Appendix

In this appendix, Section A gives a description of the semi-supervised DDGAN for unpaired data. Section B and Section C provide more details about optimization and network architectures. Section D provides extra experimental results and analysis of the loss functions we used.

A. Semi-supervised DDGAN

Although the unsupervised DDGAN is more general, the semi-supervised DDGAN can take advantage of the supervised information of unpaired data for degradation invariance learning. For this case, the training set requires a domain-wise partition, *e.g.*, a day/night division, and the degradation score predicted by the degradation discriminator should be 0-1 discrete distribution, where 0 indicates no degradation, while 1 indicates degradation.

Since the unsupervised degradation ranking loss is inappropriate for such a discrete distribution, we rewrite the degradation adversarial loss as:

$$\begin{aligned} L^s_{deg} =& \mathbb{E}[log(D_d(x_k)) + log(1 - D_d(G(f^i_c, f^j_d)))] \\ &+ \mathbb{E}[log(D_d(G(f^j_c, f^i_d))) + log(1 - D_d(x_i))] \end{aligned}$$

and

$$\begin{aligned} L^c_{deg} =& \mathbb{E}[log(D_d(x_k)) + log(1 - D_d(G(f^i_c, f^k_d)))] \\ &+ \mathbb{E}[log(D_d(G(f^k_c, f^i_d))) + log(1 - D_d(x_i))] \end{aligned}$$

for self-degradation generation and cross-degradation generation, respectively. Note that x_i is sampled from normal image domain, while x_k is sampled from degraded image domain, both of which are real-world samples.

The rest of the loss functions are consistent with that of the unsupervised DDGAN.

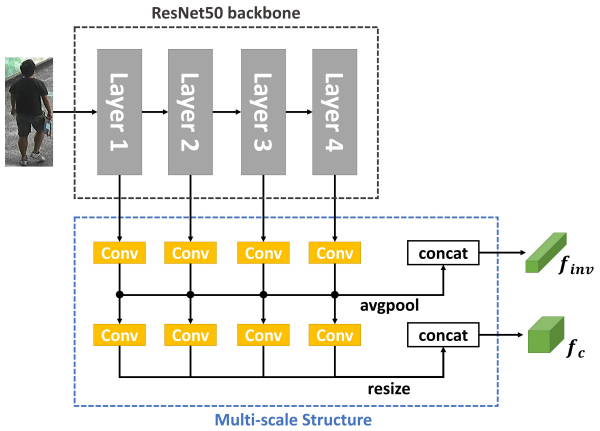

Figure 8. The multi-scale structure of the content encoder E_c.

Table 4. Architecture of the degradation encoder E_d.

Layer	Parameters	Output Size
Input	-	3 × 256 × 128
Conv1	[3×3, 64]	64 × 128 × 64
Conv2	[3×3, 64]	64 × 128 × 64
Conv3	[3×3, 128]	128 × 64 × 32
Conv4	[3×3, 256]	256 × 32 × 16
Conv5	[3×3, 256]	256 × 16 × 8
Conv6	[3×3, 256]	256 × 8 × 4
AvgPool	-	256 × 1 × 1
Conv7	[1×1, 128]	128 × 1 × 1

Table 5. Ablation Study of loss functions on the MLR-CUHK03 dataset.

Methods	Rank-1	Rank-5	Rank-10
Ours w/o L_{pre}	85.4	96.5	98.1
Ours w/o L_{id}	84.1	96.3	97.3
Ours w/o L_{invc}	84.5	96.3	98.0
Ours w/o L_{recon}	86.0	96.3	97.8
Ours w/o L_{real}	85.2	96.0	98.1
Ours w/o L_{deg}	85.1	96.0	98.0
Ours	85.7	97.1	98.6

B. Optimization Details

During the stage of degradation invariance learning, an Adam optimizer with learning rate of 0.0001, weight decay of 0.0005 and $(\beta_1, \beta_2) = (0, 0.999)$ is adopted for the DDGAN except the auxiliary identity classifier, which is optimized by a SGD optimizer with learning rate of 0.01, weight decay of 0.0001 and Nesterov momentum of 0.9. The same SGD is also applied to optimize the whole DFEN during the stage of identity representation learning. We also utilize a multi-step learning rate scheduler with a learning rate decay of 0.1 and step size of 30000 for smooth convergence.

For the degradation invariance learning stage, the DDGAN is optimized by 50,000 iterations with a batch size of 8. The parameters $\lambda_{invc}, \lambda_{recon}, \lambda_{real}, \lambda_{deg}, \lambda_{id}, \lambda_{pre}$ are set to 10, 2, 1, 1, 0.5, 0.5, respectively. Adam optimizer with a learning rate of 0.0001 is adopted.

