基于条件生成对抗网络（cGAN）的图像生成

江熠 3020207160

摘要

近年来，生成对抗网络（GAN）作为一种主流的生成式网络模型被广泛应用于图像生成等领域，并取得了较好的效果。以 GAN 为基础的条件生成对抗网络（cGAN）则进一步限定了图像生成的特征范围。然而，原生 cGAN 在训练中存在的模式坍塌、生成质量低、损失不稳定等诸多缺点，严重限制了 cGAN 模型的生成能力。针对此问题，本文基于现有的研究结果，在卡通人脸图像数据集上对于 cGAN 的训练行为进行了研究，从而得到了在训练中对于 cGAN 参数调优的一些技巧，并结合深度卷积神经网络（DCNN）的思想，提出了对于 cGAN 的网络结构进行改进的可行方案 cDCGAN，并通过实验验证了改进方案的可行性。

关键词— cGAN, 图像生成, 模式坍塌, 参数调优, DCNN

1. 引言

生成对抗网络（Generative Adversarial Networks，GAN）*[1]*是一种生成式网络模型，由生成器 G（Generator）和判别器 D（Discriminator）组成。G 试图生成与真实样本（可以是图像、音频或是文本）相似的数据，而 D 则尝试区分真实样本和生成器生成的假样本。二者在训练过程中呈现“对抗”的关系，因而得名“生成对抗网络”。通过训练不断调整二者的参数，从而使得生成器 G 可以生成高质量的假样本，达到以假乱真的效果。GAN 的应用范围非常广泛，如图像生成、风格转换、图像修复、音频生成、自然语言生成等。本文主要研究 GAN 在图像生成领域的应用。

条件生成对抗网络（conditional GAN，cGAN）*[2]*则是在 GAN 的基础上为每个样本添加了条件变量（如文本、图像标签等），使得生成器和判别器都能够获取样本的条件信息，其主要目的是使生成的结果能够更加符合预期的特征。

不同于传统的单网络模型（如 CNN、RNN 等），GANs 包含了两个相互对抗的网络，因此其训练难度也有所提升。最经常出现的训练异常情况就是“模式坍塌”（mode collapse），即生成器没有学到整个真实数据的分布，而仅仅学到了整个空间中的一个非常小的分布，从而导致生成样本的效果不佳。除此以外，在两个网络各自的学习能力出现不平衡的情况时，将会导致训练损失极度不稳定。例如，若判别器 D 的学习能力强于生成器 G，则 D 在训练的后期将会一直抑制 G 的学习，从而导致生成的图像质量不佳。本实验主要关注 cGAN 网络模型的实际训练，并从两个方面提出了避免训练异常问题的策略。

一方面可以在不改变模型结构的基础上通过调整训练参数，如调整学习率、调整生成器 G 和判别器 D 的训练轮次、平滑化标签等方式指导模型的训练行为，从而尽可能地平衡 G 和 D 的训练过程，避免训练异常行为的出现。这样的方式统称为“参数调优”。

另一方面，原生的 GAN 结构设计确实存在训练上的困难性，只凭借训练中的参数调优有的时候也难以达到预期的效果。已经有很多研究在保留原生 GAN 的思想的基础上对于其网络结构进行改进，如 WGAN*[3]*利用 Wasserstein 距离的思想解决模式坍塌问题、DCGAN*[4]*将深度卷积神经网络（Deep Convolutional Neural Networks，DCNN）与 GAN 结合提高生成器生成图像的能力等。

在本节的最后一段中，我将对本技术报告的主要贡献总结如下：

1) 实现了基本的 cGAN 网络，并在该网络的基础上完成图像生成的任务，这是进行后续任务的前提；

2) 对实现的 cGAN 网络模型进行不同层面上的参数调优，分析影响图像生成质量的关键因素，并给出结论；

3) 以现有的研究成果与论文为基础，结合 DCNN 的思想，提出了对基础的 cGAN 模型的改进方案 cDCGAN，并通过实验验证了改进方法的可行性和实验效果。

2. 研究方法

2.1 基本网络架构

下面将介绍 cGAN 的基本网络架构以及训练方法。

cGAN 分为生成器 G 和判别器 D 两部分。生成器 G 的目的是生成虚假的数据。其结构实际上是一个多层感知机（Multilayer Perceptron，MLP），其接受随机噪声 $\boldsymbol{z}$ 以及条件变量 $\boldsymbol{y}$ 作为输入。其输出为与真实数据相似的虚假数据 $G(\boldsymbol{z}|\boldsymbol{y})$。

