Supplementary Material for DGD: Density Gradient-guided Diffusion for Long-Tailed Clustering

A. Detailed Derivation of Density-Guided Diffusion

This section provides a compact derivation of the proposed density-gradient guidance mechanism within the reverse diffusion process. The formulation is fully aligned with the main paper and clarifies how the standard DDPM reverse process evolves into the density-guided mean update.

A.1. Standard Reverse Process

In a standard DDPM, the reverse transition at timestep t follows:

$$p_\theta(z_{t-1} \,|\, z_t) = \mathcal{N}\big(z_{t-1};\, \mu_\theta(z_t, t),\, \Sigma_\theta(z_t, t)\big), \tag{A.1}$$

where the mean is

$$\mu_\theta(z_t, t) = \tfrac{1}{\sqrt{\alpha_t}}\Big(z_t - \tfrac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(z_t, t)\Big), \tag{A.2}$$

and $\Sigma_\theta(z_t, t)$ is the time-dependent variance. This reverse process approximates the data distribution in an unbiased manner, but under long-tailed data, it tends to reconstruct high-density (head) regions.

A.2. Latent Density Estimation and Potential Function

Real samples are encoded by the feature extractor E_ϕ into latent features $Z_{\text{real}} = \{z_i\}_{i=1}^N$. The latent density is estimated using Gaussian KDE:

$$\hat{p}(z) = \tfrac{1}{N}\sum_{i=1}^{N} K_\sigma(z - z_i), \tag{A.3}$$

where $K_\sigma(\cdot)$ is an isotropic Gaussian kernel. We define a potential function:

$$U(z) = -\ln\big(\hat{p}(z) + \rho\big), \tag{A.4}$$

where $\rho > 0$ ensures numerical stability. $U(z)$ attains larger values in low-density (tail) regions. Its gradient field

$$\nabla_z U(z) = -\tfrac{\nabla_z \hat{p}(z)}{\hat{p}(z)+\rho}, \tag{A.5}$$

points from dense head regions toward sparse tails and serves as the guidance direction.

A.3. Density-Guided Reverse Posterior

To bias sampling toward tail regions, we reweight the reverse posterior:

$$p_{\text{guided}}(z_{t-1} \,|\, z_t) \propto p_\theta(z_{t-1} \,|\, z_t)\exp\big(\eta_t U(z_{t-1})\big), \tag{A.6}$$

where η_t is a time-dependent coefficient—strong early for exploration, weaker later for detail recovery.

A.4. Deriving the Guided Gaussian Form

Since $p_\theta(z_{t-1} \,|\, z_t)$ is sharply peaked near $\mu_\theta(z_t, t)$, we linearize $U(z_{t-1})$ around μ_θ:

$$U(z_{t-1}) \approx U(\mu_\theta) + (z_{t-1} - \mu_\theta)^\top \nabla_z U(\mu_\theta). \tag{A.7}$$

Substituting yields:

$$p_{\text{guided}}(z_{t-1} \,|\, z_t) \propto \exp\Big(-\tfrac{1}{2}(z_{t-1} - \mu_\theta)^\top \Sigma_\theta^{-1}(z_{t-1} - \mu_\theta) + \eta_t (z_{t-1} - \mu_\theta)^\top \nabla_z U(\mu_\theta)\Big). \tag{A.8}$$

Completing the square gives:

$$-\tfrac{1}{2}\big(z_{t-1} - \mu_\theta - \eta_t \Sigma_\theta \nabla_z U(\mu_\theta)\big)^\top \Sigma_\theta^{-1}\big(z_{t-1} - \mu_\theta - \eta_t \Sigma_\theta \nabla_z U(\mu_\theta)\big) + \text{const.} \tag{A.9}$$

Thus, the guided posterior remains Gaussian:

$$p_{\text{guided}}(z_{t-1} \,|\, z_t) = \mathcal{N}\big(z_{t-1}; \mu_\theta(z_t, t) + \eta_t \Sigma_\theta(z_t, t) \nabla_z U(\mu_\theta(z_t, t)), \Sigma_\theta(z_t, t)\big). \tag{A.10}$$

The guidance acts as a mean shift $\eta_t \Sigma_\theta \nabla_z U$, leaving the covariance unchanged—an exponential reweighting of the reverse Gaussian.

