高了算法的召回率。同时结合图像细粒度特征，将**浅层特征与深层特征**，有助于对小尺寸目标的检测。

YOLO9000 使用 WorldTree 来混合来自不同资源的训练数据，并使用联合优化技术同时在 ImageNet 和 COCO 数据集上进行训练，能够实时地检测超过 9000 种物体。由于 YOLO9000 的主要检测网络还是 YOLOv2，所以本文主要以讲解应用更为广泛的 YOLOv2 为主。

2.2. 主要改进点

- **批归一化（Batch Normalization）**：提高了模型训练的收敛速度，同时减少了对其他正则化形式的需求。
- **高分辨率分类器**：通过在更高分辨率的输入图像上微调分类网络，增强了模型对对象的识别能力。
- **锚框（Anchor Boxes）**：使用基于卷积层的锚框来预测边界框，简化了问题，提高了学习效率。
- **维度聚类（Dimension Clusters）**：自动为网络选择合适的边界框尺寸，而不是手动选择，提高了模型的泛化能力。
- **直接位置预测**：改进了边界框位置的预测方式，使得模型更稳定，学习更快。
- **细粒度特征（Fine-Grained Features）**：通过添加 passthrough 层，提高了对小对象的定位能力。
- **多尺度训练（Multi-Scale Training）**：允许模型在不同尺寸的输入图像上运行，提供了速度与准确性之间的灵活权衡。
- **技术创新**：YOLOv2 引入了 **Darknet-19** 作为其基础网络，这是一个定制的、高效的卷积神经网络，专为 YOLOv2 设计。
- **联合训练**：YOLOv2 提出了一种新颖的联合训练方法，允许模型同时在 COCO 检测数据集和 ImageNet 分类数据集上进行训练，从而扩展了模型能够检测的对象类别数量。

3. YOLOv2 核心技术点分析

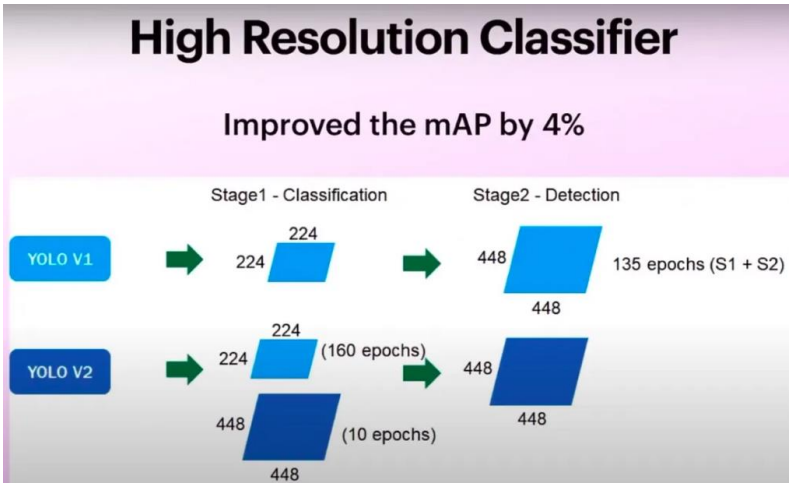
3.1. 尺寸相关改动

3.1.1. High Resolution Classifier（高分辨率的分类器）

YOLOv1 以 224x224 的分辨率训练分类器网络，并将分辨率提高到 448x448 以进行检测。这意味着网络必须同时切换到学习对象检测并适应新的输入分辨率。同时切换到学习对象检测并适应新的输入分辨率可能会导致 YOLO v1 的性能下降问题。

如果直接切换图片尺寸即分辨率大小，模型可能难以快速地对高分辨率的图片进行反应。因此，YOLOv2 训练由 2 个阶段组成。先训练一个像 VGG16 这样的分类器网络。然后用卷积层替换全连接层，并对其进行端到端的重新训练以进行目标检测。

其步骤为：先采用 224x224 的 ImageNet 图像数据集进行约 160 个 epoch 的预训练。然后将预训练模型的权重加载到新模型中，再将输入图像的尺寸更改为 448x448，再训练 10 个 epoch，进行微调。因为网络已经掌握了一些对物体检测很重要的特征，所以新模型的训练周期比预训练模型要少。如下图所示：



这样做有两个原因：

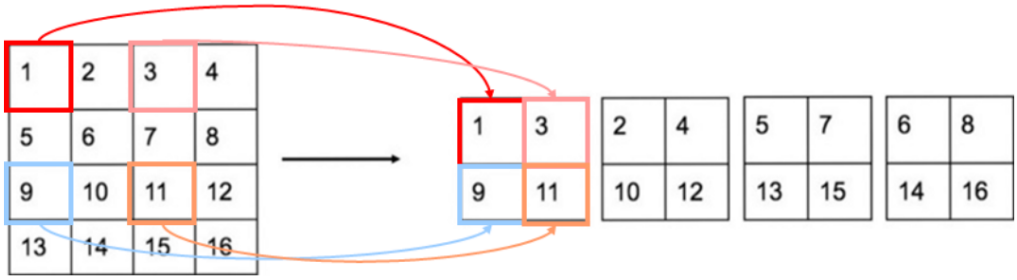
- 它为网络提供了时间来适应新的输入分辨率，然后才开始学习对象检测。
- 它可以帮助网络学习对物体检测很重要的特征，因为网络已经在大量图像数据集上进行了训练。

最终这个高分辨率分类网络使系统的 mAP 提高了近 4%。

3.1.2. Fine-Grained Features（细粒度特征）

我们知道，卷积层会逐渐减小空间维度。随着相应分辨率的降低，也就更难检测到小物体。YOLOv2 图片输入大小为 416 x 416，经过 5 次 maxpooling 后得到 13x13 的特征图，并以此特征图采用卷积做预测，如此已经可以达到检测大物体的作用。但如果需要检测小物体，则还需要更精细的特征图，因此 YOLOv2 提出了一种 Pass through Layer（直通层）以利用更精细的特征图。

它实现了不同层之间的特征融合，把高分辨率的浅层特征连接到低分辨率的深层特征（把特征堆积在不同 Channel 中）而后进行融合和检测。直通层与 ResNet 网络的 shortcut 类似，以前面更高分辨率的特征图为输入，然后将其连接到后面的低分辨率特征图上。



如上图所示，前面的特征图维度是后面的特征图的 2 倍，直通层抽取前面层的每个 2×2 的局部区域，然后将其转化为 channel 维度。对于 26×26×512 的特征图，经直通层处理之后就变成了 13×13×2048 的新特征图（特征图大小降低 4 倍，而 channles 增加 4 倍），这样就可以与后面的 13×13×1024 特征图连接在一起形成 13×13×3072 的特征图，然后在此特征图基础上卷积做预测（而在 YOLOv1 中网络的 FC 层起到了全局特征融合的作用）。

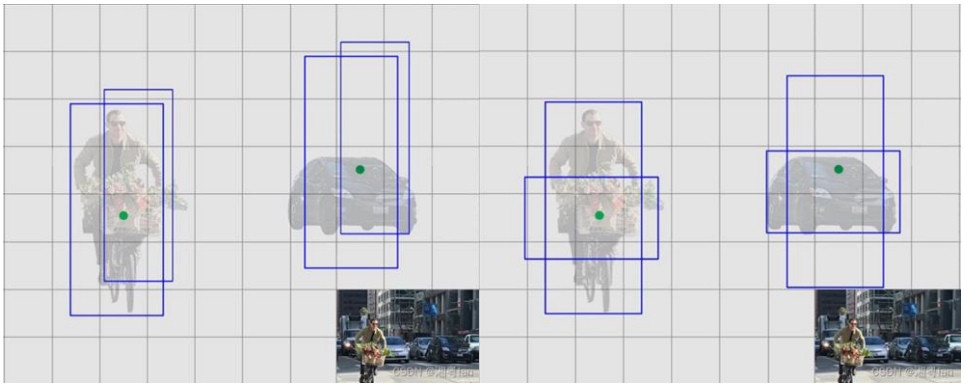
3.1.1. Multi-Scale-Training（多尺度训练）

由于模型仅使用卷积层和池化层，因此可以随时调整输入的大小。为了使 YOLOv2 在不同大小的图像上运行时具有鲁棒性，作者针对不同的输入大小训练了模型，就是在训练的过程中每间隔一定的 iterations 后改变输入的图片大小。

YOLOv2 下采样步长为 32，因此输入图片大小要选择 32 的倍数（最小尺寸为 320x320，最大尺寸为 608x608）。在训练过程，每隔 10 个 iterations 随机选择一种输入图片大小，然后只需要修改对最后检测层的处理就可以重新训练。这起到了数据增强的作用，并迫使网络对不同的输入图像尺寸和比例进行良好的预测

