**1.基于 MNIST 数据集的自编码器实现**

(1)完成数据读写并试着搭建深度自编码器网络。

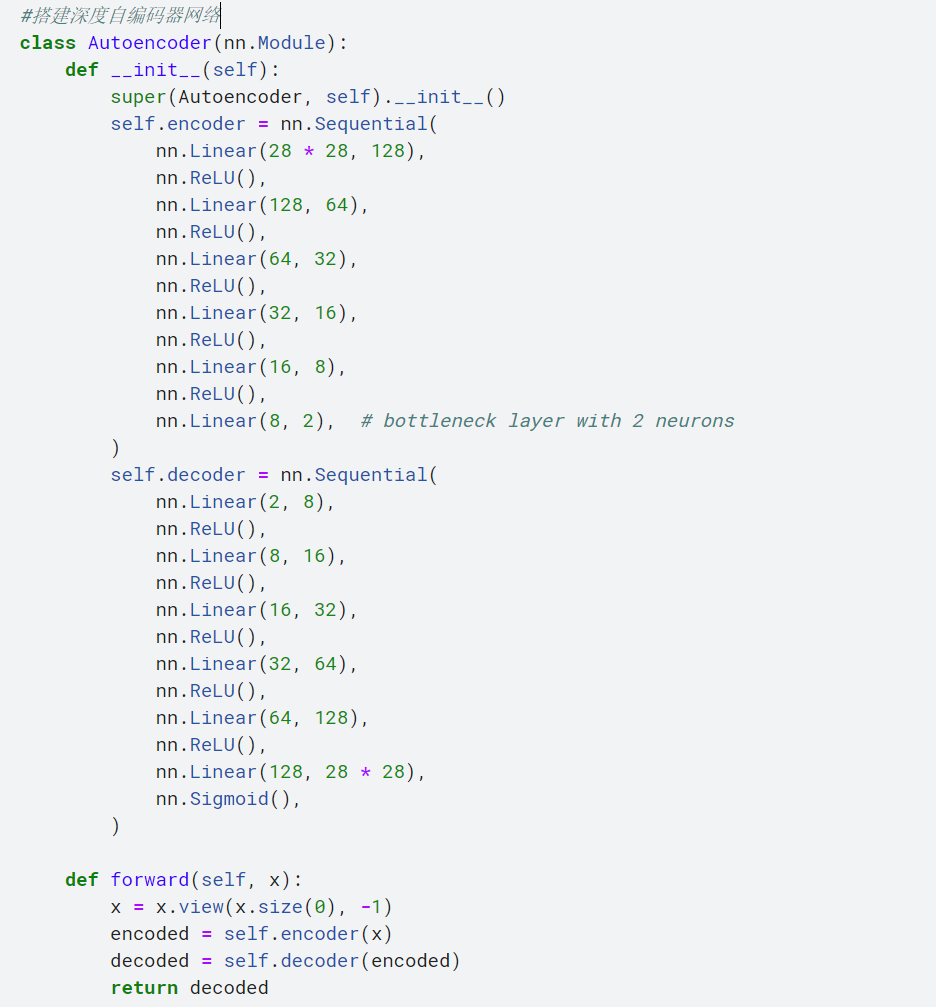
读取MNIST数据集中的数据，包括train-images、train-lables、t10k-images、t10k-labels：



进行数据预处理，将图像转换为张量，并进行标准化：



搭建深度自编码器网络：

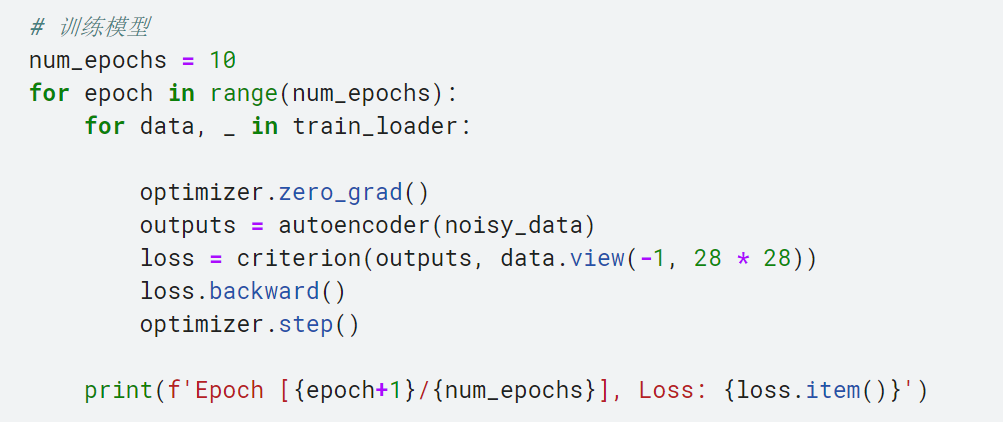


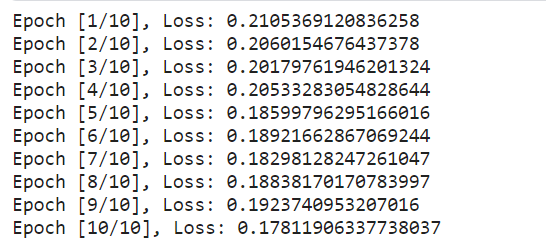
(2)选择二元交叉熵函数作为损失函数，在限制 bottleneck 层维度为 2 的情况下训练模型。



使用二元交叉熵损失函数 (nn.BCELoss()) 进行模型训练，而限制编码器（bottleneck）层的维度，通过在模型定义中设置编码器最后一层的输出维度为2来实现，即nn.Linear(8, 2)。

