湖州师范学院

2023 届毕业设计(论文)

课 题 名 称: 基于生成对抗网络的图像翻译模型

课 题 名 称 (英文): Image translation model based on Generative Adversarial Networks

学 生 姓 名: 毛泽威 学 号: 2019082102

专 业 名 称: 计算机科学与技术

指 导 教 师: 张雄涛 职 称: 副教授

所 在 学 院: 信息工程学院

完 成 日 期: 2023 年 3 月 13 日

教务处制表

基于生成对抗网络的图像翻译模型

摘要：许多图像处理、图形学和计算机视觉的任务都是图像翻译任务，即输入一张图像，输出一张对应的图像。这些问题在之前都会被当作每个单独的问题来解决，分别为不同的问题设计了不同的损失函数，不同的模型结构，不同的训练范式和不同的算法。但是这些问题都可以归结为一个问题：像素到像素的映射。本文将使用条件生成对抗网络来解决图像翻译问题，条件生成对抗网络是一个通用的框架，它不仅学习输入图像到输出图像的映射，而且可以自适应的匹配数据集，不用再为每个问题设计损失函数。本文在条件生成对抗网络模型中的生成器上使用了“U-Net”的架构来避免信息丢失过多，而对于判别器使用卷积“PatchGAN”分类器，增强输出图像的效果并对基本的 CGAN 损失函数训练进行优化。基于生成对抗网络的图像翻译方法可以减少传统方法在损失函数上的设计投入，它不仅学习从输入图像场景到输出图像场景的映射，而且学习损失函数来训练该映射，这样就可以采用一个共通方式来解决不同问题的损失函数的设计。从实验结果上看，该模型可以适应语义设计图到真实图转换的应用环境，来实现图像翻译任务。

关键词：深度学习，生成式对抗网络，图像翻译

Image translation model based on Generative Adversarial Networks

Abstract: Many tasks in image processing, graphics, and computer vision involve image translation, which involves inputting an image and outputting a corresponding image. These problems will be solved as each separate problem before. Different loss function, different model structures, different training paradigms and different algorithms are designed for different problems. But these problems can all be attributed to one problem: pixel to pixel mapping. In this paper, we will use the condition generation countermeasure network to solve the image translation problem. The condition generation countermeasure network is a general framework, which not only learns the mapping of input image to output image, but also can adaptively match the data set, without having to design a loss function for each problem. This article uses the "U-Net" architecture on the generator in the conditional generative adversarial network model to avoid excessive information loss, and uses a convolutional "PatchGAN" classifier for the discriminator to enhance the output image effect. The image translation method based on the generation countermeasure network can reduce the design investment of the traditional method in the loss function. It not only learns the mapping from the input image scene to the output image scene, but also learns the loss function to train the mapping, so that a common formula can be used to solve the design of the loss function of different problems. From the experimental results, it can be seen that the model can adapt to various application environments to achieve image translation tasks

Keywords: Deep learning, Generative Adversarial Networks, Image translation

目　录

第一章 绪论

1.1 选题的意义

生成对抗网络（Generative Adversarial Networks，简称 GAN ）是当前人工智能学界最为重要的研究热点之一。其突出的生成能力不仅可用于生成各类图像和自然语言数据，还启发和推动了各种半监督学习和无监督学习类型的任务发展。二零一六年，生成对抗网络在人工智能领域顶级会议上引起很大的轰动，从 ICLR 到 NIPS，大量高质量论文被发表和探讨。生成对抗网络被誉为是“近二十年来机器学习领域最酷的想法”。

图像与声音、语言、文字等一样是一种信息载体，是人类在社会生产和生活中传递信息和交流沟通的基本的无法替代的方式。在语言翻译领域，相同的一句话，既可以用中文表达也可以用英文表达。所以图像作为一个载体也可以有许多存在方式，但是与文字中文英文表达不同的是，图像翻译是把所提供的场景的表示方式转变成另一个被希望的场景的图像域转化。比如一个图像场景从彩色图到灰度图的转换，又或者是莫奈的一种风格图到现实生活图的转换等。有监督学习，无监督学习和半监督学习是机器学习中常见的三类不同的方法。图像翻译的主要任务目标，就是通过基于有监督学习，半监督学习和无监督学习的技术，来寻找二个不同的图像之间的映射关系。而监督学习作为机器学习中最重要的一项学习方式，已经占据了目前各种机器学习算法的绝大部分。而监督学习则主要是在了解输入与输出关系的情况下训练出一个模型，把输入准确地映射到输出。具体来说，假设我们在开始训练之前就已经清楚了输入与输出，我们的主要任务目标便是构造出一个将输入准确映射到输出的模型，并且在向模型中加入新的信息之后便能够预测出正确的输出了。但无监督学习相比监督教学而言，它更不需要监督学习的知识标记过程，而且无监督学习也更富有创造性，因为能够获得想不到的知识之间的映射关系。

基于生成对抗网络的图像翻译方法可以减少传统方法在损失函数上的设计投入，它不仅学习从输入图像场景到输出图像场景的映射，而且学习损失函数来训练该映射，这样就可以采用一个共通方式来解决不同问题的损失函数的设计。本文旨在使用 CGAN 网络结构建立一个图像翻译模型，实现图像翻译任务，这对基于生成对抗网络的图像翻译模型的应用具有一定的推动作用。

1.2 研究现状及发展趋势

1.2.1 研究现状

图像本质上声音、语言、文字等一样是一种信息载体，是人类在社会生产和生活中传递信息和交流沟通的基本的无法替代的方式。随着多媒体技术和计算机信息技术的发展，图像逐渐成为一种人们在生活中表达情感和沟通交流的不可或缺的方式。图像可以在帮助人们快速理解信息的同时，保存和传递信息。因此，关于图像的研究与日俱增，而图像翻译作为图像研究的一种，是将图像由一种表现形式转换为另一种表现形式。从根本上来说，图像翻译是一种数据增强的方式，其逼真的生成结果增

加了目标图像域的数据数量。除了实现超分辨率重建图像[1-3]、图像去噪[4]、图像修复[5,6]、图像压缩[7]等图像处理操作，图像翻译还在其它领域都得到了广泛应用。

生成对抗网络 GAN 已形成了如今在深度学习技术领域中比较前沿的技术之一，并且在深度学习领域中再次带来了一场革命。而这场变革也已经产生了一些重大的技术突破。例如，Ian Goodfellow 等人最早在“Generative Adversarial Networks”上就提出了生成对抗的网络这一个概念。现在无论时科学界还是工业界也已开始接受并欢迎生成对抗网络的存在。所以生成对抗网络的存在也不可避免。

首先，生成对抗网络最厉害的特点就是它的学习活动是无监督的。由于生成对抗网络基本上不需要记录知识，这也就要求生成对抗网络功能越来越强，因为知识记录的功能非常枯燥。它可以生成高质量的文本图片，将文字图像加强，可以根据文本生成图像，将文字图片的某个部分变换成另一个部分，还有随着时间增长的脸部变化等等。对于图像翻译的问题，国内外目前的研究方法主要分为基于监督学习的研究以及基于非监督学习的研究方法，它们所使用的网络框架主要集中在卷积神经网络 CNN 和生成式对抗网络 GAN。图像翻译问题最早的提出者是 Hertzmann[8]，他通过在一个输入和一个输出的图像训练集上使用了一个非参数的模型，从而首次实现了这个功能。后来，随着卷积式神经网络 CNN 不断深入发展并且在许多的图像翻译的领域中都取得了比较好的训练效果之后，基于卷积式神经网络的图像翻译领域也随之得到了十分快速的发展，之后 Long 和 Shelhamer[9]等人提出一种新的方法，即语义分割方法，他们对输入图像进行完全卷积，并且引入了基于每个像素对的损失函数来训练网络，来寻找像素之间的映射关系。最后，他们得到了可以将每个像素都映射到目标领域上的图像领域间的映射函数。之后 Zheng，Jayasumana[10]等人在 Long 和 Shelhamer 等人基础的之上，通过将 CRF 推断作为与其他部分联合训练的循环层，也取得了非常好的训练效果。之后 Noh，Hong[11]等人通过使用在 in-network 下采样，这个做法大大减小了特征映射的空间范围，然后再使用在 in-network 上的采样来输出最终结果，这一做法取得了很好的效果并且得到了学术界的广泛认可。但是，基于卷积神经网络的图像翻译也面临着一个不可回避的困难：尽管整个系统的训练流程都是自动的，但损失函数仍然必须人为的去设置。因此卷积神经网络必须明白在它的训练流程中所必须且尽力要减少的这个损失函数是什么，不然卷积神经网络训练出的结果会是我们所不想要的。所以损失函数对于对卷积神经网络而言非常关键，但因为损失函数的设计不够简洁，模型在训练过程中也会出现不好的结果，这个结论在 D.Pathask[12]，R.Zhang 等人[13]的项目中验证了，在他的项目中也验证了单一的网络结构会造成的图像模糊。不过，制定一个正确的损失函数，并且对卷积神经网络加以训练，得到一个非常逼真的图像存在很大的困难，完成这种工作很多地方必须掌握针对实际应用的知识才能够完成。故损失函数的设计存在着大量问题，不仅耗费大量的人力资源，而且设计出来的损失函数也不一定适合我们所要解决的问题模型，直接影响了模型的性能。

基于生成对抗网络图像翻译：是由 Goodfellow 等人在二零一四年设计的生成对抗网络(GAN)的生成模型，在这之前生成模型需要人为设计损失函数，而损失函数设计的好坏直接影响了模型的性能，并且在运行是需要足够丰富的计算机资源[14]。例如，深度信念网络[15] DBN 变分自编码器[16](Variational auto-encoder，VAE)，受限波尔兹曼[17](Restricted Boltzmann Machine，RBM), 有向图模型的赫姆霍兹[18](Helmholtz machines)等。生成对抗网络的输入为随机噪声，输出为模拟信息。生成对抗

