《人工智能导论》大作业

任务名称：基于Mnist数据集的手写数字生成

完成组号：

小组人员：范骥腾、杨威、朱宇昕

完成时间：2023.6.10

1．任务目标

利用神经网络，实现基于Mnist数据集的生成模型。要求构建的条件生成模型，能接受0~9的数字输入，随机输出对应的生成图像。

2．具体内容

（1）实施方案

我们用卷积的方法实现了**CGAN(Conditional Generative Adversarial Networks)**网络进行模型的构建。

GAN(Generative Adversarial Networks)是常用的神经网络生成模型。其基于鉴别器和生成器的对抗训练模型。一般的GAN网络，可以实现无输入的基于Mnist数据集的生成模型。但是其生成器的种子为随机tensor，故此无法实现接受输入数字输出对应图像的要求。基于以上原因，我们选择了CGAN网络，在GAN网络的基础上，CGAN加入了条件变量，让生成器根据这个条件变量生成对应的假数据。CGAN网络的特性很好地符合了题目要求。

关于CGAN网络的条件变量，我们选择用十维one-hot向量代表0~9的数字，并将其和随机噪声拼接，以此作为generator的输入，并且在计算loss时加入Mnist数据集的真实label，以此实现了功能。

（2）核心代码分析

- 鉴别器类和生成器类

  如图，**鉴别器的神经网络结构**包含如下层次：

  - 一个卷积层(nn.Conv2d)，输入通道数为1，输出通道数为32，卷积核大小为5*5，步长为1，填充为2
  - 一个LeakyReLU激活函数(nn.LeakyReLU)，负斜率为0.2
  - 一个最大池化层(nn.MaxPool2d)，池化核大小为2*2
  - 一个卷积层(nn.Conv2d)，输入通道数为32，输出通道数为64，卷积核大小为5*5，步长为1，填充为2
  - 一个LeakyReLU激活函数(nn.LeakyReLU)，负斜率为0.2
  - 一个最大池化层(nn.MaxPool2d)，池化核大小为2*2
  - 两个全连接层(nn.Linear)，分别将展平后的特征图映射到1024维和10维，并使用LeakyReLU激活函数和Sigmoid激活函数

  生成器的神经网络结构包含如下层次：

  - 一个全连接层(nn.Linear)，输入维度为100，输出维度为num_feature
  - 一个ReLU激活函数(nn.ReLU)
  - 一个卷积层(nn.Conv2d)，输入通道数为1，输出通道数为25，卷积核大小为3*3，步长为1，填充为1
  - 一个BatchNorm2d层(nn.BatchNorm2d)

    - 一个ReLU激活函数(nn.ReLU)
    - 一个卷积层(nn.Conv2d)，输入通道数为25，输出通道数为1，卷积核大小为2*2，步长为2
    - 一个Tanh激活函数(nn.Tanh)
- 损失函数和优化函数

```
# 将一个0~9的数字转换为10维的onehot向量作为输入的一部分
labels_onehot = np.zeros((self.num_img, 10))
labels_onehot[np.arange(self.num_img), label.numpy()] = 1
img = Variable(img)
real_label = Variable(torch.from_numpy(labels_onehot).float())  # 真实label为1
fake_label = Variable(torch.zeros(self.num_img, 10))  # 假的label为0

# compute loss of real_img
real_out = self.D(img)  # 真实图片送入判别器D输出0~1
d_loss_real = self.criterion(real_out, real_label)
real_scores = real_out  # 真实图片放入判别器输出越接近1越好

# compute loss of fake_img
z = Variable(torch.randn(self.num_img, self.z_dimension))  # 随机生成向量
fake_img = self.G(z)
fake_out = self.D(fake_img)  # 判别器判断假的图片
d_loss_fake = self.criterion(fake_out, fake_label)
fake_scores = fake_out  # 假的图片放入判别器输出越接近0越好

# 损失函数优化过程
d_loss = d_loss_real + d_loss_fake
self.d_optimizer.zero_grad()  # 判别器D的梯度归零
d_loss.backward()  # 反向传播
self.d_optimizer.step()  # 更新判别器D参数

# 生成器G的训练
for j in range(self.gepoch):
    z = torch.randn(self.num_img, 100)  # 随机生成向量
    z = np.concatenate((z.numpy(), labels_onehot), axis=1)
    z = Variable(torch.from_numpy(z).float())
    fake_img = self.G(z)  # 将向量放入生成网络G生成一张图片
    output = self.D(fake_img)  # 经过判别器得到结果
    g_loss = self.criterion(output, real_label)  # 得到假的图片与真实标签的loss
    # bp and optimize
    self.g_optimizer.zero_grad()  # 生成器G的梯度归零
    g_loss.backward()  # 反向传播
    self.g_optimizer.step()  # 更新生成器G参数
    temp = real_label
```