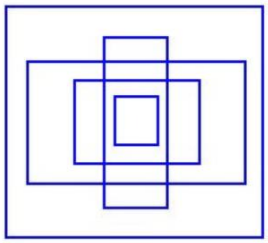
3.2. 更多的边界框

YOLOv1 在训练过程中学习适应不同物体的形状比较困难，导致其在精度定位方面表现不佳。其早期训练容易受到不稳定梯度的影响。首先，YOLO 对边界框进行随机预测。这些随机的预测可能对部分目标的检测有效果，但对另一些目标识别却很糟糕，从而导致陡然的梯度变化。在早期的训练中，预测会在专门研究什么形状上去探索，尝试，甚至竞争。



例如上图，早期训练中，对人和车预测边界框形状都是竖着的（左侧图像）。对人也许应该这样预测，因为行人的横纵比为 0.41，但汽车显然需要更扁的边界框来预测（右侧图像）。

所以，作者的想法是创建 5 个具有下面图片中形状的 anchors。我们将在 3.3 章节详细分析。



3.3. Anchor Box （锚框）机制

在 YOLOv1 中，作者设计了端对端的网路，直接对边界框的位置（x, y, w, h）进行预测。这样做虽然简单，但是由于没有类似 R-CNN 系列的推荐区域，所以网络在前期训练时非常困难，很难收敛。于是，自 YOLOv2 开始，引入了 Anchors box （锚框）机制，希望通过提前筛选得到的具有代表性先验框 Anchors，使得网络在训练时更容易收敛。

3.3.1. 什么是锚框

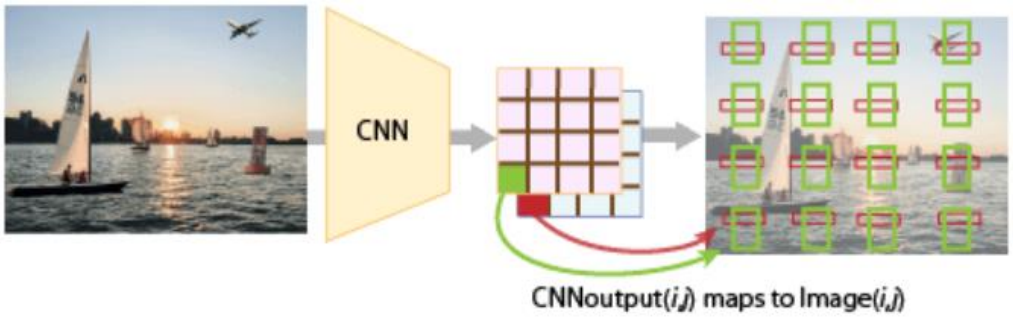
锚框是一组具有特定高度和宽度的预定义边界框。这些框用于捕获要检测的特定对象类的比例和纵横比，通常根据训练数据集中的对象大小进行选择。在检测过程中，预定义锚框会平铺在图像上。网络会预测每个平铺锚框的概率和其他属性，例如背景、IoU 和偏移量。预测结果用于优化每个单独的锚框。我们可以定义多个锚框，每个锚框对应不同的对象大小。锚框是固定的初始边界框猜测。

网络并不直接预测边界框，而是预测与平铺锚框相对应的概率和细化。网络为每个定义的锚框返回一组唯一的预测。最终的特征图表示每个类的对象检测。使用锚框使网络能够检测多个对象、不同尺度的对象和重叠的对象。

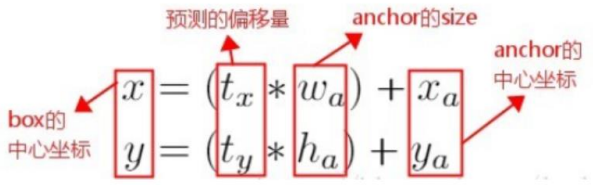
锚框在系统中作为先验，系统将从这里进行调整。

3.3.2. 锚框工作原理

锚框的位置是通过将网络输出的位置映射回输入图像来确定的。该过程对每个网络输出都重复一次。结果会在整个图像上产生一组平铺的锚框。每个锚框代表一个类的特定预测。例如，下图中有两个锚框，每个位置有两个预测。



在 Faster R-CNN 算法中，是通过预测锚框与 ground truth 的位置偏移值t_x, t_y ，从而优化锚框的位置和大小，间接得到边界框的位置（优化后的锚框即为边界框）。其公式如下：



这个公式是无约束的，预测的边界框很容易向任何方向偏移。因此，每个位置预测的边界框可以落在图片任何位置，这会导致模型的不稳定性。

因此 YOLOv2 在此方法上进行了一点改变：预测边界框中心点相对于该网格左上角坐标(C_x, C_y)的相对偏移量。同时为了将边界框的中心点约束在当前网格中，使用 sigmoid 函数将 t_x, t_y归一化处理，将值约束在（0，1），这使得模型训练更稳定，我们将在 3.3 章节详细分析这一部分。

3.4. Dimension Clusters（维度聚类）

Faster R-CNN 中锚框的大小和比例是按经验设定的，不具有很好的代表性。若一开始就选择了更好的、更有代表性的先验框锚框，那么网络就更容易学到准确的预测位置了。例如，在自动驾驶中，最常见的 2 个边界框则是不同距离的汽车和行人。

YOLOv2 在训练集的边界框上运行 K-means 聚类训练边界框，可以自动找到更好的 boxes 宽高维度。聚类目的为提高选取的锚框和同一个聚类下的 ground truth 之间的 IoU，采用的距离公式如下:

$d(box,centroid)=1-IoU(box,centroid)$

- *centroid* 为聚类时被选作中心的 *bounding box*。
- *box* 为其他的 *bounding box*。

具体实现方式如下：

① 使用的聚类原始数据是只有标注框的检测数据集，YOLOv2、v3都会生成一个包含标注框位置和类别的TXT文件，其中每行都包含$(x_j, y_j, w_j, h_j), j \in \{1, 2, ..., N\}$，即ground truth boxes相对于原图的坐标，(x_j, y_j)是框的中心点，(w_j, h_j)是框的宽和高，N是所有标注框的个数；

②首先给定k个聚类中心点$(W_i, H_i), i \in \{1, 2, ..., k\}$，这里的$W_i, H_i$是anchor boxes的宽和高尺寸，由于anchor boxes位置不固定，所以没有(x,y)的坐标，只有宽和高；

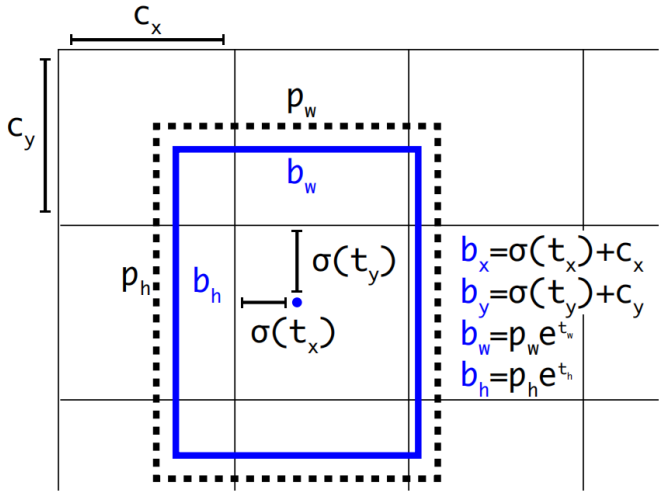
③计算每个标注框和每个聚类中心点的距离 d=1-IOU(标注框,聚类中心)，计算时每个标注框的中心点都与聚类中心重合，这样才能计算IOU值，即$d = 1 - IOU\left[(x_j, y_j, w_j, h_j), (x_j, y_j, W_i, H_i)\right], j \in \{1, 2, ..., N\}, i \in \{1, 2, ..., k\}$。将标注框分配给"距离"最近的聚类中心；

④所有标注框分配完毕以后，对每个簇重新计算聚类中心点，计算方式为$W_i' = \frac{1}{N_i}\sum w_i, H_i' = \frac{1}{N_i}\sum h_i$，$N_i$是第i个簇的标注框个数，就是求该簇中所有标注框的宽和高的平均值。

经实验表明，YOLOv2 采用 5 种 Anchor 比 Faster R-CNN 采用 9 种锚框得到的平均 IOU 还略高，并且当 YOLOv2 采用 9 种时，平均 IOU 有显著提高。说明 K-means 方法的生成的锚框更具有代表性。为了权衡精确度和速度的开销，最终选择 K=5。

