同濟大學
TONGJI UNIVERSITY

多媒体技术课程报告

项目名称：基于深度学习的图像风格迁移

学　　院：电子与信息工程学院

专　　业：计算机科学与技术

组　　号：Group 33

组　　员：1553058 肖昊坪

1652302 陈佳莉

授课老师：王瀚漓

2019 年 05 月 30 日

目录

一、选题意义与背景介绍

1.1 背景介绍

图像风格迁移是一种用不同风格渲染图像语义内容的图像处理方法。图像风格迁移算法的输入有二，分别是内容图与风格图；输出为一个，为风格迁移后的结果图。

在非真实感图形学领域，图像风格迁移技术可以大体分为基于笔触渲染的方法、基于图像类比的方法、基于图像滤波的方法。但是这些图形学技术却都没有大规模落地，它们无法大规模落地的原因如下：基于笔触渲染的方法，在算法设计之前就会确定某一风格，即每一个基于笔触渲染的算法一般只对应于一个风格，不能简单地扩展到其他风格；基于图像类比的方法，需要很多成对的原图和风格结果图作为训练集，但是对所有风格图都找到大量成对的数据集不现实；基于图像滤波的方法，这个方法的速度快且效果稳定，是可以落地实现的，但是这个方法能模拟的风格种类有限。

在视觉领域，图像风格迁移一般被认为是纹理合成的一个扩展问题。（纹理合成是给定一个纹理图，去合成更多的类似的纹理结构，最终组成一个大的纹理图）风格迁移中的风格图实际上就可以看作是一种纹理，所以当时的图像风格迁移叫纹理迁移。但是由于纹理迁移是基于低层次的图像特征实现的，没有考虑语义信息，因此结果不尽人意。

在非真实感图形学领域和视觉领域提及的方法都距大规模落地有一段距离，但是还好很多前人依旧继续扎实的基础研究，最终总结得出纹理建模的方法与图像重建的方法。图像风格迁移也就是源于纹理建模与图像重建，在神经网络深度学习的基础上发展起来，纹理建模与图像重建的方法将会在相关方法中介绍。

1.2 选题意义

基于深度学习的图像风格迁移的发展，使得图像风格迁移的效果得到很大的提升，具有广阔的应用前景。目前，基于深度学习的图像风格迁移可以用于以下几个方向：

1)图像处理：图像风格迁移可以用于图像美化，传统的图像处理技术只能对图像进行模式较固定的处理，而基于深度学习的图像风格迁移可以给图像风格设计带来更多的可能性。比如：可以用于图像修复、用于给漫画草图上色等。最常见的应用是 Prisma，这是免费提供基于深度学习的图像风格迁移服务的 App。

2)风格设计的辅助工具：图像风格迁移也可以在艺术绘画创作、建筑艺术设计、服装艺术设计等场景中充当有效的辅助工具。

3)视频处理：可以将图像风格迁移技术用于影视特效技术的创作，用于对电影进行风格化等。

二、相关方法介绍

2.1 纹理建模

纹理建模主要研究如何表示一种纹理，常用方法可以分为两大类：基于统计分布的参数化纹理建模方法、基于 MRF 的非参数化纹理建模方法。

基于统计分布的参数化纹理建模方法主要将纹理建模为 N 阶统计量，基于 MRF 的非参数化纹理建模方法主要是用 patch 相似度匹配进行逐点合成。纹理建模解决了图片风格迁移中“如何对风格图中的风格特征进行建模和提取”的问题。

2.2 图像重建

图像重建的输入是特征表达，输出是特征表达对应的图像；即把某个特征逆向重建为原来的图像。通过重建预训练的分类网络中的高层特征，可以发现重建结果中能保留高层语义信息，而摒弃了低层的颜色等信息。因此，我们将图像重建用于“与内容混合然后还原成一个相应的风格化结果”。图像重建方法可以分为两类：基于在线图像优化的慢速图像重建方法、基于离线模型优化的快速图像重建方法。

基于在线图像优化的慢速图像重建方法是在图像像素空间做梯度下降来最小化目标函数，可以看作：由随机噪声作为起始图，然后不断迭代改变图片的所有像素值来寻找一个目标结果图，这个目标结果图的特征表达和作为重建目标的目标特征表达相似。但是由于每个重建结果都需要在像素空间进行迭代优化很多次，这个方法十分耗时。

基于离线模型优化的快速图像重建方法，这个方法就是因为第一个方法太慢了，为了加速重建，大家希望能设计一个前向网络，提前用很多训练数据进行训练，训练过程将一个特征表达作为输入，重建结果图像作为输出。这个训练好的网络只需要一次前向就能输出一个结果图。

2.3 图像风格迁移

图像风格迁移就是将纹理建模方法与图像重建算法结合。下面将介绍图像风格迁移领域的相关方法。

2.3.1 基于在线图像优化的慢速图像风格化迁移算法

1）基于统计分布的参数化慢速风格化迁移算法：其中代表性的就是图像风格迁移领域的开山鼻祖 *Gatys* 等人[1]提出基于神经网络的风格迁移。这种方法的过

程简单地说，就是由随机噪声作为起始图，然后不断迭代改变图片的所有像素值来寻找一个目标结果图 x’，这个目标结果图的特征表达和我们作为重建目标的目标特征表达 Φ(x) 相似，即像素迭代的目标为 Φ(x’)≈Φ(x) 。

