图像快速迁移方法研究实验报告

December 31, 2023

Abstract

在本论文中，我们将探索传统和现代图像风格迁移算法，研究并复现其中三种经典的图像风格迁移方法。论文中首先论述了基于图像拼接方法的图像缝合算法以及纹理迁移算法的实现，展示了在 2001 年如何在不基于深度学习的情况下通过人工建模算法完成图像的风格迁移。然后，论文聚焦于基于卷积神经网络的现代图像风格迁移，展示了如何通过特征提取器对于风格图片以及内容图片的特征进行提取并最终完成图像风格迁移。论文将展示上述几种图像风格迁移的结果来比较这些方法在性能方面的差异。

1 引言

图像风格迁移是计算机视觉领域中广泛应用的一种流行技术，旨在将输入图像的内容与风格分离，并将其重新组合成具有合成风格的输出图像。这项技术在许多领域中具有广泛的应用，如艺术创作、图像编辑和视频游戏等。

传统的风格迁移方法主要基于手工设计的特征提取和优化算法，通常依赖于人为手工建模，其基于“纹理用图像局部特征的统计模型来描述”的思想，用复杂的数学模型和公式来归纳和生成纹理作为生成风格迁移后图片的特征。

现代的风格迁移方法是基于深度学习的兴起，深度学习模型可以自动学习输入图像的特征表示，并通过训练大规模的神经网络来实现更准确和逼真的风格迁移效果。但是其相较于传统风格迁移算法的缺陷在于较差的可解释性以及对于数据集和计算资源的大量需求。

2 传统图像迁移算法

2.1 算法内容

该算法基于图像拼接方法，其主要思想是将图像分割成多个小块，然后将这些小块连接在一起形成新的图像。然而，该算法的缺点是生成的图像可能有明显的缝合痕迹，因此其性能可能不如现代图像风格迁移的算法。这个算法主要包括两个方面：图像缝合算法（Image Quilting），纹理迁移算法（Texture Transfer）。

Image Quilting 算法是一种用于纹理合成的算法，它的输入是一个纹理块。我们从这个输入中采样一个 patch，记作 B_i ，其尺寸可由用户指定。所有可能的 B_i 的集合记作 S_B 。缝合的通用流程就是，按照栅格扫描（raster scan）的顺序（从左到右，从上到下）放置从 S_B 中采样得到的 B_i ，这样就可以拼成大片纹理。在正式介绍本文算法之前，先介绍两种简单的缝合方式：

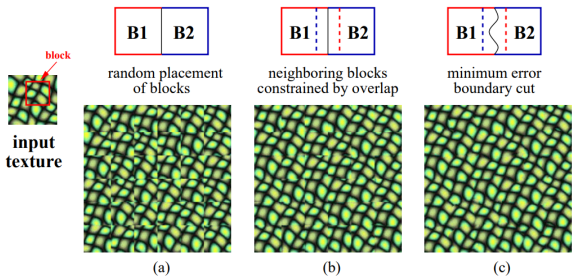

Figure 1: quilting texture.

随机抽样纹理块 + 简单并置：从 S_B 中随机采样 patch，然后按顺序直接拼在一起。然而这种方法在边界上有明显的不连续，如图一 (a)。

按相似度筛选纹理块 + 重叠并置：按照一定要求从 S_B 中采样 patch，以光栅扫描的顺序重叠放置。以第一行为例，相邻的块 B_1 和 B_2 有部分重叠，如果 B_1 已知，那么 B_2 的选取要求是：需要先在 S_B 中挑出重叠部分和 B_1 达到一定相似度的 patches，然后再从中随机挑选一个作为 B_2 重叠并置，这样做可以让两个 patch 的重叠部分尽量相似，以保证平滑过渡。最后从重合区域中线砍一刀，左侧取 B_1，右侧取 B_2。从图一 (b) 中的效果来看确实比 (a) 好很多，但还是有些许不连续的观感。

