GAN 实验

数据集介绍

本次实验使用的是 MNIST 数据集，它包含了 60000 张图片作为训练数据，10000 张图片作为测试数据。训练图像一共 60000 张，供研究人员训练出合适的模型，测试图像一共 10000 张，供研究人员测试训练的模型的性能。

MNIST 数据集主要由一些手写数字的图片和相应的标签组成，图片一共 10 类，分别对应 0~9 共 10 个阿拉伯数字。在 MNIST 数据集中的每一张图片都代表了 0~9 中的一个数字。

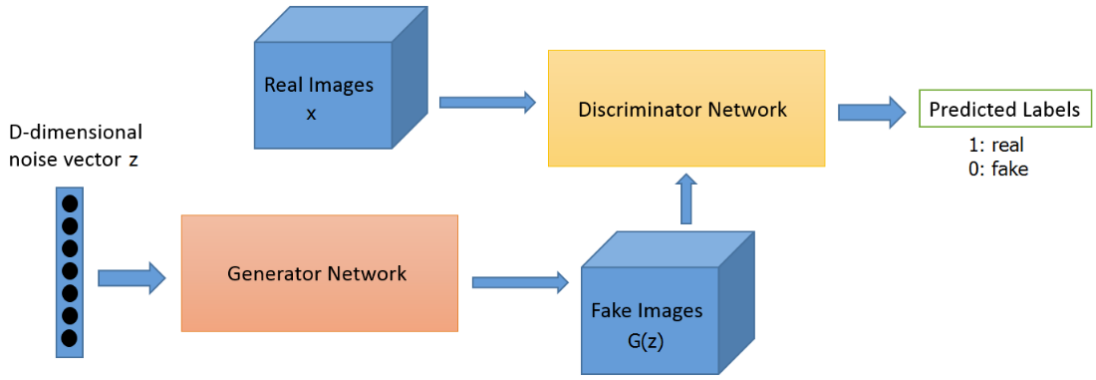
每张图片都由一个 28×28 的矩阵表示，每张图片都由一个 784 维的向量表示（28 x 28 = 784）。图片的大小都为 28 x 28，且数字都出现在图片的正中间。处理后的每一张图片是一个长度为 784 的一维数组, 这个数组中的元素对应了图片像素矩阵中的每一个数字。

实现细节

GAN 的思想：

生成器和鉴别器两个网络彼此博弈。生成器的目标是生成一个对象，并使其看起来和真的一样, 而鉴别器的目标就是找到生成出的结果和真实图像之间的差异。生成器和辨别器需要不断优化，各自提高自己的生成能力和判别能力，这个学习优化过程就是寻找二者之间的一个纳什均衡。

GAN 的结构：



用可微分函数 D 和 G 来分别表示判别器和生成器，它们的输入分别为真实数据 x 和随机变量 z。G(z)为由 G 生成的尽量服从真实数据分布的样本。如果判别器的输入来自真实数据，标注为 1。如果输入样本为 G(z)，标注为 0。

D 的目标是实现对数据来源的二分类判别：真（来源于真实数据 x 的分布）或者假（来源于生成器的伪数据 G(z)），而 G 的目标是使自己生成的伪数据 G(z)在 D 上的表现 D(G(z))和真实数据 x 在 D 上的表现 D(x)一致。

GAN 实现：

定义输入矩阵的占位符，输入层单元为 784，占位符的数值类型为 32 位浮点型；定义判别器的权重矩阵和偏置项向量，判别网络为三层全连接网络；定义生成器的输入噪声为 100 维度的向量组，输入层为 100 个神经元且接受随机噪声，输出层为 784 个神经元，并输出手写字体图片。生成网络根据原论文为三层全连接网络。

定义一个可以生成 m*n 阶随机矩阵的函数，该矩阵的元素服从均匀分布，随机生成的矩阵就为生成器的输入 z。

定义生成器，第一层先计算 y = z * G_W1 + G_b1，然后投入激活函数计算 G_h1 = ReLU(y)，G_h1 为第二次层神经网络的输出激活值，接着计算第二层传

