第3章

CHAPTER 3

图像修复

人们总是怕美好的时光一去不复返，从而千方百计地想要留住它，于是就有了照片，照片留住了我们生命中某个难忘的时刻。现如今，拍摄一张照片是再简单不过的事情，无论是山川风景、自然风光还是结婚照片或者是平日里的随手自拍，都可以被完好地保存下来。然而，对于20世纪70—80年代的人们来说，想要留下一张照片是极为不容易的，要完好地保存到现在更是不太可能的事情。

我们可能经常会在家里找到父母或者爷爷奶奶那一辈的老照片，就像下面这张图片，记录了他们年轻时候的样子，保留了他们的青春。然而这种照片大多经过岁月的洗礼，都已不太清楚，难以辨认，给我们珍贵的回忆带来了遗憾。这时候，图像修复技术就提供了莫大的帮助。最近，如图3.1所示的图像修复大火，从网友们用小程序“你我当年”一键修复老照片到热门项目“用机器学习修复老照片”，都是保存记忆的好方法。

图3.1　图片修复效果

（图片来源：https://max.book118.com/html/2016/0513/42875437.shtm）

3.1　背景介绍

什么是图像修复呢？图像修复就是对图像中缺失的区域进行修复，或是将图像中的对象抠去并进行背景填充，以取得难以用肉眼分辨的效果。用通俗的语言来讲就是既可以用这种方法来还原缺失图像，如图3.2(a)所示，也可以用此方法将图像中不想要的对象除去，

如图 3.2(b)所示，并且让人看不出毛病，以为图片本应如此。

(a) 还原缺失图像　　(b) 去除图像多余部分

图 3.2　图像修复效果

图像修复(image inpainting)的历史可以追溯到文艺复兴时期，起初是指艺术工作者对于博物馆等地方所储存的年代久远、已经出现破损或缺失的艺术作品进行手工修补的一种方法。要使图像修复的结果达到预计效果是非常困难的，它要求修复后的图片和原图难以区分，看起来毫不违和。这个问题就像是一幅画的一部分被遮住了，可以利用想象力来想象或者以逻辑来推断被遮挡的区域是什么样子。这些对人来说似乎很简单，然而要让机器做到却非常困难。

随着计算机技术的飞速发展，图像修复的方法也有所改进。Bertalmio 等在博物馆通过长时间的仔细观察，于 2000 年 7 月第一次提出了数字修补(digital inpainting)这个术语，并建立了三阶偏微分方程(Partial Differential Equation，PDE)来解决这一问题。这是一个突破性的进展，它使本来由艺术工作者手工完成的工作得以用计算机来完成。这项技术在节省时间的同时也提高了图像的可重复修改性[1]。

近年来，深度学习(Deep Learning，DL)方法在图像修复方面取得了巨大的成就。它可以通过自己学习到的数据分布来填充缺失区域的像素，还可以在缺失区域生成与未缺失区域连贯的图像结构框架，这对传统的修复方法来说几乎是不可能做到的。最早使用深度学习来进行图像修复的方法之一是上下文编码器[2]，它使用了编码器-解码器的结构。编码器将缺失区域的图像进行映射到低维度的特征空间，解码器在它的基础上构造输出图像。然而，这种方式的缺点是它的输出图像通常出现了视觉伪像，并且相对模糊。因此，Lizuka 等通过减少下采样层的数量，并用一系列的空洞卷积层替换全连接层，使用变化的膨胀因子来补偿下采样层的减少[3]。但是由于采用了大的膨胀因子产生了极为稀疏的滤波器，则必然大大增加了时间成本。对上下文编码器进行改善的另一个方法是使用预训练的 VGG 网络，通过最小化图像背景的特征差异来改善上下文编码器的输出[4]。但同样，这个方法需要迭代地求解多尺度的优化问题，在时间上也增加了计算成本。也可以引入部分卷积，其中卷积的权重由卷积滤波器当前所在窗口的掩膜区域归一化得到[5]，此方法有效解决了卷积滤波器在遍历不完整区域时捕获过多零的问题。而文献[6]中提出的方法与之前有了明显的不同与进步，它采用两个步骤来解决问题，首先，对缺失的区域进行粗略估计，接着细化网络，通过搜索与粗略估计得出具有高相似性的背景图片的集合，使用注意力机制来锐化结构。同样地，可以通过引入一个“补丁交换”层，用缺失区域内的每个补丁来替换边界上与之相似的补丁[7-8]。也可以使用手绘草图的方法来指导图像修复工作[9]，这种方法使修复的效果得到了更好的保证。然而，手绘的方法显然不是那么智能，还是需要借助于人工力量的帮

