Decision Boundary-aware Generation for Long-tailed Learning

Supplementary Material

7. Rationale

7.1. Experimental Details

Long-tailed CIFAR-10 and CIFAR100-LT. We construct imbalanced (long-tailed) versions of CIFAR-10 and CIFAR-100 by subsampling the original training set following an exponential decay rule over class indices. Given C classes and the original per-class sample count $n_{\max} = N/C$ in the balanced dataset ($N = 50{,}000$ for both CIFAR-10 and CIFAR-100), we specify an imbalance factor $\gamma \in (0, 1]$ and assign the number of training samples for class $c \in \{0, \dots, C-1\}$ as

$$n_c = n_{\max} \cdot \gamma^{\frac{c}{C-1}}.$$

Thus the smallest class approximately contains $n_{\min} \approx n_{\max} \cdot \gamma$ samples, and the effective imbalance ratio is $n_{\max}/n_{\min} \approx 1/\gamma$ (e.g., $\gamma = 0.01$ corresponds to an imbalance ratio of 100). For each class, we randomly shuffle the indices of all training images and retain the first n_c samples, resulting in IMBALANCECIFAR10 and IMBALANCECIFAR100 datasets with exponentially decaying class frequencies.

Preprocessing for classifier training. When training the classifiers, we apply standard data augmentation and normalization to all training images. For both CIFAR-10 and CIFAR100-LT, each training image is first randomly cropped to 32×32 with 4-pixel padding, followed by random horizontal flipping. The augmented image is then converted to a tensor and normalized using the dataset-specific mean and standard deviation

$$\mu = (0.4914, 0.4822, 0.4465), \quad \sigma = (0.2023, 0.1994, 0.2010).$$

For evaluation, test images are resized to 32×32, converted to tensors, and normalized with the same (μ, σ), without any random augmentation.

Diffusion model training. We follow the original CBDM-based implementation to train a class-conditional Gaussian diffusion model on the long-tailed CIFAR datasets. The forward diffusion uses $T = 1000$ timesteps with a linear variance schedule $\{\beta_t\}_{t=1}^{T}$ ranging from $\beta_1 = 1 \times 10^{-4}$ to $\beta_T = 2 \times 10^{-2}$; the corresponding $\alpha_t = 1 - \beta_t$ and cumulative products $\bar{\alpha}_t$ are defined as usual. During diffusion training, images are normalized to the range $[-1, 1]$ using a dataset-agnostic mean and standard deviation

$$\mu = (0.5, 0.5, 0.5), \quad \sigma = (0.5, 0.5, 0.5),$$

which is only used for the diffusion model and is independent of the preprocessing used for the classifier. The model is trained with a base learning rate of 2×10^{-4}, linearly warmed up during the first $5{,}000$ optimization steps, and optimized for $500{,}000$ training steps in total with a batch size of 128. Gradient norms are clipped at 1.0, and an exponential moving average of the model parameters is maintained with decay rate 0.9999. After training, the diffusion parameters are frozen and used solely as a generator for boundary-guided samples.

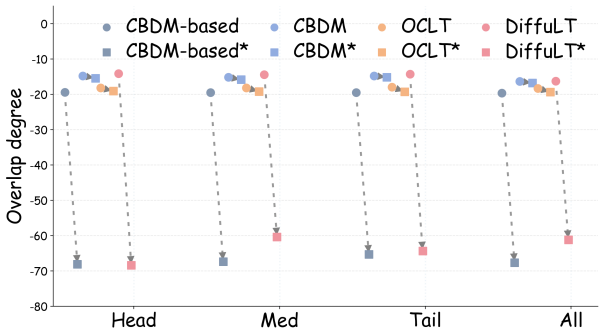
Diffusion-based boundary sample generation (DBG). During classifier training, we use the pre-trained diffusion model as an editing-based generator to synthesize boundary-guided samples online. For each mini-batch, we first run the current classifier to obtain logits for all samples. For a given input image with ground-truth label y, we mask out the logit of class y and select the top-K alternative classes from the remaining logits, where

$$K = \min\left(\frac{C}{3}, C - 1\right)$$

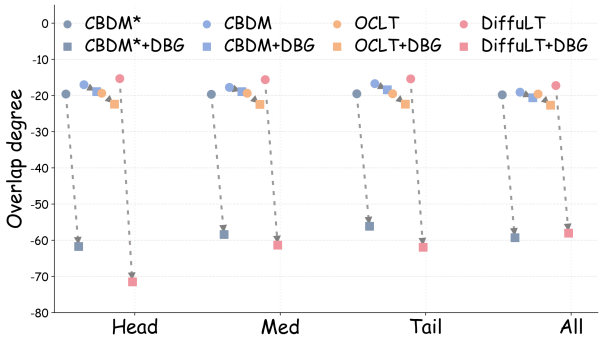
and C is the number of classes. The image is duplicated K times and each copy is paired with one of these target labels, while the original label y is kept as the source label, yielding a set of image–(source, target) triplets.