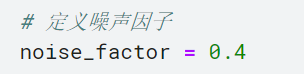
训练模型：



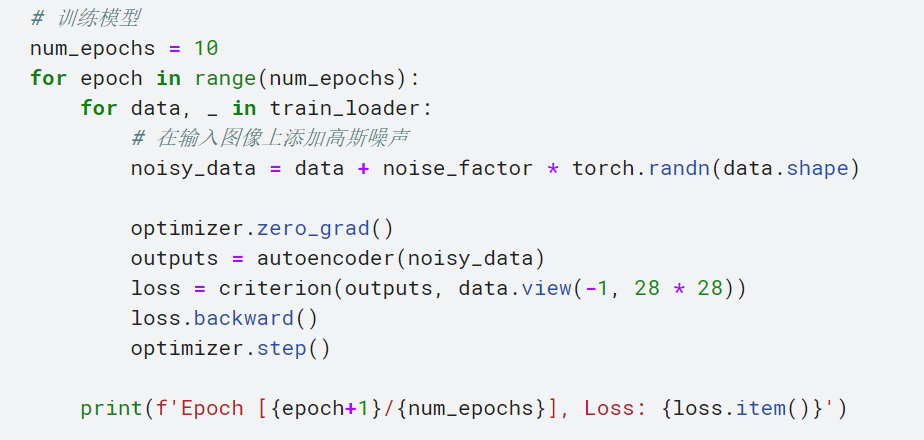


(3)设置噪声因子为 0.4，在输入图像上叠加均值为 0 且方差为 1 的标准高斯白噪声，训练降噪自编码器，并进行降噪结果展示。

设置噪声因子为0.4：



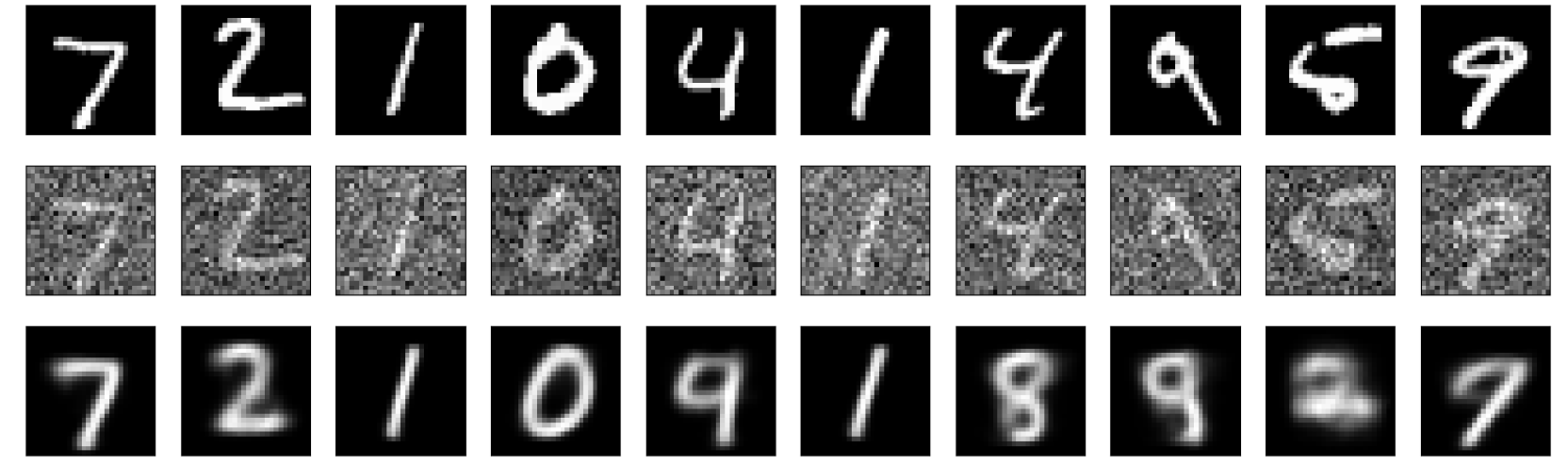
在输入图像上叠加均值为 0 且方差为 1 的标准高斯白噪声进行训练：



进行降噪结果展示：



降噪结果：



1. 试在问题(2)的基础上，对 latent code 进行均匀采样，并利用解码器对采样结果进行恢复，观察并描述所得到的结果。

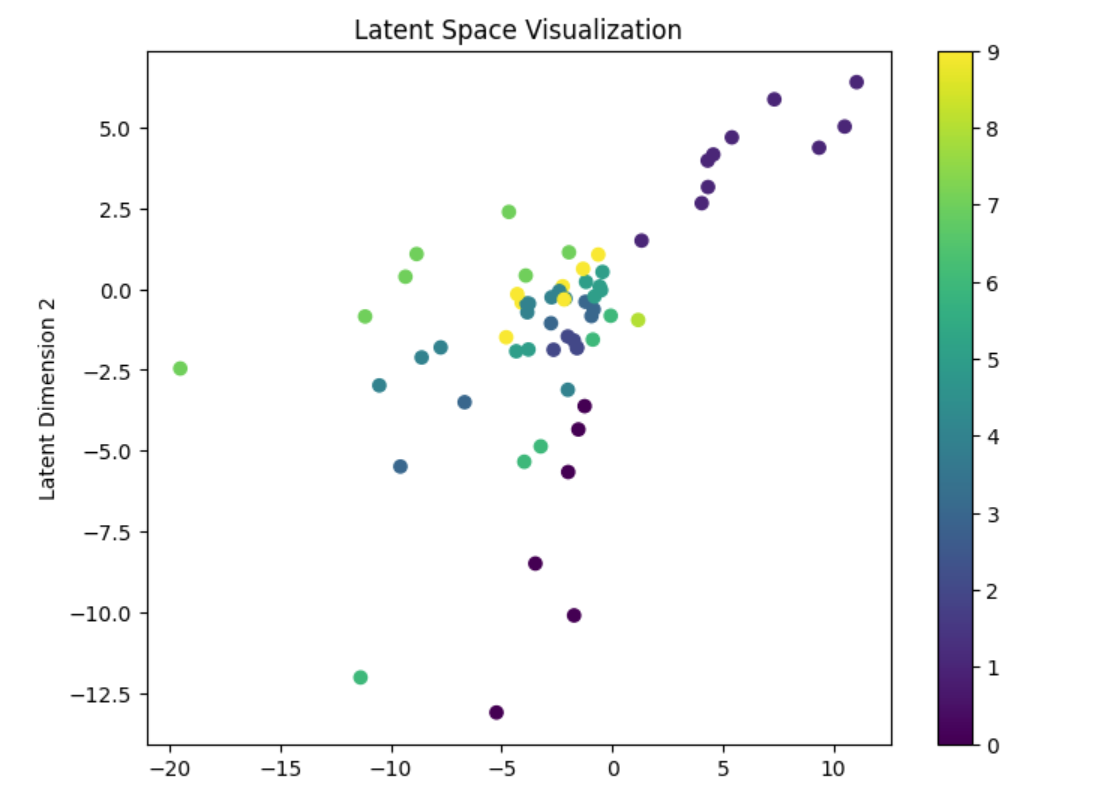
获取测试集中的图像和标签，对latent code进行均匀采样并利用解码器对采样结果进行恢复：



