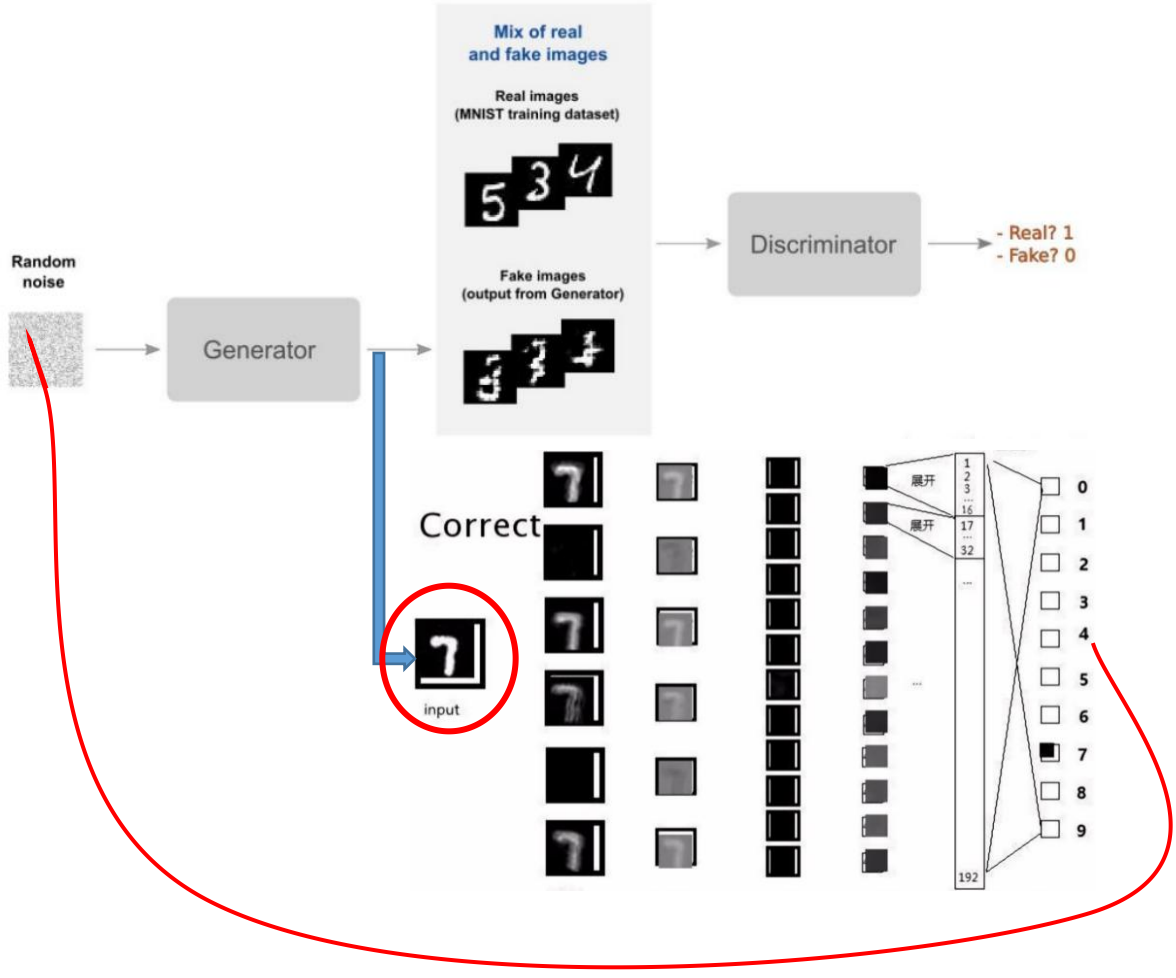
模型设计：

1. 加载 MNIST 数据集
2. 训练 WDCGAN（生成模型的架构是卷积神经网络，距离函数使用 Wasestern 距离，因为我的实验显示这个距离比普通交叉熵函数收敛更快效果更逼真；训练分类器为二分类，使用 BN 等调参技巧；具体训练代码可见 WDCGAN.py。最后得到了一个生成器，它可以把输入的随机噪声变成手写体的数字生成出来。）
3. 训练 DCNN（使用 MNIST 数据训练一个普通的小型卷积神经网络，这个网络的功能是把输入的手写体数字图片识别为 0-9 数字中的其中一个。具体训练代码可见 DCNN.py）
4. 把训练好的模型参数存到本地，在 Category_DCGAN.py 里导入。
5. 把 WDCGAN 和 DCNN 拼接起来，随机向量输入 WDCGAN 输出是生成图片，再把生成图片输入 DCNN 做分类，这个拼接模型称为 Model。
6. 使用神经科学里求感受野的办法：首先确定要生成的数字，比如说“7”，依次输入随机向量给 Model，如果在最后一层的输出向量在 7 这个神经元激活最大那么保留这个输入的随机向量（这个随机向量记为合格的向量），这样的过程一直重复最多 1000 次，如果 1000 内出现里 5 次合格向量则跳出循环，平均这 5 个向量输入 WDCGAN 得到生成的“7”这个手写体数字。如果 1000 次内少于 5 个合格向量则使用 Activation maximation 方法：Model 作为模型结构，目标函数是“7”那个神经元的激活值的相反数，使用 Adam 算法训练，被训练的参数是输入的随机向量，经过 500 次迭代训练后提取训练好的输入向量作为合格向量，把合格向量输入 WDCGAN 得到生成的“7”手写体数字。

模型架构如下：



模型运行：

1. 打开 Category_DCGAN.py 文件，翻到代码最底下

```
# Visulization the generated picture
model = Model()
category_vectors = category2vector(model, 7)
img = net_generator(category_vectors)
plt.figure()
plt.imshow(img[0,0,:].data.numpy())
plt.axis('off')
```

2. 代码底部，其中 category_vectors = category2vector(model, 7)这行代码里的数字“7”代表生成数字，可以换成 0-9 之间任意一个整数。

3. 输入 0-9 之间任意一个数字即可生成对应的手写体数字图片，且每次产生的图片不一样，完成了作业要求。

运行结果：