基于PyTorch的卷积神经网络在风格迁移中的应用

小组成员：张博(1610830313)、常雨昂(1611640119)、车兴竹(1611640130)

2019年1月9日

摘 要

【卷积神经网络】卷积神经网络（Convolutional Neural Network, CNN）是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元,对于大型图像处理有出色表现。

卷积神经网络由一个或多个卷积层和顶端的全连通层（对应经典的神经网络）组成，同时也包括关联权重和池化层（pooling layer）。与其他深度学习结构相比，卷积神经网络在图像和语音识别方面能够给出更好的结果。

【风格迁移】通过分离和重新组合图片内容与风格，CNN 可以创作出具有艺术魅力的作品。使用 CNN 将一张图片的语义内容与不同风格融合起来的过程被称为神经风格迁移（Neural Style Transfer）。

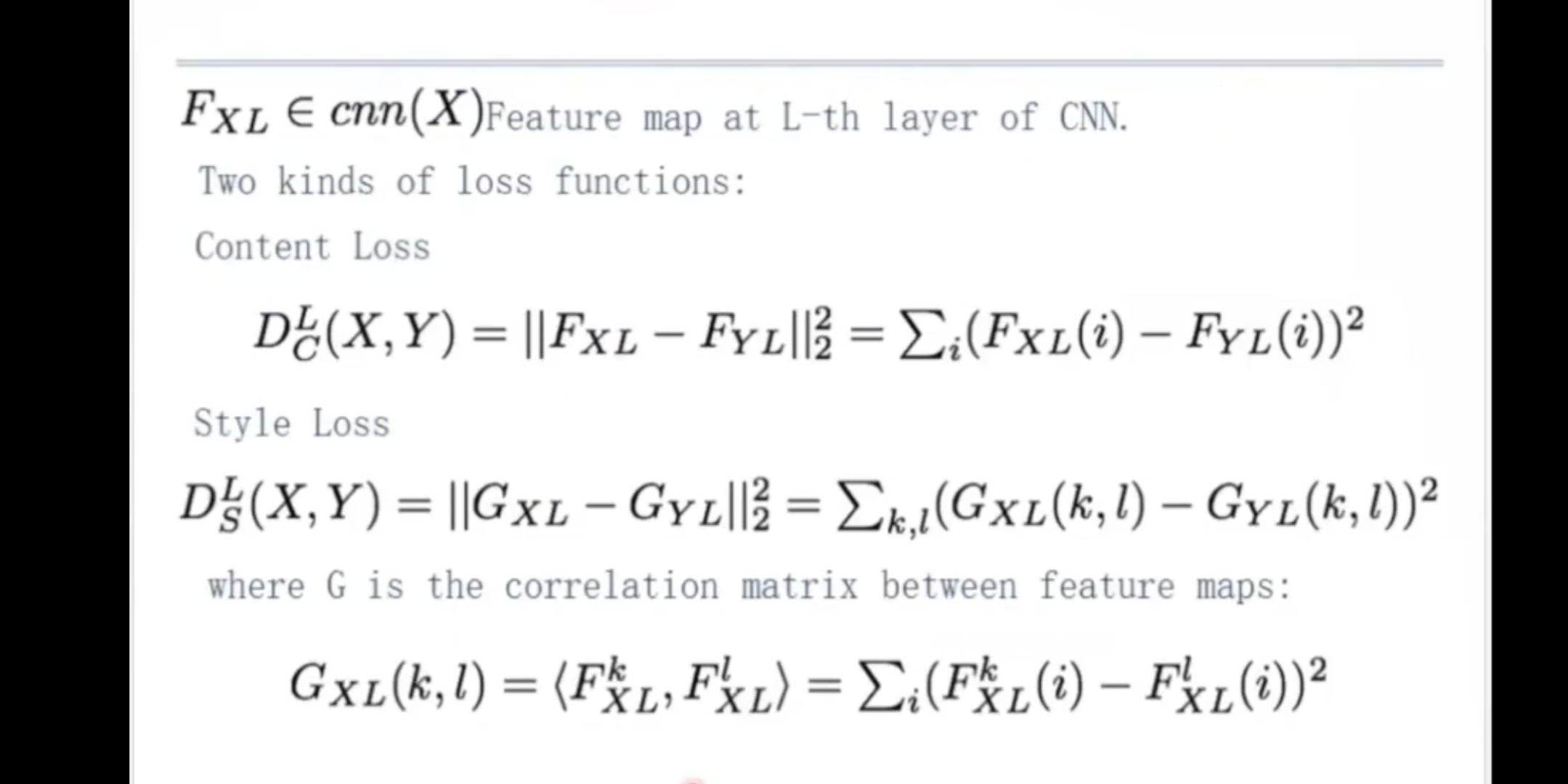
**1 问题描述**

1.1风格迁移

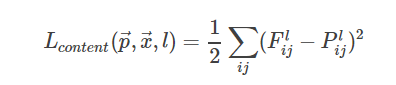
传统神经网络中图像的内容和风格的表示是可分离的。也就是说，我们可以间接地操纵这两种表现形式来产生新的、具有感知意义的图像。输入两张图片，一张用作合成图像的内容来源，一张用作合成图像的风格来源，输出的合成图像便具有特定的内容和特定的风格，形成新的艺术效果。

