6. Supplementary Materials

6.1. Implementation Details

Network Architectures. The network architectures of CR-GAN are listed in Table 8, 9, 10. We describe each layer or residual block as "conv-(K-, N-, S-, P-, PS/PV, IN/BN, LReLU)", "res(K-, N-, S-, P-, PS/PV, IN/BN, LReLU)". K: kernel size, N: number of filters, S: stride size, P: padding size, PS: padding='same', PV: padding='valid', IN: instance normalisation, BN: batch normalisation, LReLU: LeakyReLU. U: upsampling with kernel size 2×2. Input image size "$H\times W$" is 224×112.

Part Name	**Input → Output Shape**	**Layer Description**
Dual-Path Encoding	$(H, W, 3) \rightarrow (\frac{H}{2}, \frac{H}{2}, 64)$	context pathway: conv-(K-4×4, N-64, S-2, P-0, PS, LReLU)
	$(H, W, 3) \rightarrow (\frac{H}{2}, \frac{H}{2}, 64)$	identity pathway: conv-(K-4×4, N-64, S-2, P-0, PS, LReLU)
U-Net (encoder)	$(\frac{H}{2}, \frac{H}{2}, 128) \rightarrow (\frac{H}{4}, \frac{W}{4}, 128)$	res-(K-4×4, N-128, P-0, PS, LReLU)
	$(\frac{H}{4}, \frac{H}{4}, 128) \rightarrow (\frac{H}{8}, \frac{W}{8}, 256)$	res-(K-4×4, N-256, P-0, PS, LReLU)
	$(\frac{H}{8}, \frac{H}{8}, 256) \rightarrow (\frac{H}{16}, \frac{W}{16}, 512)$	res-(K-4×4, N-512, P-0, PS, LReLU)
	$(\frac{H}{16}, \frac{H}{16}, 512) \rightarrow (\frac{H}{32}, \frac{W}{32}, 512)$	res-(K-4×4, N-512, P-0, PS, LReLU)
	$(\frac{H}{32}, \frac{H}{32}, 512) \rightarrow (\frac{H}{64}, \frac{W}{64}, 512)$	res-(K-4×4, N-512, P-0, PS, LReLU)
	$(\frac{H}{64}, \frac{H}{64}, 512) \rightarrow (\frac{H}{128}, \frac{W}{128}, 512)$	res-(K-4×4, N-512, P-0, PS, LReLU)
	$(\frac{H}{128}, \frac{W}{128}, 512) \rightarrow (\frac{H}{256}, \frac{W}{256}, 512)$	conv-(K-4×4, N-512, S-2, P-0, PS)
U-Net (decoder)	$(\frac{H}{256}, \frac{H}{256}, 512) \rightarrow (\frac{H}{128}, \frac{W}{128}, 512)$	U + res-(K-4×4, N-512, P-0, PS, IN, ReLU)
	$(\frac{H}{128}, \frac{H}{128}, 1024) \rightarrow (\frac{H}{64}, \frac{W}{64}, 512)$	U + res-(K-4×4, N-512, P-0, PS, IN, ReLU)
	$(\frac{H}{64}, \frac{H}{64}, 1024) \rightarrow (\frac{H}{32}, \frac{W}{32}, 512)$	U + res-(K-4×4, N-512, P-0, PS, IN, ReLU)
	$(\frac{H}{32}, \frac{H}{32}, 1024) \rightarrow (\frac{H}{16}, \frac{W}{16}, 512)$	U + res-(K-4×4, N-512, P-0, PS, IN, ReLU)
	$(\frac{H}{16}, \frac{H}{16}, 1024) \rightarrow (\frac{H}{8}, \frac{W}{8}, 256)$	U + res-(K-4×4, N-256, P-0, PS, IN, ReLU)
	$(\frac{H}{8}, \frac{H}{8}, 512) \rightarrow (\frac{H}{4}, \frac{W}{4}, 128)$	U + res-(K-4×4, N-128, P-0, PS, IN, ReLU)
	$(\frac{H}{4}, \frac{W}{4}, 256) \rightarrow (\frac{H}{2}, \frac{W}{2}, 128)$	U + conv-(K-4×4, N-128, S-1, P-0, PS, IN, ReLU)
Decoding	$(\frac{H}{2}, \frac{W}{2}, 128) \rightarrow (H, W, 3)$	residual map: U + conv-(K-4×4, N-3, S-1, P-0, PS, tanh)
	$(\frac{H}{2}, \frac{W}{2}, 128) \rightarrow (H, W, 1)$	context mask: U + conv-(K-4×4, N-1, S-1, P-0, PS, sigmoid)

Table 8: Network architecture of dual conditional image generator. Note that the U-Net contains skip connections that are helpful to preserve the underlying image structure across network layers. Downsampling and upsampling residual blocks are depicted in Figure 8.