网络主要是由机器学习的样本量分布的生成元和判别信息真实性的判定单元所组成，同时引入了极小极大博弈的模式并完善了它。经过大量模型运用和实验验证，生成对抗网络能够得到更接近于真实世界质量的结果，也因此解决了早期模拟研究中出现的数据精度差和信息泛化能力差的问题。之后，Radford[19]等提出的(Deep convolutional GAN)，Mirza 等人[20]提出的 CGAN (Conditional Generative Adversarial Networks)，Odena 等[21]提出的 ACGAN (Auxiliary classifier Generative Adversarial Networks)，Miyato 等[22]等人探讨了不同的加入条件信息对整个建模方法的作用效果，并提出了将条件信息加入判别器中的新思路，从而提高了输入条件信息对整个图像处理技术的指导作用。Larsen 等人[23]将生成模型变分自编码器(Variational auto-encoder，VAE)[24]与 GAN 相结合提出了 VAE-GAN。Bao 等人[25]在 VAE-GAN 的基础上，将加入分类器的类别标签给予了 VAE-GAN，能够获得更为真实和多样化的信息，同时也可以广泛应用于影像恢复、超分辨率的以及影像强化方面的工具。Zhang 等人[26]在将自注意力机制融入生成器和判别器中提出了 SAGAN(Self-Attention Generative Adversarial Networks)，在 ImageNet 资料的实验证明，SAGAN 提高了 Generative Adversarial Networks 的生成的图像的准确率和清晰度。GAN 模型框架的发展，更进一步拓展了应用，为图像翻译建模以及其他具体模型提供了方便条件。如今，生成对抗网络的研究发展逐渐走向成熟，基于生成对抗网络的图像翻译模型也逐渐火热，从最早的基于 CGAN 有监督图像翻译模型 pix2pix，后来为了解决 pix2pix 架构模型多样性差的问题的 BicycleGAN，以及无监督的图像翻译模型 CycleGAN 等还有研究者们借鉴了对偶学习的思想，通过循环一致性损失实现了无配对数据的图像翻译，使得图像翻译在无监督学习的模式下也取得了显著的成果。

1.2.2 发展趋势

图像翻译技术发展迅速，成为世界计算机技术发展前沿和重要内容。图像翻译技术不仅仅使用在学术研究，更多服务于在商业民用等其他方面。在过去几十年中，图像翻译技术的发展有令人瞩目的成就，从最初的卷积神经网络到后来出现的生成对抗网络，每一步都标志着新的发展，更好地服务于社会，促进社会发展。未来，图像翻译技术将会越来越强大，在不同领域改变社会生活。

1.3 本章总结

本文的绪论部分主要介绍了选题的意义和生成对抗网络的研究现状和发展趋势，主要包括以下几个 方面：

1. 介绍了图像翻译的基本概念和不同的方法解决图像翻译问题。
2. 介绍了生成对抗网络对图像翻译的推动作用。
3. 介绍了卷积神经网络 CNN 在图像翻译的发展和生成对抗网络 GAN 对图像翻译的推动发展。
4. 介绍了卷积神经网络 CNN 与生成对抗网络 GAN 的模型提出历史过程，以及当时的人们突出的模型所解决的问题。

第二章　基于生成对抗网络的图像翻译相关理论介绍

本章节首先介绍关于图像翻译的含义，其次介绍基于卷积神经网络 CNN 的传统图像翻译的基本原理和和其在图像翻译上的缺陷，然后介绍生成对抗网络 CAN 的基本原理，最后介绍条件生成对抗网络 CGAN 的基本原理和 pix2pix 模型，并对它们的过程进行对比。

2.1　图像翻译的含义

我们首先必须明白这两个概念：第一个概念是图像内容，它是图像的固有内容，是区分不同图像的基础。第二个是图像域，图像域内的图像可以认为是图像内容被赋予了某些相似的属性特征。在图片输入莫奈的风格画，可以输出现实生活图，图像中的动物可以在斑马和马之间相互转化，图像中的季节可以的冬夏之间相互转化。这里的图像内容是抽象出来的图像内容。而图像翻译方法可以把图像中内容从一个图像域 X 转换到另一个图像域 Y，可以看作是将原始图像的某种属性特征 X 删除，重新赋予其新的特征 Y，即图像间的跨域转换。目前，图像翻译是由基于卷积神经网络 CNN 和基于生成对抗网络 GAN 这两种主要技术实现，并且图像翻译任务在图像风格化、超分辨率图像生成、颜色填充、白天黑夜的转换、四季变换等视觉领域都有着广泛的应用。

2.2　卷积神经网络的基本原理

卷积神经网络 CNN 是一种深度人工神经网络，它的基本原理是通过在数据上提取特征，然后利用这些特征来识别和分类数据，也就是所谓的模式识别。它可以用于复杂的图像识别，比如影像分析，行为识别，自然语言处理等等。CNN 的特征提取是通过给定一个输入层,传入输入数据，在传入后，用多个卷积层对输入的图片进行特征提取，在这里，称之为卷积核，卷积核是一种空间滤波器，它把一幅图中目标与背景区分开来，抽取出输入图片中部分特征。卷积层一般分为池化层和全连接层，池化层可以降低训练时间，并减少参数，全连接层可以实现对加权和激活函数处理，方便进行特征提取。在反向传播的过程中，CNN 会把图片的输入，经过激活函数，把它分为两部分，把特征的细节信息和抽象特征融合在一起，用反向传播技术把特征信息反馈给 CNN 模型，以调整权重和偏置值以达到最优解。另外，CNN 使用 dropout 技术可以有效的防止模型过拟合，既能够保持模型的准确率又不会过度拟合，让模型更加准确。卷积神经网络 CNN 应用十分广泛，例如，VDSR 残差网络(Resnet)。

卷积神经网络 CNN 共享卷积核，能处理更高纬度的数据，并且无需手动选取特征，在训练较好的网络中相同的权重就可以得到很好的效果，故卷积神经网络 CNN 在图像处理领域中有着很重要的地位。但是卷积神经网络 CNN 也同样有其缺点，在图像翻译领域中的每一个问题都需要使用一个预先训练好的 CNN 网络来完成损失函数计算，也因此在卷积神经网络 CNN 开始训练之前，就需要人为地调整参与设计的损失函数，并且调参工作也一直是深度学习中较为繁杂的问题之一，而卷积神经网络中的最后一层全连接层中参数也过于冗余，并且所有的数据都需要直接上传到全连接层计算中，也因此导致了训练复杂度比较高，并且收敛速度也比较缓慢。

2.3 生成对抗网络的基本原理

生成式对抗网络其实是一个双方的相互博弈的过程。最基础的生成式对抗网络是任给一组输入的随机噪声，经过它的网络最核心的两个功能：判别器 D(Discriminative)和生成器 G(generative)。G 的任务是随机生成以假乱真的合成数据。为了满足随机性，人们通常会使用多个随机数作为 G 的输入，例如一百个从标准正态分布中随机采样得到的随机数，记作随机噪音 z，输出则是和真实图片相同分辨率的图片 D 的任务是区分真假，即判断一张图片到底是真实图片，还是 G 合成的虚假图片。因此，D 的输入是图片，输出是一个分数，分值越高表示输入图片越真实。理想情况下，D 应当对于所有真实图片都输出高分，对于所有虚假图片都输出低分，如此一来，便可以完美实现判别真假的目标。

在具体实现上，G 和 D 都可以通过神经网络来实现，并且都包含大量的可调参数。在经过初始化之后，G 不具备任何的生成能力，输出的虚假图片和真实图片相去甚远；D 也不具备任何的判别能力，对于真实或虚假图片所输出的分数没有太大区别。现在开始模型的优化，在每次迭代中，随机选择一批真实图片 x，并随机生成一批 z，然后将 z 输入 G 得到一批虚假图片 x'=G(z)。D 的损失函数包括两方面：第一是 x 对应的分数 D(x)应当比较高，例如和 1 尽可能接近；第二是 x'对应的分数 D(x')应当比较低，例如和 0 尽可能接近。按照损失函数减小的方向，调整 D 的每一个参数，便完成了 D 的一次优化。在经过优化之后，D 对于真实或虚假图片所输出的分数，就更加有区分度了至于 G，其目标是让 D 误以为 x'是真实图片，因此 G 的损失函数可以是 D(x')和 1 之间的差距，这个差距越小，表明在 D 看来 x'越真实。按照损失函数减小的方向，调整 G 的每一个参数，便完成了 G 的一次优化。在经过优化之后，G 合成的虚假图片，在 D 的判别下，就变得更加真实了。值得一提的是，在整个优化过程中，G 并没有接触到真实图片 x，它也不在乎真实图片 x 长什么样，只要合成的虚假图片 x'在 D 看来像真的就行重复以上步骤，随着优化的进行，D 的判别能力越来越强，G 的生成能力也越来越强，两者互相博弈、共同进步。在理想的情况下，G 最终可以生成和真实图片难以区别的虚假图片，如此一来，为了合成一些不存在的图片，只需要随机生成很多 z，输入 G 即可得到对应的合成结果。其中损失函数公式(2-1)为：

$$V(D,G)=\mathbb{E}_{x\sim\mu}(x)[\log D(x)]+\mathbb{E}_{z\sim y}(z)\left[\log\left(1-D\left(G(z)\right)\right)\right] \qquad (2-1)$$

其公式中 V(D, G)表示损失函数，$\mathbb{E}$表示关于下标中指定分布的期望，$x\sim\mu$表示来自分布μ的训练样本，$z\sim\gamma$表示来自分布γ的训练样本。

故 GAN 解决极小极大值的描述为公式(2-2)：

$$\min_{G}\max_{D}V(D,G)=\mathbb{E}_{x\sim\mu}(x)[\log D(x)]+\mathbb{E}_{z\sim y}(z)\left[\log\left(1-D\left(G(z)\right)\right)\right] \qquad (2-2)$$

GAN 的训练并非普通的凸优化问题，最大最小优化问题实质上是在找一个鞍点，随着训练次数增加，G、D 的生成、判别能力越来越强，并且 D 的判别能力很容易影响 G 的生成能力，WGAN 从理论上证明当 D 训练到最好的情况下，原始 GAN 中 G 的损失函数（实质上是一种对真实分布与生成分布的距离度量，如 KL 距离、JS 距离、Wasserstein 距离等）存在自相矛盾的缺陷，这也就导致 GAN 很难训练，微调敏感。另外还有一种问题，模型学习到真实样本的分布一部分，导致模型生成的样本非常

单一，样本差异较小，模式崩溃。

2.4 条件生成对抗网络的基本原理

条件生成式对抗网络 CGAN 是在 GAN 基础上做出的一种发展，通过给原始 GAN 的生成器 G 和判别器 D 添加额外的条件 y，建立一种条件生成模型。CGAN 额外的条件信息可以是类别标签或者其它的辅助信息，因此 CGAN 的提出使得 GAN 可以利用图像与对应的标签（lable）进行训练，并在测试阶段利用给定标签（lable）生成特定图像。

2.5 Pix2pix 模型的基本原理

Pix2pix 可以将一种图像转换成另一种图像，是一种应用非常广泛的基于深度学习的图像生成模型，生成器的作用是将输入的图像转换成真实的图像，例如输入黑白图像转换成彩色图像，输入素描图转换成真实图像。在训练过程中生成器不断的生成图像，而判别器不断去判断生成的图像是否为真实的图像。如果生成的图像不真实，那么生成器不断去调整自己的参数，在重新生成图像，直到判别器认为生成的图像为真实的图像。若判别器无法判断生成器生成的图像为真还是为假时，模型训练完成。Pix2pix 的训练过程非常复杂，需要大量的计算和数据。在模型的训练过程中，需要使用非常多的图像来训练生成器和判别器，这些图像对既包括输入图像也包括输出图像。

