基于不同生成对抗网络的图像生成技术报告

2019 级计算机科学与技术 1 班
3019244266 李润泽

摘要

生成对抗网络(GAN) [1]自 2014 年被首次提出后，逐渐开始成为大热的研究方向，如今与 GAN 有关的算法以及应用扩展越来越多，例如图像生成、图像翻译等任务。

本文将生成对抗网络(GAN)以及它的变种(CGAN[2]、DCGAN[4]和 WGAN[5])进行实现，从而完成图像生成任务，并对 WGAN 进行参数调优，例如学习率以及生成器迭代中判别器的训练步长等参数，进而分析影响图像生成质量的因素。此外，在 WGAN 的基础上增加梯度惩罚机制，来尝试进一步提高图像生成的效果。从实验结果可以看出，随着学习率 lr 的上升，d_loss 值不断趋近于 0；随着裁剪参数 clip value[9]的下降，d_loss 值不断趋近于 0；随着训练步长 n_{critic} 的下降，d_loss 值不断趋近于 0。

本文成功实现四种生成对抗网络，并进行参数调优，此外，本文对 WGAN 增加了梯度惩罚机制，通过实验可以看出进行改进的 WGAN 图像生成的效果要优于原始的 WGAN。

关键词— GAN, WGAN, 参数调优, 模型改进

1. 引言

生成对抗网络(GAN)自 2014 年被首次提出后，逐渐开始成为大热的研究方向，如今与 GAN 有关的算法以及应用扩展越来越多，例如图像生成、图像翻译等任务。

GAN 属于生成模型的一种，其目的是根据给定的训练样本经过学习后生成与训练样本接近的样本。从直观上看，GAN 主要由生成器和判别器两个部分组成，它们作为对抗博弈的两方。在训练阶段，生成器接收一个随机噪声作为输入，输出生成的假样本，并使其尽可能接近真实样本；而判别器负责判断样本是真实样本还是生成器生成的假样本。生成器的能力越强，假样本越难以被判别器判别出来；判别器的能力越强，越容易判断出图片的真伪。

现如今，GAN 已经在很多不同的方面进行了改进，例如模型结构、目标函数以及应用方向等方面。其中，对于应用层面的改进，CGAN 在输入的时候加入了条件信息（类别标签或其他模态的信息），实现了有监督的 GAN；对于模型结构的改进，DCGAN 将卷积神经网络与 GAN 结合起来，在一定程度上解决了 GAN 难以训练的问题； 对于目标函数的改进，WGAN 使用了 Wasserstein 距离[10]作为衡量两个分布距离的指标。

但是，GAN 存在不收敛以及难以训练的问题，导致训练难以继续，此外，由于 GAN 无需预先建模，模型出现不可控的情况。

本文将生成对抗网络(GAN)以及它的变种(CGAN、DCGAN 和 WGAN)进行实现，从而完成图像生成任务，并对 WGAN 进行参数调优，例如学习率以及生成器迭代中判别器的训练步长等参数，进而分析影响图像生成质量的因素。此外，在 WGAN 的基础上增加梯度惩罚机制，来尝试进一步提高图像生成的效果。

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) 将 GAN、CGAN、DCGAN 与 WGAN 进行算法复现并进行图像生成任务；

2) 对 WGAN 进行参数调优，并分析影响图像生成质量的因素；

3) 对现有的 WGAN 增加梯度惩罚机制从而进行改进。

具体代码详见
https://github.com/Lrz266OuO/Deep_Learning-Neural_Network_Lab

2. 研究方法

2.1 Generative Adversarial Nets(GAN)

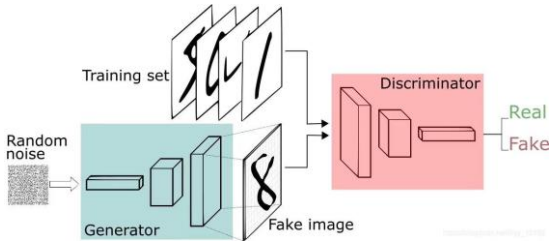
生成式对抗神经网络是一种近年来热度较高的深度学习模型，该模型主要由两个基础神经网络组成，分别是生成器神经网络(Generator Neural Network)和判别器神经网络(Discriminator Neural Network)。前者用于生成内容，而后者用户判别生成的内容。

GAN 将生成问题视作生成器与判别器两种网络的对抗。生成器从给定的图片[3]（称为噪声）中产生合成数据，试图产生更为真实的数据；而判别器分辨生成器的输出和真实数据，尝试更完美地分辨真实数据与生成

数据。由此，两种网络在对抗中共同进步，并在进步的过程中继续对抗，从而生成极为逼近真实的数据。

我们将生成数据的网络称为 G，将判别内容的网络称为 D，GAN 的整体结构如图 2-1 所示。

图 2-1 GAN 的整体结构



G 是一个生成图片的网络，它接收一个随机的假数据（噪声，记为 z），通过 z 生成图片，记为 G(z)。

D 是一个判别网络，它判别一张图片是不是“真实的”。判别网络的输入参数记为 x，即一张图片，输出值记为 D(x)，代表 x 为真实图片的概率，如果 D(x)=1，则判别网络认为这张图片一定是真实的图片；若 D(x)=0，则判别网络认为它不可能是真实的图片。

我们最终希望得到的结果是判别器的输出无限接近于 0.5，即判别器此时无法分辨出真假数据的差别，实现了生成网络与判别网络之间的纳什均衡。故我们可以训练出“造假一流”的生成模型。