A.5. Practical Guided Mean

For stability, we evaluate the gradient at the current latent z_t:

$$\mu_{\text{guided}}(z_t, t) = \mu_\theta(z_t, t) + \eta_t \Sigma_\theta(z_t, t) \nabla_z U(z_t). \tag{A.11}$$

Here, μ_θ reconstructs data, while the second term directs the process toward sparse tail regions—key to enhancing long-tailed representation.

B. Efficiency and Complexity of Density Estimation

To scale DGD to large datasets, we adopt an efficient truncated kNN-KDE approximation.

Algorithm 1: Training Procedure of DGD

Input: Training data X, encoder E_ϕ, diffusion model ϵ_θ, threshold τ, guidance strength η_0
Output: Optimized parameters ϕ, θ
1. Pre-train encoder: Train E_ϕ for 50 epochs to obtain a stable latent space.
2. Outer loop: for $o = 1$ **to** T_{outer} **do**
  Encode samples $x_i \in X$ as $z_i = E_\phi(x_i)$;
  Estimate KDE $\hat{p}(z)$ and gradient $\nabla_z U(z)$; cache them;
  Inner loop: for $i = 1$ **to** T_{inner} **do**
    Sample latent z_t;
    Compute $\mu_{\text{guided}} = \mu_\theta(z_t, t) + \eta_t \Sigma_\theta(z_t, t) \nabla_z U(z_t)$;
    Generate z_0 via reverse diffusion;
    Cluster $Z_{\text{real}} \cup Z_{\text{gen}}$ to obtain pseudo labels;
    Select high-confidence samples ($P_{\max} > \tau$);
    Update ϵ_θ, E_ϕ using $\mathcal{L}_{\text{DM}} + \lambda \mathcal{L}_{\text{InfoNCE}}$;

B.1. Naive KDE Complexity

A Gaussian KDE for N latent features $\{z_i\}$ in d dimensions requires:

$$\hat{p}(z) = \frac{1}{N} \sum_{i=1}^{N} K_\sigma(z - z_i), \tag{A.12}$$

yielding $O(Nd)$ per query and $O(N^2 d)$ overall, which is prohibitive for periodic refreshes on long-tailed datasets.

B.2. Efficient kNN-KDE Approximation

We approximate KDE using k-nearest neighbors:

$$\hat{p}_{\text{kNN}}(z) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(z)} K_\sigma(z - z_i), \tag{A.13}$$

where $\mathcal{N}_k(z)$ is the k-NN set ($k = 64$ by default). The density gradient shares the same neighbor set:

$$\nabla_z \hat{p}_{\text{kNN}}(z) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(z)} \nabla_z K_\sigma(z - z_i). \tag{A.14}$$

Since local KDE is evaluated only over the k-nearest neighbors of each sample, the estimation of density and its gradient relies only on local neighborhood information, avoiding the full computation over all N samples required by naive KDE.

B.3. Practical Implementation

We refresh the density field periodically (every 20 epochs):
1. Use a GPU-accelerated kNN search to compute $\mathcal{N}_k(z)$ for all latent points.
2. Compute and cache $\hat{p}_{\text{kNN}}(z)$ and $\nabla_z \hat{p}_{\text{kNN}}(z)$ for all z_i.

During training, $\nabla_z U(z_t)$ for each batch sample is retrieved from cache, amortizing the computational overhead across many iterations. Table 5 further reports the practical training time and GPU memory usage on CIFAR-10-LT (IR=10), showing that DGD remains efficient in practice while achieving the best clustering accuracy.

Table 5. Efficiency comparison on CIFAR-10-LT (IR=10).

Method	**Time (h)**↓	**Memory (GB)**↓	**ACC (%)**↑
SCAN	1.2	4.2	29.4
CC	1.5	4.8	39.6
CLUDI	3.5	6.0	46.1
Ours	**2.8**	**5.5**	**54.5**

C. Additional Experimental Settings

All experiments follow the same backbone, input resolution, diffusion steps, and optimization settings as Sec. 4.1. Training proceeds in two stages: pre-training for feature stabilization and joint optimization for synergy.

Pre-training. The encoder E_ϕ is trained for 50 epochs to form a stable latent space. It is then frozen while the diffusion model ϵ_θ is trained for 100 epochs using only the denoising objective.

Joint optimization. Both networks are jointly trained for 200 epochs under the combined loss. The density field is refreshed every 20 epochs (outer loop), and model parameters are updated at each iteration (inner loop). Unless otherwise stated, all results are averaged over five runs.