2.6 CGAN 与 Pix2pix 模型

2.6.1 CGAN 模型

CGAN 模型如图 2.1 所示。

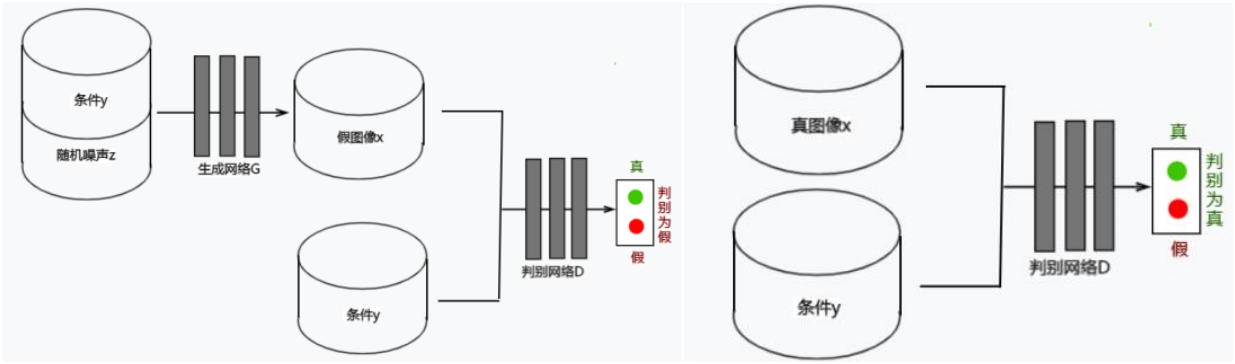

图 2.1 CGAN 模型判断的过程

在原始生成对抗网络的基础上增加一个条件，这个条件的种类是非常丰富的，可以是图像的类别，也可以人脸图像的面部表情等，这样网络训练会更加容易。生成式对抗网络输入只有噪声，条件生成式 对抗网络将噪声和条件同时输入。图 2.1 为 CGAN 模型判断为假的过程，输入生成网络 G 随机噪声 z 和条件 y，生成网络 G 输出一个 假图像 x，并将假图像 x 与条件 y 一起输入判别网络 D，判别网络 D 判断输入的图像为假图像。图 2.2 为 CGAN 模型判断为真的过程，将真图像 x 与条件 y 输入判别

网络 D，判别网络 D 判断输入的图像为真 图像。

2.6.2 Pix2pix 模型

Pix2pix 模型如图 2.2 所示。

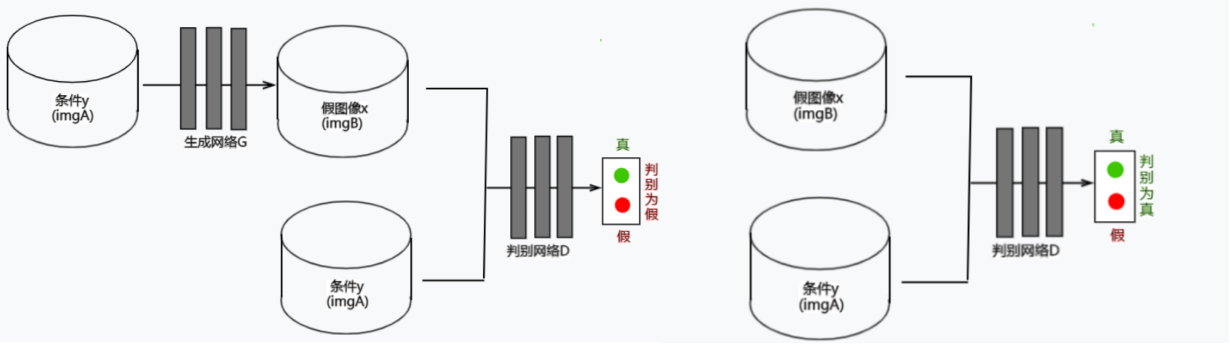

图 2.2 pix2pix 模型判断的过程

pix2pix 是 CGAN 的一个变体，能够实现从图像到图像的映射，在从标签映射合成照片、从边缘映射重建对象、图片上色等多类人物的表现较好。它比较适合于监督学习， 即图像的输入和它的输出是相互匹配的。所谓匹配数据集是指在训练集中两个互相转换的领域之间有明 确的一一对应数据。在工程实践中研究者需要自己收集这些匹配数据，但有时同时采集两个不同领域的 匹配数据是麻烦的，通常采用的方案是从更完整的数据中还原简单数据。比如直接将彩色图片通过图像 处理的方法转为黑白图片。

图 2.3 为模型判断为假的过程，生成网络 G 的输入只有一个条件 y(可以将条件 y 理解为一张图片 imgA)，生成网络 G 生成假图像 x(imgB),并将假图像 x(imgB)与 y(imgB)一起输入判别网络 D，判别网络 D 判断输入图像为假图像。图 2.4 为模型判断为真的过程，将真图像 x(imgB)与 y(imgA)输入判别网络 D， 判别网络判断输入图像为真图像。

第三章　基于生成对抗网络的图像翻译方法

本文实验在生成器上，使用了基于“U-Net”的架构，而对于判别器，使用卷积“PatchGAN”判别器。

3.1 U-Net 生成器

U-Net 生成器是一个常用于医学图像分割的全卷积模型。它分为两个部分，其中左侧是由卷积和降采样操作组成的压缩路径，右侧是由卷积和上采样组成的扩张路径，扩张的每个网络块的输入由上一层上采样的特征和压缩路径部分的特征拼接而成。网络模型整体是一个 U 形的结构，因此被叫做 U-Net。U-Net 模型详细如图 3.1 所示。先是编码器再是解码器，其中编码器逐层下采样，中间是瓶颈层，解码器逐层上采样。上采样与下采样是对称的，输入一张图在生成一张图，而输入的图与输出的图的大小是一样的。而瓶颈层的维度又是很小的，不可避免会导致信息的丢失。如果信息丢失过多会导致模型训练失败，为了避免信息的丢失在 U-Net 上引入了跳转链接，对称层的信息按通道方向摞起来，这样能够使底层特征与高层特征融合。

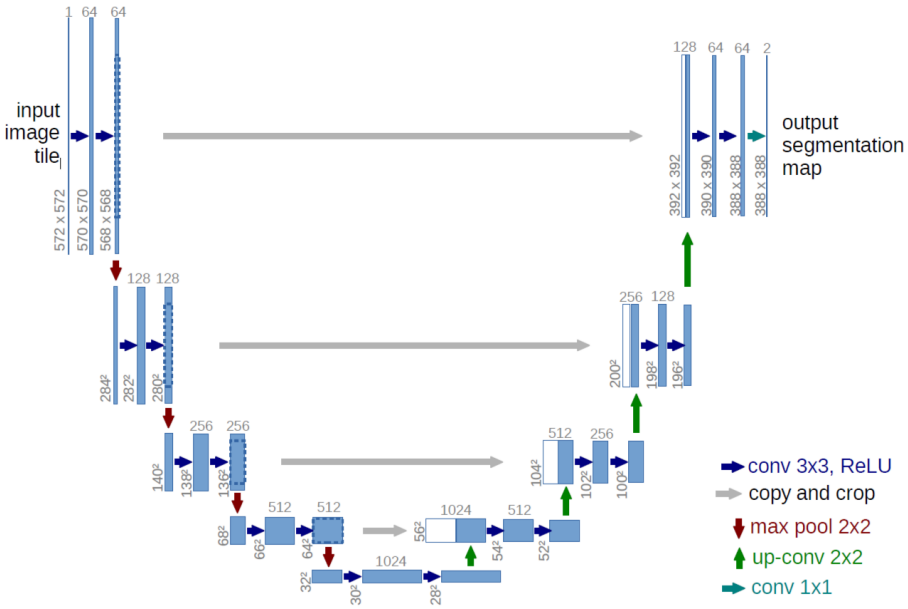

图 3.1 U-Net 模型

其中那些长条的上方的数字（如 64，128 等）都是通道数；长条侧边的数字（如 572×572）这些是尺寸大小；蓝白相间上的通道数是两份的总和，即白的和蓝的通道数各是那个通道数的一半（比如下面这个蓝白相间通道数为 128，即代表白的 64，蓝的也是 64）。蓝色和白色框表示 feature map，蓝色箭头（conv 3×3，ReLU）表示 3×3 卷积，用于特征提取；灰色箭头（copy and crop）表示 skip-connection（跳跃连接），用于特征融合；红色箭头（max pool 2×2）表示最大池化，用于降低维度；绿色箭头（up-conv 2×2）表示上采样，用于恢复维度；天蓝色箭头（conv 1×1）箭头表示 1×1 卷积，

用于输出结果。

3.2 PatchGAN 判别器

传统 GAN 的一个棘手的问题是它生成的图像普遍比较模糊，一个重要的原因是它使用了整图作为判别器的输入。不同于传统的将这个图作为判别器判别的目标（输入），本文实验将输入图像分成 N×N 个图像块(Patch)，然后将这些图像块依次提供给判别器，因此这个方法被命名为 PatchGAN，PatchGAN 可以看作针对图像纹理的损失。PathGAN 预测 N×N 区域大小的概率图，最后取平均值，更好地评估高频信息。

图像不同的区域有不同频率的信息，有些区域信息比较重要，而另一些区域信息却没这么重要。采用 Patch 这样一种判别模式，可以更好地对图像局部的真实性去评估。这是 PatchGAN 和别的判别器之间的不同之处。既然 GAN 只用于构建高频信息，那么就不需要将整张图片输入到判别器中，让判别器对图像的每个大小为 N×N 的 patch 做真假判别就可以了。因为不同的 patch 之间可以认为是相互独立的。本文对一张图片切割成不同的 N ×N 大小的 patch，判别器对每一个 patch 做真假判别，将一张图片所有 patch 的结果取平均作为最终的判别器输出。

3.3 设计损失函数

CGAN 的目标可以表示为公式(3-1)：

$$L_{CGAN}(G,D)=\mathbb{E}_{x,y}[\log D(x,y)]+\mathbb{E}_{x,z}\left[\log\left(1-D\left(x,G(x,z)\right)\right)\right] \quad (3-1)$$