For the identity representation learning stage, the DFEN is optimized by 50,000 iterations (10,000 iterations for two small datasets MLR-VIPER and CAVIAR) and the batch size is 32. The parameters $\lambda_{inv}, \lambda_{sen}, \lambda_{both}$ are set to 0.5, 0.5 and 1, respectively. The SGD algorithm with the weight decay of 0.0001 and the Nesterov momentum of 0.9 is used for the whole DFEN.

In the inference stage, both f_{inv} and f_{sen} are concatenated into a 1024-dim vector as the final identity representation for testing.

C. Architecture Details

Proposed approach is able to extract disentangled representations with a content encoder E_c and a degradation encoder E_d. As shown in Figure 8, a multi-scale ResNet50 structure is imployed for E_c. We note that both content features f_c and identity features f_{inv} are produced by the ResNet50 backbone, which promotes the content encoder E_c to take into account both degradation invariance and identity discriminability. In addition, the architecture of degradation encoder E_d is given in Table 4. Although the self-degradation encoder E'_d does not share weights with E_d, they have a completely identical structure.

For the degradation-guided attention module, we adopt a four-layer perceptron structure with the activation function of ReLU. The final attentive weights of f_{sen} are produced by a softmax layer for normalization.

D. Analysis of Loss Functions

In order to analyze the influence of each loss term, we study the contributions of loss functions in Table 5 and provide additional generation results, which are produced by our DDGAN with different loss weights, *i.e.*, λ_{pre}, λ_{id}, λ_{invc}, λ_{recon}, λ_{real} and λ_{deg}.

As shown in Figure 9, each loss item has a distinct contribution to the generation results. In particular, L_{pre} and L_{id} suppress the occurrence of artifacts by introducing identity constraints. L_{real} also suppresses the artifacts that may appear, improving the reality of generated images. As a self-reconstruction regularization constraint, L_{recon} has the most significant impact on the quality of image generation. L_{invc} provides degradation invariance constraints at the feature level, while L_{deg} is performed at the image level, which plays an important role on degradation disentanglement learning. Note that the degradation components almost cannot be captured without L_{deg}, which makes the generated images very similar to the corresponding content images (*e.g.*, 1th and 8th columns in Figure 9).

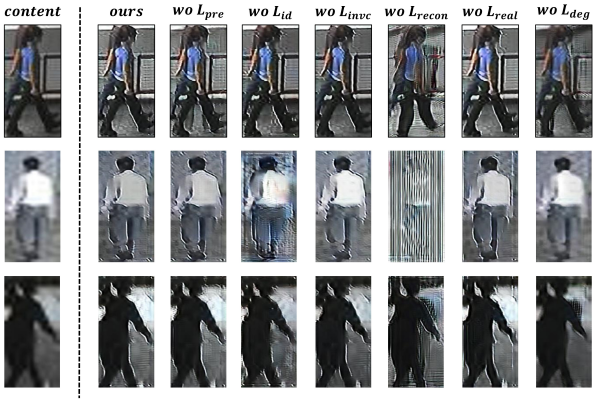

Figure 9. The contribution of each loss term for cross-degradation generation.

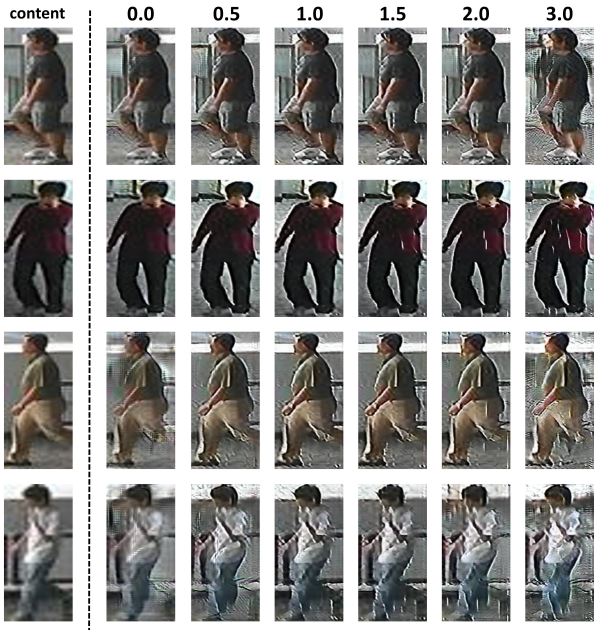

Figure 10. The influence of the hyperparameter λ_{deg} (vary from 0.0 to 3.0).

For further analysis of the degradation loss L_{deg}, we set λ_{deg} to different values and re-train the model. As illustrated in Figure 10, We observe that a larger λ_{deg} leads to over-disentanglement, while a smaller λ_{deg} might result in under-disentanglement.