3.5. 直接位置预测

下图为锚框与边界框转换示意图，其中蓝色的是要预测的边界框，黑色虚线框是锚框。



YOLOv2 在最后一个卷积层输出 13×13 的特征图，意味着一张图片被分成了 13×13 个网格。每个网格有 5 个锚框来预测 5 个边界框。我们对锚点的 offset（偏移量）进行预测。然而，如果它不受约束，我们的预测将再次随机化。因此 YOLOv2 此次预测 5 个参数：t_x，t_y，t_w，t_h，t_o(类似 YOLOv1 的 confidence）。并应用 sigmoid 函数来限制其可能的偏移范围,即（0，1）之间。引入锚框机制后，通过直接预测得到的边界框的位置的计算公式为：

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$

置信度 t_o的计算公式为：$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$

其中t_x，t_y，t_w，t_h 是预测的边界框中心点坐标和宽高。

σ 表示 Sigmoid 函数。

C_x, C_y是当前网格左上角到图像左上角的距离(要先将网格大小归一化，即令 grid 的宽高都为 1)。实际上就是一个边界框的中心点的 x 坐标和 y 坐标。

p_w，p_h是锚框的宽和高。

在这套算法下，因为使用了限制让数值变得参数化，让网络更容易学习、更稳定。

3.6. 网络结构

YOLOv2 采用 Darknet-19 作为特征提取网络，包括 19 个卷积层，10 个 maxpooling 层（5*2），其网络结构如下：

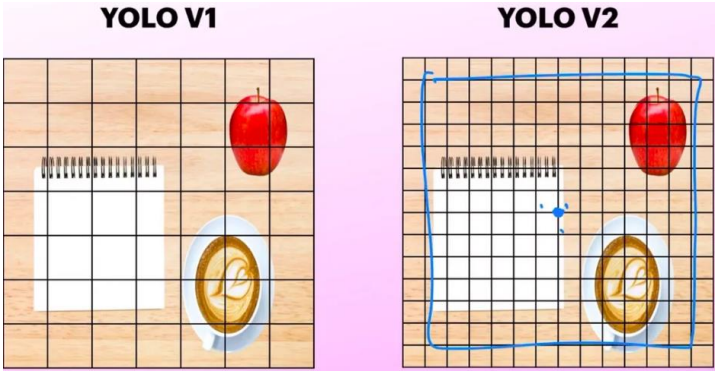
Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28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

- 与 VGG 相似，使用了很多 3×3 卷积核；并且每一次池化后，下一层的卷积核的通道数 = 池化输出的通道 × 2。
- 在每一层卷积后，都增加了 Batch Normalization（批量标准化）进行预处理。
- 采用了降维的思想，把 1×1 的卷积置于 3×3 之间，用来压缩特征。
- 在网络最后的输出增加了一个 global average pooling 层。
- 整体上采用了 19 个卷积层，11 个池化层（5*2+1）。

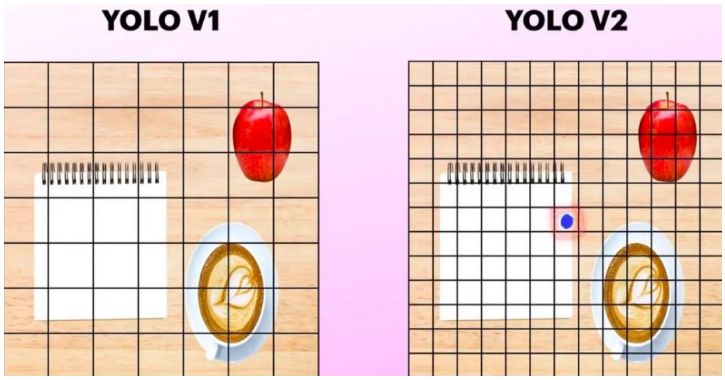
相比于 YOLOv1，YOLOv2 删除了全连接层和最后一个 maxpool 层，并将特征图的分辨率从 7x7 翻倍到 14x14。这改善了对象定位，因为网络具有更多图像的空间信息。

3.7. YOLOv2 的输出

上一节中提到的 14x14 是将特征图分为 14x14 个网格。但是现在 14x14 网格单元存在问题，如果网格单元数量为偶数，则没有单个中心位置。想象一下，我们正在尝试在这些网格中找到一个大物体的中心，但它可能无法与任何一个单元完美对齐。如下图，图像中心位置为网格交界点。



为了解决这个问题，最好使用奇数个网格单元。因此，建议使用 13x13 网格，而不是 14x14 网格。此更改可确保始终有一个中心网格单元，从而更轻松、更准确地检测图像中不同大小的物体。

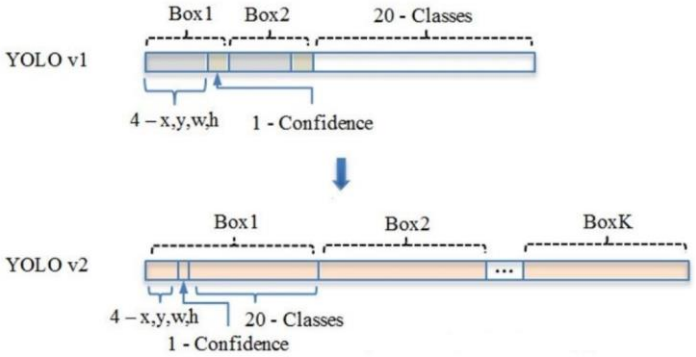


YOLOv1 在特征图(7x7)的每一个网格中预测出 2 个边界框及分类概率值，即每个边界框预测出 5 个值（4 个位置值+1 个置信度）。

总输出：7 x 7 x (5 x 2 + 20)

YOLOv2 在特征图(13x13) 的每一个网格中预测出 5 个边界框(对应 5 个 Anchor Box)，每个边界框预测出 5 个值（4 个位置值+1 个置信度）及分类概率值（置信度）。

总输出：13 x 13 x 5 x (5 + 20)



3.8. Batch Normalization（批量标准化）

批量标准化简称 BN，2015 年由 Google 研究员在论文《Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift》中提出。

BN 依靠两次连续的线性变换，希望转化后的数值满足一定的特性（分布），不仅可以加快了模型的收敛速度，也一定程度缓解了特征分布较散的问题。对数据进行预处理（统一格式、均衡化、去噪等）能够大大提高训练速度，提升训练效果。基于此，YOLOv2 对每一层输入的数据都进行批量标准化，这样网络就不需要每层都去学数据的分布，收敛会变得更快。

3.8.1. 相关原理

随着网络的深度增加，每层特征值分布会逐渐的向激活函数的输出区间的上下两端（激活函数饱和区间）靠近，长此以往则会导致梯度消失，从而无法继续训练模型。BN 就是通过将该层特征值分布重新拉回标准正态分布，特征值将落在激活函数对于输入较为敏感的区间，输入的小变化可导致损失函数较大的变化，使得梯度变大，避免梯度消失，同时也可加快收敛。其公式如下：

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

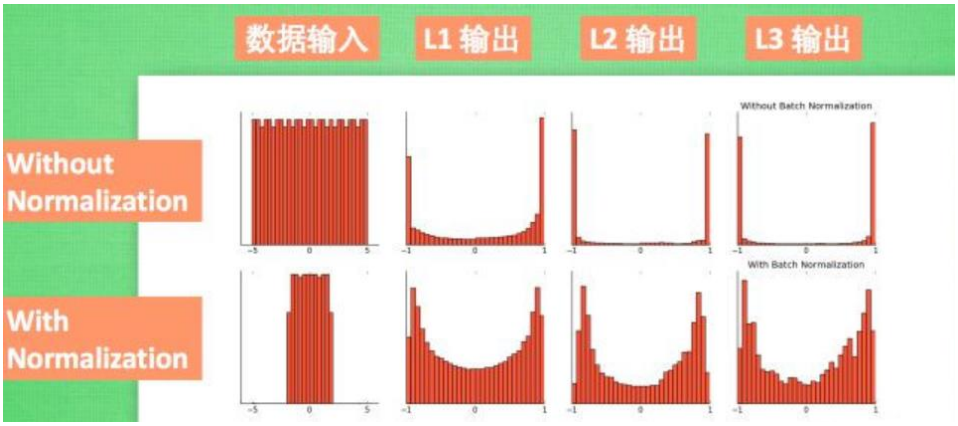
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

