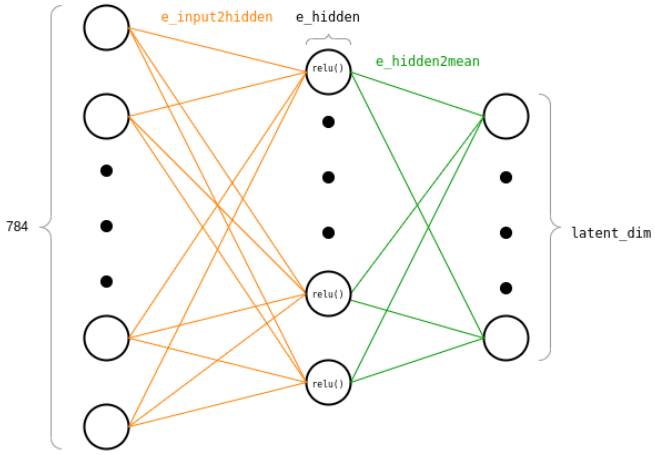
# 机器学习 第八次作业 （上机实践）

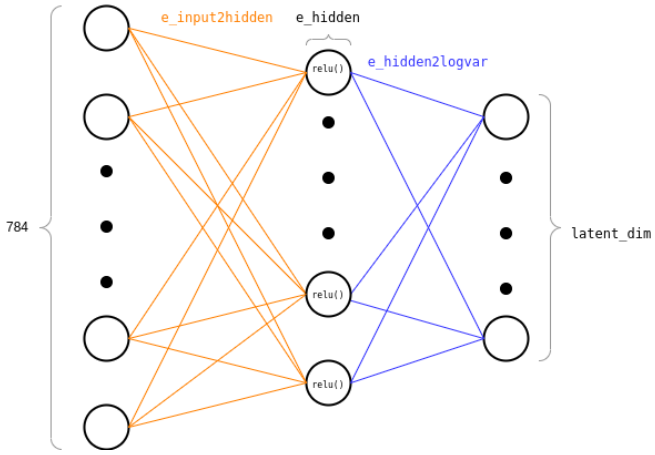
本次作业利用 pytorch 实现了一个变分自编码器，对 MNIST 手写数字数据集进行识别。源代码见另外的文件。

## 变分自编码器的结构

### 编码器的结构

其中编码器的结构如下所示：

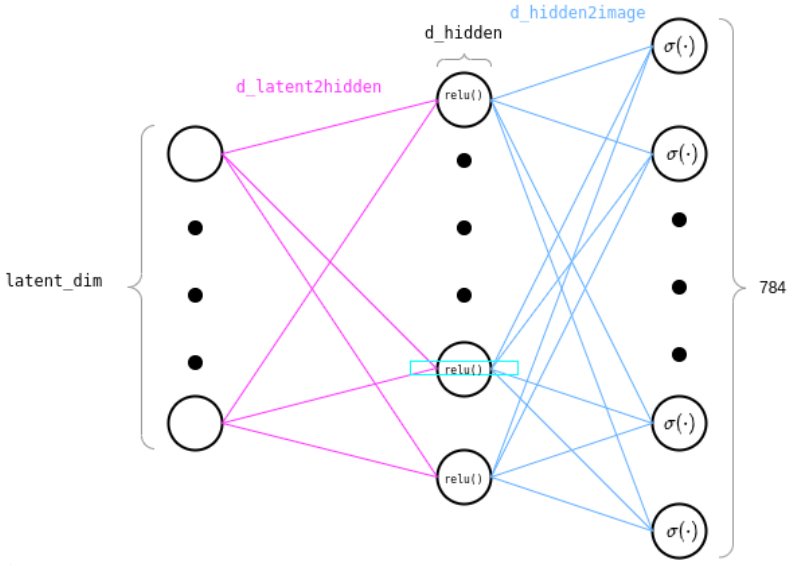



这个编码器有一个隐层和两个输出层，这两个输出层代表的就是隐空间。输入层和隐层，以及隐层与两个输出层之间，都是全连接的。权重矩阵已在图中标明：$e\_input2hidden$，$e\_hidden2mean$，$e\_hidden2logvar$，其中隐层使用激活函数是$ReLU$。为何要设置两个输出层呢？其实在这个编码器中，我们其实是将输入从[0,255]归一化到了(0,1)，并可以将这些值视为概率值。通过编码器，我们可以将输入的 784 个归一化的像素值，编码（压缩成）隐空间的二维向量。而这个二维向量的均值和方差，就是我们输出层输出的隐空间向量。故输出层有两个，一个是均值，一个是方差。

