计算摄影学作业报告
Lab 5 —— 深度学习上色

Jessie Peng

2019/04/28

1 实验内容

本实验需要用深度学习的方法给黑白图片上色。

2 实验环境

编程语言：Python
开发环境：PyCharm 2018.3
操作系统：macOS 10.14.1 (18B75)
深度学习框架：PyTorch
主要的库及版本：

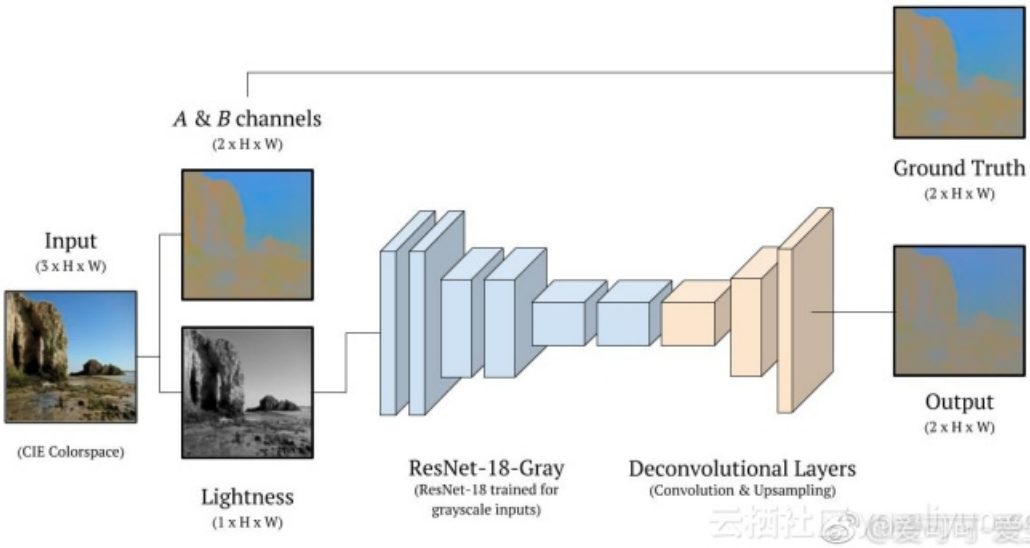
matplotlib==3.0.2
torch==0.4.1
scikit_image==0.13.1
numpy==1.14.3
torchvision==0.2.1

3 实验原理

3.1 总体原理概述

图像上色本质上就是要建立一个从黑白图像（单通道）到彩色图像（三通道）的映射 。如果使用 LAB 色彩空间（L：亮度），则简化为从 L 通道到 A 和 B 两个通道的映射。

采用迁移学习的方法来构造神经网络，模型的前半部分借用了 ResNet 网络模型，但稍作改动：修改网络的第一层以便接收灰度图而不是彩色图作为输入；切断了第六层后面的网络结构。

