华南师范大学

《人工智能导论》课程项目

课 程 项 目 报 告

项　目　题　目：图像风格迁移

所　在　学　院：计算机学院

项　目　组　长：陈建宇

小　组　成　员：何东健、叶嘉和

开　题　时　间：2022 年 3 月 15 日

一、引言

所谓风格迁移，其实就是提供一幅画(Reference style image)，将任意一张照片转化成这个风格，并尽量保留原照的内容(Content)。

自 2015 年 Gatys 首次提出神经艺术风格迁移框架以来，图像风格迁移逐渐成为计算机图形学和计算机视觉领域的一个研究热点，网络上对于该方面的知识介绍及学习资源多，便于我们掌握基础知识原理及深入学习图像风格迁移的实现代码、原理和环境搭载等知识。

图像风格迁移是指利用计算机将一个图像以不同的图像风格呈现出来的方法，其利用相关算法学习画作的风格，然后再把这种风格应用到另外一张图片上，以此实现图像的风格迁移。在图像风格迁移上，已经有较多的研究，目前深度学习的图像风格迁移方法主要包括基于图像迭代和基于模型迭代两种。

本项目主要参照 Gatys 等人于 2015 年先后发表的两篇论文[1] [2]为基础，使用 Pytorch 库复现其论文提出的基于卷积神经网络的图像风格迁移。

二、国内外研究现状

1 基于图像迭代的风格迁移

基于图像迭代的风格迁移算法主要有基于最大均值差异、基于马尔科夫随机场、基于深度图像类比三类。具有合成图像质量高、可控性好、易于调参、无需训练网络、计算时间长、对预训练模型依赖性大等特点。

1.1 基于最大均值差异的风格迁移［1］［2］

Gatys 于 2015 年第一次将 VGG19 网络应用到风格迁移中。Gatys 的关键发现在于：卷积神 经网络的内容和风格是分离的，而通过构造 Gram 矩阵可以提取出任意图像的风格特征表示。随后 Li 等人证明 Gram 矩阵的匹配方式等价于最小化特定的最大均值差异。

1.2 基于马尔科夫随机场的风格迁移（Li 2016）[3]

dCNN 的池化层压缩图像信息时，损失了大量的信息。Gatys 等人使用了 VGG 网络中的滤波金字 塔作为图像的高层表征。但是不同卷积层只能捕获图像每个像素之间的联系，而缺少空间分布上的 联系。基于以上问题，Li 等人提出了结合马尔科夫随机场和卷积神经网络的模型，将 Gatys 模 型中的 Gram 矩阵匹配替换为马尔科夫正则化模型。马尔可夫模型描述了具有同类特征信息的集合， 因此将 CNN 图像特征映射分割成许多区域块并进行匹配，可以提高合成图像在视觉上的合理性。

该方法为基于图像迭代的风格迁移提供了一个有趣的拓展，但同时也存在很多局限性：

1. 对于输入图像有很强的限制：风格图像必须能够被马尔可夫重构。例如，具有较强透射结构的图片不适用于本方法。
2. 结果图相较于原图会产生边缘模糊的缺点。这是因为网络训练过程中图像的损失。

综上，基于马尔科夫随机场的风格迁移方法只有在内容图像与风格图像具有相似的形状，同时没有强烈的角度、尺寸变化时才会产生良好的效果。

1.3 基于深度图像类比的风格迁移（Liao Jing 2017）[4]

Liao 等人通过结合深度学习（VGG19）与图像类比（PatchMatch）进行风格迁移，取得了 良好的效果。 PatchMatch 是图像类比的经典算法：给定一组图片 A 和 B，A 提供语义和内容两大主要信息， B 提供外观性（例如颜色、光照、风格等）和细节（例如纹理）信息，将两张图片都划分为固定 大小的像素块，对于 A 中的每一个像素块，在 B 中寻找与之最相近的像素块来代替。当像素块足 够小，且全部替代完成后进行平滑等处理后，即刻获得用 B 的风格表现 A 的风格迁移结果图。

2 基于模型迭代的风格迁移算法

虽然基于图像迭代的方法可以产生效果出色的风格合成图像，但是存在效率低下的问题。而基于模型迭代的图像风格迁移方法通过大量数据训练生成模型，基于模型迭代的风格迁移算法主要有基于生成模型的风格迁移和基于图像重构解码器的风格迁

