**程序报告**

学号： 22375080 姓名：杨佳宇轩

1. **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

1. **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

1. **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,var) || N(0,1)) = -0.5 \* sum(1 + logvar - mu^2 - 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

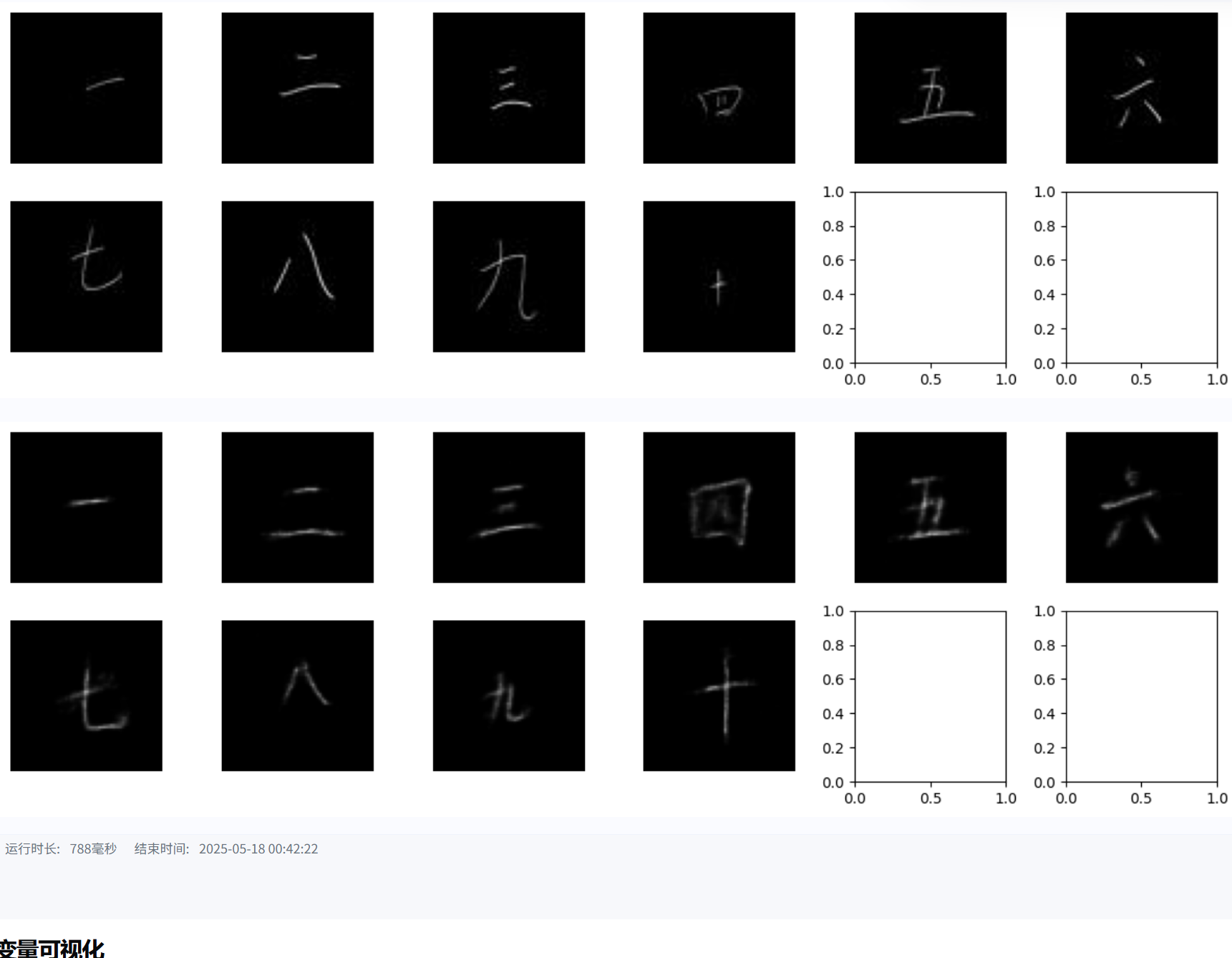
    return paddle.mean((eps - noise\_pred)\*\*2)