1.1.1图像内容损失函数（Content Loss）

用欧几里得距离表示图像的内容特征，距离越接近表示两张图像的内容越接近。



采用平方损失函数，为每个像素的损失和

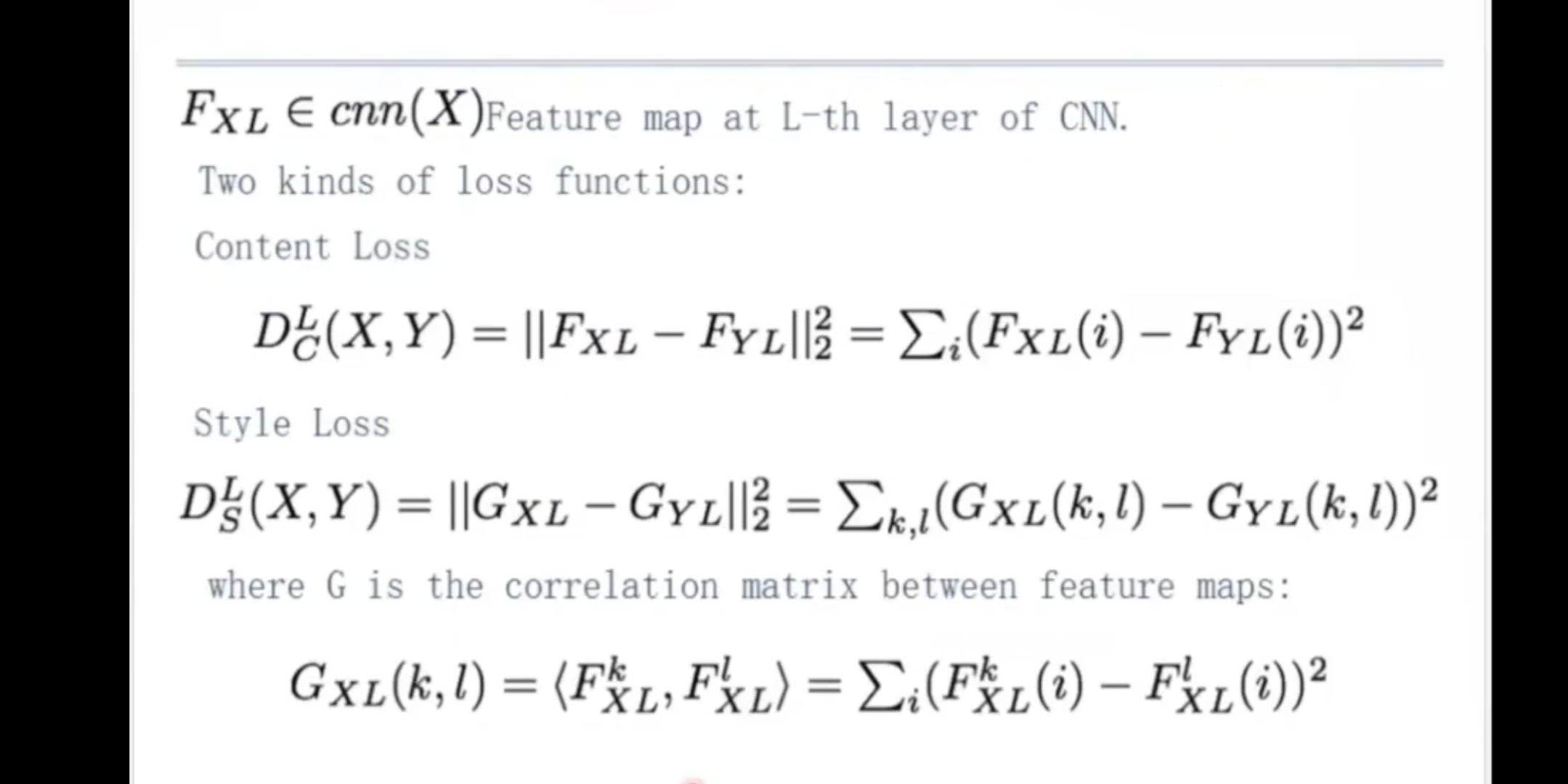


F为第l层第i个卷积的第j个位置的特征表示，用来代表内容，P为图像该位置的特征表示，x为想形成的目标图像。

可以这样理解，首先对待提取内容的图片p得到该位置的内容表示P，可以构造一个图像x在该位置的特征无限趋近于P，使得内容损失函数最小，目标就是找到这个在内容上无限接近P的x。

1.1.2图像风格损失函数（Style Loss）

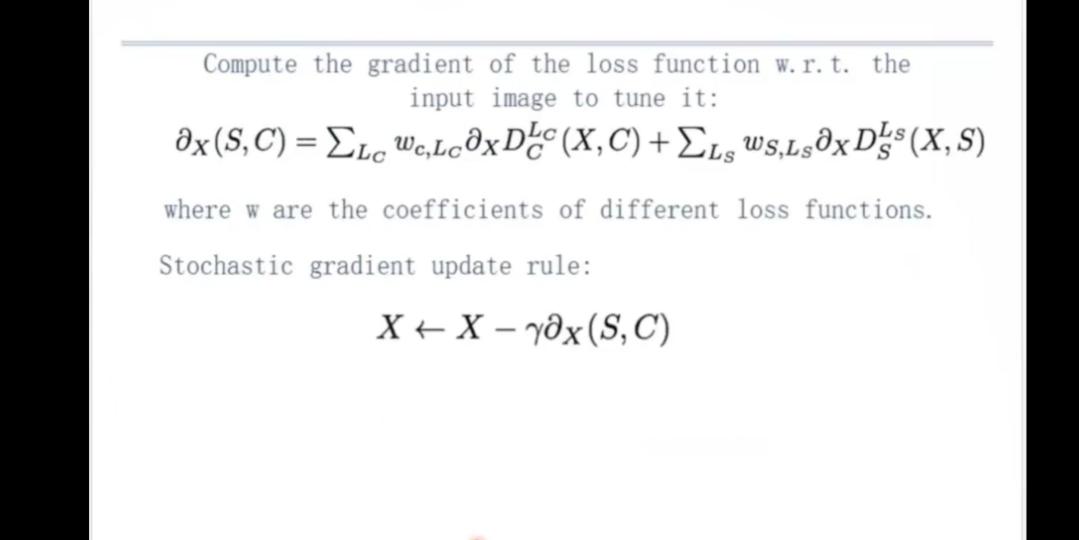
用协方差的距离表示图像的风格特征，距离越接近表示两张图像的内容越接近。



引入Gram矩阵，格拉姆矩阵可以看做feature之间的偏心协方差矩阵（即没有减去均值的协方差矩阵），在feature map中，每个数字都来自于一个特定滤波器在特定位置的卷积，因此每个数字代表一个特征的强度，而Gram计算的实际上是两两特征之间的相关性，同时，Gram的对角线元素，还体现了每个特征在图像中出现的量，因此，Gram有助于把握整个图像的大体风格。有了表示风格的Gram Matrix，要度量两个图像风格的差异，只需比较他们Gram Matrix的差异即可。

1.2总的损失函数

将Content Loss和Style Loss 加权求和得到总的损失函数。



**2 优化器**

本实验选择L-BFGS算法作为优化器：

optimizer = optim.LBFGS([input\_param])