判别器 D 的目的是区分真实数据和生成器生成的虚假数据。D 的结构也是一个 MLP，实际上是一个二分类器，接受样本数据 $\boldsymbol{x}$ 以及其对应的条件变量 $\boldsymbol{y}$ 作为输入。其输出为样本数据为真实数据的概率 $D(\boldsymbol{x}|\boldsymbol{y})$。

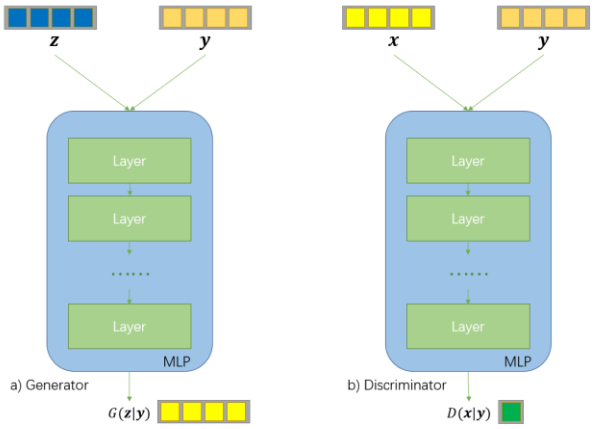

图 1 cGAN 基本结构

训练时，先训练生成器 G，再训练判别器 D。首先生成随机噪声 $\boldsymbol{z}$ 以及随机的条件变量标签 $\boldsymbol{y}$，输入生成器 G 中，得到虚假数据 $G(\boldsymbol{z}|\boldsymbol{y})$。再将 $G(\boldsymbol{z}|\boldsymbol{y})$ 作为样本数据 $\boldsymbol{x}$，和生成的条件变量标签 $\boldsymbol{y}$ 一起送入判别器 D 中，得到输出 $D(G(\boldsymbol{z}|\boldsymbol{y})|\boldsymbol{y})$。利用该结果计算 G 的损失，公式如下*[1]*：

$$L_G = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[1 - log\, D(G(\boldsymbol{z}|\boldsymbol{y})|\boldsymbol{y})] \quad (1)$$

再将该损失反向传播，即完成了 G 的一次训练。在实践中，公式 (1) 的梯度不够明显，容易造成梯度消失的问题。为了避免该问题，一般使用以下的损失函数

$$L_G = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[-log\, D(G(\boldsymbol{z}|\boldsymbol{y})|\boldsymbol{y})] = BCELoss(D(G(\boldsymbol{z}|\boldsymbol{y})|\boldsymbol{y}), 1) \quad (2)$$

训练判别器 D 时，则将虚假数据 $G(\boldsymbol{z}|\boldsymbol{y})$ 以及真实数据 $\boldsymbol{x}$ 以及它们对应的条件变量标签 $\boldsymbol{y}$ 送入判别器 D 中，分别得到输出 $D(G(\boldsymbol{z}|\boldsymbol{y}_{\boldsymbol{z}})|\boldsymbol{y}_{\boldsymbol{z}})$ 以及 $D(\boldsymbol{x}|\boldsymbol{y}_{\boldsymbol{x}})$。利用该结果计算 D 的损失，公式如下*[1]*：

$$L_D = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[1 - log\, D(G(\boldsymbol{z}|\boldsymbol{y}_{\boldsymbol{z}})|\boldsymbol{y}_{\boldsymbol{z}})] + \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[log\, D(\boldsymbol{x}|\boldsymbol{y}_{\boldsymbol{x}})] \quad (3)$$

再将该损失反向传播，即完成了 G 的一次训练。在实践中，一般使用以下的损失函数

$$L_D = BCELoss(D(G(\boldsymbol{z}|\boldsymbol{y}_{\boldsymbol{z}})|\boldsymbol{y}_{\boldsymbol{z}}), 0) + BCELoss(D(\boldsymbol{x}|\boldsymbol{y}_{\boldsymbol{x}}), 1) \quad (4)$$

