华中科技大学

计算机视觉实验报告

基于剪枝算法的深度神经网络压缩

专　　业：	计算机科学与技术
班　　级：	CS2005 班
学　　号：	U202090063
姓　　名：	董玲晶
指导教师：	刘　康

分数	
教师签名	

2023 年 4 月 15 日

1 任务要求

对实验二构建的 CIFAR-10 数据集分类神经网络进行权重剪枝实现模型压缩。

1. 实验步骤

1. 可对最后一层卷积层，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（$test\_dataset\_size \times M \times N$）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（可将待剪枝的神经元权重、偏置设为 0，即相当于神经元剪枝而不用改变网络架构），$K = 1\ to\ P - 1$

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。

2. 报告要求

1. 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$M \times N$

2. 画出横坐标为K，纵坐标为网络分类 *accuracy* 的折线图

3. 实验报告包含网络设计、实验结果图，以及必要的分析等。

2 实验

2.1 模型回顾

本次实验是在实验二构建的模型（下称 Model）基础上进行，下面对 Model 的结构和性能进行回顾。

1. 模型性能

Model 实现了在十类测试集上的平均准确率为 90.58。

2. 模型结构

Model 的最后一层卷积层其实是残差块 ResidualBlock 中的最后一层特征提取卷积层以及原始输入变换通道（对应表格中的 down_sample）中所用到的卷积层。对应的因此权重大小分别为 128×1×1×256 和 128×3×3×256，输出的特

征图的大小为 256×11×11。

表 1 残差块结构 ResidualBlock

	代码	备注
… …		
3	nn.Conv2d(out, out, 1)	out 个out×1×1 的卷积核，padding=1
4	nn.BatchNorm2d(out)	out 个通道批量归一化
5	out += down_sample(x)	将以上输出加上输入
备注：设输入为 x，每一层为 out		

2.2 剪枝

根据实验要求，要对最后一层卷积层的输出特征图求平均并排序，根据剪枝比率 K（即代码 1 中的 self.k_cut_ratio）来将前对应数目的权重设置为 0，具体代码如 1 所示

代码 1： 在 ResNet 网络结构中添加剪枝操作

```
out = self.feature2(out)
out_temp = self.res2(out)

activations = out_temp.mean(dim=(0))
each_feature_mean = activations.mean(dim=2).mean(dim=1)
sorted_indices = torch.argsort(each_feature_mean)
sorted_activations =
  torch.index_select(activations, dim=0, index=sorted_indices)

weight_last_layer = self.res2.feature2[0].weight
bias_last_layer = self.res2.feature2[0].bias
weight_res_layer = self.res2.down_sample[0].weight
bias_res_layer = self.res2.down_sample[0].bias
weight_last_layer[sorted_indices[: self.k_cut_ratio]] = 0
bias_last_layer[sorted_indices[: self.k_cut_ratio]] = 0
weight_res_layer[sorted_indices[: self.k_cut_ratio]] = 0
bias_res_layer[sorted_indices[: self.k_cut_ratio]] = 0

self.res2.feature2[0].weight = weight_last_layer
self.res2.feature2[0].bias = bias_last_layer
self.res2.down_sample[0].weight = weight_res_layer
self.res2.down_sample[0].bias = bias_res_layer
out = self.res2(out)

… …
return out, activations, sorted_activations
```

代码 1 实现了如下操作：首先将上次卷积操作得到的结果传入残差块中，计算得到输出 out_temp，使用 out_temp 计算出每个神经元的激活值的均值 activations，并按照从小到大的顺序对这些均值进行排序，得到一个排序的下标列表 sorted_indices。之后，将 out_temp 根据 sorted_indices 进行相应的排序，得到排序后的激活值 sorted_activations。然后，根据排序后的下标列表 sorted_indices，将残差块中的权重和偏置的部分元素设置为 0，从而实现神经元的剪枝。具体而言，将卷积层中的权重和偏置 weight_last_layer 和 bias_last_layer，以及残差块中的权重和偏置 weight_res_layer 和 bias_res_layer 的前 k_cut_ratio 个元素设置为 0，实现参数设置并完成剪枝。最后就是应用更新过后的参数完成卷积操作。

2.3 输出特征图绘制

在 2.2 中所描述的代码 1 是在网络 ResNet 类的 forward ()方法中实现的，当我们在 test 中调用 model 得到预测输出时会一起得到我们在代码 1 一行返回的 activations 和 sorted_activations。我们将每个 batch_idx 里得到的这两个参数求平均，作为本次测试的最后一层卷积层的平均输出特征图和排序后的输出特征图，并调用 draw_feature_pics()函数绘制，如代码 2 所示。

