图像分类问题项目报告

——李昭阳 2021013445

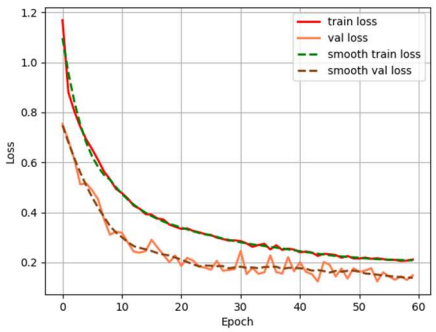
一、 设计深度学习算法

本项目改编了 Vgg16 卷积神经网络模型，使用 Pytorch 机器学习框架完成了此项目代码的编写。由于输入图像大小为 150x150 而非原模型推荐的 224x224，因此对卷积层与池化层进行一定程度的删剪，整体网络结构如下。

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=12544, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=6, bias=True)
  )
)
```

整体模型分为三个部分，即卷积抽取、平均池化、图形分类。卷积抽取部分通过多层的卷积、最大池化层提取图像的基本信息；平均池化使得 feature map 更具有泛化性，同时限定分类部分的输入大小；分类部分通过全连接层完成简单的分类，同时用随机丢弃的正则化技术防止过拟合，保证了神经网络的鲁棒性。

在训练过程中，通过观察训练集与验证集的 Loss（本任务设定为交叉熵）可以测试当前训练效果，得到如下训练过程图，观察到在 60 个 Epochs 时，两集合 Loss 均趋于收敛。



在测试最终模型时，我采用了准确率、召回率、精确率、F1 分数来评估模型的分类效果，得到如下结果。

```
准确率: 0.8964
召回率: 0.8975
精确率: 0.8985
F1分数: 0.8977
```

准确率是模型预测正确的样本数占总样本数的比例；召回率是指在所有实际为正的样本中，被模型正确识别为正的样本所占的比例，它反映了模型对正类样本的敏感度；精确率是指在所有被模型预测为正的样本中，实际为正的样本所占的比例，它反映了模型预测结果的准确性；F1 分数是精确率和召回率的调和平均值，是对这两者的综合考量。当精确率和召回率有一个显著较低时，F1 分数也会较低。

二、 模型优化

由于训练集数据并不充足，因此进行一些数据增广的方法，以获得一些已有数据集的变体训练例子。对原图像进行了尺度变换（拉伸压缩）、随机旋转、随机遮盖部分图像像素点、随机对图像颜色进行变换。变换代码如下。

```python
# 计算新的宽高比并且进行随机的扭曲
new_ar = image_w / image_h * self.rand(1 - jitter, 1 + jitter) / self.rand(1 - jitter, 1 + jitter)
scale = self.rand(.25, 2)
if new_ar < 1:
    new_h = int(scale * h)
    new_w = int(new_h * new_ar)
else:
    new_w = int(scale * w)
    new_h = int(new_w / new_ar)
image = image.resize((new_w, new_h), Image.BICUBIC)
```

```python
# 随机翻转
flip_lr = self.rand() < .5
flip_tb = self.rand() < .5
if flip_lr:
    image = image.transpose(Image.FLIP_LEFT_RIGHT)
if flip_tb:
    image = image.transpose(Image.FLIP_TOP_BOTTOM)
```

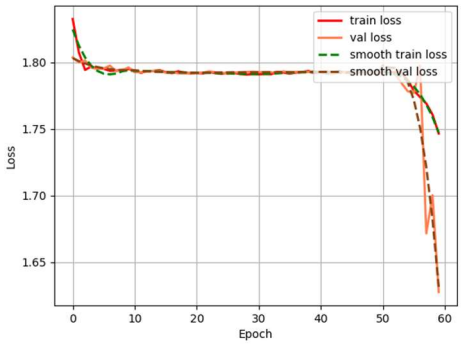
```python
# 生成随机噪声
noise = self.rand() < .5
if noise:
    # 生成一个与图像大小相同的随机二值遮罩
    mask = np.random.rand(data.shape[0], data.shape[1]) > 0.1
    # 将遮罩应用于图像
    data[np.where(mask==False)] = 0
```