其中γ允许调整标准差，β允许调整偏差，将曲线向右或向左移动。运算过程为：

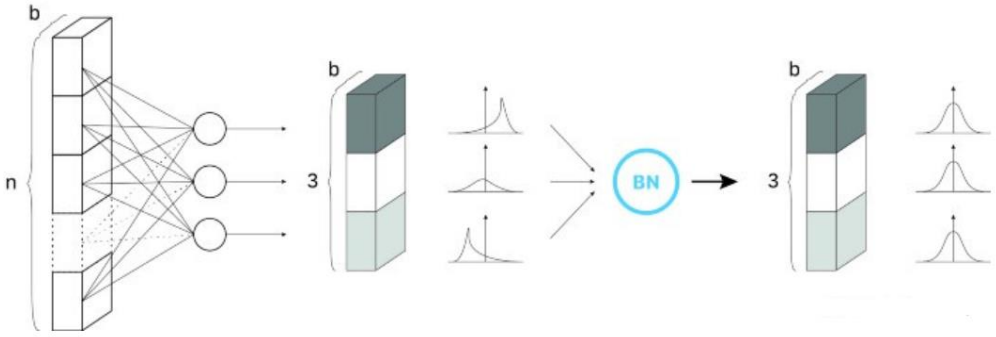
- 计算批处理数据均值（mini-batch mean）
- 计算批处理数据方差(mini-batch variance)
- 规范化(normalize)：$N \sim (0, 1)$正态分布。其中 ε 是微小正数，为了避免除数为 0。
- 尺度变换和转移(scale and shift)： 将x_i乘以 γ 调整数值大小，再加上 β 增加偏移后得到y_i，这里的 γ 是尺度因子， β 是平移因子。这一步是 BN 的精髓，由于归一化后的x_i基本会被限制在正态分布下，使得网络的表达能力下降。为解决该问题，我们引入了两个新的参数：γ 和 β。γ 和 β 是在训练时网络自己学习得到的。

3.8.2. 本质与效果

激活函数本质上想要放大差别，而随着网络深度的增加，就好像我们小时候玩的听筒传话游戏，每个人将听到的信息轮流传下去，信息便会在传递的过程中变得离谱起来。



如果数据在梯度很小的区域，那么学习率就会很慢甚至陷入长时间的停滞。减均值除方差后，数据就被移到中心区域如下图所示，对于大多数激活函数而言，这个区域的梯度都是最大的或者是有梯度的（比如 ReLU），这可以看做是一种对抗梯度消失的有效手段。对于一层如此，如果对于每一层数据都那么做的话，数据的分布总是在随着变化敏感的区域，相当于不用考虑数据分布变化了，这样训练起来更有效率。



训练的本质就是要根据 loss 去调整模型的参数，从而使得模型找到一个特定的 function 去拟合我们的数据。而 BN 的把数据从梯度较小的区域移到了梯度较大区域，如此一来，向前传播时的 loss 梯度大，模型收敛速度就快了，这也就是 sigmoid 激活函数被 ReLU 激活函数普遍替代的原因。

通过实验，BN 的优势为：

- 添加 BN 层会导致更快更好的收敛（更好意味着更高的准确性）。在大的数据集(ImageNet)上，BN 带来的提升提升比在小型 MNIST 数据集上观察到的要显着得多。
- 添加 BN 层允许我们在不影响收敛性的情况下使用更高的学习率，也即更快收敛。
- 此外，更高的学习率有助于 optimizer（优化器）避免 local-minimum（局部最小值）收敛。鼓励探索，那么 optimizer 也就将更容易收敛到更好的模型参数和解决方案。

同样，BN 有副作用。BN 依靠均值和方差来归一化隐藏层。那么，输出值与当前 batch 数据也就紧密相关。并且这种转换会增加一些噪音，具体取决于当前 batch 中使用的输入。

在实践中，我们不应该依赖 Batch Normalization 来避免 over-fitting，因为正交性很重要。简而言之，我们应该始终确保一个模块解决一个问题。如果依靠几个模块来处理不同的问题，便会使得开发过程困难许多。

3.9. YOLO9000

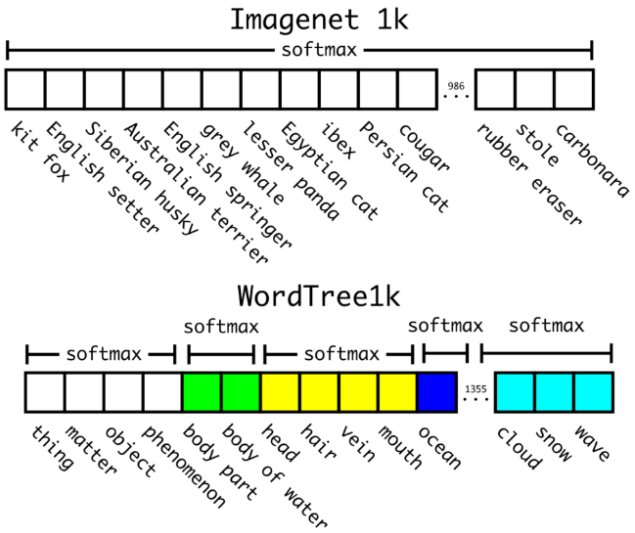
YOLO9000 是在 YOLOv2 的基础上提出的一种可以检测超过 9000 个类别的模型，锚框由原来的 5 调整到 3，对每个锚框预测其对应的边界框的位置信息 x,y,w,h 和置信度，以及所包含的物体分别属 9418 类的概率，所以每个锚框需要预测 4+1+9418=9423 个值。每个网格需要预测 3×9423=28269 个值。

YOLO9000 的主要贡献点在于提出了一种分类和检测的联合训练策略，它联合 coco 目标检测数据集和 imagenet 分类数据集。由于 coco 的数据集标签分类的比较粗，比如狗，猫，而 imagenet 分类则比较细化，比如二哈狗，金毛狗。这时候如果用 softmax 进行最后的分类，则会产生问题，因为 softmax 输出最大概率的那个分类，各种分类之间彼此互斥，若狗，二哈狗，金毛狗在一起的话就会出问题，所以要联合训练，必须让标签有一定程度上的一致性。

- 输入的若为目标检测标签的，用完整的 YOLO v2 loss 进行反向传播计算，学习预测物体的边界框 (bounding box)、置信度 (confidence) 以及物体（class）分类。
- 输入的若为分类标签的，只用分类部分的 loss 进行反向传播，仅学习（class）分类部分。

3.10. Hierarchical classification （层级分类方法）- Word Tree

联合数据集训练遇到的问题：类别之间不一定是互斥关系，例如人与女生。所以作者提出了一种层级分类方法 (Hierarchical classification)，根据类别之间的从属关系建立一种树结构 word Tree。Word Tree 的根节点为 physical object，每个节点的子节点都是属于同一个子类，在进行 softmax 时，不对所有类别进行，而是对同一层级的类别进行操作。如下图，对同一个颜色 (同一层级) 的类别进行 softmax。



在进行预测时，会从根结点向下遍历，每一层级选取机率最高的子节点，并计算该节点至根节点的所有条件机率之积。当该条件机率之积小于某个阈值时停止，以当前节点表示预测的类别。

4. YOLOv3：大佬收官之作，性能与精度的进一步提升

4.1. 引言&背景

YOLOv3（《Yolov3:An incremental improvement》）是 Joseph Redmon 大佬关于 YOLO 系列的最后一篇，由于他反对将 YOLO 用于军事和隐私窥探，2020 年 2 月宣布停止更新 YOLO

继 YOLOv2 之后，YOLOv3 的提出再次刷新了实时对象检测技术的标准。在保持 YOLO 系列快速检测的同时，YOLOv3 在多个方面进行了改进，以提高检测的准确性和效率。

YOLOv3 在 COCO 数据集上的表现与 SSD 变体相当，但在速度上快了三倍。在 AP50 指标上，YOLOv3 与 RetinaNet 相当，显示出其在对象检测上的强大能力。

4.2. 主要改进点

- **多尺度预测**：YOLOv3 在不同尺度上进行对象检测，类似于特征金字塔网络，增强了模型对不同尺寸对象的识别能力。
- **Darknet-53 网络**：YOLOv3 采用了新的网络结构 Darknet-53 作为其特征提取器，该网络结合了 Darknet-19 和残差网络的优点，具有更深的网络结构和更高效的计算效率。
- **锚框改进**：YOLOv3 继续使用锚框来预测边界框，但对预测机制进行了优化，提高了模型的稳定性和预测准确性。
- **类别预测**：采用独立的逻辑分类器进行多标签分类，避免了使用 softmax 函数，更好地处理了数据集中的重叠标签问题。
- **训练策略**：包括多尺度训练和大量数据增强，以及使用 Darknet 框架进行训练和测试。

