第九讲 基于深度网络的识别与检测（机器视觉 3）

2025 / 05 / 24

文件：机器视觉-3.pdf、code-3.zip
作业：机器视觉-3

1. 基于深度网络的图像识别技术

1.1 整体思路

- 计算机**难以直接理解**原始感官输入数据的含义，如表示为像素值集合的图像与对象标识（行人、车辆和动物）的关系
- 深度学习可以将所需的**复杂映射**分解为一系列嵌套的**简单映射**（每个简单映射由模型的不同层描述）来解决这一难题
  - 输入展示在**可见层 (visible layer)** ，这样命名的原因是因为它包含能直接观察到的变量
  - **隐藏层 (hidden layer)** 是一系列从图像中提取越来越多抽象特征的层，它们的值不在最终结果里给出
    - **第一隐藏层**描述边缘
    - **第二隐藏层**识别角和扩展轮廓的边集合
    - **第三隐藏层**可以找到角和轮廓的特定集合来检测特定对象的整个部分（**部件**）
  - 最后，根据图像描述中包含的对象部分，可以识别图像中存在的对象
- 深度神经网络在图像处理领域的意义：
  - 随着人们对深度学习的研究不断深入以及所掌握的技术不断成熟，深度神经网络逐渐成为了图像处理领域的**核心算法之一**，并在可供学习**数据充足**的时候有着稳定的表现，完成相应的图像处理任务
  - 深度神经网络用于处理一般的大规模图像分类问题是，不仅可以直接用于构建阶层分类器**进行分类任务**，还可以在精细分类识别中**专门用于提取图像的判别特征**，一共其他分类器进行学习会进行更复杂的检测等任务

1.2 网络基本结构组成

- 以 $\mathrm{AlexNet}$ 作为例子，介绍具体深度神经网路i：
  - 深度神经网络的结构
  - 相关的基本概念
- 其结构十分基础且**具有代表性**，其中：
  - 卷积层
  - ReLU 非线性激活层
  - 池化层
  - 全连接层
  - $\mathrm{softmax}$ 归一化层

1. **卷积层**

- **卷积**，也可称作**旋积**或**摺积** (Convolution，可简称为 Conv) ，本质是一种数学运算。在泛函分析中的定义是：通过两个函数 f 和 g 生成的第三个函数的一种数学算子，表征函数 f 与 g 经过翻转和平移的重叠部分函数值乘积对重叠长度的积分。设：$f(x)$ ，$g(x)$ 是 $\mathbb{R}$ 上的两个可积函数，作积分：

$$\int_{-\infty}^{\infty} f(r)g(x-r)dr$$

- 卷积操作的数学运算过程：
  - 左边的是将边界用 0 填充的原图像
  - 中间是一个 3×3 的卷积核
  - 右边是左上角像素对应的卷积结果
- 卷积核滑过整个图像，在每个位置上通过求取卷积核与原图对应区域（被卷积核覆盖的区域）之间的乘积和来得到卷积结果。
- **卷积步长**是每进行一次卷积运算卷积核所移动的像素单位
- 卷积神经网络三个基本概念：
  i. **局部感受野** ($\mathrm{local\ receptive\ fields}$)
     - 字面意思，感受野是指视觉感受区域的大小
     - 感受野的定义是卷积神经网络每一层输出的特征图上的像素在原始图像上映射的区域大小
  ii. **权共享值** ($\mathrm{shared\ weights}$)
     - 同一个隐藏层（即输出的特征图）中的所有神经元（即输出像素）都是通过同一个卷积核卷积得到，以检测同一个特征在图像的各个位置是否存在，将从输入层到隐藏层的这种映射称为特征映射
     - 该特征映射的**权重**，即**卷积核参数**，被称为**共享权值**。

- 其 “共享” 的含义源自于：输入图像中的每个局部感受野均被同样的卷积核进行卷积。为了进行图像识别，通常需要不止一个的特征映射，因此一个完整的卷积层包含若干个不同的特征映射，也就是若干个**不同的卷积核**（后面称为“**通道**”）。
- **卷积核可以改变特征图的尺寸大小**

iii. **池化** (pooling)

- 池化操作通常紧随卷积层之后使用，其作用是简化卷积层的输出

2. **非线性激活层**

- 非线性激活层是在卷积处理之后，通过激活函数进行非线性变换，从而得到输出信号。最后输出的信号具有 $f(wx+b)$ 的形式，其中 f 为**非线性激活函数**，w 为卷积核的权重矩阵，b 为偏置
- 设 $X_1 \cdots X_n$ 等 n 个输入分别对应着卷积核的权重因子 $w_1 \cdots w_n$ 以及偏置 b 。这里把输入 X_i 乘以对应的权重因子 w_i 再加上 b 的结果记为 u ，这一步骤可以看作是进行完整的卷积操作。而激活函数 f 是作用在 u 上的，也就是说这个神经元最终的输出结果为 $y_i = f(u)$

i. **线性整流函数** (ReLU - Rectified Linear Units)