1. BFGS算法就是对拟牛顿算法的一个改进。L-BFGS算法的基本思想是：算法只保存并利用最近m次迭代的曲率信息来构造海森矩阵的近似矩阵。L-BFGS算法比较适合在大规模的数值计算中，具备牛顿法收敛速度快的特点，但不需要牛顿法那样存储Hesse矩阵，因此节省了大量的空间以及计算资源。

**3 vgg16概述**

3.1 VGG原理

VGG16相比AlexNet的一个改进是采用连续的几个3x3的卷积核代替AlexNet中的较大卷积核（11x11，7x7，5x5）。对于给定的感受野（与输出有关的输入图片的局部大小），采用堆积的小卷积核是优于采用大的卷积核，因为多层非线性层可以增加网络深度来保证学习更复杂的模式，而且代价还比较小（参数更少）。简单来说，在VGG中，使用了3个3x3卷积核来代替7x7卷积核，使用了2个3x3卷积核来代替5\*5卷积核，这样做的主要目的是在保证具有相同感知野的条件下，提升了网络的深度，在一定程度上提升了神经网络的效果。比如，3个步长为1的3x3卷积核的一层层叠加作用可看成一个大小为7的感受野（其实就表示3个3x3连续卷积相当于一个7x7卷积），其参数总量为 3x(9xC^2) ，如果直接使用7x7卷积核，其参数总量为 49xC^2 ，这里 C 指的是输入和输出的通道数。很明显，27xC^2小于49xC^2，即减少了参数；而且3x3卷积核有利于更好地保持图像性质。

## 3.2 VGG网络结构

## 下面是VGG网络的结构v2-ea924e733676e0da534f677a97c98653_hd

- VGG16包含了16个隐藏层（13个卷积层和3个全连接层），如上图中的D列所示

**3.3 vgg优缺点**

3.3.1 vgg优点

VGGNet的结构非常简洁，整个网络都使用了同样大小的卷积核尺寸（3x3）和最大池化尺寸（2x2）。几个小滤波器（3x3）卷积层的组合比一个大滤波器（5x5或7x7）卷积层好：验证了通过不断加深网络结构可以提升性能。

3.3.2 VGG缺点

VGG耗费更多计算资源，并且使用了更多的参数（这里不是3x3卷积的锅），导致更多的内存占用（140M）。其中绝大多数的参数都是来自于第一个全连接层。

**4 输入描述**

4.1 输入内容图片

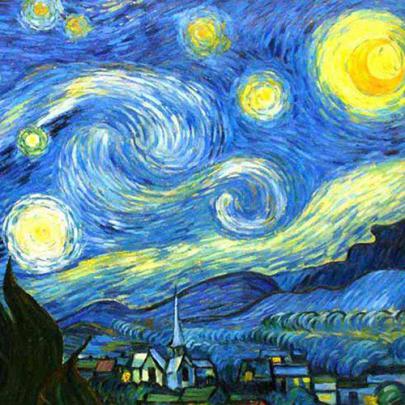
输入一张拍摄的风景图片



4.2 输入风格图片

输入三种不同风格的图片，如名家画作。

依次：现代派《坐着的裸女》（毕加索）、后印象派《星月夜》（梵高）、中国山水画（作者未知）

**5 实验结果**

5.1 使用以训练好的神经网络VGG16

Sequential(

(0): Conv2d(3, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(1): ReLU(inplace)

(2): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(3): ReLU(inplace)

(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)

(7): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(8): ReLU(inplace)

(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)

(12): Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(13): ReLU(inplace)

(14): Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(15): ReLU(inplace)

(16): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)

(17): Conv2d(256, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(18): ReLU(inplace)

(19): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(20): ReLU(inplace)

(21): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(22): ReLU(inplace)

(23): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)

(24): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(25): ReLU(inplace)

(26): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(27): ReLU(inplace)

