Domain-Specific Mappings for Generative Adversarial Style Transfer Supplementary Material

Hsin-Yu Chang, Zhixiang Wang, and Yung-Yu Chuang

National Taiwan University

A Overview

In this supplementary material, we present additional results to complement the main manuscript. We first demonstrate the trade-off between content-preserving and style translation in Section B. Second, we discuss the domain-specific mappings and the intuition behind in Section C. Third, we provide the implementation details, including network architectures and hyper-parameter settings in Section D. Then, we demonstrate the disentanglement representation ability of the latent features in Section E. Finally, we show the complete quantitative result of FID/LPIPS/SSIM and user study in Section F, and more visual comparisons in Section G.

B The trade-off problem

Most I2I methods make trade-offs between content preservation and style translation. They decompose features into style and content ones in individual spaces. Although each domain has its own style space, the content space is shared among domains, which could compromise the content representation power and generate unsatisfactory results. The problem is severe, especially when applying these methods to the tasks requiring semantic matches (Fig. B.1).[1]

For addressing the problem, the paper proposes domain-specific mappings to remap content features from the shared space to the individual space of each domain. It learns the content features in individual image domains to provide content information specific to the domain. Therefore, for style translation, it adjusts the content features so that they better match the ones in the target domains and cooperate better with the domain-specific style features for synthesizing better results. Thus, our method can reach a better trade-off between content preservation and style translation than previous methods.

[1] Note that the performance of MSGAN is close to DRIT since it is an advanced version of DRIT, and thus we only compare with MSGAN in the paper.

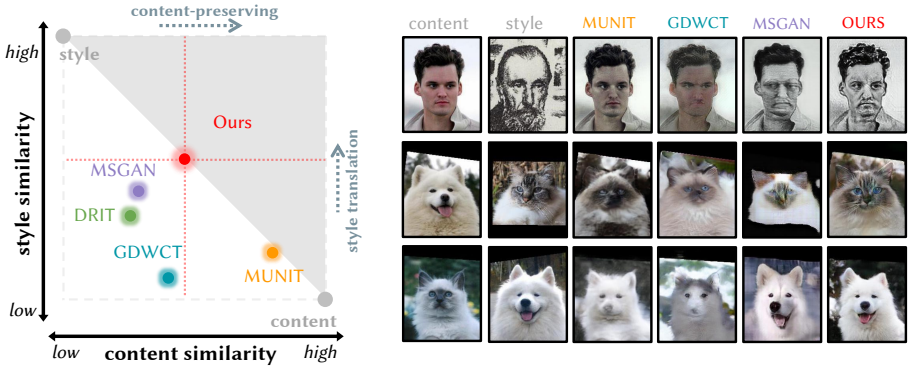


Fig. B.1: Trade-off between content-preserving and style translation.

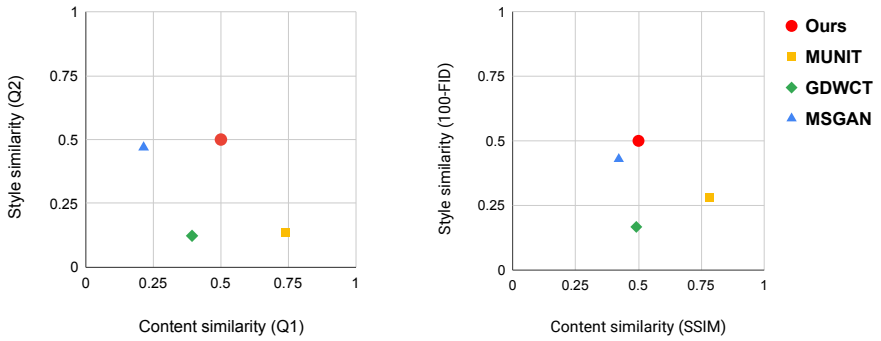


Fig. B.2: Quantitative illustration of the trade-off problem. We use the quantitative average value of FID, SSIM and user study on the different datasets to support the trade-off observation. We use 100−FID score to evaluate style similarity and SSIM scores to evaluate content similarity, and the xy-axis are both the higher the better. For evaluation, we first set our scores to 0.5 in both content and style similarity, and we change the value of other methods according to the scalar we use on our scores. Then normalize their scores to [0,1]. We put the original FID, SSIM and user study in Section.F. In both figures, the methods have the similar behavior, for example, MUNIT can perform better content-preserving while MSGAN can perform better style translation.

