基于 MNIST 数据集的条件生成对抗网络训练

3170105582 电子科学与技术 王若鹏

1. 实验要求

- 建立条件生成对抗网络，调节参数并尽可能将其调到最佳状态。
- 绘制深度神经网络模型图、绘制并分析学习曲线。
- 在训练过程中生成 MNIST 随机手写数字图像，并进行对比。

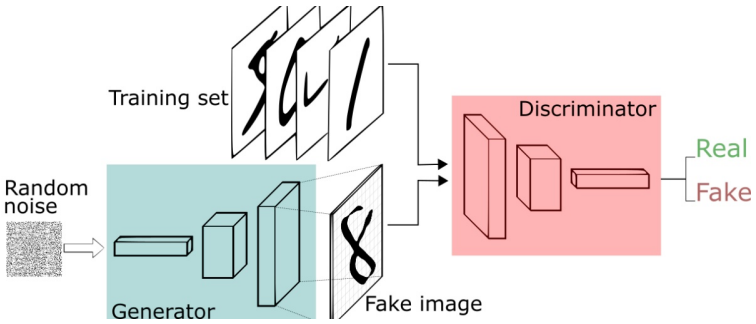
2. 实验原理

2.1 生成对抗网络 GAN

生成对抗网络（GAN）是由一个生成器 generator 和一个判别器 discriminator 组成。GAN 的核心是通过生成器和判别器两个神经网络之间的竞争对抗，不断提升彼此水平以使得生成器所生成数据（人工伪造数据）与真实数据相似，使判别器无法区分真实数据和生成数据。

在生成对抗网络中，生成网络 G 的输入 z 来自预定义的噪声分布 p(z)，输出 G(z)属于样本空间。判别网络 D 被用来判断哪些数据来自真实数据 x 以及大写数据是合成数据，输出 D(x)表示 x 来自真实数据分布 p_data(x)的概率。GAN 训练的过程就是判别网络 D 和生成网络 G 分别尝试最大化和最小化如下价值函数 V(D,G)：

$$\min_G \max_D V_{GAN}(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

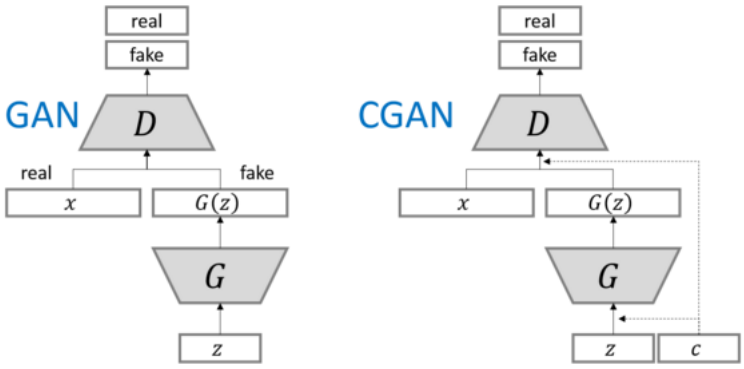


（图 1：GAN 的原理）

2.2 条件生成对抗网络 CGAN

在生成数据中，往往不仅要求所生成数据逼真，还会要求其符合一定的条件。如下图所示的条件生成对抗网络 CGAN 结构中，将条件约束 c 分别输入生成器 G 和判别器 D，生成器 G(z|c)会根据条件约束 c 和随机噪声 z 一同生成数据，判别器 D 需要判断数据是否是满足该条件的真实样本。

这样生成的数据不仅能够更接近真实数据，还能通过调节约束，获得符合特定要求的生成数据。例如，在手写数字的生成中，原始的 GAN 之内随机产生某个数的手写图像，而 CGAN 可以根据要求，产生某个特定数字的手写图像。



