****

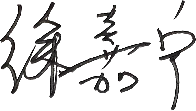
**本科毕业设计**

**（2022届）**

|  |  |
| --- | --- |
| **题 目** | 基于对抗式生成网络的角色形象生成 |
| **学 院** | 人文艺术与数字媒体学院 |
| **专 业** | 数字媒体技术 |
| **班 级** | 18222513 |
| **学 号** | 18221787 |
| **学生姓名** | 徐嘉宁 |
| **指导教师** | 王毅刚 |
| **完成日期** | 2022年6月 |

**诚 信 承 诺**

我谨在此承诺：本人所写的毕业论文《基于对抗式生成网络的角色形象生成》均系本人独立完成，没有抄袭行为，凡涉及其他作者的观点和材料，均作了注释，若有不实，后果由本人承担。

 **承诺人（签名）：**

**2022年6月1日**

摘 要

生成式对抗网络（GAN）技术被提出以来，其在图像生成、图像修复等方面被广泛应用。在卷积神经网络（CNN）技术被引入生成式对抗网络之后，深度卷积生成式对抗网络模型（DCGAN）被发明，它有效提升了非监督学习的能力，并且成功减小了监督式学习与非监督学习两个领域之间技术成熟的差距。然而，GAN自身作为比较早期的技术，在图像生成方面仍然存在许多不足，因此人们不断地探索、改进，基于GAN发明了CycleGAN、BigGAN、StyleGAN等多种GANs，旨在提升网络的效果，或者扩展原有的功能。

本论文的主要工作：1.以动漫的角色形象为对象，探究GAN在图像生成上的原理方法和缺陷，并通过改进图像数据集的质量、调整生成模型和判别模型及其参数来提升生成图片的质量；2.使用风格生成式对抗网络（StyleGAN）进行同样的工作，对比、总结新老技术在同一个应用中的区别和优劣；3.为了能够直观展示GAN生成动漫形象的过程，对真假图片的绘制进行实时的监视，本文提出了一种以Visdom为框架的可视化界面；同时为了能够展示GAN和StyleGAN训练模型的最终效果，本文设计了一种以Flask为框架的可视化界面。

最终我们可以在可视化界面中一键生成角色形象图片，以直观查看模型效果。

关键词：生成式对抗网络；动漫形象；可视化；DCGAN；StyleGAN

**ABSTRACT**

Since the generative adversarial network (GAN) technology was proposed, it has been widely used in image generation, image inpainting, etc. Especially, after the convolutional neural network (CNN) technology was introduced into GAN, the deep convolutional generative adversarial network (DCGAN) was invented, which effectively improved the ability of unsupervised learning and successfully reduced the gap of maturity between supervised learning and unsupervised learning. However, as an early technology, GAN still has many shortcomings in image generation. Therefore, in the process of continuous exploration and improvement, various GANs such as CycleGAN, BigGAN, and StyleGAN have been invented based on GAN aimed to improve the effect and create extensions.

The main work of this paper is as follows: 1. Taking anime image as the object, explore the principles, methods and defects of GAN in image generation, and improve the quality of generated images by improving the quality of image datasets and adjusting the models and parameters of generators and discriminators. 2. Use StyleGAN to do the same work, comparing and summarizing the differences, advantages and disadvantages of new and old technologies in the same application. 3. In order to visually display the process of GAN generating animation images and monitor real and fake pictures in real time, this paper proposes a visual interface in Visdom framework; at the same time, in order to show the final effect of the model after GAN and StyleGAN training, this paper designs a visual interface in Flask framework.

Finally, we can generate a character image with one click in the visualization interface to visually check the model effect.

**Keywords：**generative adversarial network；anime image；visualization；DCGAN；StyleGAN

目 录

[1. 引言 1](#_Toc105417558)

[2. 综述 3](#_Toc105417559)

[2.1 生成式对抗网络 3](#_Toc105417560)

[2.2 深层卷积生成式对抗网络 4](#_Toc105417561)

[2.3 风格生成式对抗网络 6](#_Toc105417562)

[3. 研究内容 8](#_Toc105417563)

[3.1 深度学习框架 8](#_Toc105417564)

[3.2 DCGAN模型 8](#_Toc105417565)

[3.3 StyleGAN3模型 15](#_Toc105417566)

[4. 基于对抗式生成网络的角色形象生成项目实现 17](#_Toc105417567)

[4.1 项目综述 17](#_Toc105417568)

[4.2 文件结构 18](#_Toc105417569)

[4.3 可视化 20](#_Toc105417570)

[4.4 生成效果 22](#_Toc105417571)

[4.5 优化方案 27](#_Toc105417572)

[5. 总结和讨论 32](#_Toc105417573)

[5.1 优劣分析 32](#_Toc105417574)

[5.2 区别度问题 32](#_Toc105417575)

[6. 结语 34](#_Toc105417576)

[7. 致谢 36](#_Toc105417577)

[8. 参考文献 37](#_Toc105417578)

[9. 附录 38](#_Toc105417579)

# 引言

2017年5月，由谷歌团队开发的围棋机器人alphaGo以完胜的姿态击败了人类排名第一的围棋选手柯洁，这预示机器人在围棋领域已经全面超过人类。自此，“深度学习”一词逐渐走进人们的视野之中。从21世纪10年代到20年代，机器人渐渐地取代了人类生活中的方方面面，深度学习在各个领域中都被广泛应用。甚至，在象征着人类感性精神的艺术世界都可以见到它的身影。

2014年6月，Ian J. Goodfellow等人发表了生成式对抗网络（generative adversarial network，以下简称“GAN”）的开山之作，其核心在于利用一对神经网络的相互博弈，从而使两者相互训练。GAN一经问世，就被广泛应用于图像生成、超分辨率任务、图像风格迁移等热门方向，其中许多应用都和艺术有很大的关系。例如2020年11月，来自美国普林斯顿大学的学生Alice Xue在她的毕业论文中开发了一款名为SAPGAN（Sketch-And-Paint GAN）的模型，该模型的功能是智能地生成中国山水画，其效果优秀，普通人根本无法分辨生成的图片是大师作品还是机器制造。

因此可以说，GAN为普通人接触艺术领域降低了门槛，这一点同样体现在“利用GAN生成二次元动漫头像”这一经典应用场景中。2016年7月，Alec Radford和Luke Metz等人将卷积神经网络（convolutional neural network，以下简称“CNN”）引入GAN中，发明了深度卷积生成对抗网络（deep convolutional generative adversarial networks，以下简称为“DCGAN”），DCGAN凭借其突破性的生成效果，成为了后来各种GANs的主要参考对象。该团队首次在论文中利用DCGAN对动漫头像的生成展开了相关工作。直到最近，不仅仅局限于头像生成，Crypko网站已经可以完美地生成各种拥有华丽服饰、协调身体的动漫角色形象（带身体和服饰的图片需付费）。如图 1‑1所示。

对于普通人来说，我们在社交网络平台上喜欢使用动漫人物作为头像，并且有时候希望自定义属于自己的形象（avatar），然而，这需要丰富的绘画技巧。GAN则为这些没有专业技能的普通人创造了定制形象的机会；对于动漫、游戏的创作者和从业者来说，他们可以利用自动生成的角色获得灵感，进行二次设计；角色扮演游戏开发者则可以使用没有版权的形象来减少游戏制作中的设计成本。



图 1‑1 Crypko网站生成的角色形象[[1]](#footnote-1)

因此，为了探究GAN之于动漫头像生成这一经典场景的作用，本论文主要致力于分析GAN的原理、解析DCGAN神经网络的架构、优化图像生成的方案等问题。在过程中为达到改进目的，作者主要研究了2021年发表自英伟达团队的新技术StyleGAN3，将其与DCGAN技术作对比，思考了如何改进、为什么改进、旧之劣点新之优点等问题，使用了更为干净的数据集以及更加适宜的超参数，为训练模型服务；为达到展示目的，主要设计了以Visdom为框架的训练过程展示界面和基于Flask的训练结果展示界面。

# 综述

## 生成式对抗网络

### 生成式对抗网络综述

自深度学习概念风靡以来，人们希望能够利用它解决机器学习处理中较为困难的非结构化数据问题，或称为感知类数据，例如图片处理、音频处理、办公文档、网页文本等。在GAN未被发明之前，深度学习中最为成功的模型基本都涉及到判别模型，通常的做法是，将高维数据等感觉类数据输入到类标签中。判别模型的成功在很大程度上取决于成熟的反向传播和dropout算法，反向传播算法很好地解决了参数优化的问题，而dropout算法有效地解决了模型过拟合的问题。然而，传统的深度学习在最大似然估计中出现的概率问题难以准确计算，且在生成模型中难以利用分段线性激活函数，导致优秀的生成模型数量较少。而GAN提出了一个新的深度学习框架，该框架可以为多种模型和优化算法生成特定的训练算法，区别于以往的神经网络，它由生成网络和判别网络两个神经网络构成，两者都是多层感知机。在这种情况下，生成模型通过正向传播从生成模型中采样，判别模型只使用反向传播和dropout算法，有效地规避了复杂的马尔科夫链。