由于每个重建结果都需要在像素空间进行迭代优化很多次，这种方式是很耗时的（几百乘几百的图需要几分钟），尤其是当需要的重建结果是高清图的时候，占用的计算资源以及需要的时间开销很大。

2)基于 MRF 的非参数化慢速风格化迁移算法：其[7]核心思想是提出了一个取代 Gram 损失的新的 MRF 损失。思路与传统的 MRF 非参数化纹理建模方法相似，即先将风格图和重建风格化结果图分成若干 patch，然后对于每个重建结果图中的 patch，去寻找并逼近与其最接近的风格 patch。

相比[1]的方法，优势在于：当风格图不是一幅艺术画作，而是和内容图内容相近的一张摄影照片，这种基于 patch 匹配的方式可以很好地保留图像中的局部结构等信息。

2.3.2 基于离线模型优化的快速图像风格化迁移算法

核心思想就是利用基于离线模型优化的快速图像重建方法对风格化结果进行重建，基预先训练前向网络来解决计算量大、速度慢的问题。

根据一个训练好的前向网络能够学习到多少个风格作为分类依据，我们将快速图像风格化迁移算法分为单模型单风格（PSPM）、单模型多风格（MSPM）和单模型任意风格（ASPM）的快速风格化迁移算法。

1） 单模型单风格[2]主要想法是针对每一个风格图，我们去训练一个特定的前向模型，训练数据可以用 COCO 的 8 万张图，损失函数和 Gatys[1]的慢速风格化算法相同，用 Gram 统计量来进行风格建模。这样当测试的时候，我们只需要向前向模型扔进去一张内容图，就可以前向出一个风格化结果了。

缺点是对于每一个风格都要训练一个模型，有上万、百万的风格的话，所有模型占用的空间非常大。

2） 单模型多风格主要想法是发掘出不同风格网络之间共享的部分，然后对于新的风格只去改变其有差别的部分，共享的部分保持不变。Dumoulin 等人[8]发现在训练好的一个风格化网络基础上，用 CIN 层做一个仿射变换，只需要把 CIN 层中仿射变换的参数与每一个风格进行绑定，每个新风格只需要去训练这些参数，其余部分保持不变，就可以得到一个具有完全不同风格的结果。

但是虽然随着风格的增加，参数增加的也就更多，模型大小跟着变大。所以不能支持任意风格。

3） 单模型任意风格最早是 Tian Qi Che[9]提出的， 基本思想是在 CNN 特征空间中，找到与内容 patch 匹配的风格 patch 后，进行内容 patch 与风格 patch

的交换（Style Swap），之后用快速图像重建算法的思想对交换得到的 feature map 进行快速重建。但由于 style swap 需要一定的时间开销，所以没有达到实时。

第一篇能实现实时的任意风格迁移算法是由 X. Huang[3]提出的。文章主要是受 CIN 的启发，提出一个 AdaIN 层。AdaIN 的输入是通过 VGG 提取的风格和内容特征，用数据驱动的方式，通过在大规模风格和内容图上进行训练，让 AdaIN 能够直接将图像中的内容 normalize 成不同的风格。

2.4 图像风格迁移效果评估

传统的的图像风格迁移采用的评估方法往往是采用以下指标：

1.速度内存：通过选取一定数量的风格图片和内容图片，在相同的机器上，用不同的方法做风格迁移，记录每种方法的平均每张图片的处理时间、平均占用内存。用这两个指标来反应速度，从而判断方法的好坏。

2.风格多样性：通过评价风格迁移所支持的风格数量，来衡量一个方法的好坏。

3.内容、风格损失值：通过风格训练时的风格、内容损失值的大小，收敛情况来判断图片质量的好坏

4.艺术家评判：通过邀请几个专家（Sanakoyeu 等人[10]中邀请了 3 个专家），让专家选出风格迁移后效果最好的，即内容图完美融合了风格图片的风格。通过（选出的图片数量/总样本数）比值来反应风格迁移的质量。

由于能够邀请的专家有限，因此，文章采用了风格欺骗率来进一步评价。采用训练过的 VGG16 对 Wikiart 上 624 个艺术家的作品进行分类。欺骗率是计算风格迁移的图片是否能被模型分成艺术家的作品（能被分成艺术家的作品/样品总数）。通过比较不同方法的风格欺骗率来评价效果质量。

5. **用户研究：**通过邀请用户（普通人）评判，通过统计用户的评价来评价风格迁移的质量。但是由于普通用户没有受过专业知识的训练，因此很难用某种方法去判断，完全靠主观意识。

三、具体方法介绍

纵观第二部分的相关方法，我们了解到图像风格迁移从 *Gatys* 提出基于神经网络的风格迁移以来，经过不断发展，从在线图像慢速迁移到离线的单模型任意风格，不仅在速度上有了 3 个数量级的优势，而且支持任意的内容图片和风格图片。

我们此次报告主要是根据论文 *Image Style Transfer Using Convolutional Neural Networks[1]* 复现图像风格迁移的开山之作，然后再根据论文 Arbitrary Style Transfer in Real-time with Adaptive Instance Normalizatio[3]复现实时多风格的图像迁移。并且，在这两篇论文中，应用的评估方法是速度、多样性、损失值，这并非真正意义上的图像风格迁移效果评估。因此，我们通过阅读 *Quantitative Evaluation of Style Transfer[11]*这篇文献，尝试定性地评估这两种方法。

3.1 基于卷积神经网络的图片风格迁移

本方法参考的是发表在 CVPR 2016 中的一篇论文 *Image Style Transfer Using Convolutional Neural Networks*[1]。