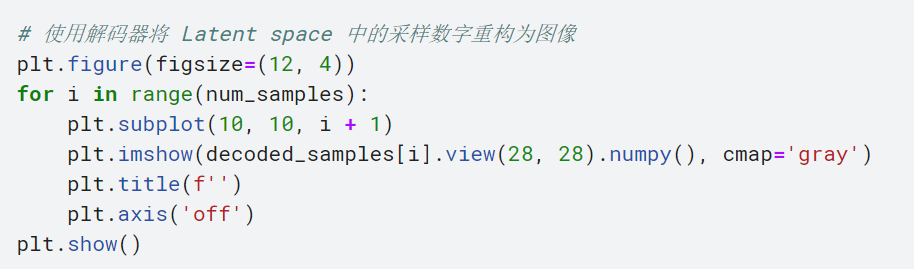
显示latent space中的数字聚类情况：



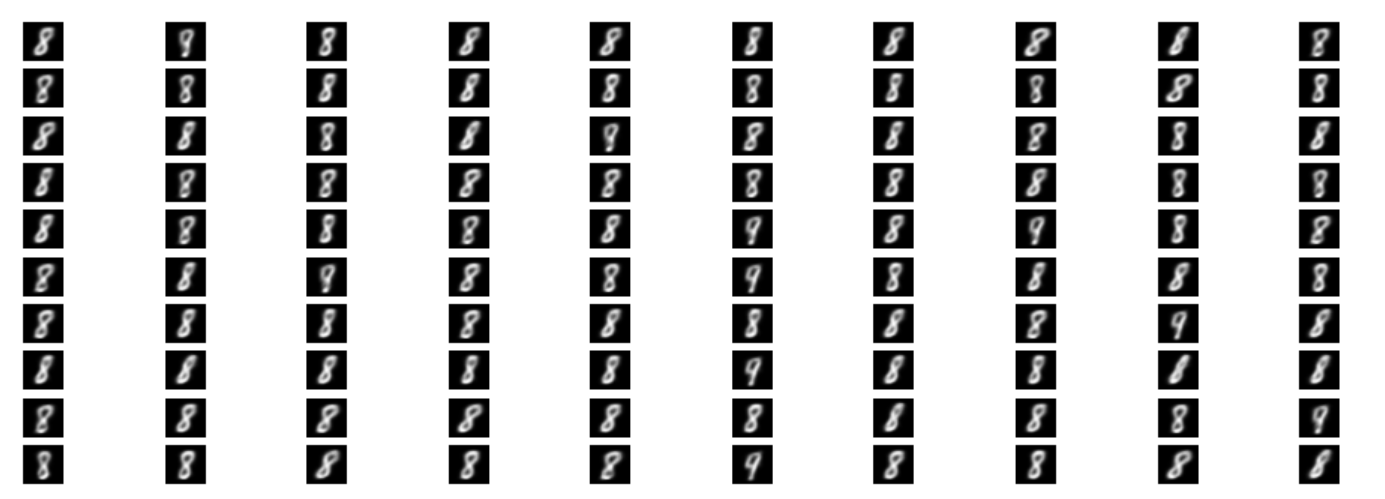
显示结果：



利用解码器将latent space中的采样数字重构为图像：

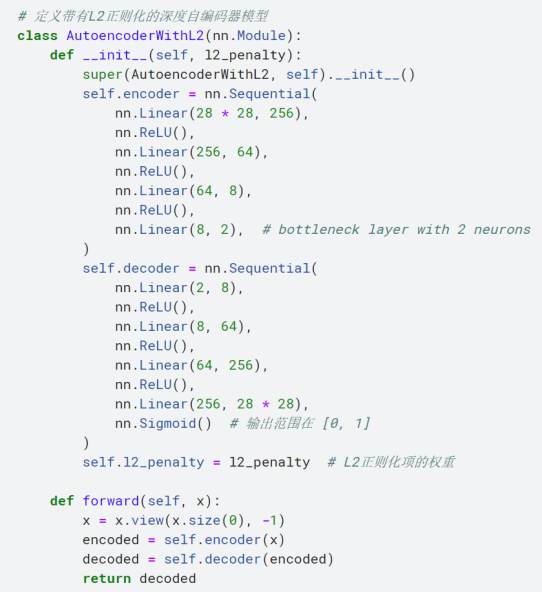


重构结果：

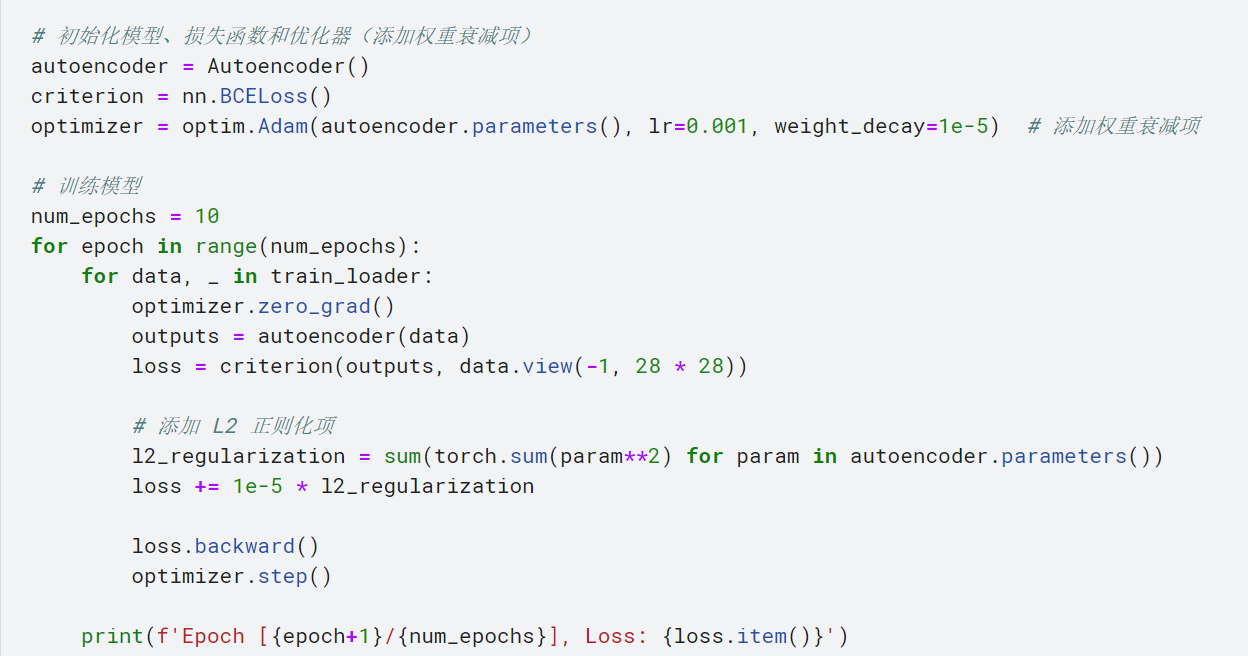


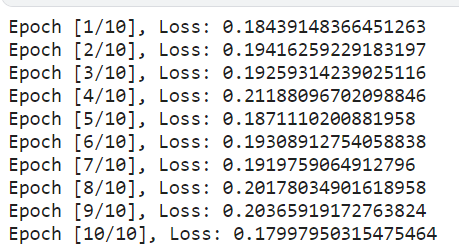
(5)试在问题(4)的基础上，在训练深度自编码器时使用 L2 正则化，观察并描述所得到的结果。