其中$D(x,y)$为判别器认为真图是真图的概率，$D\left(x,G(x,z)\right)$为判别器认为假图是真图的概率。如果输入是真图的话，判别器认为真图是真图的概率要足够的大，即：$\mathbb{E}_{x,y}[\log D(x,y)]$要足够的大；如果输入是假图的话，判别器认为是假图是真图的概率要足够的小，即：$\mathbb{E}_{x,z}\left[\log\left(1-D\left(x,G(x,z)\right)\right)\right]$要足够的 小。生成器试图最小化该目标$L_{CGAN}(G,D)$，而对手判别器试图最大化该目标$L_{CGAN}(G,D)$，这就是双人极大极小的零和博弈，即：$G^*=arg\min_G\max_D L_{CGAN}(G,D)$。但是生成器不能控制判别器认为真图是真图 的概率，即：$\mathbb{E}_{x,y}[\log D(x,y)]$的值无法被生成器改变，它只能去改变判别器认为假图是真图的概率，即：$\mathbb{E}_{x,z}\left[\log\left(1-D\left(x,G(x,z)\right)\right)\right]$，正如枯叶蝶没办法控制鸟对真实树叶的看法，它只能控制鸟对自己的看法。 为了检验判别器条件作用的重要性，还将其与判别器不观察 x 的无条件变量进行对照实验公式(3-2)：

$$L_{CGAN}(G,D)=\mathbb{E}_{y}[\log D(x,y)]+\mathbb{E}_{x,z}\left[\log\left(1-D\left(x,G(x,z)\right)\right)\right] \quad (3-2)$$

该实验中判别器不是 CGAN，生成器仍然是 CGAN。过去的算法，不光要设计 GAN 的损失函数，还要 加一个 L2 损失函数，鼓励生成的图与输入的图在像素层面上越接近越好。判别器的工作保持不变，但 生成器的任务不仅是愚弄判别器，而且还要接近地面实值输出在 L2 意义上。 故再加一个 L1 鼓励生成 图与输入图在像素上接近，即：$|y-G(x,z)|$的值要小，具体见公式(3-3)。

$$L_{L1}(G)=\mathbb{E}_{x,y,z}[\|y-G(x,z)\|_1] \quad (3-3)$$

故最终的目标为公式(3-4)：

$$G^* = arg \min_G \max_D L_{CGAN}(G,D) + \lambda L_{L1}(G) \quad (3-4)$$

GAN 中噪声是用于引入随机性的，如果没有噪声，没有随机数，网络就总会生成确定性的结果，这 个叫做狄拉克δ函数。（狄拉克δ函数:在 x 为 0 的时候，y 不为 0；在 x 不为 0 的时候，y 为 0）过去的 条件 GAN 公认噪声服从高斯分布的话，效果会比较好。噪声是输入给生成器的，直接加在条件 GAN 的上 面。但是这样的话，生成器直接忽略了这个噪声，加入条件 GAN 上无用。故在本文实验中使用 dropout 引入随机性，不管是在训练的时候，还是在预测的时候，每一次都随机掐死一部分神经元，阻断其前向 L2 或者反向信息流。随机掐死引入了随机性。但是尽管引入了 dropout，但是发现网络并没有很大的随机性。如何设计条件 GAN 让它去生成足够高质量的并且足够有随机的输出，这是未解决的问题。

3.4 优化训练方式

在训练阶段，对于生成对抗网络，交替训练生成器和判别器。在刚开始时，生成器生成的图像非常假，$D(G(x,z))$非常小，判别器可以轻松判别图像为假。当 x 接近 0 时，$log(1-x)$图像没有梯度，会导致 CGAN 出现饱和现象，不便于网络学习，故不再最小化$log(1-D(G(x,z))$，而是最大化$D(G(x,z))$。本文实验的优化目标包含两个部分。一部分是 CGAN 的优化目标；另一部分是 L1，用来约束生成图像和真实图像之间的差异，鼓励生成图与输入图在像素上接近，这部分借鉴了其他基于 GAN 做图像翻译的思想，只不过这里用 L1 损失函数而不是 L2 损失函数，鼓励生成的图与输入的图在像素层面上越接近越好，目的是减少生成图像的模糊。对于 L2 而言，要满足它是无偏估计的前提应该是目标的误差符合高斯分布；然而在图像模糊中，模糊的误差并不符合高斯分布，用 L2 进行约束的效果较差，甚至不如 L1 进行约束。因此采用了 L1 来减少生成图像的模糊。

3.5 训练评价指标

评估合成图像的质量是一个非常重要而困难的问题。传统的评价指标（如逐像素的做差平方求和算 L1 和 L2 距离的方法）实际上是不科学的，这种方法没有评估结构信息，只是从像素层面去计算，而不是从宏观的纹理特征和语义上去评估。

第一种评价指标是用人工的方式去打分，寻找一些志愿者，进行人工打分。将网络模型生成图与真实图放一起，让志愿者判断哪些是网络模型生成图，哪些是真实的图像，如果网络模型生成图骗过了这些志愿者，让这些志愿者认为生成的图片是真实的，就表明网络模型训练不错。具体如下，每一张图片只给志愿者展示一秒钟，但是有无限多的时间去反应哪张是真的哪张是假的, 前面几张张用来作于新手教学，后二十张用于正常测试，最后统计各个志愿者的判断结果。

第二种评价指标是用自动化 AI 的方式，寻找一个现成的训练好的图像分割模型，如果生成出来的照片，跑一个现成的图像分割模型，如果能判断生成的图像为真实图像，就表明网络模型训练不错。例如， 现成的图像分割模型“FCN-socre”。用 pix2pix 将语义图转为真实图片，再将真实的图片输入

到 FCN 网络预测语义图，将这个语义图与最初的语义图进行比对，即可判定生成的结果。

3.6 模型算法具体流程

具体模型算法流程图如图 3.2 所示。

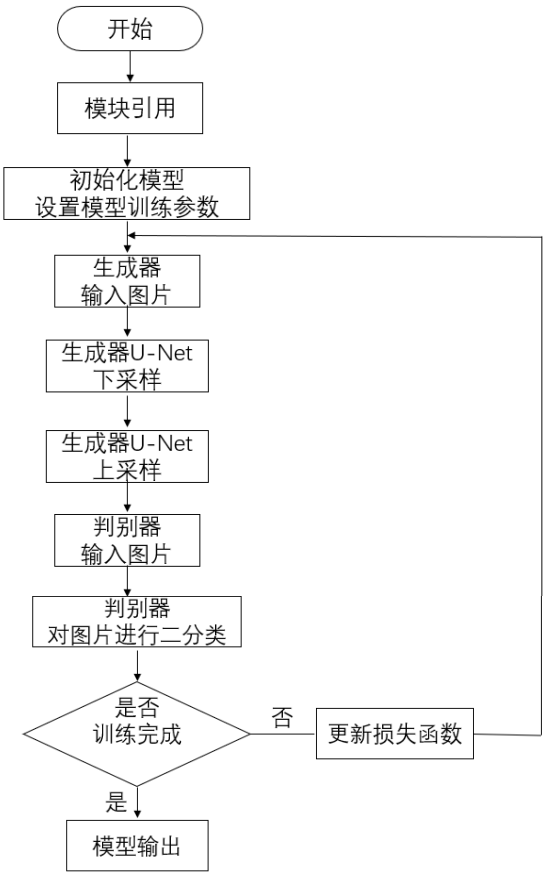

图 3.2 具体模型算法流程图

第四章 实验

4.1 实验环境配置

pix2pix(Image-to-Image Translation with Conditional Adversarial Nets)的论文作者团队在二零一七年使用 Lua 语言和用 Lua 语言实现的 Touch 深度学习框架完成了 pix2pix 的代码实现。在人工智能刚爆发的时间，Python 还没有成为人工智能、机器学习的主要语言，在那时，作数值运算、作科学运算用的很多都是 Matlab，而 Matlab 是一家商业公司的软件，不可能支撑起开源的免费社区，所以当时 pix2pix 作者团队使用 Lua 语言来实现 pix2pix 的代码。但 Python 由于其简洁优美和极高的开发效率，且具有丰富和强大的类库，同时具有优良的可扩展性和平台可移植性，它能够很轻松的把用其他语言制作的各种模块轻松地联结在一起，Python 逐渐成为人工智能的主要使用的语言。

本文将对原 Lua 语言实现的 pix2pix 代码使用 Python 语言来复现(具体核心代码见附录)，实现图像翻译的任务。

如下是实验环境配置的具体流程：

1．使用命令 conda env create -f environment.yml 来创建一个新的 conda 环境。

```
(base) C:\Users\Mao Zewei>cd C:\Users\Mao Zewei\Desktop\pytorch-CycleGAN-and-pix2pix-m

(base) C:\Users\Mao Zewei\Desktop\pytorch-CycleGAN-and-pix2pix-master>conda env create

CondaValueError: prefix already exists: C:\MySoftware\anaconda\envs\pytorch-CycleGAN-a
```

2.安装完成后，使用命令 conda info --envs 来查看当前已有环境。

```
(base) C:\Users\Mao Zewei\Desktop\pytorch-CycleGAN-and-pix2pix-master>conda info
# conda environments:
#
base                  *  C:\MySoftware\anaconda
MZW                      C:\MySoftware\anaconda\envs\MZW
pytorch-CycleGAN-and-pix2pix     C:\MySoftware\anaconda\envs\pytorch-CycleGAN-ar
```

3. 在 anaconda 里安装 visdom 插件。使用命令 pip install visdom。

```
C:\Users\Mao Zewei\Desktop\pytorch-CycleGAN-and-pix2pix-master>pip install visdom
visdom in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site-packages
tornado in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site-packages
numpy>=1.8 in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site-packa
requests in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site-package
six in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site-packages (fr
scipy in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site-packages
websocket-client in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site
jsonpatch in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site-packag
networkx in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site-package
pillow in c:\mysoftware\anaconda\envs\pytorch-cyclegan-and-pix2pix\lib\site-packages
```

4.使用命令 conda activate Pytorch-CycleGAN-and-pix2pix 进入已创建的 Pytorch-CycleGAN-and-pix2pix 环境。

```
ytorch-CycleGAN-and-pix2pix-master>conda activate pytorch-CycleGAN-and-pix2pix

sers\Mao Zewei\Desktop\pytorch-CycleGAN-and-pix2pix-master>
```

至此，实验环境配置完成。

4.2 实验设备

具体的实验平台主要配置如表 4-1 所示：

表 4-1 实验平台主要配置

实验平台	设备名称
CPU	Intel core i7-9750H
GPU	Nvidia GTX1660Ti
RAM	Samsung DDR4 2666MHz 16GB
硬盘	Samsung SSD PM981 1TB
操作系统	Windows 11
程序语言	Python 3.11
编译器	PyCharm Community
Pytorch 版本	Pytorch 1.8.1
Tourchvision	Tourchvision 0.9.1

4.3 训练集

本文将 pix2pix 更具体的应用在将建筑外立面语义设计图转成真实外立面建筑图的任务。建筑外立面语义设计图转成真实外立面建筑图这个图像翻译任务主要体现在机器学习这两个方面的任务：图像生成（图形学任务）和语义分割（计算机视觉任务）。