深度卷积神经网络的出现，产生了强大的计算机视觉系统，可以从图像中学习提取高层语义信息，应用于图片风格迁移的任务中。因此在图片风格迁移任务中，我们通过使用卷积神经网络学习图片的一般特征表示，独立处理和操作图像的内容和风格，最终实现风格迁移。

我们参考论文，使用的卷积神经网络为公开的标准 19 层 VGG 网络，其中包含 16 个卷积层和 5 个池化层；但是我们不使用任何的全连接层。图 3.1 中显示了在卷积神经网络中的图像表示，给定的输入图像在网络中的每个处理阶段都表示为一组经过过滤的图像；虽然在网络的处理层次中，过滤器的数量增加了，但是过滤后也会通过一些下采样机制（池化层）减小了图像的大小，进而使得网络中每层的单元总数减少。

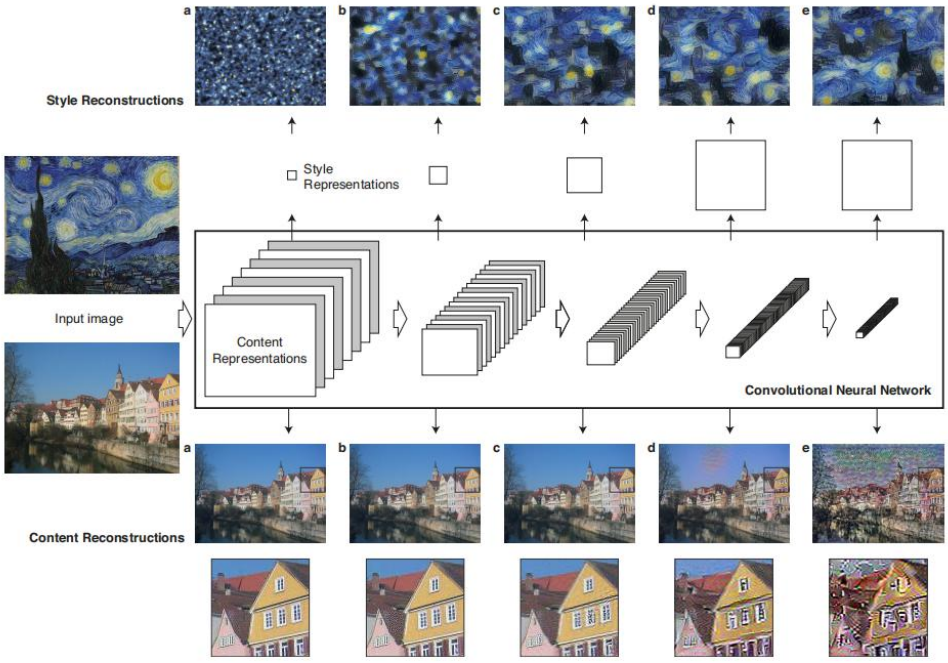


图 3.1：卷积神经网络中的图像表示

引用来源：《Image Style Transfer Using Convolutional Neural Networks》图 1.卷积神经网络中的图像表示

下面将会详细介绍如何在卷积神经网络中进行内容图像表示、风格图像表示和风格转移方法。

3.1.1 内容图像表示

给定一个图像 $\vec{\mathbf{x}}$，卷积神经网络在每一层中使用过滤器对图像进行过滤编码，在 l 层中有 N_l 个不同过滤器的层会输出 N_l 个特征图，每一个特征图的大小为 M_l，因此可以将 l 层的输出存储在矩阵 F^l 中，F^l 在 $R^{N1\times M1}$ 的特征空间中，其中 F^l_{ij} 为 l 层中第 i 个过滤器在位置 j 上的激活值。

为了可视化不同层次结构中编码的图像信息，可以对初始图像（可以设置为白噪声图像）进行梯度下降，以找到与图像 $\vec{\mathbf{x}}$ 的特征相应匹配的一幅图像。为了在卷积神经网络的不同处理阶段将信息可视化，可以从特定的一个网络层上的响应重建初始输入图像。如在图 3.1 中所示，我们可以在 VGG-19 网络中的‘conv1-2’‘conv2-2’‘conv3-2’‘conv4-2’‘conv5-2’重建输入图像，重建结果分别对应图 3.1 下半部分的(a)(b)(c)(d)(e)。从图中的黄色三角房子中可以发现，(a)(b)(c)与输入图像的匹配最相似，即在低层网络中重建越接近原始输入图像；(d)(e)已经

不完全与输入图像一样了，即在高层网络中重建，细节像素信息可能会发生丢失，但是高层的图像内容（即对象的基本结构）会被保留下来。

令$\vec{\mathbf{p}}$和$\vec{\mathbf{x}}$分别为原始图像与重建图像，P^l和F^l分别是原始图像与重建图像在 l 层的特征表示，可以定义两个图像特征表示之间的误差平方损失函数为：

$$\mathcal{L}_{\text{content}}(\vec{p},\vec{x},l)=\frac{1}{2}\sum_{i,j}\left(F_{ij}^{l}-P_{ij}^{l}\right)^{2}.$$

损失函数的偏导数对应的 l 层的激活函数为：

$$\frac{\partial\mathcal{L}_{\text{content}}}{\partial F_{ij}^{l}}=\begin{cases}\left(F^{l}-P^{l}\right)_{ij} & \text{if } F_{ij}^{l}>0\\ 0 & \text{if } F_{ij}^{l}<0,\end{cases}$$