代码 2:　绘制输出特征图

```
def draw_feature_pics(activations, title):
  column_num = 16
  row_num = int(activations.shape[0] / column_num)
  fig, axs = plt.subplots(row_num, column_num,
             figsize=(column_num*2, row_num*2), dpi=600)
  for i in range(row_num):
    for j in range(column_num):
      idx = i * column_num + j
      if idx >= activations.shape[0]: break
      axs[i,j].imshow(
            activations[idx].cpu().numpy(), cmap='gray')
      axs[i,j].set_xticks([])
      axs[i,j].set_yticks([])
  fig.savefig(f'./lab3/figures/{title}.jpg')
  plt.close()
  return
```

剪枝前的原始输出特征图和排序后的输出特征图分别如图 1 和图 2 所示，大小如前面所说为 256×11×11。综合来看，每个神经元的激活程度都较高。

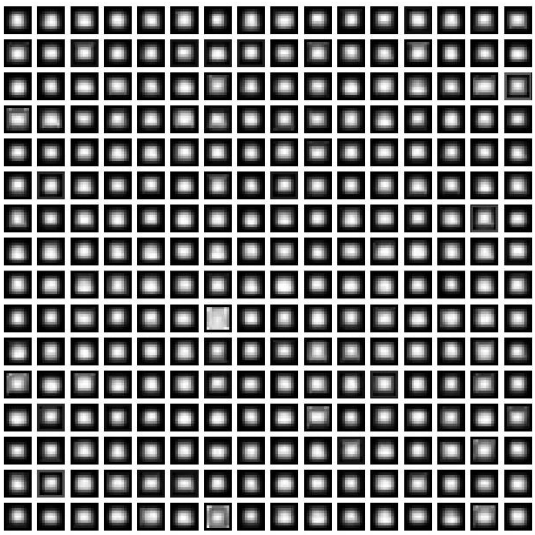

图 1：原始输出特征图

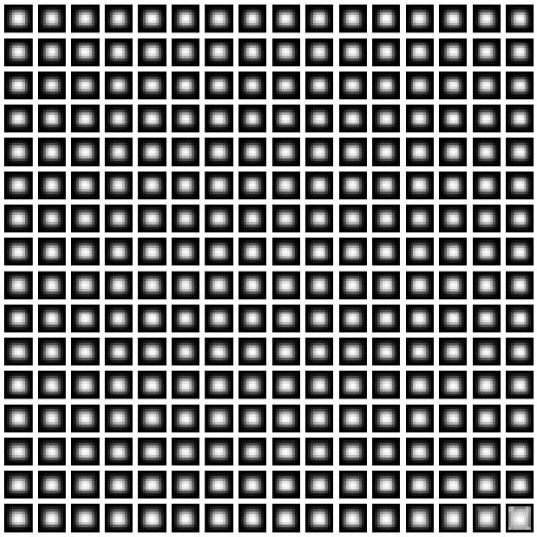

图 2：排序后的输出特征图

2.4 准确率变化图绘制

每一轮改变剪枝比例，从 0 循环到 1，步长为 0.01。将得每一轮得到的准确率以 5 为间隔采样绘制成折现图，如图 3 所示，可以看到只在一次超过了原始准确率（90.58），经历了短暂的上升又下降，且随着剪枝比例增大准确率呈现下降趋势。而按原始间隔绘制剪枝比例 0~0.1 的准确率变化图查看更精确的变化，可以观察到准确率在原始准确率附近徘徊一会上升了之后就开始逐步下降，如图 4 所示。