定义带有L2正则化的深度自编码器模型：

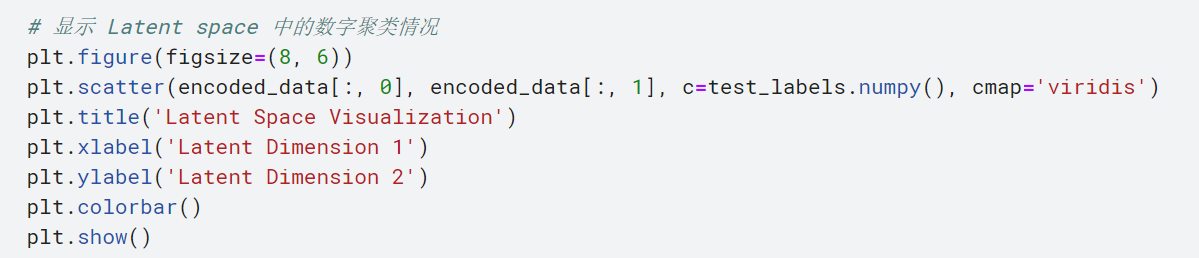


初始化并训练带有L2正则化的模型：

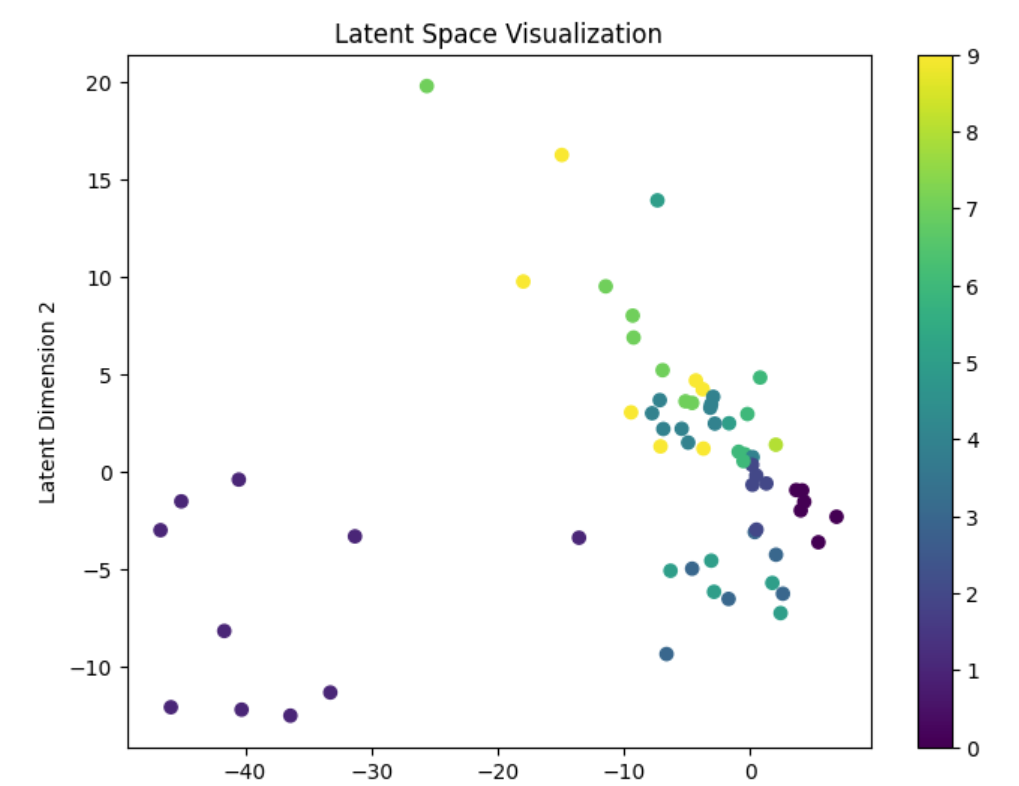
训练结果：



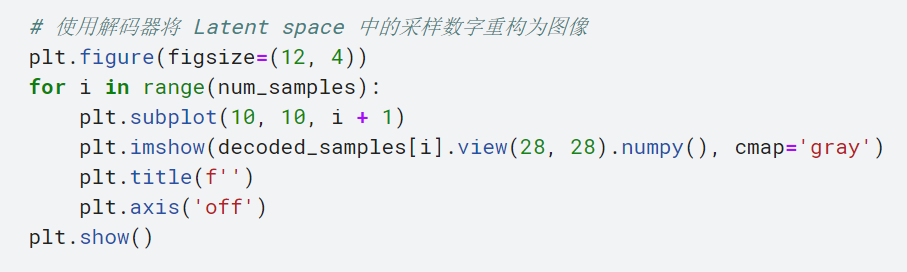
显示对应的latent space中的数字聚类情况：



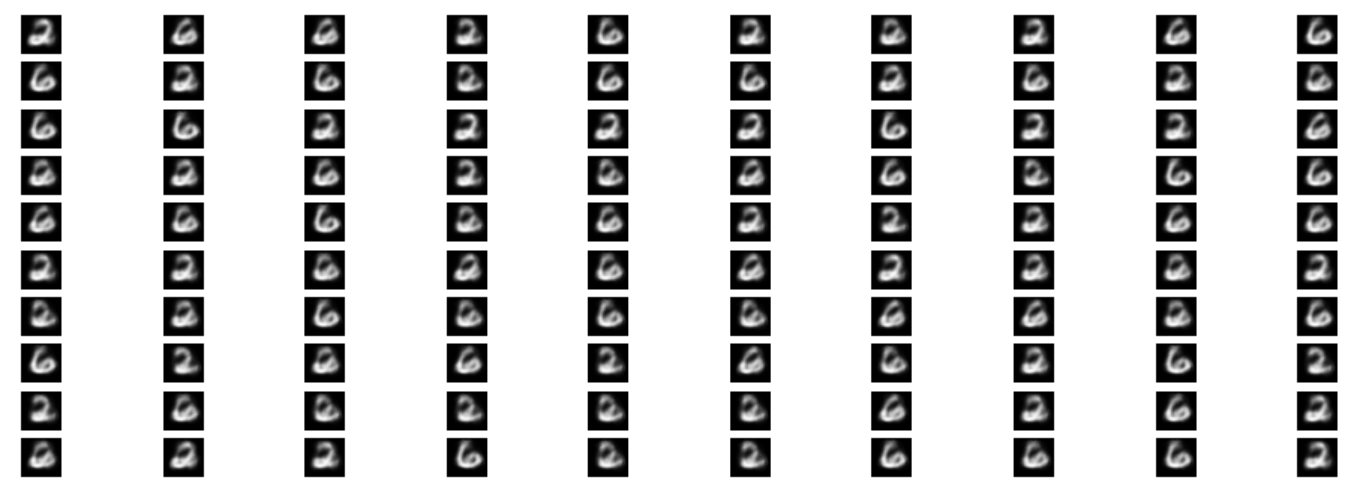
显示结果：



用解码器将latent space中的采样数字重构为图像：



重构结果：

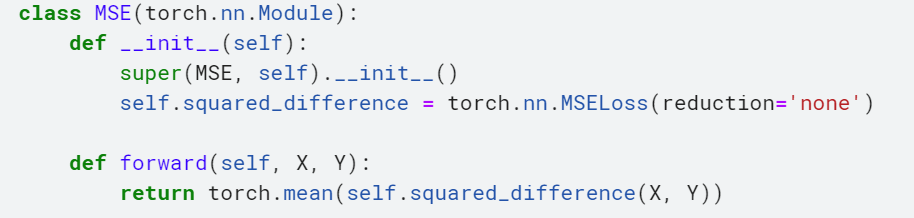


结果显示添加L2正则化后latent space的数字聚类变得分散，防止了过拟合现象的产生。