图像生成是指从现有数据集生成新的图像，给模型一组数据，希望从数据中学习到信息后的模型能够生成一组和训练集尽可能相近的数据，本文模型学习建筑外立面语义设计图与真实外立面建筑图的映射，来生成外立面建筑图。语义分割是指对图片中的像素进行类别上的分类，本文模型将建筑外立面语义设计图和真实建筑外立面图分别进行语义分割，得到不同物体的轮廓，如窗户、大门、屋顶等，然后进行生成学习。

建筑外立面语义设计图转成真实建筑图的数据集选择了 400 张建筑外立面语义设计图和 400 张真实建筑外立面图，该训练集来自于 CMP Facade 数据集并对此进行修改。

部分建筑外立面语义设计图训练数据集和部分建筑真实外立面图训练数据集见图 4.1 所示。

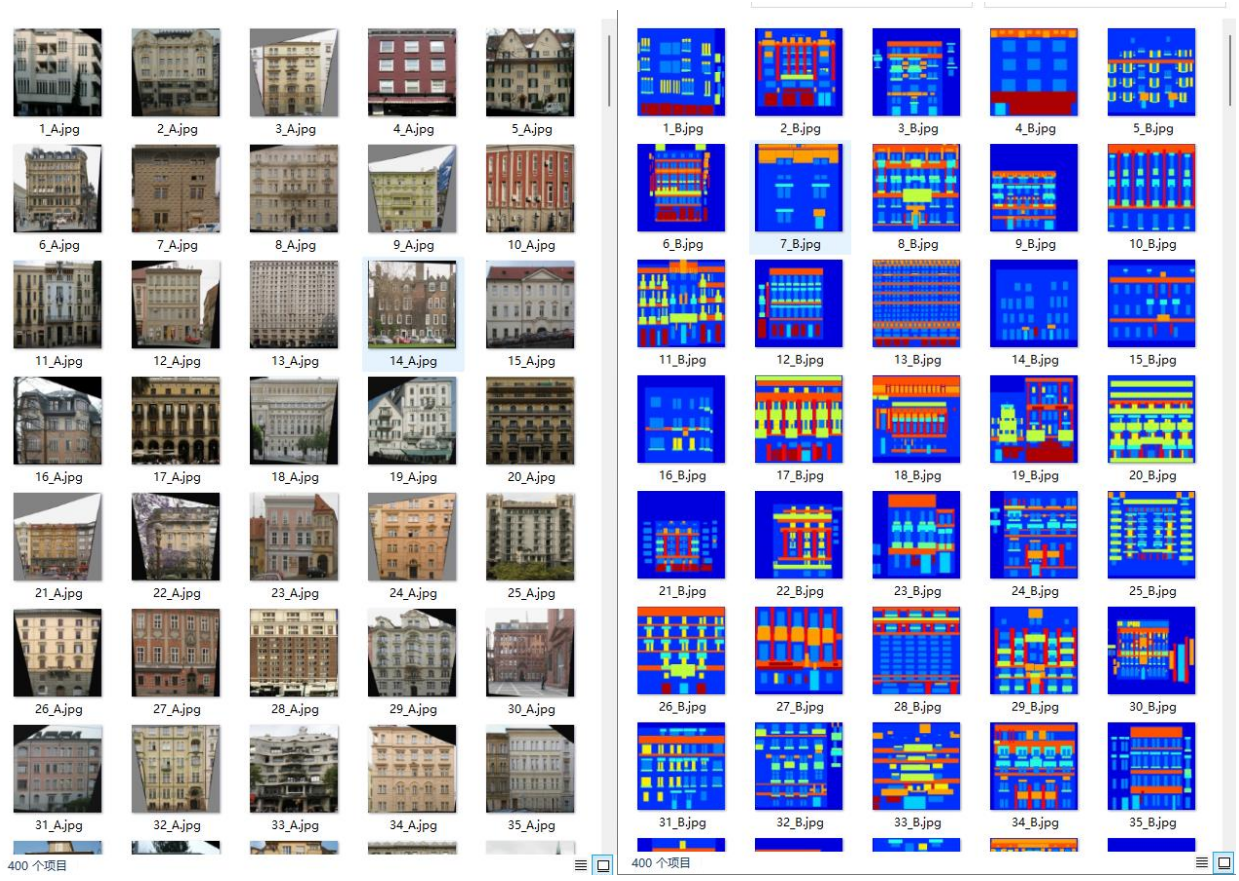

图 4.1　建筑外立面语义设计图和建筑真实外立面图部分训练集

4.4　模型训练

本文将训练两种 epoch 不同而 patch 相同的模型生成的真实外立面建筑图。

4.4.1 epoch 为 200 的训练模型

该模型具体训练参数如图 4.2 所示：

```
b1: 0.5
b2: 0.999
batch_size: 1
checkpoint_interval: -1
data_root: C:\Users\Mao Zewei\Desktop\data
dataset_name: facades
decay_epoch: 100
epoch_num: 200
epoch_start: 0
exp_name: Exp0
img_height: 256
img_result_dir: result_images
img_width: 256
in_channels: 3
lambda_pixel: 100
lr: 0.0002
model_result_dir: saved_models
n_D_layers: 3
n_cpu: 8
out_channels: 3
sample_interval: 200
which_direction: BtoA
```

图 4.2　epoch 为 200 的训练模型具体参数

4.4.2 epoch 为 100 的训练模型

该模型具体训练参数如图 4.3 所示：

```
b1: 0.5                                          img_result_dir: result_images
b2: 0.999                                        img_width: 256
batch_size: 1                                    in_channels: 3
checkpoint_interval: -1                          lambda_pixel: 100
data_root: C:\Users\Mao Zewei\Desktop\data       lr: 0.0002
dataset_name: facades                            model_result_dir: saved_models
decay_epoch: 50                                  n_D_layers: 3
epoch_num: 100                                   n_cpu: 8
epoch_start: 0                                   out_channels: 3
exp_name: Exp1                                   sample_interval: 200
img_height: 256                                  which_direction: BtoA
```

图 4.3 epoch 为 100 的训练模型具体参数

4.4.3 训练

在 PyCharm 的终端中输入命令，对模型进行训练。

（python pix2pix_train.py --data_root "C:\Users\Mao Zewei\Desktop\data" --which_direction "BtoA"）

```
[Epoch1/200]-[Batch5/800]-[Dloss:0.754814]-[Gloss:64.970436, loss_pixel:0.640951, adv:0.875305] ETA:1 day, 16:29:16.095614
[Epoch99/200]-[Batch41/800]-[Dloss:0.043880]-[Gloss:29.431644, loss_pixel:0.268012, adv:2.630481] ETA:1 day, 2:02:26.442131184
[Epoch200/200]-[Batch799/800]-[Dloss:0.000665]-[Gloss:28.952560, loss_pixel:0.218009, adv:7.151631] ETA:0:00:00.967438400763304
```

4.5 测试集

建筑外立面语义设计图转成真实建筑图的测试数据集选择了 100 张建筑外立面语义设计图和 100 张真实建筑外立面图，该测试数据集来自于 CMP Facade 数据集并对此进行修改。