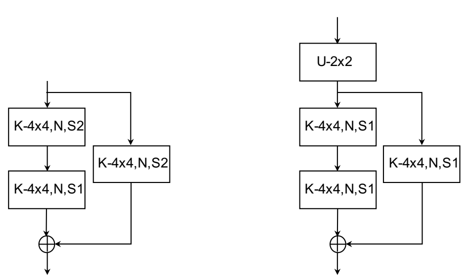

Figure 8: Left: Downsampling residual block. Right: Upsampling residual block. Note: conv layer is introduced in the shortcut connection as the number of feature maps in input and output are not necessarily the same in the U-Net.

Part Name	Input $\rightarrow$ Output Shape	Layer Description
Input Layer	$(H, W, 3) \rightarrow (H, W, 3)$	additive Gaussian noise $\mathcal{N}(0, 0.1)$
Hidden Layers	$(H, W, 3) \rightarrow (\frac{H}{2}, \frac{W}{2}, 128)$	conv-(K-4×4, N-128, S-2, P-2, PV, LReLU)
	$(\frac{H}{2}, \frac{W}{2}, 128) \rightarrow (\frac{H}{4}, \frac{W}{4}, 256)$	conv-(K-4×4, N-256, S-2, P-2, PV, IN, LReLU)
	$(\frac{H}{4}, \frac{W}{4}, 256) \rightarrow (\frac{H}{4}, \frac{W}{4}, 512)$	conv-(K-4×4, N-512, S-1, P-2, PV, IN, LReLU)
	$(\frac{H}{4}, \frac{W}{4}, 512) \rightarrow (\frac{H}{4}, \frac{W}{4}, 512)$	conv-(K-4×4, N-512, S-1, P-2, PV, IN, LReLU)
Output Layer	$(\frac{H}{4}, \frac{W}{4}, 512) \rightarrow (\frac{H}{4}, \frac{W}{4}, 1)$	conv-(K-4×4, N-1, S-1, P-2, PV, sigmoid)

Table 9: Network architecture of domain discriminator D_d.

Part Name	Input $\rightarrow$ Output Shape	Layer Description
Hidden Layers	$(H, W, 3) \rightarrow (\frac{H}{2}, \frac{W}{2}, 64)$	conv-(K-4×4, N-64, S-2, P-1, PV, LReLU)
	$(\frac{H}{2}, \frac{W}{2}, 64) \rightarrow (\frac{H}{4}, \frac{W}{4}, 128)$	conv-(K-4×4, N-128, S-2, P-1, PV, BN, LReLU)
	$(\frac{H}{4}, \frac{W}{4}, 128) \rightarrow (\frac{H}{8}, \frac{W}{8}, 256)$	conv-(K-4×4, N-256, S-2, P-1, PV, BN, LReLU)
	$(\frac{H}{8}, \frac{W}{8}, 256) \rightarrow (\frac{H}{16}, \frac{W}{16}, 512)$	conv-(K-4×4, N-512, S-2, P-1, PV, BN, LReLU)
Pooling Layer	$(\frac{H}{32}, \frac{W}{32}, 512) \rightarrow (1, 1, 512)$	average-pooling & dropout=0.999
Output Layer	$(1, 1, 512) \rightarrow$ C-way softmax	conv-(K-1×1, N-C, S-2, softmax)

Table 10: Network architecture of camera discriminator D_{cam}.

Training Procedures. As aforementioned in Alg. 1, the training process is divided into three steps. First, for initialisation, we pre-train the identity discriminator (ResNet50), camera discriminator for 30,000 iterations. Second, we train the image generator, domain discriminator from scratch for 60,000 iterations. Third, we fine-tune the ResNet50 using synthetic data produced by the image generator on-the-fly. We only apply random flipping as data augmentation.

