

Image Processing Project: Neural Style Transfer

3017207553 黄千慧

2019 年 12 月 24 日

图像风格化是将一张照片渲染成有艺术风格的画作。图像风格化算法的输入有二，分别是内容图和风格图，输出有一个，为风格迁移后的结果图。

概要 在风格迁移出现之前，比较相关的方向是纹理迁移 (texture transfer)。纹理迁移并没有流行起来呢。因为纹理迁移是基于低层次的图像特征来做的，没有考虑语义信息，所以很多结果不那么得尽如人意。但是纹理建模方法的相关研究解决了图像风格化迁移的第一个大问题：如何对风格图中的风格特征进行建模和提取。那么接下来的研究就是如何和内容混合然后还原成一个相应的风格化结果呢？这就到了另一个领域——图像重建 (Image Reconstruction) 了。图像重建的输入是特征表达，输出是特征表达对应的图像。是把某个特征逆向重建为原来的图像。图像风格化迁移算法 = 图像重建算法 + 纹理建模算法。图像风格化迁移这一领域的 Gatys 大神率先提出了新的基于 CNN 的纹理建模方法。

其核心思想是在图像经过预训练的 VGG 网络时的特征表达 (feature map) 上计算 Gram 矩阵，利用得到的 Gram 矩阵来表示一种纹理。Gram 矩阵的计算方式是先将预训练 VGG 某一层的特征表达。Gatys 发现这个 Gram 矩阵可以很好地表示大多数纹理。这个 Gram 矩阵的纹理表示方法其实是利用了二阶统计量来对纹理进行建模。我们可以解释 Gram 抽取到的是：在图片的同一位置下，不同特征之间的组合。而有学者证明了 Gram 是 MMD(maximum mean discrepancy) 的二次多项式核变种用 Gram 矩阵来对图像中的风格进行建模和提取，再利用慢速图像重建方法，让重建后的图像以梯度下降的方式更新像素值，使其 Gram 矩阵接近风格图的 Gram 矩阵 (即风格相似)，然后其 VGG 网络的高层特征表达接近内容图的特征表达 (即内容相似)，实际应用时候经常再加个总变分 TV 项来对结果进行平滑，最终重建出来的结果图就既拥有风格图的风格，又有内容图的内容。

具体实现细节见 jupiter notebook 文件。

大致思路

首先，我们初始化合成图像，例如将其初始化成内容图像该合成图像是样式迁移过程中唯一需要更新的变量，即样式迁移所需迭代的模型参数。然后，我们选择一个预训练的卷积神经网络来抽取图像的特征，其中的模型参数在训练中无须更新深度卷积神经网络凭借多个层逐级抽取图像的特征。我们可以选择其中某些层的输出作为内容特征或样式特征以图 ?? 为例，这里选取的预训练的神经网络含有 3 个卷积层

其中第二层输出图像的内容特征，而第一层和第三层的输出被作为图像的样式特征接下来，我们通过正向传播 (实线箭头方向) 计算样式迁移的损失函数并通过反向传播 (虚线箭头方向) 迭代模型参数，即不断更新合成图像样式迁移常用的损失函数由 3 部分组成：内容损失 (content loss) 使合成图像与内容图像在内容特征上接近样式损失 (style loss) 令合成图像与样式图像在样式特征上接近总变差损失 (total variation loss) 则有助于减少合成图像中的噪点最后，当模型训练结束时，我们输出样式迁移的模型参数，即得到最终的合成