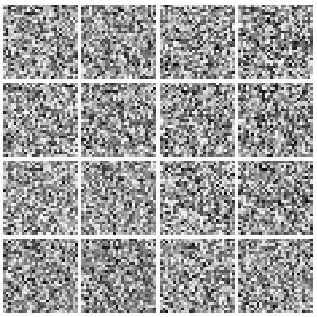
播到第三层的激活结果，第三层的激活结果是含有 784 个元素的向量，该向量转化 28 × 28 就可以表示图像。

定义辨别器，计算 D_h1 = ReLU(x * D_W1 + D_b1)，该层的输入为含 784 个元素的向量，再计算第三层的输出结果。因为使用的是 Sigmoid 函数，则该输出结果是一个取值为[0,1]间的标量，即判别输入的图像到底是真（= 1）还是假（= 0）。最后返回判别为真的概率和第三层的输入值，输出 D_logit 是为了将其输入 tf.nn.sigmoid_cross_entropy_with_logits()以构建损失函数。

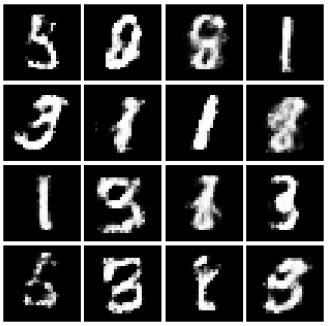
本实验使用交叉熵作为判别器和生成器的损失函数, 判别器和生成器的优化方法为 Adam 算法。本实验共迭代训练 30W 次，每 2000 次输出一张生成器生成的图片，每 500 次输出迭代数、生成器损失和判别器损失。

比较方法

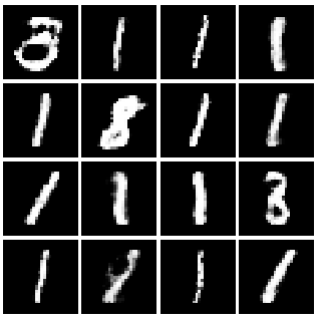
第 0 次迭代后，生成器损失为 2.608，辨别器损失为 1.123，图像为



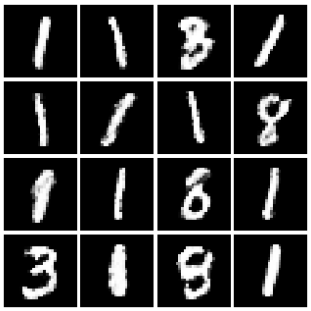
第 60000 次迭代后，生成器损失为 2.443，辨别器损失为 0.6068，图像为



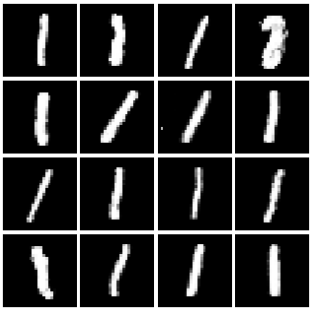
第 120000 次迭代后，生成器损失为 2.522，辨别器损失为 0.3868，图像为



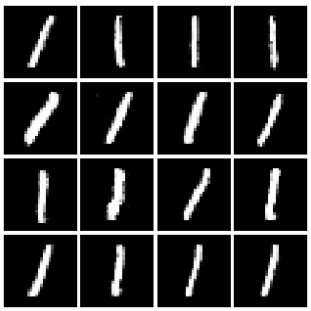
第 180000 次迭代后，生成器损失为 2.556，辨别器损失为 0.3377，图像为