### 解码器的结构

解码器的结构如下页图所示：其亦分为三层，不过输出层仅有一个，就是用于生成与图像 784 个像素值有关的数据的那一层。至于解码器的输入层，我们采取在隐空间表示中采样的办法。我们首先在隐空间采样得到隐变量，然后将$sigmoid$函数作用于其上，得到结果之后通过权重矩阵传递给隐藏层，在隐藏层中将$ReLU$函数作用于这些数据上。之后通过第二个权重矩阵传递给输出层，再次将$sigmoid$函数作用于其上得到维数为 784 的向量。




# 训练过程概述

现在我们可以开始训练这个 vae 了。首先我们需要定义损失函数。我们可以通过最大化证据下界来最小化我们的损失函数，如下：

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi}[\log p_\theta(x|z)] - KL(q_\phi(z|x)||p_\theta(z))$$

其中第一项为对数几率的期望，第二项为正则化项。论文(https://arxiv.org/pdf/1312.6114.pdf)指出：当$q_\phi$和$q_\phi(z|x)$是高斯分布的时候，

$$-KL(q_\phi(z|x)||p_\theta(z)) = \frac{1}{2}\sum_{j=1}^{J}\left[1 + \log\sigma^2 - \mu_j^2 - \sigma_j^2\right]$$

其中$J$是隐空间表示向量$z$的维数，而本次作业中，其值为 2。

故对于损失函数的一个估计可表示为如下：

$$\tilde{\mathcal{L}}_{\theta,\phi}(x) = \frac{1}{L}\sum_{i=1}^{L}\left[\log p_\theta(x \mid g_\phi(\epsilon^{(i)}, x))\right] - \frac{1}{2}\sum_{j=1}^{J}\left[1 + \log\sigma^2 - \mu_j^2 - \sigma_j^2\right]$$

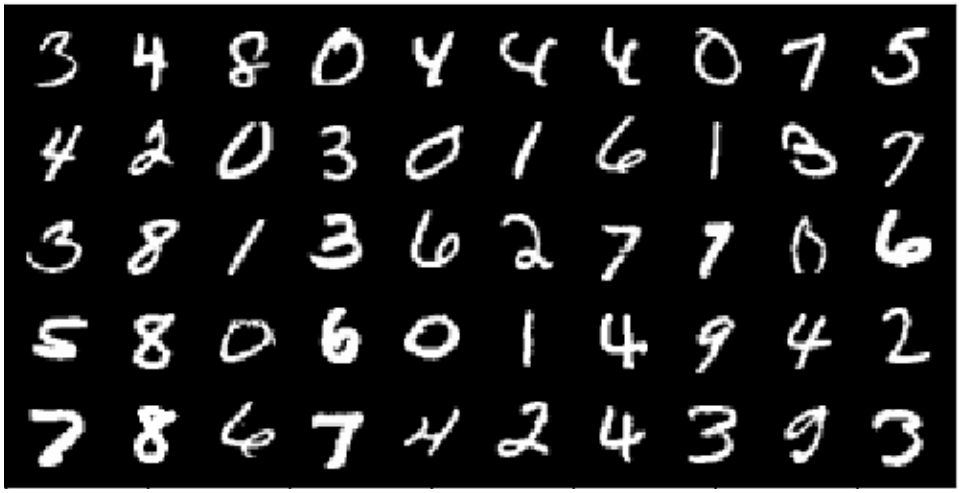
其中，$\epsilon^{(i)}$独立同分布于$\mathcal{N}(0, I)$，$L$表示训练样本数，$J$表示隐空间维数，即 2。

且$\log p_\theta(x|z) = \sum_j x_j \log p_j + (1 - x_j)\log(1 - p_j)$。

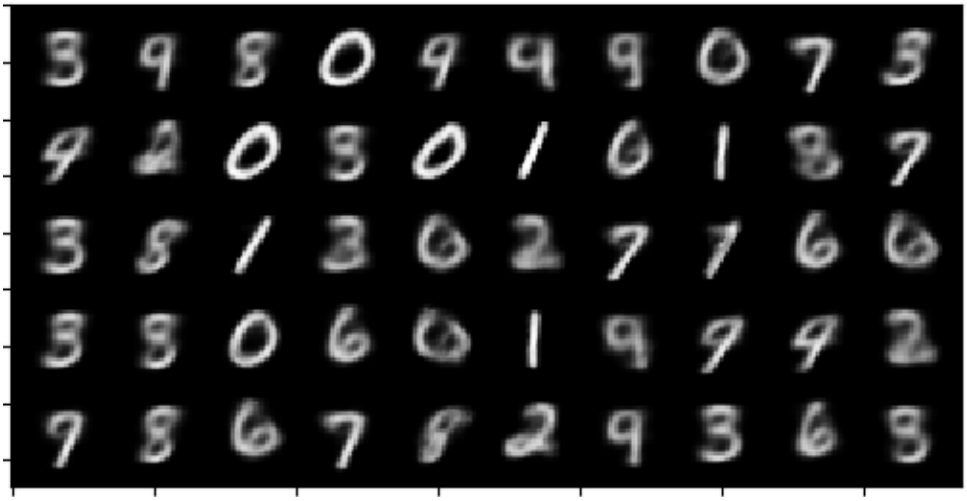
有了上述的损失函数，我们就可以进行训练了。我们编写好 vae 这个 Module 之后，便可以通过 pytorch 进行设置，对我们的数据进行训练。