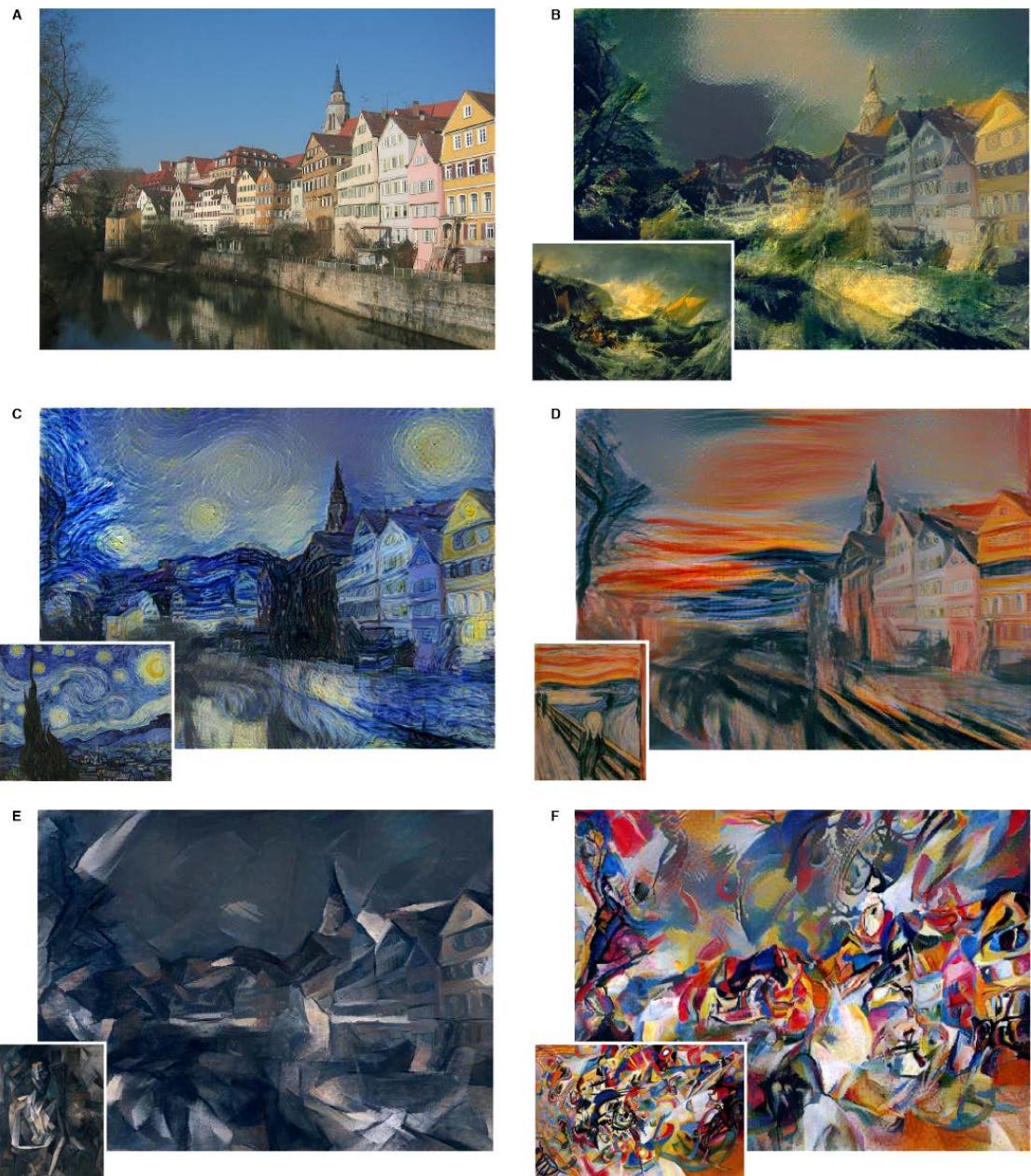


图 1: Neural Style Transfer

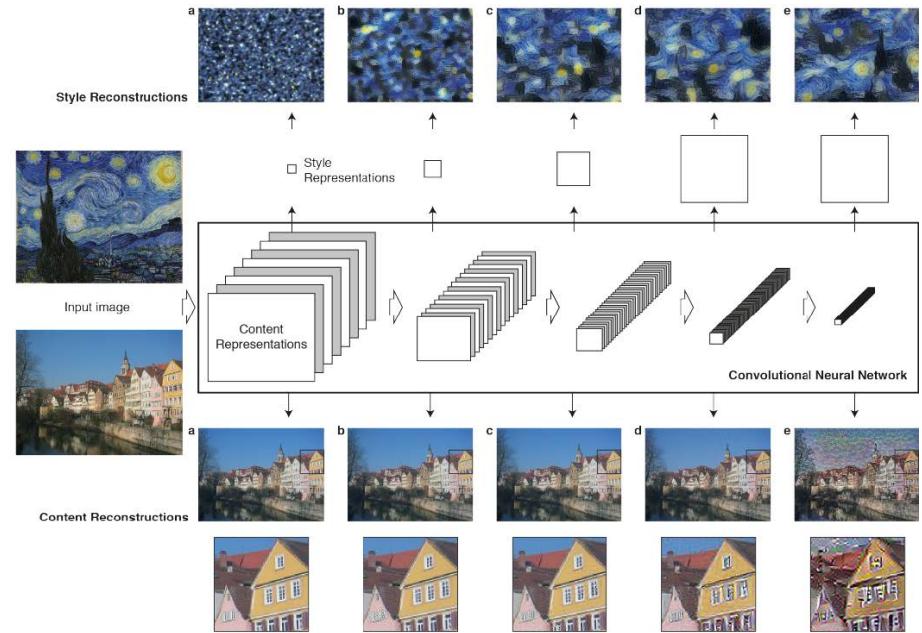
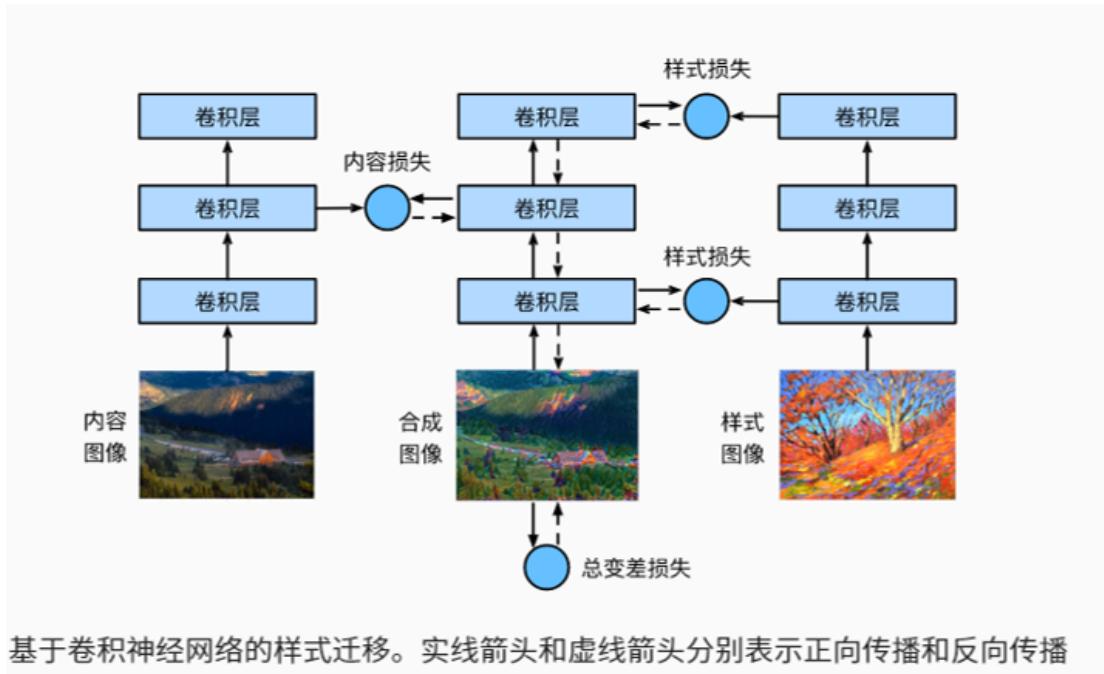