本文算法的无缝缝合，其实非常简单：就是在第二种方法的基础上，把重合部分的直线切割改进为曲线切割，这

一切割称为最小误差切割。在重合区域内，分割线左侧取 B_1，右侧取 B_2，这样可以最大程度地保证 B_1 到 B_2 纹理的连续性。现在问题的关键就是这一刀是如何切割出来的。B_1 和 B_2 在某一点之间的差异量化为 $(B_1^{ov} - B_2^{ov})^2(B_i^{ov}$ 即两者重叠部分的像素值）。现在我们要在重叠区域找一条从上边缘到底部边缘的路径，该路径每一步只能选择往左，往下或往右走一格。要求该路径经过的像素的累计误差最小，该路径正是所求的最小误差切割。这在我们算法中即是 SSD error，这条路径就是累计误差最小的最佳分割线，切割结果可以让这两片纹理拼在一起肉眼几乎察觉不出缝隙, 如图一 (c)。

纹理迁移（Texture Transfer）

纹理迁移的大部分和纹理缝合的过程一致，新的纹理块除了需要满足与已填充的纹理块重合区域的误差小于一定阈值外，还需要与待填充区域的原有图像误差小于一定的阈值（这部分来保证迁移后还能和原图像的轮廓相近），两个条件以加权和的方式组合形成总的误差，总误差小于一定阈值的块被选中。

Figure 3: Content image.

输出如下：

Figure 4: quilting texture.

2.2 风格迁移效果

我们选取了梵高自画像作为风格图片

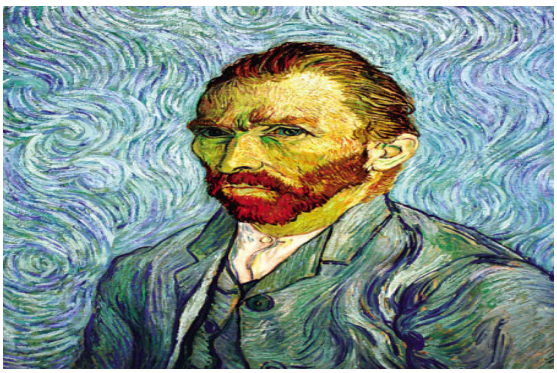

Figure 2: Style image.

选择了一张艾尔登法环的图片作为内容图片

由于算法通过将源图像分成多个小块，然后将这些小块与目标图像的相应区域进行比较，最后将它们拼接在一起来生成新的图像，生成的图像可能会出现明显的缝隙，从而影响图像的质量。我们从输出的图片也可以很明显的感觉到图像有明显的颗粒感，不像是一笔一划画出来的。加之传统算法耗时很长，训练一次需要 5-6 小时，并不具备推广或者批量迁移的能力。

3 现代图像风格迁移

3.1 固定风格固定内容的普通风格迁移

3.1.1 实验思路

固定风格固定内容的普通风格迁移的主要目的是将一张风格图片的风格应用到另一张内容图片上，最后的生成图片能既保留内容图片的大体内容，又展现出风格图片的风格 (style) 。这里的风格指一张图片的纹理、色彩、视觉模式等，而内容 (content) 则指的是图片的宏观构造。

实验中通过卷积神经网络在高层特征表示中捕捉图像的内容信息，并在不同层次上学习表示图像风格信息的特

征，然后优化生成图片来降低与内容图片的内容差异以及降低与风格图片的风格差异，迭代训练多次以后，生成的图片就会与内容图片的内容一致，同时也会与风格图片的风格一致。

具体流程图如下：

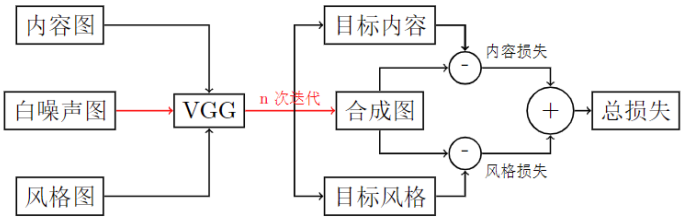

Figure 5: quilting texture.