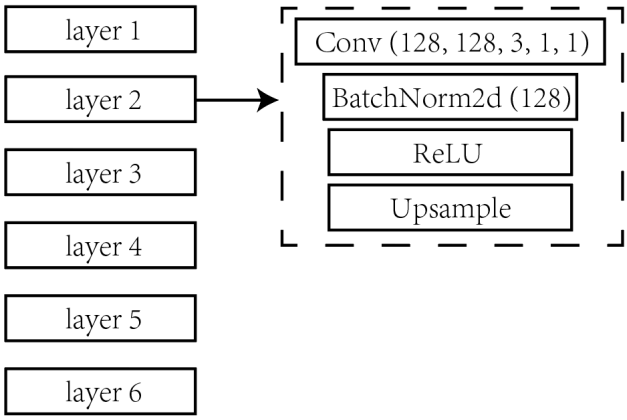


模型的后半部分是上采样的过程，采用配合 5 个卷积层进行 5 次上采样。如果将前半部分 ResNet 整体作为一层，则模型一共有 6 层。

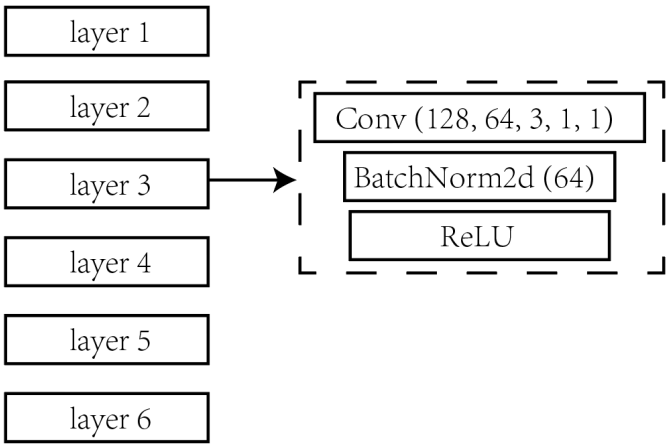
3.2 具体网络结构

ResNet 有现成的代码可以调用，本次实验的代码框架中也已经将模型的第一层实现好了，我们只需要从第二层开始。

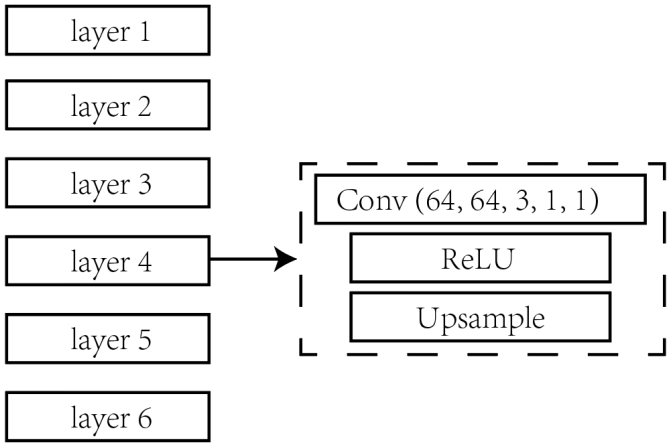
第二层做卷积然后进行上采样：



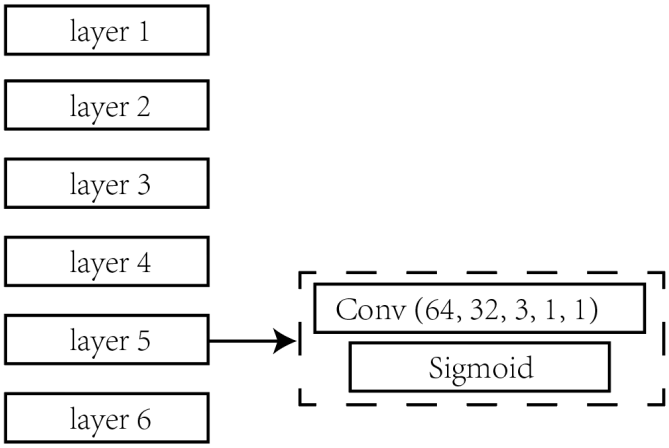
第三层只做卷积：



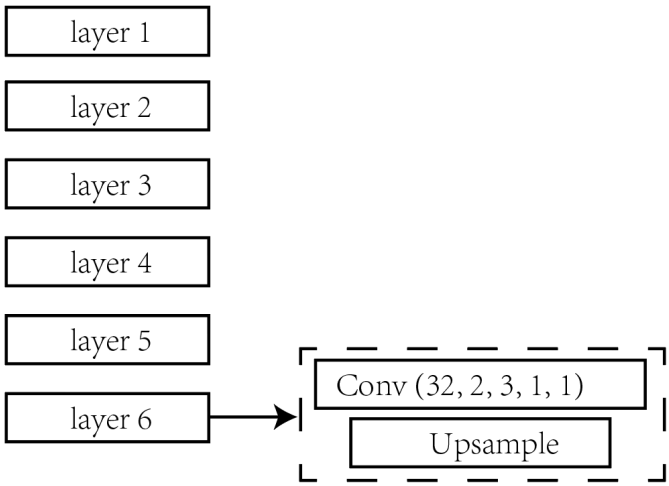
第四层进行第二次上采样：



第五层改用 Sigmoid 激活，可以使像素值映射到 0 和 1 之间：



最后一层再做一次上采样：



4 实现过程与代码分析

本次实验的代码框架已经包含了大部分代码，只需要自己补充完整上述最后 5 层的网络相关代码。

4.1 定义 operators

定义 layer 2 至 layer 6 的网络结构，这里我本来使用 nn.Upsample()函数来进行上采样，但是后来发现该函数过时了(deprecated)，于是换成了 F.interpolate()函数，因此网络的结构定义中就看不到上采样的步骤了。

```
def __init__(self):
    super(Net, self).__init__()

    # resnet that deals with gray images
    resnet = models.resnet18(num_classes=365)
    resnet.conv1.weight = nn.Parameter(torch.Tensor(64, 1, 7, 7))

    # layer1, output_shape: [None, 128, img_h, img_w]
    self.layer1 = nn.Sequential(*list(resnet.children())[0:6])

    # define colorization operators

    # layer 2
    self.conv2 = nn.Conv2d(128, 128, kernel_size=3, stride=1, padding=1)
    self.bn2 = nn.BatchNorm2d(128)
    self.act2 = nn.ReLU()

    # layer 3
    self.conv3 = nn.Conv2d(128, 64, kernel_size=3, stride=1, padding=1)
    self.bn3 = nn.BatchNorm2d(64)
    self.act3 = nn.ReLU()

    # layer 4
    self.conv4 = nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1)
    self.act4 = nn.ReLU()

    # layer 5
```

```
    self.conv5 = nn.Conv2d(64, 32, kernel_size=3, stride=1, padding=1)
    self.act5 = nn.Sigmoid()

    # layer 6
    self.conv6 = nn.Conv2d(32, 2, kernel_size=3, stride=1, padding=1)
```