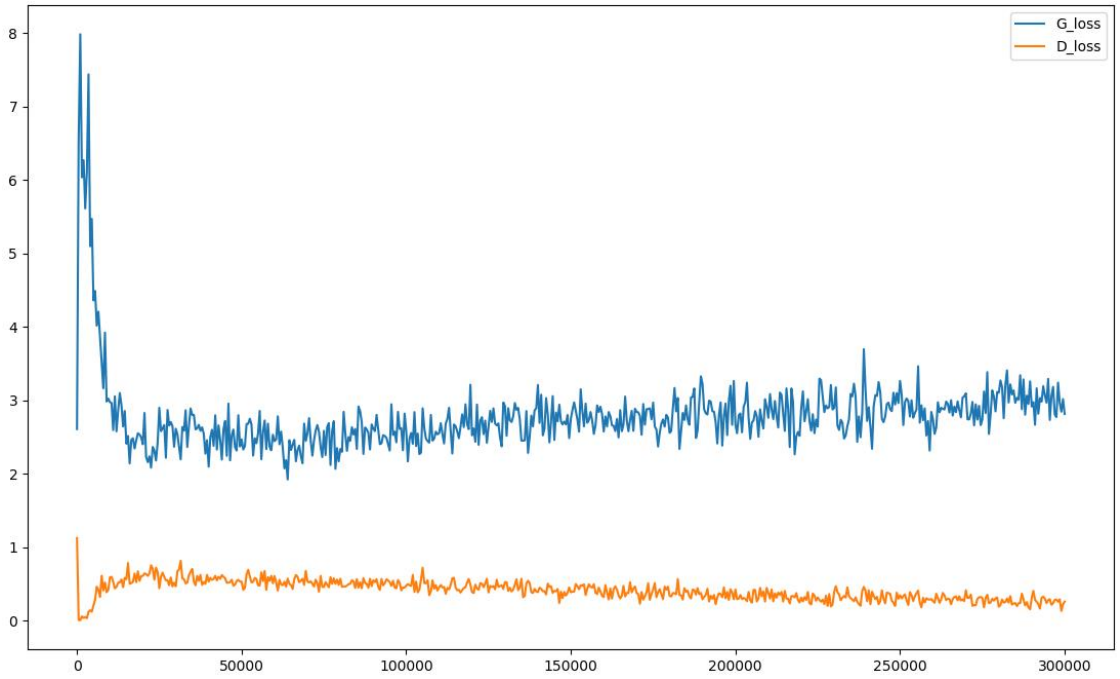
第 240000 次迭代后，生成器损失为 2.718，辨别器损失为 0.2638，图像为



第 300000 次迭代后，生成器损失为 2.813，辨别器损失为 0.2582，图像为



整个训练过程生成器和辨别器损失的变化情况如图：



由图可以看出生成器和辨别器损失趋于稳定，即生成器和辨别器处于纳什平衡状态。

评价标准

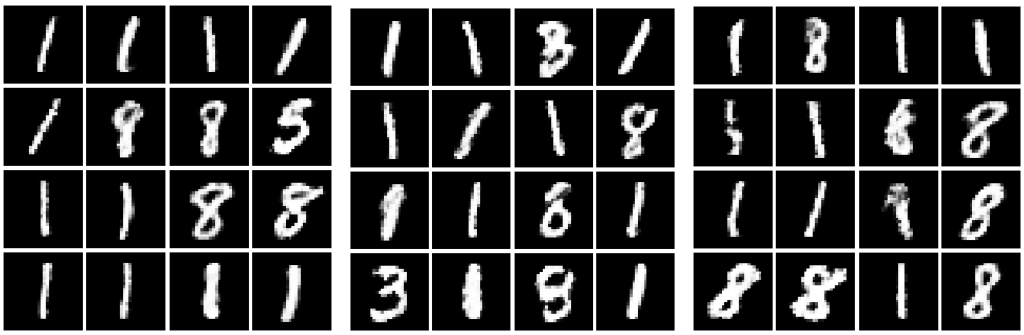
论文给出的方法是高斯 parzen 窗法进行密度估计，先用真实样本给出高斯 parzen 概率密度函数，再计算虚假样本在这个分布中的密度，密度越大表示越接近真实值。

结果

下表为在 MNIST 和 TFD 两个数据集上的实验结果。表格中每一格左侧的是需要的指标，右侧是通过交叉验证求得的高斯 parzen 窗计算时需要的参数。

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [6]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

生成图像的效果展示：



GAN 的优势：

（1） 根据实际的结果，看上去产生了更好的样本；

（2） GAN 能训练任何一种生成器网络；

（3） GAN 不需要设计遵循任何种类的因式分解的模型，任何生成器网络和任何鉴别器都会有用；

（4） GAN 无需利用马尔科夫链反复采样，无需在学习过程中进行推断，回避了近似计算棘手的概率的难题。