4.3. 技术挑战与社会影响

论文中提到了一些未成功的尝试，例如焦点损失和锚框的 x，y 偏移预测，这些尝试为未来的研究提供了有价值的参考。

最后作者讨论了计算机视觉技术的潜在负面影响，并强调了研究者在推动技术发展的同时，应考虑其对社会的影响和责任，并思考减轻这些伤害的方法。

5. YOLOv3 核心技术点分析

5.1. 网络结构

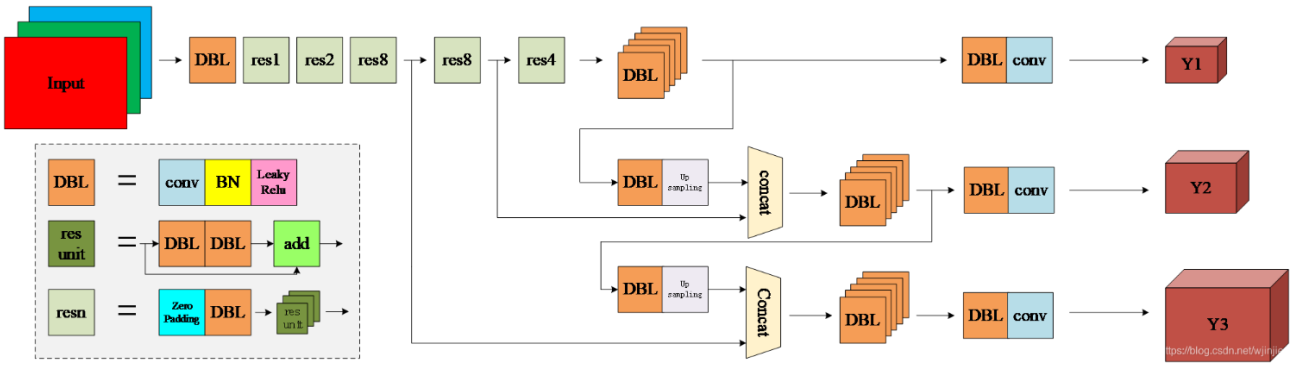
YOLOv2 采用 30 层架构，相比于 YOLOv2 的骨干网络，YOLOv3 进行了较大的改进。借助残差网络的思想，YOLOv3 将原来的 darknet-19 改进为 darknet-53。论文中给出的整体结构如下。

	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			

Darknet-53 主要 1×1 和 3×3 的卷积层组成，每个卷积层之后包含一个批量归一化层和一个 Leaky ReLU，加入这两个部分的目的是为了防止过拟合。卷积层、批量归一化层以及 Leaky ReLU 共同组成 Darknet-53 中的基本卷积单元 DBL。因为在 Darknet-53 中共包含 53 个这样的 DBL，所以称其为 Darknet-53。

跟 Darknet-19 相比，Darknet-53 去除了所有的 maxpooling 层、增加了更多的 1x1 和 3x3 的卷积层，但因为加深网路层数容易导致梯度消失或爆炸，所以 Darknet-53 加入了 ResNet **网路 (Residual Network)** 来解决梯度的问题。由上图的 Darknet-53 架构可以看到共加入了 23 个 residual block。

为了更加清晰地了解 darknet-53 的网络结构，可以看下面这张图：



以下是对结构图中主要单元的说明：

- **DBL：** 一个卷积层、一个批量归一化层和一个 Leaky ReLU 组成的基本卷积单元。
- **res unit：** 输入通过两个 DBL 后，再与原输入进行 add；这是一种常规的残差单元。残差单元的目的是为了让网络可以提取到更深层的特征，同时避免出现梯度消失或爆炸。
- **resn：** 其中的 n 表示 n 个 res unit；所以 resn = Zero Padding + DBL + n × res unit 。
- **concat：** 将 darknet-53 的中间层和后面的某一层的上采样进行张量拼接，达到多尺度特征融合的目的。这与残差层的 add 操作是不一样的，拼接会扩充张量的维度，而 add 直接相加不会导致张量维度的改变。
- **Y1、Y2、Y3：** 分别表示 YOLOv3 三种尺度的输出。

详细的计算方法，将在 5.4 章节中介绍。

5.2. Bounding Box Prediction（边界框预测）

与 YOLOv2 相同，网络为每个边界框预测 4 个坐标：t_x，t_y，t_w，t_h，并使用了同样的边界框预测公式（章节直接位置预测 3.5）。以下是 YOLOv3 相对于 v2 改进的的点：

在 YOLOv3 中，利用逻辑回归来预测每个边界框的客观性分数(object score)，也就是 YOLOv1 论文中说的 confidence:

- **正样本**：如果当前预测的包围框比之前其他的任何包围框更好的与 ground truth 对象重合，那它的置信度就是 1。

置信度意味着该预测框是或者不是一个真实物体，是一个二分类，所以标签是 1、0 更加合理。并且在学习小物体时，有很大程度会影响 IOU。如果像 YOLOv1 使用边界框与 ground truth 对象的 IOU 作为 confidence，那么 confidence score 始终很小，无法有效学习，导致检测的 Recall 不高。这就是为什么 YOLOv3 要将正样本 confidence score 设置为 1。

- **忽略样本**：如果当前预测的包围框不是最好的，但它和 ground truth 对象重合了一定的阈值（这里是 0.5）以上，神经网络会忽略这个预测。

由于 YOLOV3 采用了多尺度的特征图进行检测，而不同尺度的特征图之间会有重合检测的部分。例如检测一个物体时，在训练时它被分配到的检测框是第一个特征图的第三个边界框，IOU 为 0.98，此时恰好第二个特征图的第一个边界框与该 ground truth 对象的 IOU 为 0.95，也检测到了该 ground truth 对象，如果此时给其 confidence score 强行打 0，网络学习的效果会不理想。这就是为什么存在忽略样本。

- **负样本**:若 bounding box 没有与任一 ground truth 对象对应，那它的置信度就是 0

5.3. Class Prediction（类预测）

YOLOv3 将单标签分类改进为多标签分类。在一些数据集中，比如 Open Image Dataset，一个对象可能有多个标签。例如，一个对象可以被标记为一个女人和一个人。在这个数据集中，有很多重叠的标签。使用 softmax 进行类预测会假设每个框都只有一个类，但通常情况并非如此。

因此，YOLOv3 没有使用 softmax，而是对任何类使用独立的 **Logistic classifiers（逻辑分类器）**。使用独立的逻辑分类器，可以同时将对象检测为女性和人。

因此在训练期间，作者使用 binary cross-entropy（二元交叉熵 ）作为损失函数进行类预测，并且准确率不会下降，其公式如下：

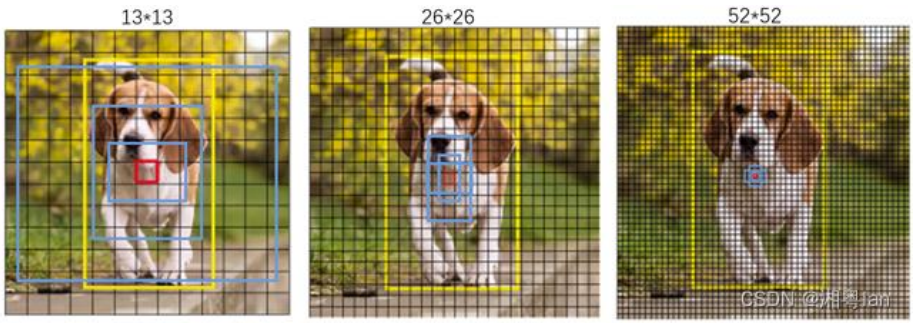
$$loss = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i),$$

$$y_i \; is \; true \; value, \hat{y}_i \; is \; predict \; value$$

5.4. Predictions Across Scales(跨尺度预测)

YOLOv3 借鉴 FPN（特征金字塔网）的方法，采用多尺度的特征图（这里是 3 个尺度）对不同大小的物体进行检测，以提升小物体的预测能力。

每个尺度的特征图会预测出 3 个锚框，而锚框的大小则采用 K-means 进行聚类分析，延续了 YOLOv2 的作法，具体可以参考 **3.4** 章节。YOLOv3 通过下采样 32 倍、16 倍、8 倍得到 3 个不同尺度的 feature map，例如输入 416x416 的图片，则会得到 13x13 (416/32)、26x26 (416/16)、52x52 (416/8)，这 3 个尺度的特征图。如下图所示：

