

# 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。

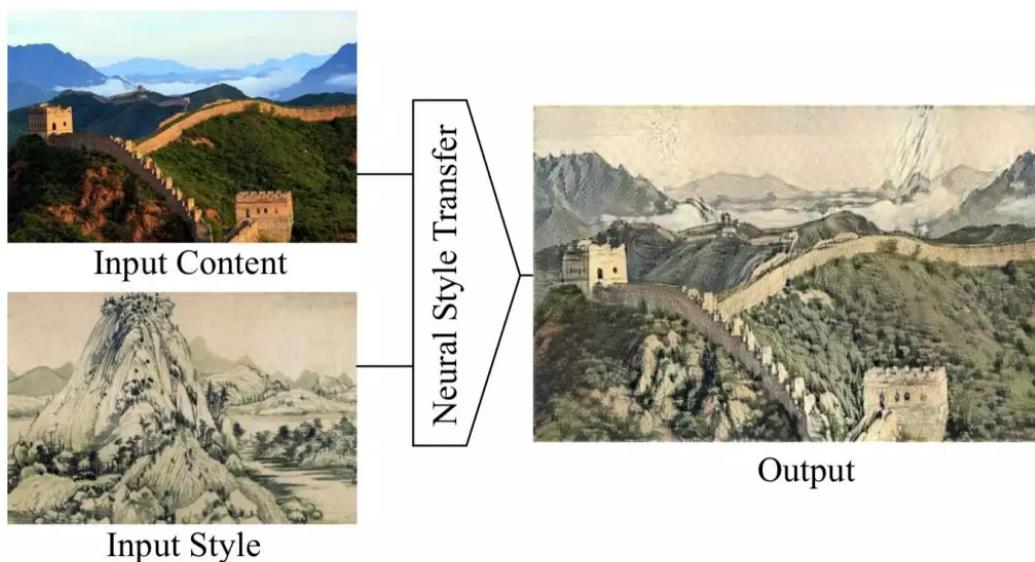


图 1: Neural Style Transfer

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 jupyter notebook 文件。

## 实验结果

### 1 测试图片

content image 和 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