3.1.2 特征提取

实现上述实验思路的关键就在于提取风格图片以及内容图片的特征，可以使用预训练的深度卷积神经网络作为特征提取器。具体实现中我们采用了 VGG 网络作为特征提取器。VGG 网络是一个经典的卷积神经网络，它在 ImageNet 数据集上进行了训练，并能够提取图像的高级语义特征。

给定输入图像 I，我们通过前向传播将其输入 VGG 网络，VGG 网络的不同层次描述了图像不同层次的信息：低层次描述小范围的边角、曲线，中层次描述方块、螺旋，高层次描述内容。前向传播让我们能够提取某些层次的特征图，其中包括内容特征图 F_{content} 和风格特征图 F_{style}。

有了不同层次的抽象特征后，下一步是糅合。对于生成的图片，我们希望它”骨架”接近内容图（content），”画风”接近接近风格图（style），具体是通过优化损失函数来实现。

3.1.3 损失函数

内容损失函数：内容损失函数用于衡量生成图像与输入图像的内容之间的差异。我们通过比较生成图像和输入图像在某一层的特征表示来计算内容损失。假设在第 l 层，输入图像的特征表示为 F^l_{content}，生成图像的特征表示为 P^l_{content}，则内容损失函数 L^l_{content} 可以定义为均方误差（Mean Squared Error）：

$$L^l_{\text{content}} = \frac{1}{2}\sum_{i,j}(F^l_{\text{content}}(i,j) - P^l_{\text{content}}(i,j))^2$$

其中，(i,j) 表示特征图中的位置。

将各层的内容损失函数加权求和，得到总体内容损失函数

$$L_{\text{content}} = \sum_l w_l L^l_{\text{content}}$$

实验中使用的是经过在 ImageNet 预训练过的 VGG16 提取的特征图，能够提取出图像的高级特征，通过优化生成图像和内容图像特征图的 mse_{loss}，可以迫使生成图像的内容与内容图像在 VGG16 的 $relu3_3$ 上输出相似的结果，保证了生成图像和内容图像在内容上是一致的。

风格损失函数：风格损失函数用于衡量生成图像与参考图像的风格之间的差异。与内容损失函数类似，我们通过比较生成图像和参考图像在不同层的特征表示来计算风格损失。

假设在第 l 层，参考图像的特征表示为 F^l_{style}，生成图像的特征表示为 P^l_{style}，则风格损失函数 L^l_{style} 可以定义为 Gram 矩阵的均方误差：

$$L^l_{\text{style}} = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G^l_{\text{style}}(i,j) - A^l_{\text{style}})(i,j)^2$$

其中，N_l 和 M_l 分别表示特征图的通道数和空间尺寸，$G^l_{\text{style}}(i,j)$ 和 $A^l_{\text{style}}(i,j)$ 分别表示参考图像和生成图像在第 l 层的 Gram 矩阵的元素。

Gram 矩阵的定义为：

$$G^l_{\text{style}}(i,j) = \sum_k F^l_{\text{style}}(i,k)F^l_{\text{style}}(j,k)$$

$$A^l_{\text{style}}(i,j) = \sum_k P^l_{\text{style}}(i,k)P^l_{\text{style}}(j,k)$$

将各层的风格损失函数加权求和，得到总体风格损失函数 L_{style}：

$$L_{\text{style}} = \sum_l w_l L^l_{\text{style}}$$

其中，w_l 表示层 l 的权重。

总体损失函数为了综合考虑内容和风格的差异，我们将内容损失函数和风格损失函数加权相加，构成总体损失函数 L_{total}：

$$L_{total} = \alpha L_{content} + \beta L_{style}$$

3.1.4 模型架构（特征提取器）

特征提取器选择的是 VGG16 网络，其架构如下：

输入层：输入层接收图像作为输入，通常是彩色图像，具有三个通道（红、绿、蓝）。

卷积层：VGG16 包含 13 个卷积层，这些卷积层使用不同的卷积核进行滤波操作，并通过非线性激活函数（通常是 ReLU）进行激活。卷积层的滤波器大小通常是 3x3，步幅为 1，填充为 1，以保持输入和输出的尺寸一致。不同的卷积层在网络中的位置决定了它们学习到的特征的抽象级别。越靠前的卷积层学习到的是低级特征（如边缘、纹理），越靠后的卷积层学习到的是高级特征（如物体部分、整体形状）。