GAN 的核心原理，即目标函数为

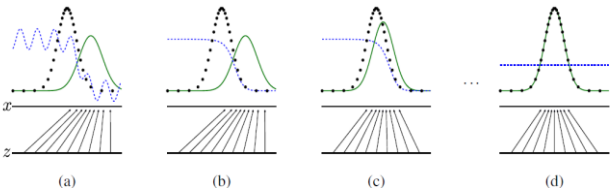
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

这里 V(D,G)相当于表示真实样本与生成样本的差异程度；$\max_D V(D,G)$表示尽可能让判别器最大化地判别出样本来自于真实数据还是生成的数据；$\min_G \max_D V(D,G)$表示在固定判别器 D 的条件下得到生成器 G，G 要求能够最小化真实样本和生成样本的差异。

通过上述的极大极小博弈过程，在理想状态下会收敛于生成分布拟合于真实分布。

GAN 的训练过程如图 2-2 所示。

图 2-2 GAN 的训练过程



其中，黑色虚线代表真实样本的概率；蓝色虚线代表判别器判别概率的数值分布情况；绿色实线代表生成样本的概率分布情况。

从图 2-2 我们可以看出，(a)为最初始的状态，生成器中生成的分布与真实分布区别较大，并且判别器判别出样本的概率不是很稳定；(b)是通过多次训练判别器所达到的状态，此时判别样本区分效果良好；(c)是通过多次训练生成器达到的样本状态，此时生成器分布逼近了真实样本分布；(d)是经过多次反复训练迭代之后最终的理论状态，生成样本分布你和与真实样本分布，并且判别器分辨不出样本是生成的还是真实的，即 D(G(z))=0.5，也就是说此时模型可以生成极为真实的样本了。

在实验中，我们设定训练的最大 epoch 为 200；生成网络与判别网络的优化算法为 Adam[8]算法，其中学习率 lr 为 0.0002，两个衰减速率 β_1 与 β_2 分别为 0.5 与 0.999；图片维度的大小设定为 28；生成网络使用 Batch Normlization[7]进行归一化，并使用 LeakyReLU[11]作为激活函数；判别网络使用 LeakyReLU 与 Sigmoid 作为激活函数；loss 值使用 BCELoss 二分类损失函数进行计算。

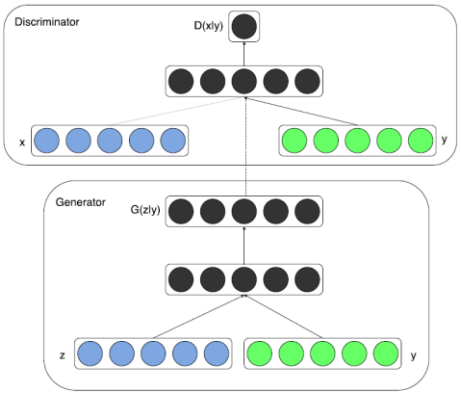
2.2 Conditional Generative Adversarial Nets(CGAN)

由于 GAN 这种不需要预先建模的方法过于自由，所以如果对于处理较大图片以及较多像素的情形，这种基于 GAN 的方法不太可控。

为解决上述问题，可以在 GAN 的基础上加入条件约束，即 Conditional Generative Adversarial Nets (CGAN)。在生成模型 G 和判别模型 D 中同时加入条件约束 y 来引导数据的生成过程。该条件可以是任何补充的信息，如类标签或其他模态的数据等，这样可以使得 GAN 能够更好地被应用于图像自动标注等跨模态问题。

CGAN 的整体结构如图 2-3 所示。

图 2-3 CGAN 的整体结构



CGAN 在生成模型和判别模型的建模中均引入了条件变量 y，这里的 y 是一种额外的条件变量，对于生成器对数据的生成具有指导作用。

在实验中，我们设定训练的最大 epoch 为 200；生成网络与判别网络的优化算法为 Adam 算法，其中学习率 lr 为 0.0002，两个衰减速率 β_1 与 β_2 分别为 0.5 与 0.999；图片维度的大小设定为 32；生成网络使用 Batch Normlization 进行归一化，并使用 LeakyReLU 作为激活

函数；判别网络使用 LeakyReLU 作为激活函数；loss 值使用 MSELoss 均方损失函数进行计算。

2.3 Deep Convolutional Generative Adversarial Nets(DCGAN)

DCGAN 将深度卷积神经网络(CNN)与生成对抗网络(GAN)进行结合，用于无监督学习领域。DCGAN 是对原始 GAN 的一种改进，在网络设计中采用了对 CNN[12]比较流行的改进方案：

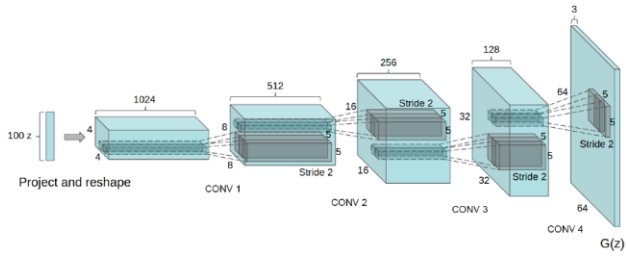
（1）将空间池化层用卷积层代替，这种替代只需要将卷积的步长 stride 设置为大于 1 的数值，其意义是下采样过程不再是固定的抛弃某些位置的像素值，而是可以让网络自己去学习下采样方式；