部分建筑外立面语义设计图训练数据集和部分建筑真实外立面图训练数据集见图 4.4 所示。

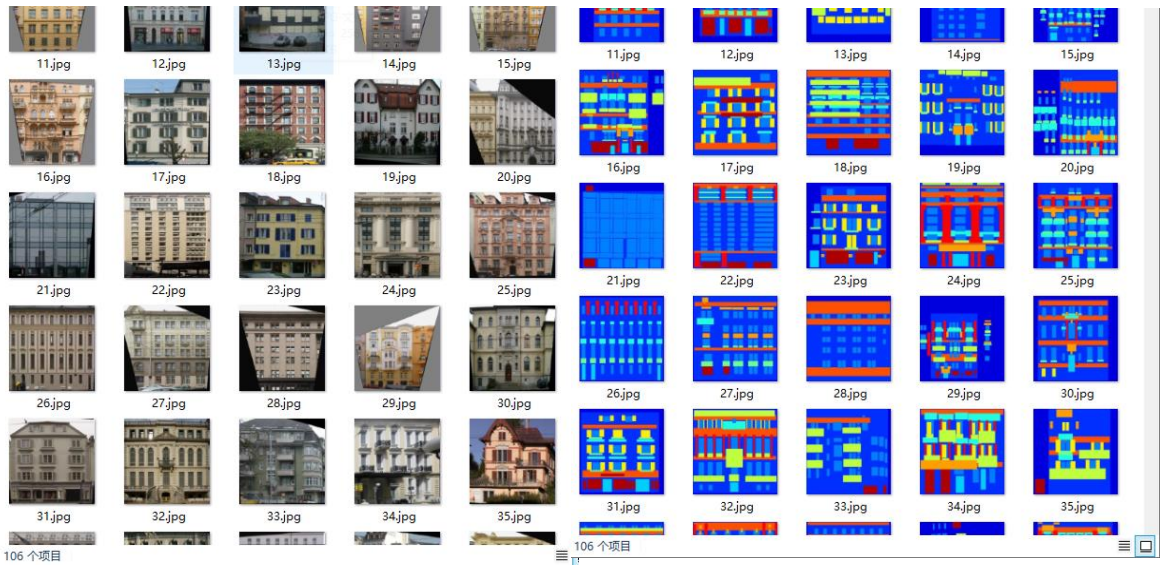

图 4.4 建筑外立面语义设计图和建筑真实外立面图部分测试集

4.6 实验实例

下列模型实例图中上方为输入的图像，中间为生成的图像，下方为真实的图像。

4.6.1 epoch 为 200 的模型

epoch 为 200 的模型的实验实例如图 4.5 所示。

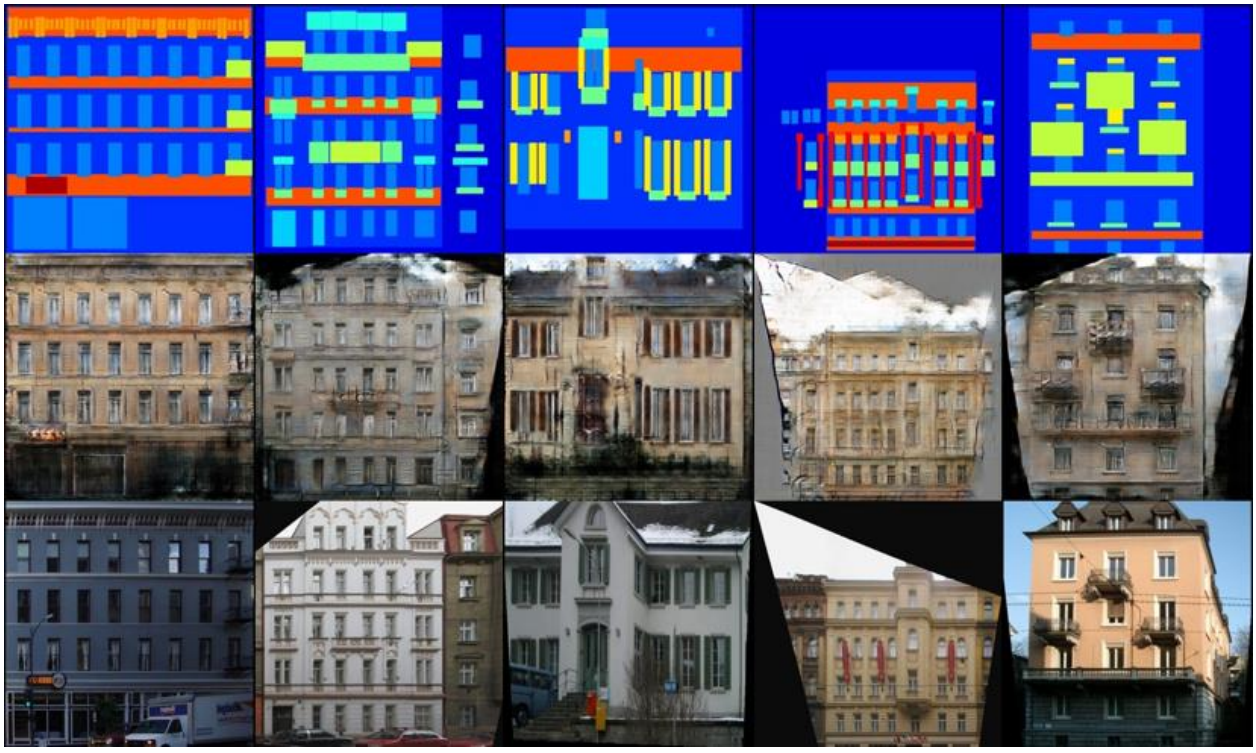

图 4.5(a)　epoch 为 200 的模型实验实例

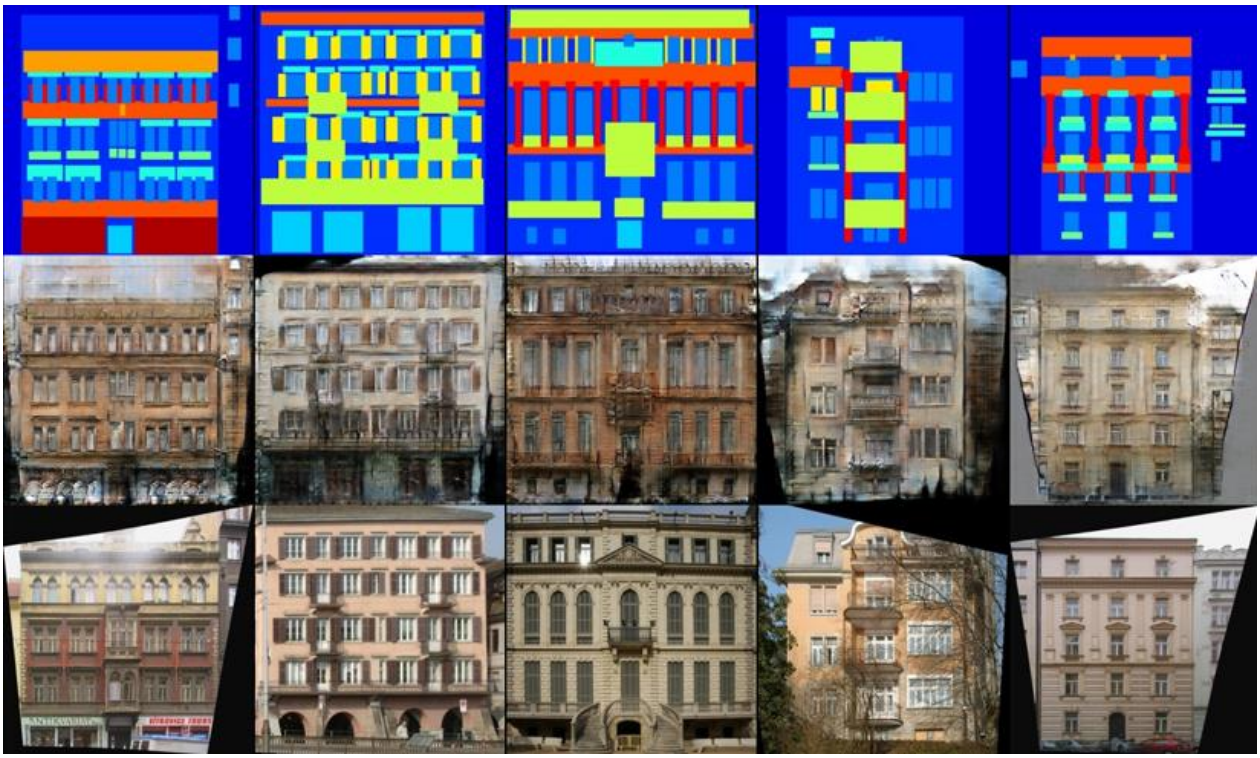

图 4.5(b)　epoch 为 200 的模型实验实例

4.6.2 epoch 为 100 的模型

epoch 为 200 的模型的实验实例如图 4.6 所示。

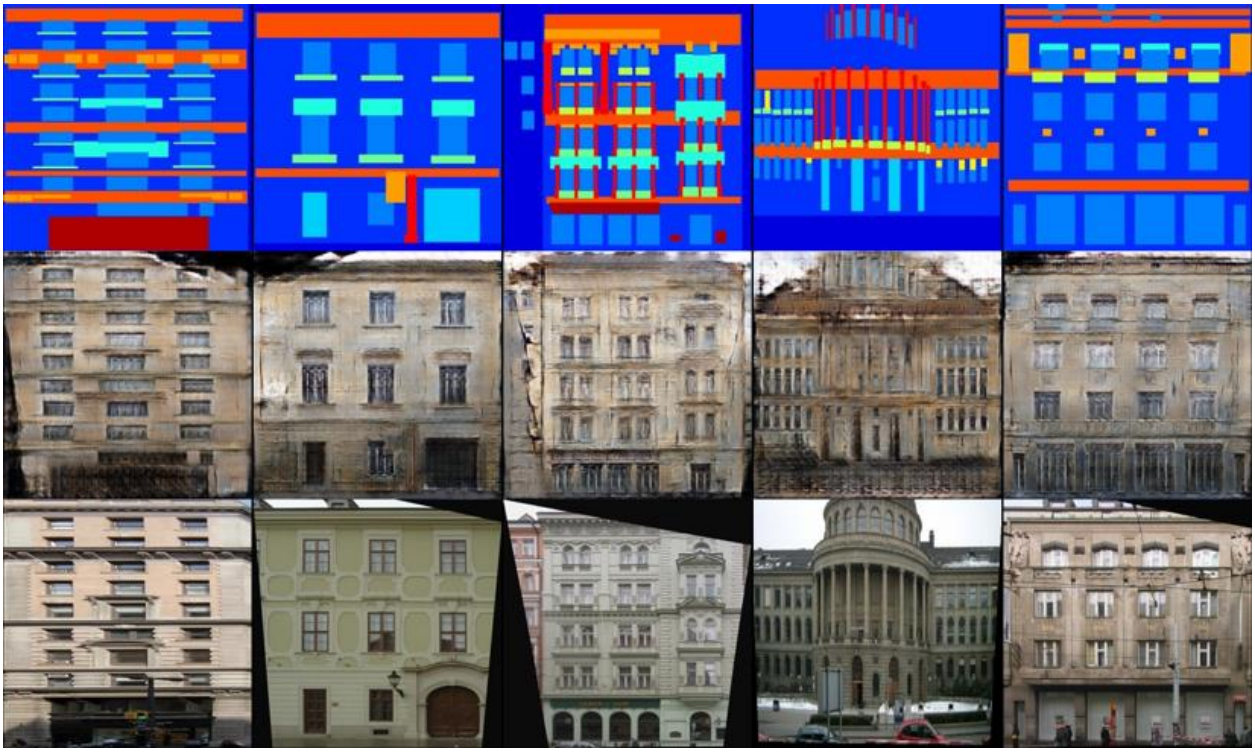

图 4.6(a)　epoch 为 100 的模型实验实例

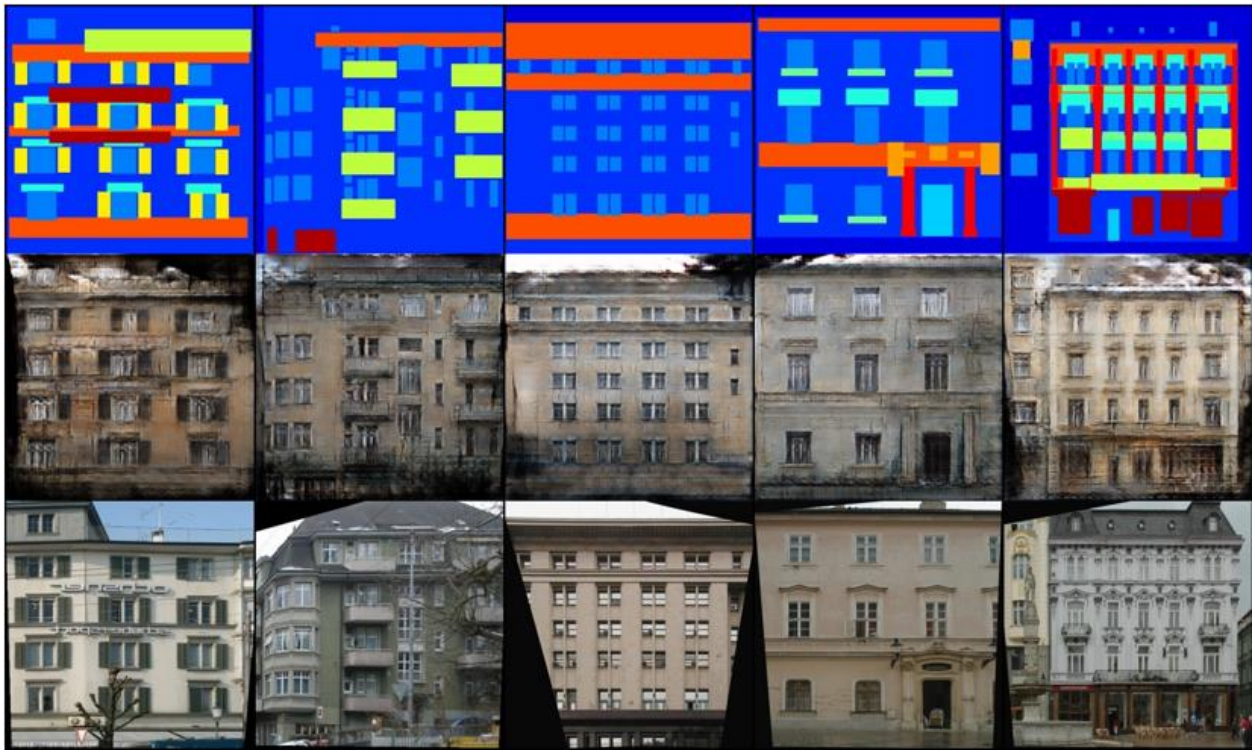

图 4.6(b)　epoch 为 100 的模型实验实例

4.7 训练结果评价

评估合成图像的质量是一个非常重要而困难的问题。传统的评价指标（如逐像素的做差平方求和算 L1 和 L2 距离的方法）实际上是不科学的，这种方法没有评估结构信息，只是从像素层面去计算，而不是从宏观的纹理特征和语义上去评估。