池化层：在卷积层之后，VGG16 网络使用了 5 个最大池化层，将特征图的空间维度减小，同时保留重要的特征。池化层通常使用 2x2 的窗口进行最大池化操作，步幅为 2，将特征图的尺寸减半。

全连接层：在卷积和池化层之后，VGG16 包含 3 个全连接层，用于对特征进行分类。全连接层的神经元与前一层的所有神经元相连接，进行特征的整合和分类。

通过选择不同的中间层，可以捕捉不同层次的图像特征。内容损失和风格损失的计算都是基于这些特征表示进行的。

3.1.5 训练过程

- 初始化生成图像为输入图像的副本。
- 通过前向传播，将输入图像和参考图像分别输入 VGG 网络，提取内容特征和风格特征。
- 计算内容损失函数，衡量生成图像和输入图像之间的内容差异。
- 将内容损失函数和风格损失函数加权相加，构成总体损失函数。
- 通过反向传播，计算总体损失函数对生成图像像素值的梯度。
- 更新生成图像的像素值，使其逐渐接近目标结果。
- 重复执行上面五步，直到达到预定的迭代次数或损失函数收敛。

3.1.6 风格迁移效果

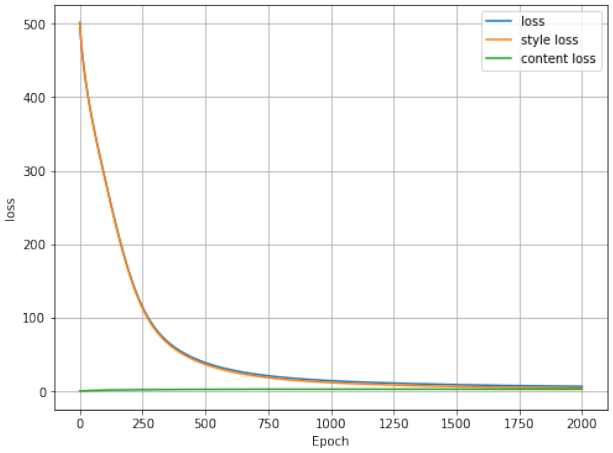

Figure 6: loss

观察损失曲线可以发现，风格损失下降很快并且迅速收敛。内容损失一直处于一个比较低的水平，因为本实验中风格迁移是基于内容图片的初始内容，并不是基于空白图片或者纯噪声，因此内容损失一直维持较低。

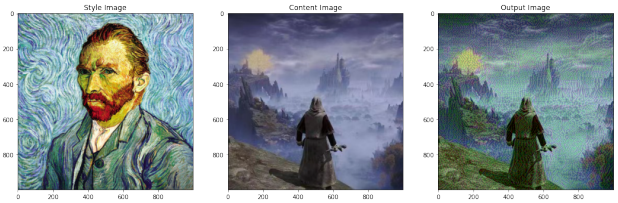

Figure 7: result

可以看出来生成图像既保留了内容图片的内容，又保留了风格图片的纹理以及色调。虽然生成图像的效果不错，但是哪怕在压缩内容图片和风格图片分辨率之后生成一张图片仍然需要十秒左右，虽然速度比起传统方法已经显著提高，但是仍有很大的提升空间，下面提出改进。

3.2 固定风格任意内容的快速风格迁移

3.2.1 实验思路

实验思路是搭建一个生成器网络，通过学习图像的低层次特征和高层次特征，最小化生成图像与目标图像在深度卷积神经网络（VGG 网络）中的中间层特征比较的感知损失来进行风格迁移。

3.2.2 损失函数

在之前所定义的内容损失以及风格损失的基础上，新引入了总变差损失 (tv loss) 用于惩罚生成图像的平滑度:

$$L_{tv} = \sum_{i,j}(\frac{1}{2}(\frac{P(i+1,j) - P(i,j)}{\sqrt{(P(i+1,j) - P(i,j))^2 + (P(i,j+1) - P(i,j))^2 + \epsilon}})^2 + \frac{1}{2}(\frac{P(i,j+1) - P(i,j)}{\sqrt{(P(i+1,j) - P(i,j))^2 + (P(i,j+1) - P(i,j))^2 + \epsilon}})^2)$$