```python
# 将图像映射到HSV色域，随机变色，色相（hue）、饱和度（sat）和亮度（val）
r = np.random.uniform(-1, 1, 3) * [hue, sat, val] + 1
hue, sat, val = cv2.split(cv2.cvtColor(data, cv2.COLOR_RGB2HSV))

x = np.arange(0, 256, dtype=r.dtype)
rand_hue = ((x * r[0]) % 180).astype(data.dtype)
rand_sat = np.clip(x * r[1], 0, 255).astype(data.dtype)
rand_val = np.clip(x * r[2], 0, 255).astype(data.dtype)

data = cv2.merge((cv2.LUT(hue, rand_hue), cv2.LUT(sat, rand_sat), cv2.LUT(val, rand_val)))
```

在训练过程中，我调节了模型 batchsize、学习率等超参数，训练后可发现，batchsize = 16 时为最佳，得到评估数据如下。

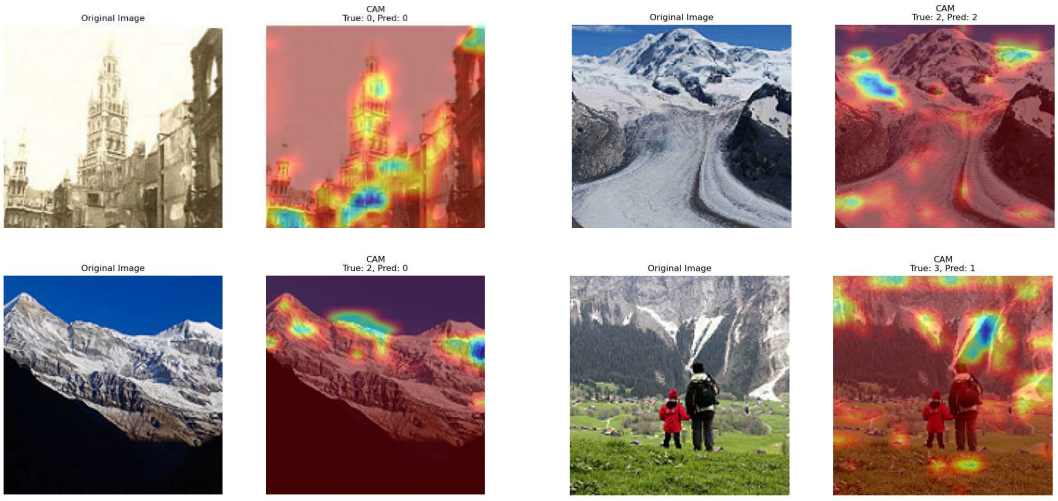
若 batchsize 过小，虽占用内存小，但训练计算效率低、收敛不稳定、训练时间长；若 batchsize 过大，则内存消耗高，学习进度慢。同理，学习率也存在一个适宜区间，过高的学习率虽然可以在起初加速收敛，但有可能震荡或无法收敛；过低的学习率虽然可以细致搜索最优解，但是可能陷入局部最优。

此外，由于初始代码使用了 Adam 的优化器，因此我尝试使用 Sgd 的优化器进行训练，保证控制变量，故未调节超参数，结果如下。



可以看到，由于未调节超参数，学习率过低，搜索速度极慢，且最终未能得到一个合格的结果。因此我认为，不同优化器有不同的超参数区间，应当根据不同任务合理选取。

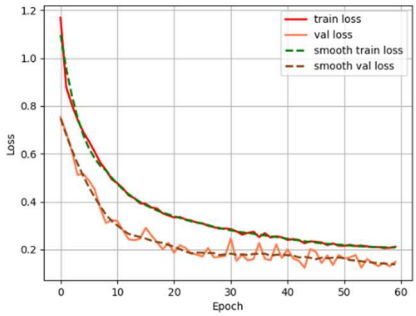
我还利用 CAM 可视化方法对训练过程进行可视化，下图中红色区域为“更受到关注的区域”，上部为两张预测成功案例，下部为预测失败案例。



在预测成功案例中，可以观察到模型关注了近处建筑特征、山坡等要素，成功进行了

分类；但在失败案例中，左图我猜测是损失山脊关键信息，与建筑物难以区分导致判断错误，而右图则是完全忽略冰川信息，只关注远端的森林导致判断错误。

最后，我设计定量实验分析样本不同类别之间的分类是否会彼此促进，即删除分类 1 和 2 的数据后再进行训练。从图中可以观察到，或许 1、2 类的图片会对其他组样本产生干扰，即森林和高山场景会影响其他场景，这在上文提到的 CAM 中也有体现——本应是冰川的图片被预测为森林，本应是高山的场景被预测为建筑。



三、 总结

本项目中，我学习了使用 Pytorch 进行神经网络的搭建，加深了对深度神经网络的理解，学习了可视化深度网络训练过程的方法。在搭建前端和编写算法的过程中，我遇到了许多不可预期的问题，包括网络收敛速度慢、结果不理想，但是在查阅资料和不懈努力最终得以解决。在以后的学习实验中，我会勤加练习，注重理解，同时也会在以后的代码编写过程中更加谨慎，以保证算法正确稳定。谢谢助教老师耐心阅读！