训练的流程如下图所示，图 a) 和 b) 分别展示了 G 和 D 的训练过程：

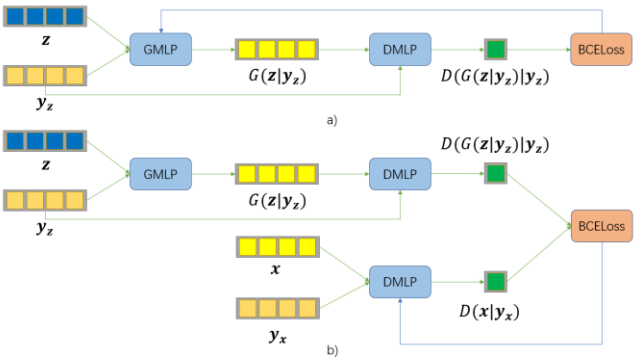

图 2 cGAN 训练流程

2.2 参数调优原则

以上的训练方式存在一些实践上的问题。最常见的问题在于，由于生成器 G 和判别器 D 的网络结构相近，都是用 MLP 作为基础网络，而 G 的任务明显要比 D 的任务难很多（D 只需要给出概率结果，而 G 需要生成完整的数据信息），因此在训练中，最经常发生的情况就是 G 的训练进度远远落后于 D，其训练完全被 D 压制了。而这是我们不希望看到的（我们最终需要的是生成器网络而不是判别器网络）。模式坍塌问题就源于此，判别器训练过于快了，导致 G 的进步空间太小，于是为了尽可能能降低损失，G 通过“欺骗”的方式，只学习一小部分样本的分布，导致生成的图片大量地同质化。

如果不考虑修改网络的结构，仅从调整参数的角度考虑，有以下几种理论上可行的参数调优方案：

1. 调整学习率*[5]*

该策略也被称为 TTUR（Two Time-Scale Update Rule，双时间尺度更新规则）。一般来说，G 和 D 的学习率设置是相同的。然而根据之前的分析，二者从本质上来说要学习的特征并不相同，因此设置相同的学习率并不一定能够达到较好的效果。

TTUR 策略强调生成器 G 和判别器 D 应当采取不同的学习速度。判别器 D 采用高速学习的策略（即高学习率），而生成器 G 采用低速学习的策略（即低学习率），这样一来，G 能够更小步地进行迭代，从而尽可能避免落入模式坍塌的“陷阱”之中。这一策略虽然简单，但却能够收获意想不到的效果。目前已经有 GAN 相关的研究将这一策略投入了实践，例如引入了自注意力机制的 SAGAN*[6]*。

2. 标签平滑化*[7]*

这个策略也同样比较简单。其主要操作就是在训练判别器 D 时，提高虚假样本的置信度，降低真实样本的置信度，例如将虚假样本的标签设置为 0.05 或 0.1 而不是 0，或者将真实样本的标签设为 0.95 或 0.9 而不是 1。

这样做的目的在于避免 D 对于自己的分类结果过于“自信”，加大 D 的学习难度，以平衡 G 和 D 的训练进度。实际上，加大 D 的学习难度还可以通过数据增强等其它手段，但标签平滑化是最为简单直接的方法。

3. 调整训练轮次

这个策略的操作为延缓 D 的更新。一般来说，每训练一次 G 就要同时训练一次 D，但由于 G 的训练进展缓慢，因此可以选择训练多次 G 后再训练一次 D，这样就可以推迟 D 的损失下降速度，从而延缓 D 的更新。

这样做的目的在于给予 G 更多的训练“机会”，让 G 不要落后于 D 太多。但由于 D 最终还是会趋近于收敛，且其训练难度远远低于 G，因此在实践中这一操作的效果有限。

2.3 网络结构改进

以上的几种策略都是从参数调优的角度对训练过程进行改进的。这些策略在理论上来说具有一定的可行性，但在实践中仅仅通过调整参数有时也难以达到理想的效果。因此，有不少研究对于原生 GAN 网络的结构提出了改进的思路。以下简要地介绍两种改进方法。

1. WGAN

WGAN，即 Wasserstein GAN，利用了 Wasserstein 距离替代了原始的损失函数，从而解决了训练不稳定的问题，并从根本上规避了模式坍塌的风险，而且对原生 GAN 的网络结构的改动并不大。具体的数学推导可以参见原论文*[3]*。在这里仅阐述 WGAN 相对于原生 GAN 的改动之处：

i) 判别器 D 的最后一层不使用 Sigmoid 函数；

ii) 在计算生成器和判别器的损失时不取对数；

iii) 限制判别器 D 的参数的绝对值不超过某个常数 c 规定的范围 $[-c, c]$；

iv) 使用 RMSprop 或 SGD 等不基于动量的优化器。

这些措施确实可以从理论上解决 GAN 的训练异常问题。但在实际使用时，常数 c 的选择却成为了难点。若过大，则限制的效果有限；若过小，则会导致梯度异常的情况。由于数据集特性、超参数设置等具体条件各不相同，实际上并没有一个通用的常数可供使用。在训练时需要根据实际情况选择合适的常数 c。这无疑为训练增添了新的麻烦。已经有策略对于这一问题提出了解决方案，如带有梯度惩罚的 WGAN-GP*[8]*。但这些策略又为网络结构增添了新的复杂性。