（图 2：GAN 与 CGAN 的区别）

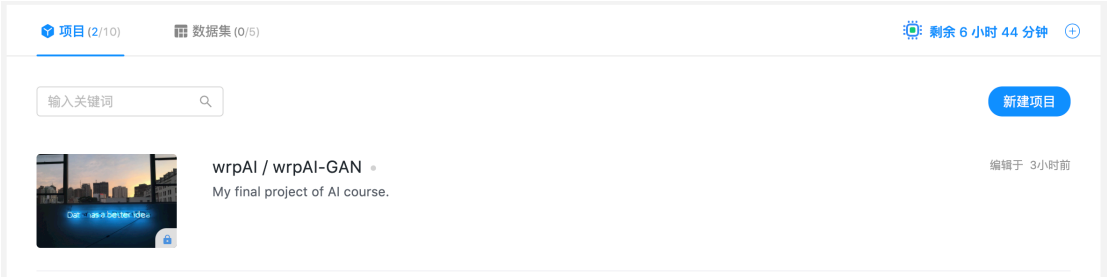
3. 开发环境

- 编程语言：Python 3.5
- 训练框架：Keras、TensorFlow
- 开发平台：Mo 人工智能训练平台
- 操作系统：MacOS Mojave 10.14.6

4. 实验内容与算法设计

4.1 创建项目

在 Mo 平台新建一个项目，如下所示：

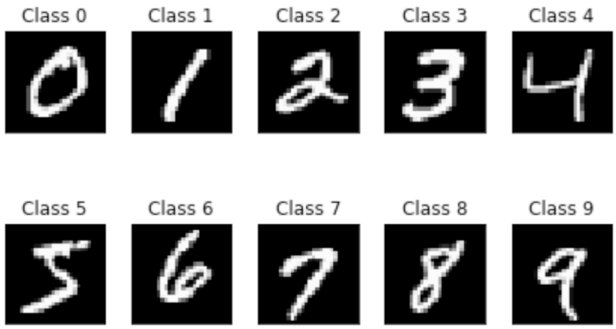


（图 3：新建项目演示）

4.2 导入数据集

MNIST 数据集来自美国国家标准与技术研究所 (NIST)。数据集由来自 250 个不同人手写的数字构成，其中训练集图片/标签有 60000 个样本，测试集图片/标签有 10000 个样本，是机器学习领域最为常用的数据集之一。采用以下方式导入数据集：

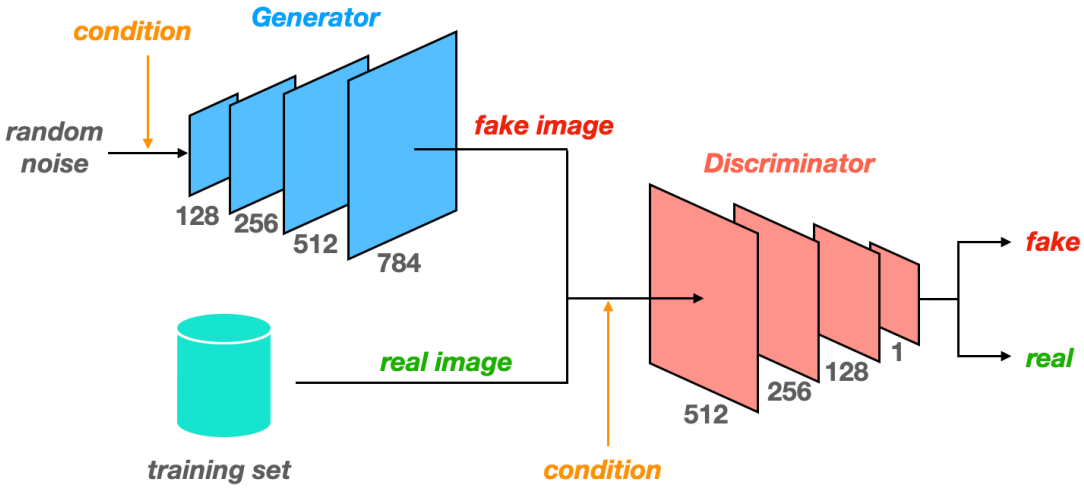
```
from keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
```



