计算机科学与技术学院自然语言处理课程实验报告

实验题目：Style Transfer（风格迁移）		学号：201600122057
日期：2019.4.11	班级： 智能 16	姓名： 贝仕成
Email：337263318@qq.com		

实验目的：

在本次实验中，将通过练习完成一个简单的 neural style transfer 网络。

- 学习通过 VGG 提取图像特征;
- 明白 neural style transfer 的基本原理以及流程。

实验软件和硬件环境：

Python 3.5.6

Jupyter notebook 5.0.0

神舟战神 Z7M-KP7S1

NVIDIA GTX1050Ti

实验原理和方法：

调用 VGGnet（本质卷积神经网络）模型来训练，将一张图片的内容和另一张图片的风格融合起来组成一张新图。

实验步骤：（不要求罗列完整源代码）

1.读取图像

```
def load_image(image_path, transforms=None, max_size=None, shape=None):
    image = Image.open(image_path)
    image_size = image.size

    if max_size is not None:
        #获取图像size，为sequence
        image_size = image.size
        #转化为float的array
        size = np.array(image_size).astype(float)
        size = max_size / size * size
        image = image.resize(size.astype(int), Image.ANTIALIAS)

    if shape is not None:
        image = image.resize(shape, Image.LANCZOS) #LANCZOS也是一种插值方法

    #必须提供transform.ToTensor，转化为4D Tensor
    if transforms is not None:
        image = transforms(image).unsqueeze(0)

    #是否拷贝到GPU
    return image.type(dtype)
```

参数值	含义
Image.NEAREST	低质量
Image.BILINEAR	双线性
Image.BICUBIC	三次样条插值
Image.ANTIALIAS	高质量

2. 用 VGG19 提取特征

```
class VGGNet(nn.Module):
    def __init__(self):
        super(VGGNet, self).__init__()
        self.select = ['0', '5', '10', '19', '28']
        self.vgg19 = models.vgg19(pretrained = True).features

    def forward(self, x):
        features = []
        #name类型为str，x为Variable
        for name, layer in self.vgg19._modules.items():
            x = layer(x)
            if name in self.select:
                features.append(x)
        return features
```

3. 构建模型，计算 loss

```
target_features = vgg(target)
content_features = vgg(Variable(content))
style_features = vgg(Variable(style))
content_loss = 0.0
style_loss = 0.0
for f1,f2,f3 in zip(target_features, content_features, style_features):
    content_loss += torch.mean((f1 - f2)**2)
    n, c, h, w = f1.size()
#将特征reshape成二维矩阵相乘，求gram矩阵
    f1 = f1.view(c, h * w)
    f3 = f3.view(c, h * w)

    f1 = torch.mm(f1, f1.t())
    f3 = torch.mm(f3, f3.t())

#计算style_loss
    style_loss += torch.mean((f1 - f3)**2) / (c * h * w)

loss = content_loss + style_loss
```

4. 利用梯度下降训练

```
optimizer.zero_grad()
loss.backward()
optimizer.step()
```

实验结果：

```
(python35) g:\warehouse\DL2019\hw4>python neural_style_transfer.py
Namespace(content='G:/warehouse/DL2019/hw4/content.jpg', log_step=10, lr=0.003, max_size=400, sample_step=100, style='G:/warehouse/DL2019/hw4/style.jpg', style_weight=100, total_step=5000)
Step [10/5000], Content Loss: 3.4430, Style Loss: 1021.5614
Step [20/5000], Content Loss: 9.0620, Style Loss: 876.9575
Step [30/5000], Content Loss: 13.0209, Style Loss: 781.0850
Step [40/5000], Content Loss: 15.7930, Style Loss: 712.8851
Step [50/5000], Content Loss: 17.8563, Style Loss: 660.9979
Step [60/5000], Content Loss: 19.4507, Style Loss: 619.4810
Step [70/5000], Content Loss: 20.7336, Style Loss: 584.7196
Step [80/5000], Content Loss: 21.8017, Style Loss: 554.5703
Step [90/5000], Content Loss: 22.6984, Style Loss: 528.0972
Step [100/5000], Content Loss: 23.4822, Style Loss: 504.3504
Step [110/5000], Content Loss: 24.1770, Style Loss: 482.7672
Step [120/5000], Content Loss: 24.7975, Style Loss: 463.0529
Step [130/5000], Content Loss: 25.3474, Style Loss: 444.8445
Step [140/5000], Content Loss: 25.8506, Style Loss: 427.9393
Step [150/5000], Content Loss: 26.3067, Style Loss: 412.1567
Step [160/5000], Content Loss: 26.7224, Style Loss: 397.3470
Step [170/5000], Content Loss: 27.1058, Style Loss: 383.4302
Step [180/5000], Content Loss: 27.4675, Style Loss: 370.2701
Step [190/5000], Content Loss: 27.8009, Style Loss: 357.8326
Step [200/5000], Content Loss: 28.1092, Style Loss: 346.0585
Step [210/5000], Content Loss: 28.4022, Style Loss: 334.8712
```