助。因此，本实验在此基础上取消了手绘草图的步骤，使其学会在缺失的区域产生合理的边缘幻觉，最终达到合理的修复效果。

图像修复技术除了修复久远的照片之外还有着非常广泛的应用。比如在日常生活中，随手的自拍照就可以用图像修复技术来去水印，消除红眼或者不喜欢的痘痘、疤痕等。与之相似地，在电视电影行业中，可以用这个技术对电影中不清晰的画面进行补全，或者修复已经久远的、保存不完整的胶片等。在文物领域，由于各种原因而损坏的历史文物，人为修复可能会存在修复失误从而造成二次损坏的情况，这时候图像修复技术就可以很好地解决这个问题。在医学领域，可利用图像修复技术去除医学图像中的噪声，增加图像的对比度和清晰度，方便观察和处理。在数字图像的编码和传输过程中也可以使用图像修复技术来替换丢失的数据。在很多方面，图像修复技术都有着不可替代的作用。本次实验只针对照片的修复进行，旨在令读者清楚图像修复的原理和思路。

3.2 算法原理

本次实验使用了两次 GAN，实验流程如图 3.3 所示，首先将缺失图像输入网络，经过边缘生成网络（第一个 GAN）生成一个完整的边缘图像，并以此为前提将图像再输入图像补全网络（第二个 GAN），最后生成完整的图像。

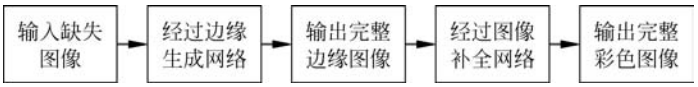


图 3.3 实验流程图

这里首先介绍一些后面网络模型会用到的概念，以帮助后续过程的理解。

3.2.1 基础知识介绍

1. 空洞卷积

如图 3.4 所示，空洞卷积（dilated convolution）又可以翻译为膨胀卷积或扩张卷积，起源于语义分割，就是在标准的卷积层中注入空洞，从而来增加计算的区域。这个卷积的好处就是在不进行导致信息损失的池化层操作的情况下，让每个卷积都可以输出尽可能大范围的信息。

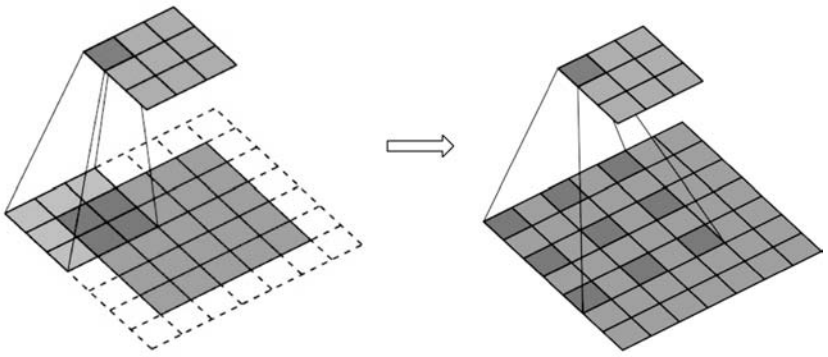


图 3.4 空洞卷积示意图

2. 掩膜

掩膜是由 0 和 1 组成的一个二进制图像。当在某一功能中应用掩膜时，1 值区域被处理，被屏蔽的 0 值区域不被包括在计算中。用这个设定好的二进制图像对要处理的图像进行遮挡，进而控制要处理的图像区域。将原图中的像素和掩膜中的像素对应进行运算，1 & 1 = 1，1 & 0 = 0。比如一个 3×3 的图像与 3×3 的掩膜进行运算，得到的结果图像如图 3.5 所示。