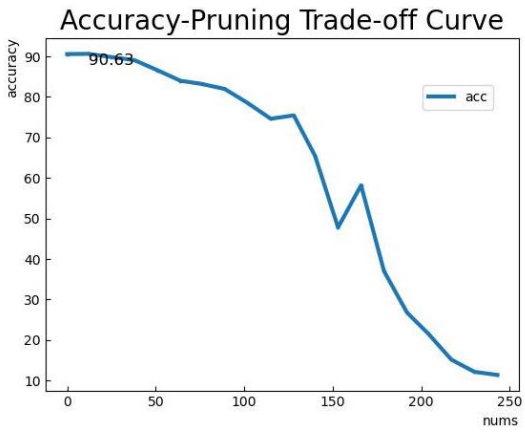

图 3：准确率随简直比例变化图（1）

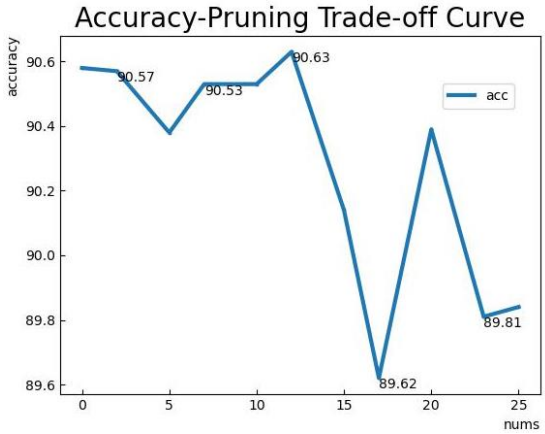

图 4：准确率随简直比例变化图（2）

对于此结果的分析将在下一个章节给出。

3 结果分析

从图 3 和图 4 我们可以看到当剪枝比例较小时剪枝对模型准确率的提升效果并不大，只是在一开始从 90.58 提升到了 90.63。其原因可能是因为本身激活水平就比较高，从图 1 和图 2 的输出特征图我们也可以看到，每张小特征图除去外圈的 padding 是全黑色的，内圈正方形正方形的颜色都很白，说明激活程度很高。对 256 个特征都求平均，使得输出特征张量的形状由 torch.size([256, 11, 11])变为 torch.size([256])以便打印输出。打印结果如图 5 所示，可以发现对于每个小图哪怕和外圈 padding（即 0）平均后激活程度还是很高，计算求得最小的为 0.7926，最大的为 1.9166，进一步佐证了激活程度很高。

```
tensor([1.1277, 1.2362, 1.1416, 1.2725, 1.3101, 1.3423, 0.8522, 1.2242, 1.1430,
        1.0686, 1.1704, 1.2639, 0.9999, 1.1497, 1.0707, 1.2141, 1.2278, 1.2702,
        1.0611, 0.9192, 1.1042, 0.8996, 1.0800, 0.9992, 1.1148, 1.3636, 0.9205,
        1.0030, 1.4723, 1.1708, 1.2768, 1.0478, 1.1698, 1.0783, 1.3109, 1.1643,
        1.0459, 0.8709, 1.1561, 0.7926, 1.3907, 1.1029, 1.0750, 1.0532, 1.3739,
        1.1213, 1.3879, 1.3990, 1.2476, 1.3848, 1.1097, 1.1017, 1.0979, 1.2925,
        1.2350, 1.4895, 1.1417, 1.0274, 1.1186, 1.1376, 0.9790, 1.2476, 0.9502,
        1.0299, 0.9586, 0.9265, 1.1295, 1.3855, 1.3632, 1.2125, 1.1509, 1.1897,
        1.1093, 1.2119, 1.2206, 1.0685, 1.1370, 1.3380, 0.9542, 1.2107, 1.1119,
        1.1957, 1.6102, 1.2509, 0.8043, 1.0471, 1.3034, 1.0255, 0.9513, 1.3443,
        0.9466, 1.2426, 1.2234, 0.9930, 1.0081, 1.1361, 1.2671, 1.0846, 1.2852,
        0.9119, 1.0502, 1.1441, 1.2635, 0.9166, 1.1351, 1.0802, 1.1776, 0.9628,
        0.9779, 1.3732, 1.1316, 0.9938, 1.2642, 1.1056, 0.9917, 1.1341, 1.3201,
        0.8616, 0.8991, 0.8118, 0.9220, 1.2650, 0.9941, 1.2749, 1.1402, 1.0839,
        1.1220, 1.0745, 1.0523, 1.2756, 1.5026, 1.1948, 1.3742, 1.1835, 1.0250,
        1.2919, 1.4918, 1.0342, 1.0437, 1.0783, 1.0567, 1.0003, 1.1590, 0.9972,
        0.8624, 1.0335, 1.2198, 1.0793, 0.9186, 1.3849, 1.9166, 1.1485, 1.0904,
        1.3815, 1.3996, 1.2584, 1.2167, 1.1700, 0.9987, 1.1773, 1.4555, 0.9887,
        1.0793, 0.8515, 1.2126, 1.1520, 1.0068, 1.1470, 1.1734, 1.1674, 0.9598,
        1.3722, 1.0587, 1.2950, 1.4247, 0.9940, 1.4352, 1.2446, 0.9076, 1.4992,
        0.9417, 1.1558, 1.1256, 1.3752])
```

图 5：对原始输出特征图每个通道求平均

而随着剪枝比例增大，网络失去了过多的信息，无法保持原有的准确率，因此准确率开始逐步下降。

4 心得与体会

本次实验通过模拟简单的剪枝操作，学会了对神经网络结构内部权重的各项操作，加强了我对神经网络内部参数形式的理解；也通过各种参考资料和亲身实践，了解到了剪枝通过剪去多余的神经元可以增强网络的泛化能力和稳定性并提升准确率，但过渡剪枝则会造成不好的影响。同时我也学会了如何可视化神经网络的特征图以及通过结合特征图来分析剪枝效果，也许以后会使用到这个技巧并一键继承这个写过的绘图代码减轻工作量 orz。