### 生成式对抗网络原理

GAN的原理简单易懂，在Goodfellow等人发表的论文中有形象的解释。这里可以使用一个制作假钞的比喻例进行说明。

GAN由生成模型G和判别模型D两个神经网络构成，其最终目标是使生成模型G生成一张优质的假钞图片。

首先，生成模型G会随机生成一张假钞图片。这张图片可能由一串随机的特征向量控制。在输入随机的特征向量之后，G按照特征向量随机生成一张图片。其次，判别模型D会判断G生成的假钞是否足够“真”。因此，D的输入是G生成的图片，输出为一个标量值，通常它被限制在0到1的范围中，越小则表示假钞越“假”；越大则表示假钞越“真”。判断标准来自于外部提供的真钞图片的数据集，即对于D来说，这些数据集中的真钞的评价值皆为1，而G生成的假钞皆为0。最后，G为做到欺骗D，D为做出正确的评价，两者展开博弈，直到G生成的假钞让D无法分辨真假。

将上述例子中的过程抽象描述，则我们可以得到GAN的原理图。如图所示。

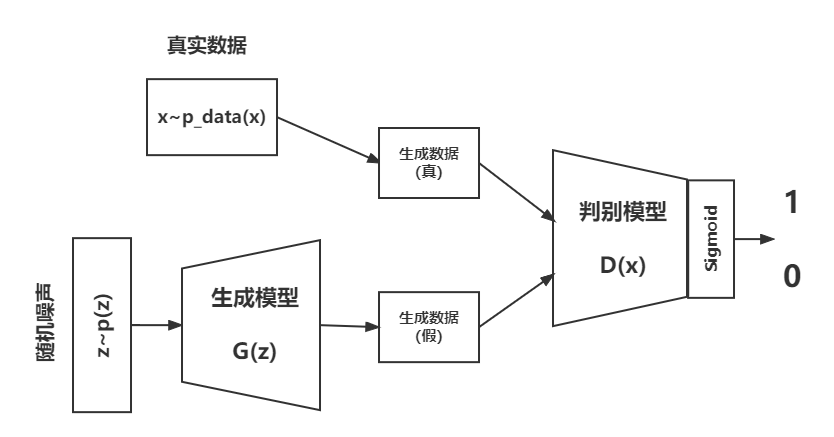


图 2‑1 GAN原理图

生成模型的输入为随机噪声，尝试输出数据样本。生成模型G从随机噪声中获得z,其中z是来自概率分布的样本。生成模型产生一个数据送入判别模型D中。同时，是真实的数据分布，经过处理，判别模型D也接收到来自真实数据的x，于是判别模型需要预测输入时真实的还是生成的。经过判别模型后的输出值为一个标量，使用Sigmoid激活函数解决一个二元分类问题，最终输出值的范围为0~1。

实际上该问题的求解过程可以看作一个极小极大值求解问题，GAN模型的需求是找到两个模型的最优参数，使得参数可满足下方的对抗损失函数：

等式右侧第一项为来自真实样本的数据，判别模型希望接收到来自真实样本数据时输出为1，即此时判别模型要求预测.

等式右侧第二项为来自生成模型输出的样本数据，判别模型希望接收这份数据时输出为0，即此时判别模型要求预测.

因此对于判别模型D来说，对抗损失函数的最大值.

然而等式左侧表示了其对抗属性，生成模型希望，即，因此对于生成模型G来说，对抗损失函数的最小值，在训练模型参数时，参数需要不断来回满足这两个条件，因此两个模型能够相互竞争相互训练。

## 深层卷积生成式对抗网络

### 深层卷积生成式对抗网络综述

深层卷积生成式对抗网络（deep convolutional generative adversarial network，以下简称“DCGAN”）是一种基于GAN的变体模型。可以说，它是最为成功的一个GAN类框架，对图像的处理有比较好的效果，影响了后续诸多其他GAN的变体。

DCGAN将卷积神经网络引入GAN模型中，将判别模型用于图像分类任务，显示出相比其他无监督算法的竞争性能。Alec等人可视化了通过GAN学习的滤波器，并且证明了不同的滤波器可以用于绘制不同的图片对象。加入CNN后的生成模型具有有趣的向量运算特性，这些特性可以让我们轻松地计算生成样本的语义特征。

图像生成模型通常分为两种：有参数型和无参数型。有参数型的模型通常从现有图像的数据库中进行匹配，常用于纹理合成、超分辨率和绘画任务。无参数型模型虽然也被应用在纹理合成中，例如MNIST数据集合纹理合成，然而真实世界的自然风景图像合成仍然比较困难，因为真实世界图片的采样常常受到模糊（Blur）的影响。另一种生成图像的方法是迭代前向扩散过程，但由于在GAN中，生成模型接收的输入常常需要加入大量噪声，这可能会造成该方法的不稳定。

加入CNN之后，DCGAN可以在一定程度上缓解上述的几个问题。

### 深层卷积生成式对抗网络原理

DCGAN的核心是采用且修改了三个对GAN架构的改进。

第一个是全卷积网络，使用跨步卷积代替了确定性空间池化函数，允许网络学习自身的空间下采样。将全卷积网络应用在生成模型和判别模型中，允许它们学习自己的空间上采样。

第二个是在倾向于在最顶层的卷积后消除全连接层。全局平均池化虽然提高了模型的稳定性，但同时降低了模型的收敛速度。如果在最顶层的卷积特征的中间区域直接连接生成模型的输出和判别模型的输入，效果也比较好。

GAN的第一层以均匀的噪声分布z作为输入，可以称其为全连接，因为它只是一个矩阵乘法，最后的结果是将其变形成4D张量并用作卷积堆栈的开始。对于判别器，最后一个卷积层被展平，叠加单个的Sigmoid输出。Alec等人的论文中，对于生成模型的结构如图 2‑2所示。

第三个是批量标准化（Batch Normalization），它通过将每个单元的输入归一化为零均值和单位方差来稳定学习。这有助于处理由于初始化不良而出现的训练问题。然而直接将批量归一化应用到所有层会导致样本振荡和模型不稳定。如果生成模型的输出层和判别模型的输入层不应用批量归一化，那么可以有效避免上述问题的出现。使用有界限的激活函数可以使模型更快地学习达到饱和，并且覆盖训练分布的颜色空间。

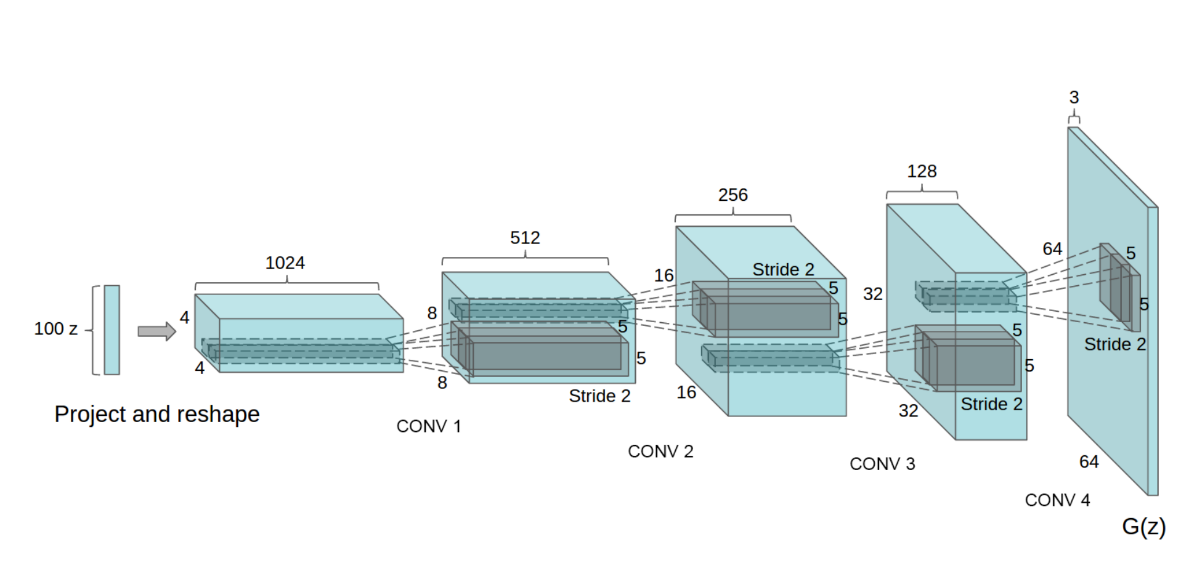


图 2‑2 DCGAN生成模型结构图

## 风格生成式对抗网络

风格生成式对抗网络（style generative adversarial network，以下简称为“StyleGAN”），由英伟达公司创立项目团队展开研究。StyleGAN经过了三次技术变革，现已经达到第三世代，即StyleGAN3。本论文的另一实验基于正是基于该技术，因为StyleGAN3拥有比DCGAN更加优秀的效果，以及更加稳定的模型训练过程，尤其是对于动漫图像等风格强烈的图片来说。