本文评价指标是用人工的方式去判断，寻找一些志愿者，进行人工打分。将网络模型生成图与真实图放一起，让志愿者判断哪些是网络模型生成图，哪些是真实的图像，如果网络模型生成图骗过了这些志愿者，让这些志愿者认为是生成的图像是真实的，就表明网络模型训练不错。本文选取五名志愿者来对模型生成的图像进行评价，选取生成的图像和真实的图像各 10 张，具体判断结果如图 4.7 所示。

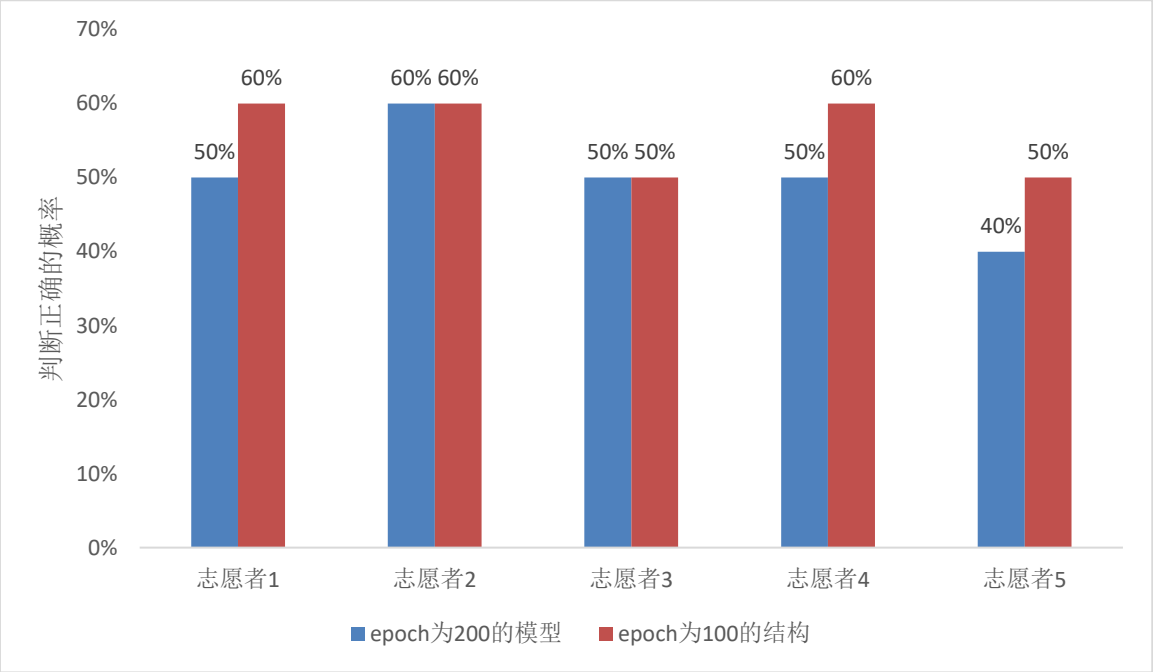

图 4.7 生成图片真假判断统计图

根据图 4.7 所示，epoch 为 100 的模型与 epoch 为 200 的模型生成的图像均有不错效果，但 epoch 为 200 的模型与 epoch 为 100 的模型比较，生成图像的表现较好。

第五章 总结和展望

随着深度学习和图像翻译技术的快速发展，利用生成对抗网络处理图像翻译问题已经在越来越多的领域实现应用。本文先对图像翻译研究现状和发展趋势进行介绍，再对基于卷积神经网络的图像翻译的应用和基于生成对抗网络的图像翻译应用的发展进行介绍，然后介绍基于卷积神经网络的图像翻译和基于生成对抗网络的图像翻译的基本原理以及它们所具有的缺点不足，最后介绍 CGAN 的基本原理和应用以及 CGAN 的其中一个变体 pix2pix 的模型。在方法和实验部分，选用 U-Net 和 PatchGAN 分别作为生成器和判别器，设计损失函数并对其训练方式进行优化，完成图像翻译任务。

本文的不足主要体现在这两个方面：其一，本文在图像生成方面的不足是 pix2pix 为解决图像翻译提供一种通用的解决办存在着生成图像不够清晰等问题且没有语义标签部分训练效果较差；其二，本文在评价指标方面的不足是不同志愿者会带来主观的评价对判断产生影响，并且人工打分无法自动化，如果有几亿张图片，在评价指标的任务就要耗费大量人力和时间，就无法选择找志愿者对这些图片依次进行打分这个方法。

条件生成对抗网络的一种变体应用 pix2pix 模型来实现图像翻译模型，该模型作为一种图像翻译问题的通用解决办法。这个模型不仅训练从输入图像到输出图像的映射，而且可以学习模型的损失函数来更好的训练模型。这样可以解决之前需要各种不同损失函数来训练模型的图像翻译任务的需要耗费大量人力去设计损失函数的问题。本文证明了该模型在建筑外立面设计图转成真实的图片是有效的。条件对抗网络是解决图像翻译任务的一种非常优秀的模型，尤其在那些涉及高度结构化的图像输出任务。条件生成对抗网络模型可以很好的学习适应所需要输出的任务和损失函数，所以改模型可以适应各种不同的应用环境，来实现图像翻译任务。

参考文献

[1] Ledig C, Theis L, Huszar F, et al. Photo-realistic single image super-resolution using a generative adversarial network [C]//Proceedings of the IEEE conference on computer vision and pattern recognition.2017:4681-4690.

[2] Wang X,Yu K,Wu S, et al. Esrgan: Enhanced super-resolution generative adversarial networks[C]//Proceedings of the European Conference on Computer Vision(ECCV).2018:211-252.

[3] Miyato T, Koyama M. cGANs with projection discriminator[J]. arXiv preprint arXiv:1802.05637,2018.

[4] Chen J,Chen J,Chao H, et al. Image blind denoising with generative adversarial network based noise modeling[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.2018:3155-3164.

[5] Kupyn O, Budzan V, Mykhailych M, et al. Deblurgan: Blind motion deblurring using conditional adversarial networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition.2018:8183-8192.

[6] Liu G, Reda F A, Shih K J, et al. Image inpainting for irregular holes using partial convolutions[C]//Proceedings of the European Conference on Computer Vision (ECCV).2018:85-100.

[7] Agustsson E, Tschannen M, Mentzer F, et al. Generative adversarial networks for extreme learned image compression[C]// Proceedings of the IEEE International Conference on Computer Vision.2019:221-231.

[8] Hertzmann A, Jacobs C E, Oliver N, et al. Image analogies[C]//Conference on Computer Graphics and Interactive Techniques. ACM, 2001:327-340.

[9] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]// IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2015:3431-3440.

[10] Zheng S, Jayasumana S，Romcra-Paredcs B, ct al. Conditional Random Ficlds as Recurrent Neural Networks[J]. 2015:1529-1537.

[11] Noh H, Hong s, Han B. Learning Deconvolution Network for Semantic Segmentation[C]// IEEE International Conference on Computer Vision. IEEE Computer Society, 2015:1520-1528.

[12] Pathak D, Krahen buhl P, Donahue J, et al. Context Encoders: Feature Learning by Inpainting[C]//IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2016:2536-2544.

[13] Zhang R, Isola P, Efros AA. Colorful Image Colorization[J]. 2016:649-666.

[14] 王万良，李卓蓉.生成式对抗网络研究进展[J].通信学报, 2018,39(2): 135-148.

[15] E.L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using Laplacian pyramid of adversarial networks[C]// International Conference on Neural Information Processing Systems. MIT Press, 2015:1486-1494.

[16] Odena A, Olah C, Shlens J. Conditional image synthesis with auxiliary classifier gans [C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 2642-2651.

[17] Miyato T, Koyama M. cGANs with projection discriminator[J]. arXiv preprint arXiv: 1802.05637,2018.

[18] Larsen A B L, Sonder by SK, Larochelle H, et al. Auto encoding beyond pixels using a learned similarity metric[J]. arXiv preprint arXiv:1512.09300,2015.

[19] Radford A， Metz L, Chintala S. Unsupervised representation learning with deep convolutional

generative adversarial networks[J]. arXiv preprint arXiv:1511.06434,2015.

[20] Mirza M, Osindero S. Conditional generative adversarial nets[J]. arXiv preprintarXiv:1411.1784,2014.

[21] Odena A, Olah C, Shlens J. Conditional image synthesis with auxiliary classifier gans [C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 2642-2651.

[22] Miyato T, Koyama M. cGANs with projection discriminator[J]. arXiv preprint arXiv: 1802.05637,201.

[23] ognition challenge[J]. International journal of computer vision, 2015,115(3): 211-252.

[24] Zhu J Y, Park T, Isola P, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks[C]//Proceedings of the IEEE international conference on computer vision.2017: 2223-2232.

[25] Yi Z, Zhang H, Tan P, et al. Dualgan: Unsupervised dual learning for image-to-image translation[C]//Proceedings of the IEEE international conference on computer vision.2017:2849-2857.

[26] Gatys L A, Ecker A S, Bethge M. A Neural Algorithm of Artistic Style[J]. Computer Science, 2015. 1486-1494.