C Domain-specific mappings and intuition

Intuition. As we mention in the paper, the domain-specific content space could encode better information that we want to use for the cross-domain generation. However, we only have domain-invariant content space in the previous method (MUNIT/DRIT). That is why we need to remap the domain-invariant space to two individual domain-specific content ones. To learn the mapping, we still

need domain-invariant content space, the common space for the cross-domain training. With domain-invariant content space, we only need to learn how to remap to domain-specific content space from domain-invariant content space. As we describe in Section 3, we can learn the mapping with L_1^{dsc}.

Why the proposed mappings powerful? Please refer to Figure 3 for the following discussion. Take $A \rightarrow B$ as the example. First, we obtain the domain-specific features $h_A \in C^{DS}$. As other methods, h_A is then encoded to the shared domain-independent space, $c_A \in C^{DI}$. For learning domain A's specific information, we find the mapping $\Phi_{C \rightarrow C_A}$ to map a feature from C^{DS} to domain A's own space C_A^{DS}. The loss L_1^{dsc} ensures the remapped feature ensembles the domain-specific feature h_A and thus is essential to the learning of $\Phi_{C \rightarrow C_A}$. After learning both $\Phi_{C \rightarrow C_A}$ and $\Phi_{C \rightarrow C_B}$, for $A \rightarrow B$, we remap c_A to $c_{A \rightarrow B} \in C_B^{DS}$ using $\Phi_{C \rightarrow C_B}$ so that the content $c_{A \rightarrow B}$ is better aligned to the target domain B. It is why the proposed method can provide better domain alignment than existing methods.

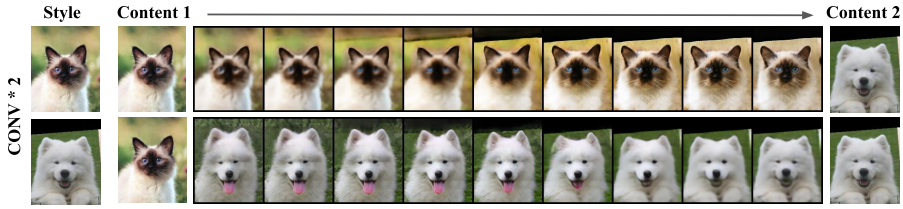
Why not use h_A and h_B directly for cross-domain training? Note that, although the content feature h_A and h_B are domain-specific, we cannot use them directly to generate cross-domain results. Because we only have $h_A \in C_A^{DS}$ but not $h_A \in C_B^{DS}$ in the training stage, we need the mapping function to remap to the target content domain.

Architecture. Rather than a complicated network such as U-Net, we implement the mapping functions with only two convolution layers, as shown below.

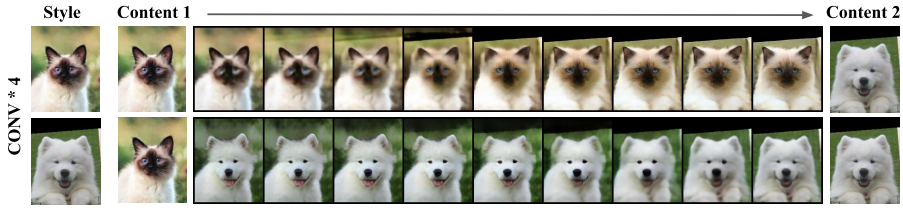
$$\text{DCONV(N256,K3,S2)} \rightarrow \text{ReLU} \rightarrow \text{CONV(N256,K3,S2)}$$

Experiments show the model is simple yet effective, thus saving much training time.