（2）将全连接层去除，全局均值池化有助于模型的稳定性但是降低了模型的收敛速度，我们可以将生成器输入的噪声 reshape 成 4D 的张量，来实现卷积的效果；

（3）采用 BN 层，Batch Normalization 是一种常用于卷积层后面的归一化方法，可以起到帮助网络收敛等作用。

DCGAN 的改进后的生成器结构如图 2-4 所示。

图 2-4 DCGAN 的生成器结构



与原始的 GAN 相比，DCGAN 进行的修改如下：

（1）使用指定步长的卷积层代替池化层；

（2）将 BN 这个归一化方法应用于生成器与判别器中；

（3）移除全连接层；

（4）在生成器中，除了输出层采用 Tanh 之外，全部使用 ReLU 作为激活函数；

（5）判别器的所有层都是用 LeakyReLU 作为激活函数。

在实验中，我们设定训练的最大 epoch 为 200；生成网络与判别网络的优化算法为 Adam 算法，其中学习率 lr 为 0.0002，两个衰减速率 β_1 与 β_2 分别为 0.5 与 0.999；图片维度的大小设定为 32；生成网络与卷积神经网络进行结合，使用 Batch Normlization 进行归一化，并使用 LeakyReLU 作为激活函数；判别网络使用 Batch Normlization 进行归一化，并使用 LeakyReLU 作为激活函数；loss 值使用 BCELoss 二分类损失函数进行计算。

2.4 Wasserstein Generative Adversarial Nets(WGAN)

原始的 GAN 实际上存在超参数敏感与模式崩塌[13]等缺陷。超参数敏感指的是网络的结构设定、学习率以及初始化状态等超参数对网络的训练过程影响较大，微量的超参数调整就可能导致训练结果截然不同。在 DCGAN 部分中，为了更好地训练网络，DCGAN 提出不使用 Pooling 层，而是 Batch Normalization 层；不使用全连接层；在生成网络中使用 ReLU 和 tanh 激活函数；在判别网络中使用 LeakyLeLU 激活函数。但实际上这些改变只能在一定程度上避免训练不稳定的现象。

而模式崩塌指的是模型生成的样本单一、多样性很差的现象。由于判别器只能鉴别单个样本的部分区间中的少量高质量样本，以此在判别器中获得较高的概率值，而不会学习到全部的真实分布。该现象在 GAN 中较为常见。

WGAN 的算法流程如图 2-5 所示。

图 2-5 WGAN 的算法流程

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

WGAN 主要从损失函数的角度上对 GAN 做了改进，这样可以使改进之后的 WGAN 即便在全连接层上也能得到很好的表现结果。WGAN 对 GAN 的改进主要有：判别器的最后一层去掉 sigmoid 函数；生成器和判别器的 loss 不取对数值，每次更新判别器的参数之后把它们的绝对值截断到不超过一个固定常数 c；不使用基于动量的优化算法（包括 momentum[14]和 Adam），可以使用 RMSProp 或 SGD。

GAN 中的交叉熵（即 JS 散度[15]）不适合衡量数据生成分布和真实数据分布的距离，如果通过优化 JS 散度训练 GAN 会导致找不到正确的优化目标。所以，WGAN 提出使用 Wassertein 距离作为优化方式训练 GAN。

Wasserstein 距离又叫 Earth-Mover（EM）距离，定义如下：

$$W(P_r, P_g) = \inf_{\gamma \sim \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

其中，$\Pi(P_r, P_g)$是P_r和P_g组合起来的所有可能的联合分布的集合，反过来说，$\Pi(P_r, P_g)$中每一个分布的边缘分布都是P_r和P_g。对于每一个可能的联合分布 γ 而言，可以从中采样$(x, y) \sim \gamma$得到一个真实样本 x 和一个生成样本 y，并算出这对样本的距离$\|x - y\|$，所以可以计算该联合分布 γ 下样本对距离的期望值$\mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$。在所有可能的联合分布中能够对这个期望值取到的下界$\inf_{\gamma \sim \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$，就定义为 Wasserstein 距离。

直观上可以把$\mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$理解为在 γ 这个“路径规划”下把$P_r$这堆“沙土”挪到$P_g$“位置”所需的“消耗”，而$W(P_r, P_g)$就是“最优路径规划”下的“最小消耗”，所以才叫 Earth-Mover 距离。

Wasserstein 距离相比 KL 散度[16]、JS 散度的优越性在于，即便两个分布没有重叠，Wasserstein 距离仍然能够反映它们的远近。WGAN 本作通过简单的例子展示了这一点。考虑如下二维空间中的两个分布 P_1 和 P_2，P_1 在线段 AB 上均匀分布，P_2 在线段 CD 上均匀分布，通过控制参数 θ 可以控制着两个分布的距离远近。

在实验中，我们设定训练的最大 epoch 为 200；生成网络与判别网络的优化算法为 RMSprop 算法，其中学习率 lr 为 0.00005；图片维度的大小设定为 28；每个生成器迭代中判别器的训练步长 n_{critic} 为 5；每次更新的值限定在常数 c=0.01 之内；生成网络使用 Batch Normlization 进行归一化，并使用 LeakyReLU 与 Tanh 作为激活函数；判别网络使用 LeakyReLU 作为激活函数。

3. 实验与结果分析

3.1 实验数据集：MNIST

MNIST 数据集是一种手写数字图像集，是机器学习领域最为经典的数据集之一，它被用于各种图像训练与生成的任务中。MNIST 数据集由 0-9 的数字图像构成，训练图像有 6 万张，测试图像有 1 万张，可以用于学习和推理中。

MNIST 的训练集(training set)由来自 250 个不同人手写的数字构成，其中 50%是高中学生，50%来自人口普查局的工作人员；而测试集(test set)也是同样比例的手写数字数据。

