深度学习原理与实践

MindFlow Author

2026 年 1 月 19 日

第 1 章 *

目录

1 * i

1 神经网络基础 1

1.1 感知机与激活函数 1

1.2 反向传播算法 2

2 卷积神经网络 (CNN) 4

2.1 卷积运算 4

2.2 多图排版演示 5

2.2.1 figurerow 环境 5

2.2.2 figuregrid 环境 5

2.3 现代架构演进 5

2.4 代码实现 6

3 * 7

术语与符号表 7

第 1 章 神经网络基础

本章导读

本章介绍神经网络的基本构成单元，包括感知机模型、激活函数选择、以及反向传播算法的数学原理。理解这些基础是深入学习深度学习的前提。

1.1 感知机与激活函数

神经网络的基本单元是感知机。给定输入 $\boldsymbol{x} \in \mathbb{R}^n$，输出由下式给出：

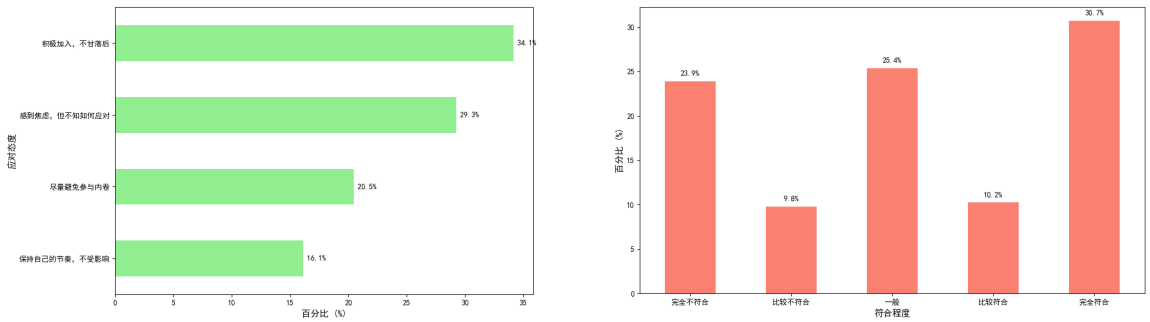
$$y = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

其中 $\boldsymbol{W}$ 是权重，$\boldsymbol{b}$ 是偏置，σ 是激活函数。

定义 1.1.1 (激活函数)

激活函数 $\sigma : \mathbb{R} \to \mathbb{R}$ 为神经网络引入非线性。常见选择包括：

- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- ReLU: $\mathrm{ReLU}(x) = \max(0, x)$
- Tanh: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$



(a) ReLU: $\max(0, x)$ (b) Sigmoid: $1/(1+e^{-x})$

图 1-1　常见激活函数可视化

注

ReLU (Rectified Linear Unit) 因其计算简单且缓解了梯度消失问题，已成为现代深度网络的默认选择。但 ReLU 存在" 死亡神经元" 问题，Leaky ReLU 和 GELU 是常见替代方案。

1.2　反向传播算法

反向传播 (Backpropagation) 是训练神经网络的核心算法。其本质是链式法则的高效实现。

定理 1.2.1 (链式法则)

设 $y = f(u)$ 且 $u = g(x)$，则：

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial x}$$

推广到向量形式，若 $\boldsymbol{y} = f(\boldsymbol{u}), \boldsymbol{u} = g(\boldsymbol{x})$，则 Jacobian 矩阵满足：

$$J_{\boldsymbol{y}}(\boldsymbol{x}) = J_{\boldsymbol{y}}(\boldsymbol{u}) \cdot J_{\boldsymbol{u}}(\boldsymbol{x})$$

算法 1 计算图反向传播

对计算图进行拓扑排序
for 每个节点 v_i (按前向顺序) **do**
    计算 v_i 的输出值
end for
初始化输出梯度: $\bar{y} = 1$
for 每个节点 v_i (按反向顺序) **do**
    计算局部梯度 $\frac{\partial v_j}{\partial v_i}$
    累积梯度: $\bar{v}_i = \sum_{j \in \text{Children}(v_i)} \bar{v}_j \frac{\partial v_j}{\partial v_i}$
end for

本章小结

1. 感知机是神经网络的基本单元，由线性变换和非线性激活组成
2. 激活函数的选择影响网络的表达能力和训练稳定性
3. 反向传播是链式法则的高效实现，计算复杂度与前向传播相同

第 2 章 卷积神经网络 (CNN)