- 损失函数：**交叉熵损失函数(nn.BCELoss)**。其是一种常见的损失函数，通常用于分类问题中。它将真实标签和预测概率进行比较，即将真实标签所对应的概率值作为权重，计算预测值与真实值之间的交叉熵。
- 优化函数：**随机梯度下降(SGD)优化器**。其是一种常见的优化器，用于更新神经网络中的参数。它的基本思想是在每次迭代时，使用一个随机选择的样本来计算梯度，然后更新模型参数。SGD优化器的优点是计算简单，适用于大规模数据集，但缺点是容易陷入局部最优解。

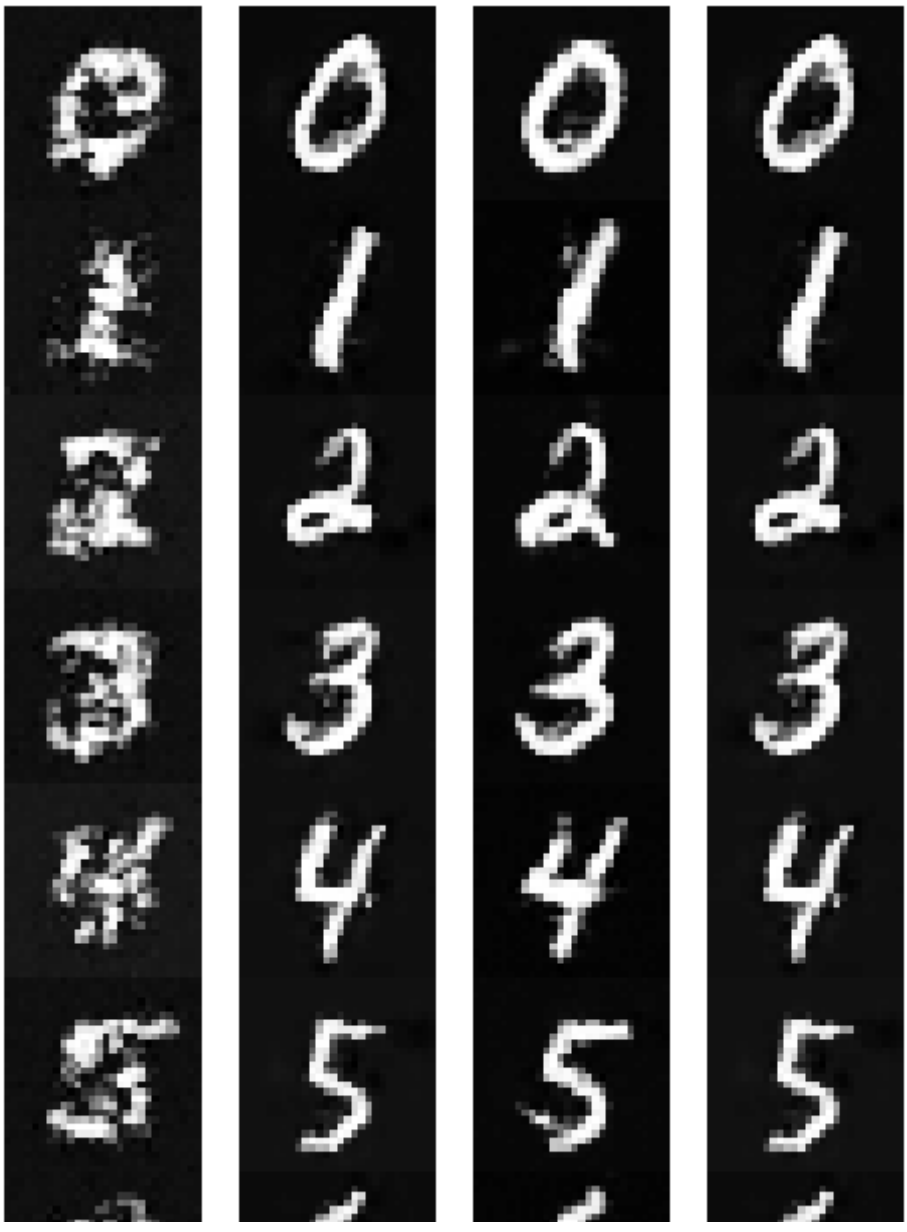
（3）测试结果分析

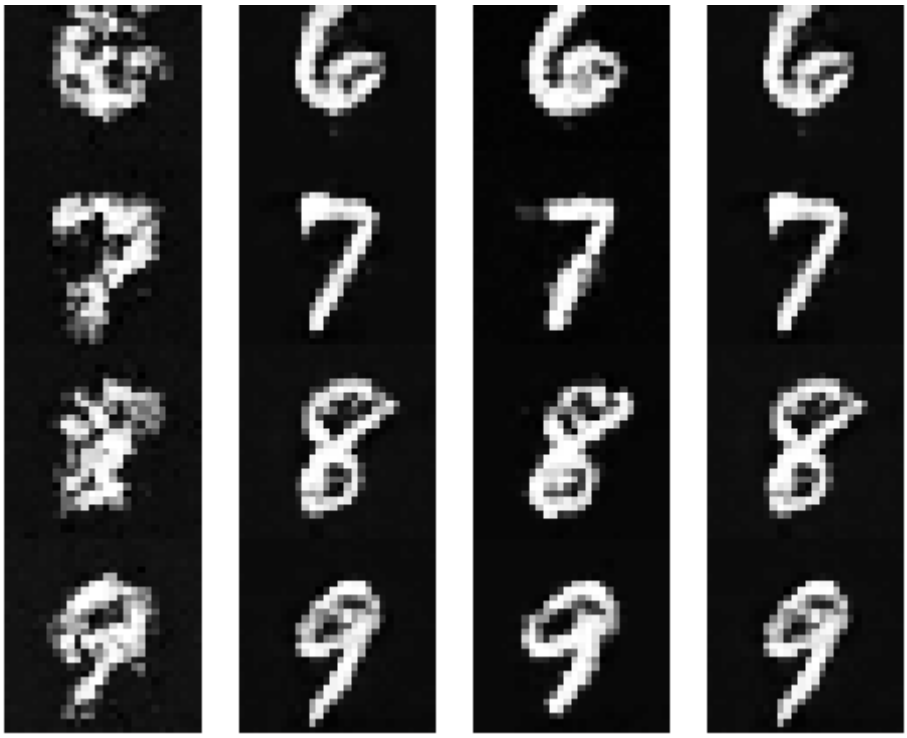
- 图像分析

  分别测试了$epoch = 10, 30, 50, 70$时的生成图像。

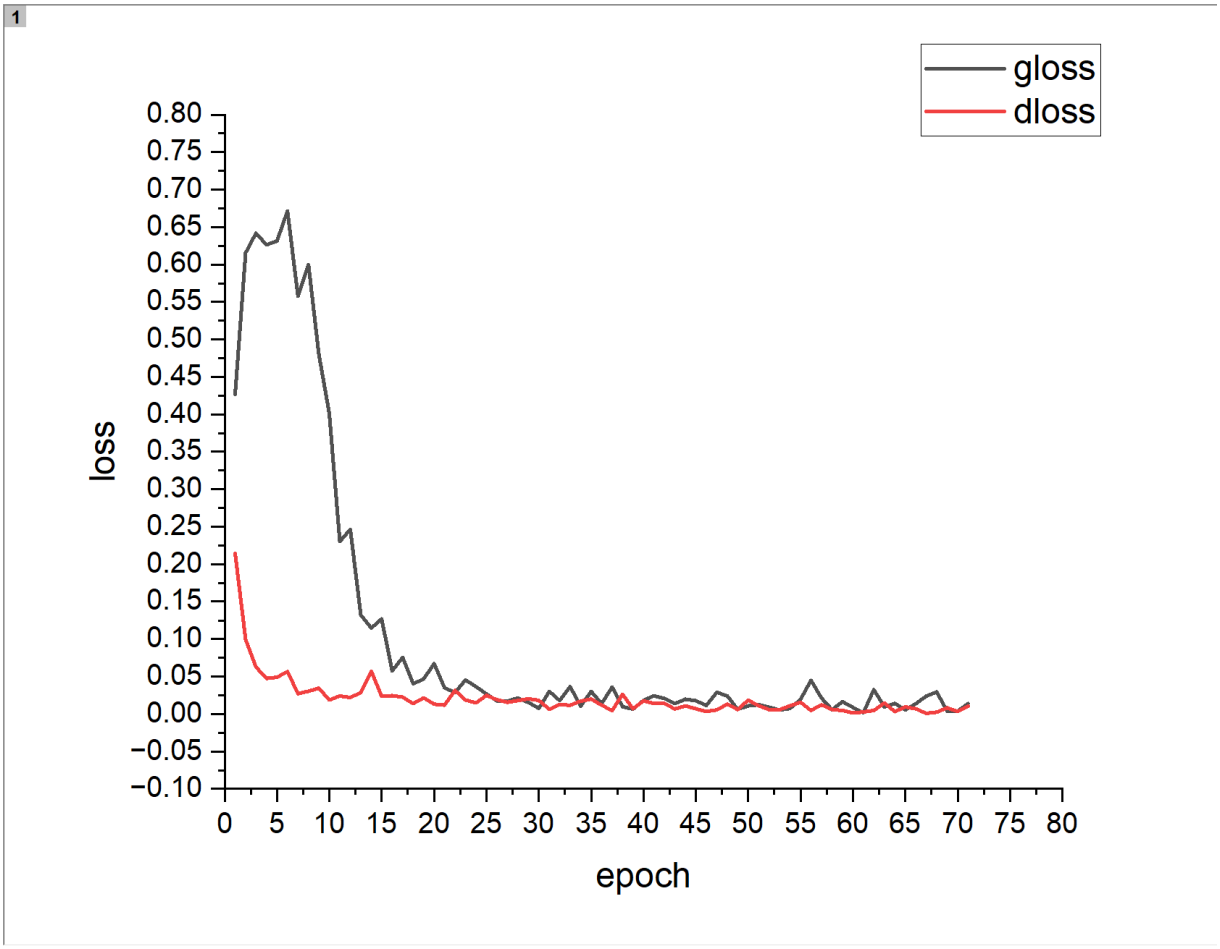
  以下分别为用四个模型生成0~9的图像，可以看到，随着epoch的增加，投喂模型的数据增多，生成器的输出越来越清晰，越来越接近Mnist集。