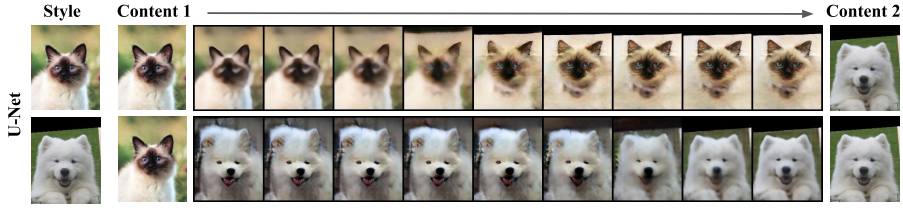
Why not use other architectures? We have experimented with several architectures for the mapping function: (a) two CONV layers, (b) four CONV layers, (c) U-Net, and (d) our setting depicted above. The mapping function needs to remap properly domain-invariant content features to domain-specific content ones. Thus, for validating which architecture is effective, we perform the latent interpolation, that is, performing interpolation between features in the latent space and using the mapping function to remap the interpolated content feature to the domain-specific space. The remapped feature is then combined with the style feature to generate the final image. If the mapping is effective, the results better match the target domain. We use the checkpoint model for inference at step $150,000$. Fig. C.3 and Fig. C.4 show the interpolation results for different mapping architectures. The tow-layer CONVs (Fig. C.3(a)) performs worse than its four-layer counterpart (Fig. C.3(b)). U-Net (Fig. C.3(c)) works poorly because it needs much more time to be trained well. Fig. C.3(d) shows the result of our setting. Both the four-layer CONVs (Fig. C.3(b)) and our setting (Fig. C.3(d)) are good enough for learning the mapping. We choose our setting for less training complexity and slightly better results.



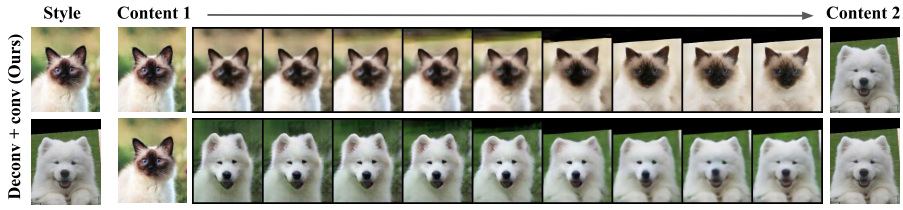
(a) Two CONV layers



(b) Four CONV layers

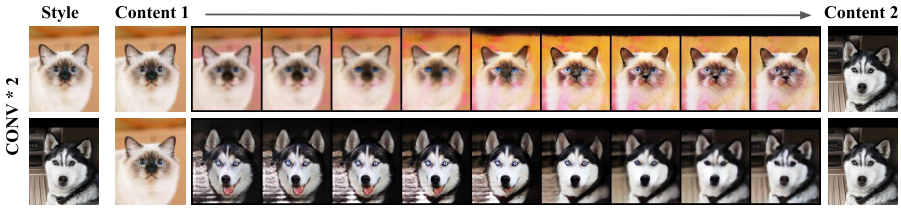


(c) U-Net

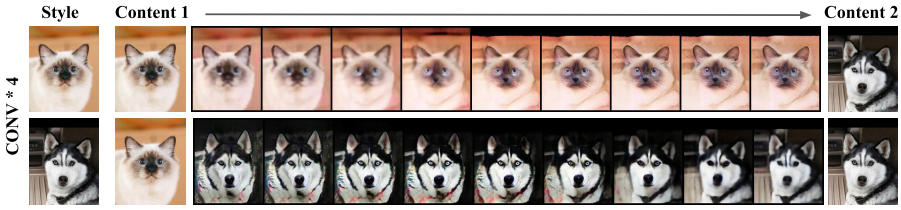


(d) Our setting

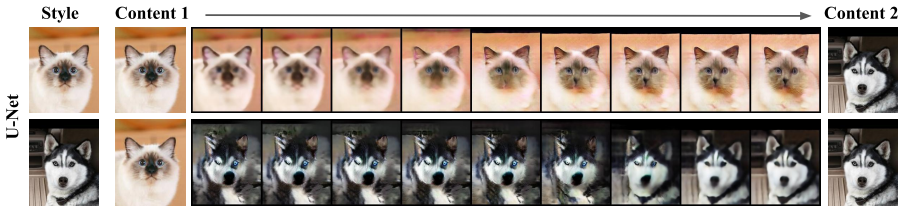
Fig. C.3: Comparisons of different mapping architectures.



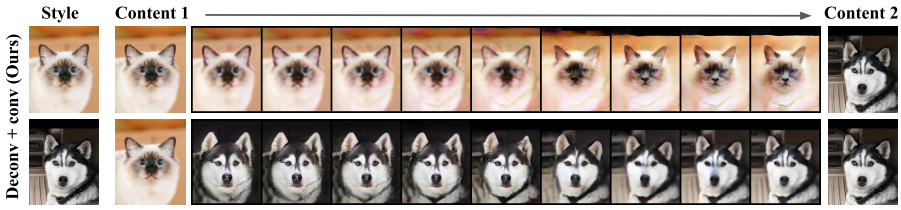
(a) Two CONV layers



(b) Four CONV layers



(c) U-Net



(d) Our setting

Fig. C.4: Comparisons of different mapping architectures (cont.).

