YOLO 系列综述-Part 5：
番外 —— SSP 论文阅读笔记

论文名：

Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

目录

1. 简介 3
2. 论文概述 4
   2.1. 摘要(Abstract) 4
   2.2. 引言(Introduction) 4
   2.3. 网络结构（Network Structure） 4
   2.4. 训练网络（Training the Network) 4
   2.5. 实验（Experiments) 4
   2.6. 结论（Conclusion） 5
3. 研究背景与动机 5
   3.1. 输入尺寸限制 5
      3.1.1. 关于图像尺寸的理解 6
   3.2. 冗余计算 6
   3.3. 解决思路 6
4. 论文核心方法实现步骤 7
   4.1. pooling（池化层） 7
   4.2. 金字塔池化 7
   4.1. SPP-Net 工作流程 8
   4.2. 与 R-CNN 的对比 8
5. 论文涉及方法分析 9
   5.1. R-CNN 9
      5.1.1. Region Proposal（候选区域） 9
      5.1.2. Selective Search（选择性搜索算法） 10
         5.1.2.1. Selective Search 的分层分组算法流程 10

5.1.2.2. 相似度计算 11
5.1.2.3. 颜色相似度 11
5.1.2.4. 纹理相似度 11
5.1.2.5. 尺度相似度 11
5.1.2.6. 空间交叠相似度 12
5.1.3. 模型 pre-training 12
5.1.4. 模型微调 12
5.1.1. 基于 SVM 的训练 13
5.1.2. Why SVM? 13
5.1.3. Bounding Box Regression（边界框回归） 14
5.1.4. R-CNN 存在的问题 14
5.2. SVM（支持向量机） 15
5.2.1. SVM 的工作原理 15
5.2.2. 点到超平面的距离公式 16
5.2.3. 硬间隔、软间隔和非线性 SVM 16
5.2.4. 用 SVM 解决多分类问题 17
5.2.4.1. 一对多法 18
5.2.4.2. 一对一法 18

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。

作为系列的第五章，是番外章节，将作为本系列第三章关于 YOLOv4 主干网络中，SSP（空间金字塔池化）部分的补充。本文将详细分析 SSP 的结构与优势，并回顾 R-CNN 架构设计的相关研究。以帮助读者可以对 YOLOv4 以及其他目标检测网络有更全面的理解。

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，所有黑色加粗并配有下划线的关键词，均配有交叉引用。

曹倬瑄

2024/7/5 于惠州

2. 论文概述

这篇论文的标题是《Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition》，作者是 Kaiming He, Xiangyu Zhang, Shaoqing Ren 和 Jian Sun。这篇论文主要介绍了一种新的网络结构——SPP-net（Spatial Pyramid Pooling Network），它通过引入空间金字塔池化（Spatial Pyramid Pooling, SPP）层来解决传统卷积神经网络（CNN）在处理不同尺寸输入图像时的限制。

2.1. 摘要(Abstract)

论文指出，传统 CNN 需要固定尺寸的输入图像，限制了输入图像的尺寸和比例，可能会降低识别精度。

作者提出了 SPP-net，它通过空间金字塔池化层生成固定长度的特征表示，不受图像尺寸/比例的影响。实验表明，SPP-net 对物体变形鲁棒，能够提升基于 CNN 的图像分类方法，并在 ImageNet 2012 数据集上展示了多种 CNN 架构的准确性提升。

2.2. 引言(Introduction)

论文讨论了深度学习在视觉识别领域的快速发展，尤其是 CNN 和大规模训练数据的可用性。

并指出了现有 CNN 在输入尺寸固定性方面的技术问题，并提出了 SPP 作为一种解决方案。

2.3. 网络结构（Network Structure）

卷积层和特征图：讨论了卷积层如何产生特征图，这些特征图不仅包含响应的强度，还包含空间位置信息。

空间金字塔池化层：介绍了 SPP 层如何替代传统的池化层，以适应任意尺寸的输入图像。

2.4. 训练网络（Training the Network）

描述了如何使用标准反向传播算法训练网络，尽管输入图像的尺寸可以变化。

2.5. 实验（Experiments）

- 在 ImageNet 2012 数据集上，展示了 SPP-net 如何提升不同 CNN 架构的准确性。
- 讨论了多尺寸训练方法，以及如何通过在训练中使用不同尺寸的图像来提高网络的尺度不变性和减少过拟合。
- 将 SPP-net 应用于对象检测任务，并与 R-CNN 方法进行了比较，展示了 SPP-net 在速度和准确性上的优势。

2.6. 结论（Conclusion）

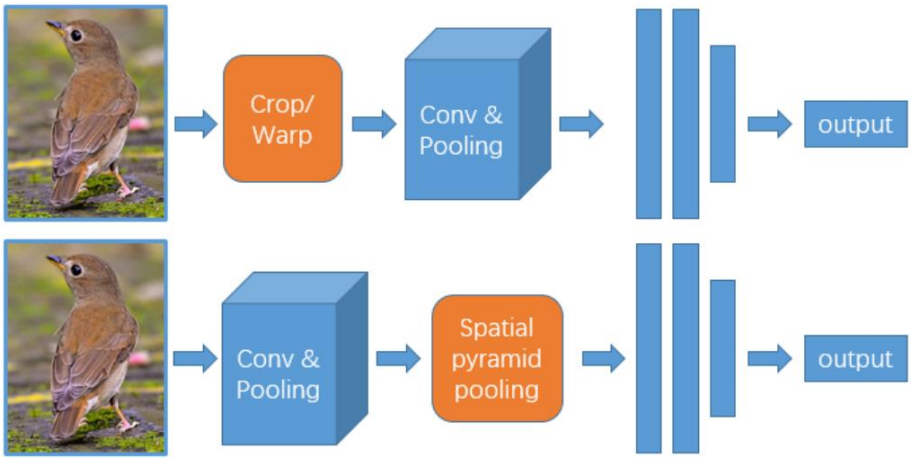
SPP 为处理不同尺度、尺寸和宽高比提供了灵活的解决方案，并且在视觉识别任务中显示出卓越的准确性和速度。

3. 研究背景与动机

近年来，深度卷积神经网络（CNN）在图像分类、对象检测以及其他视觉识别任务中取得了革命性的进展。这些进步在很大程度上得益于大型训练数据集的可用性以及深度学习模型的发展。