重建图像$\vec{\mathbf{x}}$的梯度可以通过标准误差反馈传播计算，如图 3.2 右侧，因此我们可以不断迭代改变初始输入图像$\vec{\mathbf{x}}$，直到$\vec{\mathbf{x}}$与$\vec{\mathbf{p}}$的特征表示相同或者误差较小。

总结来说，卷积神经网络可以生成对内容图像的表示，在网络不同层级的处理过程中会对输入图像的真正内容结构越来越敏感，对确切的细节变得相对不敏感。高层网络会保留图像的高层语义信息，比如物体的基本结构，但是不会精确保留像素值信息，在图 3.1 中的(d)(e)中可以看到；低层网络只是简单的保留原始图像中的像素值，在图 3.1 中的(a)(b)(c)中可以看到。因此我们将网络中的高层特征输出作为内容图像的表示。通过误差反馈传播迭代得到与内容图像高层特征相似的重建图像。

3.1.2 风格图像表示

我们通过特征空间获得图像的风格表示，特征空间可以在网络的任意层的过滤器输出上构建，它由不同过滤器输出之间的相关性组成。l 层上的特征关系，可以用 gram 矩阵G^l表示，G^l在特征空间$R^{N1\times N1}$上，其中：

$$G_{ij}^{l}=\sum_{k}F_{ik}^{l}F_{jk}^{l}.$$

通过包含多层的特征关联，我们可以得到图像的平稳多尺度表示，它能够捕获图像的纹理信息，但没有全局结构信息。与内容图像表示类似，我们在网络不同层可视化特征信息，如图 3.1 中上侧(a)(b)(c)(d)(e)，它们是分别在网络层“conv1-1”“conv1-1、conv2-1”“conv1-1、conv2-1、conv3-1”“conv1-1、conv2-1、conv3-1、

conv4-1”“conv1-1、conv2-1、conv3-1、conv4-1、conc5-1”上构建的风格表示。可以看到随着网络层的增加，构建的风格表示会逐步匹配图像的风格，而丢失全局结构信息。因此我们可以在初始输入图像（白噪声图像）上使用梯度下降算法最小化原始风格图像的 gram 矩阵与重建图像的 gram 矩阵的距离来实现图像重建。

令 $\vec{\mathbf{a}}$ 和 $\vec{\mathbf{x}}$ 分别为原始图像与重建图像，A^l 和 G^l 分别是原始图像与重建图像在 l 层的特征表示，那么 l 层相对于总风格损失的贡献为：

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

总的风格损失函数为：

$$\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l,$$

其中 w_l 是网络中每层对于总损失函数的贡献权重。损失函数的偏导数对 l 层的激活函数为：

$$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left((F^l)^{\mathrm{T}} \left(G^l - A^l\right)\right)_{ji} & \text{if } F_{ij}^l > 0 \\ 0 & \text{if } F_{ij}^l < 0 \,. \end{cases}$$

E_l 对重建图像 $\vec{\mathbf{x}}$ 的梯度可以通过标准误差反馈传播计算，如图 3.2 左侧，因此我们可以不断迭代改变初始输入图像 $\vec{\mathbf{x}}$，直到 $\vec{\mathbf{x}}$ 与 $\vec{\mathbf{a}}$ 的特征表示相同或者误差较小。

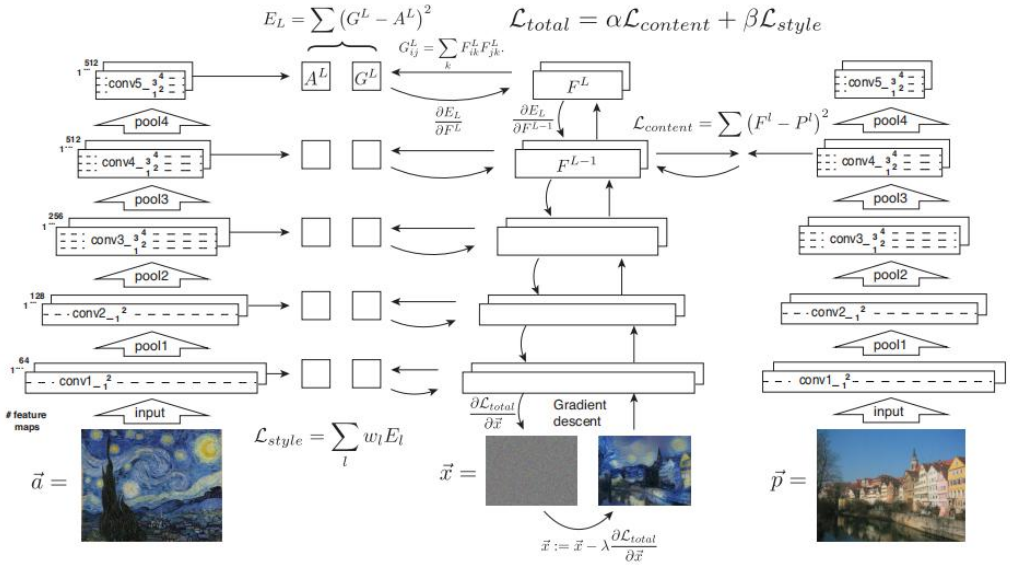


图 3.2 风格迁移算法

引用来源：《Image Style Transfer Using Convolutional Neural Networks》图 2.风格迁移算法

3.1.3 风格转移

为了将风格图片$\vec{\mathbf{a}}$的风格转移到内容图片$\vec{\mathbf{p}}$上，我们同时匹配$\vec{\mathbf{a}}$的风格表示和$\vec{\mathbf{p}}$的内容表示，重建一个新图像，如图 3.2。在图 3.2 中的风格迁移算法中，首先提取和存储内容图像与风格图像的特征，风格图像$\vec{\mathbf{a}}$通过网络传递，在包含的所有层上的特征表示 A^l被计算和存储，如图 3.2 左。内容图像$\vec{\mathbf{p}}$通过网络传递，在某一层中的特征表示 P^l被计算和存储，如图 3.2 右。