content.jpg

style.jpg

每训练一百次存下转换后的图片：

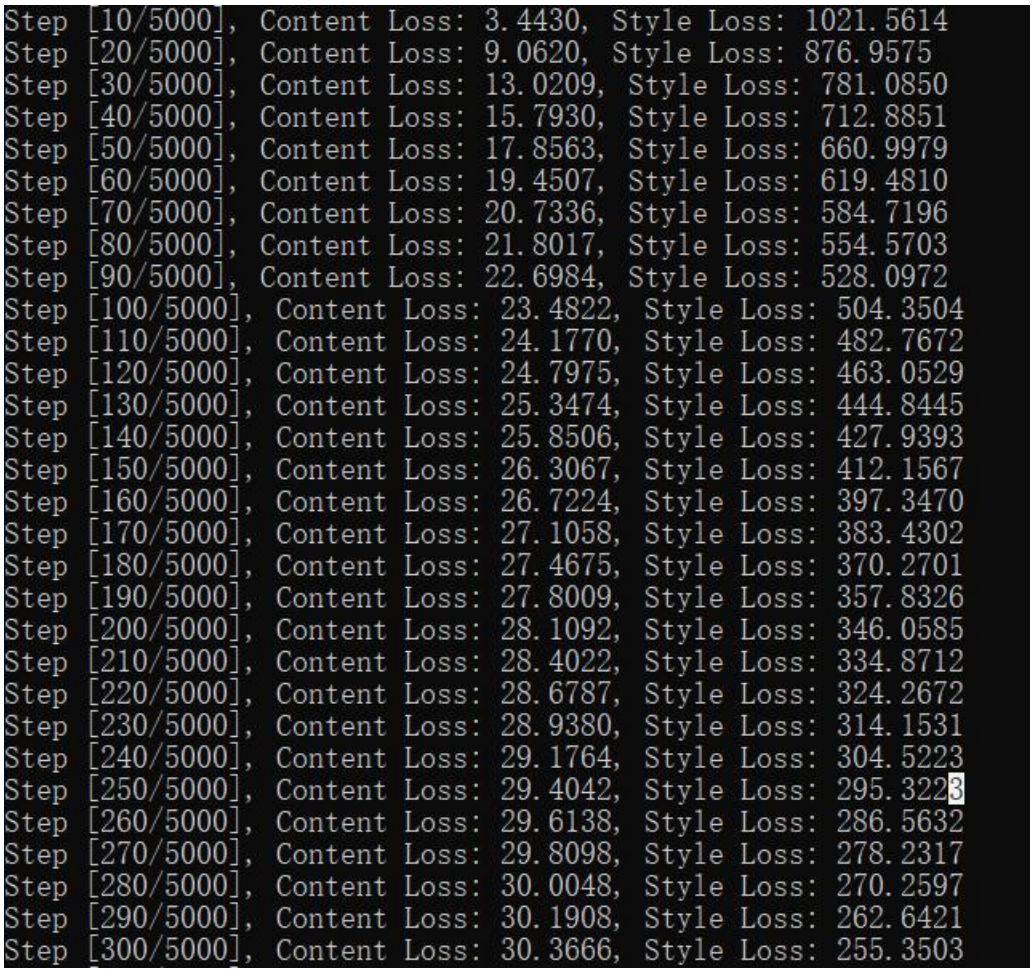
训练 100 次

训练 500 次

训练 2000 次

训练 5000 次

随着训练次数的增加，图片风格确实在向 style.jpg 方向靠拢。



从图片上来看，刚开始 content_loss 一直在缓慢上升，而 style_loss 则在明显下降。

```
Step [3770/5000], Content Loss: 35.2789, Style Loss: 22.5463
Step [3780/5000], Content Loss: 35.3054, Style Loss: 22.4559
Step [3790/5000], Content Loss: 35.3036, Style Loss: 22.3899
Step [3800/5000], Content Loss: 35.3052, Style Loss: 22.3229
Step [3810/5000], Content Loss: 35.2893, Style Loss: 22.2780
Step [3820/5000], Content Loss: 35.2620, Style Loss: 22.2398
Step [3830/5000], Content Loss: 35.2553, Style Loss: 22.1808
Step [3840/5000], Content Loss: 35.2563, Style Loss: 22.1180
Step [3850/5000], Content Loss: 35.2766, Style Loss: 22.0380
Step [3860/5000], Content Loss: 35.2934, Style Loss: 21.9567
Step [3870/5000], Content Loss: 35.3176, Style Loss: 21.8678
Step [3880/5000], Content Loss: 35.3287, Style Loss: 21.7928
Step [3890/5000], Content Loss: 35.3159, Style Loss: 21.7442
Step [3900/5000], Content Loss: 35.2668, Style Loss: 21.7336
Step [3910/5000], Content Loss: 35.2132, Style Loss: 21.7366
Step [3920/5000], Content Loss: 35.1714, Style Loss: 21.7253
Step [3930/5000], Content Loss: 35.1897, Style Loss: 21.6554
Step [3940/5000], Content Loss: 35.2273, Style Loss: 21.5813
Step [3950/5000], Content Loss: 35.1788, Style Loss: 21.6054
Step [3960/5000], Content Loss: 35.1269, Style Loss: 21.6805
Step [3970/5000], Content Loss: 35.0900, Style Loss: 21.8119
Step [3980/5000], Content Loss: 35.1927, Style Loss: 21.3291