D Implementation details

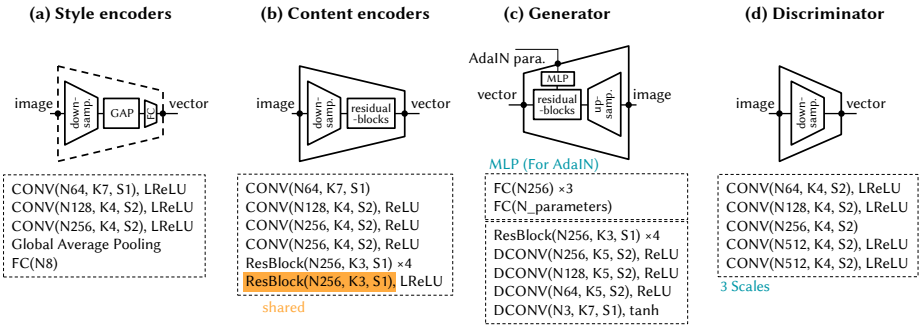


Fig. D.5: Network Architectures.

Style encoders. For the style encoders E^s_A and E^s_B, we use three convolution layers and a global average pooling then followed by one fully-connected layer. We set the dimension of style feature to 8 in our experiments (Fig. D.5(a)).
Content encoders. For the content encoders E^c_A and E^c_B, we use four convolution layers followed by four individual residual blocks and one residual block, which shares weights between domains (Fig. D.5(b)). After the shared residual block, we remap the domain-invariant content space to the domain-specific content space.
Generators. For the generators G_A and G_B, we use four residual blocks followed by four de-convolution layers to do up-sampling. To get AdaIN parameters, we feed style feature to three fully connected layers then assign to residual blocks in generator G_A and G_B (Fig. D.5(c)).
Discriminators. For the discriminators D_A and D_B, we apply three scale discriminators, there are three convolution layers and one fully-connected layer in each scale (Fig. D.5(d)).
Number of network parameters. Table D.1 reports the sizes for several models. We construct our model by following MUNIT and adding the proposed mapping functions. Our model is only slightly larger than MUNIT. Thus, the superior results come from the designs of the mapping scheme and losses more than the model size. For comparisons, MSGAN has 75.4M parameters.

Table D.1: Number of network parameters.

	MUNIT	GDWCT	MSGAN	Ours
# of parameters	46.6M	51.0M	75.4M	55.2M

Hyper-parameter setting. We adopt the same hyper-parameter setting in all experiments. We also found the L^x_1 and L^s_1 are more important than L^{dic}_1 and L^{dsc}_1. We use Adam optimizer with the learning rate of $1e^{-4}$, and empirically set the weights as $\lambda_{cc} = 6$, $\lambda_x = 10$, $\lambda_s = 10$, $\lambda_{dsc} = 2$, $\lambda_{dic} = 2$ and $\lambda_{adv} = 1$.

E Disentanglement representation

Fig. E.6–E.9 show the disentanglement representation of latent features when giving the same content image or the same style image. The images of the same row come from the same content image while those of the same column share the same style. The images of the same row have similar spatial layouts, such as sizes and poses, showing that the content is preserved well. At the same time, the images of the same column look like the same species, meaning that the style is well transferred.

