注意事项：

1. 进入A1-1系统，密码100875（不要进入其他系统）。
2. 由于本课程和AI课程共享一个系统，所以还有别的同学在别的课程的作业，所以不要删除系统内任何已经存在的文件!
3. 由于下一组同学和大家使用相同的系统，所以请大家离开教室前做好备份（U盘or邮件）并删除208机房上你的作业。
4. 每一次上机后，助教都会检查每一台电脑中作业是否删除，如果还存在，助教会删除。

程序保存目录

- home\目录下，新建“image_processing”文件夹。
- 这里面你们可以随意**鼓捣**！

Deep Learning for Computer Vision: Experiment2: Denoising Auto encoder

Libao Zhang, Jie Ma

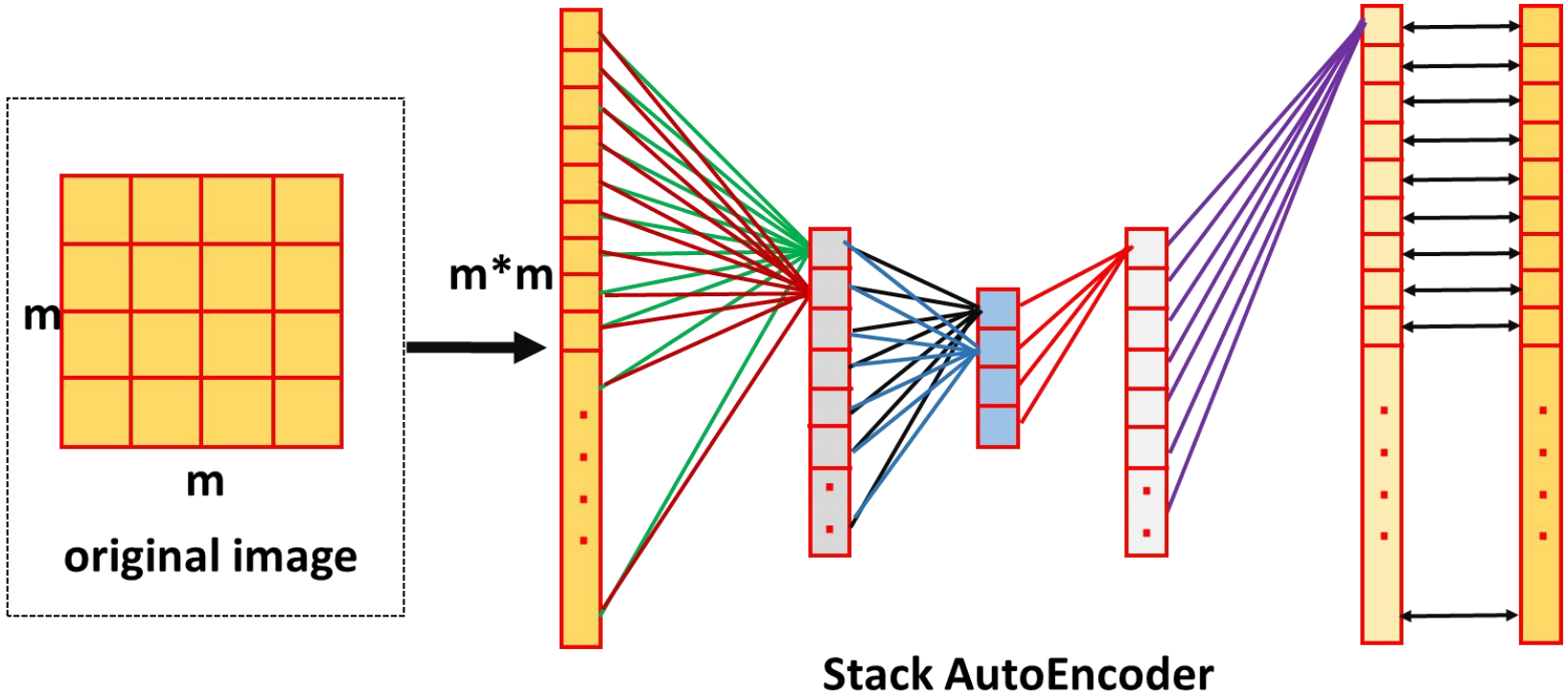
2017.12.19

内容

- 实验原理
- 网络结构
- 实验步骤

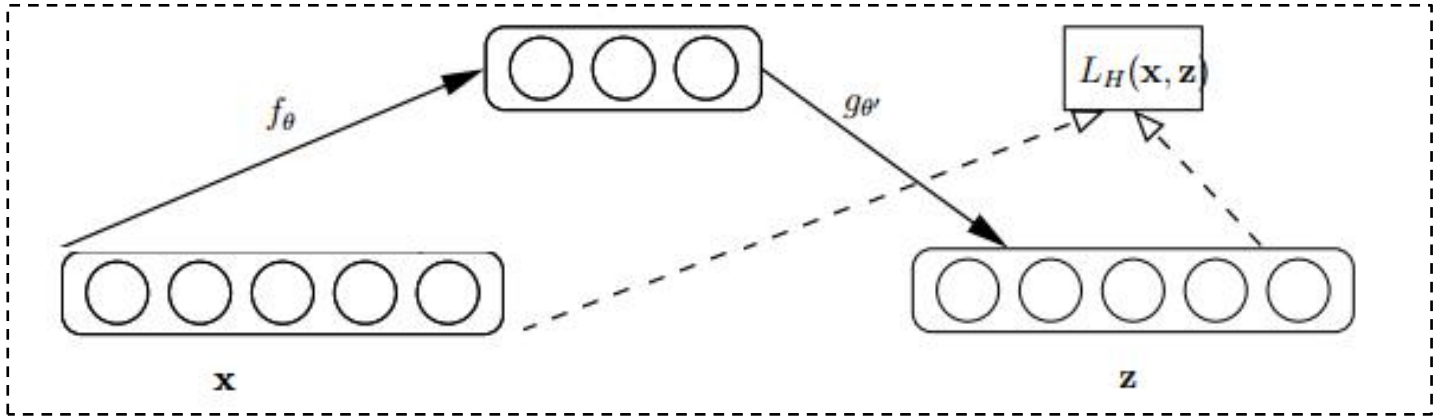
实验原理

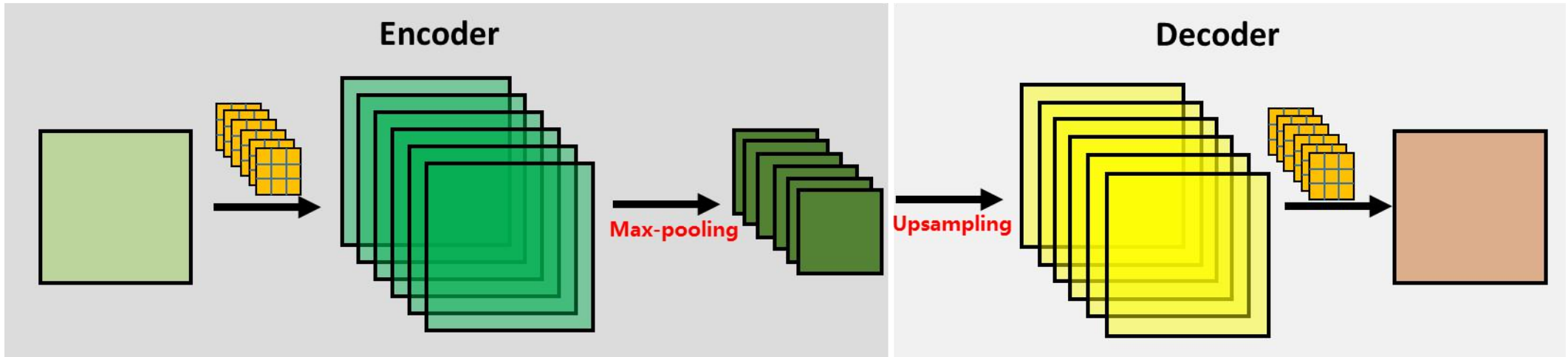
- 自编码器



实验原理

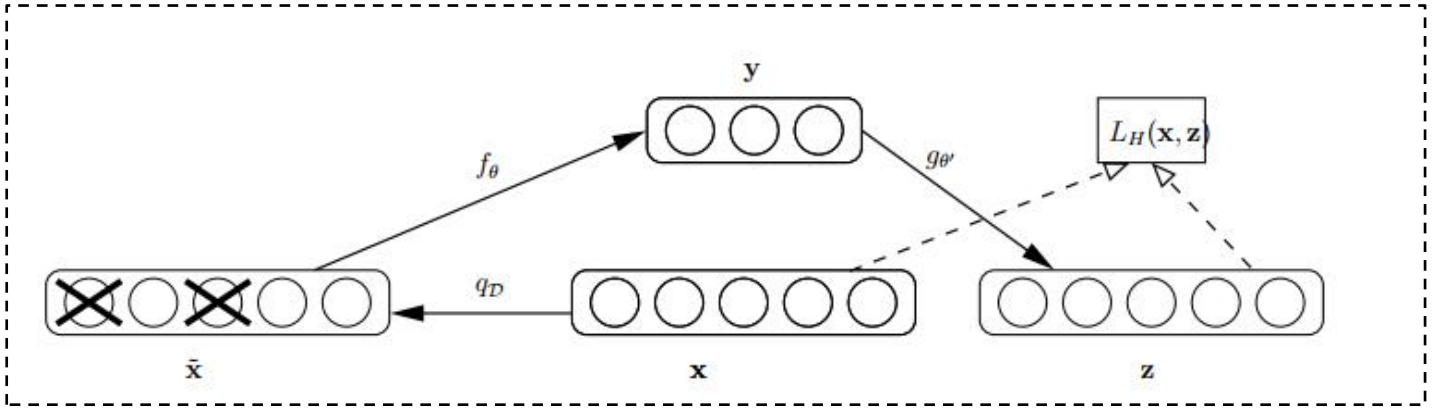
- 卷积自编码器

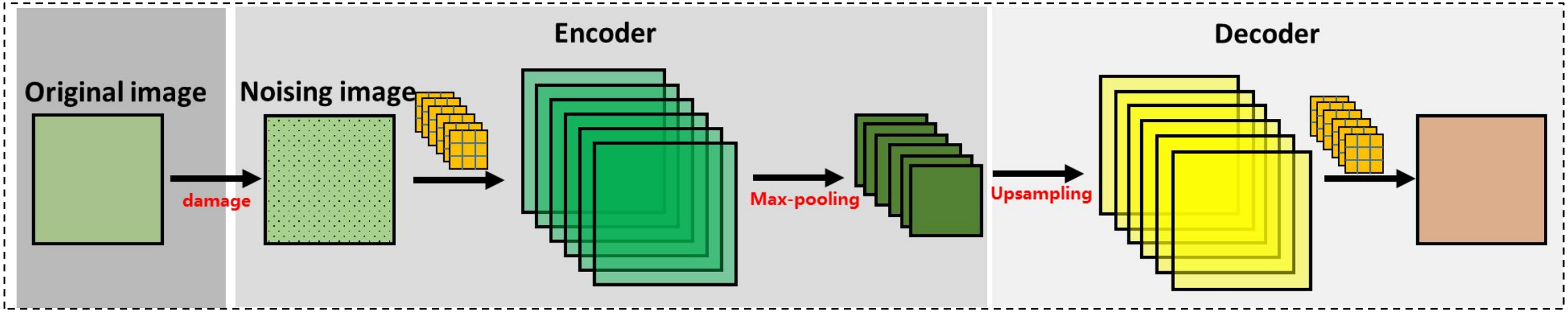

f_θ
$g_{\theta'}$
$L_H(\mathbf{x}, \mathbf{z})$
x
z


Encoder
Max-pooling
Decoder
Upsampling

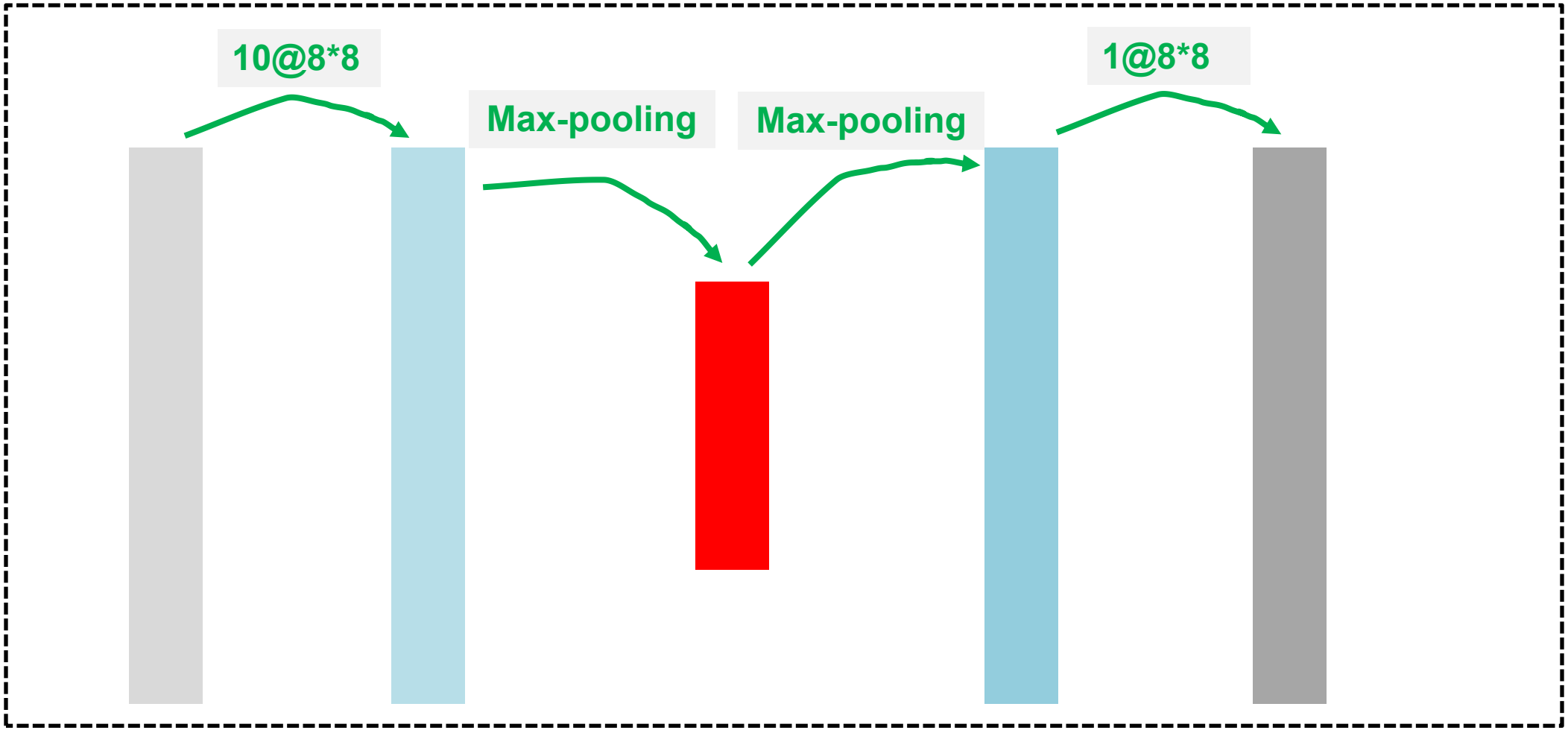
实验原理

- 降噪卷积自编码器