图 2: 模型图



基于卷积神经网络的样式迁移。实线箭头和虚线箭头分别表示正向传播和反向传播

图 3: 3 个卷积层的示意图

图像。

内容损失

与线性回归中的损失函数类似，内容损失通过平方误差函数衡量合成图像与内容图像在内容特征上的差异

样式损失

样式损失也一样通过平方误差函数衡量合成图像与样式图像在样式上的差异假设该输出的样本数为 1，通道数为 c ，高和宽分别为 h 和 w ，我们可以把输出变成 c 行 hw 列的矩阵 X 。矩阵 X 可以看作是由 c 个长度为 hw 的向量 x_1, \dots, x_c 组成的其中向量 x_i 代表了通道 i 上的样式特征

这些向量的格拉姆矩阵 (Gram matrix) XX^T $R^{c \times c}$ 中 i 行 j 列的元素 x_{ij} 即向量 x_i 与 x_j 的内积它表达了通道 i 和通道 j 上样式特征的相关性我们用 GRAM 矩阵表达样式层输出的样式需要注意的是，当 hw 的值较大时，格拉姆矩阵中的元素容易出现较大的值此外，格拉姆矩阵的高和宽皆为通道数 c 为了让样式损失不受这些值的大小影响，下面定义的格拉姆矩阵除以了矩阵中元素的个数，即 chw

Gram matrix 为什么能捕获风格信息呢？

假设输入图像经过卷积后，得到的 feature map 为 $[b, c, h, w]$ 我们经过 flatten 和矩阵转置操作，可以变形为 $[b, c, hw]$ 和 $[b, hw, c]$ 的矩阵再对 1, 2 维作矩阵内积得到 $[b, c, c]$ 大小的矩阵

比如我们假设输入图像经过卷积后得到的 $[b, c, hw]$ 的 feature map 其中我们用 f_m 表示第 m 个通道的特征层， f_n 为第 n 个通道特征层则 Gram Matrices 中元素 $f_m * f_n$ 代表的就是 m 通道和 n 通道特征 flatten 后按位相乘（内积）图形化计算过程就是 ?? :

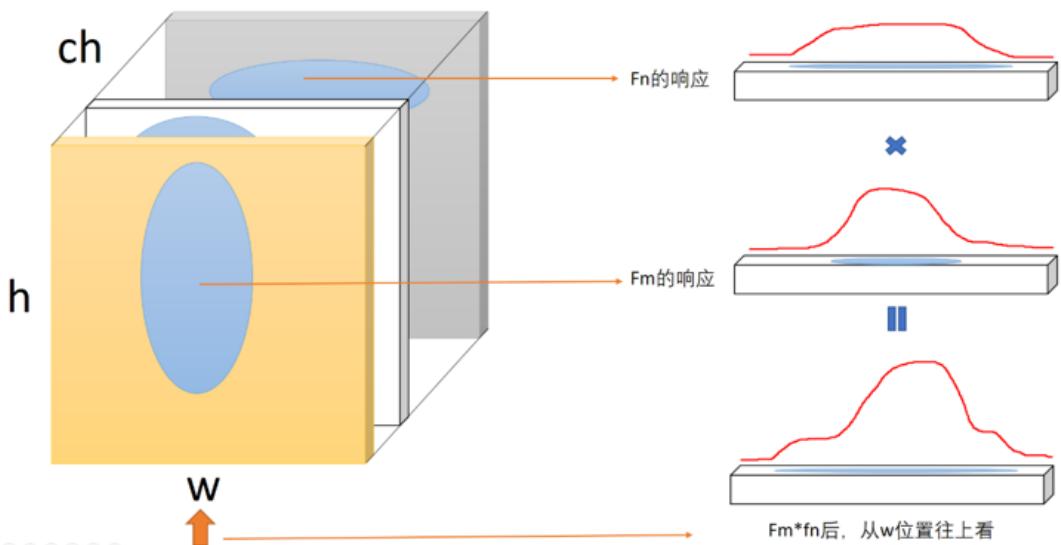


图 4: Gram

用蓝色色表示 feature 响应值大，其它色表示响应值小。其中黄色为 m 通道特征，灰色为 n 通道特征可以看到， $f_m * f_n$ 记录了第 m 通道特征和第 n 通道特征在不同位置下融合后的响应。而乘法的作用可以理解为只是让大的更大，小的更小，通过增大特征间的变化率，突出不同层融合特征间的差异 Gram 矩阵的每个值可以说是代表 m 通道的特征与 n 通道的