移两种。

2.1 基于生成模型的风格迁移（Johnson 2016）[5]

Johnson 等人最早提出了迭代优化生成模型的图像风格迁移方法，即快速风格迁移。与之 前训练生成模型时使用逐个像素比较的损失函数相比，感知损失函数对预训练 VGG 模型提取的高 层抽象特征表示进行平方求差，这部分的问题求解与 Gatys 等人算法是一致的。快速风格迁移算 法开创了新的风格迁移思路。此外，Ulyanov 等人也采用了类似的网络架构(Ulyanov 2016)，并通过实验表明 了在生成模型训练过程中，使用实例归一化替代批量归一化可以显著提高生成图像的质量。Zhang 等人构建了一个可以训练多种风格的生成模型，实现了多风格的快速风格迁移。

2.2 基于图像重构解码器的风格迁移［6］［7］

基于图像迭代存在着参数调整和效率低下两个弊端，而快速风格迁移虽然缓解了效率低下的问 题，但只能针对特定风格进行模型训练，并且仍然无法避免参数调整的问题。为了克服这些问题，Li 等人提出了一种基于图像重构解码器的风格迁移算法(Li 2017)，使得网络可以不经过重复训练而对 任意风格进行迁移。该算法的主要创新点如下：

1.通过特征迁移（例如：whitening and coloring），直接将内容特征数据匹配到风格图像的
深层特征空间。

2.将特征变换与预先训练的一般编解码器网络相结合，使迁移过程可以通过简单的前馈操作来
实现。

三、模型和算法

1. 基本原理

原理很简单：定义两个差异值，分别用于图像的内容和风格，用来衡量两幅图像在内容和风格上的差异，生成的图像相对风格图像的风格损失和相对内容图像的内容

损失都尽可能的小。

2.导入必要库与模块，选择运行设备

图像风格转换所需包的汇总。

- torch, torch.nn, numpy：使用 PyTorch 进行风格转换必不可少的包
- torch.optim：高效的梯度下降
- PIL, PIL.Image, matplotlib.pyplot：加载和展示图片
- torchvision.transforms：将 PIL 图片转换成张量
- torchvision.models：训练或加载预训练模型
- copy：对模型进行深度拷贝；系统包

```
# Pytorch 神经网络
import torch
import torch.nn as nn
import torch.nn.functional as F
# 梯度下降
import torch.optim as optim

# 加载、显示图像
from PIL import Image
import matplotlib.pyplot as plt

# 将 PIL 图像转换为 tensor 张量
import torchvision.transforms as transforms
# 训练模型或加载预训练的模型
import torchvision.models as models

# 深度拷贝模型
import copy
```

本项目选择在 Nvidia RTX 3050 上运行，借助 Nvidai 推出的并行计算架构 CUDA

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

3.损失函数

- 内容损失

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

• 风格损失

$$\mathcal{L}_{\text{style}}(\vec{a},\vec{x})=\sum_{l=0}^{L}w_{l}E_{l}, \qquad (5)$$

• 总损失

$$\mathcal{L}_{\text{total}}(\vec{p},\vec{a},\vec{x})=\alpha\mathcal{L}_{\text{content}}(\vec{p},\vec{x})+\beta\mathcal{L}_{\text{style}}(\vec{a},\vec{x}) \qquad (7)$$

公式（1）中，F 、P 分别是生成图像的卷积特征和源图像的卷积特征。

公式（5）中，F 是生成图像的卷积特征，G 是 F 的 Gram 矩阵，A 是源图像卷积特征的 Gram 矩阵，E_l 表示第 l 层的风格误差。在论文中，总风格误差是某几层风格误差的加权和，其中权重为 W_l 。事实上，不仅总风格误差可以用多层风格误差的加权和表示，总内容误差也可以用多层内容误差的加权和表示。只是在原论文中，只使用了一层的内容误差。

公式（7）中，α， β 分别是内容误差的权重和风格误差的权重。实际上，我们只用考虑 α， β 的比值即可。如果 α 较大，则说明优化内容的权重更大，生成出来的图像更靠近内容图像。反之亦然。