网络结构



The architecture of autoencoder

实验步骤

1. 完成demo1中”**1.1网络结构“及“1.2 网络训练”**部分，根据Demo1中的网络结构写出每一层的网络结构和参数个数。
2. 运行demo1.py，**更改迭代次数**，探究迭代次数对于图像重建的影响，完成demo1中“1.3”(**展示的图像为对应自己学号的mnist数字**)。
3. 参考demo1.py的网络结构，自己设计一个**层数更高的卷积自编码器**，完成“2.1-2.3”(展示的图像为对应自己学号的mnist数字)。
4. 将demo2.py中的网络结构应用于demo3的**降噪自编码器**，完成“3.1-3.3”(展示的图像为对应自己学号的mnist数字)。
5. 结合课件, demo1.py, demo2.py, demo3.py, 完成“问题”部分。

Demo1.py

```
#*****************************data preprocessing*************************************
#load data
img_rows, img_cols = 28, 28
input_shape = 28, 28,1
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = np.double(np.reshape(x_train, (len(x_train), img_rows , img_cols,1)))/255
x_test = np.double(np.reshape(x_test, (len(x_test), img_rows,img_cols,1)))/255

#************************construct a convolutional stack autoencoder*****************

auto_encoder = Sequential()
auto_encoder.add(Conv2D(5, (8, 8), activation='relu', padding='same',input_shape=input_shape))
auto_encoder.add(MaxPooling2D((2, 2), padding='same'))
auto_encoder.add(UpSampling2D((2, 2)))
auto_encoder.add(Conv2D(5, (8, 8), activation='relu', padding='same'))
auto_encoder.add(Conv2D(1, (3, 3), activation='sigmoid', padding='same'))
```

数据预处理

编码

解码

Demo1.py

```
#**********************************model compile*************************************

auto_encoder.compile(optimizer='sgd', loss='mean_squared_error')
auto_encoder.summary()

#*************************************training*****************************************

auto_encoder.fit(x_train, x_train,
                 epochs=5,