特征的互相关程度由于滤波器的性质，它可以是高通可以是低通，或多或少会对诸如高光，阴影，饱和度等特征敏感这样在滤波后就会在诸如高光，阴影，饱和度高的地区产生更高的响应幅值这样通过在特征图的同一个位置下两两融合该位置的不同特性滤波器得到的特征响应诸如高光响应幅值和高饱和度响应幅值的融合响应幅值，即得到了 Gram Matrices 的元素 $f_m * f_n$ 然后通过比较两张输入图 Gram Matrices 的 element-wise 的差别即元素 $f_m * f_n$ 和 $f_n * f_m$ 之间的差别来区分风格 我们可以解释 Gram Matrices 抽取到的是：在图片的同一位置下，不同特征之间的组合

而有人证明了 Gram Matrices 是 MMD 的二次多项式核变种，那 MMD 表征的是什么信息呢？

MMD: maximum mean discrepancy。最大平均差异

基于两个分布的样本，通过寻找在样本空间上的连续函数 f ，求不同分布的样本在 f 上的函数值的均值通过把两个均值作差可以得到两个分布对应于 f 的 mean discrepancy 寻找一个 f 使得这个 mean discrepancy 有最大值，就得到了 MMD 最后取 MMD 作为检验统计量 (test statistic)，从而判断两个分布是否相同如果这个值足够小，就认为两个分布相同，否则就认为它们不相同时这个值也用来判断两个分布之间的相似程度。在迁移学习中，这个 f 一般是用高斯核函数 (RBF) 由图可知， M_1 和 N_1 分别表示通道和 $H * W$ flatten 后的特征点其证明了我们是对不同通道间的各个位置特征点对应相乘和对不同位置特征点下各个通道对应相乘是一样的图解就是这样的 ?? :

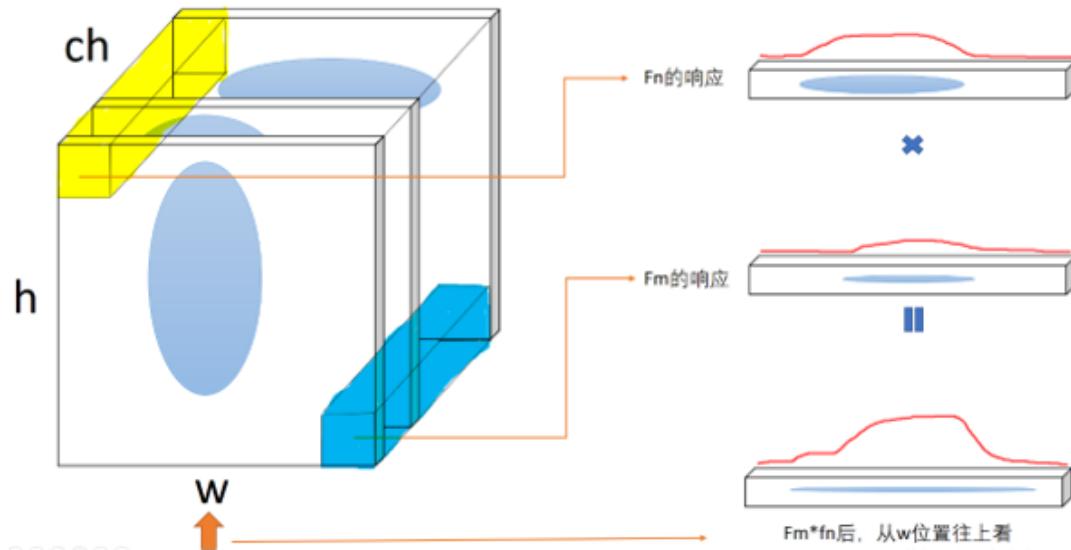


图 5: MMD

黄色和蓝色分别代表两个特征位置下各个通道的特征点集合，相乘后得到右边结果解释类似上面 Gram Matrices。因为滤波器滤波后得到的每个特征点表征的是输入图片的语义信息，所以 我们可以解释 MMD 抽取到的是：在图片的同一特征下，不同语义信息之间的组合