**2.隐空间特性探究**

（1）以下图结构为参考，以MSE作为损失函数，设置c的维度（latentsize）为8\*8\*16，搭建并训练深度自编码器网络

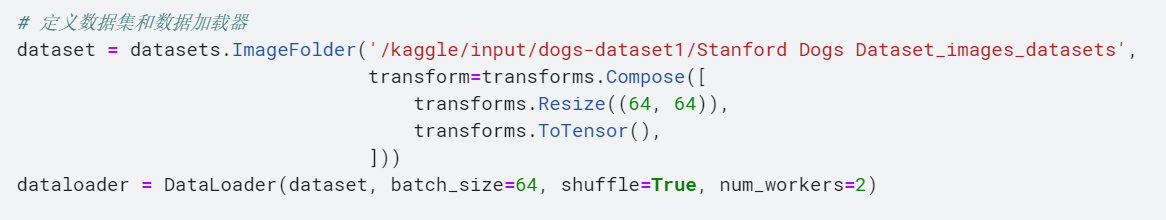
以MSE为损失函数：



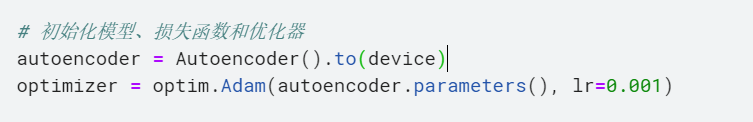
搭建深度自编码器模型：



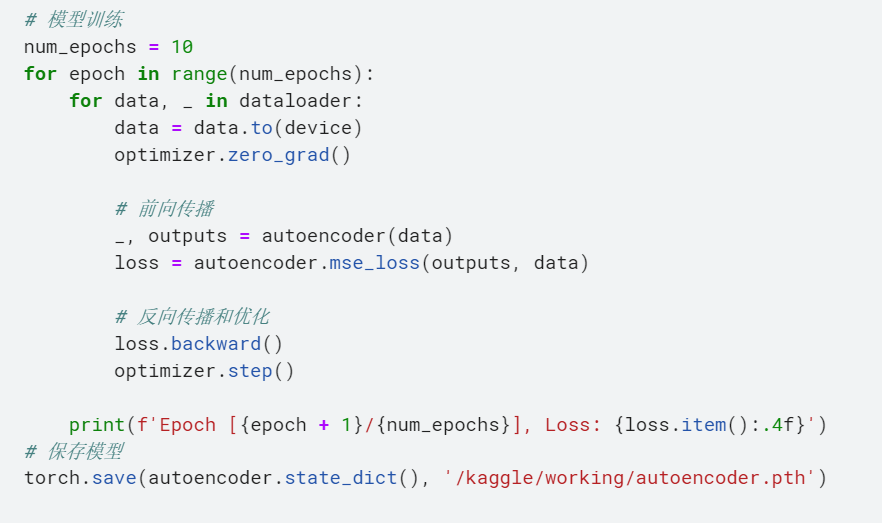
定义数据集和数据加载器：



初始化模型、损失函数和优化器：



训练模型：



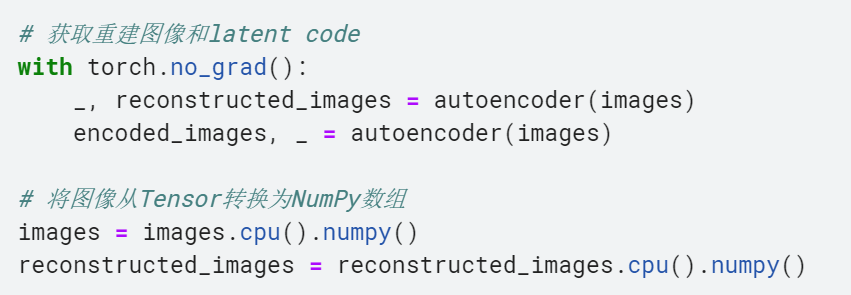


1. 随机选取9张图，展示原图和重建图像，并对latent code进行可视化。

从数据加载器中随机选择9张图片：



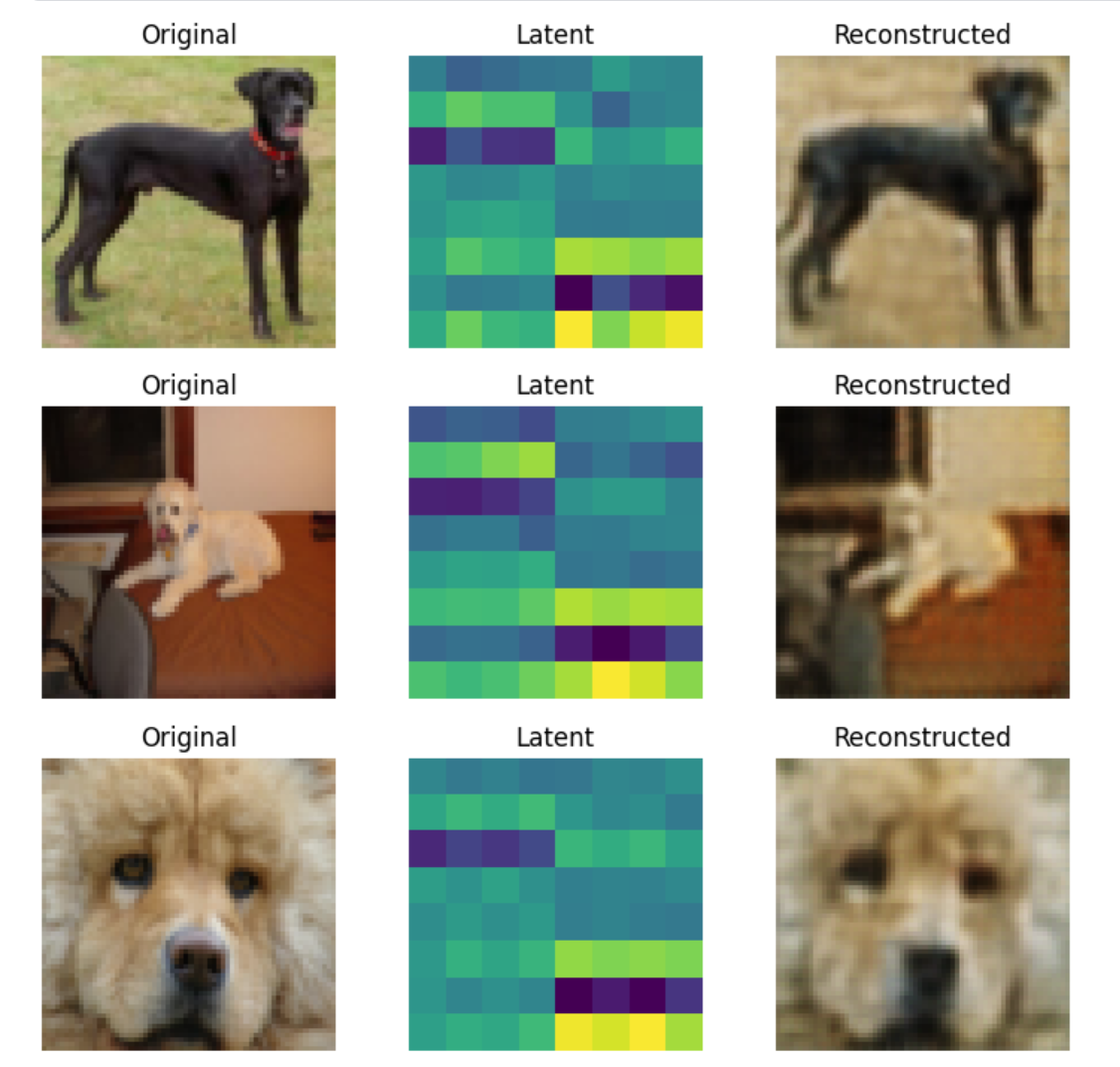