6.2. Additional Ablation Study

We additionally illustrate the superiority of using CR-GAN to produce realistic synthetic data in comparison to an easy "*cut, paste and learn*" [12] image synthesis approach originally proposed for instance detection. Specifically, we first *cut* the source person segment and *paste* it to the target background. Then, we train the re-id model upon the "*cut and paste"* synthetic data. Figure 9 illustrates that the "*cut and paste"* synthetic data not only contains various artifacts – some identity relevant cue (e.g. backpack) is missing due to incomplete person mask; but it also cannot capture the lighting nor colour tones of the target domain. These limitations are in line with its weaker performance as shown in Table 11, where "*cut, paste and learn*" yields even worse re-id results than "Direct Transfer". Overall, this demonstrates the necessity of designing our CR-GAN to generate synthetic training data in higher fidelity and diversity for enhancing the cross-domain generalisability.

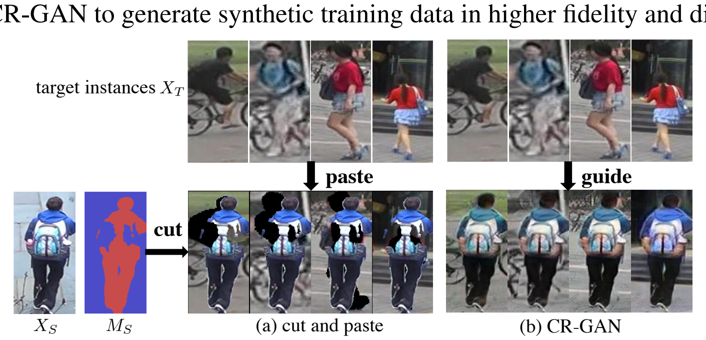

Figure 9: Synthetic images by (a) "*cut and paste"* and (b) CR-GAN. X_S: source image; X_T: target image; M_S: parsing mask of X_S.

$S \rightarrow T$	Market→Duke		Duke→Market	
Metrics (%)	R1	mAP	R1	mAP
Direct Transfer	36.9	20.5	47.5	20.0
cut, paste and learn [12]	21.6 ↓	9.0 ↓	26.5 ↓	11.3 ↓
CR-GAN	**52.2**	**30.0**	**59.6**	**29.6**
CR-GAN+LMP	**56.0**	**33.3**	**64.5**	**33.2**

Table 11: Ablation study in comparison to "*cut, paste and learn*".

6.3. Additional Qualitative Results

We additionally visualise the synthetic data by CR-GAN on four different domain pairs as shown in Figure 10, 11, 12, 13. The visualisation shows that CR-GAN is capable of producing abundant data augmented with different *background clutters*, *colour tones* and *lighting conditions*, explicitly guided by the target instances randomly sampled from the target domain.