个人理解 CNN 下的风格为：图片的语义信息的组合和特征信息的组合

实验结果

1 测试图片

content image 和 style image

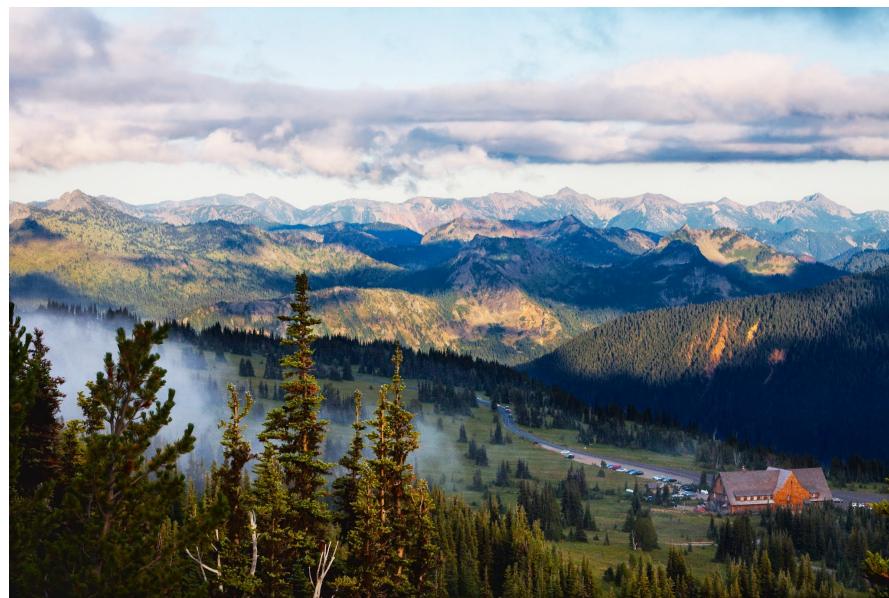


图 6: content image

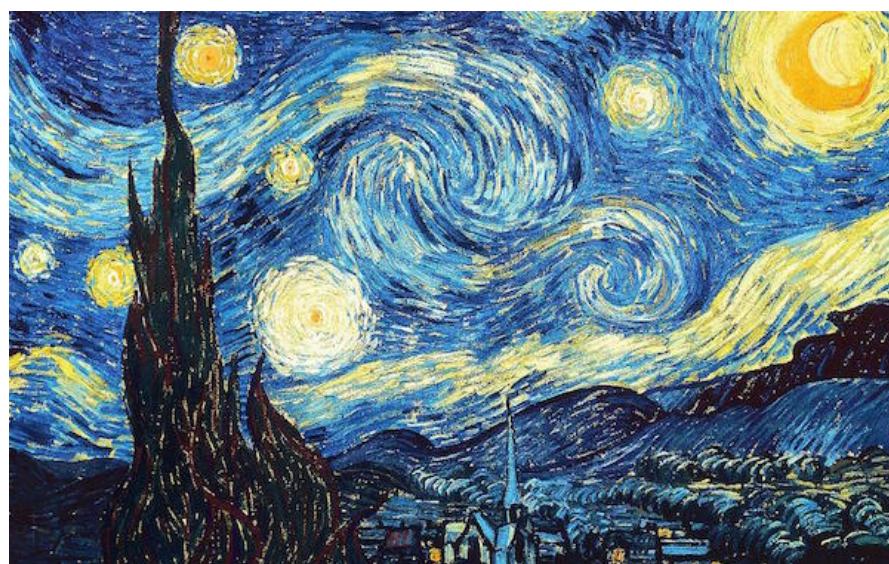


图 7: style image

2 比较不同模型的结果

2.1 普通版本

2.1.1 普通版本

另一种风格

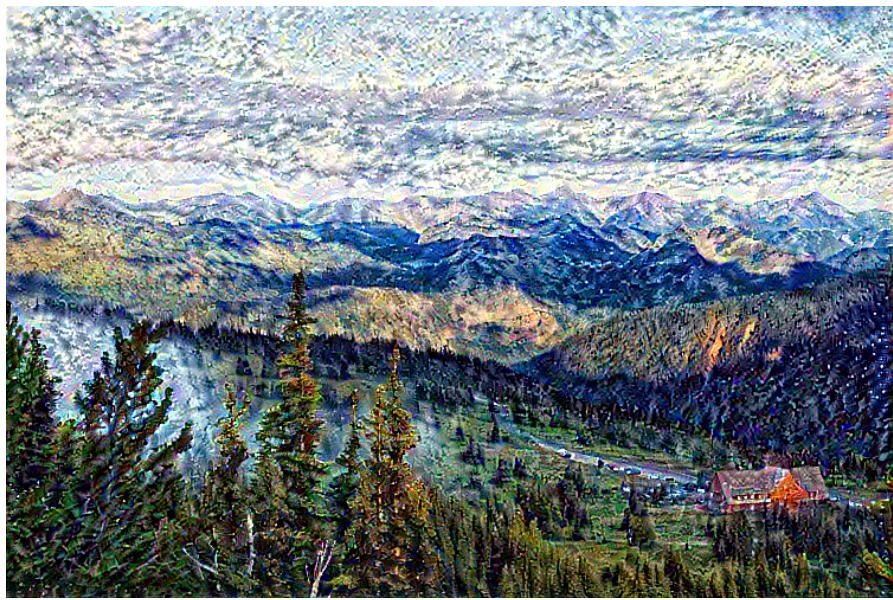


图 8: 迭代 200 次



图 9: 迭代 300 次



图 10: 迭 600 次