Given a batch of such triplets, DBG sampling proceeds in two phases. In the first phase (*forward remove class*), we start from the clean input x_0 at an intermediate diffusion time and run a short forward diffusion trajectory using a deterministic DDIM-style update. Concretely, we begin at $t_{\text{start}} \approx 0.5\,(T-1)$ and stop at $t_{\max} \approx 0.6\,(T-1)$ with a fixed step size of $\texttt{ddim\_skip\_step} = 10$ (i.e., every 10 timesteps). At each step, the diffusion model is queried with the source label, and we use a negative guidance weight fixed at $w_{\text{neg}} = 1$ with $\eta = 0$ (purely deterministic DDIM). This pushes the sample to a noisier state while suppressing features associated with the source class, moving it towards a class-agnostic region near the decision boundary.

In the second phase (*denoise towards target*), we take the noisy intermediate sample at time $t_{\max}$ and run a reverse DDIM trajectory back to $t = 0$ with the same step size of 10. In this stage, we apply classifier-free guidance with a fixed guidance scale $w = 1.0$: at each step, the model is evaluated both conditionally (using the target label) and unconditionally, and the two predictions are combined using this guidance weight to steer the denoising towards the chosen target class while preserving overall image structure.



(a) CIFAR100-LT with imbalanced ratio set to 10.



(b) CIFAR100-LT with imbalanced ratio set to 200.

Figure 9. Inter-class overlap changes across baselines with DBG on ResNet-32 with imbalanced rate set to 10 and 200 for CIFAR100-LT.

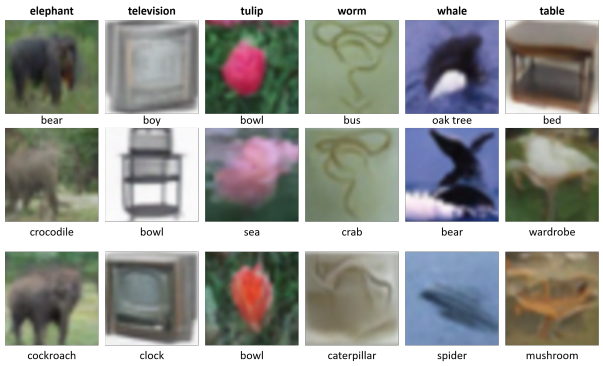


Figure 10. Some of the samples generated by DBG.

Throughout the reverse process, the same diffusion schedule ($T = 1000$, $\beta_1 = 1 \times 10^{-4}$, $\beta_T = 2 \times 10^{-2}$, $\eta = 0$) is used. After reaching $t = 0$, the final sample is mapped back from $[-1, 1]$ to $[0, 1]$ and used as a DBG sample; these generated boundary-guided images are then mixed with the original training data and the baseline-generated images to train the classifier.

7.2. Additional Results & Visualization

As shown in Fig. 9, when training ResNet32 on the CIFAR-100LT dataset with imbalance factors of 10 and 200, our method significantly reduces the inter-class overlap in the decision space, demonstrating that our method is robust to the imbalance factor.

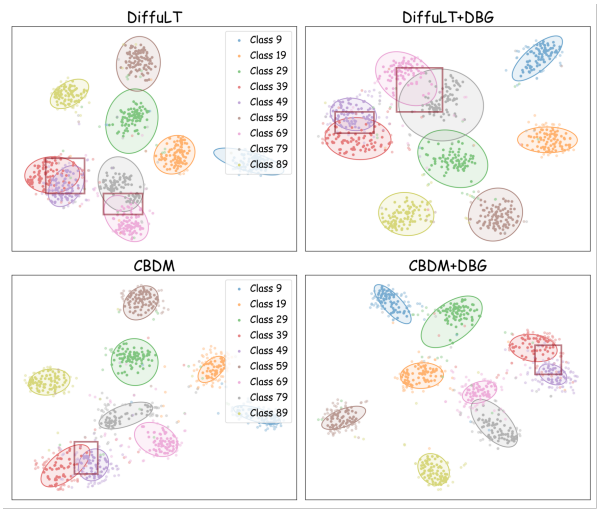


Figure 11. Changes in Feature space under different baselines.

Fig. 10 shows some of the DBG-generated data. It can be observed that different initial images indeed provide additional distinctive features for augmenting the overall dataset features. Moreover, it is clear that images generated from more similar classes visually resemble each other more closely, validating our motivation: generating images from similar classes to improve the effectiveness of generated images and better supplement boundary information.

Fig. 11 shows the changes in the decision space after injecting DBG-generated data for both CBDM and DiffuLT methods. The inter-class distances increase, overlap decreases, and the decision space for tail classes expands, which aids in learning for the tail classes.

To validate the impact of the biased classifier for DBG, experiments with poorly-trained Res-32 (50 epochs) are conducted on CIFAR-100 and confirms the robustness against circular dependency of DBG as shown blow.

Method	Head	Tail	All	Method	Head	Tail	All
CBDM*	66.30	25.93	48.81	**OCLT**	68.70	25.73	49.28
+DBG 50E	**67.03**	**28.07**	**50.09**	**+DBG 50E**	**68.73**	**27.30**	**50.93**

To verify the generalization of DBG on large-scale datasets, a pre-trained stable diffusion model and Res-32 are used to verify the role of DBG on tiny-Imagenet. As shown in the right table, and the results show that DBG can be applied to large-scale datasets, the boundary data set generated by DBG can also improve the recognition effect on large-scale dataset.

Method	ST-diffusion	+DBG
Head	44.7	44.4
Tail	8.5	9.0
All	26.4	**26.8**↑