##### 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 的内容有点糊，不过还是能看出来。所以 style weight 和 content weight 参数需要好好调正。

#### 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 ??

##### 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



图 2: content image



图 3: style image

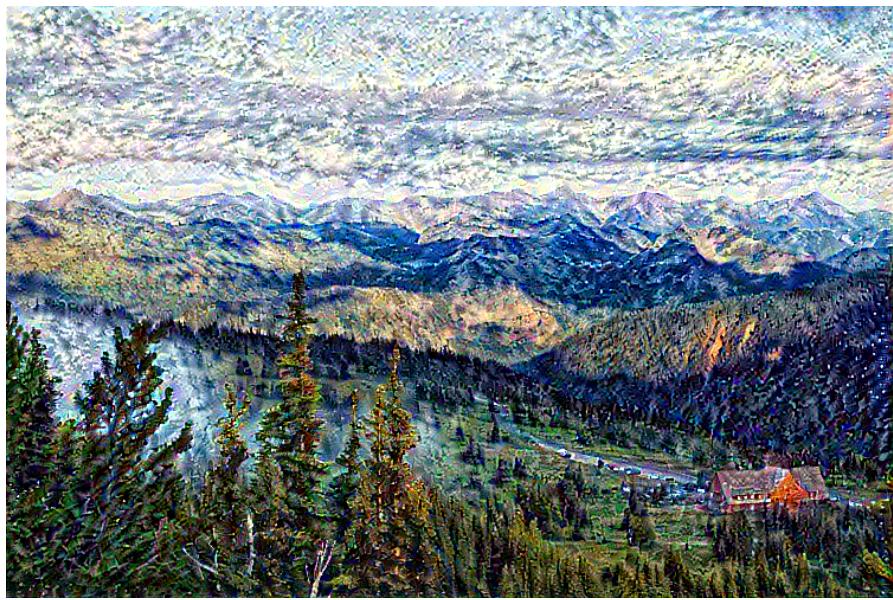


