

華中科技大學

# 实验报告

课程名称：计算机视觉

专业班级：CS2209

学    号：U202214056

姓    名：赵子昕

指导教师：刘康

报告日期：2024/1/2

计算机科学与技术学院

## 实验内容：

1. 对最后一层卷积层，依据输出特征图的神经元激活的排序，进行依次剪枝。例如：若最后一层卷积层的权重大小为  $D \times 3 \times 3 \times P$ ，输出特征图大小为  $M \times N \times P$ ，在测试数据集上对  $P$  个输出特征图的神经元激活 ( $\text{test\_dataset\_size} \times M \times N$ ) 求平均并进行排序。按激活水平由低到高，对前  $K$  个神经元权重进行剪枝， $K = 1 \text{ to } P - 1$ 。
2. 剪枝后的卷积层权重大小为  $D \times 3 \times 3 \times (P - K)$ ，测试此时神经网络分类准确率。
3. 画出最后一层卷积层（剪枝前）在整个测试数据集上的平均输出特征图（大小为  $M \times N \times P$ ）。示例如下，共  $P$  个特征图（如下图为 6 行 10 列， $P = 60$ ），每个特征图的大小为  $M \times N$ 。
4. 画出横坐标为  $K$ ，纵坐标为网络分类 accuracy 的折线图。

## 实验数据处理：

上一次实验过程由于没有仔细阅读要求没有加入卷积层，其中有些方法放到本次实验不妥，根据实际情况我进行了一些修改。测试集和训练集的选取方法基本不变，仍然为选取 6000 样本，然后进行 1:9 的组合。

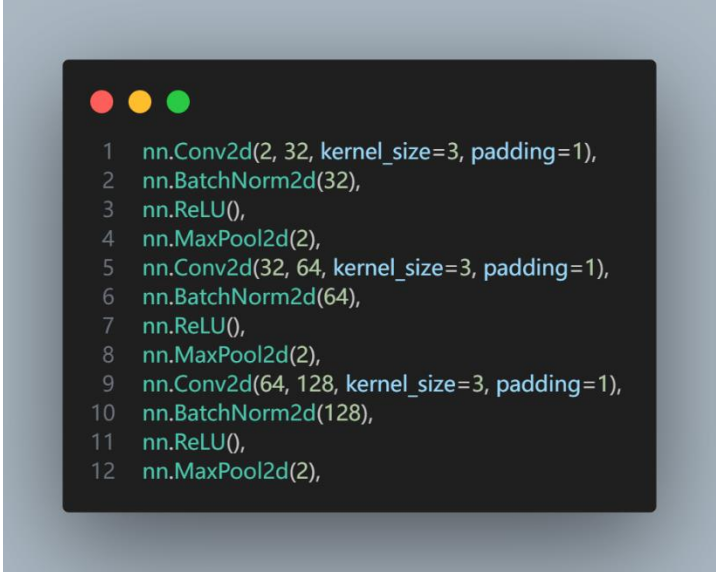
此次没有使用全为全连接层的孪生神经网络，使用了加入了三层卷积层和两层全连接层的普通卷积神经网络，所以修改了最初测试集和训练集的格式。

将两张图片堆叠起来，形成一个形状为 (2, 28, 28) 的数据结构，2 表示通道数，其中第一通道是 vec1，第二通道是 vec2。(28, 28) 是图片的宽和高。封装成 PairDataset 传递给网络进行训练。