MNIST 数据集的一般使用方法是：先用训练图像进行学习，再用学习到的模型对测试图像进行正确的分类。针对于该实验，生成对抗网络在收到噪声进行图像生成后与数据集混合进行测试，检查生成网络是否能生成足以无法让判别网络进行正确判别的图像。

在本实验中，所有对抗生成网络均使用 MNIST 数据集进行测试。

3.2 实验平台：PyTorch

PyTorch 是一个针对深度学习，并且使用 GPU 和 CPU 来进行优化的张量库。PyTorch 这一 Python 科学计算框架有以下两种目的：首先是可以替换 NumPy，并通过利用 GPU 的算力来实现神经网络的加速；其次是可以通过自动微分机制，得以让神经网络的实现变得更加容易。其中，张量(Tensor)是一种特殊的数据结构，在 PyTorch 中，神经网络的输入、输出以及网络的参数等数据，都是使用张量来进行描述。

在本实验中，所有算法均基于 PyTorch 进行实现。

3.3 算法实现

通过对上述 4 种不同生成对抗网络的实现并在 PyTorch 平台运行可以得到训练图像如图 3-1 所示。

图 3-1 不同生成对抗网络在训练不同阶段产生的图像

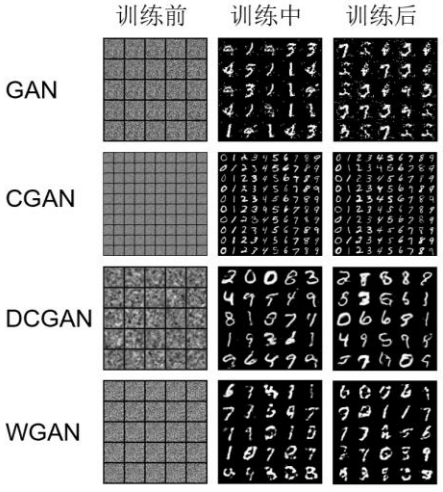
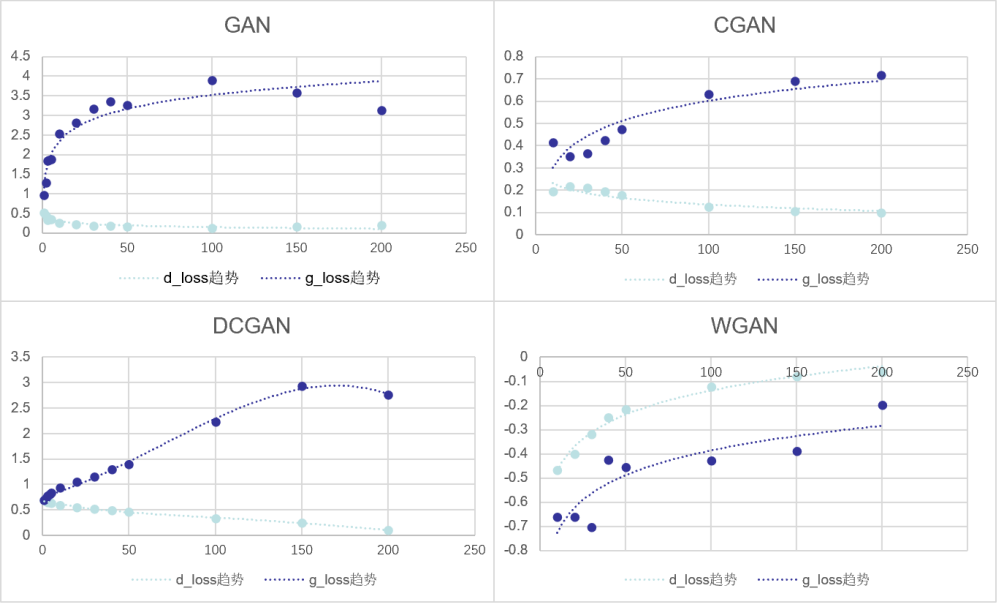


表 3-1 不同生成对抗网络在训练不同阶段时计算出的 d_loss 值与 g_loss 值

模型	epoch	1	5	10	50	100	150	200
GAN	d_loss	0.510469	0.342887	0.248609	0.170712	0.120419	0.169989	0.192855
	g_loss	0.956933	1.874407	2.537149	3.263798	3.905378	3.570554	3.132934
CGAN	d_loss	0.106481	0.135711	0.194618	0.17816	0.124881	0.106104	0.097463
	g_loss	0.660399	0.585134	0.415263	0.474954	0.632873	0.689216	0.718078
DCGAN	d_loss	0.692203	0.631018	0.588311	0.463967	0.334903	0.242517	0.103577
	g_loss	0.694127	0.829934	0.936831	1.402277	2.230488	2.936166	2.757655
WGAN	d_loss	-0.291029	-0.329138	-0.468464	-0.214993	-0.123098	-0.080621	-0.060724
	g_loss	-12.6155	-0.209976	-0.662237	-0.456756	-0.427529	-0.388925	-0.197952

图 3-2 不同生成对抗网络训练过程中的 d_loss 与 g_loss 趋势曲线



在图 3-1 中，从行来看，第一排表示 GAN 训练的图像，第二排表示 CGAN 训练的图像，第三排表示 DCGAN 训练的图像，第四排表示 WGAN 训练的图像。

从列来看，第一列表示输入的噪声，第二列表示训练过程中的图像（每个 images 文件夹中的 100000.png 文件），第三列表示在训练 200 个 epoch 后的图像。