（图 4：MNIST 数据集展示）

4.3 创建生成网络与判别网络

构建生成器 generator 和判别器 discriminator，模型参数如下所示：首先初始化为一个随机图像，生成器的参数为 128-256-512-784，其中 784 为生成图像的总像素数(28x28)；判别器的参数为 512-256-128-1，其中 1 为判定结果 true/false。对于条件 condition，将其添加至随机噪声生成后和判别器的输入端，将模型结构绘制如下：



（图 5：创建的 CGAN 模型结构示意图）

使用 Keras 的 Sequential 结构实现模型。全连接层之间使用带泄露线性整流函数 Leaky ReLU 作为激活函数。在创建好生成器 generator 和判别器 discriminator 后，要将条件嵌入至网络中，为每个图像设定标签 label，该部分代码如下所示：

```
# Generator network
generator = Sequential()
generator.add(Dense(128, input_shape=(latent_dim,), kernel_initializer=init))
generator.add(LeakyReLU(alpha=0.2))
generator.add(BatchNormalization(momentum=0.8))
generator.add(Dense(256))
generator.add(LeakyReLU(alpha=0.2))
generator.add(BatchNormalization(momentum=0.8))
generator.add(Dense(512))
generator.add(LeakyReLU(alpha=0.2))
generator.add(BatchNormalization(momentum=0.8))
generator.add(Dense(784, activation='tanh'))

# Embedding condition into G
num_classes = 10
label = Input(shape=(1,), dtype='int32')
label_embedding = Embedding(num_classes, latent_dim)(label)
label_embedding = Flatten()(label_embedding)
z = Input(shape=(latent_dim,))
input_generator = multiply([z, label_embedding])
img = generator(input_generator)
generator = Model([z, label], img)
```

```
# Discriminator network
discriminator = Sequential()
discriminator.add(Dense(512, input_shape=(img_dim,), kernel_initializer=init))
discriminator.add(LeakyReLU(alpha=0.2))
discriminator.add(Dense(256))
discriminator.add(LeakyReLU(alpha=0.2))
discriminator.add(Dense(128))
discriminator.add(LeakyReLU(alpha=0.2))
discriminator.add(Dense(1, activation='sigmoid'))

# Embedding condition into D
label_d = Input(shape=(1,), dtype='int32')
label_embedding_d = Embedding(num_classes, img_dim)(label_d)
label_embedding_d = Flatten()(label_embedding_d)
img_d = Input(shape=(img_dim,))
input_discriminator = multiply([img_d, label_embedding_d])
validity = discriminator(input_discriminator)
discriminator = Model([img_d, label_d], validity)
```

然后将两个网络进行结合，使用以下语句：

```
# Combine G and D
optimizer = Adam(lr=0.0002, beta_1=0.5)
discriminator.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['binary_accuracy'])
discriminator.trainable = False
validity = discriminator([generator([z, label]), label])
d_g = Model([z, label], validity)
d_g.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['binary_accuracy'])
```

4.4 训练网络

GAN 的训练与一般的 CNN 网络训练有相通之处，核心思想在于通过上百个 epoch 的训练，使得对抗网络生成的 fake samples 与原数据集的 real samples 尽可能相似，从而使得判别器难以分辨真假。核心代码如下所示：

```
    # Train Discriminator weights
    discriminator.trainable = True
    # Real samples
    X_batch = X_train[i*batch_size:(i+1)*batch_size]
    real_labels = y_train[i*batch_size:(i+1)*batch_size].reshape(-1, 1)
    d_loss_real = discriminator.train_on_batch(x=[X_batch, real_labels], y=real * (1 - smooth))
    # Fake Samples
    z = np.random.normal(loc=0, scale=1, size=(batch_size, latent_dim))
    random_labels = np.random.randint(0, 10, batch_size).reshape(-1, 1)
    X_fake = generator.predict_on_batch([z, random_labels])
    d_loss_fake = discriminator.train_on_batch(x=[X_fake, random_labels], y=fake)
    # Discriminator loss
    d_loss_batch = 0.5 * (d_loss_real[0] + d_loss_fake[0])
    # Train Generator weights
    discriminator.trainable = False
    z = np.random.normal(loc=0, scale=1, size=(batch_size, latent_dim))
    random_labels = np.random.randint(0, 10, batch_size).reshape(-1, 1)
    d_g_loss_batch = d_g.train_on_batch(x=[z, random_labels], y=real)
```