本章导读

卷积神经网络是处理网格结构数据（如图像）的利器。本章介绍卷积运算的数学定义、CNN 的核心组件、以及经典架构的演进历程。

2.1 卷积运算

卷积层通过滑动窗口提取局部特征。二维离散卷积定义为：

$$(I * K)(i, j) = \sum_{m} \sum_{n} I(i+m, j+n) K(m, n)$$

其中 I 是输入图像，K 是卷积核。

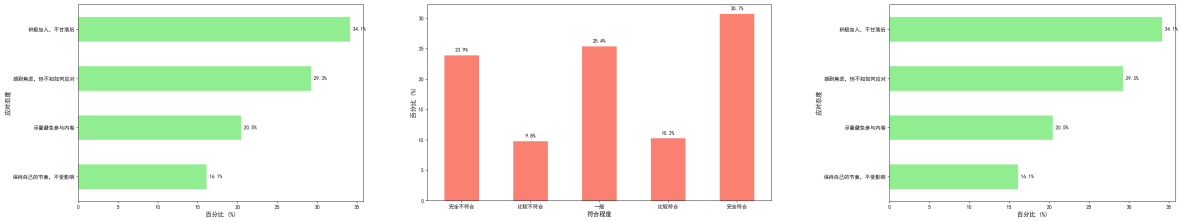
【定义】CNN 关键参数

- **Kernel Size ($k \times k$)**: 卷积核大小，决定感受野
- **Stride (s)**: 步长，控制输出特征图尺寸
- **Padding (p)**: 填充，保持边界信息完整性
- **Dilation (d)**: 膨胀率，扩大感受野而不增加参数

输出尺寸公式：$O = \lfloor \frac{I+2p-d(k-1)-1}{s} \rfloor + 1$

2.2 多图排版演示

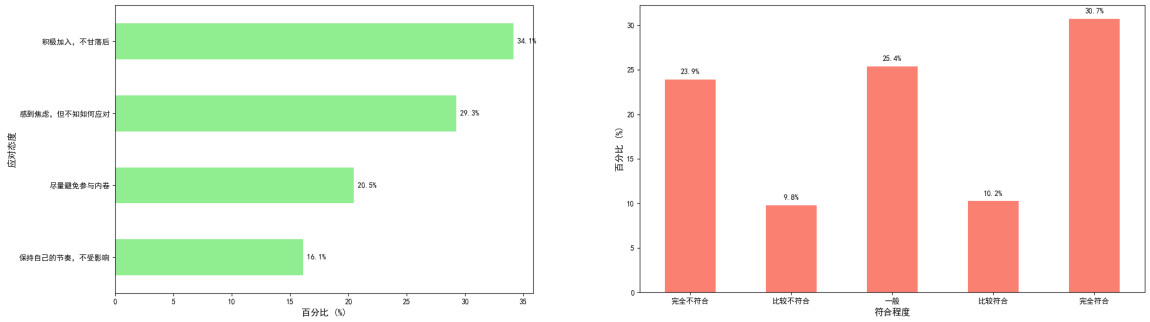
2.2.1 figurerow 环境



(a) 输入图像　(b) 卷积核　(c) 特征图

图 2-1　CNN 卷积过程可视化

2.2.2 figuregrid 环境



(a) 标准残差块　(b) 瓶颈残差块

图 2-2　ResNet 残差块结构

2.3 现代架构演进

从 LeNet 到 Transformer，视觉模型架构不断演进：

LeNet-5 (1998) 第一个成功的 CNN，用于手写数字识别

AlexNet (2012) 引入 ReLU 和 Dropout，使用 GPU 训练，ImageNet 冠军

VGG (2014) 使用小卷积核 (3×3) 堆叠，证明深度的重要性

ResNet (2015) 引入残差连接 $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}) + \boldsymbol{x}$

ViT (2020) 将 Transformer 应用于图像，开启新范式

NOTE (备注)

R

esNet 的成功表明，让网络学习恒等映射 (Identity Mapping) 比学习零映射更容易。这一洞察启发了后续的 DenseNet、Highway Network 等架构。

2.4 代码实现

代码 2.4-1：**PyTorch 残差块实现**

```python
import torch.nn as nn

class ResidualBlock(nn.Module):
    def __init__(self, channels):
        super().__init__()
        self.conv1 = nn.Conv2d(channels, channels, 3, padding=1)
        self.bn1 = nn.BatchNorm2d(channels)
        self.conv2 = nn.Conv2d(channels, channels, 3, padding=1)
        self.bn2 = nn.BatchNorm2d(channels)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        identity = x
        out = self.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        out += identity  # 残差连接
        return self.relu(out)
```

本章小结

- 卷积通过参数共享实现平移不变性
- 深度是 CNN 成功的关键因素
- 残差连接解决了深层网络的退化问题

第 3 章*

术语与符号表

η	学习率 (Learning Rate)
$\mathcal{L}$	损失函数 (Loss Function)
θ	模型参数 (Model Parameters)
CNN	卷积神经网络 (Convolutional Neural Network)
RNN	循环神经网络 (Recurrent Neural Network)