3.1. 输入尺寸限制

在 SSP 问世之前，市面上主流的目标检测模型是 R-CNN（**我们将在章节 5.1 详细介绍 R-CNN**），R-CNN 算法的核心思想就是对每个候选框通过 CNN 提取特征，然后接上一个分类器预测这个区域包含一个感兴趣对象的置信度，也就是说，转换成了一个图像分类问题，后面接的这个分类器可以是独立训练的 SVM。R-CNN 的一个主要瓶颈是，经过 Selective Search（选择性搜索算法）算法选出的每一个候选区域都要经过一次 CNN 前向传播，如图片 1 上面的框架，输入是原始图片经过 SS 算法选取候选区域后，由于后面全连接层的限制，所以候选区域必须得先进行尺寸固定，固定到相同尺寸，才能输入到网络。也就是说，**R-CNN 不能适应不同尺寸的输入**。它必须将候选区缩放至指定大小随后再输入到模型中进行特征提取。



图片 1：R-CNN 与 SSP 框架

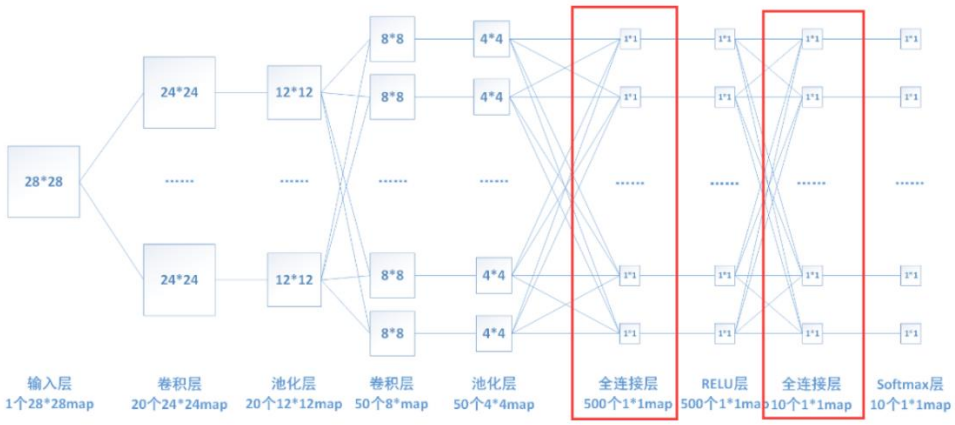
我们知道图像经过裁剪与缩放之后，可能会丢失很多信息。这是一种次优的操作。如图片 2 所示：

图片 2：裁剪与缩放

3.1.1. 关于图像尺寸的理解

在传统的 CNN 中，卷积层能够适应任意尺寸的图像，但全连接层则需要固定大小的输入。这一需求导致 R-CNN 在处理目标检测任务时面临一个主要瓶颈：尽管 SS 算法能够智能地从原始图像中选取候选区域，每个候选区域仍必须调整至固定尺寸（例如 227x227 像素），因为后续全连接层的要求权重矩阵是固定的，即每一次特征图的输入必须都得是一定的大小（即与权重矩阵正好可以相乘的大小），所以网络一开始输入图像的尺寸必须固定，才能保证传送到全连接层的特征图的大小跟全连接层的权重矩阵匹配。但是这一调整过程不仅可能引起图像失真，丢失重要信息，而且限制了网络处理不同尺寸输入的能力。

在图片 3 中，全连接层的区域是连接最为密集的部分，直观地反映出其参数规模之大。这种设计确保了全连接层能够有效地将卷积层提取的特征进行综合分析，为最终的分类或其他任务提供决策支持。



图片 3：网络结构图

3.2. 冗余计算

同时，由于 SS 算法生成的候选区域往往在图像中存在重叠，这意味着很多区域实际上共享着相似的视觉内容。然而，R-CNN 的独立特征提取过程并没有利用这一特点来减少重复计算，反而在这些重叠区域造成了大量冗余的卷积操作。这种冗余不仅增加了处理时间，也增加了对计算资源的需求，限制了 R-CNN 在需要实时或近实时反馈的应用场景中的实用性。

3.3. 解决思路

因此，SPP 提出了一种新的思路：既然卷积层可以适应任意尺寸的输入，那么在最后的特征图与全连接层之间加入一种特殊的机制——空间金字塔池化层，也应该可以适应不同尺寸的输入。如图片 1 中的下面的框架，原图像经过选择性搜索算法选取候选区域，然后经过卷积层处理，再通过空间金字塔池化层将不同尺寸的特征图转换为固定长度的特征向量，进而输入到全连接层，使得网络可以适应不同尺寸的输入。

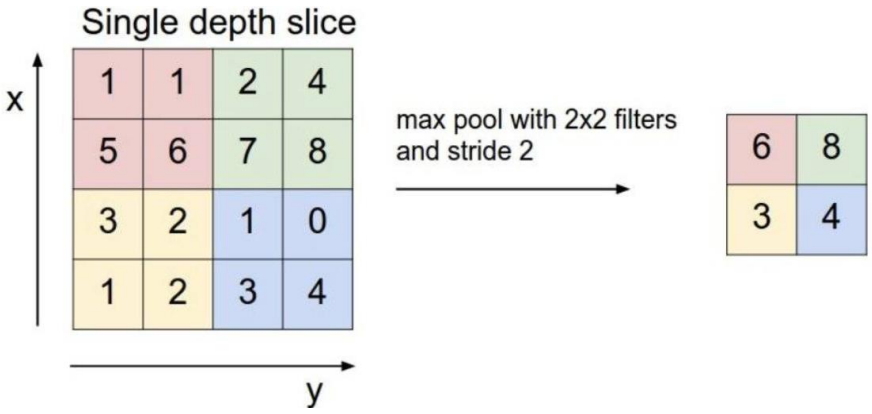
同时 SPP 允许网络只对整个图像计算一次特征图，然后在此基础上对任意区域进行特征池化，生成固定长度的特征表示。这种方法不仅减少了冗余计算，还提高了特征提取的效率，使得网络能够更快地处理大量候选区域，同时保持了较高的检测精度。