23	22	89		0	0	1		0	0	89
0	0	255	&	1	0	1	⇨	0	0	255
90	0	23		1	1	1		90	0	23

图 3.5　掩膜原理示意图

3. Canny 边缘检测器

Canny 边缘检测是一种非常流行的边缘检测算法，由 John Canny 在 1986 年提出的。它是一个多阶段的算法，该算法可分为以下步骤：

(1) 图像灰度化，只有灰度图才能进行边缘检测。

(2) 使用高斯滤波器，以平滑图像，滤除噪声。

(3) 计算图像中每个像素点的梯度强度和方向。

(4) 应用非极大值抑制(non-maximum suppression)消除边缘检测带来的杂散响应。

(5) 应用双阈值(double threshold)检测来确定真实的和潜在的边缘。

(6) 通过抑制孤立的弱边缘最终完成边缘检测。

本实验使用了一个两阶段的边缘连接模型，第一个步骤生成了完整的边缘图像，使用了边缘生成网络。第二个步骤将图片修复完整，使用图像补全网络。

3.2.2　边缘生成网络

边缘生成网络中使用了一次 GAN，如图 3.6 所示，将缺失的彩色图像变为灰度图像，并且提取出它的边缘图像，再加上该图像对应的掩膜图像，将这三种图像输入网络，通过训练好的 G_1 网络使其输出相应的完整边缘图像。其中，G_1 网络包含两个进行编码的下采样层、

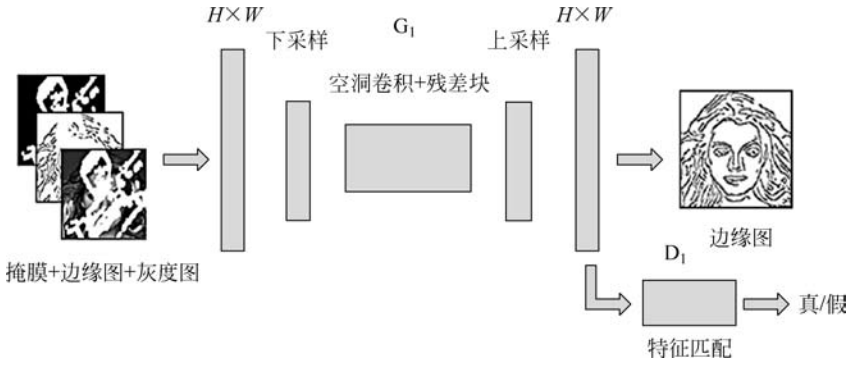
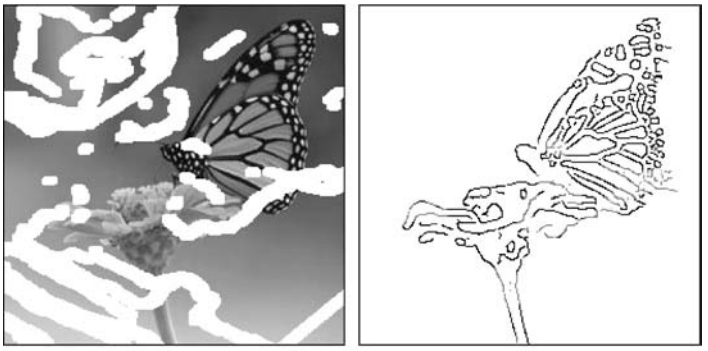


图 3.6　边缘生成网络

8个残差模块和一个解码的上采样层，用两个带有伸缩因子的空洞卷积层在残差块中代替常规的卷积，使其在最终的残差块中生成区域。

在第一个阶段中，输入网络的缺失图像中，白色部分为缺失区域。使用边缘生成模型对缺失的区域产生“幻觉”生成一个完整的边缘图像，这个边缘图像中用黑色轮廓线表示从输入图像的现有部分提取出的边缘图像，用蓝色的轮廓线表示通过生成模型所补全的缺失区域的边缘图像。