4. 导入模型

使用 VGG-19 模型

PyTorch 对 VGG 的实现模块包括两个子模块：特征（features，包含卷积层和池化层）和分类器（classifier，包含全连接层）。我们将使用 features 模块，因为我们需要各个卷积层的输出来计算内容和风格损失。有些层次在训练过程中与评估过程中的行为不同，因此必须使用评估模式。

```
cnn = models.vgg19(pretrained=True).features.to(device).eval()
```

5.梯度下降

损失函数用来衡量机器学习模型的精确度。一般来说，损失函数的值越小，模型的精确度就越高。如果要提高机器学习模型的精确度，就需要尽可能降低损失函数的值。而降低损失函数的值，我们一般采用梯度下降这个方法。所以，梯度下降的目的，就是为了最小化损失函数。

按照本算法作者 Leon Gatys 的建议，我们使用 L-BFGS 算法来实现梯度下降。与训练网络不同，我们训练输入图像，以尽量减少内容和风格损失。我们将使用一个 Pytorch L-BFGS 优化器，并将输入图像转换为要优化的张量作为参数传入。

对于不同的风格图像和内容图像，达到最小损失所需迭代次数都不同

6.算法执行

创建一个执行整个迁移算法的函数。对于网络的每次迭代，它会获取更新后的输入并计算新的损失值。我们将运行每个损失模块的方法来动态计算它们的梯度。优化器需要一个“闭包”函数，该函数重新评估模块并返回损失。

最后，网络有可能使用超过图像的张量范围（0-1）的值来优化输入。我们可以通过在每次运行网络时将输入值修正为 0 到 1 来解决。

算法执行过程中，梯度下降算法每迭代 50 次，我们就输出当前的风格损失和内容损失值以供观察记录。

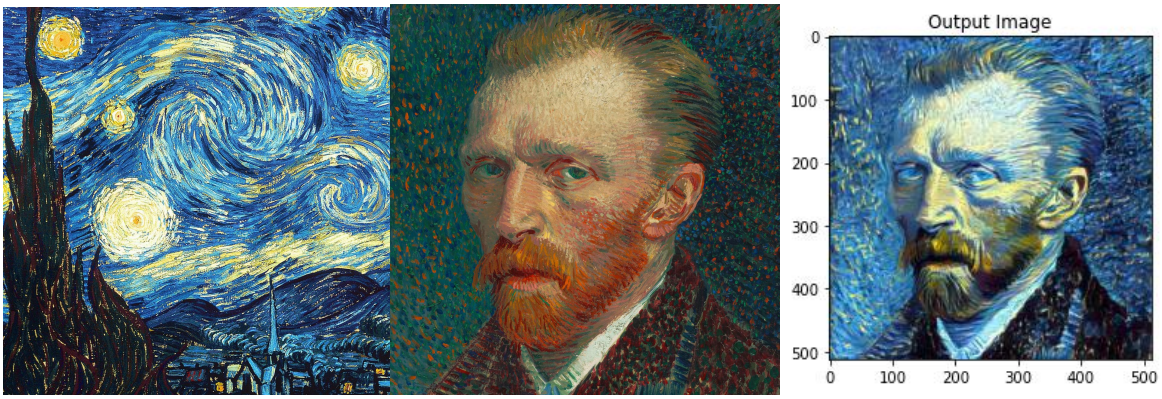
示例：

```
In [6]: runfile('D:/data-work/Codes/vscode-python/Tutorial-
Neural_Transfer.py', wdir='D:/data-work/Codes/vscode-python')
正在建立风格迁移模型..
正在优化中..
run [50]:
风格损失: 211.648300 内容损失: 8.925686

run [100]:
风格损失: 82.145294 内容损失: 9.396108

run [150]:
风格损失: 36.617504 内容损失: 9.899523

run [200]:
风格损失: 20.507767 内容损失: 9.642317

run [250]:
风格损失: 14.293995 内容损失: 9.150565

run [300]:
风格损失: 10.263545 内容损失: 8.746164
```