4. 论文核心方法实现步骤

4.1. pooling（池化层）

同卷积层一样，池化层每次对输入数据的一个固定形状窗口（又称池化窗口）中的元素计算输出。不同于卷积层里计算输入和核的互相关性，池化层直接计算池化窗口内元素的最大值或者平均值。该运算也分别叫做最大池化或平均池化。在二维最大池化中，池化窗口从输入数组的最左上方开始，按从左往右、从上往下的顺序，依次在输入数组上滑动。当池化窗口滑动到某一位置时，窗口中的输入子数组的最大值即输出数组中相应位置的元素。

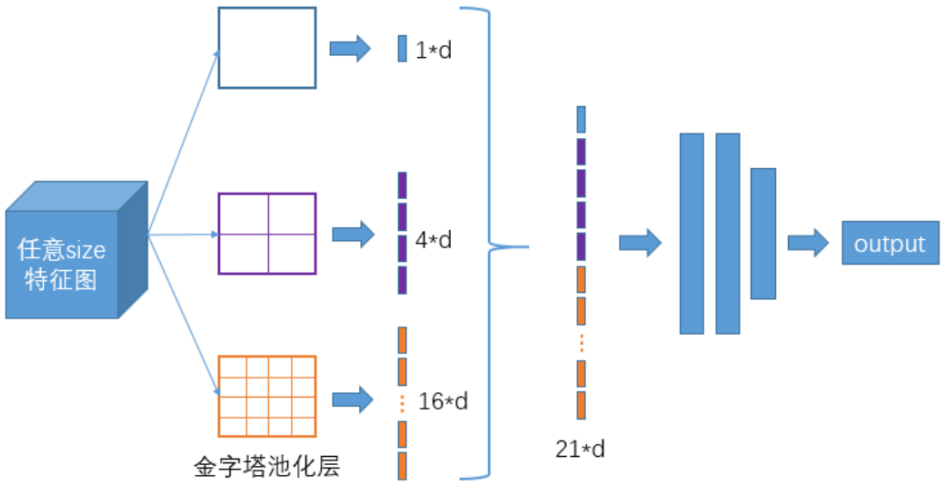
下图展示了池化窗口形状为 2×2 的最大池化，阴影部分为第一个输出元素及其计算所使用的输入元素。输出数组的高和宽分别为 2。



在处理多通道输入数据时，池化层对每个输入通道分别池化，而不是像卷积层那样将各通道的输入按通道相加。这意味着池化层的输出通道数与输入通道数相等。

4.2. 金字塔池化

如何让不同尺寸的特征图进入全连接层，我们直接看下图，下图中对任意尺寸的特征图直接进行固定尺寸的池化，来得到固定数量的特征。



以 3 个尺寸的池化为例，我们先对特征图进行一个最大值池化，即一张特征图得取其最大值，得到 1*d(d 是特征图的维度)个特征；再对特征图进行网格划分为 2x2 的网格，然后对每个网格进行最大值池化，那么得到 4*d 个特征；最后，对特征图进行网格划分为 4x4 个网格，对每个网格进行最大值池化，得到 16*d 个特征。

接着将每个池化得到的特征拼接起来即得到固定长度的特征个数（特征图的维度是固定的），在这个例子中结果是 21 维的输出。

如此一来，无论输入特征图的尺寸发生如何变化，SSP 均可将其转化为固定大小的尺寸进行输出。

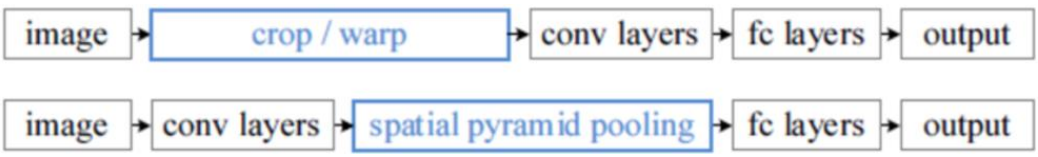
4.1. SPP-Net 工作流程

1. **候选区域生成**：使用 SS 算法从原始图像中快速生成大量高质量的候选区域。
2. **特征提取**：将原始图像输入到 CNN 中，通过卷积层进行特征提取，生成特征图。这些特征图捕捉了图像在不同层次的抽象特征。
3. **空间金字塔池化**：将特征图输入到 SPP 层。SPP 层将特征图划分为不同尺度的池化区域（例如 1x1, 2x2, 3x3 等），这些区域覆盖从细节到更粗糙的表示。并在每个区域内执行池化操作，通常是最大池化，以提取关键特征。
4. **多尺度特征整合**：SPP 层将不同尺度的池化结果整合，形成一个固定长度的特征向量。这一步骤保留了多尺度的特征信息，并且使得输出尺寸与输入尺寸无关。
5. **全连接层处理**：将 SPP 层输出的固定长度特征向量送入一个或多个全连接层，进行进一步的特征整合和分类或检测任务的决策。
6. **多尺度训练策略**：为了提高网络对不同尺寸输入的泛化能力，SPP-Net 在训练时使用多尺度图像。这意味着在每个训练周期中，网络会被不同尺寸的图像所训练，从而提高其尺度不变性。
7. **特征图的多视图测试**：在测试阶段，SPP-Net 可以从特征图的不同区域和不同尺度提取特征，以生成多个固定长度的特征表示，进一步提高任务性能。
8. **最终预测**：经过全连接层处理后，网络输出最终的预测结果，这可以是类别概率（用于分类任务）或对象的边界框和类别（用于检测任务）。

