APPENDIX

A. Dataset Splits and Training Details

The number of images and text samples in the CUB-200 and Oxford-102 datasets, along with their dataset splits, are summarized in Table 4. The proposed RobustTGIC is implemented in PyTorch and optimized using the Adam optimizer with a batch size of 8. Our model is trained for 150 epochs, followed by an additional 10 epochs of fine-tuning when handling missing modalities. The learning rate is set to 1×10^{-4}. The hyperparameters of the loss function, k_d, k_p, k_g, and a, are empirically set to 0.075×2^{-5}, 5, 0.15, and 4, respectively. The rate control parameter λ is set to $(2^3, 2^2, 2^1, 2^0)$ depending on the target bit rate.

Table 4. Training/Testing set of CUB-200 and Oxford-102 Datasets

Dataset	Training	Testing	Total	Text Caption
CUB-200	8855	2933	11788	10 / image
Oxford-102	7034	1155	8189	10 / image

B. Visualization of Fine-Tuning Effects

Figure 8 presents the visualized reconstructed images before and after text modality missing, as well as after missing-modality fine-tuning. When the text is missing, the reconstructed image quality significantly degrades, with noticeable color distortion, especially in the background regions. However, after fine-tuning, the image quality substantially improves, showing only a slight decrease compared to the non-missing case, with no perceptible quality difference.

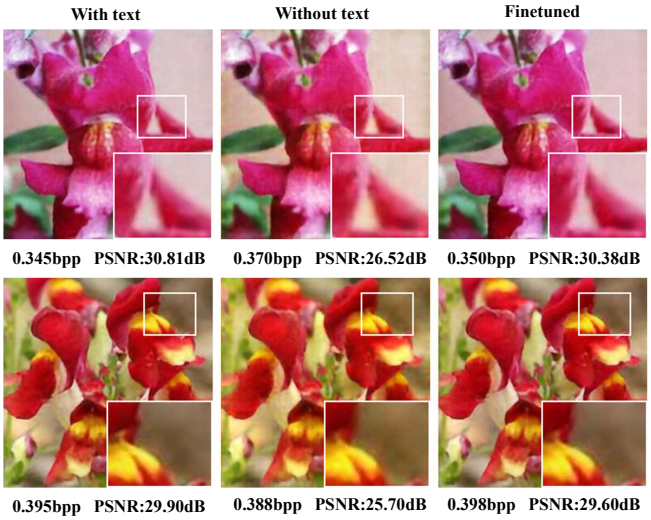

Fig. 8. Visual Comparison Before and After Modality Missing, as Well as After Fine-Tuning.

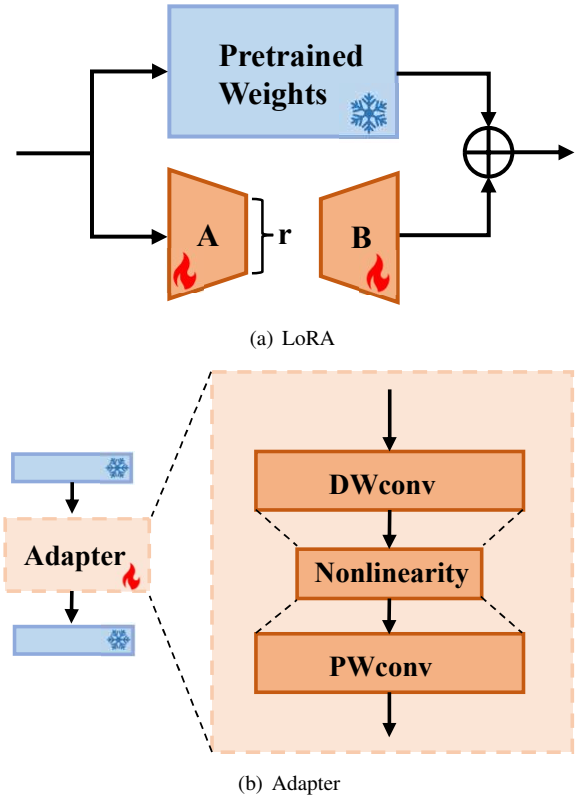

Fig. 9. Structure illustration of LoRA and Adapter.

C. Efficient Fine-Tuning Methods

The LoRA method, illustrated in Figure 9(a), freezes the pretrained layers during fine-tuning and inserts two trainable low-rank matrices, A and B, into the layer. The final output is obtained by adding the output of the low-rank matrices to the frozen pretrained model's output. In contrast, the Adapter method, as shown in Figure 9(b), inserts an adaptation module after the pretrained model to enable efficient fine-tuning.