在训练的过程中，记录生成器和判别器的 loss 值，方便最后打印学习曲线。同时每隔 25 个 epoch，保存对应的模型，并生成 0-9 的一组手写数字图像作为样本进行保存，便于最后的对比。由于这部分的代码不难，故不在报告中展示了。

5. 实现结果

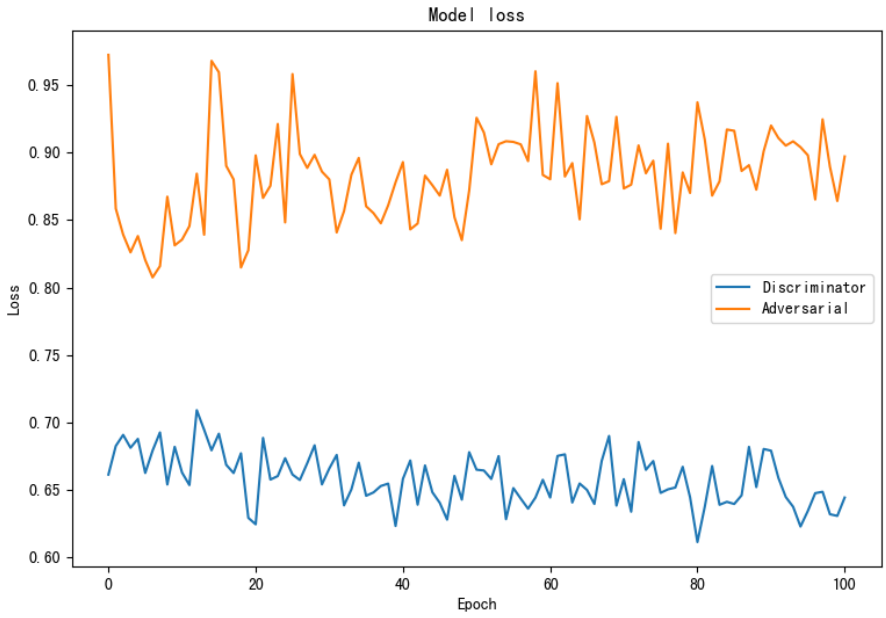
5.1 模型训练

在 Mo 平台上使用 GPU 进行模型训练，训练成功完成，持续时间 36 分钟。

（图 6：模型训练过程截图）

5.2 学习曲线

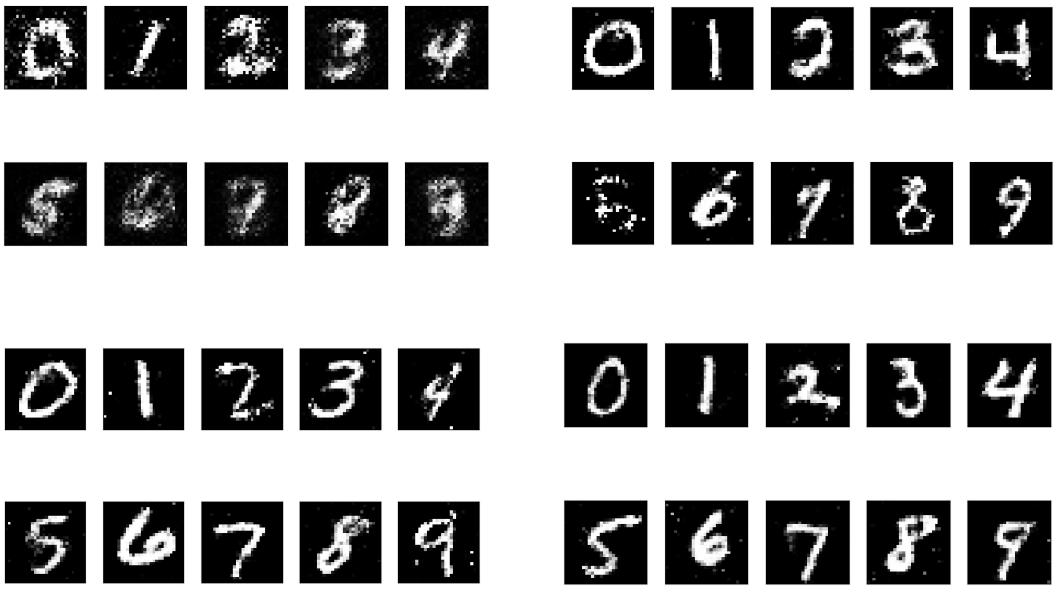
针对训练过程中的 loss 值绘制学习曲线，如图 7 所示。可以看到对抗网络和判别网络的 loss 值都有下降的趋势，但不太明显。对于分类网络，loss 值可以反映分类的效果。但对于 CGAN，其衡量标准应该是生成图像和原始图像的相似程度，loss 不能完全反映 CGAN 的性能，故仅供参考。



（图 7：学习曲线绘制）

5.3 生成图像对比

每隔 25 个 epoch 生成一组手写数字图像，每个图像分为 2 行，第一行为 0-4，第二行为 5-9。以下从左到右从上到下的四个区域，依次为第 1、25、50、75 个 epoch 生成的图像。可以看到图像的生成越来越清晰。



（图 8：第 1、25、50、75 个 epoch 生成的图像）

最后一轮 epoch 生成的手写数字图像如图 9 所示，可以看到比先前生成的要清晰不少，并且和原数据集也非常相似。

（图 9：最后一轮 epoch 生成的图像）

6. 评价

