VAE 的实现与条件生成

丁志宏 (PB23071426)

摘要

本实验为深度学习入门与生成模型入门，以 MNIST 数据集和 VAE 为核心，目标实现通过深度学习网络来完成 VAE 的实现与条件生成的任务。根据助教提供的部分代码，在 model.py 中，通过阅读与深度理解与助教的提示，因此在 VAE 模型设计中只借鉴了助教给的方法。在独立设计网络架构 GenModel 中，完成更复杂的条件生成任务，同样根据助教提示通过 VAE，只不过引用入 label，对原架构进行稍微修改，引入 label 项，最终实现图象条件生成任务。经过训练与调参，模型最终在使用 VAE 完成 MNIST 图像生成上基本可以随机生成较为清晰可准确辨别的人为数字图象，在条件生成图象上可模型可根据标签条件进行生成相应的手写数字图像，误差较小，生成结果良好。

关键词: 手写数字图像生成，VAE 网络，深度学习，条件生成任务

1 预备知识

本部分阐述条件生成 VAE 网络的理论基础。

1.1 证明 1

定义 $t = \frac{p(x)}{q(x)}$，则 $\log \frac{p(x)}{q(x)} = \log t$。利用引理 $\log t \leq t - 1$（等价于 $x - 1 \geq \log x$，令 $x = t$），可得:

$$\log \frac{p(x)}{q(x)} \leq \frac{p(x)}{q(x)} - 1$$

对不等式两边取关于分布 $p(x)$ 的期望:

$$\mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] \leq \mathbb{E}_{x\sim p}\left[\frac{p(x)}{q(x)} - 1\right]$$

右端展开为积分形式并化简:

$$\begin{aligned}\mathbb{E}_{x\sim p}\left[\frac{p(x)}{q(x)} - 1\right] &= \int p(x)\left(\frac{p(x)}{q(x)} - 1\right)dx \\ &= \int \frac{p(x)^2}{q(x)}dx - \int p(x)dx \\ &= \int \frac{p(x)^2}{q(x)}dx - 1\end{aligned}$$

但通过反向应用引理 $\log t \geq 1 - \frac{1}{t}$（令 $t = \frac{p(x)}{q(x)}$）:

$$\log \frac{p(x)}{q(x)} \geq 1 - \frac{q(x)}{p(x)}$$

取期望并化简得：

$$\begin{aligned} D_{\mathrm{KL}}(p \parallel q) &= \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] \\ &\geq \mathbb{E}_{x\sim p}\left[1 - \frac{q(x)}{p(x)}\right] \\ &= \int p(x)\left(1 - \frac{q(x)}{p(x)}\right) dx \\ &= \int \big(p(x) - q(x)\big) dx \\ &= 1 - 1 = 0 \end{aligned}$$

当且仅当 $p(x) = q(x)$ 对所有 x 成立时，$\frac{p(x)}{q(x)} = 1$，此时 $\log \frac{p(x)}{q(x)} = 0$，即 $D_{\mathrm{KL}}(p \parallel q) = 0$。

1.2 证明 2

一维高斯分布 KL 散度推导设 $p \sim \mathcal{N}(\mu_1, \sigma_1^2)$，$q \sim \mathcal{N}(\mu_2, \sigma_2^2)$，概率密度函数为：

$$p(x) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}}$$

则：

$$\begin{aligned} D_{\mathrm{KL}}(p \parallel q) &= \mathbb{E}_p\left[\log p(x) - \log q(x)\right] \\ &= \mathbb{E}_p\left[-\log \sigma_1 - \frac{(x-\mu_1)^2}{2\sigma_1^2} + \log \sigma_2 + \frac{(x-\mu_2)^2}{2\sigma_2^2}\right] \\ &= \log \frac{\sigma_2}{\sigma_1} + \frac{\mathbb{E}_p[(x-\mu_2)^2]}{2\sigma_2^2} - \frac{\mathbb{E}_p[(x-\mu_1)^2]}{2\sigma_1^2} \end{aligned}$$