## 神经网络构建：

在上一次的实验代码的基础上在最初加入了三层卷积层，之后再与两层全连接层进行网络构建。

首先简单介绍加入的三层卷积层。



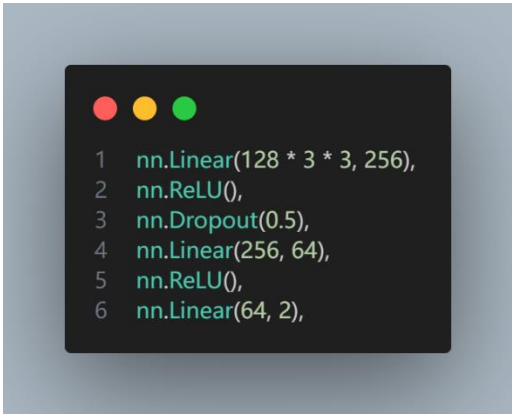
```

1  nn.Conv2d(2, 32, kernel_size=3, padding=1),
2  nn.BatchNorm2d(32),
3  nn.ReLU(),
4  nn.MaxPool2d(2),
5  nn.Conv2d(32, 64, kernel_size=3, padding=1),
6  nn.BatchNorm2d(64),
7  nn.ReLU(),
8  nn.MaxPool2d(2),
9  nn.Conv2d(64, 128, kernel_size=3, padding=1),
10 nn.BatchNorm2d(128),
11 nn.ReLU(),
12 nn.MaxPool2d(2),

```

第一层的输入通道为 2（初始为 2），输出通道为 32（卷积核个数），使用了大小 3\*3 的卷积核，并且进行了填充。随后进行归一化处理（增加训练稳定性）和引入激活函数，最后最大池化提取重要信息减少计算量。后面两层同理。

接下来是全连接层。



```

1  nn.Linear(128 * 3 * 3, 256),
2  nn.ReLU(),
3  nn.Dropout(0.5),
4  nn.Linear(256, 64),
5  nn.ReLU(),
6  nn.Linear(64, 2),

```

全连接层第一层先将特征图展平为二维张量，再通过 ReLU 激活后传入第二层，最后输出为 2 类。

### 剪枝过程解释：

这次的剪枝是非常简单的剪枝，没有采用比较复杂的混合方法，而是直接采用了简单粗暴的权重，在测试数上对 P 个输出特征图的神经元激活求平均并进行排序，减去前若干个神经元，保留后若干个神经元，再进行准确率测试。



```
1 def prune_conv_layer(model, test_loader, device, prune_rate=0.1):
2     model.eval()
3
4     activations = []
5     def hook_fn(module, input, output):
6         activations.append(output.detach())
7     hook = model.conv_layers[-4].register_forward_hook(hook_fn)
8     for images, _ in test_loader:
9         images = images.to(device)
10        _ = model(images)
11    hook.remove()
12    activations = torch.cat(activations, dim=0)
13    mean_activations = torch.abs(activations.mean(dim=(0, 2, 3)))
14    _, indices = torch.sort(mean_activations)
15
16    _, indices = torch.sort(mean_activations)
17    prune_count = int(len(indices) * prune_rate)
18    prune_indices = indices[:prune_count]
19
20    last_conv_layer = model.conv_layers[-4]
21    weight = last_conv_layer.weight.data
22    weight[prune_indices] = 0
23
24    print(f'Pruned {prune_count} channels out of {len(indices)}')
25    return model
```