图 3.7(a)为输入的缺失图像，缺失的区域用白色来表示。然后计算出图像的边缘掩膜，图 3.7(b)中的深色边缘线是用 Canny 边缘检测器对已知区域边缘的计算，而浅色边缘线是边缘生成网络对缺失区域的补全效果。



(a) 缺失图像　　(b) 补全效果

图 3.7　边缘补全效果

3.2.3　图像补全网络

然后进入第二个阶段，在图像补全网络中又一次使用了 GAN。如图 3.8 所示，将得到的这个完整的边缘图像和要补全的图像输入图像补全网络，通过训练好的 G_2 网络处理后得到完整的补全后的图像。该 G_2 网络中的具体构造与图像生成网络相同，包含两个进行编码的下采样层、8 个残差模块和一个解码的上采样层。对于 D 网络，使用 70×70 的 patch GAN 架构[10]来判断重叠部分是否正确，并用实例正则化来遍历网络的所有的层。其中，示例正则化是指对一个批次中的单个图片进行归一化，而不是像批量归一化(batch normalization)一样对整个批次的所有图片进行归一化。如果读者想进一步了解其具体原

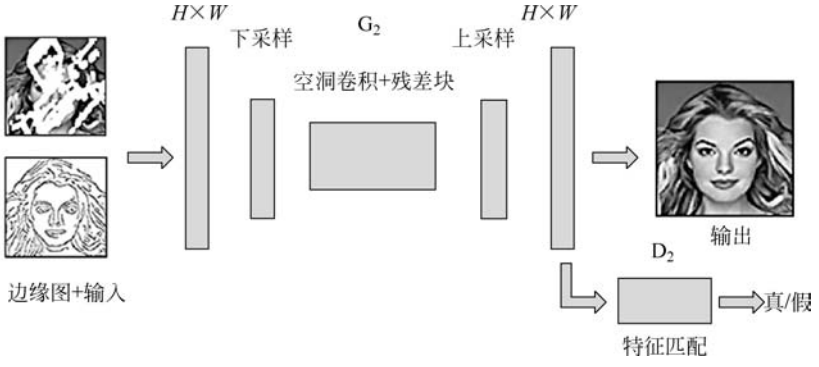


图 3.8　图像补全网络

理，可参考 Johnson 等的论文[11]来了解其更具体的网络构成。

图 3.9(a)为缺失的彩色图像，即要补全的图像，缺失区域用白色部分表示。图 3.9(b)为经过第一个阶段修复好的完整边缘图像。以这两个图像为依据，输入图像补全网络，从而就可以得到完整的修复后的效果图，如图 3.9(c)所示。

(a) 缺失图像　(b) 边缘图像　(c) 完整修复

图 3.9　图像补全效果

3.2.4　网络结构介绍

整个网络分为两个 GAN 的组合。GAN 主要包括了两个部分，即生成模型 G(Generator)与判别模型 D(Discriminator)。G 模型是一个图片生成网络，它的输入是一系列无规律的随机样本，输出是由这些样本所生成的图片，而 D 模型是一个判别网络，它的输入是 G 网络输入的图片，输出是一个表示概率的数字。如果该数字为 1，则代表是真实的图片；如果是 0，那么它一定不是真实的图片。G 模型主要通过学习真实输入的图像来让自己生成的图像更加的“真实”，从而“骗过”D 模型。而 D 模型则主要对接收到的图片进行真伪的判断。这个生成器生成更加真实的图像和判别器努力识别图像真伪的过程相当于一个二人博弈的过程，随着不断地互相完善，G 模型和 D 模型最终会达到一个动态平衡的状态，即 G 模型可生成接近真实的图像，而 D 模型可判断它为真，对于给定图像的预测为真的概率基本接近于 0.5 即可。