典型的生成对抗网络具有分层卷积性质，生成图片的细节处的像素会以一种不健康的方式依赖于绝对像素坐标，人眼在观察时会感觉细节处似乎粘连在屏幕内侧，而不是图像中物体的表面上。例如，图片中人物的发丝常常出现平移、涂抹、模糊等现象。这是因为，在现实世界中，不同尺度范围上的细节往往是层次化的。例如，转动头部，颈部的形状会改变；提高鼻梁，周围的毛孔同样也会提高……典型GAN的生成模型采用的也是这种仿真结构，在训练网络的过程中，粗糙的、低分辨率的特征控制着细节的、高分辨率的特征，通过上采样层逐渐被细化，通过卷积实现各个特征的混合，实际上就是不断添加噪声模拟，对细节不断的完善和细化。

事实表明，目前的网络结构可以通过使用图像边界参考、像素的噪声输入、位置编码和混叠等手段来绕过模拟现实的理想的层次结构，从而打破连贯物体在空间中移动的错觉，使得其中每个特征的确切像素位置完全继承自底层的粗糙特征。

同时，基于一维卷积的StyleGAN3模型经过修理后，使得卷积核拥有径向对称的性质，最终卷积操作获得平移和旋转等变性，且FID[[2]](#footnote-2)和StyleGAN2没有很大差距。

由于StyleGAN3的原理较为复杂，在实验中仅用作GAN的对比项，并无涉及到优化或者更深层的应用，因此论文中对于其原理就不多赘述。本文主要以DCGAN框架为主要研究对象。

# 研究内容

## 深度学习框架

现在主流的深度学习框架主要有三个，分别是Tensorflow、Pytorch和Keras。本论文中实验所用到的所有GAN模型都基于Pytorch框架。Pytorch对于电脑运行环境、依赖库版本、GPU版本都有比较高的要求，如果版本匹配有问题，可能不能正常运行。参考环境配置可见表格 3‑1。

表格 3‑1 pytorch环境配置表

|  |  |
| --- | --- |
| 依赖库名称 | 版本号 |
| torch | 1.8.2+cu111 |
| torchvision | 0.9.2 |
| torchnet | 0.0.5.1 |

选择Pytorch为深度学习框架主要有以下两个原因：

1. StyleGAN3的官方开源代码基于Pytorch，虽然在本地同样可以使用Tensorflow或者Keras复现训练过程，但是StyleGAN3的原理比较复杂，使用原生框架可以避免兼容性、参数误差等问题，并且可以更加方便的学习代码。
2. Pytorch的优势在于定义网络结构简单，语句清晰直观，层与层间构成简单易懂。Pytorch更适合研究人员或者爱好者快速进行开发原型工作。

由于深度学习模型的训练环境涉及到cuda核心版本和一部分依赖项版本，中间可能会遇到各种性能和兼容性问题，因此，在开源的Linux系统环境下训练模型会更加稳定。本论文的模型都是选择在Linux中训练完成的。

## DCGAN模型

### 模型构成

DCGAN的模型结构如图 3‑1，分为生成模型和判别模型两部分。其中的“ConvTr2d”代表的是逆卷积层，“Conv2d”代表的是卷积层，即上文提到的DCGAN新引入的卷积神经网络。“BN”表示批量标准化层（Batch normalization），“ReLU”与“LeakyReLU”都是隐层间的激活函数，“Tanh”和“Sigmoid”为输出结果层的激活函数。在生成模型和判别模型中，“k”表示卷积核大小，“n”代表特征图个数，“s”代表步长。特征图是卷积核进行卷积操作过程中，卷积核每划过一块区域就生成一个特征图。

在生成图像的时候，我们需要不断扩大图像的尺寸，逆卷积操作（ConvTransport2d）就是一个可行的方法；相反，对于判别模型来说，判别模型最终需要将获得的输入值，即图像信息转换成一个具有评价功能的标量，因此我们可以使用卷积操作（Conv2d）将图像不断缩小。

观察图中两个神经网络的结构，实际上生成模型和判别模型两者互为逆过程。

关于BN层和各个激活函数的细节会在下几节具体分析。

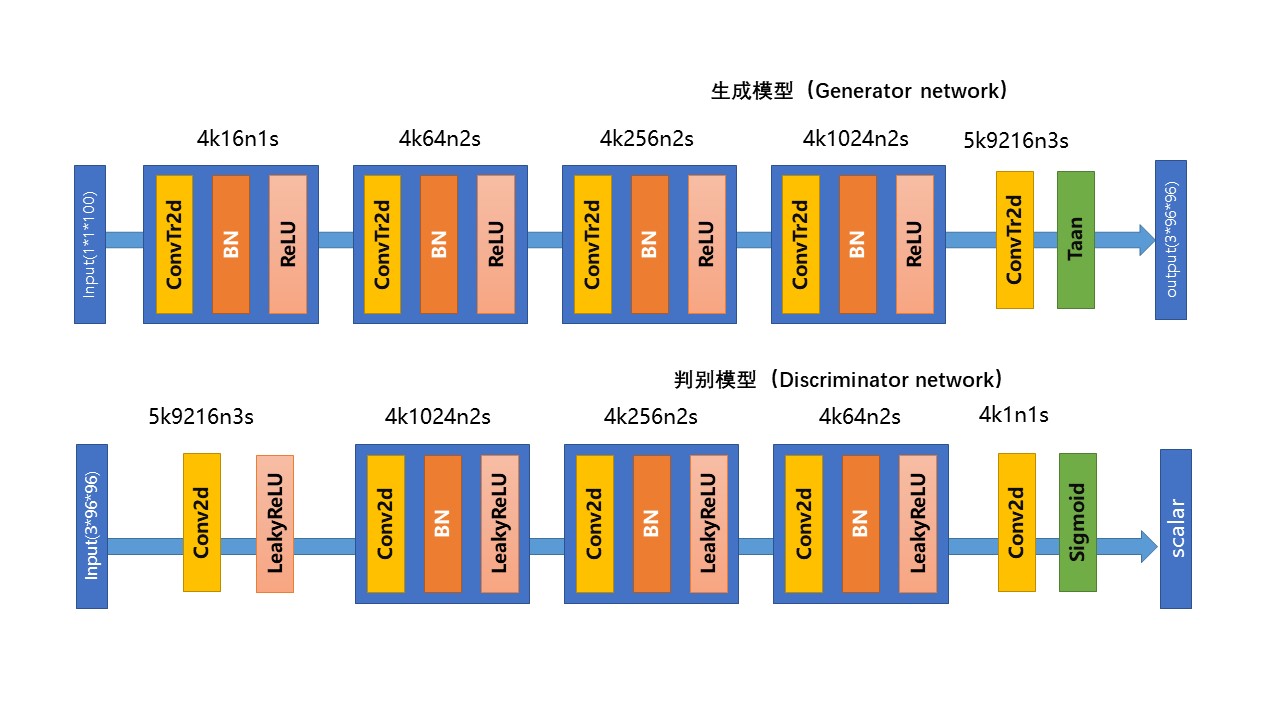


图 3‑1 DCGAN模型结构图

### 批量标准化层

根据Sergey Ioffe等人的论文，BN计算步骤为：

1. 计算样本均值；
2. 计算样本方差；
3. 样本数据标准化处理；
4. 进行平移和缩放。过程中引入了和两个参数。通过训练这两个参数可以让网络学习恢复出原始网络要学习的特征分布。

计算过程如图 3‑2所示。

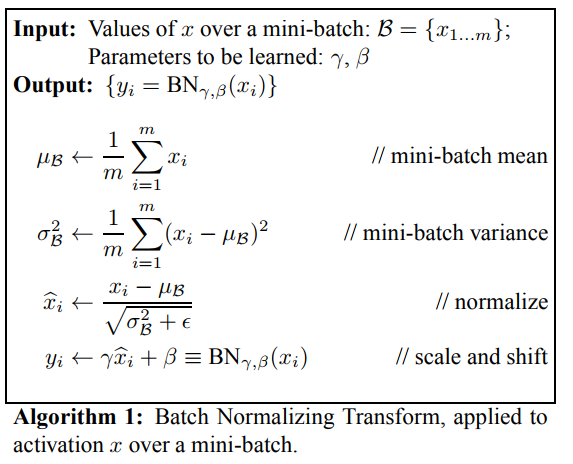


图 3‑2 BN层计算过程[[3]](#footnote-3)