图 4: 迭代 200 次

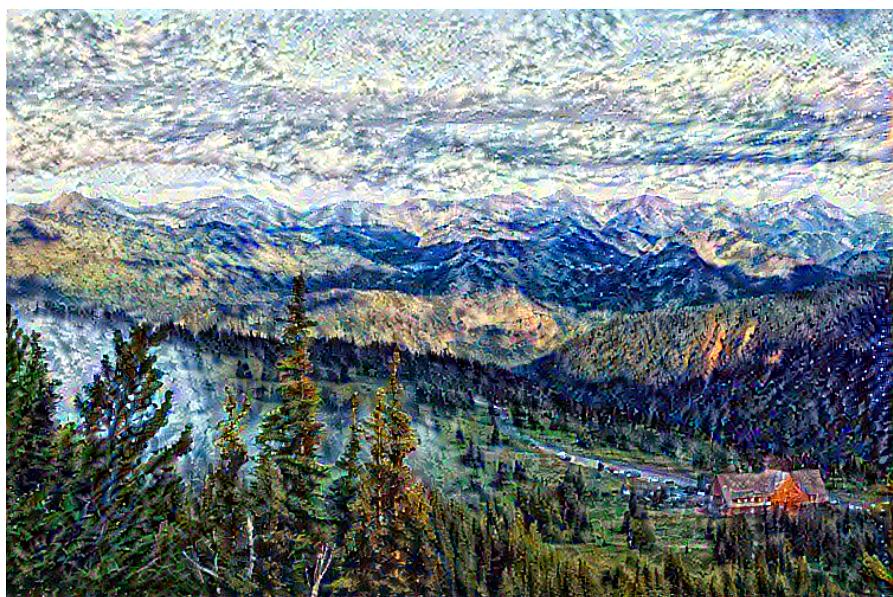


图 5: 迭代 300 次

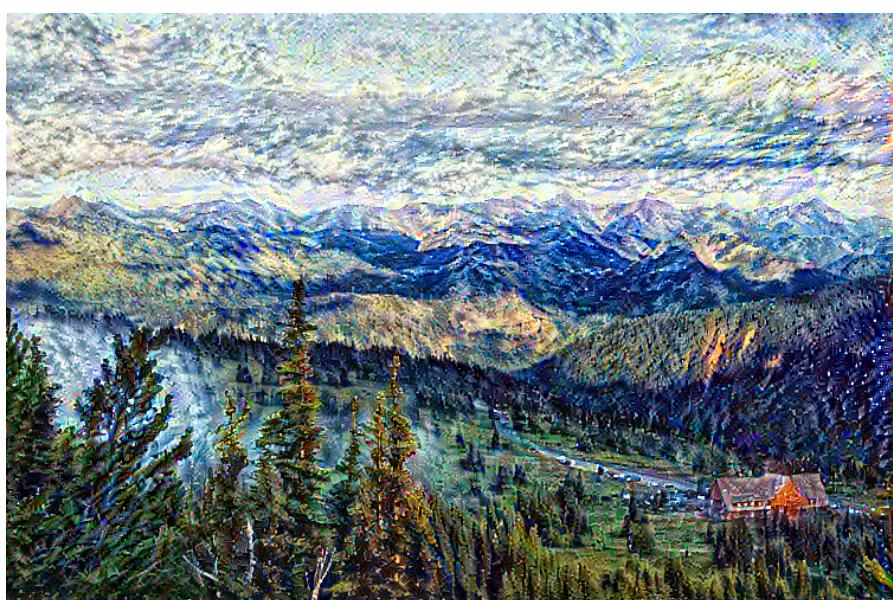


图 6: 迭 600 次

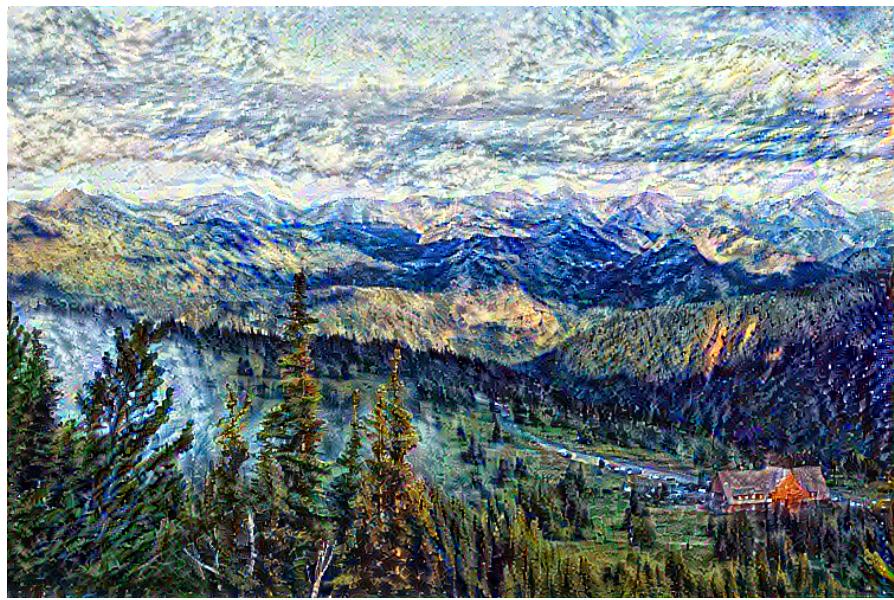


图 7: 迭 700 次

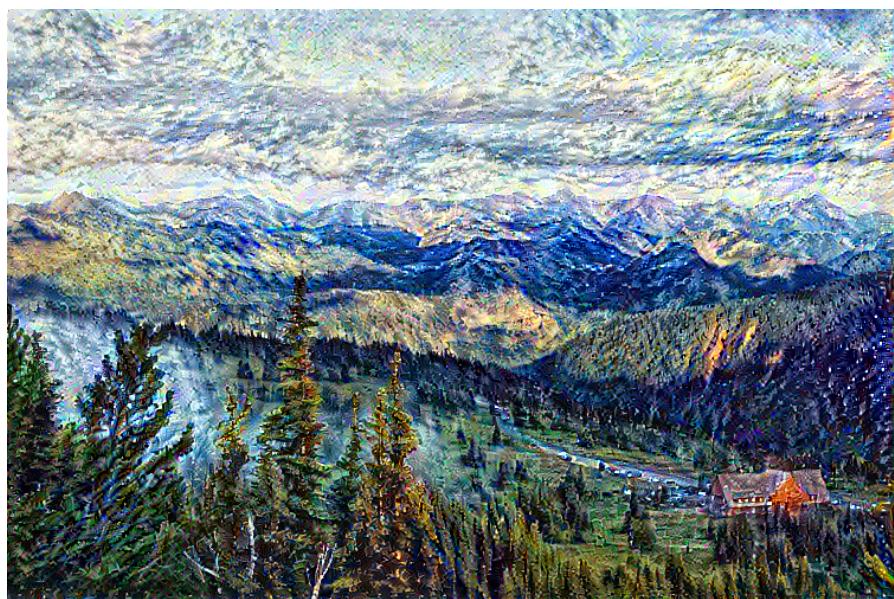


图 8: 迭 800 次

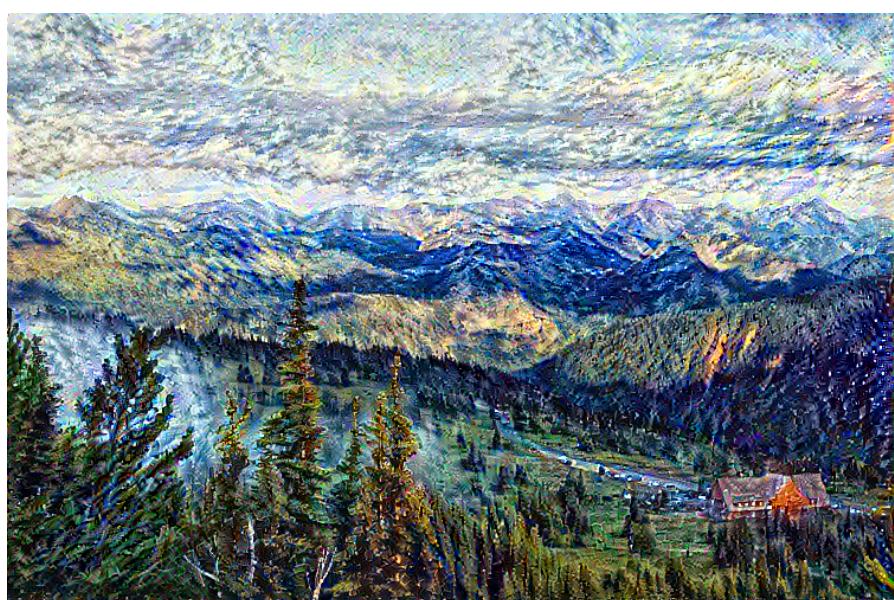