## 实验结果

在结果展示中，我们首先展示一组原始图像和重建图像，通过对比来展示我们这个变分自编码器的效果。



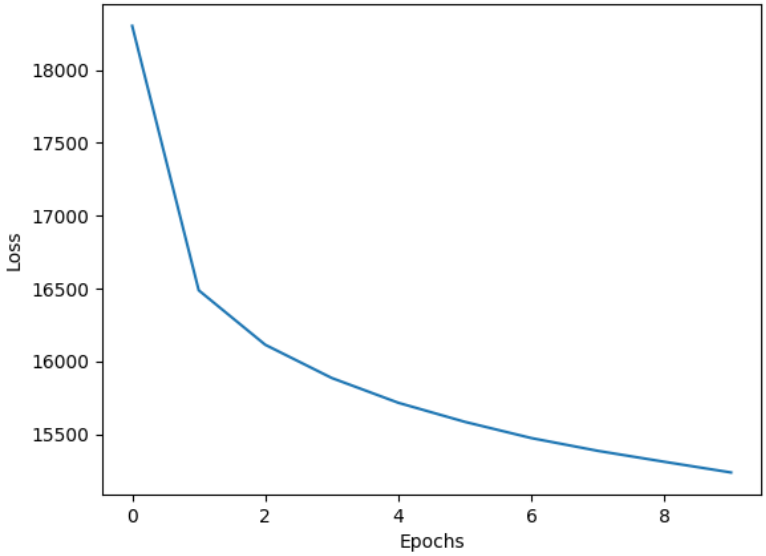
原始图像



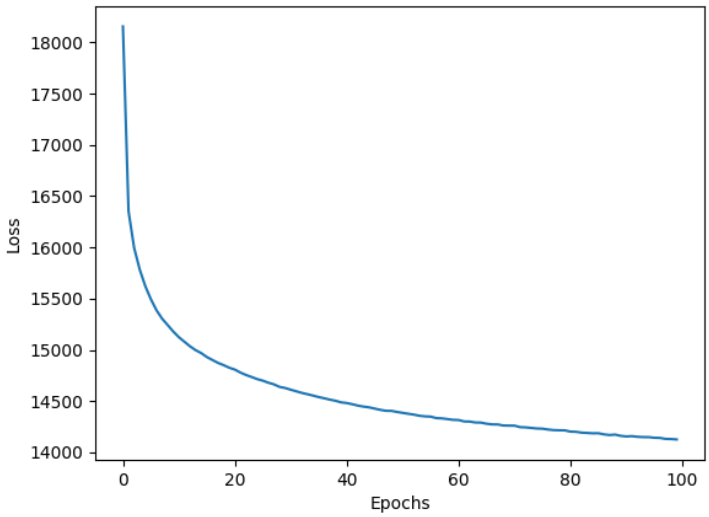
重建的图像

可以看出通过重建的图像相比于原图，有不少数字是和原数字不同的。最典型的有 9 和 4，5 和 8，3 和 8。以及还有一些离谱的 6 和 9 等。

事实上将训练的次数(Epochs)由 10 改成 100 也并不会使得重建的效果更好一些。Epochs=10 和 100 时，误差下降的曲线如下页图所示。可以看出 epochs 有 10 变成 100 时，损失函数在数值上下降了 1000 左右，相比于原始值的量级，仅有 6 ~ 7%左右。而前十个 Epoch 就可以使损失函数下降 20 左右，说明越到后面训练越困难。