  观察到，在$epoch$达到30左右时，生成图像的优化明显下降，这表示模型在$epoch = 30$附近收敛。

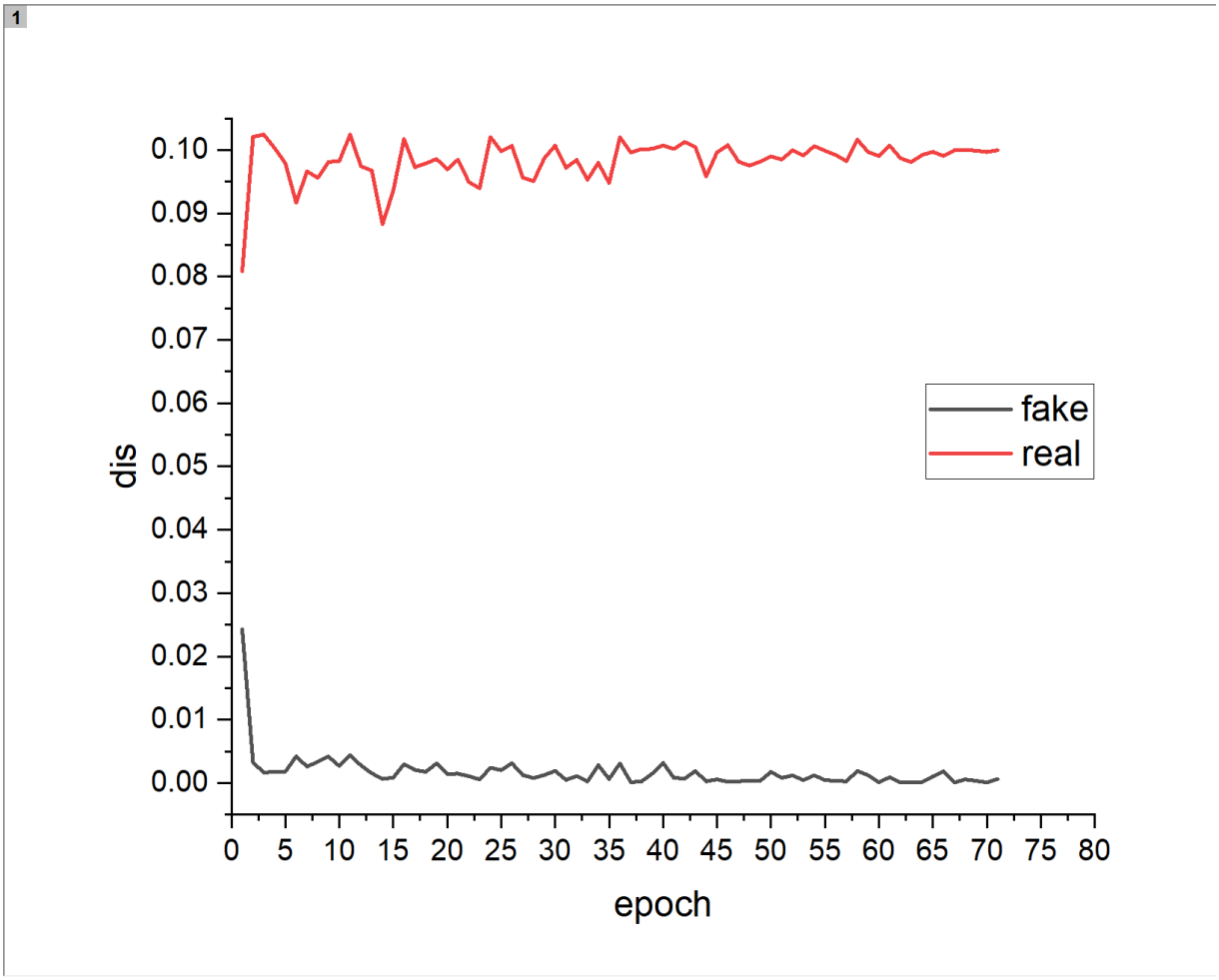




- 损失函数分析



如图为分类器和鉴别器的loss函数随着训练轮次增加而变化示意图，可见在0~15轮loss函数明显下降，而30轮之后图像趋于平缓，和实验得出的结论符合，理论实践相符。



如图为鉴别器的真假值判断能力随着训练轮次增加而变化示意图。同样可以看到，其快速收敛，图像在5轮之后趋于平缓。

3．工作总结

（1）收获、心得

经过这次实践，我们首次亲身体验了人工智能的构建过程，揭开了对于我们来说十分神秘的AI的面纱。我们发现AI的基础理论其实很简洁，但是简洁的理论能实现这么神奇的功能让我们十分兴奋。我们经历了刚开始构建模型的激动，中期调试受挫的烦躁，最终成果实现的喜悦。通过这次大作业，我们的团队合作能力、资料搜集能力、代码写作能力进一步提升，收获颇丰。

（2）遇到问题及解决思路

- 对神经网络各层的了解欠缺，不知道怎么样用卷积、激活等层构建神经网络合适。

  解决思路：首先回顾了各个层的定义，之后检索了类似神经网络模型的代码，观察他们的结构设置。过程中用到了chatgpt进行资料的查找，最终整合，形成了自己的网络结构。