- **优点**：以较少的参数、不太复杂的模型，实现了相对较为清晰的手写数字图像生成。相比于随机生成图像的普通对抗网络，我采用了条件对抗网络 CGAN，从而限制输出图像的标签，能够通过定向生成 0-9 所有的数字更直观有序地展现网络的性能。
- **缺点**：清晰度的提升还有很大空间，可通过模型预训练、增加模型复杂度等方式改进。

7. 心得体会

得知期末考试改成了考察，我花了很长时间思考，究竟选择什么样的课题比较合适。最终我选择了条件生成对抗网络 CGAN 这个课题，原因是本学期在做科研的时候读 paper 读到过一篇通过 CGAN 实现冷冻电镜超分辨率处理（super-resolution）的论文，当时对这个有不少兴趣但没有动手去做，于是决定趁 AI 期末考察的机会，做一个简单的 CGAN 网络看一看。虽然课程内容里没有直接讲 GAN 的内容，但老师上课时候口头提到过，也不算太超纲。在查阅了参考书和参考文献以后，我开始着手去做这个项目。

对我来说，CGAN 中，生成器 G 和判别器 D 的设计与垃圾分类大作业中的网络构建思路类似，陌生的地方主要在于如何引入条件 c 以及网络的连接与传递。在参考了 GitHub 上的一些项目后，我决定使用 MNIST 这个经典的数据集，照猫画虎写出了自己的 CGAN，通过不断调试验证，最终得以成功运行。

一学期结束了，在 AI 这门课上，我不仅学习了不少基础的模型与算法，而且还了解了许多机器学习前沿的研究内容。每当我有疑问时，王老师和助教都非常热心帮我答疑解惑，Mo 平台的老师还会协助我 debug，非常感谢你们！

8. 参考资料

Keras 官方教程：https://keras-cn.readthedocs.io/

文献：https://arxiv.org/pdf/1411.1784.pdf