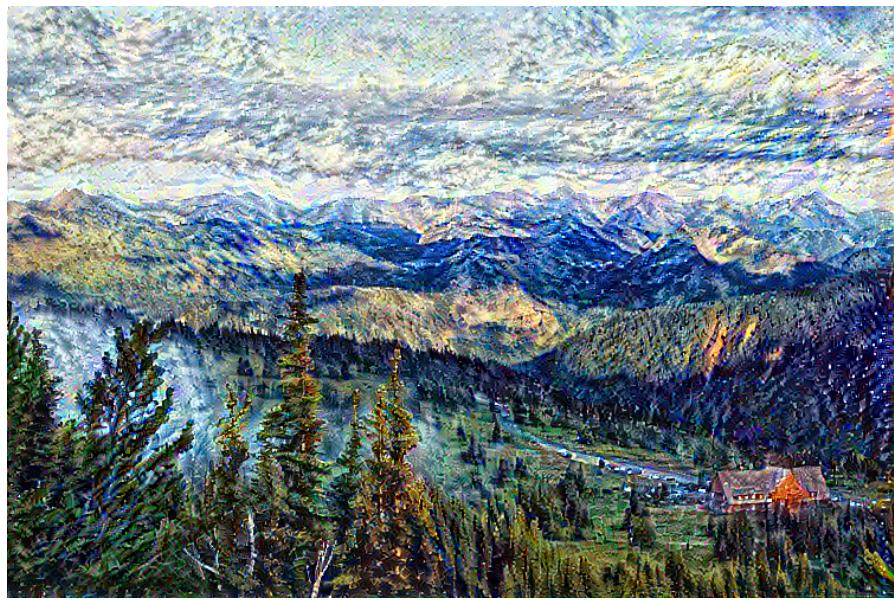


图 11: 迭 700 次



图 12: 迭 800 次

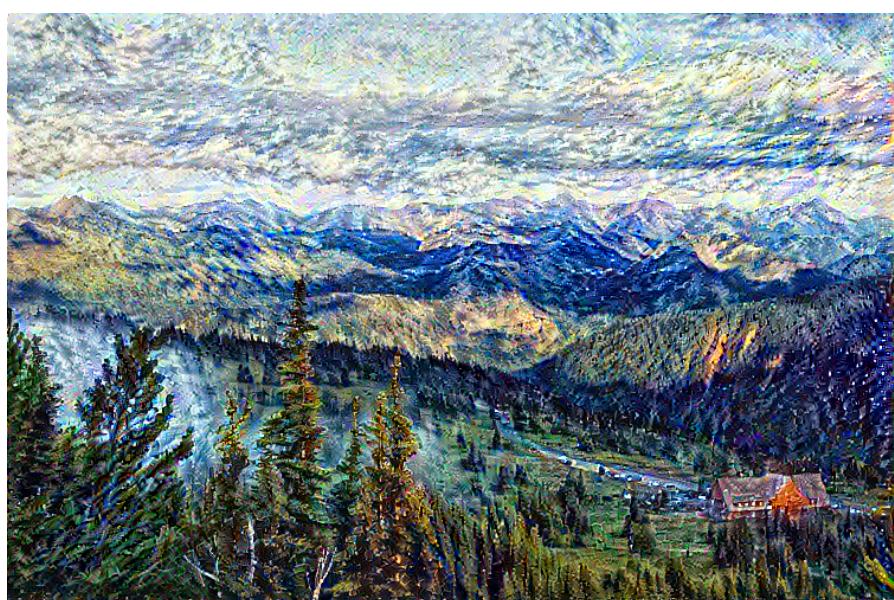


图 13: 最终结果



图 14: style2 image

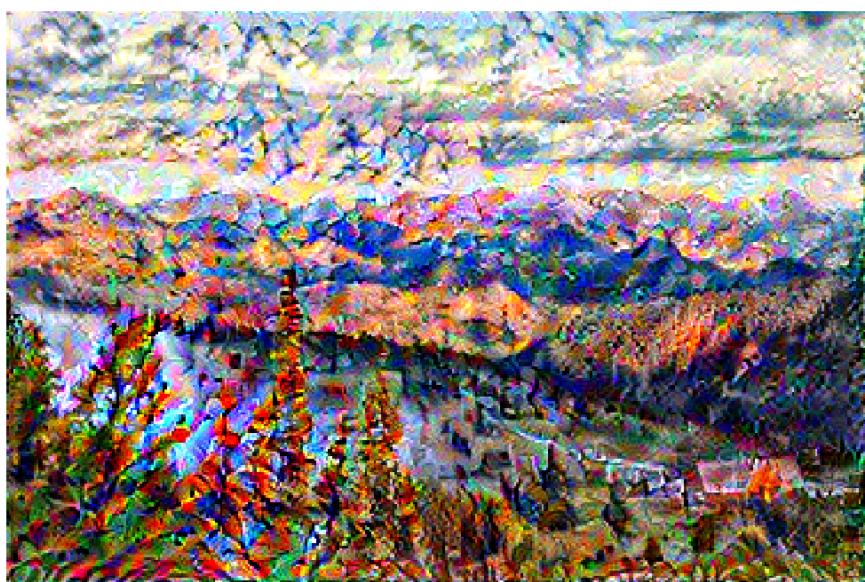


图 15: 最终结果

2.1.2 普通版本增大 style weight

可以看到结果更加接近风格图片，不管是纹理还是颜色。但是相对 content 的内容有点不清楚，不过还是能看出来。所以 style weight 和 content weight 参数需要好好调正。