对于检测任务，可能需要进行边框回归来微调候选区域的位置。此外，如果使用分类器（如 SVM），则需要对分类器进行训练，以便对候选区域进行分类。

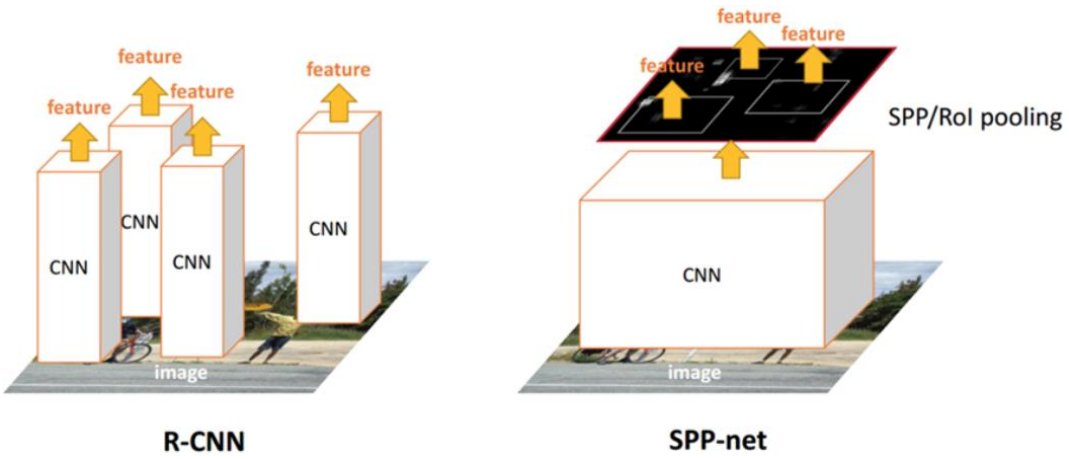
SPP-Net 的设计减少了冗余计算，提高了处理速度，并且由于多尺度特征的整合，提高了对不同尺寸和变形的鲁棒性。

4.2. 与 R-CNN 的对比



在上图中，第一排为 R-CNN 的流程图，第二排为 SSP-Net 的流程图。

相比于 R-CNN 对于多个候选区域依次进入特征提取网络进行图像特征提取，SPP-Net 采用直接将全图输入到卷积神经网络中提取图像特征，然后在图像特征图上找到候选区域的图像特征。此外，采用 SSP 将不同尺寸的候选区域特征转化为固定尺寸的值，避免了 RCNN 采用的缩放操作。如下图所示：



R-CNN 要对每个区域计算卷积，而 SPP-Net 只需要计算一次卷积，从而节省了大量的计算时间，相比 R-CNN 有一百倍左右的提速。

5. 论文涉及方法分析

5.1. R-CNN

2014 年是计算机视觉领域的关键一年，R-CNN 论文的发表标志着这一年的发展。这篇论文不仅展示了 CNN 在物体检测方面实现高性能的潜力，还解决了一个具有挑战性的问题：通过创建边界框来精确定位图像中的物体。

虽然 CNN 在图像分类任务中已经占据了主导地位，但物体检测需要更复杂的解决方案。这就是 R-CNN 算法发挥作用的地方，它为这一复杂问题提供了一种新颖的方法。R-CNN 为后续物体检测领域的创新铺平了道路，包括 Fast R-CNN、Faster R-CNN 和 Mask R-CNN，它们都建立在前身的基础上并加以增强。要掌握这些高级 R-CNN 变体的细微差别，必须在原始 R-CNN 架构中打下坚实的基础。

R-CNN 全称 region with CNN features，其实它的名字就是一个很好的解释。用 CNN 提取出 Region Proposals 中的 featues，然后进行 SVM 分类与 bbox 的回归。

5.1.1. Region Proposal（候选区域）

R-CNN 首先将输入图像划分为多个子区域 (1k~2k 个)。这些区域称为“候选区域”。候选区域步骤负责在图像中生成一组可能包含对象的潜在区域。R-CNN 不会自行生成这些提议；相反，它依靠 Selective Search 或 EdgeBox 等外部方法来生成区域提议。

5.1.2. Selective Search（选择性搜索算法）

在两阶段（two-stage）目标检测算法中，生成高质量的候选区域是首要步骤。传统方法，如穷举法或滑动窗口技术，通过在原始图像上应用不同尺度和大小的窗口来确定可能包含物体的区域。这种方法虽然全面，但存在明显缺陷：它不仅计算成本高昂，而且由于需要评估大量候选区域，往往产生大量冗余，导致效率低下。此外，这种方法很难覆盖所有可能的尺度，因此准确性也受到限制，这在实际应用中是不切实际的。

我们先来看一组图片：

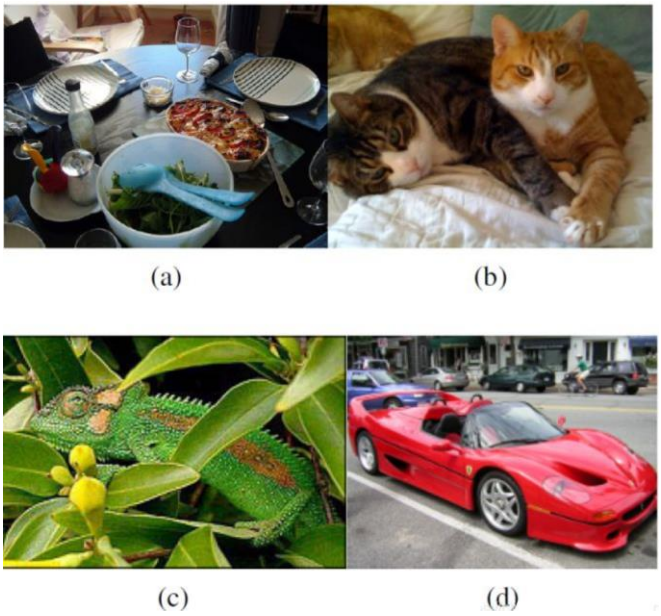


图 a 展示了物体之间可能存在层级关系，比如：一个勺子位于碗中；

图 b 揭示了颜色特征在某些情况下（区分颜色不同的猫）的有用性，而纹理特征则不足以区分它们；

图 c 则展示了纹理特征在区分颜色相似的变色龙时的重要性，而颜色特征在这里并不适用；

图 d 强调了上下文关系在物体识别中的作用，例如轮胎作为车辆的一部分，并非因为颜色或纹理相似，而是因为它与车辆的组成关系。

在目标检测中，一种传统且直接的方法是使用不同尺寸的矩形框，通过逐行扫描图像来提取特征并判断是否包含待检测物体。这种方法，被称为 exhaustive search，因其高计算复杂度而受到限制。

Selective Search 算法提供了一种有效的解决方案。它通过智能地合并图像中的区域来生成候选区域，有效地减少了冗余，并显著降低了计算量。Selective Search 通过评估区域的相似性和完整性，优先选择那些更有可能包含目标物体的区域，同时排除那些不太可能的区域。这种方法不仅提高了检测的准确性，而且由于减少了候选区域的数量，也极大地提高了目标检测的效率。