图 9: 最终结果

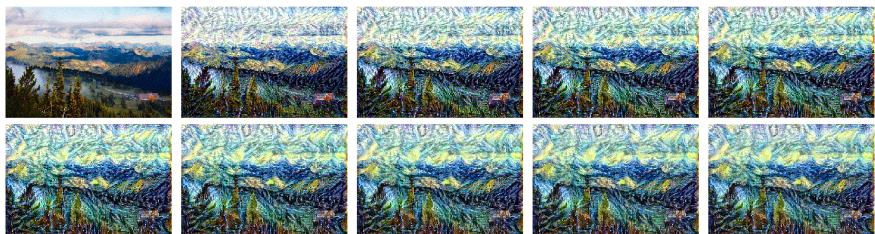


图 10: 普通版本增大 style weight 过程

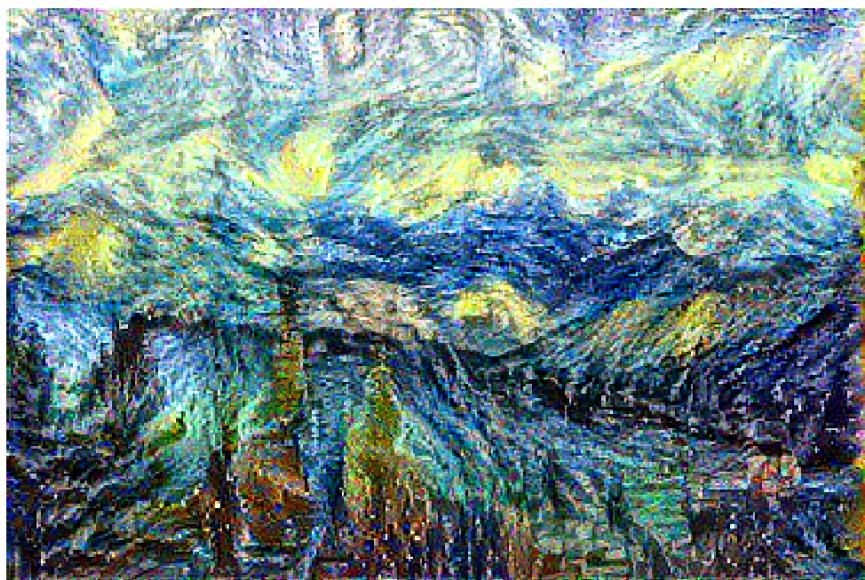


图 11: 普通版本增大 style weight 结果

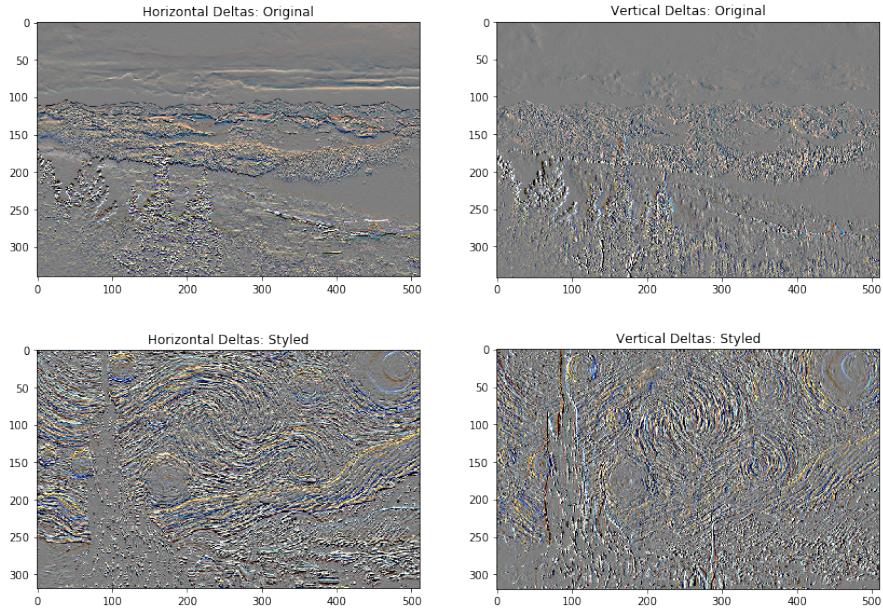


图 12: 边缘检测结果

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 13: with-hight-total-variation-weight 过程