不同的生成对抗网络在不同训练时期计算得到的 loss 值如表 3-1 与图 3-2 所示。

可以看到，每种生成对抗网络在训练到一定时期，均会保持在一定的数值，基本保持不变。

3.4 对比实验

关于对比实验，本文将 WGAN 进行对比实验，在算法（图 2-5）中，选取不同学习率 lr、每次更新的限定值 c 与生成器迭代中判别器的训练步长 n_{critic} 作为指标进行对比实验。

在原始的 WGAN 中，lr=0.00005，c=0.01，n_{critic}=5。本文仅对 WGAN 分别进行单个参数的调整，具体的调整如表 3-2 所示。

表 3-2 对比实验中参数的调整值

WGAN	learning rate	clipping value	n_critic
未调参	0.00005	0.010	5
已调参	0.00003	0.010	5
	0.00007		
	0.00010		
	0.00005	0.005	5
		0.015	
	0.00005	0.010	3
			7
			10

通过参数的调整，每次实验得到的每一迭代的 d_loss 与 g_loss 值如表 3-2 所示。

对于同一个参数而言，d_loss 与 g_loss 值的趋势曲线如图 3-3 所示。

在进行对比实验之后，生成的不同阶段图像的对比如图 3-4（**文末**）所示。

从图表可以看出，随着学习率 lr 的上升，d_loss 值不断趋近于 0；随着裁剪参数 clip value 的下降，d_loss 值不断趋近于 0；随着训练步长 n_{critic} 的下降，d_loss 值不断趋近于 0。

图 3-3 WGAN 不同对比实验的 d_loss 与 g_loss 值趋势曲线

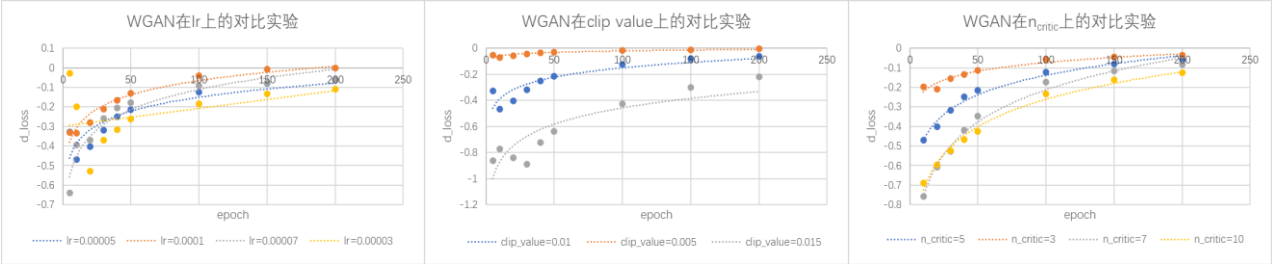


表 3-2 进行不同参数调整后 WGAN 在训练不同阶段时计算出的 d_loss 值与 g_loss 值
其中，“未调参数”指的是该数据使用的算法及其参数与论文相同(lr=0.00005, c=0.01, n_{critic}=5)，
其他数据仅发生了一个参数的调整

WGAN	epoch	1	5	10	50	100	150	200
未调参数	d_loss	0.510469	0.342887	0.248609	0.170712	0.120419	0.169989	0.192855
	g_loss	0.956933	1.874407	2.537149	3.263798	3.905378	3.570554	3.132934
lr=0.00003	d_loss	-1.324088	-0.029177	-0.19869	-0.26232	-0.184315	-0.132002	-0.10799
	g_loss	-16.281578	-2.192851	-0.176058	-0.529895	-0.318548	-0.275052	-0.280452
lr=0.00007	d_loss	-0.139081	-0.639934	-0.393988	-0.178239	-0.092003	-0.076987	-0.063076
	g_loss	-10.967121	-1.407782	-2.590253	-3.496062	-2.997306	-1.995171	-1.888707
lr=0.0001	d_loss	-1.604285	-0.329401	-0.333293	-0.12884	-0.039707	-0.006389	-0.0024
	g_loss	-15.343741	-1.025346	-0.521675	-0.354607	-0.026305	0.07179	-0.002204
c=0.005	d_loss	-0.040514	-0.051155	-0.070213	-0.030586	-0.016997	-0.01095	-0.005632
	g_loss	-1.686385	-0.065633	-0.115359	-0.038974	-0.034914	-0.026053	0.000908
c=0.015	d_loss	-0.638625	-0.860805	-0.769807	-0.638455	-0.427418	-0.302225	-0.221161
	g_loss	-44.014339	-1.568025	-3.538378	-1.534467	-0.791353	-1.165717	-0.927949
n_{critic}=3	d_loss	-0.074927	-0.05581	-0.197879	-0.112377	-0.056622	-0.044831	-0.03488
	g_loss	-9.313015	-1.317773	-0.411108	-0.244945	-0.341394	-0.118015	0.020226
n_{critic}=7	d_loss	-0.761776	-0.141849	-0.758091	-0.348	-0.173833	-0.116963	-0.083538
	g_loss	-14.145625	-1.877592	-0.981328	-0.862451	-3.758328	-3.239967	-3.069723
n_{critic}=10	d_loss	-1.933298	-0.037985	-0.688947	-0.426126	-0.234467	-0.160199	-0.125829
	g_loss	-16.400726	-11.756281	-2.850781	-0.710306	-0.388319	-0.390491	-0.404861