5.1.2.1. Selective Search 的分层分组算法流程

1. 使用《Efficient Graph-Based Image Segmentation》中的方法初始化区域集 R；
2. 计算 R 中相邻区域的相似度，并以此构建相似度集 S；
3. 如果 S 不为空，则执行以下 7 个子步骤，否则，跳至步骤 4：

- 获取 S 中的最大值 $s(r_i,r_j)$;
- 将 r_i 与 r_j 并成一个新的区域 r_t ,即$r_t = r_i \cup r_j$;
- 将 S 中与 r_i 有关的值 $s(r_i,r_*)$ 剔除掉，即$S = S \setminus s(r_i,r_*)$);
- 将 S 中与 r_j 有关的值 $s(r_*,r_j)$ 剔隙掉，即$S = S \setminus s(r_*,r_j)$;
- 使用步骤 2 中的方法,构建 S_t,它是 S 的元素与 r_t 之间的相似度构成的集合;
- 将 S_t 中的元素全部添加到 S 中，即$S = S \cup S_t$;
- 将 r_t 放入 R 中，即$R = R \cup r_t$。

4. 将 R 中的区域作为目标的位置框 L LL，这就是算法的执行结果。

5.1.2.2. 相似度计算

原论文里考虑了四个方面的相似度：颜色s_{colour}、纹理$s_{texture}$、尺度s_{size}、空间交叠s_{fill},并将这四个相似度以线性组合的方式综合在一起，作为最终被使用的相似度$s(r_i,r_j)$:

$$s(r_i,r_j) = \alpha_1 s_{colour}(r_i,r_j) + \alpha_2 s_{texture}(r_i,r_j) + \alpha_3 s_{size}(r_i,r_j) + \alpha_4 s_{fill}(r_i,r_j)$$

接下来我们将详细介绍四种相似度的计算方法。

5.1.2.3. 颜色相似度

将区域的颜色空间转换为直方图，三个颜色通道的 bins 取 25。于是我们可以得到某个区域 r_i的颜色直方图向量：$C_i = \{c_i^1,\ldots,c_i^n\}$，其中 n=75（计算方式：bins×n_channels = 25×3），并且C_i是用区域的L_1范数归一化后的向量。关于$r_t = r_i \cup r_j$的C_t，计算方式是这样的：

$$C_t = \frac{size(r_i) \times C_i + size(r_j) \times C_j}{size(r_i) + size(r_j)}$$

而r_t尺寸的计算方式为：$size(r_t) = size(r_i) + size(r_j)$

5.1.2.4. 纹理相似度

对每一个颜色通道，在 8 个方向上提取高斯导数($\sigma = 1$)。在每个颜色通道的每个方向上，提取一个 bins 为 10 的直方图，从而得到每个区域r_i的纹理直方图向量$T_i = \{t_i^1,\ldots,t_i^n\}$，其中 n=240（计算方式：$n\_orientations \times bins \times n\_channels = 8 \times 10 \times 3$），$T_i$也是用区域的$L_1$范数归一化后的向量。

纹理相似度的计算公式：

$$s_{texture}(r_i,r_j) = \sum_{k=1}^{n} min(t_i^k,t_j^k)$$

5.1.2.5. 尺度相似度

用于优先合并小区域。其计算公式为：

$$s_{size}(r_i,r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)}$$

其中，size(im)是整张图片的像素级的尺寸。

5.1.2.6. 空间交叠相似度

用于优先合并被包含进其他区域的区域。其计算公式为：

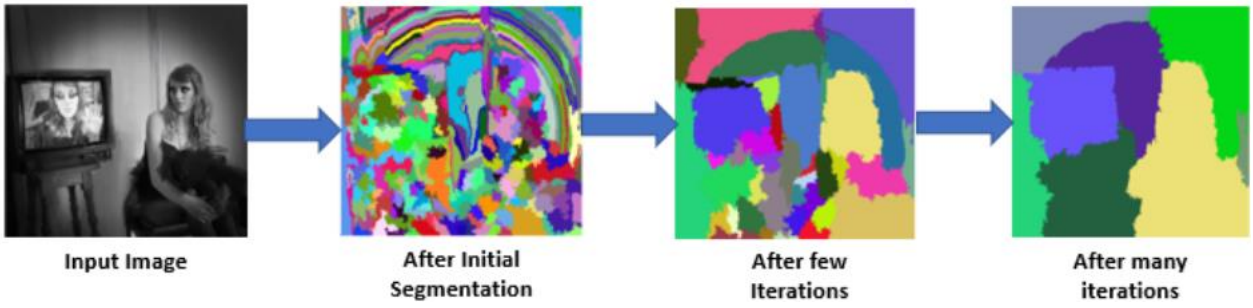
$$s_{fill}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)}$$

其中，BB_{ij}是能够框住r_i与r_j的最小矩形框。

总结来说，SS 算法的工作流程为：

- 使用一种过分割手段，将图像分割成小区域（1k~2k 个）
- 计算所有邻近区域之间的相似性，包括颜色、纹理、尺度等
- 将相似度比较高的区域合并到一起
- 计算合并区域和临近区域的相似度
- 重复 3、4 过程，直到整个图片变成一个区域

在每次迭代中，形成更大的区域并将其添加到区域提议列表中。这种自下而上的方式可以创建从小到大的不同尺寸的候选区域，如图所示：



5.1.3. 模型 pre-training

在实际测试的时候，模型需要通过 CNN 提取出候选区域中的特征，用于后面的分类与回归。所以，如何训练好 CNN 成为重中之重。

由于物体标签训练数据少，如果要直接采用随机初始化 CNN 参数的方法是不足以从零开始训练出一个好的 CNN 模型。基于此，采用有监督的预训练，使用一个大的数据集（ImageNet ILSVC 2012）来训练 AlexNet，得到一个 1000 分类的预训练（Pre-trained）模型。

5.1.4. 模型微调

在将 R-CNN 模型应用于 PASCAL VOC 测试集时，模型首先利用 SS 算法在每张图像上定位大约 2000 个候选区域。由于这些候选区域的大小不一，而 AlexNet 作为 R-CNN 的卷积神经网络基础，要求输入图像的尺寸固定为 227×227 像素，因此必须对 RP 进行尺寸调整（resize）和变形处理以适应网络输入的要求。

