

程序报告

学号：22375080

姓名：杨佳宇轩

一、问题重述

(简单描述对问题的理解，从问题中抓住主干，必填)

=====

本次作业要求实现并比较三种生成式模型：简单自编码器（AE）、变分自编码器（VAE）以及扩散模型（Diffusion Model）。对于简单自编码器，要求探索其将图像嵌入隐空间并还原图像的完整过程；对于变分自编码器，目标是对比 AE，学习 VAE 从分布采样并生成图像的方法；最后编写 Diffusion Model 相关代码，透彻理解扩散模型加噪、去噪的流程

二、设计思想

(所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等... 思考题，非必填)

=====

AE:
TODO: 按照要求实现 `AutoEncoder` 的网络结构
优化方向：可以调整隐藏层维度、增加网络深度或引入 Dropout 这个泽华改善重构质量

VAE:
TODO: 请使用 `paddle` 实现重参数化函数 `self.reparameterize`
TODO: 尝试修改 VAE 的损失函数相关部分,使其生成的效果更好、更清晰。
TODO: 请使用 `paddle` 实现 VAE 中的 KL 散度损失 `kl_div_loss`
方法：在自编码器的基础上，引入高斯先验 $p(z) = N(0, 1)$ ，用编码器输出潜在分布的均值和对数方差
优化方向：可以尝试不同的调整损失函数、KL 权重 β 、改进网络结构或使用更复杂先验实现更连续的潜在空间

Diffusion Model:
TODO: 完成加噪函数 `add_noise(x0, t, eps)`
TODO: 完成去噪函数 `remove_noise(xt, t, eps)`
TODO: 完成损失函数 `loss(x0, t, eps)`
方法：定义正向扩散过程 $q(x_t | x_{t-1})$ 添加高斯噪声，逆向网络 $p_{\theta}(x_{t-1} | x_t)$ 学习去噪还原原始样本
优化方向：可尝试改进噪声调度、调整 U-Net 深度及 attetion 模块

三、代码内容

(能体现解题思路的主要代码，有多个文件或模块可用多个"====="隔开，必填)

=====

AE:
TODO: 按照要求实现 `AutoEncoder` 的网络结构

```
self.encoder = nn.Sequential(
```

```

        # Input: [B, 3, 64, 64]
        nn.Conv2D(in_channels=3, out_channels=16, kernel_size=3, stride=1, padding=1), #
-> [B, 16, 64, 64]
        nn.ReLU(),
        nn.BatchNorm2D(16),
        nn.MaxPool2D(kernel_size=2, stride=2), # -> [B,
16, 32, 32]
        nn.Conv2D(in_channels=16, out_channels=32, kernel_size=3, stride=1, padding=1),#
-> [B, 32, 32, 32]
        nn.ReLU(),
        nn.BatchNorm2D(32),
        nn.MaxPool2D(kernel_size=2, stride=2) # -> [B,
32, 16, 16]
    )
    self.decoder = nn.Sequential(
        # Input: [B, 32, 16, 16]
        nn.Conv2DTranspose(in_channels=32, out_channels=16, kernel_size=2, stride=2), #
-> [B, 16, 32, 32]
        nn.ReLU(),
        nn.BatchNorm2D(16),
        nn.Conv2DTranspose(in_channels=16, out_channels=3, kernel_size=2, stride=2), #
-> [B, 3, 64, 64]
        nn.Sigmoid()
    )

```

VAE:

TODO: 请使用 **paddle** 实现重参数化函数 **self.reparameterize**

```

def reparameterize(self, mu, logvar, eps = None):
    # 变分重参数化: 采样 z = mu + std * eps
    var = paddle.exp(logvar)
    std = paddle.sqrt(var + 1e-8)
    if eps is None:
        eps = paddle.randn(mu.shape)
    return mu + eps * std