表 4-1 改进前后的 WGAN 对比

WGAN	epoch	1	5	10	50	100	150	200
未改进	d_loss	0.510469	0.342887	0.248609	0.170712	0.120419	0.169989	0.192855
	g_loss	0.956933	1.874407	2.537149	3.263798	3.905378	3.570554	3.132934
增加 gradient penalty	d_loss	-0.246925	-7.846449	-6.713724	-3.10867	-2.067324	-1.634501	-1.393292
	g_loss	-14.636739	0.762477	1.745449	-2.624076	-2.965248	-2.781996	-2.585317

图 4-2 引入 gradient penalty 后的 WGAN 算法

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

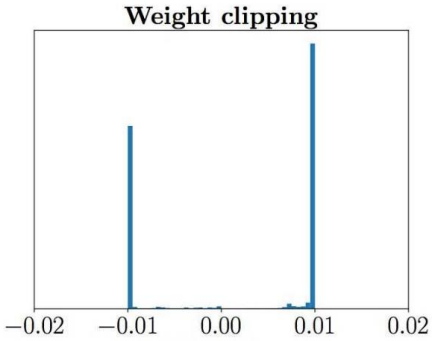
Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

4. 对 WGAN 的改进方法

实际上，WGAN 在处理 Lipschitz 限制条件时直接采用了 weight clipping，即每当更新一次判别器的参数之后，就会检查判别器的所有参数的绝对值是否超过该阈值（例如 c=0.01），会保持在[-0.01, 0.01]范围内。在训练的过程中，判别器 loss 希望尽可能扩大真假样本的差别，但是，weight clipping 独立地限制了每一个网格参数的取值范围，在这种情况下，最优的策略就是尽可能让参数趋于极端值，即取最大值(0.01)或最小值(-0.01)，如图 4-1 所示。

图 4-1 WGAN 在训练中参数分布的位置



此外，这种对参数的剪切容易导致梯度消失（即权重得不到更新信息）与梯度爆炸（即权重每次更新都产生了很大的变化），造成的原因是 c 的选择，过小会导致梯度消失，而过大会导致梯度爆炸。

为解决上述问题，可以使用梯度惩罚的方式满足上述 Lipschitz 条件。梯度惩罚机制是需要建立一个损失函数 d(D(x))，然后建立与梯度限制之间的二范数即可实现简单的损失函数。对于图片这种数据集，可以对每一批次的样本中采样即可。目标函数如图 4-3 所示。

图 4-3 引入 gradient penalty[6]后 WGAN 的目标函数

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}.$$

此外，由于模型对每个样本独立施加梯度惩罚，故判别器的模型不能使用 Batch Normalization，因为它会引入同一个 batch 中不同样本的相互依赖关系。

图 4-2 表示的是加入 gradient penalty 后的的算法。

其中 gradient penalty 的选取仅仅是在真假分布之间进行抽样处理，处理过程如图 4-4 所示。

加入梯度惩罚机制后，参数的分布如图 4-5 所示。

表 4-2 改进前后的 WGAN 的 loss 值对比

WGAN	epoch	1	5	10	50	100	150	200
未改进	d_loss	-0.291029	-0.329138	-0.468464	-0.214993	-0.123098	-0.080621	-0.060724
	g_loss	-12.6155	-0.209976	-0.662237	-0.456756	-0.427529	-0.388925	-0.197952
增加 gradient penalty	d_loss	-0.246925	-7.846449	-6.713724	-3.10867	-2.067324	-1.634501	-1.393292
	g_loss	-14.636739	0.762477	1.745449	-2.624076	-2.965248	-2.781996	-2.585317

图 4-4 gradient penalty 的处理过程

$$W(P_{data}, P_G) \approx \max_D \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)] - \lambda E_{x\sim P_{penalty}}[max(0, \|\nabla_x D(x)\| - 1)]\}$$

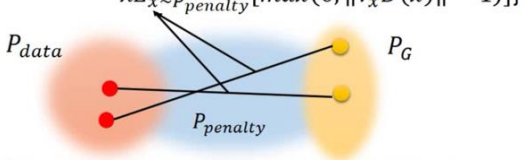


图 4-5 加入 Gradient penalty 后参数的分布

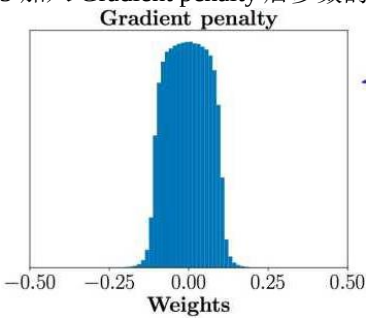
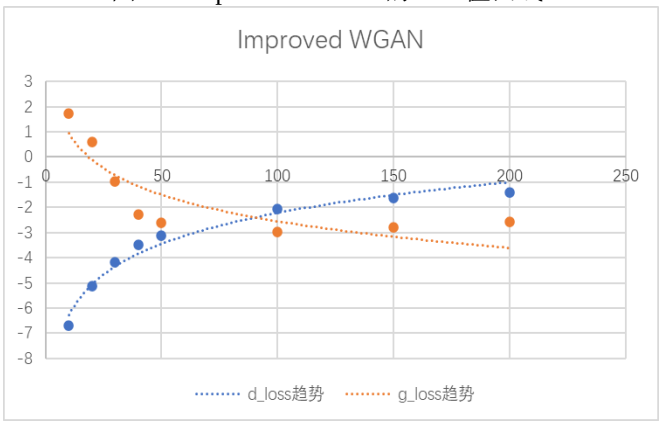
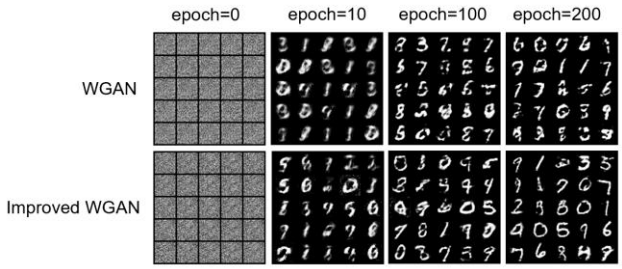