4.2 使用 operators

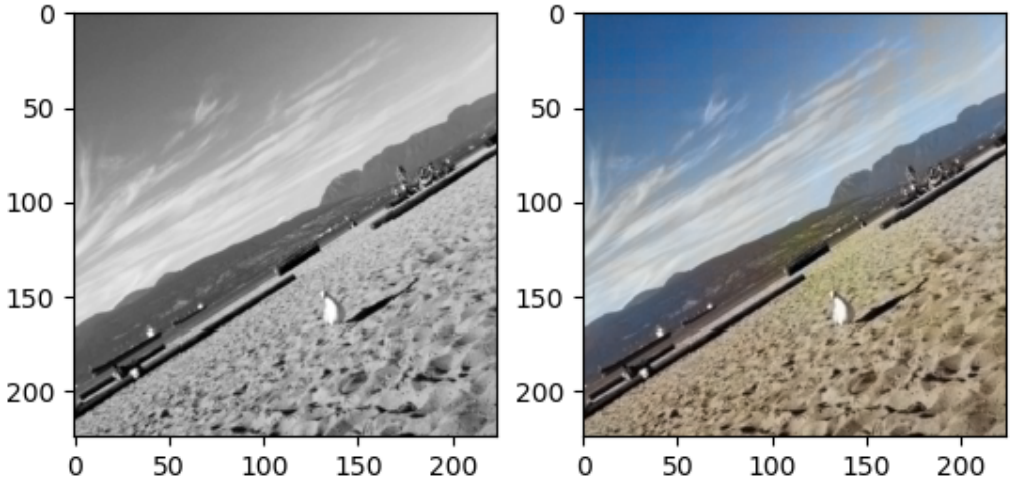
使用定义好的网络进行正向计算，这里要加入上采样：

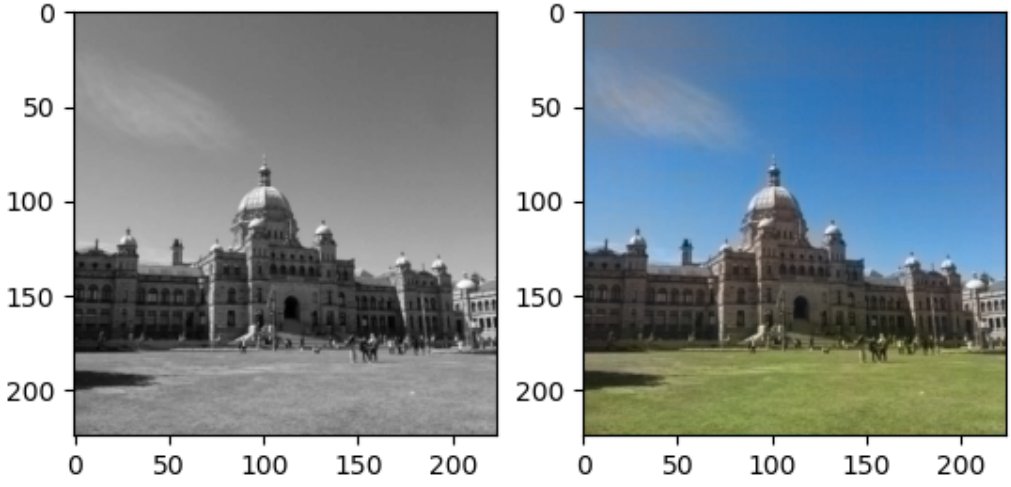
```
def forward(self, l_input):
    """
    l_input: [None, 1, img_h, img_w]
    """
    # layer 1
    features = self.layer1(l_input)

    # set colorization operations

    # layer 2
    x = F.interpolate(self.act2(self.bn2(self.conv2(features))), scale_factor=2)
    # layer 3
    x = self.act3(self.bn3(self.conv3(x)))
    # layer 4
    x = F.interpolate(self.act4(self.conv4(x)), scale_factor=2)
    # layer 5
    x = self.act5(self.conv5(x))
    # layer 6
    x = F.interpolate(self.conv6(x), scale_factor=2)
    return x
```

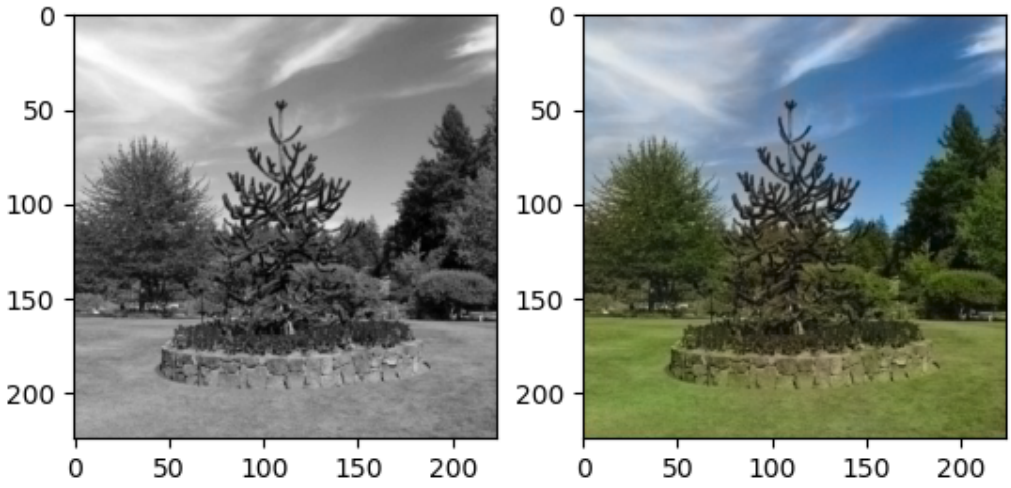
5 结果分析与实验总结

5.1 上色效果

使用深度学习给黑白图像上色效果如下：


0
50
100
150
200
0
50
100
150
200
0
50
100
150
200
0
50
100
150
200


0
50
100
150
200
0
50
100
150
200
0
50
100
150
200
0
50
100
150
200



5.2 问题与解决

本次实验比较简单，因为代码框架已经比较完整了，主要遇到以下几个小问题：

- 从 warning 得知上采样函数 nn.Upsample()过时了，于是换成了 F.Interpolate()函数，因此在 operators 的定义中就看不到上采样层了，需要在 forward 的时候再进行上采样。
- 刚开始运行的时候，由于不是在 GPU 上运行而报错，在 demo.py 中调用 torch.load()函数的时候，加上参数 map_location='cpu'则可以解决该问题。
- 该模型并不是对所有图片上色效果都很好，一开始我使用一些红色或紫色较多的图片，效果很不好，原因是这种图片与训练集的整体风格不匹配，后来使用了蓝天绿草黄沙这类图片，上色效果就比较好了。

6 参考文献

课程网站：http://www.cad.zju.edu.cn/home/gfzhang/course/computational-photography/lab5-deep-learning/deep-learning.html
老旧黑白片修复机——使用卷积神经网络图像自动着色实战：
https://yq.aliyun.com/articles/599156