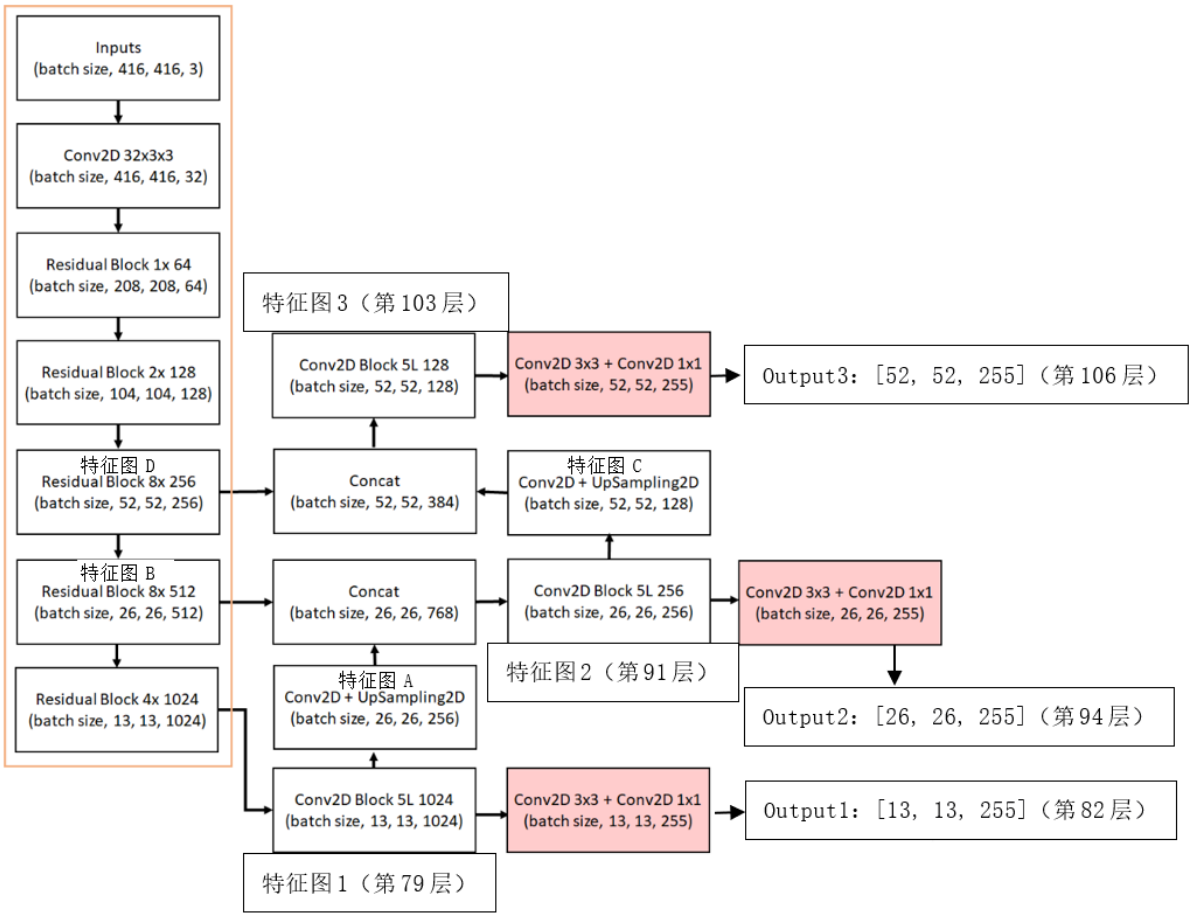


- 13x13 的特征图(有最大的感受野) 用于检测大物体，所以用较大的锚框(116x90), (156x198), (373x326)
- 26x26 的特征图(中等的感受野) 用于检测中等大小的物体，所以用中等的锚框(30x61), (62x45), (59x119)
- 52x52 的特征图(较小的感受野) 用于检测小物体，所以用较小的锚框(10x13), (16x30), (33x23)

可以在三个不同的尺度上进行检测是 YOLOv3 最显着的特点是。YOLOv2 是一个全卷积网络，其最终输出是通过在特征图上应用 1 x 1 kernel 生成的。在 YOLO v3 中，检测是通过在网络中三个不同位置的三个不同大小的特征图上应用 1 x 1 kernel 来完成的。

以 COCO 数据集为例，其共有 80 个类。所以网络输出的张量应该是：N ×N ×[3*(4 + 1 + 80)]。（分别是 N x N 个 gird cell ，3 种尺度的锚框，4 个边界框偏移值、1 个目标预测置信度以及 80 种类别的预测概率。）

该方法允许从上采样的特征中获取更有意义的语义信息，从早期的特征图中获取更细粒度的信息。如下图所示：



尺度 1：

- **特征图**：若输入 416x416 的图片，在 79 层卷积层后，图片已经经过 32 倍的**下采样**，我们得到了 13x13 的特征图 1（第 79 层）。
- **预测**：在特征图 1 后添加 2 个卷积层（3x3，1x1），即上图中第 3 个红色部分。最后输出一个 13 × 13 × [3 * (4 + 1 + 80)] 的张量表示预测（第 82 层）。最终输出为：[13, 13, 255]

尺度 2：

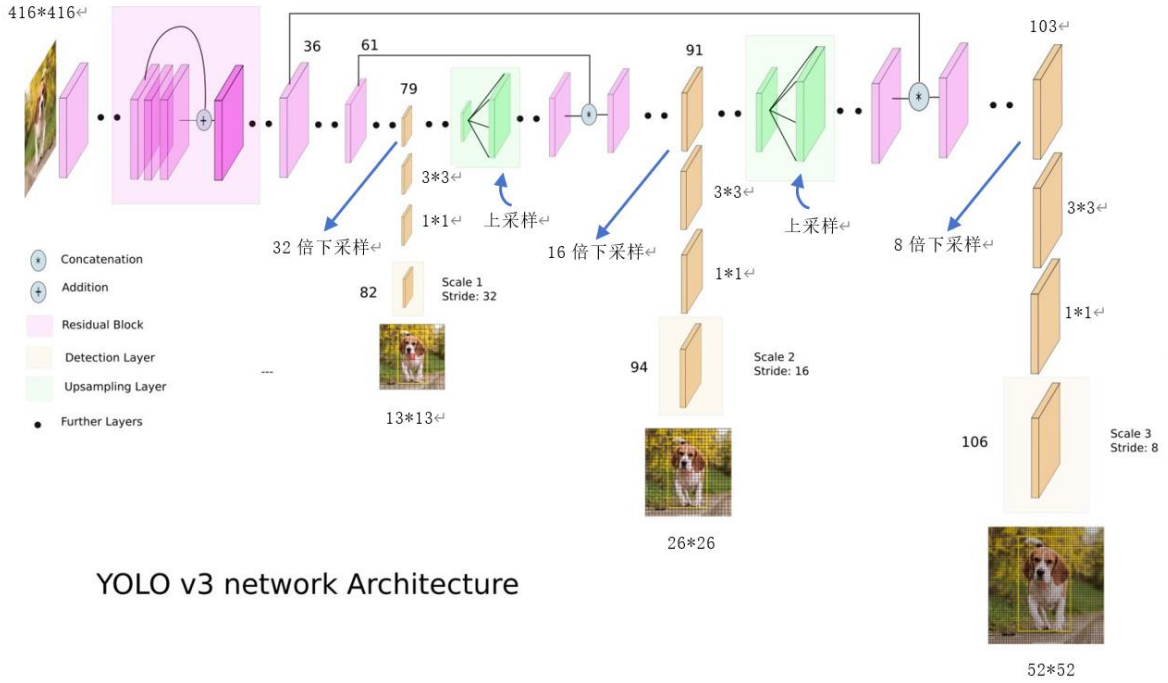
- **特征图**：将第 79 层 13x13 的特征图 1 进行**上采样**，得到 26x26 特征图 A。与第 61 层对原图 16 倍下采样得到的 26x26 的特征图 B 合并（Concat）后，得到 26x26 的特征图 2(第 91 层)。
- **预测**：在特征图 2 后添加 2 个卷积层（3x3，1x1），即上图中第 2 个红色部分。最后输出一个 26 × 26 × [3 * (4 + 1 + 80)] 的张量表示预测（第 94 层）。最终输出为：[26, 26, 255]

尺度 3：

- **特征图**：将第 91 层 26x26 的特征图 2 进行上采样，得到 52x52 特征图 C。与第 36 层对原图 8 倍下采样得到的 52x52 的特征图 D 合并（Concat）后，得到 52x52 的特征图 2(第 103 层)。

- **预测**：在特征图 2 后添加 2 个卷积层（3x3，1x1），即上图中第 1 个红色部分。最后输出一个 52 × 52 × [3 * (4 + 1 + 80)] 的张量表示预测（第 106 层）。最终输出为：[52, 52, 255]