其中，$P(i,j)$ 表示生成图像在位置 (i,j) 处的像素值，$P(i+1,j)$ 和 $P(i,j+1)$ 表示生成图像在水平和垂直方向上相邻的像素值。ϵ 是一个非常小的正数，用于避免除以零的情况。

该公式计算了生成图像在水平和垂直方向上相邻像素值的差异，并对其进行平方和归一化。通过最小化总变差损失，可以提高生成图像的平滑度，减少噪点和纹理伪影的出现。

结合内容损失以及风格损失，新的总体损失函数表达式为:

$$L_{total} = \alpha L_{content} + \beta L_{style} + \gamma L_{tv}$$

3.2.3 模型架构

定义了三个重要的模型组件：ConvLayer、ResidualBlock 和 TransformNet。

ConvLayer 函数用于创建一个卷积层，可以选择是否进行上采样、是否使用实例归一化（Instance Normalization）和是否应用 ReLU 激活函数。该函数返回一个包含上述操作的层列表。

ResidualBlock 类定义了残差块，它由两个连续的卷积层组成，每个卷积层后面可以选择是否使用 ReLU 激活函数。残差块的输入通过卷积层后，再与输入相加，实现了残差连接。这有助于减轻梯度消失问题和增强模型的学习能力。

TransformNet 类是整个图像风格迁移模型的主要部分。它包括了下采样、残差块和上采样三个阶段。下采样阶段通过多个卷积层实现图像的分辨率降低，提取图像的低层次特征。残差块阶段包含了多个残差块，用于学习图像的高层次特征。上采样阶段通过多个卷积层实现图像的分辨率提高，最终生成风格迁移后的图像。

3.2.4 模型训练

- 生成器网络通过将输入图像作为输入，并输出生成的图像作为生成图像的初始值。
- 生成器网络接受输入图像作为输入，并通过前向传播提取内容特征和风格特征。
- 计算生成器网络生成的图像与内容图像之间的内容损失、与风格图片之间的风格损失以及总变差损失。
- 将内容损失、风格损失以及总变差损失按权相加，得到总体损失。
- 通过总体损失函数对生成图像的像素值进行反向传播，计算得到生成图像像素值的梯度。
- 生成器网络根据计算得到的梯度更新生成图像的像素值，并且不断迭代使生成图像逐渐接近期望的目标结果。

3.2.5 风格迁移效果

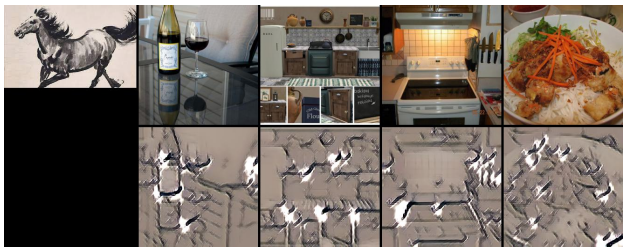

Figure 8: 15 epoch

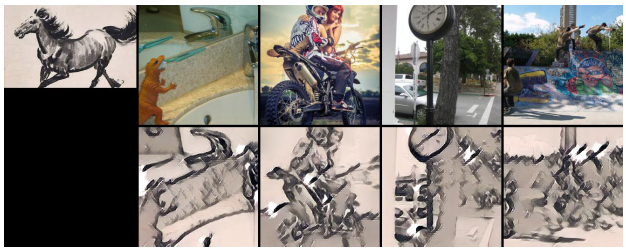

Figure 9: 30 epoch

Figure 10: 50 epoch

可以发现随着迭代次数的增加，生成图片逐渐学习到了更多风格图片的特征。并且对于固定的风格，训练的过

程实际上就是在优化定义的转换网络（TransformNet）的权值，因此在训练结束后这个转换网络可以接收任何内容图片来完成对应风格的迁移，速度相较于固定风格固定内容的风格迁移有明显提升。