为便于理解，这里列举一个普通归一化的例子。简单的网络如图 3‑3所示：

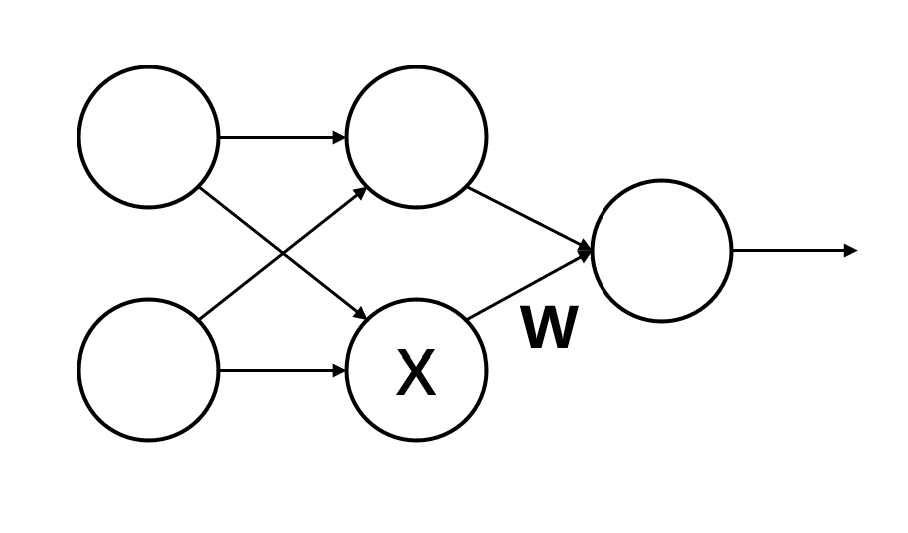


图 3‑3 简单网络例图

网络中存在某个神经元，该神经元对应权重值为，。现先后取值如下，；则先后计算得；此时加上一层激活函数，我们使用双曲正切函数（以下简称“tanh”）作为激活函数，将代入函数求得函数值，先后为.

观察图 3‑4不难发现，tanh函数图像在两侧后坡度明显变缓，当x轴趋向无穷时，函数的导数趋向于0，这种情况被称之为“饱和”。也就是说，在经过激活函数处理后，若值本就处在激活函数饱和范围内，无论怎么增大，激活函数的输出值始终接近1，没有显著性区别，这对于判断样本对模型的影响是相当不利的，在很大程度上会导致权重无法科学地更新。于是，我们需要使用标准化（normalization）函数进行处理。然而这种情况同样会在网络隐层中出现，类似的，我们需要用一种标准化手段处理隐层。批量标准化的做法是，在每一层网络输入下一层前，插入一个标准化网络层，在每一批参数前向传递的时候进行标准化处理。

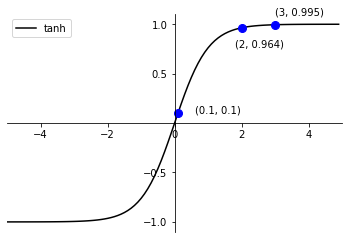


图 3‑4 tanh实例图

BN层的具体作用指向三个方向：

1.加速网络收敛。如果网络中每一层的数据分布不一致，将会导致训练无比困难而且效率变低。均值为0方差为1的标准化工程可以使网络每一层同质化收敛。

2.改善梯度弥散。由于层间神经元输出值受激活函数的控制——激活或者抑制，其原本值的范围可能受到一定程度上的规范，这可能会造成该神经元的影响力度远远小于或大于原本该有的影响力度，导致梯度消失或者梯度爆炸。

3.防止过拟合。在网络训练中，数据被标准化的时候是一小批一小批绑定在一起，共同接收处理的，即样本是被关联在一起的，因此模型不会根据某一个特定的样本训练出一个特定的值，而是综合了属于同一批的样本集合完成训练，这一定程度上避免了模型过拟合。

### 激活函数

激活函数的作用是为神经网络引入非线性因素，非线性因素可以增强神经网络的分类表达能力。深度学习的本质是一个复杂分类问题，例如利用深度学习区分猫和狗的照片。此问题可以类比在一个二维直角坐标系中，用线划分区域的问题。划分区域有两种手段：线性手段和非线性手段。前者的意思是只能用直线，后者的意思是可以用曲线或者分段线，因此可以想象，曲线的划分区域能力显然要比单纯的直线更加细腻和优秀。如图 3‑5所示。一个单纯的多层感知机无论输入怎么改变，结构多么复杂，最终输出都可以用多项线性表达式表达，而添加激活函数层之后就无法做到了；就如同要在平面上取一个圆形，用无数段小直线可以拟合逼近，效果却始终没有曲线好。

|  |  |
| --- | --- |
|  |  |
| (a) | | (b) |

图 3‑5 划分区域图

(a)线性划分图 (b)非线性划分图

DCGAN网络中的激活函数主要分为两大部分：输出前激活函数和隐层间激活函数。图 3‑6表示模型中所用到的所有具体的激活函数。

判别模型的输出前激活函数为Sigmoid函数，是神经网络中非常常用的激活函数，因为其模式在物理意义上最接近生物神经元。表达式如下：

Sigmoid函数值域为0~1，正好我们所需的判别标量的输出范围也是0~1.考虑到这一点，DCGAN模型选择在判别模型中使用该激活函数。然而Sigmoid函数具有饱和性的缺点，如同上一节提到的，当输入值，即图中的x过大或者过小时，Sigmoid函数的导数趋向0，梯度变化在饱和范围内变换缓慢，这可能会使参数无法更新，向下一层传递的梯度值过小，导致梯度消失。

生成模型的输出前激活函数为tanh函数，其表达式如下：

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |

图 3‑6 DCGAN中的激活函数

(a)Sigmoid函数 (b)tanh函数 (c)ReLU函数 (d)LeakyReLU函数

Tanh函数的值域为-1~1。生成模型的输出是图片，像素范围为0~255，因此只要对tanh函数的输出加1再乘以127.5即可得到相同范围。其作用与Sigmoid函数类似，而且和Sigmoid函数一样具有饱和性的缺点。但是tanh函数均值为0，在DCGAN论文中提到，tanh函数相比Sigmoid函数，可以让GAN的收敛更加快速、更加稳定，生成图片的质量更高。

然而，深度学习网络隐层深、神经元个数多、层间的数据范围不稳定，难免出现输入值过小或者过大的情况。如果仍然使用Sigmoid函数或者tanh函数，隐层无法应对过饱和的问题，网络一旦达到饱和状态，学习算法就不得不不断地调整权重参数来强行提升模型的性能，此外，输入值可能对激活函数中段部分尤其敏感，这样两极的情况在处理大批量数据时无法占据优势。其次，在涉及到梯度的反向传播时需要计算激活函数的导数，观察公式(3-1)和(3-2)得知，Sigmoid和tanh函数的表达式中都存在分式，这导致求导需要大量的计算，一旦神经元数量一多，计算量就需要成几何倍数增长，因此如果较深的网络在隐层中使用这两个激活函数，将会导致训练困难。

虽然非线性激活函数为深度学习带来了更优秀的分类表达能力，但这是牺牲了计算成本换来的，我们可以接受其在输出时进行工作，但不能让其担任隐层间数据交换的使者。因此，我们需要一个性质看起来像线性的激活函数，但实际上它是一个非线性函数，这种情况下，它即可以提高模型的分类表达能力，又可以接纳隐层间复杂的数据关系。于是我们找到了以下两种函数。

生成模型中的隐层间激活函数为ReLU函数，公式为：

观察图 3‑6(c)，可以发现当时，函数值为0，当时，函数值为，以原点为界，两侧的函数实际上都是线性关系，但组合成分段函数时，就变成了非线性关系，因此它又被称为PLU（piecewise linear unit, 分段线性单元）。显然，ReLU函数的计算简单，求导数也不困难，如公式(3-4).