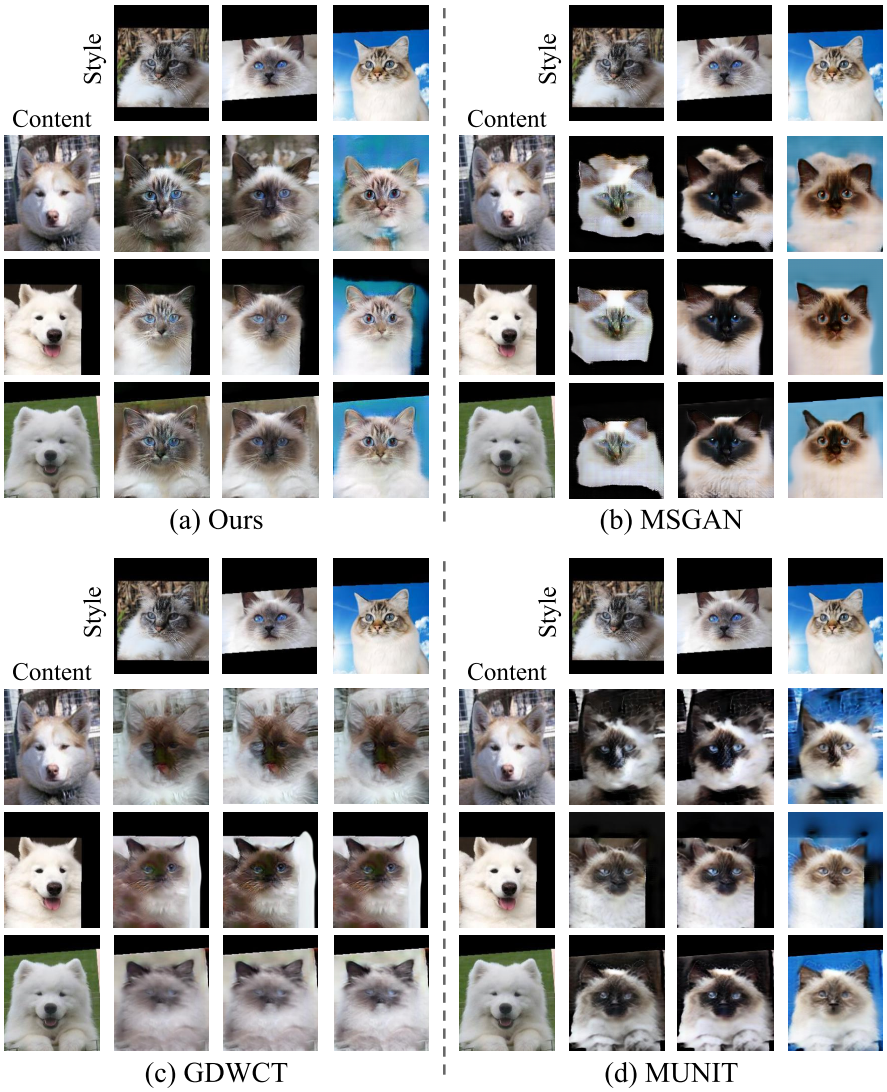


Fig. E.6: Compared results of disentanglement representation on `dog→cat`.