在变形之前，先在候选区域周围添加了 16 像素的填充（padding），然后进行各向异性缩放，以保持区域内容的完整性。这种预处理步骤虽然改变了图像的原始尺寸，但却提高了模型的平均精度均值（mean Average Precision, mAP）达 3 到 5 个百分点，显示出其在检测性能上的优势。

然而，原始的 CNN 模型是为了在无变形的 ImageNet 数据集图像上提取特征而设计的。现在，当我们将其应用于 VOC 检测数据集上的变形图像时，就需要对模型进行领域特定的参数调优，也就是微调（fine-tuning）。这一过程使用从 VOC 训练图像中搜索到的候选区域作为训练样本。

微调的目的是让 CNN 适应新的检测任务和变形后的图像领域。为此，我们将预训练模型最后的 1000 类全连接层替换为 21 类（20 种物体类别加背景）的分类层。接着，计算每个候选区域与真实标注（ground truth）的交并比（IoU）。IoU 值大于 0.5 的 RP 被视为正样本，其余则视为负样本。

鉴于一张图像中负样本的数量通常远大于正样本，为保证样本分布均衡，我们采用上采样技术来增加正样本的比例。在每次迭代中，我们通过分层采样从两张图像中选取 32 个正样本和 96 个负样本，构成一个 128 个样本的 mini-batch，保持正负样本比例为 1:3。

R-CNN 采用 0.001 的学习率和随机梯度下降（SGD）算法来优化模型。通过这种方式，我们能够训练出一个更加精准的目标检测模型，它能够适应 PASCAL VOC 数据集的特定需求。

5.1.1. 基于 SVM 的训练

在 R-CNN 模型中，作者采用了支持向量机（SVM）对每一类物体进行分类，总共训练了 N 个 SVM 分类器，其中 N 等于类别的数量（在此案例中为 21）。让我们深入了解 SVM 分类器的训练和使用过程。

在 SVM 的训练阶段，作者定义了正例和负例的标准：与 ground-truth 重叠度小于 0.3 的区域被视为负例，而 IOU 高于 0.7 的则被视为正例。不满足这两个条件的区域则被排除在训练集之外。SVM 分类器根据这些样本进行训练，并输出预测标签。随后，这些预测标签与真实标签进行比较，计算损失函数（loss），并据此优化 SVM 参数。

训练过程中的一个关键细节是，由于 SVM 通常在小样本数据集上训练，容易出现负样本数量远超过正样本的情况。为了解决这一问题，作者采用了困难负样本挖掘（hard negative mining）的方法。初始阶段，所有样本都被用于训练。然后，通过选择那些最有可能被错误分类的负样本（即具有最高得分的负样本），并将它们加入到训练集中，以提高分类器的性能。这个过程重复进行，直到达到某个停止条件，例如分类器性能不再提升。

困难负样本挖掘的重要性在于，当负样本数量巨大时，它们中的大多数可能对优化过程贡献很小，因为它们远离最优分界面。而通过专注于那些位于分界面附近的困难负样本，可以显著提高优化效率并减少过拟合的风险。这种方法确保了 SVM 在面对样本不平衡的情况时，依然能够维持良好的泛化能力。

通过这种方法，作者成功地训练了适用于小样本数据集的 SVM 分类器，即使在正负样本数量极度不平衡的情况下，也能有效地进行分类任务。

5.1.2. Why SVM?

为什么在 R-CNN 中选择使用 SVM 进行分类，而不是直接使用 CNN 最后一层的 softmax 分类器？

仔细观察可以发现，在训练 CNN 提取特征时，我们将 IOU（交并比）大于 0.5 的 bounding box 视为正样本，小于 0.5 的视为负样本，这是一种大样本训练策略。然而，在 SVM 分类阶段，我们只将完全包围物体的 bounding box（即 IOU 大于 0.7）视为正样本，IOU 小于 0.3 的视为负样本，这是一种小样本训练策略。CNN 训练需要大量数据以避免过拟合，因此采用了较低的 IOU 阈值，即使 bounding box

只包含物体的一部分，也会被标记为正样本。相比之下，现在我们对训练样本的 IOU 要求更为严格，所以使用更适合小样本训练的 SVM。

这种差异导致了在使用 softmax 层进行分类时可能存在的问题。如果使用较低的 IOU 阈值，可能会导致最终的 bounding box 位置不准确，因为如果 bounding box 与真实标注差距太大，通过线性回归可能无法收敛。同时，这也会影响类别识别的精度，实验表明，这样做会导致 mAP（平均精度均值）下降几个百分点。如果我们提高 IOU 阈值，又可能导致训练样本数量太少，从而发生过拟合。

这就是所谓的"鱼和熊掌不可得兼"的困境。实际上，这个问题的根源在于 VOC 数据集的规模较小，限制了优化操作。因此，作者最终选择了 SVM 进行分类，而 CNN 仅用于提取特征。

5.1.3. Bounding Box Regression（边界框回归）

R-CNN 中的边框回归器是一个关键组件，它负责微调候选区域的位置，以便更准确地定位目标物体。以下是 R-CNN 边框回归器训练的完整过程：

在 R-CNN 的训练阶段，我们首先通过 SS 算法从输入图像中提取出大量的候选区域。这些候选区域随后被用于边框回归器的训练。我们只选择那些与真实标注（ground truth）的 IOU（交并比）高于特定阈值（例如 0.7）的候选区域作为正样本，因为它们与目标物体的边界框有较高的重合度。

对于每个正样本，我们计算其边界框与真实标注边界框之间的偏移量，这个偏移量是一个四元素的向量 (dx, dy, dw, dh)，分别表示中心点在 x 和 y 方向上的偏移，以及边界框宽度和高度的相对变化。这些偏移量将作为边框回归器的训练标签。

边框回归器通常采用线性回归模型，学习候选区域特征向量与偏移量之间的关系。特征向量由之前训练好的 CNN 模型提取，而损失函数则使用平滑 L1 损失来衡量预测偏移量与真实偏移量之间的差异。

训练过程中，我们将特征向量和对应的偏移量标签输入回归器，并通过反向传播算法来更新回归器的参数，目的是最小化损失函数。通过多次迭代训练，不断优化回归器的参数，直到模型在验证集上的性能不再显著提升。