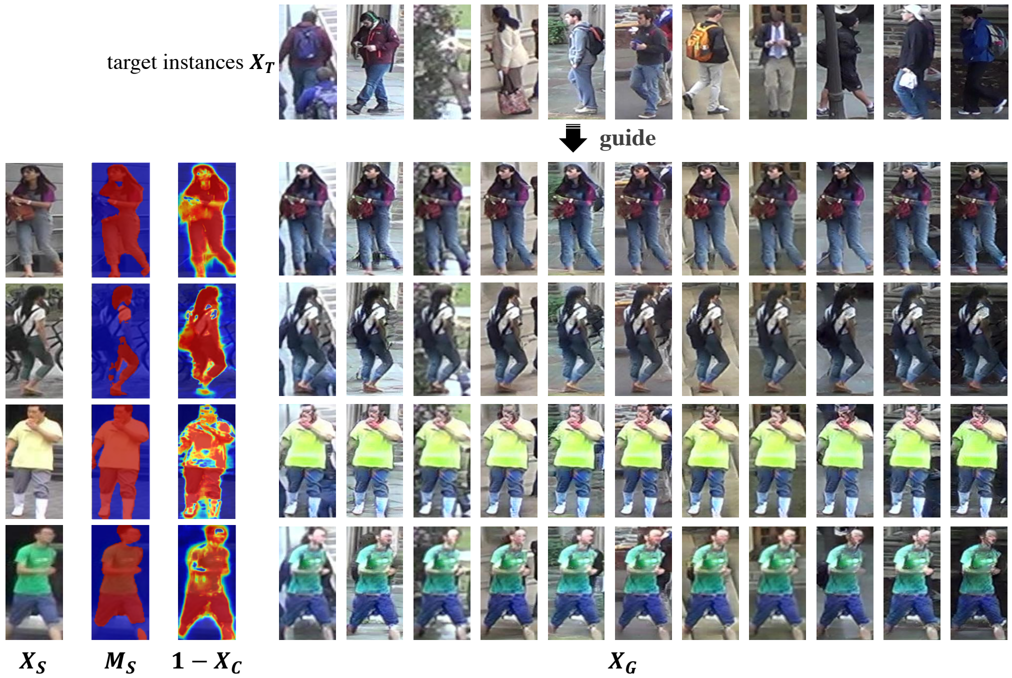

Figure 10: Synthetic data by CR-GAN on Market1501→DukeMTMCreID. X_S: source image; X_T: target image; M_S: parsing mask of X_S; $1 - X_C$: the inverse of context mask; X_G: generated image.

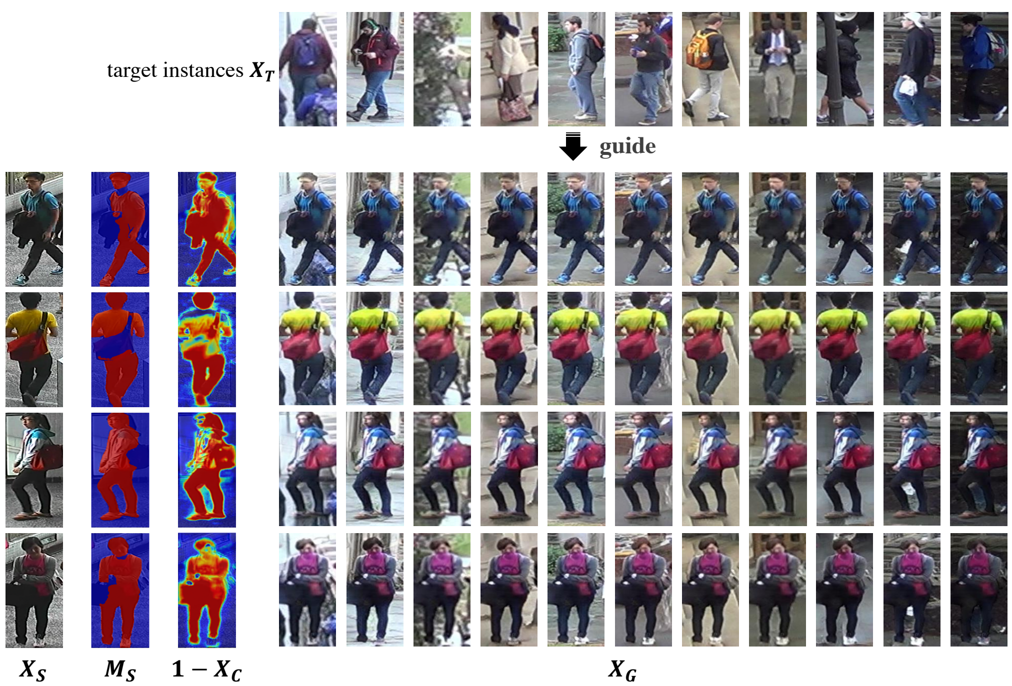

Figure 11: Synthetic data by CR-GAN on CUHK03 → DukeMTMCreID. X_S: source image; X_T: target image; M_S: parsing mask of X_S; $1 - X_C$: the inverse of context mask; X_G: generated image.