致谢

行文至此，论文也开始了最后的收尾，就如我大学四年的生活一般，也进入了尾声。有的人的大学可能是丰富多彩的，充满着各种神奇而又美好的事情，但要说我四年的大学生活似乎都在新冠的陪伴下度过。从大一那年的疫情爆发，到大四疫情的放开，我似乎觉得自己见证了一段历史的诞生，虽然疫情放开了，但是我也要离开这座承载我四年回忆的地方了。在这我要感谢我的指导老师，张雄涛老师。他还是我很多主修课程的任课老师，不仅在学习上帮助我，还指导我完成了这次的毕业论文。无论是学习上的为人师表还是生活中的尽心尽力，张老师对我的人生指引我都深感于心。未来的日子里，希望张老师在自己热爱的事业上更进一步，桃李满天下。

万爱千恩百苦，感谢疼我爱我的父母。在外求学的安心，是因为父母在家里给我的后盾，感恩你们的付出，也希望你们以我为骄傲。

十载求学路仍未结束，希望自己在以后硕士研究生的学习阶段以更高的标准严格要求自己，抱着谦虚谨慎、求真求实的态度去钻研课题的，为国家、社会的发展贡献自己的力量。

附录

```
#模型训练的核心代码
#各模块引用
import argparse
import os
import numpy as np
import time
import datetime
import sys

import torch
from torch.autograd import Variable

from models import Create_nets
from datasets import Get_dataloader
from options import TrainOptions
from optimizer import *
from utils import sample_images , LambdaLR
#设置模型训练参数
args = TrainOptions().parse()
#计算判别器的图像输出结果
D_out_size = 256//(2**args.n_D_layers) - 2
#输出上述结果
print(D_out_size)
#计算 patch
patch = (1, D_out_size, D_out_size)

#初始化生成器和判别器
generator, discriminator = Create_nets(args)
#损失函数
criterion_GAN, criterion_pixelwise = Get_loss_func(args)
#优化器
optimizer_G, optimizer_D = Get_optimizers(args, generator, discriminator)

# 配置数据加载器
train_dataloader, test_dataloader,_ = Get_dataloader(args)
#训练
```

```
prev_time = time.time()
# Tensor = torch.cuda.FloatTensor if cuda else torch.FloatTensor

for epoch in range(args.epoch_start, args.epoch_num):
    for i, batch in enumerate(train_dataloader):

        #模型输入
        real_A = Variable(batch['A'].type(torch.FloatTensor))
        real_B = Variable(batch['B'].type(torch.FloatTensor))

        #生成器和判别器的对抗训练
        valid = Variable(torch.FloatTensor(np.ones((real_A.size(0), *patch))), requires_grad=False)
        fake = Variable(torch.FloatTensor(np.zeros((real_A.size(0), *patch))), requires_grad=False)

        #更新学习的损失函数
        #lr_scheduler_G.step(epoch)
        #lr_scheduler_D.step(epoch)

        # 训练生成器
        optimizer_G.zero_grad()

        #损失计算
        fake_B = generator(real_A)
        pred_fake = discriminator(fake_B, real_A)
        # print("pred_fake: ",pred_fake.size(),"valid: ", valid.size())
        loss_GAN = criterion_GAN(pred_fake, valid)
        # Pixel-wise 的损失计算
        loss_pixel = criterion_pixelwise(fake_B, real_B)

        #总体损失计算
        loss_G = loss_GAN + args.lambda_pixel * loss_pixel
        loss_G.backward()
        optimizer_G.step()

        #训练判别器
        optimizer_D.zero_grad()
        #真实图像的损失计算
        pred_real = discriminator(real_B, real_A)
```

```
        loss_real = criterion_GAN(pred_real, valid)

        #生成的假图像的损失计算
        pred_fake = discriminator(fake_B.detach(), real_A)
        loss_fake = criterion_GAN(pred_fake, fake)

        #总体损失计算
        loss_D = 0.5 * (loss_real + loss_fake)
        loss_D.backward()
        optimizer_D.step()

        #计算剩余训练时间
        batches_done = epoch * len(train_dataloader) + i
        batches_left = args.epoch_num * len(train_dataloader) - batches_done
        time_left = datetime.timedelta(seconds=batches_left * (time.time() - prev_time))
        prev_time = time.time()

        #输出具体训练的参数(Epoch、Batch、剩余时间等)
        sys.stdout.write("\r[Epoch%d/%d]-[Batch%d/%d]-[Dloss:%f]-[Gloss:%f, loss_pixel:%f, adv:%f]
ETA:%s" %
                                        (epoch+1, args.epoch_num,
                                        i, len(train_dataloader),
                                        loss_D.data.cpu(), loss_G.data.cpu(),
                                        loss_pixel.data.cpu(), loss_GAN.data.cpu(),
                                        time_left))

        #保存图像
        if batches_done % args.sample_interval == 0:
            sample_images(generator, test_dataloader, args, epoch, batches_done)

    if args.checkpoint_interval != -1 and epoch % args.checkpoint_interval == 0:
        #保存模型
        torch.save(generator.state_dict(), '%s/%s/generator_%d.pth' % (args.model_result_dir,args.dataset_name,
epoch))

        torch.save(discriminator.state_dict(), '%s/%s/discriminator_%d.pth' %
(args.model_result_dir,args.dataset_name, epoch))
```

```
#模型定义的核心代码
#U-Net 下采样
class UNetDown(nn.Module):
    def __init__(self, in_size, out_size, normalize=True, dropout=0.0):
        super(UNetDown, self).__init__()
        layers = [nn.Conv2d(in_size, out_size, 4, 2, 1, bias=False)]
        if normalize:
            layers.append(nn.InstanceNorm2d(out_size))
        layers.append(nn.LeakyReLU(0.2))
        if dropout:
            layers.append(nn.Dropout(dropout))
        self.model = nn.Sequential(*layers)

    def forward(self, x):
        return self.model(x)

#U-Net 上采样
class UNetUp(nn.Module):
    def __init__(self, in_size, out_size, dropout=0.0):
        super(UNetUp, self).__init__()
        layers = [  nn.ConvTranspose2d(in_size, out_size, 4, 2, 1, bias=False),
                    nn.InstanceNorm2d(out_size),
                    nn.ReLU(inplace=True)]
        if dropout:
            layers.append(nn.Dropout(dropout))

        self.model = nn.Sequential(*layers)

    def forward(self, x, skip_input):
        x = self.model(x)
        x = torch.cat((x, skip_input), 1)

        return x

#定义 U-Net 生成器
class GeneratorUNet(nn.Module):
    def __init__(self, in_channels=3, out_channels=3):
        super(GeneratorUNet, self).__init__()
```

```
        self.down1 = UNetDown(in_channels, 64, normalize=False)
        self.down2 = UNetDown(64, 128)
        self.down3 = UNetDown(128, 256)
        self.down4 = UNetDown(256, 512, dropout=0.5)
        self.down5 = UNetDown(512, 512, dropout=0.5)
        self.down6 = UNetDown(512, 512, dropout=0.5)
        self.down7 = UNetDown(512, 512, dropout=0.5)
        self.down8 = UNetDown(512, 512, normalize=False, dropout=0.5)

        self.up1 = UNetUp(512, 512, dropout=0.5)
        self.up2 = UNetUp(1024, 512, dropout=0.5)
        self.up3 = UNetUp(1024, 512, dropout=0.5)
        self.up4 = UNetUp(1024, 512, dropout=0.5)
        self.up5 = UNetUp(1024, 256)
        self.up6 = UNetUp(512, 128)
        self.up7 = UNetUp(256, 64)
        self.up8 = nn.Sequential(
            nn.ConvTranspose2d(128, out_channels, 4, 2, 1),
            nn.Tanh()
            )

def forward(self, x):
    # U-Net 生成器，具有从编码器到解码器的跳过连接
    d1 = self.down1(x)
    d2 = self.down2(d1)
    d3 = self.down3(d2)
    d4 = self.down4(d3)
    d5 = self.down5(d4)
    d6 = self.down6(d5)
    d7 = self.down7(d6)
    d8 = self.down8(d7)
    u1 = self.up1(d8, d7)
    u2 = self.up2(u1, d6)
    u3 = self.up3(u2, d5)
    u4 = self.up4(u3, d4)
    u5 = self.up5(u4, d3)
    u6 = self.up6(u5, d2)
    u7 = self.up7(u6, d1)
```

```
    return self.up8(u7)
#判别器
class Discriminator_n_layers(nn.Module):
    def __init__(self, args ):
        super(Discriminator_n_layers, self).__init__()

        n_layers = args.n_D_layers
        in_channels = args.out_channels
        def discriminator_block(in_filters, out_filters, k=4, s=2, p=1, norm=True, sigmoid=False):
            """返回每个判别器块的下采样层"""
            layers = [nn.Conv2d(in_filters, out_filters, kernel_size=k, stride=s, padding=p)]
            if norm:
                layers.append(nn.BatchNorm2d(out_filters))
            layers.append(nn.LeakyReLU(0.2, inplace=True))
            if sigmoid:
                layers.append(nn.Sigmoid())
                print('use sigmoid')
            return layers

        sequence = [*discriminator_block(in_channels*2, 64, norm=False)] # (1,64,128,128)

        assert n_layers<=5

        if (n_layers == 1):
            '当 n_layers 为 1 时, patch 大小为 (16x16)'
            out_filters = 64* 2**(n_layers-1)

        elif (1 < n_layers & n_layers<= 4):
            '''
            当 n_layers 为 2 时, patch 大小为 (34x34)
            当 n_layers 为 3 时, patch 大小为(70x70), this is the size used in the paper
            当 n_layers 为 4 时, tpatch 大小为(142x142)
            '''
            for k in range(1,n_layers): # k=1,2,3
                sequence += [*discriminator_block(2**(5+k), 2**(6+k))]
            out_filters = 64* 2**(n_layers-1)
```

```
        elif (n_layers == 5):
            '''
            当 n_layers 为 5 时, patch_size 为 (286x286), 将会大于图像的大小(256),
            故满足所有图像条件
            '''
            for k in range(1,4): # k=1,2,3
                sequence += [*discriminator_block(2**(5+k), 2**(6+k))]
                # k=4
            sequence += [*discriminator_block(2**9, 2**9)] #
            out_filters = 2**9
        num_of_filter = min(2*out_filters, 2**9)

        sequence += [*discriminator_block(out_filters, num_of_filter, k=4, s=1, p=1)]
        sequence += [*discriminator_block(num_of_filter, 1, k=4, s=1, p=1, norm=False, sigmoid=True)]
        self.model = nn.Sequential(*sequence)

    def forward(self, img_A, img_B):
        # 按通道连接图像和条件图像来输入
        img_input = torch.cat((img_A, img_B), 1)
        #print("self.model(img_input):  ",self.model(img_input).size())
        return self.model(img_input)
#初始化生成器和判别器
def Create_nets(args):
    generator = GeneratorUNet()
    discriminator = Discriminator_n_layers(args)
    if args.epoch_start != 0:
        # 加载预训练模型
        generator.load_state_dict(torch.load('saved_models/%s/generator_%d.pth' % (args.dataset_name,
args.epoch)))
        discriminator.load_state_dict(torch.load('saved_models/%s/discriminator_%d.pth' % (args.dataset_name,
args.epoch)))
    else:
        # 初始化参数权重
        generator.apply(weights_init_normal)
        discriminator.apply(weights_init_normal)
        print_network(generator)
        print_network(discriminator)
    return generator, discriminators
```