随机白噪声图像$\vec{\mathbf{x}}$通过网络，计算其风格特征 G^l与内容特征 F^l，在风格特征表示中包含的每一层上计算 G^l与 A^l的差平方，得到风格损失如图 3.2 左。同时计算 F^l与 P^l的差平方，得到了 1 的内容损失，如图 3.2 右。总损失是内容损失与风格损失之间的线性组合，它对像素值的导数用于误差反向传播的计算，如图 3.2 中间，这个误差反向传播用于迭代更新图像$\vec{\mathbf{x}}$，直到$\vec{\mathbf{x}}$同时匹配风格特征与内容特征。

我们共同将白噪声图像的特征表示与一层的内容表示和风格表示的距离最小化，最小化的损失函数为：

$$\mathcal{L}_{\text{total}}(\vec{p},\vec{a},\vec{x}) = \alpha\mathcal{L}_{\text{content}}(\vec{p},\vec{x}) + \beta\mathcal{L}_{\text{style}}(\vec{a},\vec{x})$$

其中 α 和 β 分别是内容和风格重建的权重因子，我们将对像素值$\frac{\partial\mathcal{L}_{\text{total}}}{\partial\vec{x}}$作为 L-BFGS 的输入值，L-BFGS 优化策略可以非常好的应用于图像合成中。

3.2 基于 AdaIN 层的实时任意风格迁移

本方法参考的是发表在 ICCV 2017 中的一篇论文 *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization*[3]。

虽然 Gatys 等人[1]提出了基于卷积神经网络的图片风格迁移，但是需要缓慢的迭代优化过程，这限制了其实际应用。后来 Johnson 等[2]提出了一种基于前馈神经网络的快速逼近方法，以加快神经网络的传输速度。不幸的是，速度的提高需要付出代价：网络通常绑定到一组固定的样式，无法适应任意的新样式。

而这篇论文首次实现了任意样式的实时传输。方法的核心是一个的自适应实例规范化（AdaIN）层，它将内容特征的均值和方差与风格特征的均值和方差对齐。它实现了与现有最　　快方法相媲美的速度，却不受对预定义风格集的限制。此外，我们的方法

允许用户进行灵活的控制，如内容-风格比值权衡，风格插值，颜色和空间控制等，所有这些都使用单一的前馈神经网络。

论文在 CIN 的基础上做了一个改进，提出了 AdaIN（自适应 IN 层）。顾名思义，就是自己根据风格图像调整缩放和平移参数，不在需要像 CIN 一样保存风格特征的均值和方差，而是在将风格图像经过卷积网络后计算出均值和方差。

3.2.1 背景知识介绍

为了更好地理解 AdaIN，有必要简单介绍一下此前的一些工作。

1) Batch Normalization

Ioffe and Szeged[4]在论文中引入了批量归一化层（BN），这个 BN 层通过归一化特征数据来加速前馈网络的训练，同时在生成图像建模中也起到一定的作用。归一化一批样例以一个单一风格为中心，但是每个样本仍然可能有自己的风格。具体计算如下：

$$\mathrm{BN}(x) = \gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \beta \tag{1}$$

where $\gamma, \beta \in \mathbb{R}^C$ are affine parameters learned from data; $\mu(x), \sigma(x) \in \mathbb{R}^C$ are the mean and standard deviation, computed across batch size and spatial dimensions independently for each feature channel:

$$\mu_c(x) = \frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw} \tag{2}$$

$$\sigma_c(x) = \sqrt{\frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw}-\mu_c(x))^2 + \epsilon}$$

2) Instance Normalization

Ulyanov 等人[5]提出归一化每个样例（IN）有更好的效果，在和 BN 层的对比实验中，在同样的迭代次数下，IN 层总是比 BN 层有更小的损失值。

$$\mathrm{IN}(x) = \gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \beta \tag{4}$$

Different from BN layers, here $\mu(x)$ and $\sigma(x)$ are computed across spatial dimensions independently for each channel *and each sample*:

$$\mu_{nc}(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw} \tag{5}$$

$$\sigma_{nc}(x) = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw}-\mu_{nc}(x))^2 + \epsilon}$$

总的来说 batch normalization 是对一个 batch 里所有的图片的所有像素求均值和标准差。而 instance normalization 是对单个图片的所有像素求均值和标准差。

3) Conditional Instance Normalization

条件实例归一化，Dumouli 等人[6]在论文中主要提到的方法。网络可以通过使用相同的卷积参数来生成完全不同风格的图像，而只需要对归一化的结果进行一个平移和缩放，平移和缩放就是 β^s 和 γ^s(s 代表风格)，每一个风格就是要学习这两个参数。训练多个风格就需要多组数据，如有 c 个 feature maps 和需要训练 N 个风格，那么总共就有 2*N*C 个参数需要训练。

$$\mathrm{CIN}(x;s)=\gamma^s\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta^s$$