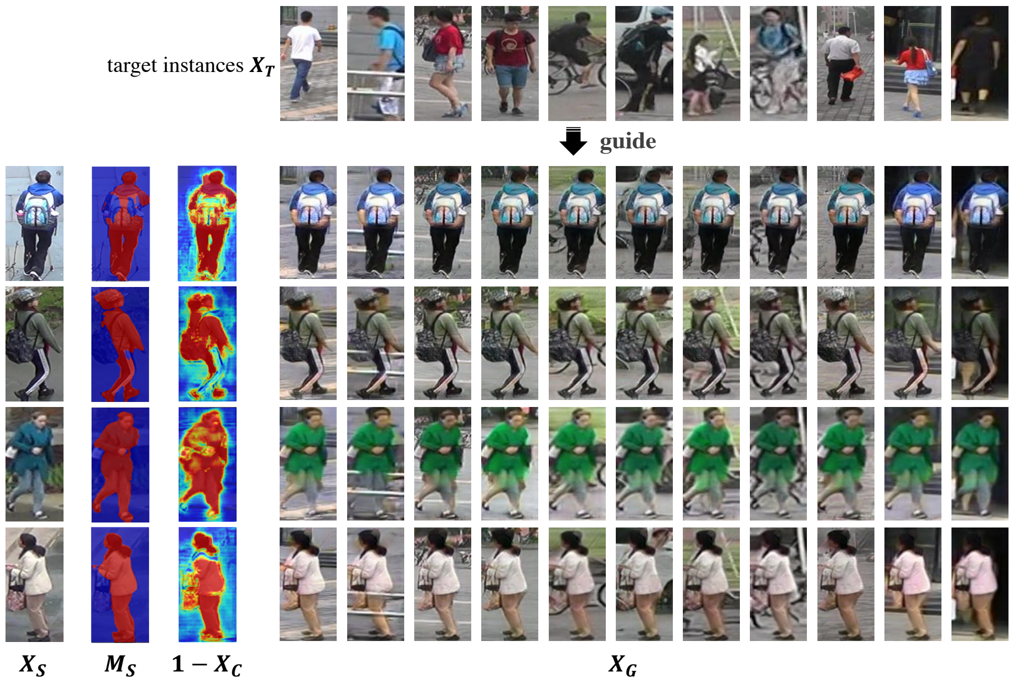

Figure 12: Synthetic data by CR-GAN on DukeMTMCreID → Market1501. X_S: source image; X_T: target image; M_S: parsing mask of X_S; $1 - X_C$: the inverse of context mask; X_G: generated image.

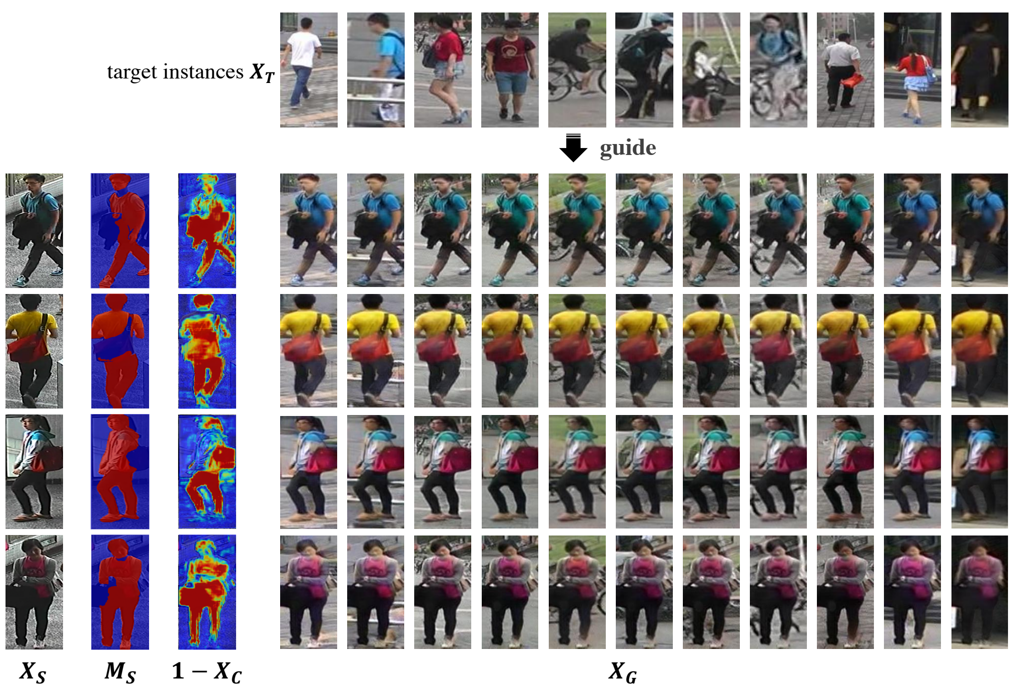

Figure 13: Synthetic data by CR-GAN on CUHK03 → Market1501. X_S: source image; X_T: target image; M_S: parsing mask of X_S; $1 - X_C$: the inverse of context mask; X_G: generated image.