获取重建图像和latent code并将图像转换为NumPy数组：

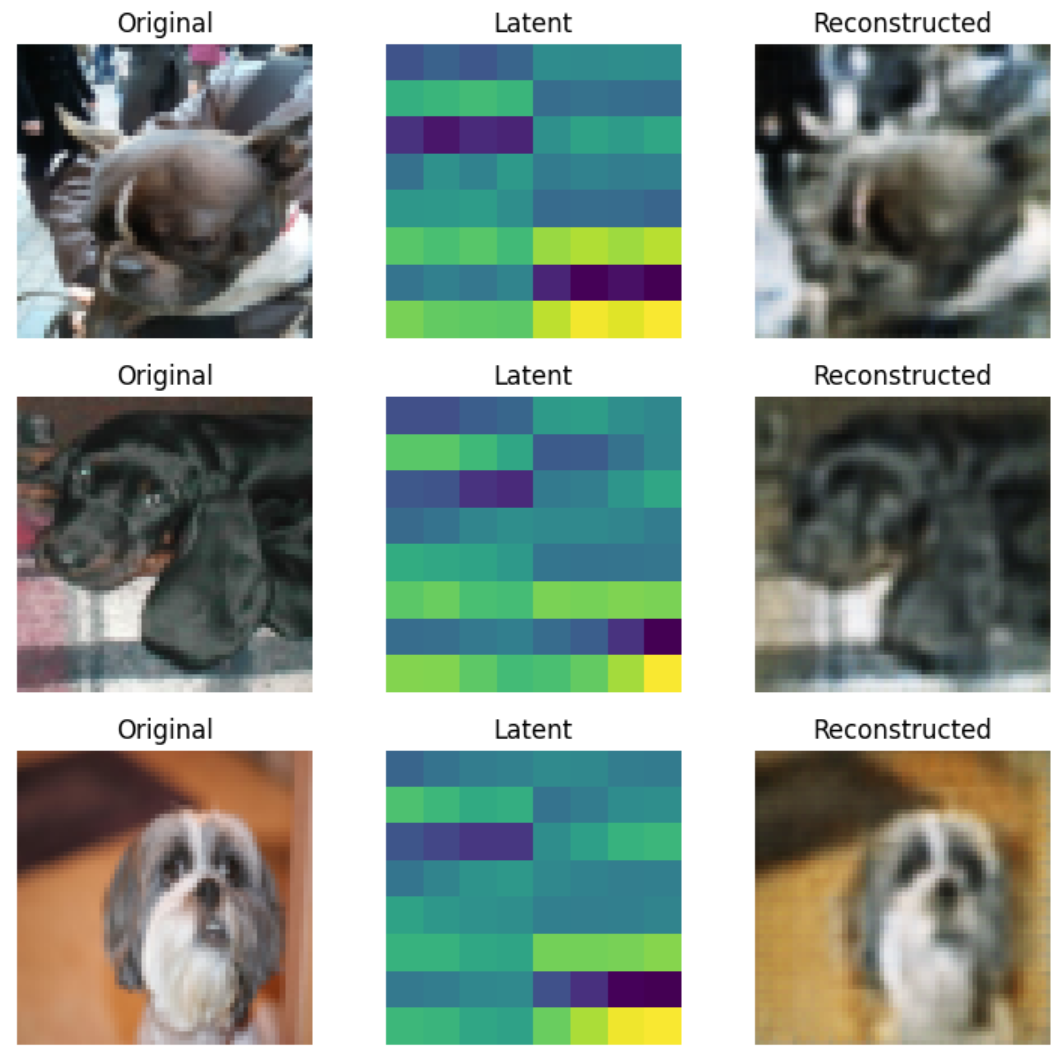


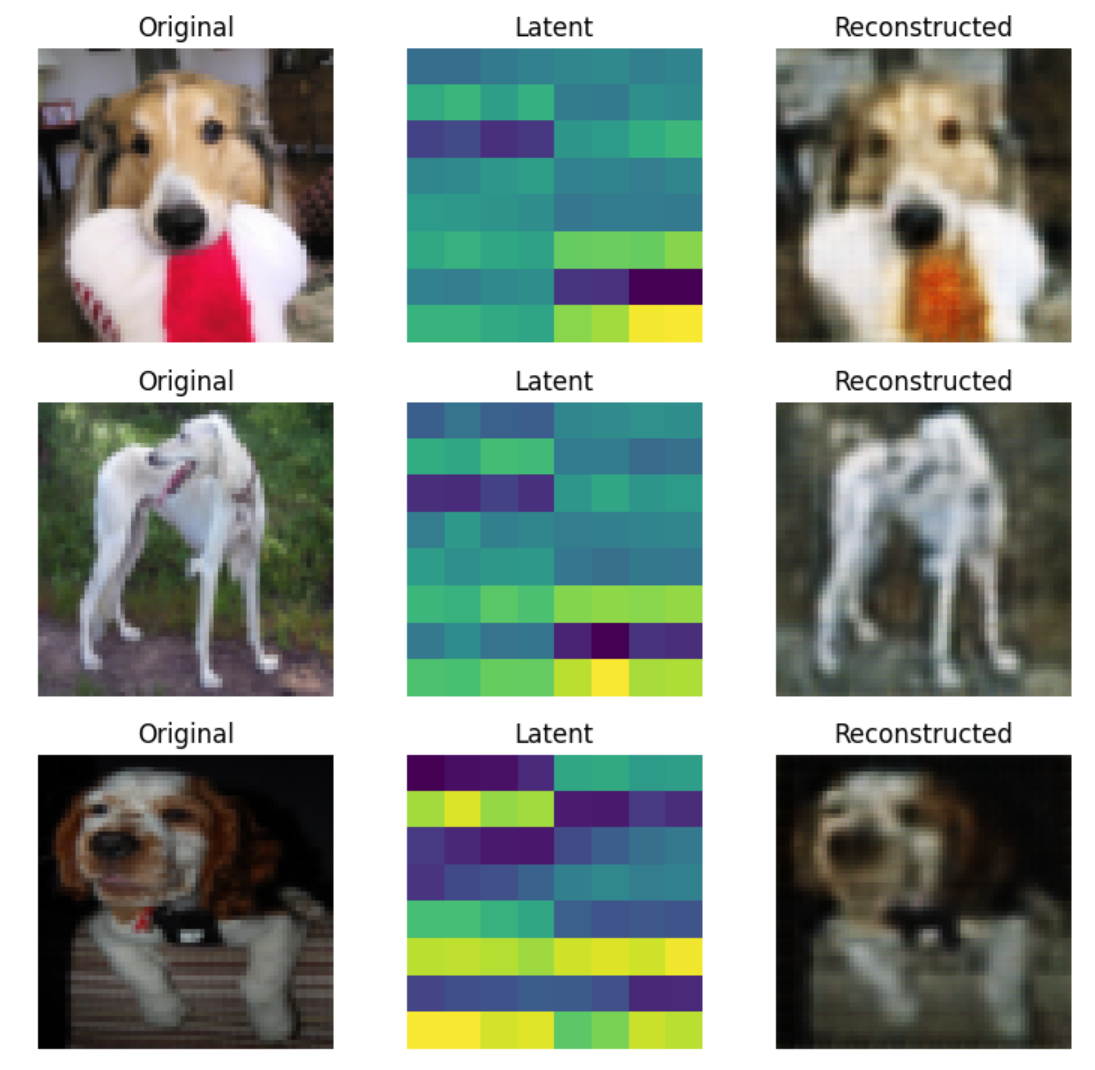
可视化原图、latent code和重建后的图像，分组进行展示：



展示结果：

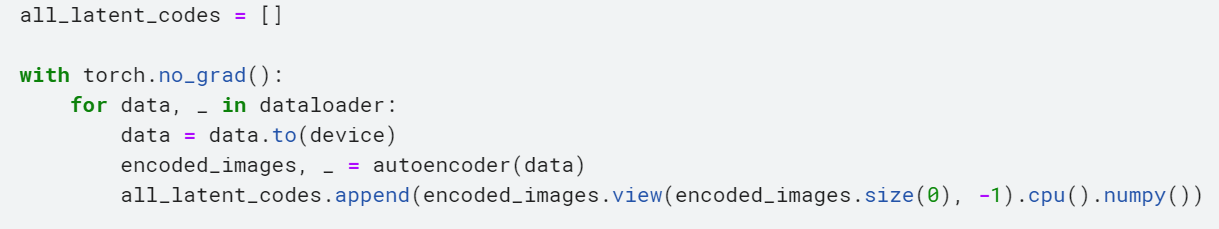




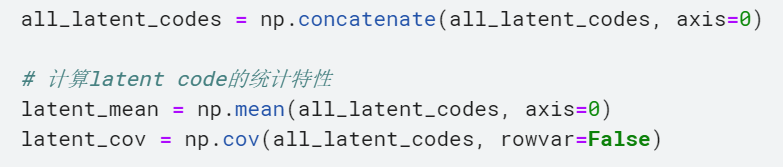


1. 随机选取256张图片，通过构造的encoder得到对应的latent code，计算这些latent code的统计 特性，并以此为参数构造高斯分布。试在所得到的高斯分布上进行9次随机采样后，将这9组latent code送入decoder，观察所得图像并描述。