3.2.2 AdaIN 介绍

如果 IN 可以将输入规范化为仿射参数指定的单一风格，是否可以通过使用自适应仿射变换使其适应任意给定的风格？基于这个想法，论文[3]提出 AdaIN 层，AdaIN 层输入内容图片 x 和风格图片 y，通过对齐内容图每通道的 feature map 的均值和方差来匹配风格图每通道 feature map 的均值和方差。和 BN，IN，CIN 不同的是，AdaIN 的仿射参数不需要通过训练，它可以自适应地从风格图片去计算仿射参数。

$$\mathrm{AdaIN}(x,y)=\sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\mu(y)$$

简单地说，自适应实例归一化，就是把 content image 的 feature 分布拉到 style image 的 feature 分布，这样就在特征空间完成了风格变换。而由于 stlye image 可以任意输入，可以实现任意风格变换。这篇论文中采用的是预训练好的 vgg 网络 relu4_1 的 feature 空间结果作为输入的。

而 AdaIN 和 CIN 很大的区别是参数不需要进行训练。

3.2.3 网络结构

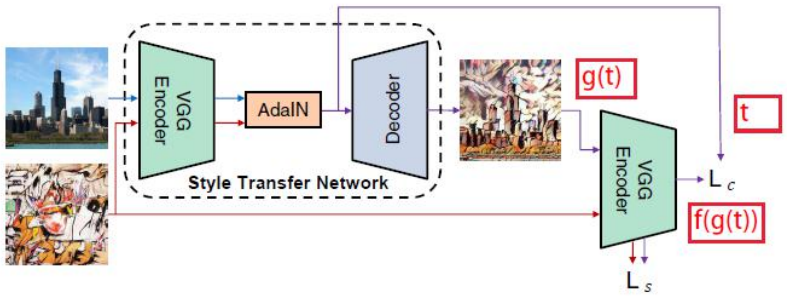


图 3.3 风格迁移算法网络示意图

引用来源：《Arbitary Style Transfer in Real-time with Adaptive Instance Normalization》图 2.风格迁移概览

网络结构主要分为两个部分：生成网络即 style transfer network 和计算损

耗网络。生成网络是一个前向网络，后期用来进行风格转换。计算损耗网络是用来训练时进行约束的。风格转换生成网络由 Encoder-AdaIN-Decoder 这 3 部分组成。

Encoder 部分是采用预训练好的 VGG 网络，只使用到了 Relu4_1，将风格和内容图的图像都从图像空间转到特征空间。

Decoder 部分是一个将 feature 空间转成图像空间的网络，这部分网络采用和 encoder 对称的网络结构，整个网络中需要训练的就是这部分网络的权重参数信息，初始可以随机一些初始化参数，通过梯度下降可以不断进行更新参数以使整个损耗函数比较小、网络逐渐收敛。池化层一般是替换成采用最近邻上采样的方式来防止棋盘效应，在 encoder 和 decoder 部分的 padding 采用反射填充避免边界 artifacts。decoder 中没有使用归一化层，因为 IN/BN 这些实例归一化和批归一化都是针对单个风格的。

内容图和风格图通过 Encoder，取 Relu4_1 层的 feature map，喂入 AdaIN 层，AdaIN 层将对齐内容图每通道的 feature map 的均值和方差来匹配风格图每通道 feature map 的均值和方差。这个过程用公式表示如下：

$$t = \mathrm{AdaIN}(f(c), f(s))$$

然后 AdaIN 层输出的 t 送入 Decoder，将 t 从特征空间转化成图像空间，就生成了最后的图像。这个过程用公式表示如下：

$$T(c, s) = g(t)$$

3.2.4 训练

内容图集：Microsoft COCO dataset（选取其中的 8W 张图片）

风格图集：WikiArt dataset （选取其中的 8W 张图片）

损耗函数主要由两部分组成：内容损耗以及风格损耗。和最早 Gatys 等人[1]提出的方法一致，也是采用预训练好的 vgg 网络的特征 maps 进行计算损耗。

$$\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_s$$

内容损耗：风格转换后图像在 vgg 网络中 Relu4_1 的特征和 adaIN 输出 feature maps 的欧式距离。

$$\mathcal{L}_c = \|f(g(t)) - t\|_2$$

风格损耗：没有采用 Gram 矩阵这种方式，即使会产生相似的结果。因为在 AdaIN 层只传递了风格特征的均值和方差，所以风格损耗只做了这些数据的匹配。同样也采用了 relu1_1,relu2_1,relu3_1,relu4_1 四层的 feature maps。即风格损耗只

是基于 IN 统计的损耗。

$$\mathcal{L}_s = \sum_{i=1}^{L} \|\mu(\phi_i(g(t))) - \mu(\phi_i(s))\|_2 \quad + \\ \sum_{i=1}^{L} \|\sigma(\phi_i(g(t))) - \sigma(\phi_i(s))\|_2$$