在使用网络对图像修复之前，应该先对整个模型算法进行训练，使其拥有能够补全输入图形的能力。接着可以用一张图片进行测试，看看它到底能不能达到所要求的补全效果。本实验的训练中，需要训练两个网络，即边缘生成网络和图像补全网络。经过两次 GAN 的训练，将得到实验所需要的完整网络模型。训练好网络模型后，就可以开始对它的效果进行测试。如图 3.6 和图 3.8 所示的那样，首先将缺失边缘图像、缺失灰度图像、缺失图像掩膜输入 GAN 的生成模型中，经过训练好的 G1 网络就可以生成所需要的完整的边缘图像，再将这个图像和一开始要补全的那个彩色图像输入第二个 GAN，经过训练好的 G2 网络，最终得到所需要的补全后的图像。

3.3　实验操作

3.3.1　代码介绍

1. 实验环境

图像修复实验环境如表 3.1 所示。

表 3.1 图像修复实验环境

条　　件	环　　境
操作系统	Ubuntu 16.04
开发语言	Python 3.6
深度学习框架	Pytorch 1.0
相关库	Numpy=1.14.3 Scipy=1.0.1 Future=0.16.0 Matplotlib=2.2.2 Pillow=5.0.0 opencv-python=3.4.0 scikit-image=0.14.0 pyaml

2. 实验代码下载地址

扫描二维码下载实验代码。

3. 代码文件目录结构

```
├── checkpoints ·····用来存放训练好的模型
│   ├── celeba
│   ├── places2
│   └── results ·····用来存放补全的结果图片
├── config.yml.example
├── examples ·····用于测试的图片(可换成自己的)
│   ├── celeba
│   │   ├── images ·····缺失图片
│   │   └── masks ·····图片掩膜
│   └── places2
│       ├── images ·····缺失图片
│       └── masks ·····图片掩膜
├── main.py
├── README.md ·····运行代码前阅读
├── requirements.txt ·····需要的库函数
├── scripts
│   ├── fid_score.py ·····测量 Fréchet 的初始距离(FID 得分)
│   ├── flist.py ·····生成训练、测试和验证集的文件列表
│   └── metrics.py ·····评估模型
├── src
│   ├── config.py ·····模型配置
│   ├── loss.py ·····计算损失
│   ├── metrics.py ·····计算精确边缘图
│   ├── models.py ·····模型搭建
│   ├── networks.py ·····搭建网络
│   └── utils.py ·····功能函数,生成掩膜、显示图片等函数
├── test.py ·····测试程序
└── train.py ·····训练程序
```

3.3.2 数据集介绍

1. Places2 数据集

Places2 数据集如图 3.10 所示,是一个场景图像数据集,包含 1000 万张图片,400 多个

不同类型的场景环境，如餐厅、森林、码头、街道、游乐场等。该数据集可用于以场景和环境为应用内容的视觉认知任务，由 MIT 维护。

图 3.10　Places2 数据集部分图像展示

Places365-Standard 是 Places2 数据库的核心。Places365 的类别列表位于 Categories_places365.txt。Places365-Standard 的图像数据有 3 种类型：Places365-Standard 的训练数据（TRAINING）、验证数据（VALIDATION）和测试数据（TEST）。3 种数据源没有重叠：培训、验证和测试。所有这 3 组数据均包含 365 种场景的图像。高分辨图像档案中图像已调整大小，最小尺寸为 512，同时保留图像的长宽比。小尺寸图像档案中，如果图像尺寸小于 512，则原始图像保持不变。在简洁目录下的小尺寸图像文档中，不管图像原始宽高比如何，数据集中的图像均已调整为 256×256。这些图像是 256×256 图像，采用的是更友好的目录结构。