```

TODO: 尝试修改 VAE 的损失函数相关部分,使其生成的效果更好、更清晰。

beta = 1.0

learning_rate = 0.0003

num_epochs = 150

batch_size = 64

white_threshold = 0.7

white_region_loss_coef = 1.0

black_region_loss_coef = 0.3

```
cvae = CVAE(3, 64, 32, MAX_DIGIT)
```

TODO: 请使用 paddle 实现 VAE 中的 KL 散度损失 kl_div_loss

```
def kl_div_loss(mu, logvar):  
    #  $KL(N(\mu, \text{var}) \parallel N(0, 1)) = -0.5 * \sum(1 + \log \text{var} - \mu^2 - \text{var})$   
    var = paddle.exp(logvar)  
    kl = -0.5 * paddle.sum(1 + logvar - mu**2 - var, axis=1)  
    return paddle.mean(kl)
```

Diffusion Model:

TODO: 完成加噪函数 add_noise(x0, t, eps)

```
def add_noise(x0, t, eps):  
    # 提取对应步长的系数  
    sqrt_ab = paddle.sqrt(alpha_bar[t])  
    sqrt_omb = paddle.sqrt(1.0 - alpha_bar[t])  
    # 重参数化采样  
    xt = sqrt_ab * x0 + sqrt_omb * eps  
    return xt
```

TODO: 完成去噪函数 remove_noise(xt, t, eps)

```
def remove_noise(xt, t, eps):  
    # 预测噪声  
    noise_pred = predict_noise(xt, t)  
    # 计算均值  
    coef1 = 1.0 / paddle.sqrt(alpha[t])  
    coef2 = beta[t] / paddle.sqrt(1.0 - alpha_bar[t])  
    mean = coef1 * (xt - coef2 * noise_pred)  
    # 计算标准差  
    var = sigma_square[t]  
    std = paddle.sqrt(var)  
    # 重参数化采样  
    x_prev = mean + std * eps  
    return x_prev
```


TODO: 完成损失函数 loss(x0, t, eps)


```
def loss(x0, t, eps):  
    # 先加噪  
    xt = add_noise(x0, t, eps)  
    # 用 UNet 预测噪声  
    noise_pred = predict_noise(xt, t)  
    # 计算 MSE 损失  
    return paddle.mean((eps - noise_pred)**2)
```


四、实验结果


(实验结果，必填)


VAE 生成结果:
