具体的层数与上下采样的细节见下图：



5.5. 损失函数

YOLOv3 这篇文中并没有给出损失函数的表达式。通过对源码的分析，反推出其的损失函数表达式：

loss function =

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}(2-b_{wi}\times b_{hi})[(b_{xi}-\widehat{b_{xi}})^2+(b_{yi}-\widehat{b_{yi}})^2]$$ bndBox的中心點座標計算loss

$$+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}(2-b_{wi}\times b_{hi})[(b_{wi}-\widehat{b_{wi}})^2+(b_{hi}-\widehat{b_{hi}})^2]$$ bndBox的寬高計算loss

$$-\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}c_{ij}\log(\widehat{c_{ij}})+(1-c_{ij})\log(1-\widehat{c_{ij}})$$ bndBox 有物件的confidence計算loss

$$-\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}c_{ij}\log(\widehat{c_{ij}})+(1-c_{ij})\log(1-\widehat{c_{ij}})$$ bndBox 無物件的confidence計算loss

$$-\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\sum_{ceclasses}[p_{ij}(c)\log(\widehat{p_{ij}}(c))+(1-p_{ij}(c))\log(1-\widehat{p_{ij}}(c))]$$ bndBox物件類別計算loss

对比 YOLOv1 中的损失函数：位置损失部分并没有改变，仍然采用的是 sum-square error 的损失计算方法。但是置信度损失和类别预测均由 sum-square error 改为了交叉熵的损失计算方法。

1_{ij}^{noobj}为第为第 i 个网络中第 j 个边界框是否为负例，若是则输出 1，否则为 0。

如果是忽略样例（既不是正例也不是负例），则都输出 0（即没有任何 loss）。

有物体的$\lambda_{nocoord}$为 1，没有物体的为 0.5，使得模型专注于物体的识别，并降低找不到物体的情况。

5.6. What This All Means

在 YOLOv3 论文的结尾部分，作者 Joseph Redmon 和 Ali Farhadi 提出了对当前评估指标的深刻反思，并探讨了计算机视觉技术的社会责任。

他们质疑了为何要改变评估指标，指出最初的 COCO 论文并未充分论证 IOU 阈值的选择，同时指出人类难以区分低 IOU 值的差异，暗示了对高精度评估指标的过分追求可能并不必要。

此外，作者提出了一个关键问题：在拥有先进检测技术之后，我们应该如何使用它们？他们表达了对技术可能被滥用于收集个人信息和军事目的的担忧，并强调了作为研究者，我们有责任考虑我们的工作可能带来的负面影响，并寻求减轻这些影响的方法。尽管存在潜在的负面用途，作者仍对计算机视觉技术在积极领域，如野生动物保护和家庭娱乐中的应用抱有希望。最后，作者以幽默的方式结束讨论，同时呼吁业界人士对技术的使用持有批判性思维，确保技术进步与社会责任并行。

讽刺的是，大佬本人的退出并无法阻止 YOLO 系列的研发更新。而且可以确认的是，作者最担心的事情已经发生，YOLO 已经在 CN 大量部署用于天网的识别，个人信息的收集已经被滥用。AI 发展到今日，道德问题越来越明显。

6. YOLOv1、V2、v3 对比

	YOLOv1	YOLOv2	YOLOv3
输入图像尺寸	输入的是 448×448 的三通道图像	输入的是 416×416 的三通道图像	输入的是 416×416 的三通道图像
grid cell	每一张图像划分为 7×7=49 个 grid cell	每一张图像划分为 13×13=169 个 grid cell	yolov3 会产生三个尺度：13×13、26×26、52×52，也对应着 grid cell 个数。
bbox/anchor	每个 grid cell 生成 2 个 bbox（没有 anchor），与真实框 IOU 最大的那个框负责拟合真实框	每个 grid cell 生成 5 个 anchor 框，通过 IOU 计算选一个 anchor 产生预测框去拟合真实框	每个 grid cell 生成 3 个 anchor 框，通过与 gt 的 IOU 计算选一个 anchor 产生预测框去拟合真实框
输出张量	输出 7 * 7 * 30 维的张量	输出 13 * 13 * 125 维张量	输出三个不同尺寸的张量，但最后都是 255，比如 S * S * 255。
预测框数量	7 * 7 *2 = 98 个预测框	13 * 13 * 5 = 845 个预测框	(52 * 52 + 26 * 26 +13 * 13) * 3 = 10647 个预测框

7. 笔记中涉及的相关技术点

7.1. SSD（Single Shot MultiBox Detector）

7.1.1. 核心概念与特点

SSD（Single Shot MultiBox Detector）是一种流行且高效的实时目标检测模型，由刘卫华等人在 2016 年提出。这种模型特别设计用于在单次前向传播中实现目标检测，以此提高处理速度，非常适用于实时处理任务。

- **单次检测（Single Shot）：**与需要多次处理的区域建议网络（如 R-CNN 系列）不同，SSD 在单次前向传播中同时预测目标类别和边界框位置。这种方法消除了候选区域提取和后续重复处理的需要，显著提高了检测速度。
- **多尺度特征图：**SSD 利用基础网络（常用 VGG16 或 ResNet 等）不同层的特征图来检测不同大小的对象。较浅的层识别小物体，较深的层识别大物体，这样的设计使得 SSD 能够有效处理多种尺寸的目标。
- **固定形状的边界框（Anchors）：**类似于 Faster R-CNN 中的 anchor boxes，SSD 在特征图的每个位置上定义了一系列固定形状和大小的边界框，这些边界框用于预测相对位置的偏移以及类别概率。
- **端到端训练：**SSD 可以通过端到端的方式进行训练，不需要复杂的训练阶段或额外的候选区域生成步骤。

7.1.2. 优势与应用

- **高效性：**由于其单次检测架构，SSD 能够在不牺牲准确性的情况下，提供极高的处理速度，适合实时应用。
- **准确性：**通过使用多尺度特征图，SSD 能够在广泛的物体大小上实现较高的检测准确性。
- **灵活性：**SSD 能够适应不同的基础网络架构，这使得它可以根据具体的性能和准确性需求进行定制。

SSD 由于其出色的速度和准确性，广泛应用于许多实时视觉识别任务中。包括：视频监控、自动驾驶、机器人视觉、AR。

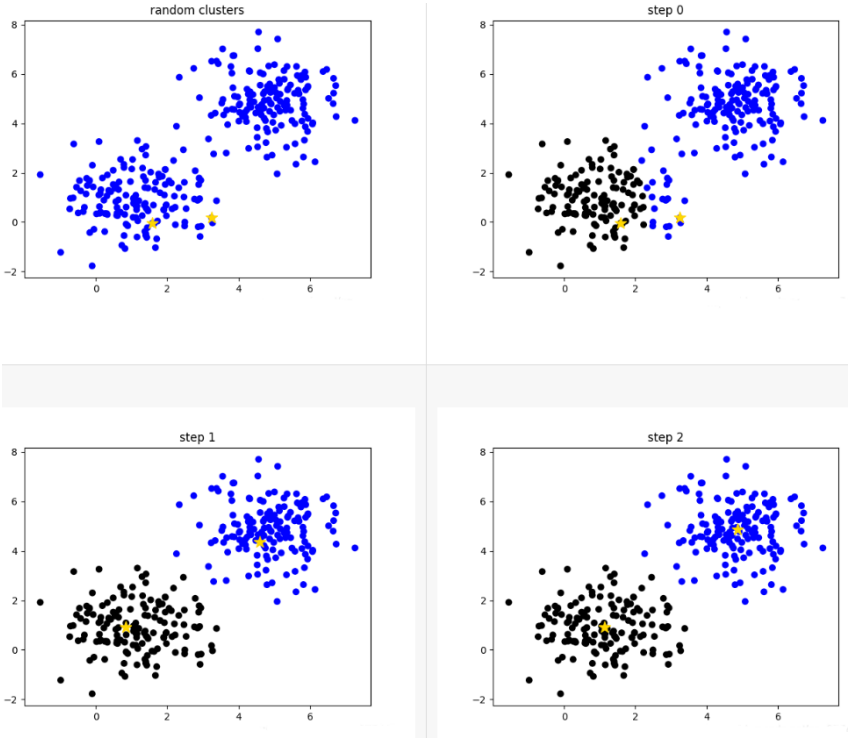
7.2. K-means 聚类

k-means 是非常经典且有效的聚类方法，通过计算样本之间的距离（相似程度）将较近的样本聚为同一类别（簇）。使用 k-means 时主要关注两个问题：

- 如何表示样本与样本之间的距离（核心问题）？这个一般需要根据具体场景去设计，不同的方法聚类效果也不同，最常见的就是欧式距离。
- 分为几类？这个也是需要根据应用场景取选择的，也是一个超参数。

k-means 算法主要流程如下：

- 手动设定簇的个数 k，假设 k=2。
- 在所有样本中随机选取 k 个样本作为簇的初始中心，如下图（random clusters）中两个黄色的小星星代表随机初始化的两个簇中心。
- 计算每个样本离每个簇中心的距离（这里以欧式距离为例），然后将样本划分到离它最近的簇中。如下图（step 0）用不同的颜色区分不同的簇。
- 更新簇的中心，计算每个簇中所有样本的均值（方法不唯一）作为新的簇中心。如下图（step 1）所示，两个黄色的小星星已经移动到对应簇的中心。
- 重复第 3 步到第 4 步直到簇中心不在变化或者簇中心变化很小满足给定终止条件。如下图（step2）所示，最终聚类结果。