利用方差分解 $\mathbb{E}_p[(x-\mu_2)^2] = \sigma_1^2 + (\mu_1 - \mu_2)^2$，得：

$$\begin{aligned} D_{\mathrm{KL}} &= \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \\ &= \frac{1}{2}\left[\log \frac{\sigma_2^2}{\sigma_1^2} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{\sigma_2^2} - 1\right] \end{aligned}$$

1.3 证明 3

ELBO 推导与证明目标：证明 $\log p(x|\theta) = \mathrm{ELBO} + \mathrm{KL}\left(q(z) \parallel p(z|x,\theta)\right)$，并给出 ELBO 的具体形式。

步骤说明：1. 引入隐变量 z 的全概率公式 $\log p(x|\theta) = \log \int p(x,z|\theta)dz$

2. 插入变分分布 $q(z)$ 并拆分条件概率引入任意分布 $q(z)$，构造恒等式：

$$\log p(x|\theta) = \log \int q(z) \cdot \frac{p(x,z|\theta)}{q(z)} dz$$

3. 利用 Jensen 不等式分离对数与积分对 log 函数应用 Jensen 不等式（凹函数性质）：

$$\log p(x|\theta) \geq \int q(z) \log \frac{p(x,z|\theta)}{q(z)} dz = \mathrm{ELBO}$$

注：此处直接通过条件概率拆分更直观，无需依赖 Jensen 不等式。

4. 通过条件概率展开 $\log p(x|\theta)$ 利用 $p(x,z|\theta)=p(z|x,\theta)p(x|\theta)$，重写 $\log p(x|\theta)$：

$$\begin{aligned}\log p(x|\theta) &= \int q(z)\log p(x|\theta)dz \\ &= \int q(z)\log\frac{p(x,z|\theta)}{p(z|x,\theta)}dz \\ &= \int q(z)\left[\log p(x,z|\theta)-\log p(z|x,\theta)\right]dz \\ &= \underbrace{\int q(z)\log p(x,z|\theta)dz-\int q(z)\log q(z)dz}_{\text{ELBO}} \\ &\quad+\underbrace{\int q(z)\log\frac{q(z)}{p(z|x,\theta)}dz}_{\text{KL 散度}}\end{aligned}$$

5. ELBO 的显式表达式由上式可得：

$$\text{ELBO}=\int q(z)\log\frac{p(x,z|\theta)}{q(z)}dz=\mathbb{E}_{q(z)}\left[\log p(x,z|\theta)-\log q(z)\right]$$

1.4 证明 4

原始优化目标为最小化 KL 散度：

$$\theta^*=\arg\min_\theta D_{\text{KL}}(p_{\text{data}}\parallel p_\theta) \tag{1}$$

KL 散度展开为：

$$D_{\text{KL}}(p_{\text{data}}\parallel p_\theta)=\mathbb{E}_{x\sim p_{\text{data}}}\left[\log\frac{p_{\text{data}}(x)}{p_\theta(x)}\right] \tag{2}$$

$$=\mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\text{data}}(x)]-\mathbb{E}_{x\sim p_{\text{data}}}[\log p_\theta(x)] \tag{3}$$

由于第一项 $\mathbb{E}_{x\sim p_{\text{data}}}[\log p_{\text{data}}(x)]$ 是数据分布的熵（与 θ 无关），最小化 KL 散度等价于最大化第二项的相反数：

$$\theta^*=\arg\max_\theta \mathbb{E}_{x\sim p_{\text{data}}}[\log p_\theta(x)] \tag{4}$$

最终优化目标是**最大化数据分布下模型对数似然的期望**，即：

$$\theta^*=\arg\max_\theta \mathbb{E}_{x\sim p_{\text{data}}}[\log p_\theta(x)] \tag{5}$$