2. DCGAN*[4]*

原生 GAN 的一大缺点就在于生成器 G 的网络结构为 MLP，其表现能力有限，即使不断增加 MLP 的层数也很难提高生成图像的质量。

DCGAN 基于深度卷积神经网络（DCNN），提出了将生成器和判别器的 MLP 均替换为 CNN 结构，以增强网络的表现能力。改进策略具体表现为以下几点：

i) 生成器和判别器均使用不带池化层的 CNN 结构，生成器使用反卷积模块替代卷积模块，而判别器使用正常的卷积模块；

ii) 在生成器和判别器中都使用批量归一化（Batch Normalization，BN）模块；

iii) 生成器 G 的中间层使用 ReLU 作为激活函数，输出层采用 Tanh 作为激活函数；

iv) 判别器 D 的输出层采用 LeakyReLU 作为激活函数；

v) 实际训练时采用较小的学习率与动量，保证训练的稳定性。

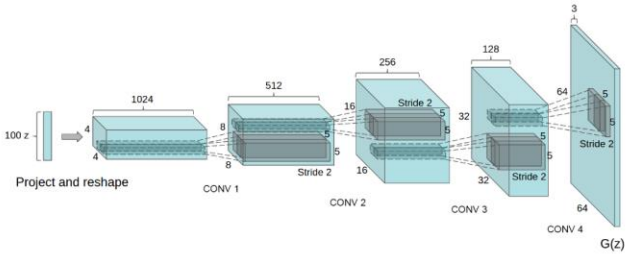

图 3 DCGAN 生成器结构*[4]*

原论文中提出的 DCGAN 是基于不带条件变量的 GAN 的。对于 cGAN 来说，还需要对 DCGAN 的结构做出一些变化。最重要的变化在于如何将 DCGAN 原来的单输入改为 $\boldsymbol{z}$ 或 $\boldsymbol{x}$ 与 $\boldsymbol{y}$ 的双输入。本文称其为“条件化改进”。

对于生成器 G，由于隐含编码 $\boldsymbol{z}$ 与条件变量 $\boldsymbol{y}$ 的维数是相同的（只有 1 个维度），故直接将二者连接在一起输入网络即可。也就是说，生成器 G 仅需要改动网络的第一个反卷积层，将输入通道数扩展为 $\boldsymbol{z}$ 与 $\boldsymbol{y}$ 的通道数之和。

对于判别器 D，样本数据 $\boldsymbol{x}$ 的维度（图像数据一般有 3 个维度）与条件变量 $\boldsymbol{y}$ 的维度不同，因此不能够直接将二者连接在一起。针对这一问题有两种解决方案。一种方案是使用条件卷积（或者动态卷积），将条件信息作为滤波器的权重，自然地将条件信息和图片信息融合在一起。但这种方案实现较为复杂，计算量较大。另一种方案是先对图像进行卷积，生成图像的特征向量，然后将条件信息与卷积后的图像特征向量拼接在一起，再通过 MLP 结构输出最终的概率值。这种方案实现比较简单，额外的 MLP 的参数量也相对较小，不会造成计算量的大幅增长。因此本文采用了后一种方案。

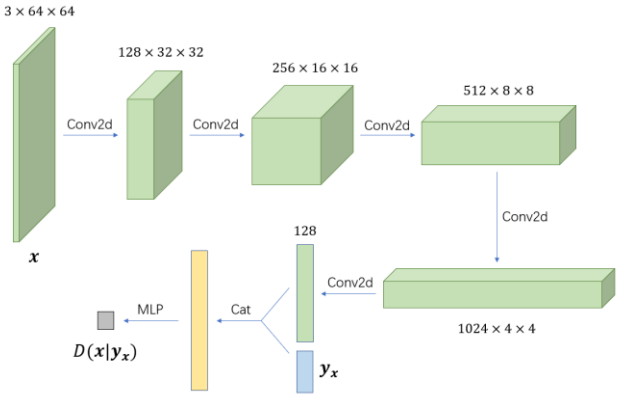

图 4 DCGAN 判别器条件化改进示意图

基于上述的改动思路，本文提出了 cDCGAN（conditional Deep Convolution GAN，条件深度卷积生成对抗网络）这一模型，将 cGAN 与 DCGAN 进行融合，并使用该网络模型与原生 GAN 进行对比实验。