图 4-6 Improved WGAN 的 loss 值曲线



在实验中，我们设定训练的最大 epoch 为 200；生成网络与判别网络的优化算法为 Adam 算法，其中学习率 lr 为 0.0001, 两个衰减速率 β_1 与 β_2 分别为 0 与 0.9；图片维度的大小设定为 28；每个生成器迭代中判别器的训练步长 n_{critic} 为 5；每次更新的值限定在常数 c=0.01 之内；梯度惩罚的权重为 10；生成网络使用 Batch Normlization 进行归一化，并使用 LeakyReLU 作为激活函数；判别网络使用 LeakyReLU 作为激活函数。

图 4-7 改进前后 WGAN 生成图像对比



通过实验可以得到不同 epoch 的 d_loss 与 g_loss 值的趋势曲线，如表 4-2，图 4-6 所示。生成的图像如图 4-7 所示。

通过图表可以看出，进行改进的 WGAN 生成的图像优于改进前的效果。

5. 结论

本文实现了四种生成对抗网络（GAN、CGAN、DCGAN 与 WGAN）并完成了图像生成任务。根据 WGAN 的原理进行了不同参数的调优并进行对比实验。随着学习率 lr 的上升，d_loss 值不断趋近于 0；随着裁剪参数 clip value 的下降，d_loss 值不断趋近于 0；随着训练步长 n_{critic} 的下降，d_loss 值不断趋近于 0。此外，本文还根据 WGAN 的缺陷进行相应的改进，即增加了梯度惩罚机制，实现了生成对抗网络的优化，提升了图像生成的质量。

但实际上，根据图表可知，这几种生成对抗网络还是没有实现足以“以假乱真”的地步，即实现 D(G(z))=0.5。且 GAN 是一种比较难训练的模型，需要生成网络与判别网络实现很好的同步。未来需要尝试增加一种更好的机制来实现 D 与 G 同步的问题。

6. 参考文献

[1] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[J]. Advances in neural information processing systems, 2014, 27.

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

[3] Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks[J]. arXiv preprint arXiv:1511.06434, 2015.

[4] Reed S, Akata Z, Yan X, et al. Generative adversarial text to image synthesis[C]//International Conference on Machine Learning. PMLR, 2016: 1060-1069.

[5] Arjovsky M, Chintala S, Bottou L. Wasserstein generative adversarial networks[C]//International conference on machine learning. PMLR, 2017: 214-223.

[6] Gulrajani I, Ahmed F, Arjovsky M, et al. Improved training of wasserstein gans[J]. arXiv preprint arXiv:1704.00028, 2017.

[7] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International conference on machine learning. PMLR, 2015: 448-456.

[8] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[9] Zhang J, He T, Sra S, et al. Why gradient clipping accelerates training: A theoretical justification for adaptivity[J]. arXiv preprint arXiv:1905.11881, 2019.

[10] Weng L. From gan to wgan[J]. arXiv preprint arXiv:1904.08994, 2019.

[11] Maas, Andrew L.. "Rectifier Nonlinearities Improve Neural Network Acoustic Models." Paper presented at the meeting of the , 2013.

[12] O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.