数据集下载地址：http://places2.csail.mit.edu/index.html。

2. CelebA 数据集

CelebFaces 属性数据集（CelebA）如图 3.11 所示，是一个大型数据集，包含二十多万张名人图像，每个图像有 40 页的属性注释。此数据集中的图像涵盖了较大的姿势变化和背景杂波。CelebA 具有多种多样，数量众多且注释丰富的特点，其身份数量为 10 177，人脸图像为 202 599 张，还包括 5 个地标位置，每个图像有 40 个二进制属性注释。该数据集可用作为以下计算机视觉任务的训练和测试集：面部属性识别、面部检测、界标（或面部部分）定位以及面部编辑和合成。CelebFaces 属性数据集是香港中文大学的开放数据。

该数据集包含 3 个文件夹，Img 文件夹下是所有的图片，图片又分为 3 类。其中 img_celeba.7z 文件夹是没有做裁剪的图片，img_align_celeba_png.7z 和 img_align_celeba.zip 是把 img_celeba.7z 文件裁剪出人脸部分之后的图片，其中 img_align_celeba_png.7z 是 png 格式的，img_align_celeba.zip 是 jpg 格式的。

数据集下载地址：http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html。

图 3.11　CelebA 数据集部分图像展示

3.3.3　实验操作及结果

1. 预处理图像

本实验使用了 Places2 和 CelebA 这两个数据集，在它们的基础上训练模型，读者可根据数据集介绍部分了解和下载。

下载后运行 scripts 中的 flist.py 来生成训练和测试的文件列表。例如，要在 Places2 这个数据集上来生成训练集文件列表，可运行如下命令（在实际操作时应 path path_to_places2_train_set 改为数据集所在目录）：

```
$ mkdir datasets
$ python ./scripts/flist.py -- path path_to_places2_train_set\
  -- output./datasets/places_train.flist
```

此步骤将新建一个名为 datasets 的文件夹，并在这个文件夹下生成名为 places_train.flist 的文件列表。

2. 掩膜处理

可从 http://masc.cs.gmu.edu/wiki/partialconv 处下载由 Liu 等提供的公开的不规则掩膜数据集，并且使用 scripts 中的 flist.py 来生成训练和测试的掩膜文件列表。

3. 训练网络模型

开始训练模型之前，先下载一个类似 example config file（示例配置文件）的 config.yaml 文件，并将其复制到程序中 checkpoints 的文件夹下。

训练模型可以使用如下指令：

```
$ python train.py -- model [stage] -- checkpoints [path to checkpoints]
```

例如，要在./checkpoints/places2 目录下的 places2 数据集上训练边缘模型：

```
$ python train.py -- model 1 -- checkpoints ./checkpoints/places2
```

4. 测试网络模型

同样，在开始测试模型之前，先下载一个类似 example config file（示例配置文件）的 config.yaml 文件，并将其复制到程序中 checkpoints 的文件夹下（若在训练步骤已做过相关步骤，此处可以省略）。

在测试前应将之前训练好的模型或者可以下载已有的预训练模型放在 checkpoints 文件夹下（若要下载预训练模型，可运行这个指令：bash ./scripts/download_model.sh）。

接着，测试图像时需要提供一个带掩膜的输入图像和一个灰度掩膜文件，应该确保掩膜文件覆盖输入图像中的整个掩膜区域，接着用以下代码设置测试时需要使用的参数：

```
$ python test.py \
-- model [stage]
-- checkpoints [path to checkpoints] \
-- input [path to input directory or file] \
-- mask [path to masks directory or mask file] \
-- output [path to the output directory]
```

例如用 places2 模型中的图像来测试：

```
$ python test.py \
 -- checkpoints ./checkpoints/places2 \
 -- input ./examples/places2/images \
 -- mask ./examples/places2/mask\
 -- output ./checkpoints/results
```

以上命令将在./examples/places2/images 中使用和./examples/places2/mask 对应的掩膜图像，并将结果保存在./checkpoints/results 目录中。可在./checkpoints/results 下查看图像补全的结果。