获取256张图片的latent code：



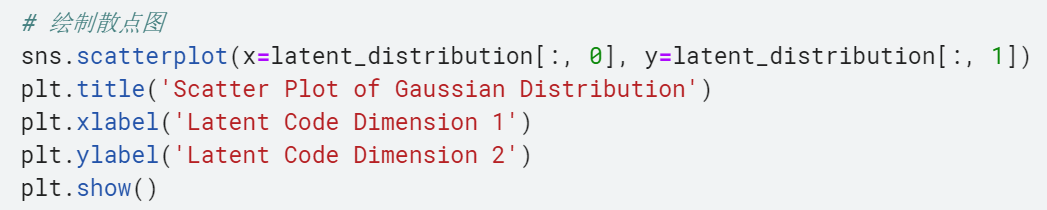
将latent code合并为一个数组并计算其统计特性：



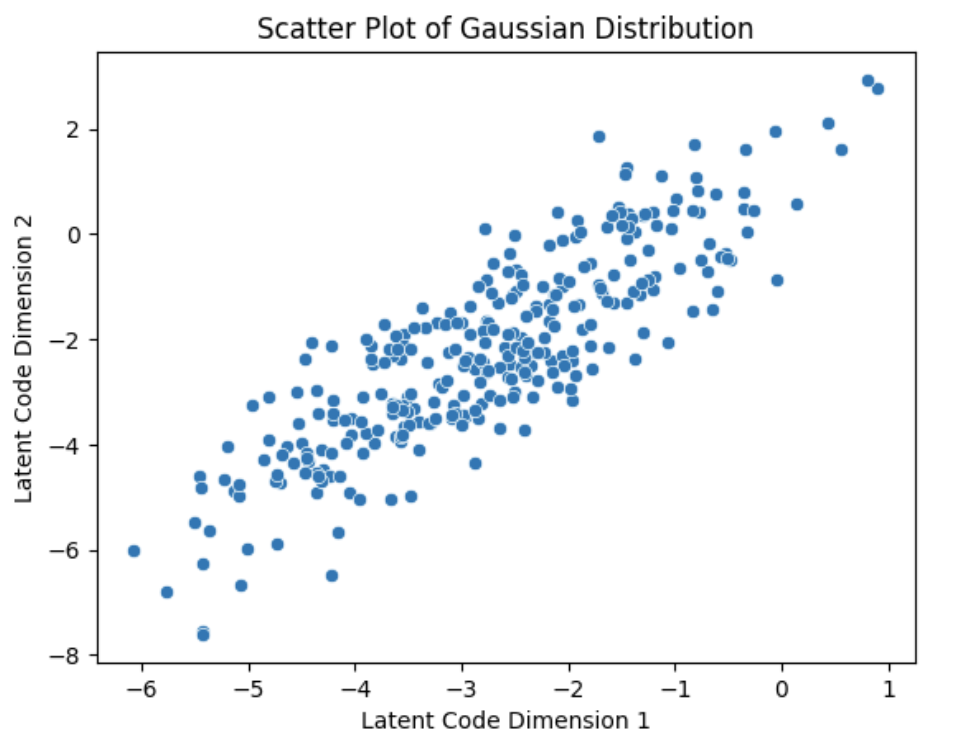
构造高斯分布并进行采样：



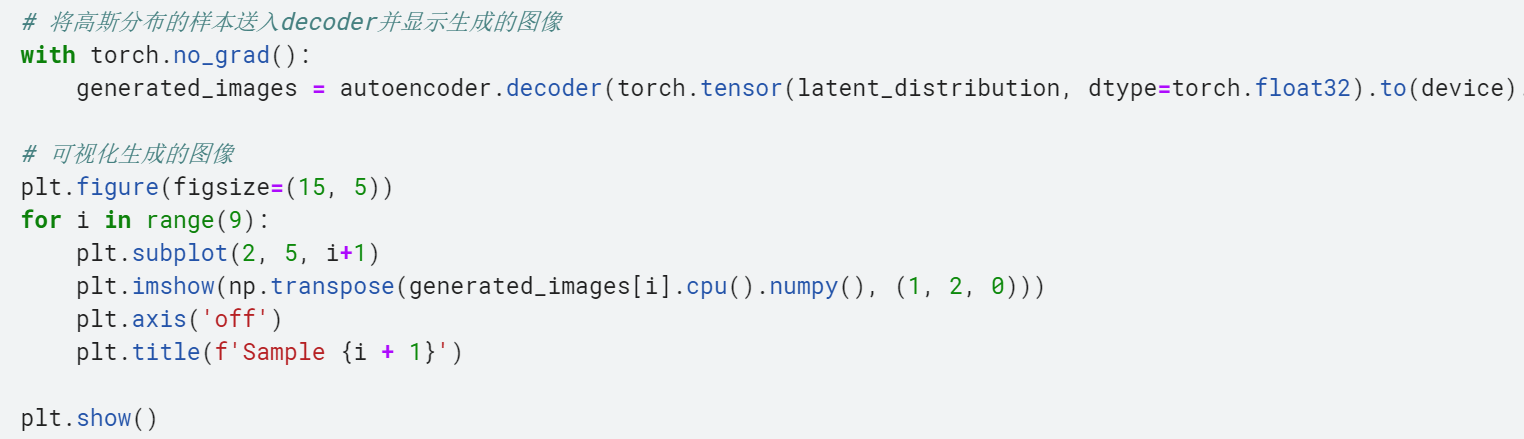
绘制高斯分布的散点图：



绘制结果：



将高斯分布的样本送入decoder并显示生成的图像：



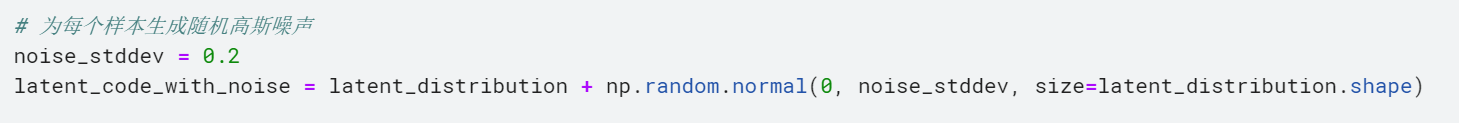
显示结果：



所得图像依稀可见原图的形状，但出现了很高的模糊。

1. 在（3）的基础上，在这9张图片的latent code上叠加随机的高斯噪声扰动，然后送入 decoder，观察现象并解释。

为每个样本生成随机高斯噪声：



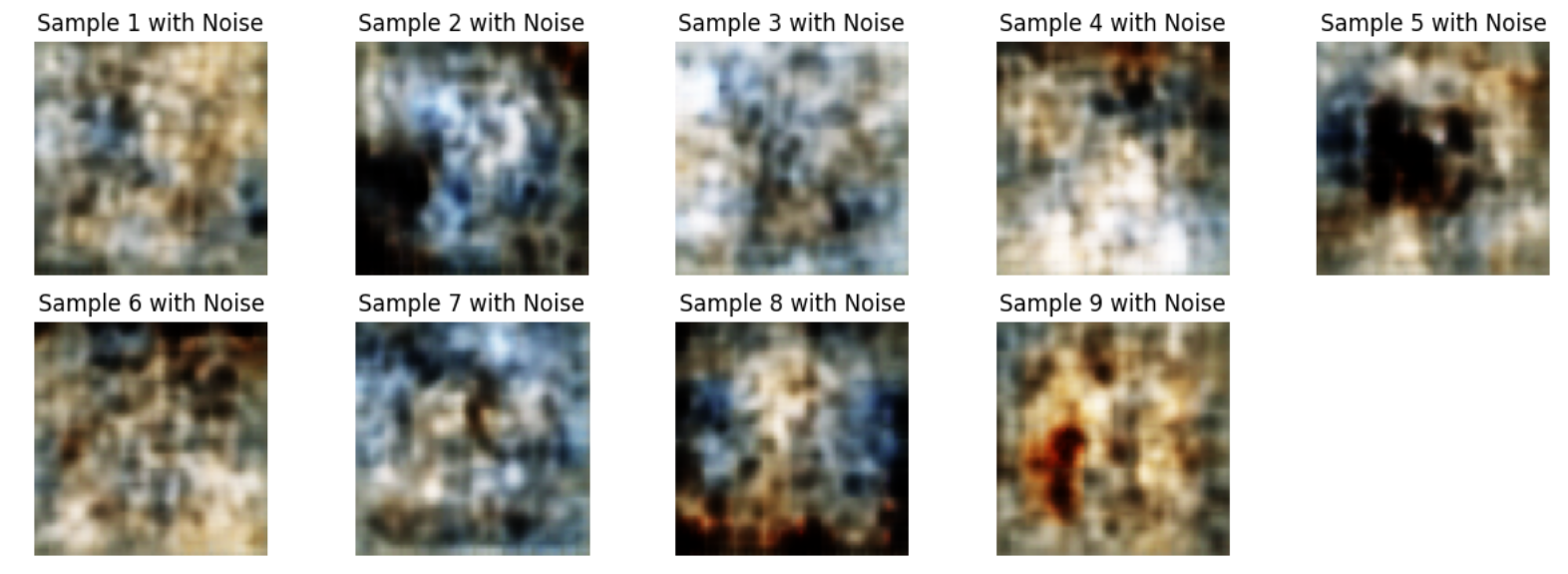
将改动后的latent code送入decoder：



可视化生成的图像：



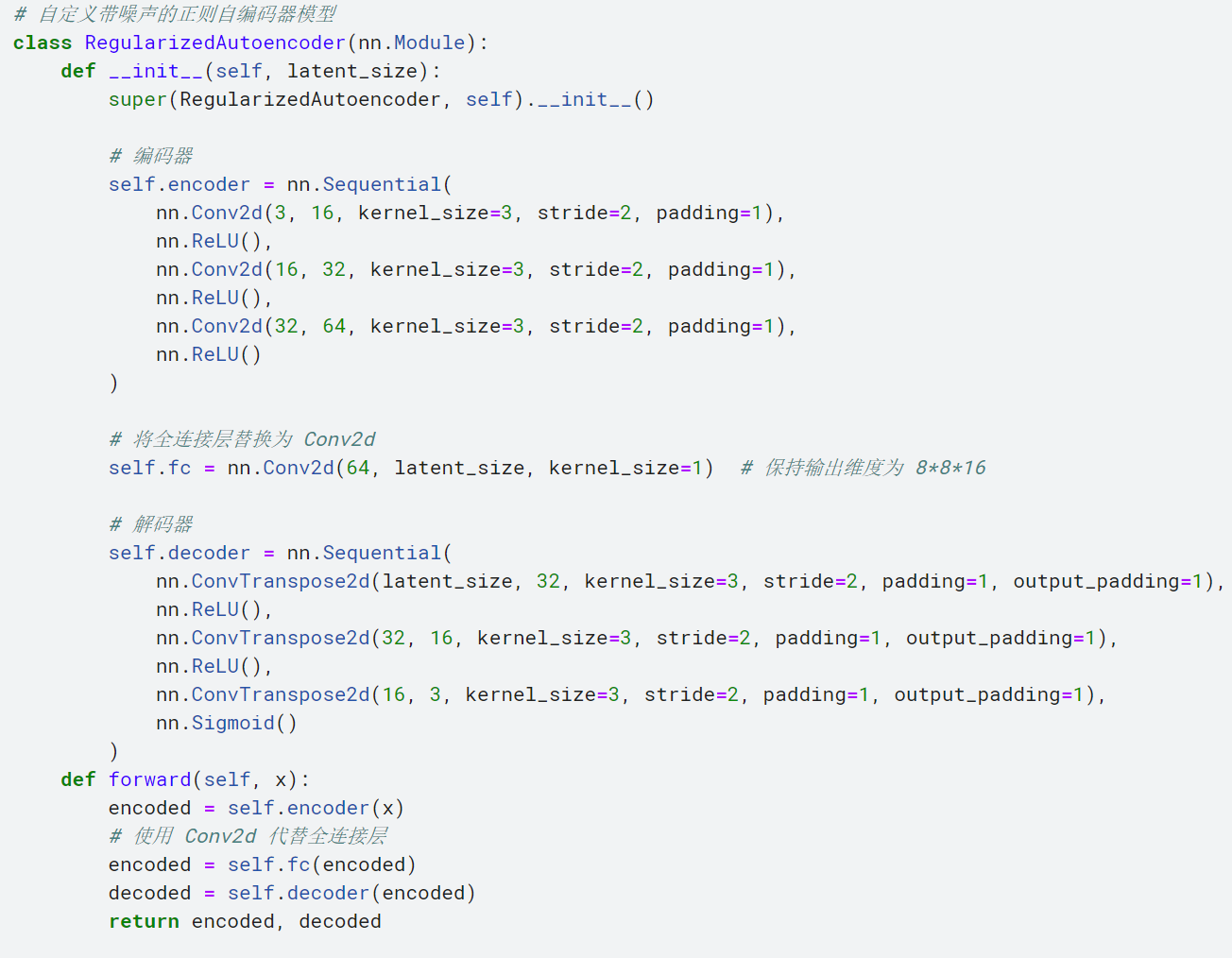
显示结果：



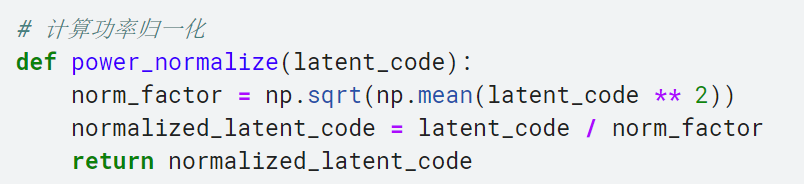
所得图像变得更加模糊，原因是高斯噪声扰动对latent code进行了干扰，导致decoder 在进行解码的时候准确度降低。