在图像识别领域，有时会使用聚类生成先验锚框，但有时会反向升级，我们应该注意以下两方面：

在进行边界框聚类分析时，必须采取相同的缩放策略。这是因为，如果不对边界框进行适当的缩放，那么聚类得到的锚框可能无法准确反映训练时的实际需求。举例来说，如果原始图像尺寸为 1280x1280 像素，且边界框尺寸为 100x100 像素，而没有进行适当的缩放，那么聚类得到的锚框可能会集中在 100x100 像素附近。然而，在实际训练过程中，这些边界框已经被缩放到 50x50 像素。因此，为了确保锚框的有效性，理想的锚框尺寸应当与训练时使用的边界框尺寸相匹配，即 50x50 像素。确保图像和边界框的一致性缩放对于模型学习正确的空间尺度至关重要，这有助于提高检测算法的准确性和鲁棒性。

当利用预训练权重进行模型微调时，应当谨慎考虑权重冻结的策略。在针对特定数据集进行训练时，常见的做法是采用在大规模数据集 COCO 上预训练得到的权重。这些权重是基于相应数据集的特征和边界框分布聚类得出的，未必直接适用于新数据集的特定需求。由于网络需要适应新的锚框，它必须调整一定数量的权重。如果冻结了过多的层（例如，仅对最终的预测层进行微调，而保持其他所有层的

权重不变），可能会导致模型无法充分利用新数据集的特性，从而影响检测性能。在多数情况下，适度解冻更多的层以允许网络学习，将有助于更好地适应新的锚框，从而提高模型的整体性能。随着可训练权重数量的增加，使用针对自己数据集聚类得到的锚框通常会带来更佳的结果，前提是这些锚框是通过合理的方法得出的。这种方法使得模型能够更精确地学习数据集中对象的尺寸和形状，进而提升检测精度。

7.3. 浅层特征与深层特征

在卷积神经网络（CNN）中，浅层特征（shallow features）和深层特征（deep features）指的是网络在不同层次上提取的特征表示，它们具有不同的特点和用途：

7.3.1. 浅层特征（Shallow Features）

- **定义**：浅层特征通常指的是网络前几层提取的特征，这些层接近输入数据。
- **属性**：这些特征倾向于捕捉局部的、低级的图像属性，如边缘、纹理和颜色等。
- **空间信息**：浅层特征通常保留更多的空间信息，因为它们在网络中经过的层数较少，下采样（downsampling）的程度较低。
- **作用**：浅层特征对于图像的初步理解非常重要，它们可以用于简单的图像分类、去噪或特征增强等任务。

7.3.2. 深层特征（Deep Features）

- **定义**：深层特征是指网络较深层次提取的特征，这些层远离输入数据。
- **属性**：深层特征能够捕捉更抽象、更高级的图像属性，如物体的形状、部件和语义信息等。
- **空间信息**：随着网络层次的加深，深层特征的空间分辨率逐渐降低，因为多次下采样导致空间尺寸减小。
- **作用**：深层特征对于复杂的图像理解和高级视觉任务至关重要，它们可以用于精确的对象识别、场景解析和图像检索等任务。

7.3.3. 区别和联系

- **层次**：浅层特征来自网络的较浅层次，而深层特征来自较深层次。
- **抽象程度**：浅层特征较为具体，反映图像的基本结构；深层特征较为抽象，反映图像的高级语义。
- **空间分辨率**：浅层特征具有较高的空间分辨率，深层特征的空间分辨率较低。
- **应用**：浅层特征适用于需要保留图像细节的任务，而深层特征适用于需要理解图像内容的任务。

在设计卷积神经网络时，合理利用浅层和深层特征可以提高模型的性能和泛化能力。例如，在特征金字塔网络（FPN）中，结合不同层次的特征可以同时捕获图像的局部细节和全局上下文信息。

7.4. VGG16

VGG16 是一种深度卷积神经网络（CNN）架构，由牛津大学的视觉几何组（Visual Geometry Group）在 2014 年提出，主要作者为 K. Simonyan 和 A. Zisserman。VGG16 是 VGG 网络的两个版本之一，另一个是 VGG19，两者的主要区别在于网络的深度——VGG16 有 16 个卷积层，而 VGG19 有 19 个卷积层。

以下是 VGG16 的一些关键特点：

- **深度架构**：VGG16 包含 13 个卷积层和 3 个全连接层，其深度是当时大多数 CNN 架构的几倍。
- **3x3 卷积核**：VGG16 在所有卷积层中统一使用了 3x3 的卷积核，这简化了模型的复杂性并提高了参数利用效率。
- **激活函数**：VGG16 使用 ReLU 作为非线性激活函数。
- **Dropout**：在全连接层之前，VGG16 使用了 dropout 技术来减少过拟合。
- **预训练权重**：VGG16 通常在大规模数据集（如 ImageNet）上进行预训练，然后可以用于迁移学习，对新数据集进行微调。
- **性能**：尽管 VGG16 在提出时在 ImageNet 竞赛中取得了优异的成绩，但随着更深层次网络（如 ResNet）的出现，其性能已经被超越。

VGG16 是一个具有里程碑意义的模型，它推动了深度学习和卷积神经网络在图像识别和分类任务中的应用。尽管存在一些局限性，如计算资源消耗大和训练时间长，VGG16 仍然在许多视觉任务中被广泛使用和研究。

7.5. ResNet

ResNet（残差网络）是一种深度卷积神经网络（CNN），由微软研究院的何恺明等人在 2015 年提出。它通过引入了所谓的“残差学习”（residual learning）来解决传统深度网络在训练过程中常见的梯度消失和梯度爆炸问题，特别是在网络非常深的情况下。这种结构在 2015 年的 ImageNet 竞赛中取得了突破性的成果，极大地推动了深度学习在多个领域的应用。

7.5.1. 残差学习

ResNet 的核心是其“残差块”（residual block），每个残差块都包括两个主要的部分：一组卷积层和一个“跳过连接”（skip connection），也称为“快捷连接”（shortcut connection）。跳过连接的作用是将块的输入直接添加到卷积层的输出上。这样设计的目的是让网络学习输入与输出之间的残差，即 F(x) = H(x) - x，其中 H(x)是块的输出，x 是输入，F(x)是卷积层需要学习的残差部分。这种结构使得网络可以非常深而不会导致梯度消失，因为梯度可以通过跳过连接直接流向更早的层。

通过残差块的使用，ResNet 可以构建非常深的网络，例如 ResNet-50, ResNet-101, ResNet-152 等，其中数字代表网络的层数。更深的网络能够学习更加复杂和抽象的特征，但通常也更难训练。ResNet 通过残差块简化了这一训练过程，实现了更好的训练效果。

7.5.2. 优势

- **解决梯度消失问题**：残差块使得即使在非常深的网络中，梯度也能有效地传播，从而避免了梯度消失。
- **加速收敛**：残差学习有助于加速深层网络的收敛，比传统的网络训练时间更短。
- **提高性能**：ResNet 能够在各种视觉任务中达到非常高的性能，包括图像分类、物体检测和图像分割。

7.6. Logistic classifiers（逻辑分类器）

逻辑分类器，也称为逻辑回归分类器，是一种广泛应用于二分类问题的统计模型，尽管它也可以扩展到多分类场景。逻辑分类器的核心是逻辑函数，也称为 Sigmoid 函数。其关键特点：

逻辑分类器使用逻辑函数将线性回归的输出映射到 0 和 1 之间，这使得它可以用于表示概率。对于二分类问题，这个输出可以被解释为属于某个类别的概率。

$$\text{公式：}\sigma(x) = \frac{1}{1+e^{-x}}$$

其中，z是输入值，通常是线性回归模型的输出，$\sigma(z)$是经过 Sigmoid 函数转换后的概率值。

逻辑分类器适用于各种分类问题，包括信用评分、疾病诊断、垃圾邮件检测等。

逻辑分类器是机器学习和数据挖掘中的基础工具，尽管在某些复杂任务中可能被更高级的模型所取代，但在许多实际应用中仍然表现出色。在深度学习中，逻辑分类器的概念也被用于构建更复杂的神经网络架构中的分类层。