3.2.5 灵活性

即使我们在训练的时候，通过设置 λ 的值来控制内容和风格的比重。但是我们在使用的时候，可以利用如下公式，达到不用训练也可以控制内容和风格的比重。

$$T(c, s, \alpha) = g((1-\alpha)f(c) + \alpha \mathbf{AdaIN}(f(c), f(s)))$$

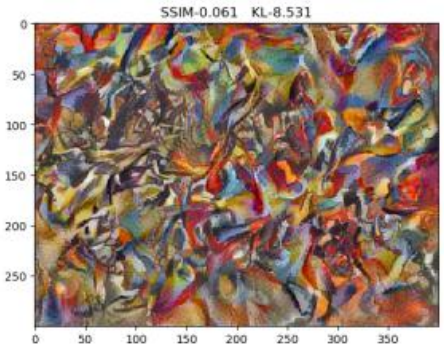
四、实验结果

4.1 基于卷积神经网络的图片风格迁移

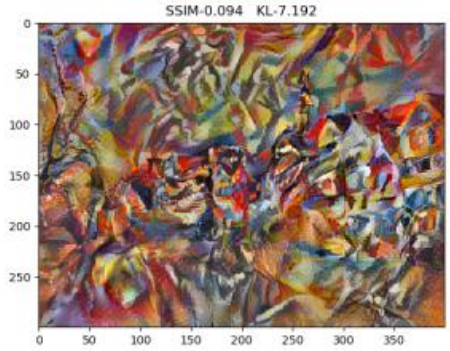
4.1.1 相同图片不同比值

content

style



α / β =0. 0001

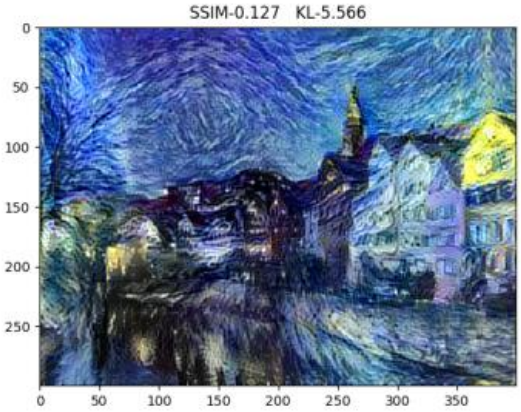


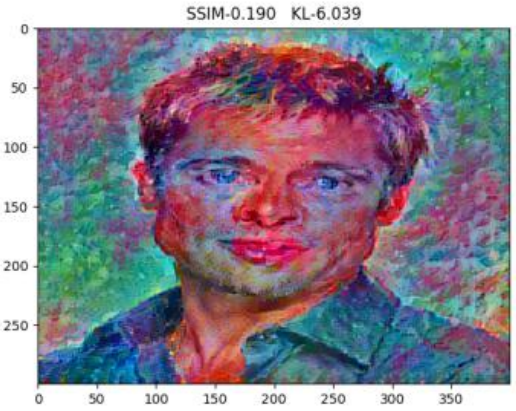
α / β =0. 001

α / β =0. 01

α / β =0. 1

4.1.2 相同比值不同图片





4.2 基于 AdaIN 层的实时任意风格迁移

4.2.1 相同图片不同比值

4.2.2 相同比值不同图片

五、总结分析

实验基于 google 的 colaboratory 平台上的 Tesla T4 GPU

基于卷积神经网络的图片风格迁移由随机噪声作为起始图，然后不断迭代改变图片的所有像素值来寻找一个目标结果图 x'，通过设置 α / β 值来调整风格和内容的比重。可以完成对任意风格的迁移，但是耗时比较长。

基于 AdaIN 层的实时任意风格迁移通过向 Encoder 输入两张图片，由 AdaIN 完成特征空间的转换，由训练好的 Decoder 转换层图片。在训练 Decoder 时，可以调整 λ 的值来调整风格和的比重。即使训练完成的比重是 1： λ，也可以通过调整 α 的值【T(c,s,α) = g((1 − α)f(c) + α AdaIN(f(c),f(s)))】，通过内容图的稀释，灵活地改变内容和风格的比重

5.1 速度

图像风格迁移的速度主要取决于图像的大小。为了公平地进行速度上的比较，我们将图像大小统一为 300*400。

基于卷积神经网络的图片风格迁移：我们的实验采用迭代 150 次，平均每轮时间 5s（在 Tesla K80 GPU 上）。平均花费 12.5min 获得一张风格图片

基于 AdaIN 层的实时任意风格迁移：单独转换一张图片，平均转换时间是 3-4s。而如果批量风格转换图片，则第一张图片需要 3-4s，在第一张图片转换成功后，由于模型已经加载完毕，后面的图片平均 0.05s 一张完成风格转换，达到实时转换。

```
num1used: 3.3834667205810547
num2used: 0.053102731704711914
num3used: 0.05160856246948242
num4used: 0.06259942054748535
num5used: 0.04625225067138672
num6used: 0.05284380912780762
num7used: 0.051950931549072266
num8used: 0.05796003341674805
num9used: 0.0444178581237793
num10used: 0.05307292938232422
num11used: 0.05278420448303223
num12used: 0.057814836502075195
```

5.2 多样性

两种方法都支持任意的内容图片、风格图片输入。

5.3 转换效果

5.3.1 EC 指标

由于普通用户没有受过专业知识的训练，判断完全靠主观意识。而用内容、风格损失值来定性转换效果从艺术角度略显勉强，邀请艺术家受限。我们查找文献发现 *Yeh* 等人提出[11]通过 EC 图来取代用户研究。

EC 图的 E 代表 Effectiveness，意思是风格迁移的有效性。通过计算生成的风格图片与提供的风格图片的 feature map 的匹配程度来代表有效性。

计算方法如下：

1)获得风格图和生成图的在 VGG19 模型里不同层（我们代码里取['block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1', 'block5_conv1']共 5 层）的 feature map。