1. 将latent code叠加零均值高斯噪声作为一类正则自编码器方法，带噪训练新的正则自编码器，（latentsize为8\*8\*16）。为了保证高斯噪声具有稳定效果，需要在叠加噪声前对latent code进行功率归一化，请在噪声方差分别为0.05、0.1、0.15时，给出Dog数据集上重构图像PSNR的平均值，探究此时从latent space采样是否有生成效果。

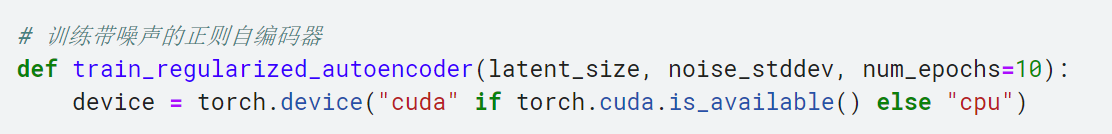
构建带噪声的正则自编码器模型：



计算功率归一化：



训练带噪声的正则自编码器：



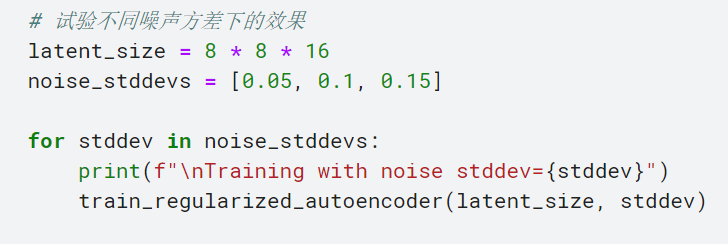
初始化并训练模型：



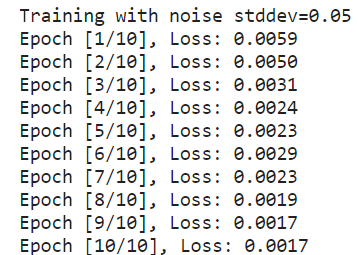
构建测试模型函数并计算平均PSNR：

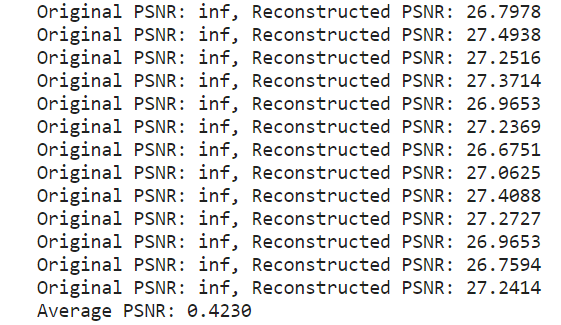


测试不同噪声方差下的效果：

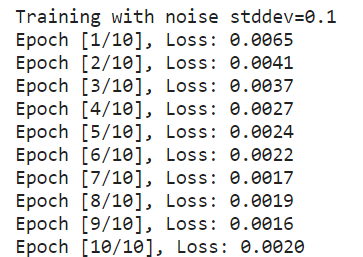


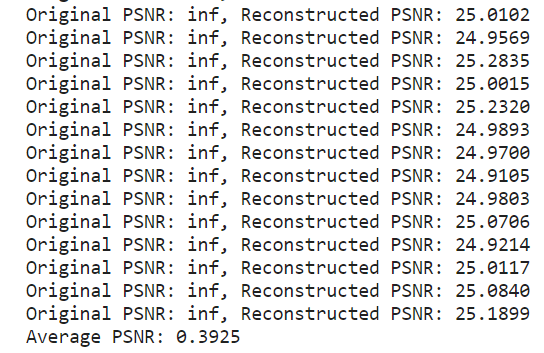
当噪声方差为0.05时：



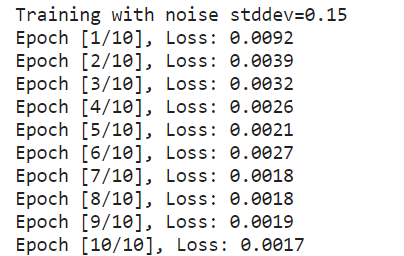


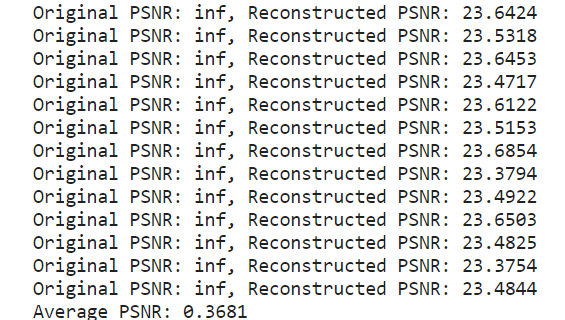
当噪声方差为0.10时：





当噪声方差为0.15时：





此时从latent space采样无生成效果。