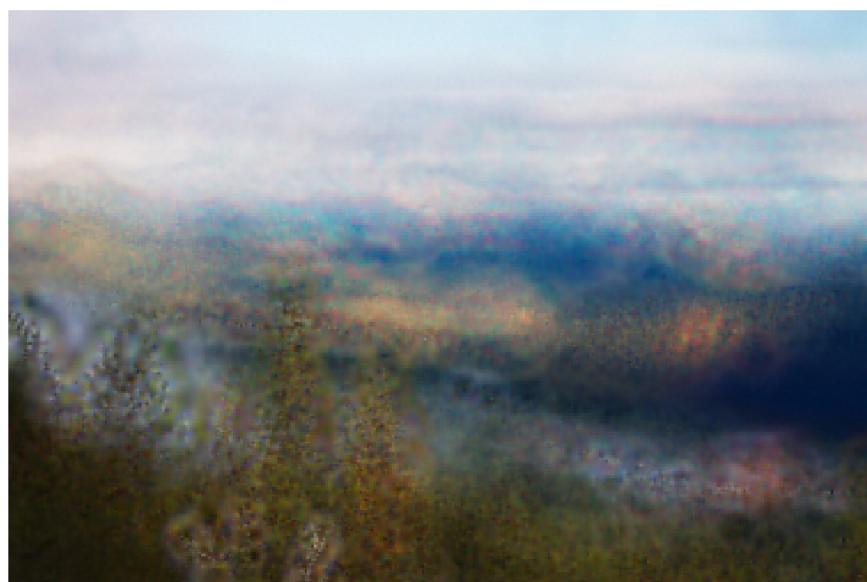


图 14: with-hight-total-variation-weight



图 15: low-total-variation-weight 过程



图 16: low-hight-total-variation-weight



图 17: 最终过程

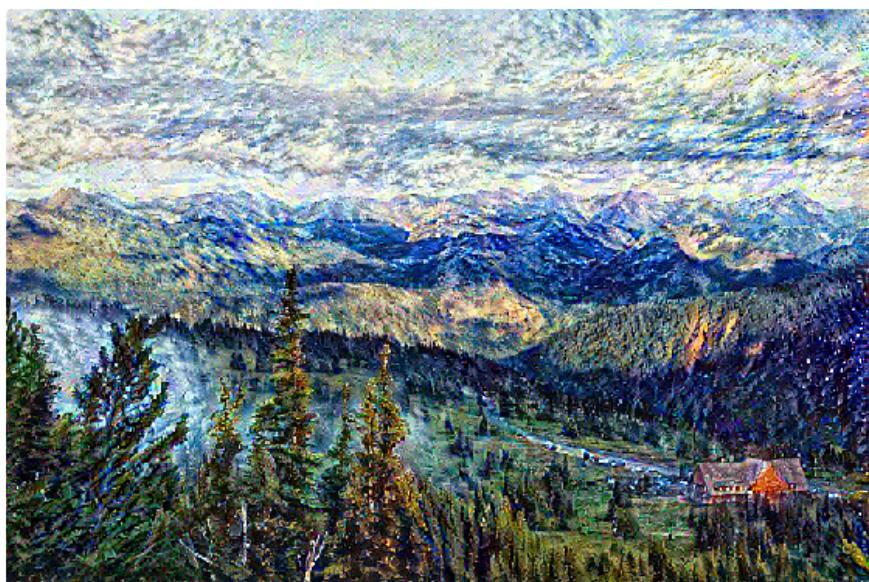


图 18: 最终结果

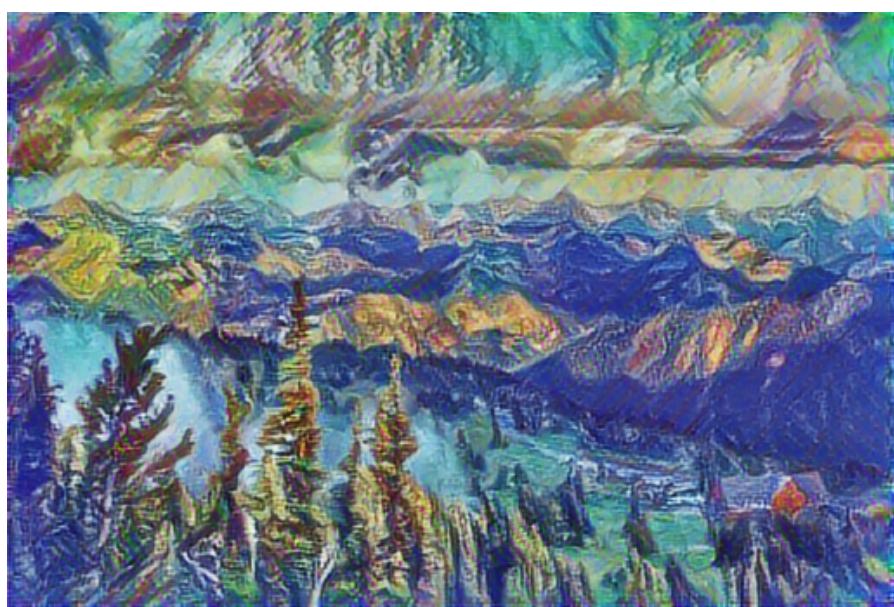


图 19: Fast Style Transfer 结果