3. 实验与结果分析

3.1 实验数据集与实验环境

本实验的数据集来自于百度飞桨 AI Studio 公开数据集平台，下载地址为 https://aistudio.baidu.com/aistudio/datasetdetail/82357。该数据集包含了 36,740 张大小为 64×64 的卡通人脸图像，并按照发色和瞳色进行了标注。其中发色分为 12 类，瞳色分为 10 类。由于其具有图片尺寸较小、数据量适中、类别数量适中且具有完整明确的分类标注信息，故适合于本实验中对于 cGAN 网络模型的训练。

为了加快数据的读取速度，需要对数据集进行预处理，将图片与标签进行合并。方法为：先为发色和瞳色编号，再将图片重命名为“图片编号_发色编号_瞳色编号”的格式。预处理后的数据集如下图所示。

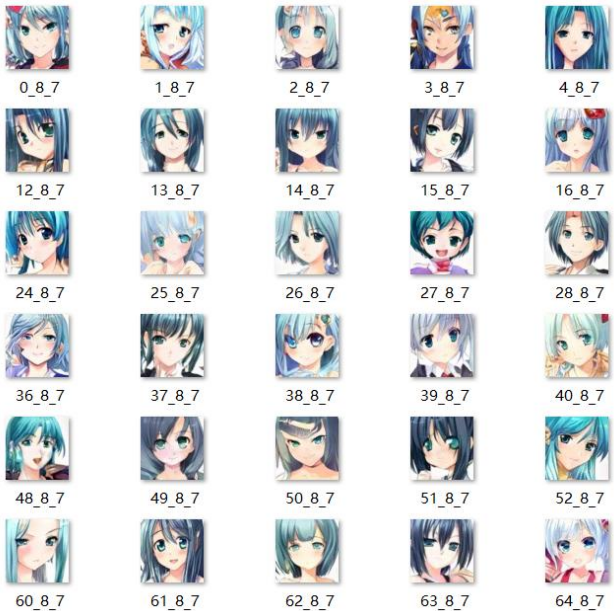

图 5 实验用数据集展示

本实验的系统环境为 Windows 10 操作系统，编程语言为 Python 3.8，主要采用 PyTorch 库构建神经网络。实

验所用 GPU 为 NVIDIA GeForce RTX 2060，显存大小为 6G。实验所用 PyTorch 库的版本为 1.10，CUDA 计算库的版本为 11.3。除此以外，本实验还使用了 torchvision、numpy、pandas、torchsummary 等第三方库。

3.2 实验内容

本实验的主要目的为比较各种对于 cGAN 进行参数调优的策略在实践中的效果，以及 cDCGAN 相较于 cGAN 的提升。基于这样的实验目的，设计了以下 5 组实验用于对比：

1. 原生 cGAN：batch 大小为 512。生成器和判别器均使用 Adam 优化器，学习率一致，均为$lr_g = lr_d = 10^{-3}$，参数设置为$\beta = (0.5,0.999)$。训练 D 时真实样本的标签设为 1.0，虚假样本的标签设为 0.0。共训练 100 轮，生成器与判别器训练次数一致。

2. TTUR：生成器学习率设定为$lr_g = 10^{-4}$，其它参数与 1 相同。

3. 标签平滑化：真实样本的标签设定为0.9，其它参数与 1 相同。

4. 调整训练轮次：生成器每训练 3 次，判别器训练 1 次，其它参数与 1 相同。

5. cDCGAN：生成器和判别器采用 CNN 结构，batch 大小为 256。生成器和判别器均使用 Adam 优化器，学习率一致，均为$lr_g = lr_d = 2 \times 10^{-4}$，参数设置为$\beta = (0.5,0.999)$。训练 D 时真实样本的标签设为 1.0，虚假样本的标签设为 0.0。共训练 60 轮，生成器与判别器训练次数一致。