1. **实验结果**

（实验结果，必填）

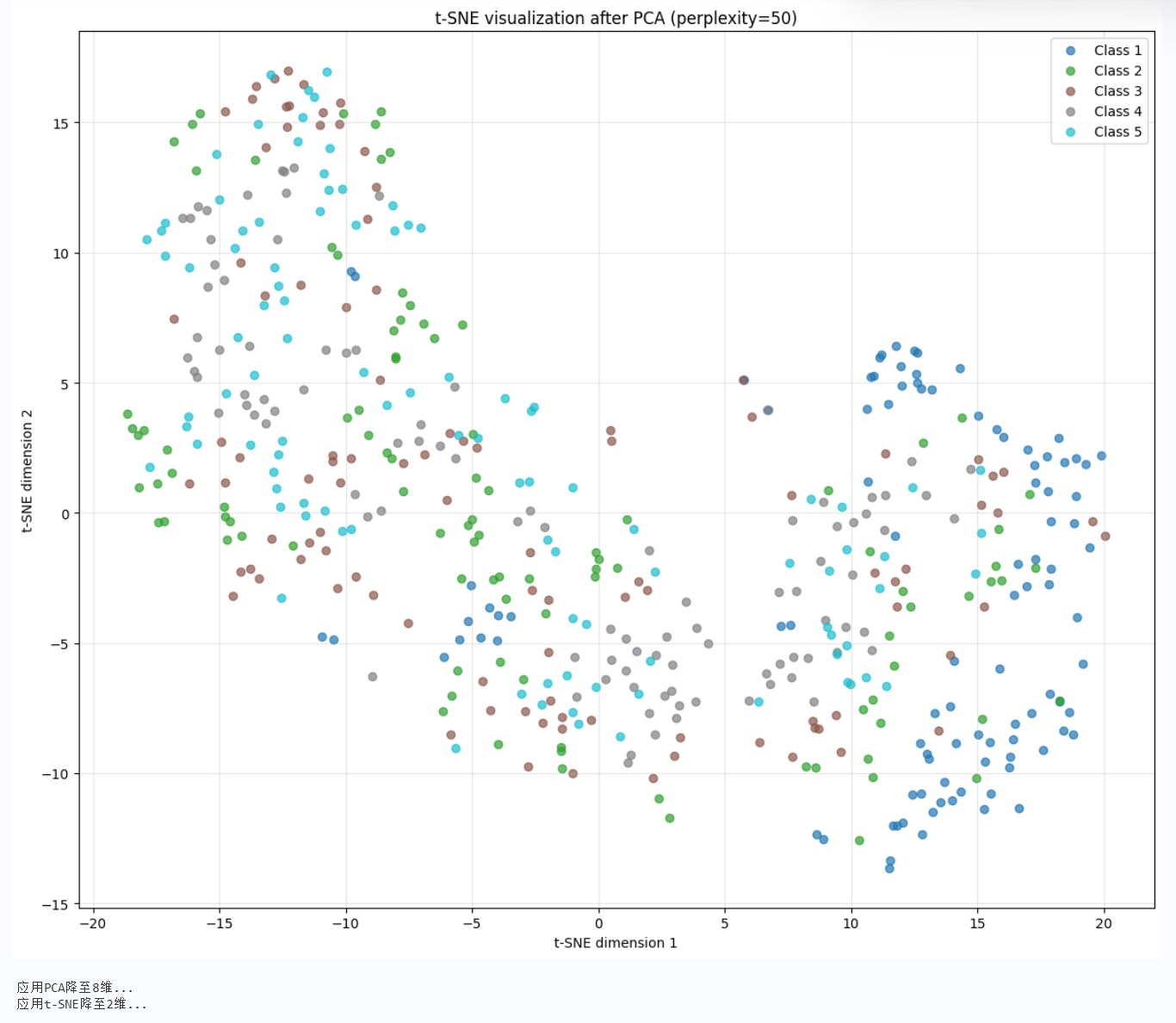
====================================================================