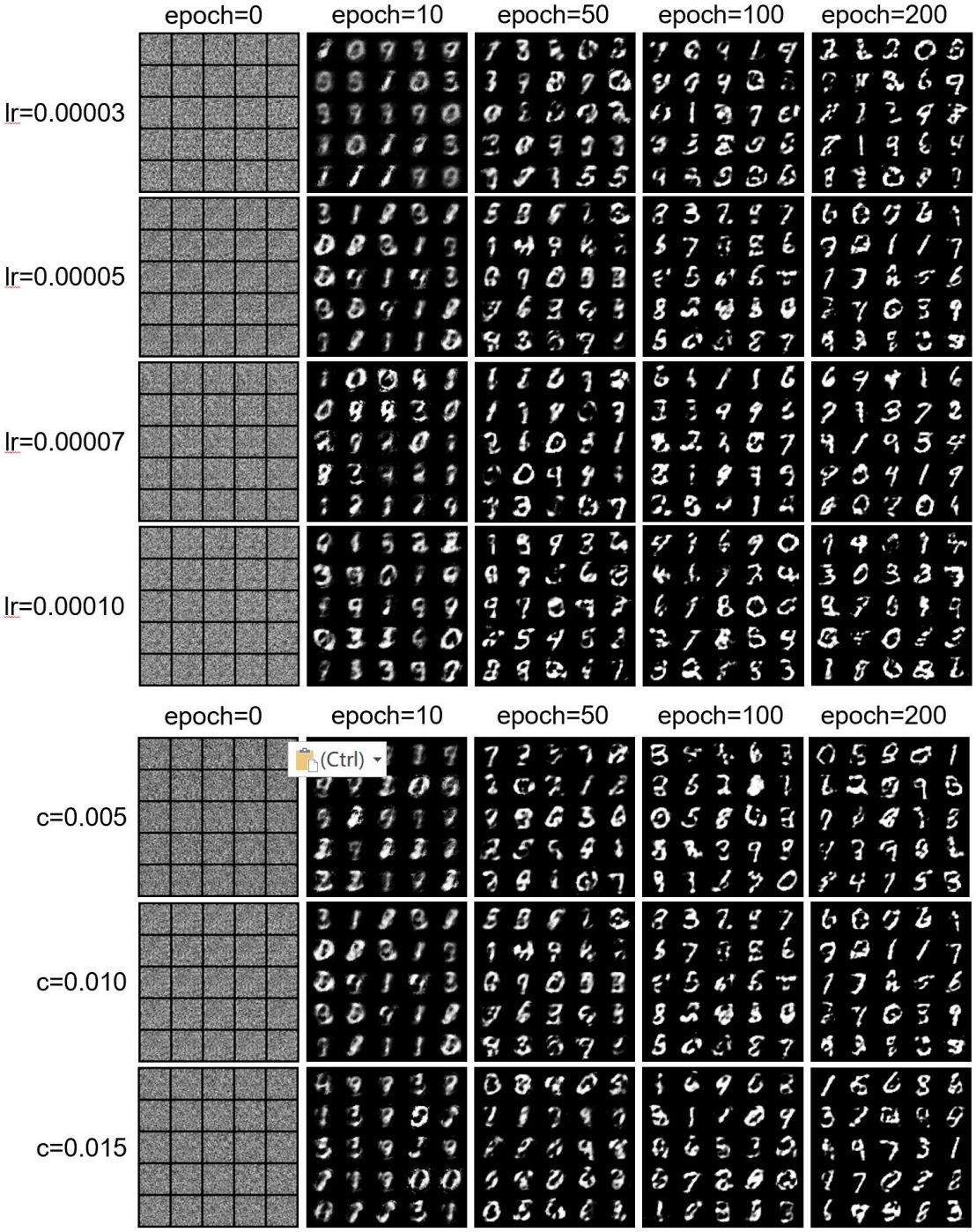
[13] Kodali N, Abernethy J, Hays J, et al. On convergence and stability of gans[J]. arXiv preprint arXiv:1705.07215, 2017.

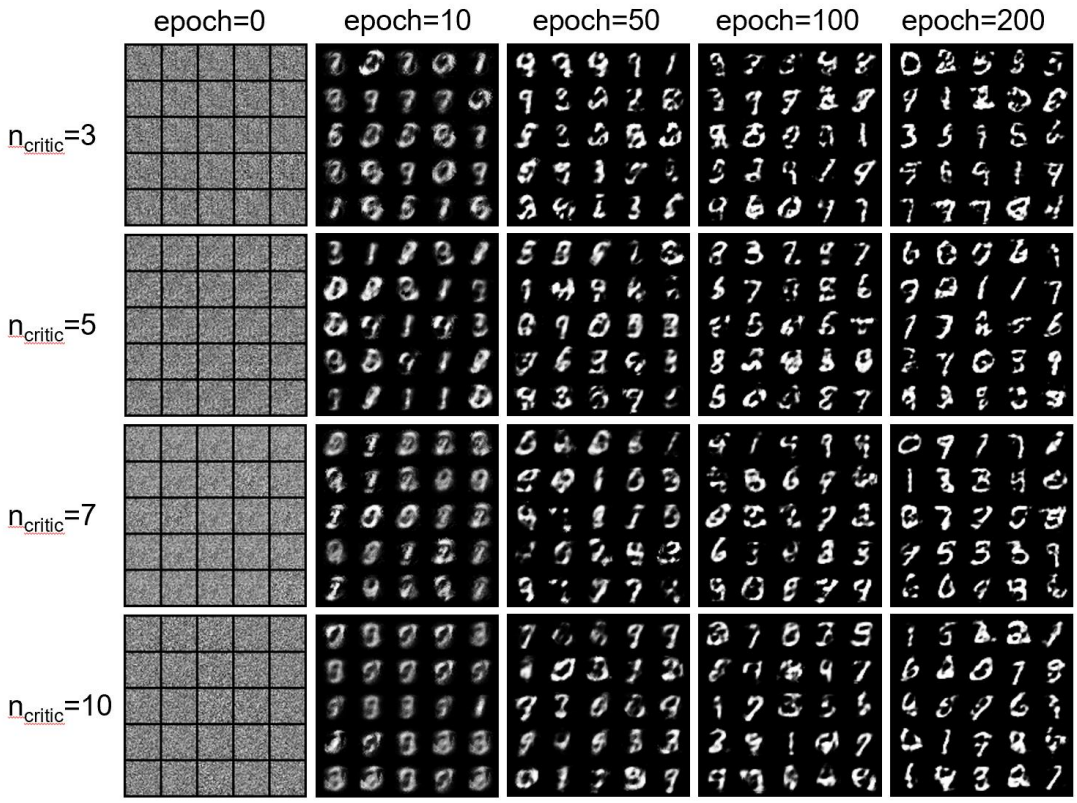
[14] He K, Fan H, Wu Y, et al. Momentum contrast for unsupervised visual representation learning[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 9729-9738.

[15] Naghshvar M, Javidi T, Wigger M. Extrinsic Jensen–Shannon divergence: Applications to variable-length coding[J]. IEEE Transactions on Information Theory, 2015, 61(4): 2148-2164.

[16] Hershey, John R. and Peder A. Olsen. “Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models.” *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07* 4 (2007): IV-317-IV-320.

图 3-4 WGAN 进行单个调参的图像生成对比




epoch=0
epoch=10
epoch=50
epoch=100
epoch=200
n_critic=3
n_critic=5
n_critic=7
n_critic=10