为了比较训练的效果，完成每一轮的所有迭代后，将计算该轮训练中所有迭代的 G 和 D 的损失的平均值，并作损失-迭代轮次图线，以便于观察 G 和 D 的损失的变化趋势。除此以外，为了更加直观地感受生成器生成图像的质量的变化，每一轮训练均随机保存 25 张生成器生成的虚假图像样本，以供实验分析。代码仓库的地址为 https://github.com/OgisoSetsuna1/NNDL。

3.3 实验结果与分析

1. 原生 cGAN

本组为对照组。观察损失-迭代轮次图线可知，对于原生的 cGAN 网络，G 的损失一直呈上升的趋势，而 D 的损失则一直呈下降的趋势。且在训练的后期 G 的损失波动越来越明显。观察生成的图像，可以明显看到从第

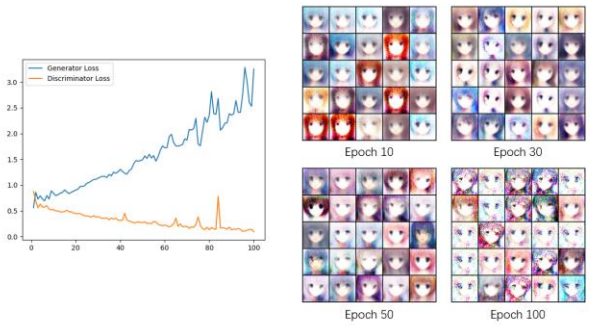

图 6 原生 cGAN 实验结果

10 轮到第 30 轮，色彩的多样性有大幅度的提升，生成图像的噪点也减少了；但从第 30 轮到第 50 轮并没有明显的提升，甚至到了第 100 轮时，图像的质量反而下降了，不仅模糊的情况没有得到改善，而且图片中还出现了大量的噪点，且有多幅相似的图像，这是模式坍塌的表现。

出现这一情况的原因主要在于原生 cGAN 网络的不平衡性，导致判别器过于强势，生成器进步的空间十分有限，生成器只好选择模拟一小部分样本的分布以“欺骗”判别器，从而产生了模式坍塌。这与本文之前的分析是一致的。

2. TTUR

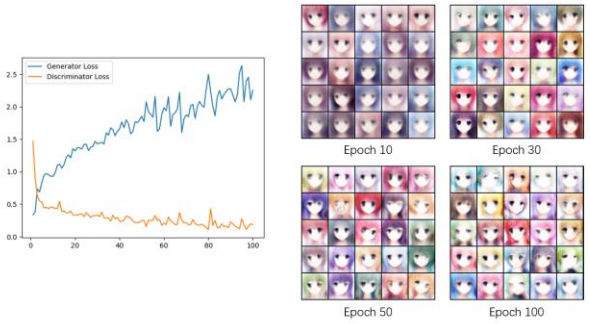

图 7 TTUR 实验结果

观察损失-迭代轮次图线可知，相较于原生的 cGAN，采取了 TTUR 策略以后，虽然 G 和 D 的损失依然保持上升和下降的趋势，但 G 损失的上升趋势明显变缓。生成的图像也没有出现明显的噪点和重复。这是由于生成器保持较慢的迭代速度，从而避免了模式坍塌的问题。

然而，G 的损失在后期波动的问题依然没有得到解决，且后期生成的图像的质量虽然没有下降，但也没有

明显的提升，依然非常模糊。这是生成器本身的网络结构所导致的。即使采用了 TTUR 的策略，也难以从根本上改善生成图像的质量。

3. 标签平滑化

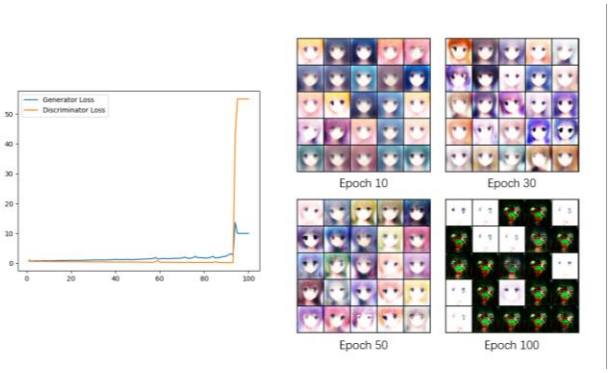

图 8 标签平滑化实验结果

标签平滑化在前期与原生 cGAN 的效果基本一致，在损失的波动性上要略好于原生 cGAN。然而到了训练后期的某一个迭代时，G 和 D 的损失突然急剧升高并达到一个很大的值，随后在这个值保持不动，出现了梯度爆炸的现象。观察生成的图像，在第 100 轮时出现了严重的模式坍塌现象。不同于原生 cGAN 出现的轻微模式坍塌，本组实验中出现的模式坍塌十分极端，任何的输入数据都被极端地分化为两种模式。这样的模型继续训练已经没有任何意义了。