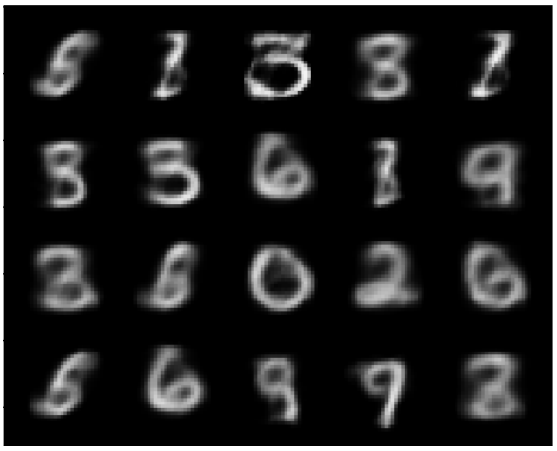

Epochs=10 时的误差下降曲线



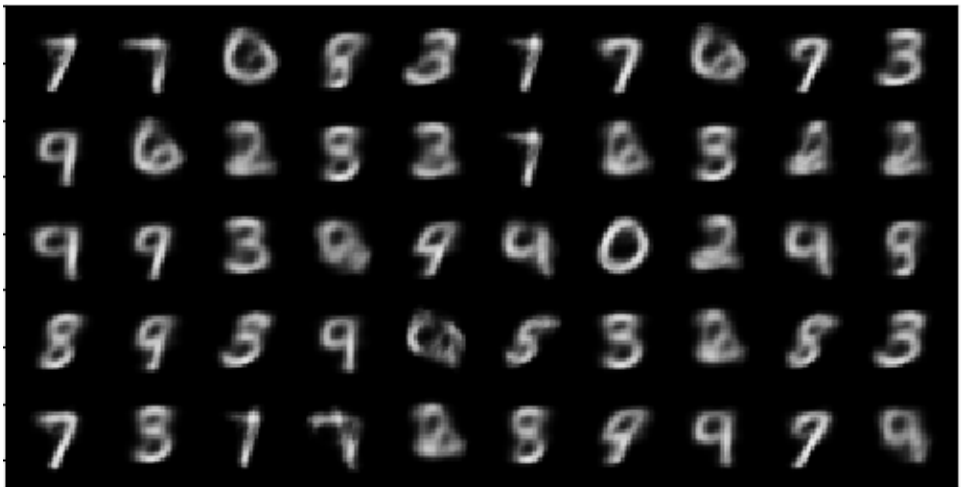

Epochs=100 时的误差下降曲线

接下来展示作业中要求展示的部分。

1. 使用随机的隐变量，生成 20 幅新的图像。如下页图所示。事实上 20 个图像显得模糊不行，很多图像效果并不明显，所以展示 50 个图像显得更佳，如下页图所示。



生成的 20 个图像



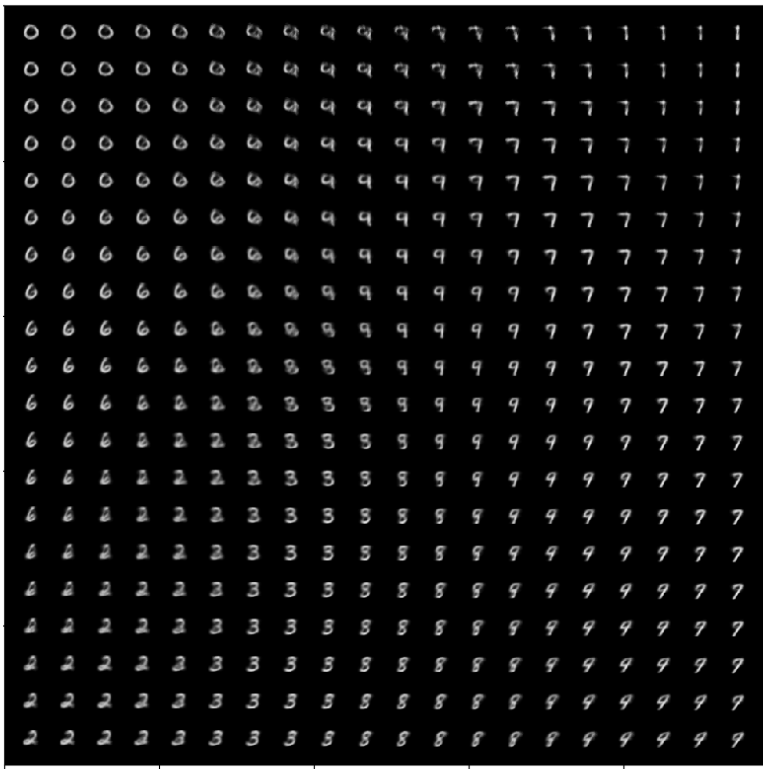
生成的 50 个图像

2. 隐空间插值。

这一题目要求选取两个不同的数字生成 20 个中间隐变量并生成对应的图像。而本任务中涉及到隐空间的向量为 2 维。如果对每个变量都进行线性等距插值，则组合之后可以对应于隐空间的 400 个向量，即可以生成 400 个数字图像，如下页图所示。若只需要看 20 个图像，可沿着对角线观看。

选取的两个数字可视为处于对角线的数字。例如左上的 0 和右下的 9，还有右上的 1 和左下的 2。

以左上的 0 和右下的 9 为例，我们可沿着三个方向观察。观察第一列，则是 0 逐渐变成 6，再逐渐变成 2 的过程。观察第一行，则是 0 由一个中间状态主角变成 1 的过程。沿着对角线，则是 0 逐渐形似于 6，再逐渐形似于 8，在逐渐变成 9 的过程。



隐空间插值图像