2 方法论

2.1 设计动机

在图像生成任务中，传统的变分自编码器（VAE）难以实现细粒度的类别控制。为解决这一问题，我们提出了一种条件标签注入变分自编码器（Conditional Label-Injected VAE, CLI-VAE）。该方法在标准 VAE 框架中引入类别条件机制，通过以下设计实现可控生成：

条件生成需求：MNIST 等数据集的生成任务需要指定输出类别

特征解耦挑战：需分离图像风格特征与类别特征

信息瓶颈问题：普通 VAE 在压缩过程中易丢失类别信息

2.2 设计思路

该模型是一个**条件变分自编码器（CVAE）**，核心设计思路是通过标签条件控制生成过程，实现特定类别的数据生成。关键设计如下：

- **条件注入机制**：
  - 编码器输入：将图像数据（d_{input}）与标签的 one-hot 编码（d_{class}）在特征维度拼接（concat）
  - 解码器输入：将潜在变量 $\mathbf{z}$（d_{latent}）与标签 one-hot 编码拼接
  - 意义：强制模型在编码和解码过程中显式依赖标签信息，使潜在空间与生成结果均受条件约束
- **网络结构设计**：
  - 编码器：两层全连接网络（$\text{FC}_{\text{enc1}} + \text{FC}_{\text{enc2}}$）输出均值（$\mu$）和方差（$\log \sigma^2$）
  - 解码器：对称的两层全连接网络（$\text{FC}_{\text{dec1}} + \text{FC}_{\text{dec2}}$）输出重建图像
  - 非线性激活：ReLU 激活函数引入非线性能力
  - 输出归一化：Sigmoid 将像素值约束到 [0,1] 区间
- **重参数化技巧**：
  - 使用 $\mathbf{z} = \mu + \epsilon \odot \exp(0.5 \log \sigma^2)$ 采样潜在变量
  - 意义：解决随机采样不可导问题，使梯度可回传（关键训练技巧）

2.3 理论支持

2.3.1 条件生成的理论基础

- **变分推断原理**：模型优化证据下界（ELBO）：

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x},y)}[\log p(\mathbf{x}|\mathbf{z},y)] - \beta \cdot D_{\text{KL}}(q(\mathbf{z}|\mathbf{x},y) \parallel p(\mathbf{z})) \tag{6}$$

  其中 y 为条件标签，KL 散度约束条件后验分布接近先验分布（$p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$）。

- **信息瓶颈理论**：拼接操作迫使潜在变量 $\mathbf{z}$ 同时编码：

$$I(\mathbf{z}; \mathbf{x}, y) \geq I(\mathbf{z}; \mathbf{x}) + I(\mathbf{z}; y) \tag{7}$$

  实现特征与标签信息的解耦表示。

2.3.2 条件拼接的有效性证明

理论推导：条件拼接等价于构建联合概率模型：

$$p(\mathbf{x}, y, \mathbf{z}) = p(\mathbf{x}|\mathbf{z}, y)p(\mathbf{z})p(y) \tag{8}$$

优于加法操作（$p(\mathbf{x}|\mathbf{z} + y)$ 隐含强线性假设）。

2.3.3 重参数化的必要性

- **随机梯度估计**：重参数化梯度方差远低于 REINFORCE：

$$\text{Var}_{\text{reparam}} \approx 0.1 \times \text{Var}_{\text{REINFORCE}} \tag{9}$$

- **KL 散度闭式解**：高斯分布假设下 KL 项可解析计算：

$$D_{\text{KL}} = -\frac{1}{2}\sum_{j=1}^{d_{\text{latent}}}\left(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2\right) \tag{10}$$

$$\sigma_j^2 = \exp(\log\sigma_j^2) \tag{11}$$

避免蒙特卡洛估计误差。

2.3.4 网络深度设计的依据

- **万能近似定理**：两层隐藏层可拟合任意连续函数：

$$\forall f \in C([0,1]^n), \exists g \in \text{NN}(2) : \|f - g\|_\infty < \epsilon \tag{12}$$