此问题并不是偶然现象。多次重复训练测试表明，标签平滑化在本数据集上极易引发模式坍塌的问题。目前尚不清楚导致该问题的具体原因。猜测可能是因为平滑后的数据标签导致了 D 的损失函数计算异常。由于 G 和 D 是相互影响的，这一异常将影响 G 的梯度变化，而 G 的梯度变化也会通过损失函数反过来影响 D 的梯度变化。在这个过程中异常被逐渐放大，最终导致训练陷入了梯度爆炸的恶性循环当中。这也揭示了原生 cGAN 训练的困难性，小的异常就有可能引发整个模型的完全崩溃。

针对这一问题也有理论上的解决的方法，包括 TTUR 与标签平滑化结合，或者采用梯度截断以防止异常梯度的无限放大等。

4. 调整训练轮次

观察损失-迭代轮次图线可知，相较于原生的 cGAN，G 和 D 的损失依然保持上升和下降的趋势，但 G 损失的

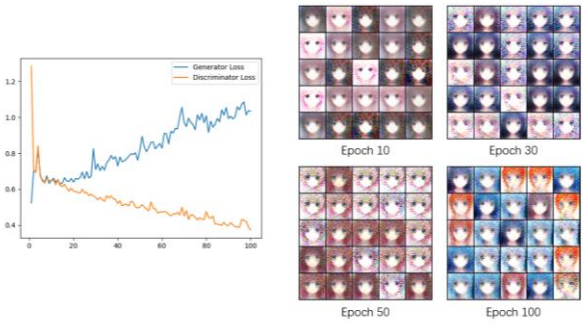

图 9 调整训练轮次实验结果

上升趋势也明显变缓。并且训练后期的波动性也要好于原生 cGAN。但实际图像显示的效果则相反，图像中噪点和模糊的现象比较严重，且模式坍塌的问题比原生 cGAN 还要严重。

这是由于调整训练轮次的策略仅仅是延缓了 D 的更新，看似是给予了 G 更多更新的机会，然而 D 的判断力下降同样延缓了 G 的更新，最终反而减慢了整个模型的训练速度。实际上，第 100 轮时的“模式坍塌”实际上更有可能是模式还未训练到位的表现。所以，这一策略的有效性仅体现在理论层面上，在实践中其有效性并不突出。

5. cDCGAN

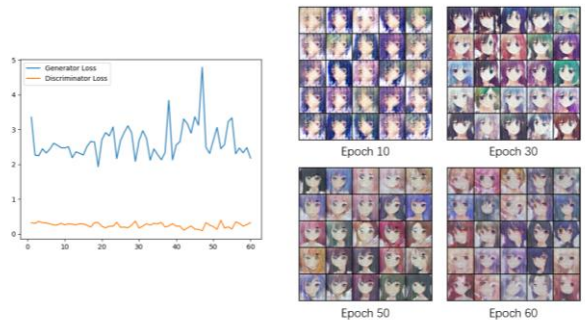

图 10 cDCGAN 实验结果

观察损失-迭代轮次图线可知，相较于 cGAN，cDCGAN 中 G 和 D 的损失一直在上下波动，并没有明显的上升或者下降的趋势。观察生成的图像可以看到，生成图像的质量一直在提高，且到第 30 轮时生成的图像相较于 cGAN 生成的图像清晰度有大幅度的提高，并且没有出现明显的噪点现象。在第 60 轮时，模型已经能够表

现出头发的纹理、人脸的大致五官、不同的瞳色等细节信息，而这些细节在 cGAN 训练中是难以被表现出来的。

这是由于反卷积层生成信息的能力本身就比全连接层强，因此反卷积层构建的生成器网络比普通的MLP的表现能力也好上很多。在这样的网络上进行训练，其色彩、细节、清晰度等方面的表现也会有很大的提升。

然而，cDCGAN 也有一些缺点。最显著的缺点在于cDCGAN 虽然参数量与原生 cGAN 相差不多，但由于卷积操作本身的特性，其计算量比 cGAN 高出不少，训练和推理耗时都比较长。也正是因为这个原因，对于cDCGAN 的训练轮次要少于 cGAN。