四、实验结果分析

效果展示：


Output Image
0
100
200
300
400
500
0
100
200
300
400
500


Output Image
0
100
200
300
400
500
0
100
200
300
400
500


Output Image
0
100
200
300
400
500
0
100
200
300
400
500


Output Image
0
100
200
300
400
500
0
100
200
300
400
500

可以看出大多数风格图像和内容图像的组合迁移后的视觉效果都不错。

实际测试过程中，尽管有些组合最终的风格损失和内容损失都非常小，但视觉效果却不尽人意。尤其是当风格图像色彩种类比较单一或色调比较相近时，输出图像往往惨不忍睹。以及当风格图像的内容在我们人类看来寥寥无几（如现代抽象画）时，最终效果也往往一言难尽。

五、结论

1 关于梯度下降迭代次数

对于不同的风格图像和内容图像，在梯度下降环节时，达到最佳迁移效果所需 L-BFGS 算法迭代的次数往往是不同的。

逐渐增大迭代次数，部分结果中风格损失和内容损失值，往往在逐渐降低到一个相对最低点后，逐渐增大甚至陡然增大。

2 关于图像选取

对于不同类型的风格图像与内容图像的迁移效果不同，如对于内容图像是人或动物头像时，使用风景类型的图像往往得不到很好的视觉效果。

3 启发

神经网络不仅可以用于在大批数据集上训练，完成一项通用的任务，还可以经过预训练，当作一个特征提取器，为其他任务提供额外的信息。同样，要记住神经网络只是优化任务的一项特例，我们完全可以把梯度下降法用于普通的优化任务中。在这种利用了神经网络的参数，而不去更新神经网络参数的优化任务中，梯度下降法也是适用的。

4 思考

其实本项目参照的论文在现在看来，是比较早期的使用神经网络进行风格迁移的作品。在近几年里，肯定已经有许多试图改进此方法的研究出现。时至今日，再去深究这篇文章里的一些细节（为什么用 Gram 矩阵，应该用 VGG 的哪些层做拟合）已经意义不大了。我认为值得关注的是该论文提供的主要思想。

该论文给我最大启发是：神经网络不仅可以用于在大批数据集上的训练，完成一项通用的任务，还可以经过预训练，作为一个特征提取器，为其他任务提供额外的信息。同样，神经网络只是优化任务的一种特例，我们完全可以把梯度下降法用于普通的优化任务中。在这种使用了神经网络的参数，但没有更新神经网络参数的优化任务中，梯度下降法也同样适用的。

此外，这篇文章中提到的“风格”也是一项很有趣的属性。该论文算是首次利用了神经网络中的信息，用于提取内容、风格等图像属性。这种提取属性（尤其是提取风格）的想法被运用到了很多的后续研究中，比如大名鼎鼎的 StyleGAN。

长期以来，人们总是把神经网络当成黑盒。但是，这篇文章给了我们一个掀开黑盒的思路：通过拟合神经网络中卷积核的特征，我们能够窥见神经网络每一层保留了哪些信息。

六、参考文献

[1] Gatys L A, Ecker A S, Bethge M. (Gatys 2016). "Image style transfer using convolutional neural networks" [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2414-2423.

[2] Gatys L, Ecker A S, Bethge M. (Gatys 2015) "Texture synthesis using convolutional neural networks" [J]. Advances in neural information processing systems, 2015, 28.

[3] Li Chuan, Wand M. (Li 2016) "Combining Markov random fields and convolutional neural networks for image synthesis" [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition, Piscataway, NJ: IEEE Press,2016: 2479-2486

[4] Liao Jing, Yao Yuan, Yuan Lu, et al. (Liao 2017) "Visual attribute transfer through deep image analogy" [J]. arXiv preprint arXiv: 1705. 01088, 2017

[5] Johnson J, Alahi A, Li Feifei. (Johnson 2016)"Perceptual losses for real-time style transfer andsuper-resolution" [C]// Proc of European Conference on Computer Vision. [S. I.] : Springer Press, 2016: 694-711

[6] Ulyanov D, Lebedev V, Vedaldi A, et al. (Ulyanov 2016) "Texture networks: feed-forward synthesis of textures and stylized images" [EB /OL]. arXiv preprint arXiv: 1705.08086, 2016

[7] Li Yijun, Fang Chen, Yang Jimei, et al. (Li 2017) "Universal style transfer via feature transforms" [EB/OL]. arXiv preprint arXiv: 1603.03417v1, 2017

[8] 代码实现：https://pytorch.org/tutorials/advanced/neural_style_tutorial.html