3.3 两种方法的区别

3.3.1 损失函数的区别

- 固定风格固定内容的普通风格迁移中使用了两个主要的损失函数：内容损失和风格损失。内容损失是通过对比生成图像与内容图像在某些层次的特征表示之间的差异来衡量的。风格损失是通过对比生成图像与风格图像在多个层次的特征表示之间的差异来衡量的。这两个损失函数都是通过计算特征表示之间的差异来定义的。

- 固定风格任意内容的快速风格迁移中使用了更多的损失函数：内容损失、总变差损失和风格损失。总变差损失用于促进生成图像的平滑性，通过对比生成图像中相邻像素之间的差异来定义。除了内容损失和风格损失，总变差损失在图像风格迁移中起到了正则化的作用。

3.3.2 网络结构的区别

- 固定风格固定内容的普通风格迁移中使用的是一个预训练的卷积神经网络（VGG 网络）来计算图像的特征表示。该网络被分为内容层和风格层，分别用于提取内容图像和风格图像的特征表示。

- 固定风格任意内容的快速风格迁移论文中使用了一个生成器网络（TransformNet）来进行图像风格迁移。该网络包括下采样阶段、残差块阶段和上采样阶段，通过学习图像的低层次特征和高层次特征进行风格迁移。

4 结论

本实验中复现了传统的基于图像拼接的以及两种基于深度学习的风格迁移方法。传统方法时间慢，效果较差但是有较好的可解释性，并且不需要大规模数据集来预训练；基于深度学习的方法效率更高，但是需要大数据集来预训练特征提取网络。在我们复现的方法中没有去从白噪声生成图像，而是基于内容图像去给它赋上风格图片的风格，因为从之前有关 VAE 的实验中也可以知道很难去生成一个内容很贴切的图像。事实上，如果从白噪声去生成图像，那么本实验中得到的只有简单的线条和隐约的轮廓，这样与风格迁移的根本目的相背离，因为风格迁移并不一定要完全重新生成图片，而是可以基于内容图片来完成，事实证明我们的实验效果也比较理想。

本次实验不仅让我们更加深入了解了风格迁移的发展历史，更让我们认识到了其广阔的应用前景，以及发展需要解决的问题。首先，风格迁移算法的效果往往依赖于训练数据的质量和数量，而获取高质量的训练数据并不容易。其次，当前的风格迁移算法往往需要大量的计算资源，这在一定程度上限制了其在移动设备和嵌入式系统上的应用。此外，如何在保持内容信息的同时进行风格迁移，以及如何实现多风格迁移，也是当前研究的热点和难点。

尽管存在这些挑战，但随着深度学习技术的不断发展，风格迁移的应用将会更加广泛，其性能也会得到进一步提升，在未来带来更多的创新和惊喜。

References

[1] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer[C], Proceedings of the 28th annual conference on Computer graphics and interactive techniques. 2001: 341-346.

[2] Gatys A L ,Ecker S A ,Bethge M .A Neural Algorithm of Artistic Style.[J].CoRR,2015,abs/1508.06576

[3] Johnson J ,Alahi A ,Li F .Perceptual Losses for Real-Time Style Transfer and Super-Resolution.[J].CoRR,2016,abs/1603.08155

[4] jcjohnson. fast-neural-style. 2016. https://github.com/jcjohnson/fast-neural-style/blob/master/doc/training.md.

[5] Cat food. 纹理合成：Image Quilting for Texture Synthesis and Transfer 论文解读. 2021. https://zhuanlan.zhihu.com/p/401031355.

[6] 李嘉铭. 图像风格迁移 (Neural Style) 简史. 2018. https://zhuanlan.zhihu.com/p/26746283.

[7] tezansahu. Image Quilting for Texture Synthesis Transfer. 2020. https://github.com/tezansahu/ImageQuilting/blob/master/README.md.

[8] ypwhs. StyleTransferTrilogy. 2018. https://github.com/CortexFoundation/StyleTransferTrilogy?tab=readme-ov-file.