(28): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

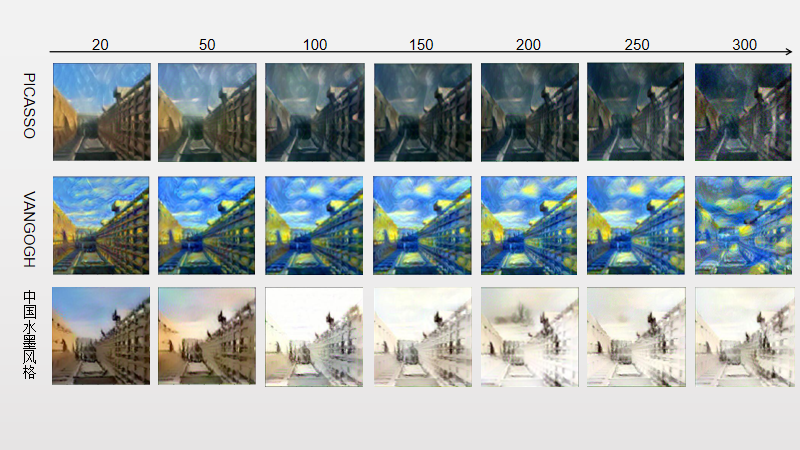
(29): ReLU(inplace)

(30): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)

)

5.2 输出结果

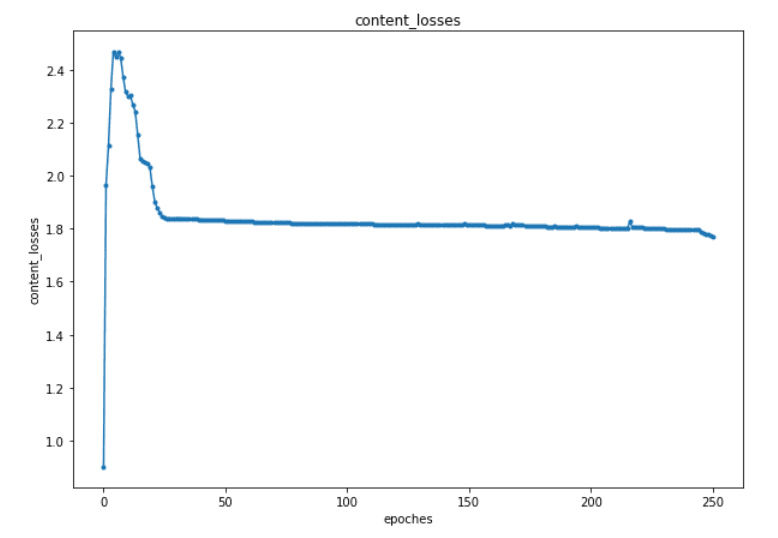
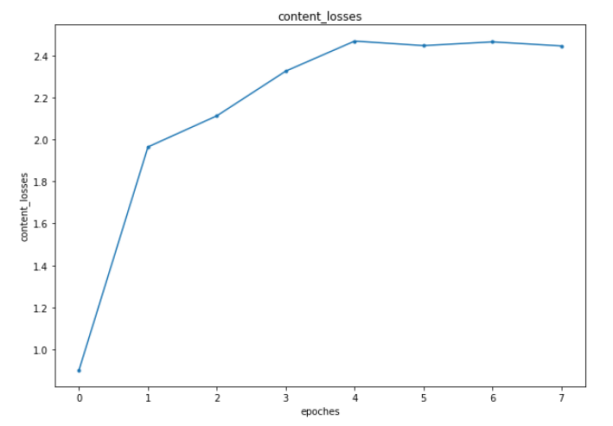
更改训练次数，得到不同程度风格化的图片