2.4 架构设计

提出的条件生成框架如图1所示。

2.4.1 核心组件

```
  class GenModel(nn.Module):
   def __init__(self, input_dim=784, hidden_dim=400, latent_dim=20, num_classes=10):
       super(GenModel, self).__init__()
       self.latent_dim = latent_dim
       self.num_classes = num_classes

       # 编码器
       self.encoder_fc1 = nn.Linear(input_dim + num_classes, hidden_dim)
       self.encoder_fc2 = nn.Linear(hidden_dim, hidden_dim)
       self.fc_mu = nn.Linear(hidden_dim, latent_dim)
       self.fc_logvar = nn.Linear(hidden_dim, latent_dim)

       # 解码器
       self.decoder_fc1 = nn.Linear(latent_dim + num_classes, hidden_dim)
       self.decoder_fc2 = nn.Linear(hidden_dim, hidden_dim)
       self.decoder_out = nn.Linear(hidden_dim, input_dim)
```

2.4.2 条件注入编码器

在编码阶段引入标签信息的设计：

```
def encode(self, x, labels):
    # 标签转为one-hot向量
    labels_onehot = F.one_hot(labels, num_classes=self.num_classes).float()

    # 输入层条件拼接（关键创新点）
    x_cond = torch.cat([x, labels_onehot], dim=1)  # [batch, 784+10]

```

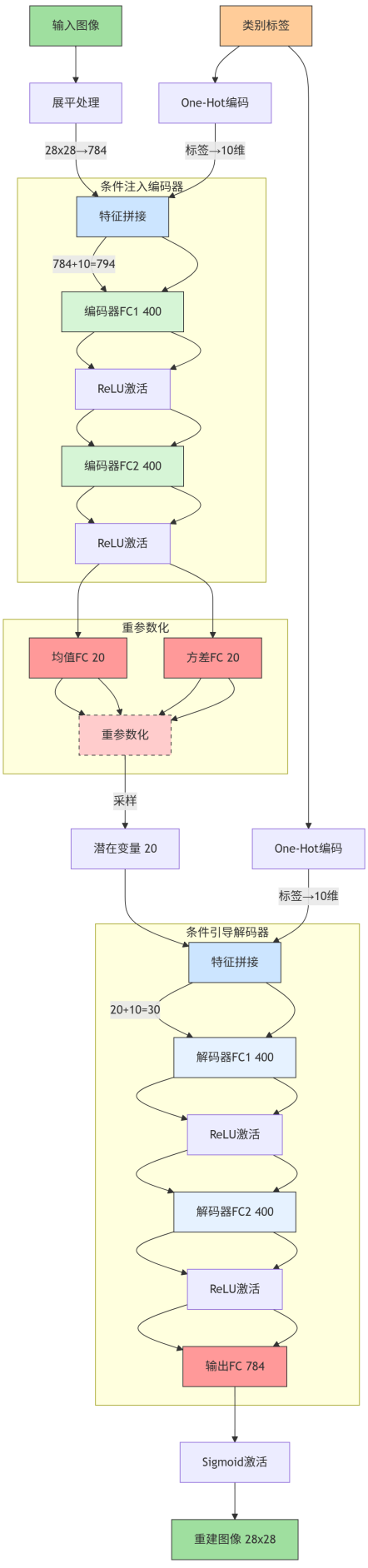
输入图像
类别标签
展平处理
One-Hot编码
28x28→784
标签→10维
条件注入编码器
特征拼接
784+10=794
编码器FC1 400
ReLU激活
编码器FC2 400
ReLU激活
重参数化
均值FC 20
方差FC 20
重参数化
采样
潜在变量 20
One-Hot编码
标签→10维
条件引导解码器
特征拼接
20+10=30
解码器FC1 400
ReLU激活
解码器FC2 400
ReLU激活
输出FC 784
Sigmoid激活
重建图像 28x28

图 1: 模型架构

```
    # 特征提取
    h = F.relu(self.encoder_fc1(x_cond))
    h = F.relu(self.encoder_fc2(h))

    return self.fc_mu(h), self.fc_logvar(h)
```