                 batch_size=128,
                 shuffle=True,
                 validation_data=(x_test, x_test))
```

模型配置，训练

```
#obtain the model predicetions

decoded_imgs = auto_encoder.predict(x_test)

#save the model
auto_encoder.save('denoise_epoch_5'+'.h5')
```

计算模型输出，保存模型

Demo1.py

注意这里要输出学号对应的数字

```
plt.figure(figsize=(20, 4))
n=10

#show the differents between the groudtruth and model predictions
for i in range(n):
      # display original
          ax = plt.subplot(2, n, i + 1)
          plt.imshow(x_test[i].reshape(28, 28))
          plt.gray()
          ax.get_xaxis().set_visible(False)
          ax.get_yaxis().set_visible(False)
          # display reconstruction
          ax = plt.subplot(2, n, i + 1 + n)
          plt.imshow(decoded_imgs[i].reshape(28, 28))
          plt.gray()
          ax.get_xaxis().set_visible(False)
          ax.get_yaxis().set_visible(False)

plt.show()
```

展示模型输出

Demo3.py

```
#**********************************data preprocessing*************************************************
#load data
img_rows, img_cols = 28, 28
input_shape = 28, 28,1
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = np.double(np.reshape(x_train, (len(x_train), img_rows , img_cols,1)))/255
x_test = np.double(np.reshape(x_test, (len(x_test), img_rows,img_cols,1)))/255

#add noise
noise_factor = 0.3
x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0, scale=1, size=x_train.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1, size=x_test.shape)
x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)
```

加噪