结果展示了三种不同风格与同一内容在不同训练次数下得到的合成图片。

5.3 损失函数图像

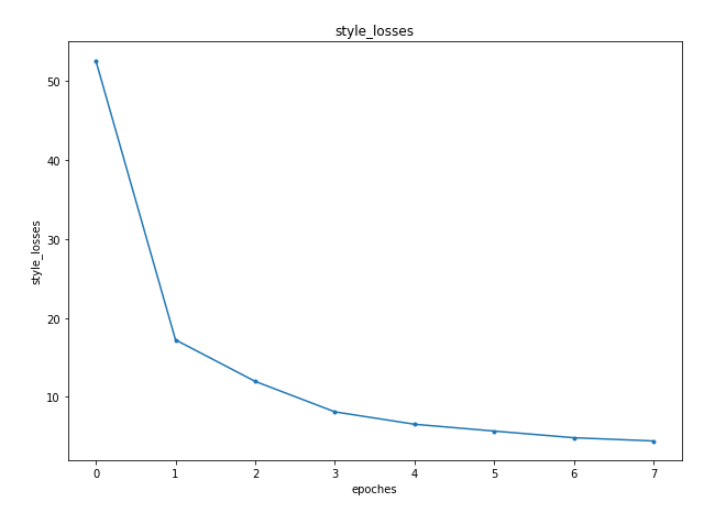
5.3.1 图像内容损失函数（Content Loss）

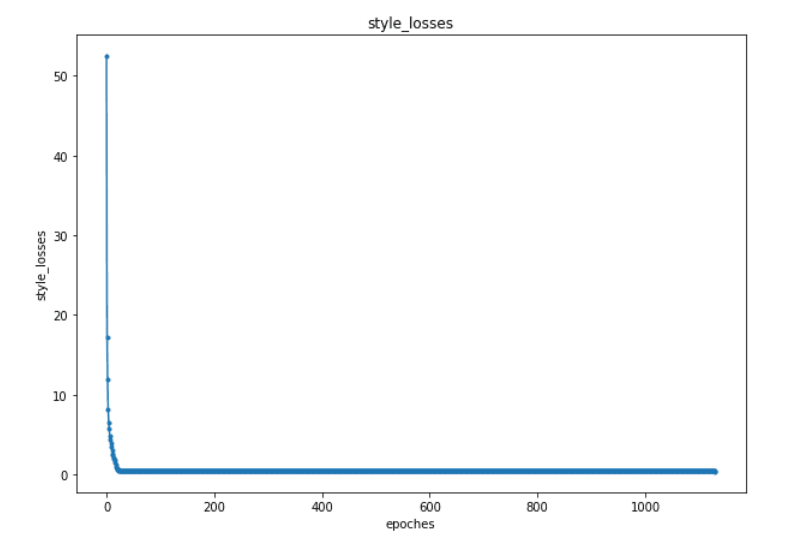
训练次数和多训练次数的图像内容损失函数比较



5.3.2 图像风格损失函数（Style Loss）

少训练次数和多训练次数的图像风格损失函数比较





**6 实验结果分析**

本次实验我们选取了毕加索，梵高和的画作作为风格迁移的目标，对天工大风景照分别进行了20，50，100，150，200，250，300次训练（实验结果图即为输出结果）。每组第一张图片为训练20次的结果，经过卷积，batchnormnal, relu之后保留大部分特征原图片的特征，之后经过又，多次训练，以此向前传到，可以观察到原本图片的形状在逐渐消失，特征越来越模糊，亮度也随着需求图片的亮度变化，直到最后，已经变成更贴合所求风格的没有太多形状的的色块。

经过对比，发现当训练次数不断增加时，合成图片的风格特征越来越明显，而图片的内容特征会逐渐的减弱。可以想象当训练次数无限增大时，合成图片会越来约接近输入的风格图片。所以可以通过改变训练次数，得到不同程度风格化的合成图片，以满足不同的需求。  
 通过比较两种损失函数，可以看出，训练4次之后就已经达到了较为理想的结果，而内容的损失函数，在训练多次后达到峰值，之后会下降，最后趋于稳定。  
在一开始运行代码的时候，我们采用的是vgg19，以及用cuda调用GPU进行计算，但是由于运行数据太多，笔记本电脑的硬件条件不够，没有运行成功。后来采用vgg16，用CPU计算，在训练次数为20次时，能在一分钟左右输出结果。因为本实验不用分类，所以所用的卷积神经网络没有全连接层。

卷积神经网络中最主要的两个部分就是卷积层和池化层。

卷积层主要的作用就是提取图片的线条特征、色彩特征以及一些其他的形状特征，且越深层的卷积提取出来的特征就越多。卷积核的作用和动物视觉细胞的分工类似。 除此之外，卷积层还能够减少冗余的数据，抛弃一些不重要的数据，减少训练的负担。 池化层的主要作用就是降采样，减少数据的数量，提高网络的计算效率。并且池化能够帮助输入保持一种平移不变性，既不用关心某个特征出现的位置，而只关心是否具有该特征。

**7 小组分工说明**

张博：前期资料收集，代码编写，实验图片提供

常雨昂：前期资料收集，代码调试，实验报告编写

车兴竹：代码编写，实验报告编写

参考资料

[1] 参考论文 https://arxiv.org/pdf/1508.06576.pdf

[2] PyTroch [官网网站](https://pytorch.org/)

[3] https://zhuanlan.zhihu.com/p/41423739作者：Amusi