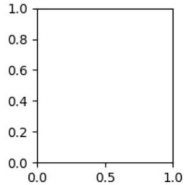


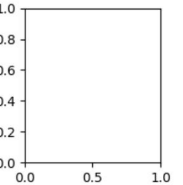





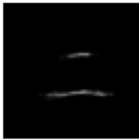


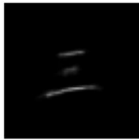


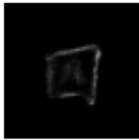




















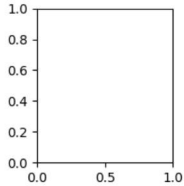


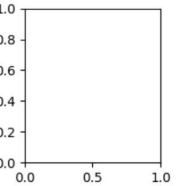








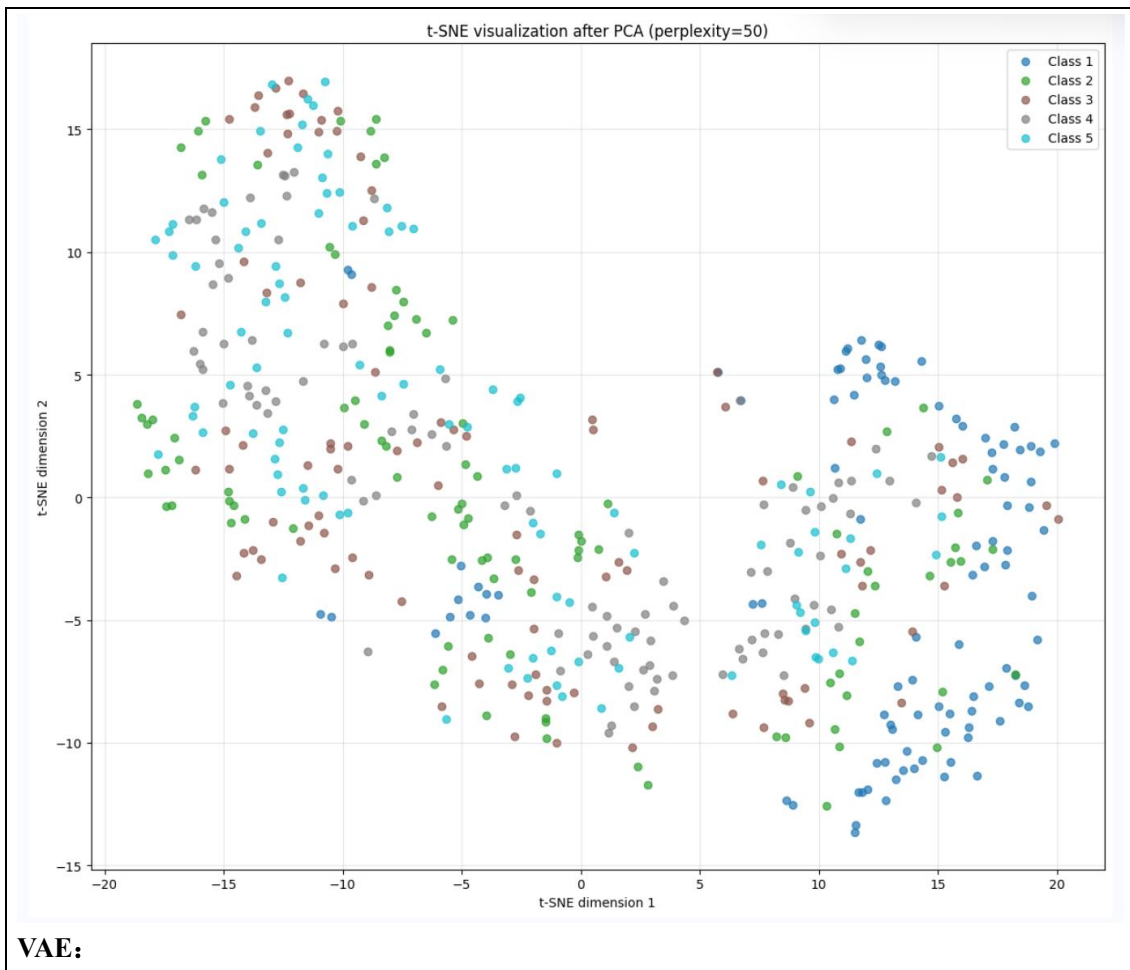


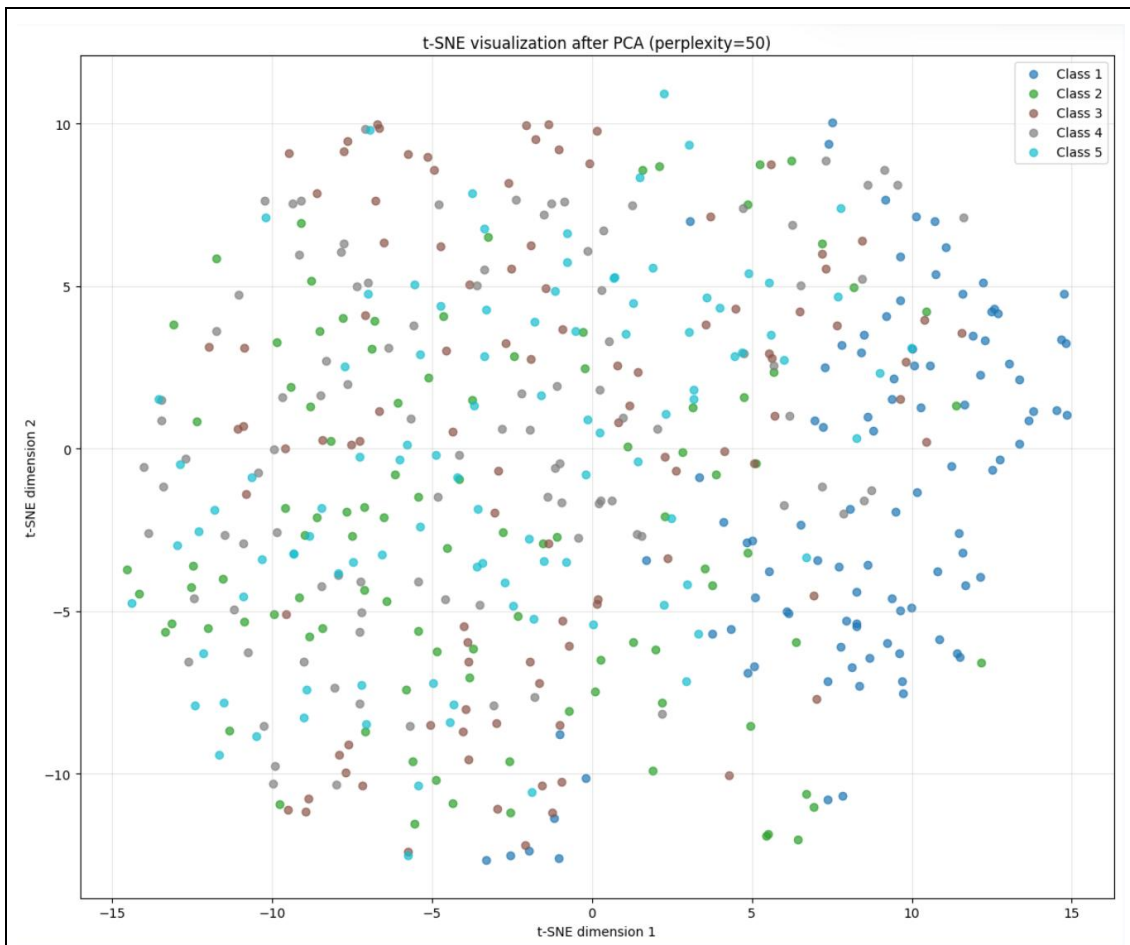


运行时长: 788毫秒 结束时间: 2025-05-18 00:42:22

可视化结果:

AE:





比较二者区别：

可以发现，在 t-SNE 降维后，AE 会形成紧密但非规则的簇，对应不同数字类别。而 VAE 由于引入了 KL 散度，VAE 被强制让潜在分布尽量贴近标准正态分布，其降维图上更加连续均匀的点云，类别簇之间无明显的空洞

理论解释：

重构 vs 正则的权衡：AE 只关注重构，潜空间可以任意分布以最小化重构误差；VAE 在 ELBO（证据下界）中增加了 KL 正则项，使潜空间分布贴近先验，从而牺牲一部分重构质量以换取更好的生成连贯性和潜空间结构化。

生成能力：VAE 的连续潜空间能够从任何潜在向量生成样本，适合“插值”与多样化生成；AE 潜空间缺乏结构化先验，若从未见过的潜在区域采样，往往生成不合理或失真图像。

平滑性：VAE 潜在空间的平滑性更高，有助于下游任务（如分类、聚类）获得一致性表示；AE 表示在类别边界附近可能出现断层，不利于这些任务。

五、总结

（自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

对于 VAE 的预期并不是最佳，但也是很有收获的，通过修改损失函数，得到了更加贴近的图像。困难更多来自对于模型的陌生以及参数调整所需时间的影响，成本很高。对于 VAE 还是总结了一些提升性能的方法，比如调整学习率、簇大小、黑白区域损失比例，模型潜在空间维数等。当然，这些也是学习机器学习的必经之路！