**VAE生成结果：**

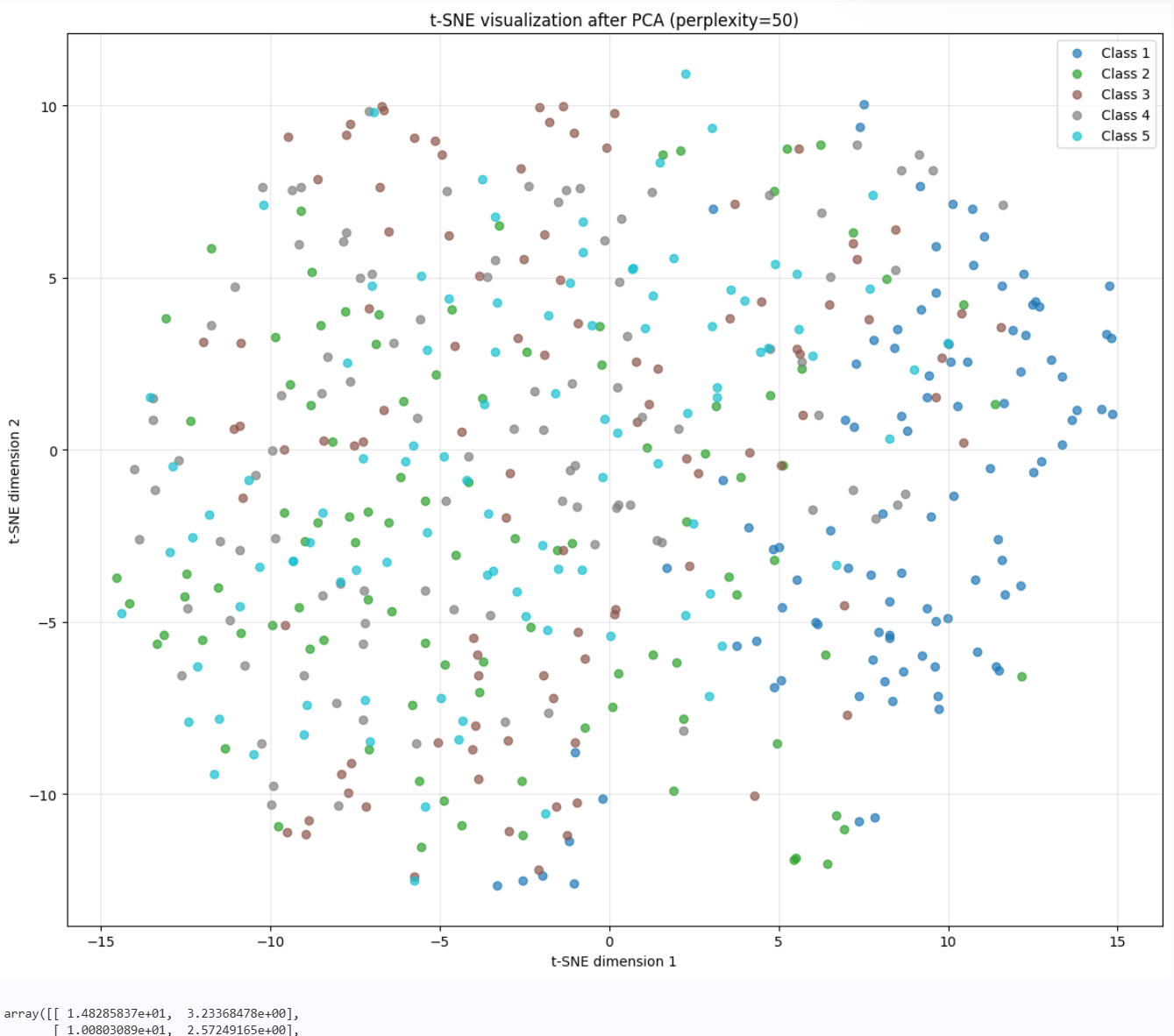


**可视化结果：**

**AE：**



**VAE：**



**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。

1. **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！