在测试阶段，我们使用训练好的回归器对 SVM 分类器筛选出的候选区域进行边框微调。这一步骤通过调整候选区域的位置和尺寸，进一步提高了检测的准确性。

值得注意的是，R-CNN 选择使用 SVM 进行分类，而不是直接使用 CNN 的 softmax 层，是因为 SVM 更适合小样本训练，能够在样本不平衡的情况下避免过拟合。而 CNN 则专注于提取特征，这些特征随后被用于 SVM 分类和边框回归。

通过这种方式，R-CNN 的边框回归器不仅提高了检测的精度，而且通过微调候选区域的位置，确保了检测结果的可靠性。

5.1.4. R-CNN 存在的问题

- **训练时间长**：主要原因是分阶段多次训练，而且对于每个候选区域都要单独计算一次特征图，导致整体的时间变长。
- **占用空间大**：每个候选区域的特征图都要写入硬盘中保存，以供后续的步骤使用。
- **multi-stage**：文章中提出的模型包括多个模块，每个模块都是相互独立的，训练也是分开的。这会导致精度不高，因为整体没有一个训练联动性，都是不共享分割训练的，自然最重要的 CNN 特征提取也不会做的太好。

- **测试时间长**：由于不共享计算，所以对于 test image，也要为每个候选区域单独计算一次特征图，因此测试时间也很长。

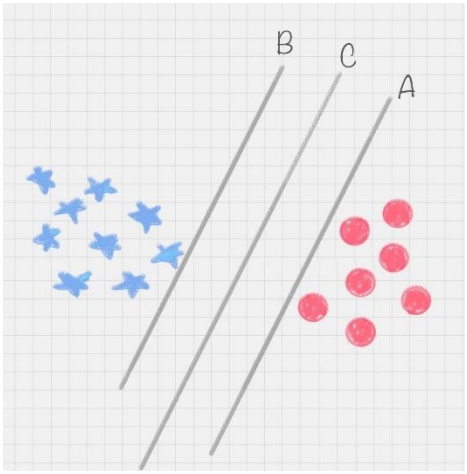
5.2. SVM（支持向量机）

支持向量机（Support Vector Machine，SVM） 是一种广泛应用于机器学习领域的分类技术。作为一种有监督学习模型，SVM 依赖于预先标注好的数据集进行训练，其中每个数据样本都被赋予了相应的分类标签。这种标签提供了数据与特定类别之间的对应关系，指导机器学习模型识别和理解不同类别的特征。

与有监督学习相对的是无监督学习，在无监督学习中，数据样本并未附带任何分类标签。这通常发生在缺乏先验知识或标注成本过高的情况下。无监督学习的目标是探索数据的内在结构，如通过聚类分析将数据分组，为进一步的人工分析提供便利。

5.2.1. SVM 的工作原理

用 SVM 计算的过程就是帮我们找到一个超平面，能够将样本区分的过程，这个超平面就是我们的 SVM 分类器。比如下图所示的直线 A、直线 B 和直线 C，究竟哪种才是更好的划分呢？



很明显图中的直线 B 更靠近蓝色球，但是在真实环境下，球再多一些的话，蓝色球可能就被划分到了直线 B 的右侧，被认为是红色球。同样直线 A 更靠近红色球，在真实环境下，如果红色球再多一些，也可能会被误认。所以相比于直线 A 和直线 B，直线 C 的划分更优，因为它的鲁棒性更强。

为了寻找到直线 C 这个更优的答案，我们需要引入一个 SVM 特有的概念：**分类间隔（margin）**。

在实际应用中，我们面临的分类问题往往不局限于二维平面，而是存在于更高维度的空间中。在这种情况下，简单的直线分类边界（C）扩展为多维空间中的**决策超平面（Hyperplane C）**。SVM 的目标是在保持决策超平面 C 的分类能力不变且不引起分类错误的前提下，寻找一个最优的位置。

为了找到这个最优决策超平面，我们可以想象将决策超平面在多维空间中移动，直到达到两个极限位置，如图中所示的决策超平面 A 和 B。这两个极限位置是关键的，因为一旦超出这些位置，就会引起分类错误。换言之，极限位置 A 和 B 是分类间隔的边界，它们与最优决策超平面 C 之间的距离定义了分类间隔的大小。

分类间隔是 SVM 中区分不同类别数据的关键区域，其宽度反映了模型对数据点的判别能力。SVM 算法的优化目标是最大化这个间隔（max margin），同时确保分类的正确性。通过这种方式，SVM 不仅能够准确地进行分类，还能够提高模型的泛化能力，使其在面对新的、未见过的数据时，也能做出可靠的预测。

5.2.2. 点到超平面的距离公式

超平面的数学表达可以写成：

$$g(x) = \omega^T x + b, \text{其中} \omega, x \in \mathbb{R}^n$$

在这个公式里，w、x 是 n 维空间里的向量，其中 x 是函数变量；w 是法向量，b 是偏置项。法向量这里指的是垂直于平面的直线所表示的向量，它决定了超平面的方向。

在 SVM 寻找超平面的过程中，支持向量就是离分类超平面最近的样本点，实际上如果确定了支持向量也就确定了这个超平面。所以支持向量决定了分类间隔到底是多少，而在最大间隔以外的样本点，其实对分类都没有意义。

所以说， SVM 就是求解最大分类间隔的过程，我们还需要对分类间隔的大小进行定义。

在支持向量机（SVM）的理论框架中，我们定义一类样本集合到决策超平面的“距离”为该集合中所有样本点到超平面的最短欧氏距离。对于给定的样本点 xi，其到超平面 $w \cdot x + b = 0$ 的距离 di 被量化为该点与超平面的最短距离，这是一个关键的几何属性。