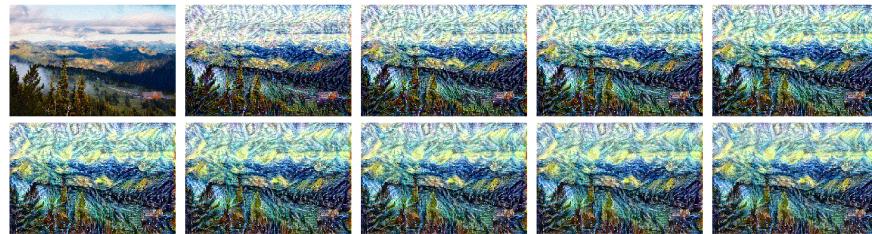


图 16: 普通版本增大 style weight 过程

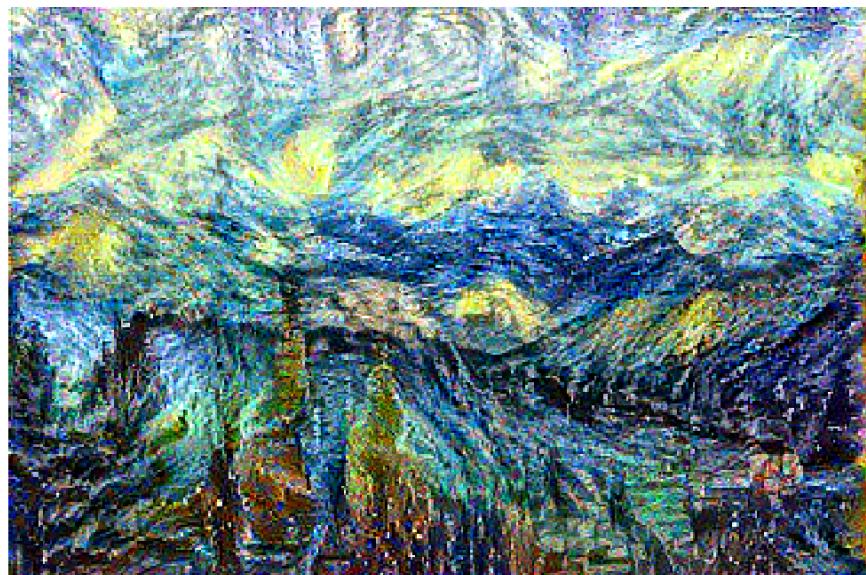


图 17: 普通版本增大 style weight 结果

2.2 普通版本加上 total variation loss

可以看到上面基础版本的结果会产生大量的高频误差。我们可以直接通过正则化图像的高频分量来减少这些高频误差。在风格转移中，这通常被称为 total variation loss。本质上高频分量是边缘信息。我们可以用 Sobel 边缘检测器获得 ??

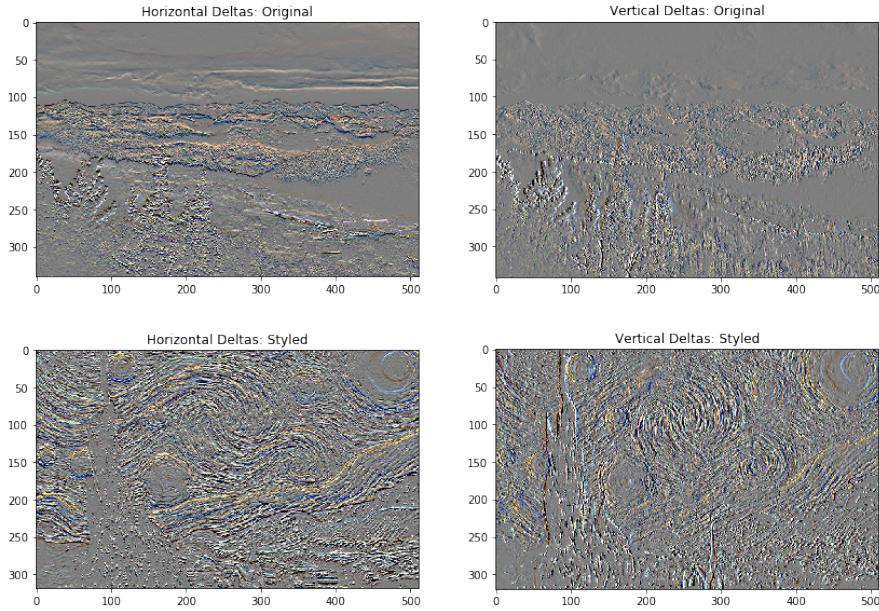


图 18: 边缘检测结果

2.2.1 hight-total-variation-weight 版本

可以看到边缘被平滑，权重过大使得图片比较糊

2.2.2 low-total-variation-weight 版本

可以看到比上一个好多了。对边缘的平滑有一定的效果，但是颜色都糊成一团。说明权重还是有点偏大，后面我们再将权重调小一点。

2.3 最终调好参数的结果

2.4 Fast Style Transfer

这里是基于 Google Tensorflow hub 提供的模型实现。这个模型的实现基于谷歌大脑和蒙特利尔大学合作的论文 Exploring the structure of a real-time, arbitrary neural artistic stylization network。是一个单模型任意风格的快速风格化迁移算法。他们发现在训练好的一个风格化网络基础上，只通过在 Instance Normalization 层上做一个仿射变换（他们起了个名字叫 Conditional Instance Normalization，简称 CIN），就可以得到一个具有完全不同风格的结果。既然通过改变 CIN 层中仿射变换的参数，就可以得到不同的 style，换言之，只要任意给一个风格，我们只需要知道他的 CIN 层中的仿射变换的参数就可以了