Step [3990/5000], Content Loss: 35.1524, Style Loss: 21.2990
Step [4000/5000], Content Loss: 35.1918, Style Loss: 21.2047
Step [4010/5000], Content Loss: 35.1982, Style Loss: 21.1408
Step [4020/5000], Content Loss: 35.1862, Style Loss: 21.0959
Step [4030/5000], Content Loss: 35.1774, Style Loss: 21.0460
Step [4040/5000], Content Loss: 35.1765, Style Loss: 20.9933
Step [4050/5000], Content Loss: 35.1875, Style Loss: 20.9270
Step [4060/5000], Content Loss: 35.2244, Style Loss: 20.8420
Step [4070/5000], Content Loss: 35.3023, Style Loss: 20.7296
Step [4080/5000], Content Loss: 35.3025, Style Loss: 20.6828
Step [4090/5000], Content Loss: 35.2673, Style Loss: 20.6988
```

之后 content_loss 稳定在 35 附近，而 style_loss 仍在下降，只是速度有所放缓。

结论分析与体会：

风格迁移，实质上是将一张图片的内容和另一张图片的风格融合在一起生成一张新图片。需要计算融合出来的图片和内容图片在内容上的差异性，同时也要计算融合出来的图片和风格图片在风格上的差异性，并想办法降低这些差异。

图片的内容容易理解，直接和像素值相关，但图片的风格确实比较抽象，但也可以通过一种方式量化——Gram 矩阵。（Gram 矩阵中每个元素都表示两层特征图的一种组合）总而言之，这是一个有趣的功能，感觉就像是用一些图片处理软件给源图加上滤镜生成新图一样。

Pytorch 我之前没怎么接触过，一直都是用 keras，所以代码读起来有点吃力，这方面需要加强学习。

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1－3 道问答题：

1. 用 grammer 矩阵为何要除以/(c*w*h)？

如果不除以（c*w*h）的话，style_loss 的值会很高，远远超过 content_loss，这样的话生成图片会学习到更多的 style，这里除以（c*w*h）可能是为了考虑两个 loss 的平衡。

2. 为什么 content_loss 在上升，style_loss 在下降？

某种程度上来看这是必然的，毕竟生成图 target 刚开始就是 content.jpg 的 copy，内容是完全一样的，但是随着训练次数的增加，target 其图片风格越来越向 style.jpg 靠拢，各个点像素值可以说一直在发生变化，这样在内容方面就和初始的 content.jpg 差异越来越大，所以 content_loss 上升是正常的。那么由于 target 的风格在向 style.jpg 靠拢，那么两者风格上的差异也就越来越小，所以 style_loss 也就会不断下降。