以下为求解反向传播过程：

则损失函数关于第层的偏导如下：

ReLU函数的性质为它带来了若干个优点。

一，单侧饱和性。观察图图 3‑6(c)或者公式(3-5)可知，当神经元输出为负值时，ReLU可以使它变为0值，这种性质被称为单侧饱和性。不同于Sigmoid等函数的双侧饱和性，单侧饱和性能让网络更具有鲁棒性。通常来说，正值的大小表示了神经网络传递的有效信息，例如特征信号；负值则一般表示无效信息，例如多余的噪声、不需要的特征信号。正侧的不饱和使得有效信息得以几乎完全保留，负侧的饱和使得无效信息得以几乎完全去除。宏观上来看，神经网络此时拥有了稀疏性，即有部分神经元处于抑制状态，不参与工作，参数之间依赖关系被减轻，模型过拟合的情况被缓解。实际上这和人类大脑的工作情况相似，当人的大脑在处理信息时，并非所有神经元都参与工作，而是由一些基本的神经元传递低维的信息，组合成高维的信息表达。

二，计算简单。ReLU函数求导不涉及浮点数运算，在计算梯度反向传播时能够节省大量的算力，提高整个神经网络的收敛速度。

三，线性行为。从函数图像来看，ReLU更像是一个线性函数。线性函数的优点在于它容易被优化，因此它对权重更新更加友好。

ReLU的性质同时也给它带来了缺点，因为并不是神经元输出的负值都是因为无效信息。通常，激活函数会设置一项偏置项（bias），如果偏置项取值太小，可能无论特征值取多少，输出值永远都是负的，那么经过反向传播算法，该神经元的梯度恒等于0，永远无法更新权重参数，这种情况被称为“神经元死亡”(dying ReLU problem)。

对于生成模型来说，它的工作涉及到大量工作的神经元，因此继续使用ReLU函数有风险造成“神经元死亡”，因此，我们可以对ReLU函数进行微调。

生成模型中的隐层间激活函数为Leaky ReLU函数，公式为：

观察图 3‑6(d)，可以发现当时，函数值不再为0，而是有了一个很小的梯度。值很小，一般取.因为在负值区域给网络泄露（leaky）一点梯度，因此被取名Leaky ReLU函数。Leaky ReLU函数很好的解决了“神经元死亡”问题，但是这也多出了一个超参数需要我们在训练模型时人工调整。其导数公式为：

反向传播求解过程如下：

损失函数关于第层的偏导：

## StyleGAN3模型

StyleGAN3模型在判别模型方面主要基于原有GAN提出的模型，在生成模型上改动较多，且较为复杂，相关论文中只给出了生成模型的结构图，如图 3‑7。

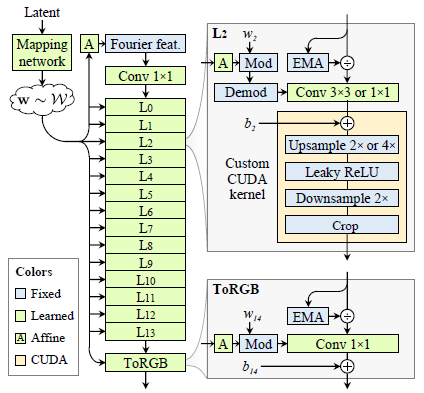
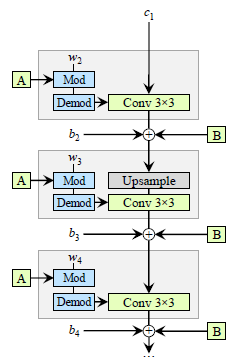


图 3‑7 StyleGAN3生成模型结构图[[4]](#footnote-4)

可以看到，该模型的生成模型共有13个隐层，是一个极深的神经网络，每一层中使用了上采样、leaky ReLU激活函数、下采样，以及crop\_layer层。Crop\_layer层主要是通过裁剪掉最初为了卷积而操作的padding，来实现两个输入图片尺寸保持一致的功能。

因为其模型深度之深，训练StyleGAN3模型要比训练DCGAN花费数十倍的时间，但效果却好了百倍。

# 基于对抗式生成网络的角色形象生成项目实现

## 项目综述

### 项目目标

项目的目标在于直观展示GAN生成图像的能力以及图像的质量，因此，最终结果的可视化是必须的。另外，由于开源的GAN和StyleGAN3仅仅提供训练方法，而不提供模型实例——模型实例需要在本地环境训练得到，因此为了能够从头到尾监控模型，对于训练过程的可视化也是很有必要的。

作者对于项目结果的预期是，项目中所使用到的两个模型都能训练成功，能够做到生成稳定的动漫人物头像的图片；对于最终项目生成的图片结果，我们应该可以比较容易地获取，接着可以进行对比、分析和总结；而从效果上来说，老技术（DCGAN）应该不及新技术（StyleGAN3），因此，对于老技术的改进也应该是项目中重要的一环。

### 项目基础

项目中关于神经网络模型的定义、模型的训练方法等核心代码均继承自官方开源的项目。例如，DCGAN模型的定义和训练来自其论文作者在github上发布的源码，地址为<https://github.com/Newmu/dcgan_code>；同样StyleGAN3模型的定义和训练来自英伟达官方在github上开源的代码，地址为<https://github.com/NVlabs/stylegan3>.

最终，作者从以上开源项目中选择出所需要的代码文件，并将两部分整合在同一个项目中，具体的文件结构可参看下一小节。

## 文件结构

项目文件树目录如下：

.

│ check\_img.py

│ main.py

│ requirements.txt

│ train\_gan.py

│ train\_style.py

│

├─arithmetic

│ ├─gan

│ ├─stylegan3

│

├─data

│ └─faces

│

├─dnnlib

│

├─model

│ ├─gan

│ │ │ netd.pth

│ │ │ netg.pth

│ │

│ └─stylegan3

│ │ network-snapshot-2000.pkl

│

├─torch\_utils

│ ├─ops

│

├─training

│

└─web

├─static

├─templates

各文件夹（文件）和存放文件（内容）对应关系如表格 4‑1所示：

表格 4‑1 源码文件内容描述表

|  |  |
| --- | --- |
| 文件夹/文件 | 存放文件/内容 |
| dnnlib | StyleGAN3本地依赖库 |
| torch\_utils | StyleGAN3本地依赖库 |
| arithmetic | DCGAN、StyleGAN3网络结构类 |
| tranining | StyleGAN3训练需要的函数代码 |
| web | Flask框架初始化和输出结果 |
| data | faces数据集文件，包含约三万张动漫人物头像图片 |
| model | 训练得到的DCGAN模型和StyleGAN3模型 |
| requirements.txt | 环境配置依赖库目录 |
| train\_gan.py | 训练DCGAN的代码文件 |
| train\_style.py | 训练StyleGAN3的代码文件 |
| main | flask可视化界面的启动文件 |

本论文在实验中主要对DCGAN的训练做了研究和优化。表格 4‑2是train\_gan.py中初始化的参数，这些参数在不同方面影响图片生成的结果。

表格 4‑2 train\_gan.py 参数定义和取值

|  |  |  |
| --- | --- | --- |
| 参数 | 参数值 | 参数含义 |
| num\_workers | 4 | 多进程加载数据所用进程数 |
| image\_size | 96 | 图片尺寸 |
| batch\_size | 256 | 图片批处理大小 |
| max\_epoch | 500 | 迭代轮次 |
| lr1 | 2e-4 | 生成模型学习率 |
| lr2 | 2e-4 | 判别模型学习率 |
| beta1 | 0.9 | Adam优化器的beta1参数 |
| gpu | True | 是否使用GPU |
| vis | True | 是否使用visdom |
| d\_every | 1 | 每1个batch训练一次生成模型 |
| g\_every | 5 | 每5个batch训练一次判别模型 |
| nz | 100 | 输入的噪声维度 |
| ngf | 64 | 生成模型feature map数 |
| ndf | 64 | 判别模型feature map数 |

## 可视化

### visdom可视化

visdom是facebook开发的一款可视化工具，对Pytorch支持友好，其本质上是一个基于网页端口的web服务器。在训练模型的过程中，我们可以使用visdom监听并可视化数据，例如损失函数、当前图像等。

在运行环境中使用pip语句安装visdom依赖库，启动visdom需要先在终端输入“visdom”开启监听命令。visdom成功启动后会返回用户一个端口，一般默认为[localhost:8097](http://localhost:8097).

