同济大学

TONGJI UNIVERSITY

毕业设计（论文）

课题名称	卷积神经网络模型的数据维度与参数量分析
副 标 题	人工智能导论课程作业
学　　院	软件学院
专　　业	软件工程
学生姓名	林继申
学　　号	2250758
指导教师	曾进
日　　期	2024 年 4 月 2 日

1 分析原理

1.1 卷积层（CONV k-N）

卷积层通过在输入数据上滑动多个卷积核（或过滤器）来提取特征。每个卷积核负责学习输入数据的一种特征。

k 表示卷积核的大小（高度和宽度），N 表示卷积核的数量。每个卷积核有 $k \times k \times C_{in}$ 个权重参数，其中 C_{in} 是输入通道数。加上偏置项（每个卷积核一个），总参数量为 $((k \times k \times C_i n) + 1) \times N$。

当使用 $padding = 1$ 和 $stride = 1$ 时，输出高度和宽度与输入相同，输出通道数为卷积核数量 N。因此，输出维度是 $H \times W \times N$。

1.2 池化层（POOL-n）

池化层用于降低特征图的空间尺寸（高度和宽度），以减少计算量并防止过拟合。

池化层没有参数。

使用 $n \times n$ 的窗口和 $stride = n$（通常没有 $padding$），输出的高度和宽度是输入的 $1/n$，输出通道数不变。输出维度是 $H/n \times W/n \times C$。

1.3 全连接层（FC-N）

全连接层将学习到的“特征”表示映射到最终的输出，如分类标签。每个输入节点都连接到输出节点。

如果输入被展平为 D 个元素，输出为 N 个节点，则参数量为 $(D+1) \times N$，其中 +1 代表偏置项。

输出是一个 N 维向量，因此维度为 $1 \times 1 \times N$。

1.4 Leaky ReLU

Leaky ReLU（Leaky Rectified Linear Unit）是 ReLU 激活函数的一个变种，允许小的梯度值流过，防止神经元完全死亡。其公式为 $f(x) = \begin{cases} x & x > 0 \\ \alpha x & \text{otherwise} \end{cases}$，其中 α 是一个很小的常数。

Leaky ReLU 没有参数。

不改变输入的尺寸，因此输出维度与输入维度相同。

1.5 FLATTEN

Flatten 层将多维的输入展平为一维，通常用在卷积层和全连接层之间。

Flatten 层没有参数。

如果输入维度是 $H \times W \times C$，输出维度为 $1 \times 1 \times (H \times W \times C)$。

2 分析过程

1. 输入层：数据维度是 $28 \times 28 \times 3$，没有参数。
2. CONV 3-16：数据维度变为 $28 \times 28 \times 16$。参数量为 $((3 \times 3 \times 3) + 1) \times 16 = 448$。
3. Leaky ReLU：这是一个激活函数，不改变数据维度，也没有参数，维度保持 $28 \times 28 \times 16$。
4. POOL-2：数据维度变为 $14 \times 14 \times 16$，没有参数。
5. CONV 3-32：数据维度变为 $14 \times 14 \times 32$。参数量为 $((3 \times 3 \times 16) + 1) \times 32 = 4640$。
6. Leaky ReLU：同上，数据维度保持为 $14 \times 14 \times 32$，没有参数。
7. POOL-2：数据维度变为 $7 \times 7 \times 32$，没有参数。
8. FLATTEN：数据被展平为一维向量，数据维度变为 $1 \times 1 \times (7 \times 7 \times 32) = 1 \times 1 \times 1568$。
9. FC-10：数据维度变为 $1 \times 1 \times 10$。参数量为 $(1568 + 1) \times 10 = 15690$。

3 分析结果

以下是卷积神经网络各层的数据维度和参数量。

表 3.1 卷积神经网络各层的数据维度和参数量

网络层	数据维度	参数量
输入层	28×28×3	0
CONV 3-16	28×28×16	448 (weights: 432, biases: 16)
Leaky ReLU (after CONV 3-16)	28×28×16	0
POOL-2 (after CONV 3-16)	14×14×16	0
CONV 3-32	14×14×32	4640 (weights: 4608, biases: 32)
Leaky ReLU (after CONV 3-32)	14×14×32	0
POOL-2 (after CONV 3-32)	7×7×32	0
FLATTEN	1×1×1568	0
FC-10	1×1×10	15690 (weights: 15680, biases: 10)

4 结果验证

通过代码 4.1 定义 CNN 模型，并通过代码 4.2 打印 CNN 模型参数量，可以得到以下输出结果，结果验证正确。

```
conv1.weight has 432 parameters
conv1.bias has 16 parameters
conv2.weight has 4608 parameters
conv2.bias has 32 parameters
fc.weight has 15680 parameters
fc.bias has 10 parameters
total parameters: 20778
```

装 订 线

```python
class MyCNN(nn.Module):
    def __init__(self):
        super(MyCNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc = nn.Linear(in_features=7 * 7 * 32, out_features=10)

    def forward(self, x):
        x = F.leaky_relu(self.conv1(x))
        x = self.pool(x)
        x = F.leaky_relu(self.conv2(x))
        x = self.pool(x)
        x = torch.flatten(x, 1)
        x = self.fc(x)
        return x
```

代码 4.1 定义 CNN 模型代码段

```python
total_params = 0
for name, parameter in model.named_parameters():
    if not parameter.requires_grad: continue
    param = parameter.numel()
    print(f'{name} has {param} parameters')
    total_params += param
print(f'total parameters: {total_params}')
```

代码 4.2 打印 CNN 模型参数量代码段