5. 结果展示

图 3.12 给出了部分需要修补的缺失图像，图 3.13 为补全效果图。

图 3.12　缺失图像展示

图 3.13　补全效果图展示

3.4　总结与展望

本实验所提出的图像补全方法证明了边缘信息可以优化图像补全的效果，但在此不做过多证明，有兴趣的读者可以自己做一些模型简化测试来进行验证。

同样，这个实验所用到的模型也不止可以用来作为图像补全的模型。例如，可以使用一张图片的左半部分的人脸信息和另一张图片的右半部分的人脸信息生成一个完整的人脸轮廓的边缘图像，再用此边缘图像生成一个彩色图像，得到一个全新的拥有两张图片特点的人。或者可以去除图像中的目标区域，利用掩膜处理图像中不想要的区域，再用本实验中的模型进行补全，就可以得到最终的效果图。诸如这种有趣的实验还很多，需要读者发挥自己的想象力去探索。

对于本实验，想更深入进行了解的读者可阅读文章 *EdgeConnect*：*Generative Image Inpainting with Adversarial Edge Learning* 进行深入理解和学习。2019 年的 CVPR 中的一篇文章 *Foreground-aware Image Inpainting* 也用了相似的思路，先推断生成轮廓边缘，来帮助修复缺失的区域，读者也可进行参考学习。另外，在有关基于深度学习的图像补全的方法中，2019 年发表的论文 *Coherent Semantic Attention for Image Inpainting* 中提出了另一种思路。这篇文章提出由于局部像素的不连续性，现有的基于深度学习的图像修复方法经常产生具有模糊纹理和扭曲结构的内容。为了解决这个问题，他们提出了一种基于深度生成模型的精细方法，不仅可以保留上下文结构，而且可以对缺失部分进行更有效的预测，通过对孔特征之间的语义相关性进行建模。任务分为粗略和精炼两个步骤，并在 U-Net 架构下使用神经网络对每个步骤建模。实验证明，该方法可以获得高质量的修复结果。在 2019 年的 CVPR 文章 *Pluralistic Image Completion* 中提出了一种用于多元图像完成的方法，一种新颖且以概率为原则的框架，该框架具有两条平行的路径，两者均受 GAN 支持。还引入了一个新的短期和长期关注层，该层利用了解码器和编码器功能之间的关系，

从而改善了外观一致性。在数据集上进行测试时，该方法不仅生成了更高质量的完成结果，而且还具有多种多样的合理输出，有兴趣的读者也可以进行深入阅读和学习。

3.5 参考文献

[1] 王妮娜. 图像修补方法的研究[D]. 南京：东南大学，2005.

[2] Pathak D，Krahenbuhl P，Donahue J，et al. Efros. Context encoders：Feature learning by inpainting. [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)，2016.

[3] Iizuka S，Simo-Serra E，Ishikawa H. Globally and locally consistent image completion[J]. ACM Transactions on Graphics (TOG)，2017，36(4)：107.

[4] Yang C，Lu X，Lin Z，et al. High-resolution image inpainting using multi-scale neural patch synthesis [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)，2017.

[5] Liu G，Reda F A，K. J. Shih K J，et al. Image inpainting for irregular holes using partial convolutions [C]//European Conference on Computer Vision (ECCV)，2018.

[6] Yu J，Lin Z，Yang J，et al. Generative image inpainting with contextual attention[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)，2018.

[7] Song Y，Yang C，Lin Z，et al. Contextual-based image inpainting：Infer，match，and translate[C]// European Conference on Computer Vision (ECCV)，2018：3-19.

[8] Yu J，Lin Z，Yang J，et al. Free-form image inpainting with gated convolution[EB/OL]. [2020-08-20]. http arXiv preprint arXiv：1806.03589.

[9] Isola P，Zhu J Y，Zhou T，et al. Image-to-image translation with conditional adversarial networks [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)，2017.

[10] J.-Y. Zhu J Y，Park T，Isola P，et al. Unpaired imageto-image translation using cycle-consistent adversarial networks[C]//The IEEE International Conference on ComputerVision (ICCV)，2017.

[11] Johnson J，Alahi A，Li. F F. Perceptual losses for real-time style transfer and super-resolution[C]// European Conference on Computer Vision (ECCV)，2016：694-711.