我们可以使用visdom.image方法将生成模型生成的图片和当前模型参照的真图片输送到visdom主界面中。

这样我们在训练过程中就可以实时看到生成图片的效果了。情况如图 4‑4，左侧为假图片，右侧为真图片，每20个epoch更新绘制一次图片：



图 4‑4 visdom监视窗口

### flask可视化

flask是一款基于Python的超轻量级web框架，使用flask可以帮助我们编写简单的网页。本论文中网络训练得到的模型结果都在flask网页上显示。

在web文件夹中定义了网页界面的初始化工作，使用语句app = Flask(\_\_name\_\_)定义一个flask网页，在main.py中使用app.run方法拉起该网页，用户可以自定义浏览器端口，本论文源码定义端口为“7001”。在初始化代码文件中，使用@app.route方法定义网页路由。该网页共有两个路由，@app.route('/gan/generate')和@app.route('/stylegan3/generate')，分别对应网页中的“基于GAN”和“基于StyleGAN3”标签页，网页中的标签、点击事件、方法等都定义在./web/templates/content.html文件中。标签页定义代码请见附录代码段 1所示。

可视化界面最终效果如图 4‑5所示。每一个界面都设置了若干个可以手动调整的参数。调整参数则可以改变图片生成的效果，详细请参看论文4.4.2小节的内容。

|  |
| --- |
|  |
| (a) |
|  |
| (b) |

图 4‑5 flask可视化界面

(a)“基于GAN”标签页 (b)“基于StyleGAN3”标签页

## 生成效果

### 模型训练及结果

运行训练代码文件，模型开始训练参数，可以观察到模型的训练过程如图 4‑1所示。

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |

图 4‑1 DCGAN训练过程图

(a) 50个epoch (b) 100个epoch (c) 300个epoch (d) 500个epoch

可以看出，模型从刚开始训练的时候是充满噪声的状态，因为其本来就是输入的100维噪声维度。刚开始的时候只能看到黑白的噪点，就像电视机收不到信号时候出现的雪花屏，随着训练的进行，到50个epoch后，图像中出现了人物的大致形状，100个epoch后头发有了固定的颜色，300个epoch后人物逐渐成型。整个过程就是噪声逐渐消失，并转化为符合动漫人物头像的特征，例如五官的位置，脸庞的形状，头发的细节、反光等，在500个epoch后，模型训练完成，通过调用该模型，我们可以随机生成任意多张动漫人物头像。最终效果如图 4‑2所示。



图 4‑2 DCGAN最终效果图

然而，观察生成的64张头像，其中仍然有将近一半的头像呈扭曲、模糊的状态，不像一张正常的角色脸。在所有的图片中，两只眼睛的位置和大小普遍不协调、嘴巴的部分摇摆不定。训练最好的部分是人物的头发，该部分比较好地还原了发丝的细节、刘海的形状以及头发反光的状态。可以说，DCGAN对于图片整体的把控和大范围内容的生成是差强人意的，但是对于脸型、五官或者五官之间的联系这类较为精巧的细节并不能很好地还原，因此造成了人物五官时常模糊或者扭曲，看起来不像是一个“人”；且图片整体涂抹感严重，感觉像是加了一层雾状的面纱。下面来看StyleGAN3生成的图片，进行比较。

StyleGAN3的训练时间很长，其模型文件占用空间很大，如果希望其能和DCGAN一样一次性生成64张人物头像，需要电脑计算相当长的时间。为节省计算时间，本实验中只让它一次生成一张图。图 4‑3是选择的四张输出样图。

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |

图 4‑3 StyleGAN3生成结果图

(a)种子号：1010 (b)种子号：1002 (c)种子号：905 (d)种子号：1209

可以看到，StyleGAN3生成图片的效果明显要比DCGAN效果好。首先，StyleGAN3图像质量高，画面更加清晰，当然这和训练使用的数据集密切相关，StyleGAN3使用的数据集图片尺寸更大、更清晰；其次，它对于五官的把控明显优于DCGAN，尤其是眼睛部分，大多数的人物不仅眼睛大小相同，颜色相同，甚至符合透视关系，鼻子部分也比较符合动漫人物的画风，摆放在正确的位置；对于眉毛的细节、头发的细节的还原更上一层楼。

虽然模糊问题得以改善，但仍然可能出现扭曲的情况，例如图 4‑3(d)，图像的扭曲在很大程度上是模型的过拟合引起的，过拟合问题无法完全解决，但是这种情况在StyleGAN3中已经得到了明显的改善。经过测试，大约每4张随机图像中有一张图像呈扭曲状态，坏张率为25%，几乎是DCGAN的一半。

不过因为StyleGAN3的清晰度高，一旦出现人物扭曲，该问题在视觉尺度上会被放大，即更加明显了。

### 改变控制参数后的结果

“基于GAN”标签页每次最多可以生成图片的数量受“总生成数”栏控制，如输入“512”，则每次最多可生成512张图片。最终图片数决定了界面上展示的图片数量。程序首先会计算所有生成图片的损失函数得到损失值，从小到大排序进行输出，因此界面上展示的图片是一批中质量最好的图片。噪声的均值和方差决定了这一批图片的整体质量。均值增大会让生成图片逐渐趋于一致，如图 4‑6所示；方差增大会让生成图片趋于不一致，颜色差别增大，如图 4‑7所示。但两者过度增大，都会造成生成图片的质量降低，人物主体变得异常扭曲。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (a) | (b) | (c) |

图 4‑6 DCGAN方差为1时均值顺序图

(a)均值=0 (b)均值=1 (c)均值=2

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (a) | (b) | (c) |

图 4‑7 DCGAN均值为0时方差顺序图

(a)方差=1 (b)方差=2 (c)方差=3

“基于StyleGAN3”标签页展示的图片受到右侧属性控制。第一项模型是训练得到模型的文件名，后台通过文件名调用对应模型的参数。第二项是随机种子，随机种子必须为整数，每一个数字随机生成一张图像，数字可以是无穷大或者无穷小，其实当输入为负数时，种子会随机映射到一个正数上然后返回它。

Psi截断则类似DCGAN中的均值与方差，psi截断值一般取值在0~1。Psi截断值越小，则生成图像的差异性就越小，当psi截断值为0时，无论随机种子设置为多少，其生成图像不变，永远为图 4‑8(a)，图 4‑8为一张图像psi截断值从0变化到1的情况，间隔为0.25。不难发现，psi截断的作用实际上就是为图像叠加上随机生成的特征，psi截断值越大，生成图片获得特征的程度就越深，也就越和原本的图片以及其他的图片不同。虽然psi截断值越大会让图片越“特立独行”、“与众不同”，但这并不表示它越大越好，因为随着psi截断的增大，人物扭曲的情况越来越容易出现，过多的特征可能导致无效特征的混入，从而让图像变得不再像一个人脸。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| (a) | (b) | (c) | (d) | (e) |

图 4‑8 psi截断值顺序图

(a)psi截断=0 (b) psi截断=0.25 (c) psi截断=0.5 (d) psi截断=0.75 (e) psi截断=1

“旋转”一项与我们常识中的“图片旋转”不是同一概念：并不是让图片整体旋转一个角度，而是像素级别的处理。在2.3节中提到，StyleGAN3为了解决生成图像像素细节的问题，例如发丝粘连屏幕、涂抹等，增加了旋转等变性。通过旋转可以实现像素级别的调整，让生成图像质量更高，因此通常来说，如果要对图像进行优化，旋转角度不宜设置太大，一般2~3°即可，如果旋转角设置过大，则可能会破坏整张图片。如图 4‑9所示。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (a) | (b) | (c) |

图 4‑9 StyleGAN3旋转值顺序图

(a)旋转=0 (b)旋转=10 (c)旋转=90

## 优化方案

在看到StyleGAN3效果比DCGAN好这么多倍后，本论文尝试对DCGAN源码进行改进，希望可以提升其生成图像的质量。于是本实验分析了各个可能影响图像质量的方面，最终尝试了三种主要方法。经实验表明，改进后的DCGAN在生成图像的能力上确实有一定提升，但提升不大。

### 更纯净的数据集和数据清洗

由Yanghua Jin等人发表的相关论文中指出，数据集的质量在很大程度上影响了生成图像的质量，因为DCGAN模型对于数据集图片中的“人工痕迹”特别敏感，而动漫头像图片中则充满了“人工痕迹”。

与“人工痕迹”对应的是“自然痕迹”。试想一张拍摄的山水风景图，其中的自然主体的颜色是连续的，物体与物体之间没有特别明显的分界线，是“自然”的表达，DCGAN模型对这样的图片内容没有那么敏感，即使在生成不同的物体时有轻微的模糊和扭曲处理，也不会影响到图片整体的观感，因此，DCGAN在生成自然风景图这一场景更具有优势，事实上，本来该模型在诞生初期就是被用来模拟生成风景和现实中存在的物体的；然而这种情况不同样适用于动漫头像，动漫图片的色块明显，人物的脸部细节特征需要被精准表达，因此一旦有模糊或者扭曲，这对于图像来说就是致命性的。