- torch的各种库函数完全陌生，不了解其能实现怎样的功能，在没有了解到nn.Sequential()等模块前甚至想手动实现卷积、池化层等操作。

解决思路：检索了几个mnsit识别的神经网络，详细阅读并理解其代码，一边阅读一边借助官方文档等资料，增进对torch的了解。

- 中途发现GAN网络不能实现根据输入生成，前面写的很多代码成了无用功。

  解决思路：快速调整心理，积极投入到检索适合任务的模型，最终找到了CGAN模型。在原有代码基础上进行修改，将之前的代码重新利用，节省了大量时间。

- 繁多的数据类型如Variable、tensor、list等及其size，对代码编写造成很大困扰。

  解决思路：记忆各种情况下的报错信息，一看到报错就知道什么变量转换错误，久病成良医。

4．课程建议

- 我们对本次课程的几次实践作业都十分感兴趣，这种亲手参与到一个神经网络模型的构建、调试的经历，真的十分有成就感，尽管我们实现的模型可能十分简陋粗糙。希望课程能够加入更多类似的增强动手能力、培养同学相关兴趣的实践内容，理论与实践结合，同学们才能学会更多、理解更多。
- 大作业要求pytorch版本为1.8.0，该版本不支持30、40系列的显卡，而同学们的PC显卡大多数为30、40系，导致GPU资源无法利用，模型训练速度有显著下降。建议可以考虑更新pytorch和对应的torchvision、torchaudio的版本要求。