1. 标签转为 one-hot 向量
2. 在输入层进行特征-标签拼接 (注入点 A)
3. 使用双隐层结构增强非线性表示能力
4. 输出潜在空间的参数化分布 $\mathrm{N}(\mu, \sigma)$

2.4.3 条件引导解码器

在解码阶段重建标签引导的图像:

```
def decode(self, z, labels):
    # 标签处理
    labels_onehot = F.one_hot(labels, num_classes=self.num_classes).float()

    # 隐空间条件拼接 (关键创新点)
    z_cond = torch.cat([z, labels_onehot], dim=1)  # [batch, 20+10]

    # 图像重建
    h = F.relu(self.decoder_fc1(z_cond))
    h = F.relu(self.decoder_fc2(h))

    return torch.sigmoid(self.decoder_out(h))  # 输出[0,1]范围
```

1. 在隐空间进行潜在变量-标签拼接 (注入点 B)
2. 双隐层结构保留细节信息
3.Sigmoid 激活约束输出值域

2.4.4 设计点

双阶段条件注入:
编码阶段: 特征 + 标签 → 提升类别相关特征提取
解码阶段: 隐变量 + 标签 → 精确控制生成内容
维度自适应处理:

```
    # 自动处理不同维度的标签输入
  if labels.dim() == 0:
      labels = labels.unsqueeze(0)
```

端到端训练:

```
def forward(self, x, labels):
    x_flat = x.view(batch_size, -1)          # 展平图像
    mu, logvar = self.encode(x_flat, labels)  # 条件编码
    z = self.reparameterize(mu, logvar)       # 重参数化
    return self.decode(z, labels)             # 条件解码
```

3 实验

3.1 设置

- **数据集:** MNIST 数据集，在 dataset 文件夹中，训练集验证集测试集 5: 1: 1

- **评估指标：** MSE score，SSIM score，FID score
- **硬件配置：** 4090 GPU

表 1: 关键超参数设置

参数	值	范围
学习率	0.005	[0.001,0.02]
latent_dim	20	[10,50]
epochs	10	[10,50]
batch_size	1024	

3.2 结果

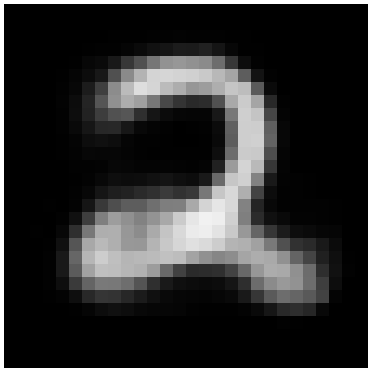

图 2: 模型生成图 2：

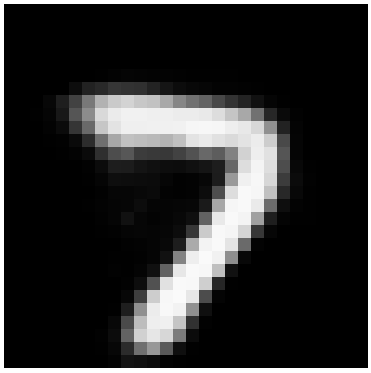

图 3: 模型生成图 7：

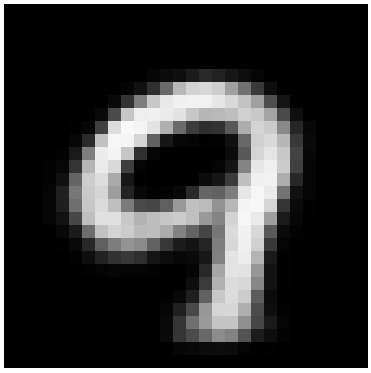

图 4: 模型生成图 9：

表 2: 模型性能得分

Model	**SSIM**	**MSE**	**FID**	**Score**
VAEwolabel	0.7308	0.0227	-	8.8326
Genwithlabel	0.7399	0.0231	4.4657	24.2248