4. 结论

4.1 实验结论

通过上一节的实验结果和分析，我们可以得到以下结论。

1. 原生 GAN 的训练难度确实较大，非常容易导致模式坍塌等问题，而且在训练过程中的稳定性表现也不佳。

2. TTUR 能够上避免模式坍塌问题的出现，一定程度上提高生成图像的质量。但其作用受到网络结构的限制，并不一定能够提供图像质量上的明显提升。

3. 标签平滑化对 GAN 训练的提升是客观存在的，但也容易引发严重的模式坍塌问题，需要与 TTUR 等策略配合使用。

4. 调整训练轮次对 GAN 的训练的效果是有限的，在某些情况下它反而会使得训练减慢，因此这种策略需要谨慎使用。

5. 参数调优策略对于图像生成质量的影响都是有限的，影响图像生成质量的最重要的因素还是生成器网络的结构。

6. 将生成器和判别器的结构替换为 DCNN 后，生成图像的质量有了明显的提升，体现在色彩、清晰度、细节等方面。然而，DCNN 在训练时的算力消耗明显比MLP 高，这是一个不可忽视的因素。

需要注意，这些结论是通过实验得出的，对于其它数据集以及实验环境来说并不一定成立。但这些结论确实可以为搭建 cGAN 网络提供一定的思路，并指导实践中的训练过程。

4.2 实验方法分析

本实验中通过对于 cGAN 网络进行参数调优并比较相同条件下的训练效果，提出了一些供 cGAN 训练的参考技巧，具有一定的借鉴意义。同时，本实验尝试将cGAN 与 DCGAN 进行融合，提出了 cDCGAN 这一全新的网络结构，并取得了较好的训练效果。

然而，本实验的方法也存在一定的缺陷。首先，受到计算资源的限制，本实验中的训练轮次并不多，尤其是 cDCGAN，仅训练了 60 轮。如果继续对于 cDCGAN 进行训练，则有可能收获更好的训练效果。其次，由于训练时间有限，本实验中仅单独尝试了三种调参策略并将它们的效果进行对比，并没有尝试将这三种策略进行组合，进行消融实验等。此外本实验中也没有将调参策略与 cDCGAN 结合起来进行实验。

4.3 未来工作的探讨

针对上一节中提出的缺陷，未来本实验研究的课题还可以拓展的方面包括：

1. 延长训练轮次，观察 cGAN（包括 cDCGAN）训练中损失和图像质量的长久变化；

2. 还有很多参数调优的策略没有尝试，例如数据增强，替换损失函数等等。此外，参数调整的方向和幅度还可以进一步实验；

3. 网络结构还可以进一步优化，例如本文中提到的WGAN-GP，可以将其思想与 cDCGAN 融合，进一步优化 cGAN 网络的训练；再如本文中还提到了将自主意力机制引入 GAN 训练的 SAGAN。未来可以多加尝试这些优化结构的方案，发掘每种方案的适用范围以及实践效果。

5. 参考文献

[1] I. J. Goodfellow *et al.*, “Generative Adversarial Networks.” arXiv, Jun. 10, 2014. Accessed: Jan. 15, 2023. [Online]. Available: http://arxiv.org/abs/1406.2661

[2] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets.” arXiv, Nov. 06, 2014. Accessed: Jan. 15, 2023. [Online]. Available: http://arxiv.org/abs/1411.1784

[3] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN.” arXiv, Dec. 06, 2017. Accessed: Jan. 15, 2023. [Online]. Available: http://arxiv.org/abs/1701.07875

[4] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.” arXiv, Jan. 07,

2016. Accessed: Jan. 15, 2023. [Online]. Available: http://arxiv.org/abs/1511.06434
[5] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium.” arXiv, Jan. 12, 2018. Accessed: Jan. 16, 2023. [Online]. Available: http://arxiv.org/abs/1706.08500
[6] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-Attention Generative Adversarial Networks.” arXiv, Jun. 14, 2019. Accessed: Jan. 16, 2023. [Online]. Available: http://arxiv.org/abs/1805.08318
[7] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved Techniques for Training GANs.” arXiv, Jun. 10, 2016. Accessed: Jan. 16, 2023. [Online]. Available: http://arxiv.org/abs/1606.03498
[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, “Improved Training of Wasserstein GANs.” arXiv, Dec. 25, 2017. Accessed: Jan. 17, 2023. [Online]. Available: http://arxiv.org/abs/1704.00028