为了计算 di，我们首先需要确定样本点 xi 与超平面之间的距离公式。在欧几里得空间中，点 xi 到超平面 $w \cdot x + b = 0$ 的距离 di 可以通过以下公式计算得出：

$$d_i = \frac{|\omega x_i + b|}{\|\omega\|}$$

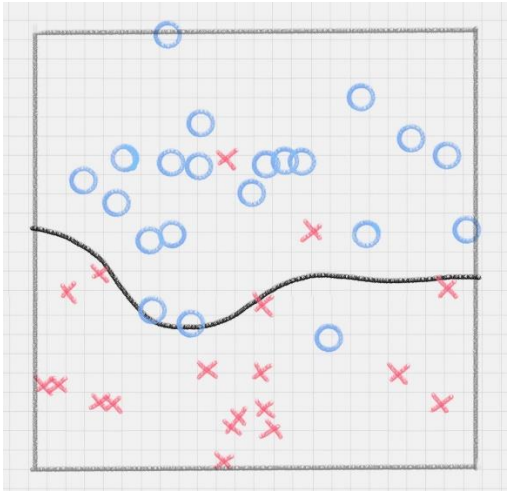
其中|w|为超平面的范数（即其欧氏长度）。而 $|w \cdot xi + b|$ 表示点 xi 到超平面的有符号距离。我们感兴趣的是距离的绝对值，因为它提供了到超平面的实际距离，而不考虑方向。

我们的目标就是找出所有分类间隔中最大的那个值对应的超平面。在数学上，这是一个凸优化问题（凸优化就是关于求凸集中的凸函数最小化的问题，这里不具体展开）。通过凸优化问题，最后可以求出最优的 w 和 b，也就是我们想要找的最优超平面。中间求解的过程会用到拉格朗日乘子，和 KKT（Karush-Kuhn-Tucker）条件。数学公式比较多，这里不进行展开。

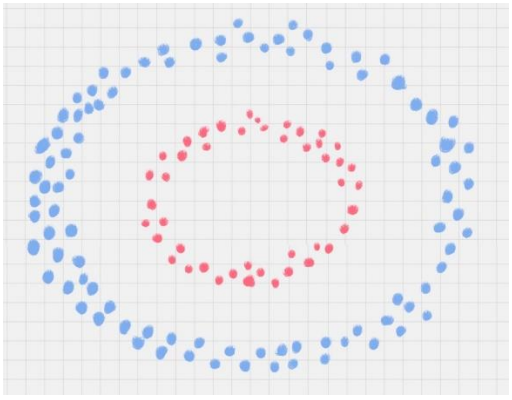
5.2.3. 硬间隔、软间隔和非线性 SVM

在理想情况下，如果数据集是完全线性可分的，我们可以通过学习得到一个模型，称之为硬间隔支持向量机（Hard-Margin SVM）。硬间隔的概念意味着模型在训练数据上实现了完美的分类，没有任何分类错误。换句话说，硬间隔 SVM 要求模型在区分不同类别时没有任何妥协，所有样本点都必须完全正确地被分类。

相对地，软间隔支持向量机（Soft-Margin SVM）引入了一定的容忍度，允许模型在训练过程中出现少量的分类错误。这种方法认识到了现实世界数据的复杂性，理解到在许多情况下，获取完全无噪声的数据是不现实的。软间隔 SVM 通过允许一些误分类的样本点，换取了在新数据上的更好的泛化性能。如下图所示：



除了硬间隔和软间隔 SVM，还存在一种处理非线性可分数据的 SVM 变体，称为非线性支持向量机（Non-linear SVM）。在某些情况下，我们可能会遇到本质上是非线性的数据分布。例如，下图中的样本集，其中的两类数据分别以两个圆圈的形状分布。对于这样的数据集，即使采用最先进的线性分类器，也无法实现有效的分离，因为不存在一个线性的映射函数可以将它们完美分开。在这种情况下，传统的 SVM 同样会面临挑战。



为了解决这一问题，我们需要引入一个强大的概念——核函数（Kernel Function）。核函数的基本作用是将原始数据样本映射到一个更高维的特征空间，在这个新的特征空间中，原本非线性可分的数据变得线性可分。这一过程不改变数据的本质特征，而是通过映射扩展了数据的表示能力。

在高维特征空间中，我们可以继续使用标准的 SVM 推导和计算方法。所有的计算都在这个新的特征空间中进行，而不是原始的输入空间。通过这种方式，非线性 SVM 能够找到最优的超平面，即使在原始空间中这个超平面是非线性的。

核函数的使用极大地扩展了 SVM 的应用范围，使其不仅能够处理线性可分的数据，也能够有效地处理非线性数据。常见的核函数包括线性核、多项式核、径向基函数（RBF）核等，它们各自适用于不同类型的数据分布和分类问题。

5.2.4. 用 SVM 解决多分类问题

SVM 本身是一个二值分类器，最初是为二分类问题设计的，也就是回答 Yes 或者是 No。而实际上我们要解决的问题，可能是多分类的情况，比如对文本进行分类，或者对图像进行识别。针对这种情况，我们可以将多个二分类器组合起来形成一个多分类器，常见的方法有一对多法和一对一法两种。

5.2.4.1. 一对多法

假设我们要把物体分成 A、B、C、D 四种分类，那么我们可以先把其中的一类作为分类 1，其他类统一归为分类 2。这样我们可以构造 4 种 SVM，分别为以下的情况：

（1）样本 A 作为正集，B，C，D 作为负集；

（2）样本 B 作为正集，A，C，D 作为负集；

（3）样本 C 作为正集，A，B，D 作为负集；

（4）样本 D 作为正集，A，B，C 作为负集。

这种方法，针对 K 个分类，需要训练 K 个分类器，分类速度较快，但训练速度较慢，因为每个分类器都需要对全部样本进行训练，而且负样本数量远大于正样本数量，会造成样本不对称的情况，而且当增加新的分类，比如第 K+1 类时，需要重新对分类器进行构造。

5.2.4.2. 一对一法

一对一法的初衷是想在训练的时候更加灵活。我们可以在任意两类样本之间构造一个 SVM，这样针对 K 类的样本，就会有 C(k,2) 类分类器。

比如我们想要划分 A、B、C 三个类，可以构造 3 个分类器：

（1）分类器 1：A、B；

（2）分类器 2：A、C；

（3）分类器 3：B、C。

当对一个未知样本进行分类时，每一个分类器都会有一个分类结果，即为 1 票，最终得票最多的类别就是整个未知样本的类别。

这样做的好处是，如果新增一类，不需要重新训练所有的 SVM，只需要训练和新增这一类样本的分类器。而且这种方式在训练单个 SVM 模型的时候，训练速度快。

但这种方法的不足在于，分类器的个数与 K 的平方成正比，所以当 K 较大时，训练和测试的时间会比较慢。