运行结果在文件目录的 output 目录下



图 19: with-hight-total-variation-weight 过程

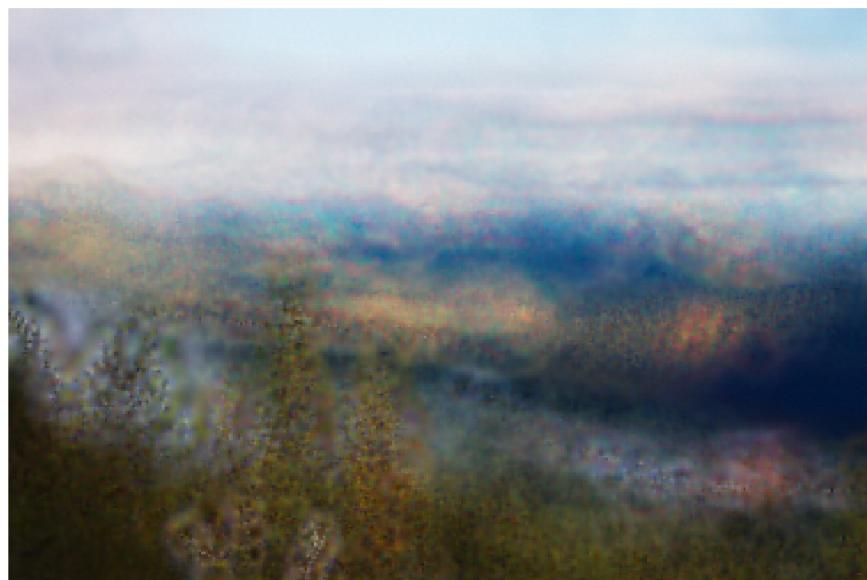


图 20: with-hight-total-variation-weight



图 21: low-total-variation-weight 过程



图 22: low-hight-total-variation-weight



图 23: 最终过程

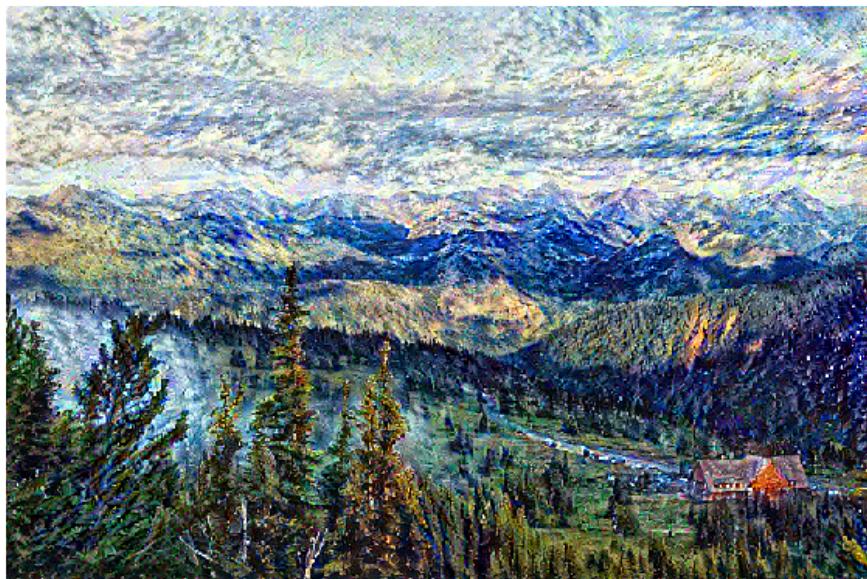


图 24: 最终结果



图 25: ¹⁴最终结果 2

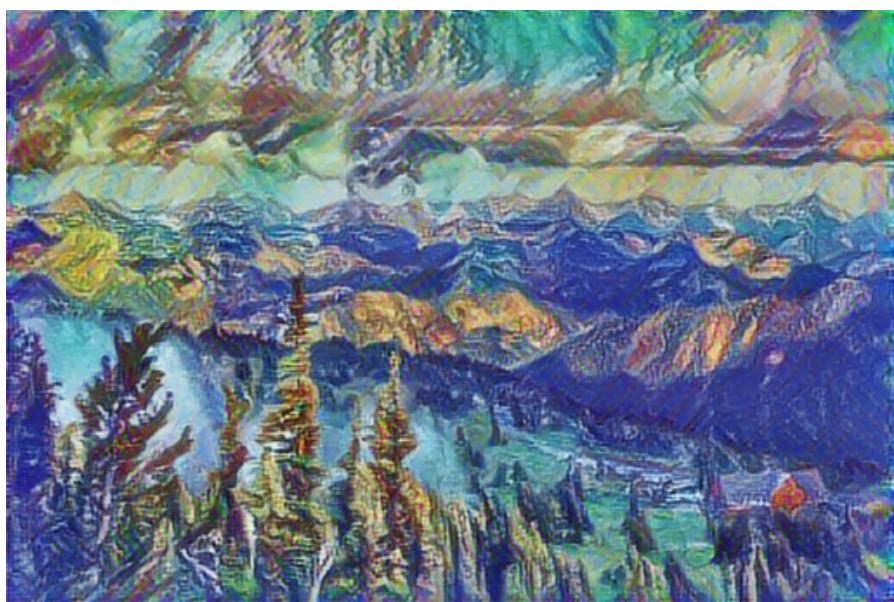


图 26: Fast Style Transfer 结果

参考文献

- [1] A Neural Algorithm of Artistic Style
<https://arxiv.org/pdf/1508.06576.pdf>

- [2] 综述：图像风格化算法最全盘点
<https://www.jiqizhixin.com/articles/2018-05-15-5>

- [3] examples/neural_style_transfer
https://github.com/raskutti/examples/blob/master/community/en/neural_style_transfer/neural_style_transfer.ipynb

- [4] [译] Perceptual Losses for Real-Time Style Transfer and Super-Resolution (Stanford University)
<https://cloud.tencent.com/developer/article/1125595>

- [5] Neural-Style-Transfer-Papers
<https://github.com/ycjing/Neural-Style-Transfer-Papers>

- [6] Gram Matrices 理解
<https://blog.csdn.net/tunhuzhuang1836/article/details/78474129>

- [7] TensorFlow 之深入理解 Fast Neural Style
<https://zhuanlan.zhihu.com/p/30486310>

- [8] deeplearning.al 卷积神经网络课程
<https://www.coursera.org/specializations/deep-learning>

- [9] 动手学深度学习课程
http://zh.d2l.ai/chapter_computer-vision/neural-style.html

- [10] Understanding your Convolution network with Visualizations
<https://towardsdatascience.com/understanding-your-convolution-network-with-visualizations-a48834415>