- 近年来的神经网络倾向于使用 ReLU 替代掉 Sigmoid 函数作为隐藏层的激活函数，它的定义如下：

$$f(x) = \max(x, 0)$$

- 由于在 ReLU 函数进行激活之前，卷积核卷积所得到的结果有正有负：正的结果说明感受野与卷积核相关性较大，倾向于找到特征，因此被 ReLU 函数所激活；而负的结果说明卷积核并未找到特征，该结果可以忽略不计的，因此用此函数进行舍弃

ii. **Softmax 函数**

- 在机器学习尤其是深度学习中，Softmax 是个非常常用而且比较重要的函数，尤其在多分类的场景中使用广泛。它把一些输入映射为 $0 \sim 1$ 之间的实数，并且归一化保证和为 1 ，因此多分类的概率之和为 1
- 顾名思义，Softmax 由两个单词组成，
  a. 其中一个是 max 。比如有两个变量 a ，b。如果 $a > b$ ，则 max 为 a ，反之为 b
  - 问题：如果将 max 看成一个分类问题，结果就是非黑即白，最后的输出是一个确定的变量。然而更多的时候，不仅希望分值大的那一项能作为最终分类的结果，同时分值相对小的那些项能作为取值的参考

  b. 用 soft 的概念，即最后的输出是每个分类都有概率被取到
- 假设这里有一个数组 Z ，Z_j 表示 Z 中的第 j 个元素，那么这个元素的 softmax 值就是：

$$y_i = \frac{e^{Z_i}}{\sum_j e^{Z_j}}$$

- softmax 就是将某数组作为原始输出，通过 softmax 函数，映射到 $(0, 1)$ 区间里的值，而这些值的累和为 1（满足概率的性质），因此可以将它理解成概率。在最后选取输出结点的时候，选取概率最大（即对应值最大）的结点作为预测目标

3. **池化层**

- **池化层**的主要目的是通过降采样的方式，在不影响图像质量的情况下，**压缩图像、减少参数**
- 简单来说，假设特征图像大小为 3×3，池化层采用 Max Pooling 方法，大小为 2×2 ，步长为 1 ，**池化的步长和卷积相同**，也是池化核每两次运算间移动的距离。那么图像的尺寸就会从 3×3 变为 2×2
- **池化可以调整特征图的尺寸大小**
- 通常来说，池化方法一般有以下两种：
  i. Max Pooling ，取滑动窗口里最大的值
  ii. Average Pooling ，取滑动窗口内所有值的平均值
- 每一个卷积核可以看作一个特征提取器，不同的卷积核负责提取不同的特征，假如第一个卷积核能够提取出“垂直”方向的特征，第二个卷积核能够提取出“水平”方向的特征，那么池化对其进行 Max Pooling 池化操作后，提取出的是真正能够识别到特征的数值，其余被舍弃的数值，这些数值对于提取特定的特征并没有特别大的帮助。但在进行后续计算中，减小了特征图的尺寸，从而减少了参数，达到减小计算量却不损失效果的目的

4. **全连接层**

- 到**全连接层 (fully-connected)** 这一步，其实一个完整的“卷积部分”就算完成了，如果想要叠加层数，一般也是叠加 " Conv - MaxPooing " 组合层，因为叠加卷积层可以通过提取更多更抽象的特征来达到更好的检测效果。
- 在进入到全连接层之前，需要把特征图展开成一维向量，输出到 Flatten 层（拉伸成一维向量），然后把 Flatten 层的输出输入到全连接层里，对其进行分类
- **Flatten 层**：将所有特征图拉伸成一维，可以大大减少特征位置对分类带来的影响。也就是说无论在画面哪个位置提取到特征，在进行全连接以后均可被检测到，增强进行检测任务时的鲁棒性

- 卷积层、非线性激活层和池化层可以看作是一组操作，组成了最基本的**卷积神经网络**的主体框架。而深度网络就是不断叠加这组操作，并加上不同的改进措施，将更加有效的特征不断筛选出来。而全连接层的意义就是将这些 feature map$ 连接成一维向量，最后再使用已有训练集训练好的分类器进行判别

2. 基于简单卷积网络标志牌识别

3. 基于深度网络的图像检测技术

3.1 问题的提出

- **传统的目标检测**
  - 传统目标检测流程：
    a. 区域选择（穷举策略：采用滑动窗口，且设置不同的大小，不同的长宽比对图像进行遍历，时间复杂度高）
    b. 特征提取（形态多样性、光照变化多样性、背景多样性使得特征鲁棒性差）
    c. 分类器分类（主要有 SVM 、神经网络等）
  - 传统的目标检测算法存在着一些无法避免的局限性，主要表现在以下几个方面：
    - 候选框生成方式时间复杂度高；
    - 计算效率较低；
    - 特征构建复杂，计算资源占用大；
    - 设计的特征质量直接影响分类的准确性；
    - 某些特征提取方式应用面窄；
    - 分类器在大数据集上表现不理想
- 基于深度学习技术的目标检测算法对比经典目标检测算法有一些优势，主要体现在以下几个方面：
  - **在特征构建上**：深度学习网络可以构造更有利于检测任务的特征；
  - **在训练方面**：采用端到端的训练架构，实现了性能与效率的整体优化；
  - **在硬件上**：能使用的计算资源增多，网络的学习训练与预测速度逐渐提升。