动漫图片中最能影响到模型的“人工痕迹”主要有三点：1.人脸和背景的分界；2.五官及五官的相对位置；3.图像整体的清晰程度。

如果数据集中人脸和背景区别度不高，或者图片整体质量不够清晰，那么在生成图像中很容易出现大量横竖的条纹，或者棱角明显的斑块。

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

图 4‑10 数据集对比图

(a) Anime-Face-Dataset[[5]](#footnote-5) (b) anime-faces[[6]](#footnote-6)

实验初始使用的数据集如图 4‑10(a)，共包含约六万四千张动漫人物图像。可以发现，该数据集中的图像大小不一，普遍尺寸较小，像素感严重，而且有部分图片中的人物和背景不能较好的区分，这对于训练来说造成了非常不利的影响。尤其在读入数据集信息时，模型需要先构建一个图像变换库对数据进行处理，以保证模型内的图片大小一致，大小由表格 4‑2中的“image\_size”项控制。一旦图片本身的尺寸小于或大于该设定值，则变换进来的图像需要通过像素插值等手段进行变换，会有一定程度的损失（一般来说小于设定值的图像会损失得远超过大于设定值的图像）。变换部分代码请见附录代码段 2。

经过改进后，实验中使用了如图 4‑10(b)图的数据集，该数据集由约两万一千张图片构成，其数量仅仅是老数据集的三分之一。然而，该数据集的质量比一开始使用的数据集的质量高了很多，主要体现在以下方面：

1. 数据集中图片普遍较大，且大小保持一致。则读入图片时每张图片的损失比较接近。
2. 数据集图片整体比较清晰，细节比较明确。
3. 数据集图片背景基本都为纯白色，很好地与人物主体分离。

接下来，进一步对新数据集进行数据清洗。因为从Kaggle下载的数据集中有些图片无法正常打开，所以我们可以编写一个脚本，依次打开文件夹中的数据集图片，如果无法正常打开，则删除这张图片。脚本代码请看附录代码段 3 check\_img.py。Kaggle数据集中有极少数图片明显不符合要求，例如图片中只有半个人脸，或者是人物的胸部、腿部等其他部位，对于这些图片，我们需要使用opencv工具，对数据集图片中的人脸重新定位和判断，如果不符合opencv判断结果，则将该图片删除。

重新训练模型后，发现生成图片的效果得到一定的改善，结果如图 4‑11所示。新的生成图片展示出了更优越的平滑感，图片中不必要的横竖条纹明显减少了，人物两只眼睛的比例更加协调了。但是图片整体变得暗淡了一点，这应该与数据集的整体颜色有一定关系。

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

图 4‑11 不同数据集训练结果对比图

(a) Anime-Face-Dataset训练结果 (b) anime-faces训练结果

### 参数调整

上一节的研究中提到，影响生成图片质量的两大因素——人脸和背景的分界、真图片的清晰度，可以通过优化数据集来解决，但是该方法对模型的性能提升收效甚微。模型对五官及五官相对位置的掌控是其自身的能力属性，无法通过数据集来影响。因此我们需要调整模型的超参数来改变其训练机能。实验中选取了多个方面的参数进行研究和调整，最终发现对生成图片直接影响较大的有两个方面，分别为模型的学习率和Adam的优化参数，分别由表格 4‑2的“lr1”、“lr2”和“beta1”项控制。

显而易见，抽象来说学习率指的是模型学习的速率，具体来说是指神经网络权重的更新幅度，学习率越大，网络权重更新的幅度就越大。学习率如果设置太大，则网络在梯度下降的时候越过损失极小点，导致网络无法收敛；如果设置太小，就会导致权重更新速度太慢。

实验中两个模型初始的学习率都是2-e4（0.0002），这也是DCGAN论文中推荐的参数设置。现在将其下调至1-e4（0.0001）重新训练。

Adam优化算法（Adaptive Moment Estimation，自适应矩估计）是对梯度下降算法的一种改进算法。训练神经网络时我们需要使用梯度下降法找到全局最优值，即损失函数最小的时候，然而普通的梯度下降法可能会越过全局最优点或者停在局部最优点，导致最后模型参数的结果不是我们所期望的最优值，Adam优化算法通过引入动量和其他相关的因素来更新参数，具体包含梯度的指数加权平均数、平方数和平方根。Adam共有三个超参数，根据Adam原论文作者Diederik P. Kingma等人的推荐，三个参数默认取值为.现将调整为0.5，0.5为DCGAN论文中推荐的参数设置。

重新进行训练，由于降低的学习率，这次训练结束的时间增加了不少，对比结果如图 4‑12。

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

图 4‑12 不同超参数训练结果对比图

(a)lr1=lr2=2e-4, beta1=0.9 (b)lr1=lr2=1e-4, beta1=0.5

观察图 4‑12(b)，可以看出新的参数提升了模型的细节表达能力，生成图片对五官，特别是眼睛和嘴巴，在一定程度上提高了掌控力。

### 预训练模型迭代

一次训练结束的默认迭代次数是500个epoch，增加迭代次数也可以提高生成图像的质量。例如直接将代码中的max\_epoch改成1000.

不过实验中发现，通过预训练模型的迭代效果要稍好一点。预训练模型指的是已经训练到一定程度的模型。例如，现在需要模型训练1000个epoch，我们可以不让模型一次性完成训练，而是先让其训练500个epoch，然后将训练得到的模型输出，此时已经拥有了一个模型的权重文件。再次运行训练的代码，此时将刚刚输出的模型作为初始权重开始训练，运行结束后，我们就可以得到一个训练了“500+500”个epoch的模型。

对比效果如下图：

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

图 4‑13 预训练模型结果对比图

(a)max\_epoch=1000 (b)max\_epoch=”500+500”

图 4‑13(b)是结合了三种优化方式的最终效果图，总体上来说，相比于最开始训练得到的结果，其生成效果有了一定的改善，坏张率有了一定的减少，但是仍然没有质的提升。如果要使DCGAN突破其固有的性能，最终还是要改变其网络结构、增加网络深度、添加功能性网络层，才可以达到类似StyleGAN3的效果，只不过在那个时候，DCGAN恐怕需要改一个新的名字了。图 4‑14是模型随机生成单个头像样图的结果。



图 4‑14 改进DCGAN模型生成单个样图

# 总结和讨论

## 优劣分析

对比两者，DCGAN的隐层只有区区四层，StyleGAN3的优势毋庸置疑，更多的隐层为训练增加了广泛的分类表达能力，而同时，训练时长和计算量都大大增加，是CPU和GPU等硬件技术的发展为GAN技术的提升带来了空间，这正是新技术优于老技术的主要原因。其次，StyleGAN3在DCGAN框架中添加了多种针对性的网络层，目的是改善生成图像某些特定的方面，因此，在功能性和目的性上，新技术自然比老技术更胜一筹。

DCGAN作为最早一批的GAN技术，其性能有限是情理之中，加之其诞生原本是为了加强生成真实的人脸、风景、物体的能力，原本就不擅长生成“人工痕迹”为重的图片。重要的是DCGAN引入了卷积神经网络，为后世生成图像的方向奠定了基础，StyleGAN3等其他效果非常好的GAN变体也都是基于DCGAN的思想被创造出来的。

当然，框架之于不同的数据集，每一次训练结果都会不同，对于不同的数据集，即使是使用同一个模型，也同样存在优化的空间，即可以通过调整超参数来让模型拥有更好的效果。因此，在做模型训练时，模型框架的选择，参数的调整都是非常重要的。

## 区别度问题

当我们已经拥有了如此大规模的数据集后，我们为什么还要使用GAN训练生成模型，再让模型生成“模仿”的图片？或者问，生成图片和数据集图片的区别度在哪里，有什么必要大费周章去得到原本就存在的事物？

首先，理论上来说，如果我们拥有无限庞大的数据集，确实没有必要创造一个生成模型。现实是，我们的数据集是有限的，然而有限数据集的特征可以组合出近乎无限的高维特征，例如头发的颜色加上眼睛的颜色，光是这两个特征就可以组成数百种可能。作为人类，要认真地创造一个蓝头发加红眼睛的角色就可能需要耗费许多时间成本，而生成模型允许我们仅改变一个参数，就能实现一个角色的诞生。因此，GAN能够成为节省时间的利器。

其次，从创新的角度来说，GAN帮助我们训练出来的模型并不是顽固的，即使我们使用同一个模型，设置同样的属性，生成的结果也会有微小的不同，这是顽固的数据集无法做到的。

最后，毕竟GAN不仅仅只有生成动漫图片这一应用场景。就拿StyleGAN3来说，图 4‑8展现了GAN与图像连贯性之间的联系，这让人不禁想到另一个场景——动画，而现在确实也有人将GAN技术应用在动画生成中，实际上在引言中提到的Crypko网站也在进行同样的项目。因此从创新和交叉领域的角度来说，GAN技术是相当有意义的。