2)然后每一层的 feature map 乘一个随机的单位向量，获得两组来自 5 个 feature map 的标量数据

3)累加这 5 个标量数据集，求两组均值，标准差（一组来自风格图片，一组来自生成图片）

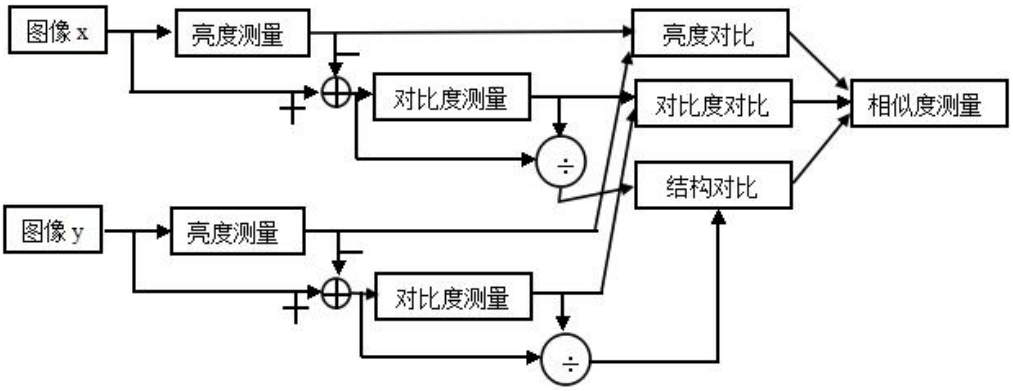
4)论文中假定数据分布是正太分布，因此利用 3)中的均值和标准差正态分布下的 KL 散度值来表示两个风格的相似度。

5)在下图的公式中，k 根据论文取 128，意味着步骤 2)、3)、4)重复 128 次。获得 128 组 KL 散度值。然后取均值。由于散度值越低，代表风格越相似，有效性越高（成反比），同时数值也比较小。在这个基础上取对数再取反。

$$d(\mathbf{v}_k) = -log\sigma_1 + \frac{\sigma_1^2 + \mu_1^2}{2} - \frac{1}{2}$$

$$E = -\log\left(\frac{1}{R}\sum_k d(\mathbf{v}_k)\right)$$

EC 图的 C 代表 Coherence，即内容上，生成图片和输入的内容图片需要保证相干性。论文上提供了两种衡量方法，一种是反映边界的保护程度，但是论文中提出的方法效果并不好。另外一种是反映物体的保护程度，主要利用的是图像质量评估算法 SSIM，一种主要用来衡量图片相似度的方法。



SSIM 测量系统

图 5.1 SSIM 测量系统

由 SSIM 测量系统可得相似度的测量可由三种对比模块组成，分别为：亮度，对比度，结构。接下来我们将会对这三模块函数进行定义。

首先，对于离散信号，我们以平均灰度来作为亮度测量的估计：

$$\mu_x = \frac{1}{N}\sum_{1}^{N} x_i$$

亮度对比函数 l（x,y）是关于μ_x, μ_y的函数。

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

然后，由测量系统知道要把平均灰度值从信号中去除，对于离散信号$x - \mu_x$，可使用标准差来做对比度估量值。

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}$$

对比度对比函数 c(x,y)就是σ_x, σ_y的函数。

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

接下来，信号被自己的标准差相除，结构对比函数就被定义成$\frac{(x-\mu_x)}{\sigma_x}$和$\frac{(y-\mu_y)}{\sigma_y}$的函数。

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

最后，三个对比模块组合成一个完整的相似测量函数：

$$S(x,y)=f\left(l(x,y),c(x,y),s(x,y)\right)$$

求出 KL 值、SSIM，分别代表着 E，C，即可从内容、风格上衡量一张图片的效果。

5.3.2 评估结果

为了公平得进行对比评估，内容和风格的比值统一为 1:2。在同样的比重下，对比有效性和相干性。

从结果上看，在内容与风格比例为 1/2 时，第二种方法的 SSIM、KL 均比第一种的大。即基于 AdaIN 的风格迁移无论从内容上还是从风格有效性上看，都优于第

一种方法。因此总结来说，第二种方法明显优于第一种方法。因此总结来说，两种方法都支持任意的风格迁移，但是从速度、转换效果来说，第二种方法明显优于第一种方法。

六、参考文献

[1] L.A. Gatys, A.S.Ecker, and M.Bethge,“Image style transfer using convolutional neural networks” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2414－2423.

[2] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution” in European Conference on Computer Vision, 2016, pp. 694－711.

[3] X. Huang and S. Belongie,“Arbitrary style transfer in real-time with adaptive instance normalization” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1501－1510.

[4] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In JMLR, 2015.

[5] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In CVPR, 2017

[6] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In ICLR, 2017.

[7] C. Li and M. Wand, “Combining markov random fields and convolutional neural networks for image synthesis,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2479－2486

[8] V. Dumoulin, J. Shlens, and M. Kudlur, “A learned representation for artistic style,” in International Conference on Learning Representations, 2017.

[9] T. Q. Chen and M. Schmidt,“Fast patch-based style transfer of arbitrary style,” in Proceedings of the NIPS Workshop on Constructive Machine Learning, 2016.

[10] Sanakoyeu, Artsiom , et al. ″A Style-Aware Content Loss for Real-time HD Style Transfer.″ (2018).

[11]Yeh, Mao Chuang , et al. ″Quantitative Evaluation of Style Transfer.″ (2018).