3.2 基于区域的卷积神经网络 R-CNN

- 2014 年，$Girshick$ 等人成功地将卷积神经网络应用于目标检测算法中，提出了 **R-CNN (Region-based CNN ，基于区域的卷积神经网络) 架构**
- R-CNN 保留了传统目标检测方法中的思路，即：
  i. 先从输入图像中提取大量候选区域
  ii. 再从候选区域中提取特征，将特征送入分类器进行分类得到类别信息与概率
  iii. 最后使用 NMS（非极大值抑制）的方法剔除与邻域有较多重叠且预测概率较低的区域
- 对比经典的目标检测算法，R-CNN 做了以下调整：
  - 使用**选择性搜索** (Selective Search, SS) 方法来进行候选区域提取 → 候选区域数量更少，有更大的概率包括有意义的图像内容；
  - 使用 **AlexNet**（ 2012 年 $ImageNet$ 竞赛冠军网络）作为用于特征提取所使用的网络（后将其称之为 backbone 网络）→ 更好的提取特征；
  - 使用了迁移学习的方法 → 缩短了模型的训练时间，也提高了模型的泛化性能
- **选择性搜索 (SS)**
  - Selective search 算法考虑了 4 种相似性度量，取值都在 $[0,1]$ 之间，越大越相似:
    - 颜色相似性 $S_{colour}(r_i,r_j)$
    - 纹理相似性 $S_{texture}(r_i,r_j)$
    - 大小相似性 $S_{size}(r_i,r_j)$ ，促使小的区域之间优先合并\
    - 形状相似性 $S_{fill}(r_i,r_j)$ ，合并只能在紧邻的两个区域间进行，远离的两个区域不能合并
  - 最终的相似性度量是上述四个度量的组合：

$$S(r_i,r_j)=a_1\times S_{colour}(r_i,r_j)+a_2\times S_{texture}(r_i,r_j)+a_3\times S_{size}(r_i,r_j)+a_4\times S_{fill}(r_i,r_j)$$

    其中 a_k 取 0 或 1
  - 步骤：
    a. 生成基于 oversegmented 得到细分的区域作为原始区域集合：$R=r_1,r_2,\cdots,r_n$
    b. 根据图像相邻区域特征相似度 $s(r_i,r_j)$ ，计算相似度集合：$S=s(r_i,r_j),\cdots$
    c. 将相似度最高的两个区域合并为新区域 r_{new} ，并将其加入 R
    d. 从 S 中移除所有与 步骤 c 中有关的子集
    e. 计算新区域 r_{new} 与相邻区域的相似度，并加入集合 S
    f. 跳至 步骤 c ，直至：$S=\emptyset$
- R-CNN 在训练与使用的过程中，也表现出了一些缺陷：
  - 对候选区域缩放到固定尺寸再输入卷积神经网络 → **尺寸变换**（破坏图像中原物体的宽高比）的过程会导致图像信息的丢失及位置信息扭曲
  - 对每张图片提取的上千个候选区域**逐个、重复调用卷积神经网络** → 特征提取的计算十分耗时，计算开销也十分巨大，很大程度上影响了训练与检测速度

3.3 快速的基于区域的卷积神经网络 Fast-R-CNN

- 针对 R-CNN 的缺陷，2015 年 *Girshick* 等人提出了 Fast R-CNN ，该方法参考了 2014 年 *He* 等人提出的 SPP-Net (Spatial Pyramid Pooling in Deep Convolutional Networks)
- RoI Pooling 层
  - Fast R-CNN 中的 RoI Pooling 层有两种输入：
    - 卷积网络提取的完整特征图
    - 候选区域位置对应的特征图区域块
  - 两次量化：
    a. RoI 候选区坐标除以 16 并取整（第一次量化），与特征图一致
    b. 映射到特征图上的 RoI 划分成 $W \times H$ 个块，由于划分过程得到的块的坐标是浮点值，所以这里还要将块的坐标做第二次量化，采用最大池化操作处理每个块，最终输出的 $W \times H$ 大小的 RoI Pooling 特征

作业：

1. 补全代码中空缺的部分:

```
line 056 : x = x.view(batch_size, -1)

line 084 : running_correct += (predicted == target).sum().item()

line 103 : correct += (predicted == labels).sum().item()
```

2. 修改网络结构：

修改如下两句：

```
line 37 : torch.nn.Conv2d(1, 10, kernel_size=9),

line 42 : torch.nn.Conv2d(10, 20, kernel_size=3),
```