# 结语

从初识GAN到成功输出一张图片，已经过去了近两个月的时间。在这段时间内，笔者阅读了大量的论文，接触到了各式各样的GAN技术，最终，笔者决定研究DCGAN和StyleGAN3，并将两者进行对比。学习GAN需要非常多的前提知识储备，例如机器学习、神经网络、深度学习算法等等，对于从零开始的笔者来说，妄图在GAN技术上有所创新是一件困难的事情，尽管如此，笔者还是尽其所能，在模型优化和可视化上下足了功夫。尤其在优化方案上，笔者细致地钻研了两者的训练代码，通过对比，从StyleGAN3中吸取利于生成模型的要素，将其运用在DCGAN的模型中，最终得出了文中的三种优化方案。可视化则让程序更加直观、有趣，文中选取的StyleGAN3图例的种子，都是笔者和室友的生日，每次在输入种子得到图片之前，我们的心中都会有无限的期待，也许这就是GAN的魅力所在。GAN的应用范围广泛，毕竟“万物皆可GAN”，它充分地满足了人类对于挥洒创造力的渴望。

研究代码的过程让笔者更深刻地了解了神经网络和深度学习的知识，体会到了新老技术的变迁。实际上，DCGAN在以现在的视角看来，只是一个框架，而并不是一个具体模型，因为它已经不再能够满足人类的需求，而StyleGAN3更能被称作是“模型”。如同造房子，DCGAN是地基，在地基上一点一点地加入不同的网络层，就可以造出不同功能的建筑，StyleGAN3则是其中一栋优质的大厦。技术正是如此，需要一点一点的突破，不断累积到一定程度，技术就会发生质变。解决问题的途径同样如此，笔者在实验中遇到了非常多的问题，例如代码在windows环境下的不兼容，cuda版本的不适配，生成模型的结果非常差，无法在网页端正确输出图片，无法利用opencv正确清洗数据等等，有些遇到的问题已经记不清了。通过持续的努力，一个个问题最终都被克服，本论文最终能够顺利结束。

当然，本实验中也存在着比较严重的问题。由于GAN涉及到的知识体系非常庞大，笔者在实验之前，花费了大量的时间（约一个月）学习网课，却没有实践。在上手之后才发现，实践才是学习效率最高的途径。受限于硬件，解决bug和训练模型都需要很长的时间，更不用说还需要调整参数重新训练模型，以获取多面的实验结果。这导致本论文中的实验数据非常有限，模型样本较少（DCGAN有三个，StyleGAN3仅有一个），不能很好地避免偶然性，数据分析工作也大大受限。因此在未来，笔者会继续研究、跟进，在训练模型上花费更多的时间，能让样本更加丰满，力求更好的生成效果。

最后笔者认为，GAN技术灵活、应用范围广，且最重要的是，它满足了人类对于“创造”、“艺术”、“好奇心”的渴望，从这一点来说，GAN在未来的发展一定会愈加蓬勃，诞生出越来越多不同的变体，丰富这个充满活力的大家族。笔者对于这样的未来甚是期待。

# 致谢

在将近两个月的时间里，我身边的老师和同学们给予了我莫大的帮助。

首先我要感谢我的指导老师王毅刚老师。可以说没有王老师，我的实验和这篇论文绝对无法顺利完成。由于我的电脑系统是windows，与源码之间存在兼容性问题，导致实验在根本上就无法进行。在我将这一点反馈给王老师后，他毫不犹豫地借给了我一台实验室的电脑，并且允许我任意地操作。于是我在该电脑上安装了Linux，成功完成了实验。在过程中我多次找他解决问题，他都非常耐心地、积极地帮助我解决，顺利推动了我的进度。在此，我要再一次感谢王老师的奉献。

其次，我要感谢我的两位室友。毕业项目实践和毕业论文产出的过程必然是无比枯燥的，然而他们在这段枯燥的日子里给予了我许多轻松的、愉悦的时刻，这让我有了继续的动力。尽管论文题目大相径庭，我们还是常常互相帮助，比如研究论文的格式、排版，帮助传送资料等等。可以说，我的室友无论在精神层面还是物质层面，都是我生活中有力的推进器。

最后，我要感谢母校对我四年精心的栽培，在学校的四年，我渐渐地从象牙塔里走出来，接触到科研，接触到社会，认识到一个成年人肩上要担的责任，虽然都只是冰山一角，但它们都成为了我宝贵的经验。大学的四年是我人生中最值得珍藏的时光。

# 参考文献

* 1. Goodfellow I J , Pouget-Abadie J , Mirza M , et al. Generative Adversarial Networks[J]. Advances in Neural Information Processing Systems, 2014, 3:2672-2680.
  2. Xue A . End-to-End Chinese Landscape Painting Creation Using Generative Adversarial Networks[J]. 2020.
  3. Radford A , Metz L , Chintala S . Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks[J]. Computer ence, 2015.
  4. Karras T , Laine S , Aila T . A Style-Based Generator Architecture for Generative Adversarial Networks[C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019.
  5. Karras T , Aittala M , Laine S , et al. Alias-Free Generative Adversarial Networks[J]. 2021.
  6. Nunn E J , Khadivi P , Samavi S . Compound Frechet Inception Distance for Quality Assessment of GAN Created Images[J]. 2021.
  7. Ioffe S , Szegedy C . Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. JMLR.org, 2015.
  8. Glorot X , Bor De S A , Bengio Y . Deep Sparse Rectifier Neural Networks[C]// Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS). 2011:315-323.
  9. Jin Y , Zhang J , Li M , et al. Towards the Automatic Anime Characters Creation with Generative Adversarial Networks[J]. 2017.
  10. Kingma D , Ba J . Adam: A Method for Stochastic Optimization[J]. Computer Science, 2014.

# 附录

代码段 1标签页定义

# 基于gan的动漫头像生成

@app.route('/gan/generate',methods=['POST'])

def generate\_image\_gan():

# 获取所有的参数

data = request.form

# 可以通过 request 的 args 属性来获取参数

generate(

gen\_search\_num=int(data.get("gen\_search\_num")),

gen\_num=int(data.get("gen\_num")),

gen\_mean=int(data.get("gen\_mean")),

gen\_std=int(data.get("gen\_std")),

)

# 返回json类型字符串

return return\_json({

"res": "/static/gan\_img.png"

})

# 基于stylegan3的动漫头像生成

@app.route('/stylegan3/generate', methods=['POST','GET'])

def generate\_image():

# 获取所有的参数

data = request.form

print(data)

seed = int(data.get("seed"))

# 如果不填的话就自动生成

if seed < 0:

seed = random.randint(0, 4294967295)

print("种子", seed)

generate\_images(

seed=seed,

truncation\_psi=float(data.get("truncation\_psi")),

noise\_mode=data.get("noise\_mode"),

rotate=float(data.get("rotate")),

network\_pkl="model/stylegan3/%s" % data.get("model")

)

# 返回生成的图片和种子

return return\_json({

"res": "/static/style\_img.png",

'seed': seed

})

代码段 2 构建图形变换库

# 构建一个图形变换库，这里是对我们的图像进行处理

transforms = tv.transforms.Compose([

tv.transforms.Resize(opt.image\_size),

tv.transforms.CenterCrop(opt.image\_size),

tv.transforms.ToTensor(),

tv.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))

])

import os

from PIL import Image

# 读取所有文件

for file in os.listdir("data/faces"):

filename = "data/faces/%s" % file

try:

Image.open(filename)

except:

os.remove(filename)

print("%s错误" % filename)

print("mission completed!")

代码段 3 check\_img.py

1. 图片源自crypko网站，网站网址：https://crypko.ai/ [↑](#footnote-ref-1)
2. FID指Frechet Inception Distance score，即Frechet Inception 距离得分，是计算真实图像和生成图像的特征向量之间距离的一种度量。FID从原始图像的计算机视觉特征的统计方面的相似度来衡量两组图像的相似度，分数越低代表两组图像越相似，FID 在最佳情况下的得分为0.0，表示两组图像相同。FID 分数被用于评估由生成性对抗网络生成的图像的质量，较低的分数与较高质量的图像有很高的相关性。 [↑](#footnote-ref-2)
3. 图片源自论文《Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift》 [↑](#footnote-ref-3)
4. 图例来自论文《Alias-Free Generative Adversarial Networks》 [↑](#footnote-ref-4)
5. Anime-Face-Dataset数据集源自<https://github.com/bchao1/Anime-Face-Dataset> [↑](#footnote-ref-5)
6. anime-faces数据集源自[Anime Faces | Kaggle](https://www.kaggle.com/datasets/soumikrakshit/anime-faces) [↑](#footnote-ref-6)