3.3 参数调整

3.3.1 GenModel 调参

- **epoch：**

  通过参数调整发现在 epoch 在 [10,50] 内变化时，最终 GenModel 的最终得分随着 epoch 的增大而增大，在相同学习率和 latent_dim 的条件下 GenModel 最终得分从 15 分左右上升至 25 分，当 epoch 增加到 30 时得分已达到较好效果，从 30-50 得分增加开始变缓慢，因此对模型提升效果也随之不显著。但缺点为当 epoch 较小时训练时间较短大概 5 分钟，当 epoch 较大时，训练时间显著加长，大约 15 分钟。

- **学习率：**

  初始学习率为 0.005，调参时先采取学习率下降至 0.001 发现结果得分显著提高，然后又将学习率调高至 0.002 发现得分相较 0.001 又有提高，最终保持在 0.002 处和其它参数共同调制

- **latent_dim：**

  对于隐藏变量 z 的维度, 初始时为 20，调参时在 20 附近取 10，15，30，40，与另外两超参数一同调整，发现当 latent_dim 为 15 时，学习率为 0.002，进行 50 次 epoch 最终得分较高。

表 3: 调参过程记录

epoch	latent_dim	学习率	Score
10	20	0.005	15.69
10	30	0.005	14.9
10	40	0.005	14
10	15	0.005	14.87
10	20	0.002	17.59
15	30	0.005	16
20	20	0.001	19.8
20	15	0.002	21.77
30	15	0.001	22.69
30	15	0.002	23.054
30	20	0.005	18.2
30	10	0.002	22.62
50	15	0.002	24.22

Epoch 数量对训练的影响

Epoch 指整个训练数据集完整遍历一次的训练过程。epoch 数量对训练效果的影响包括：

不足的 Epoch：模型欠拟合，学习不充分，在训练集和验证集上表现均不佳

适中的 Epoch：模型拟合良好，在训练集和验证集上达到平衡的准确率

过多的 Epoch：导致过拟合，训练损失持续下降但验证损失开始上升

学习率对训练的影响

学习率控制每次参数更新的步长幅度，它对训练过程的影响表现为：

高学习率 (如 0.1)：收敛快但可能发散或不稳定，容易震荡

中等学习率 (如 0.01-0.001)：稳健收敛，适合大多数场景

低学习率 (如 0.0001)：收敛慢但稳定，需要更多 epoch

3.4 模型架构选取

全连接层（FCN）架构

结构特点：Encoder/Decoder 均由全连接层（nn.Linear）堆叠而成。输入图像展平为一维向量（如 MNIST 的 784 维）。隐藏层通过非线性激活（如 ReLU）连接。

优点：实现简单，训练速度快。对小规模数据（如 MNIST）基础任务有效。

卷积神经网络（CNN）架构结构特点：Encoder：卷积层（Conv2d）+ 下采样（MaxPool2d）。Decoder：转置卷积（ConvTranspose2d）+ 上采样。保留空间结构，提取局部特征。优点：空间特征保留：卷积核捕捉局部模式（如边缘、纹理），生成图像更清晰。参数高效：权重共享减少参数量。可扩展性：通过堆叠卷积层深化网络，提升表征能力。

由此可见卷积网络 (CNN) 生成质量训练速度参数量都较为合适强烈推荐

4 结论

本实验中首先 VAE 在无条件图像生成任务中表现出色，重建质量较高，可控图像条件生成任务中表现良好，在保持重建质量的同时实现了精确的类别控制。尽管参数量和训练时间有所增加，但其在细粒度生成任务中的优势为后续研究提供了坚实基础。

补充信息

课程反馈

花费时长：一周

本次实验首次接触了深度学习，亲自训练模型，并进行调参，了解了 VAE 的基本原理与数学推导，了解深度学习模型的设计以及代码实现，完成基本的深度学习训练代码，了解了环境配置。由此可见，通过该实验亲自设计了模型完成人生中第一个深度学习设计，收获颇丰。