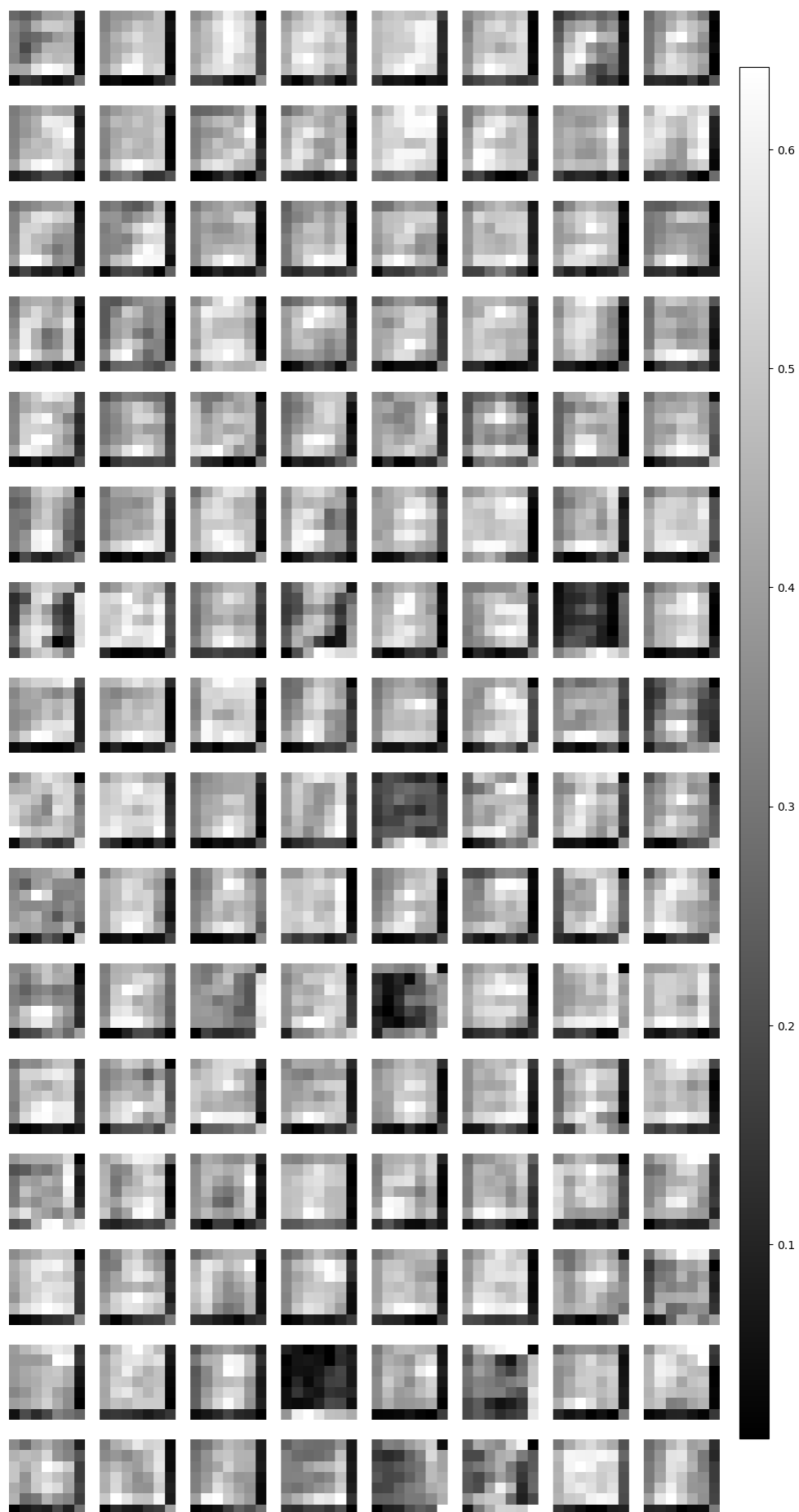
为了最后对剪枝情况进行评估（后续绘制热力图），在卷积层最后一层还注册了一个 hook 函数暂存状态，暂存之后清除防止干扰训练。

### 训练过程解释：

训练过程与实验二基本一致。

### 剪枝相关结果解释：

最后一层卷积层（剪枝前）在整个测试集上的平均特征图（经过了归一化处理）：



最后一共有 128 个卷积图，容易观察发现有若干卷积图近乎全黑，说明这些神经元对于训练几乎没有什么影响，剪枝的时候可以优先去除。

下面是剪枝去除的神经元根据 accuracy 绘制的曲线图，可以明显观察到，当剪枝个数非常少的时候，几乎不影响 accuracy（因为减去的都是一些不重要的神经元）；当减去的个数过多的时候，accuracy 下降非常快，这是因为减去了一些参与的重要的神经元。

最后有一些波动，在思考之后注意到生成的数据是不均衡的（0.9 的数据集是不同的，0.1 是相同的），可能某些神经元侧重将数据集标为相同（就是无论输入什么它都有很大概率输出不同），这些神经元保留了下来导致最后有波动。