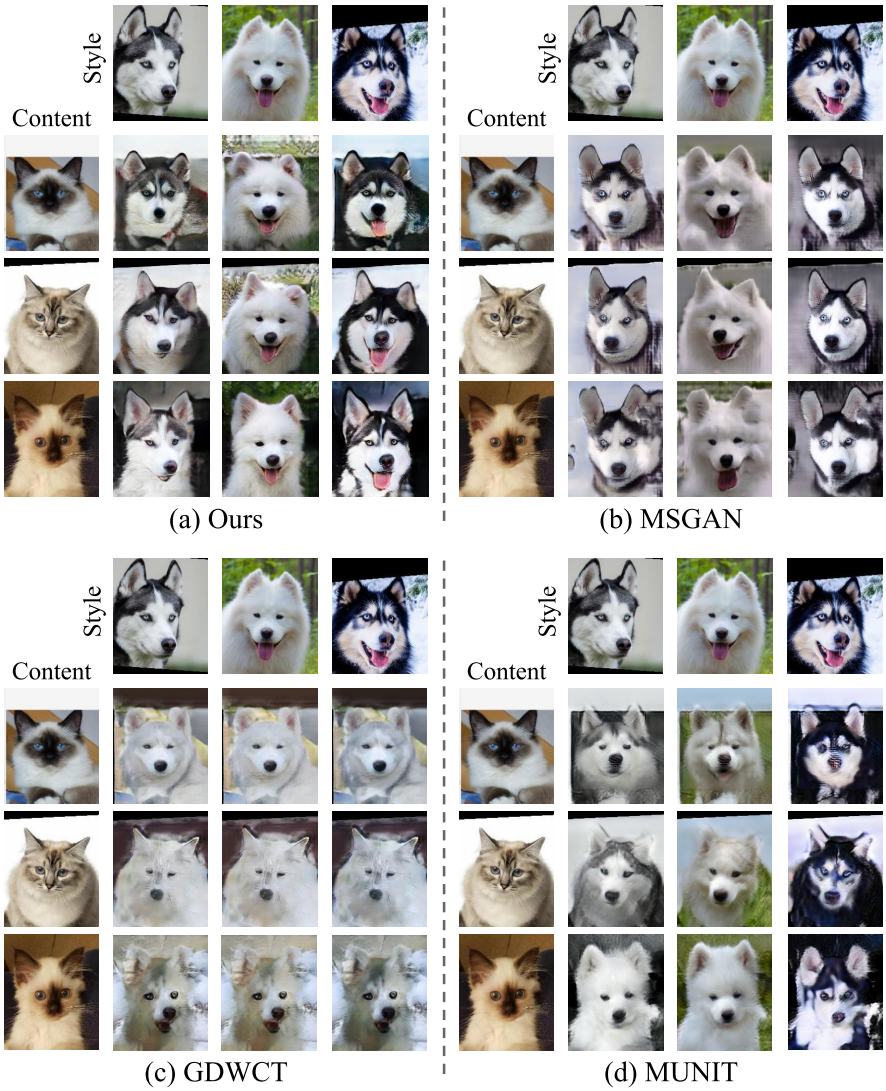


Fig. E.7: Compared results of disentanglement representation on `cat`→`dog`.

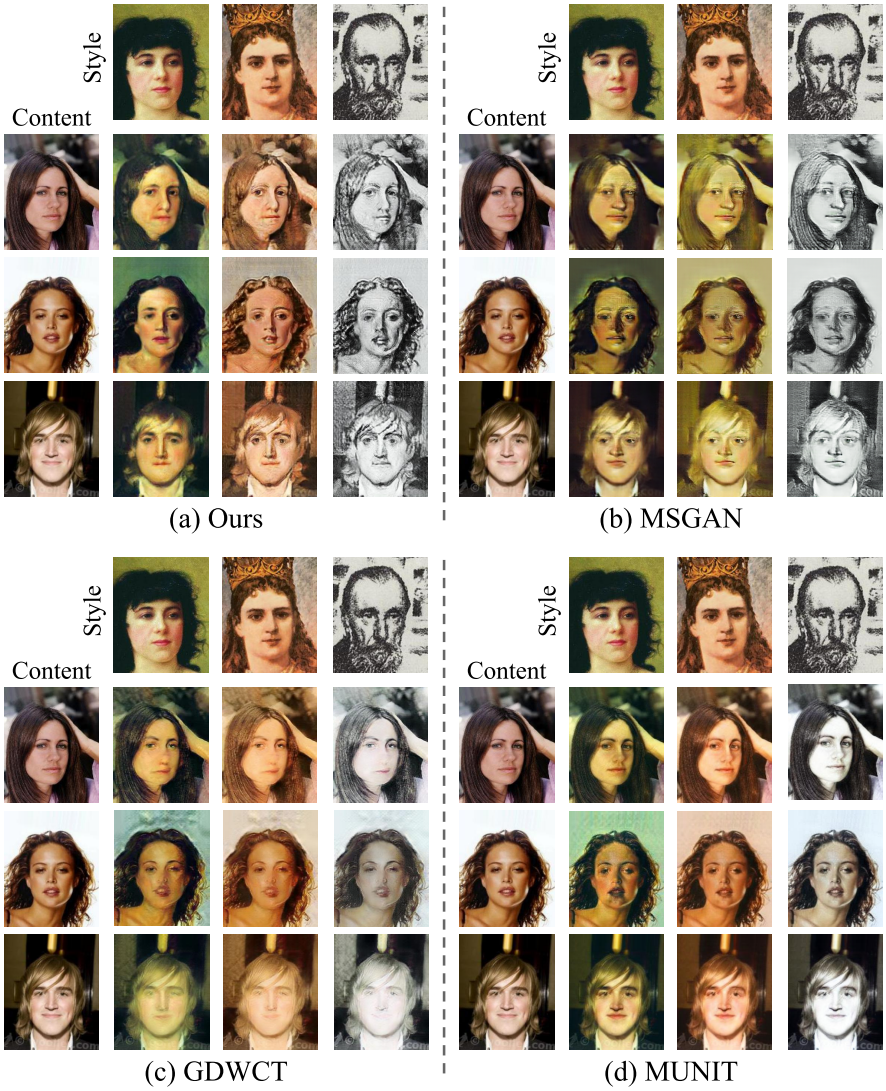


Fig. E.8: Compared results of disentanglement representation on `photograph→portrait`.

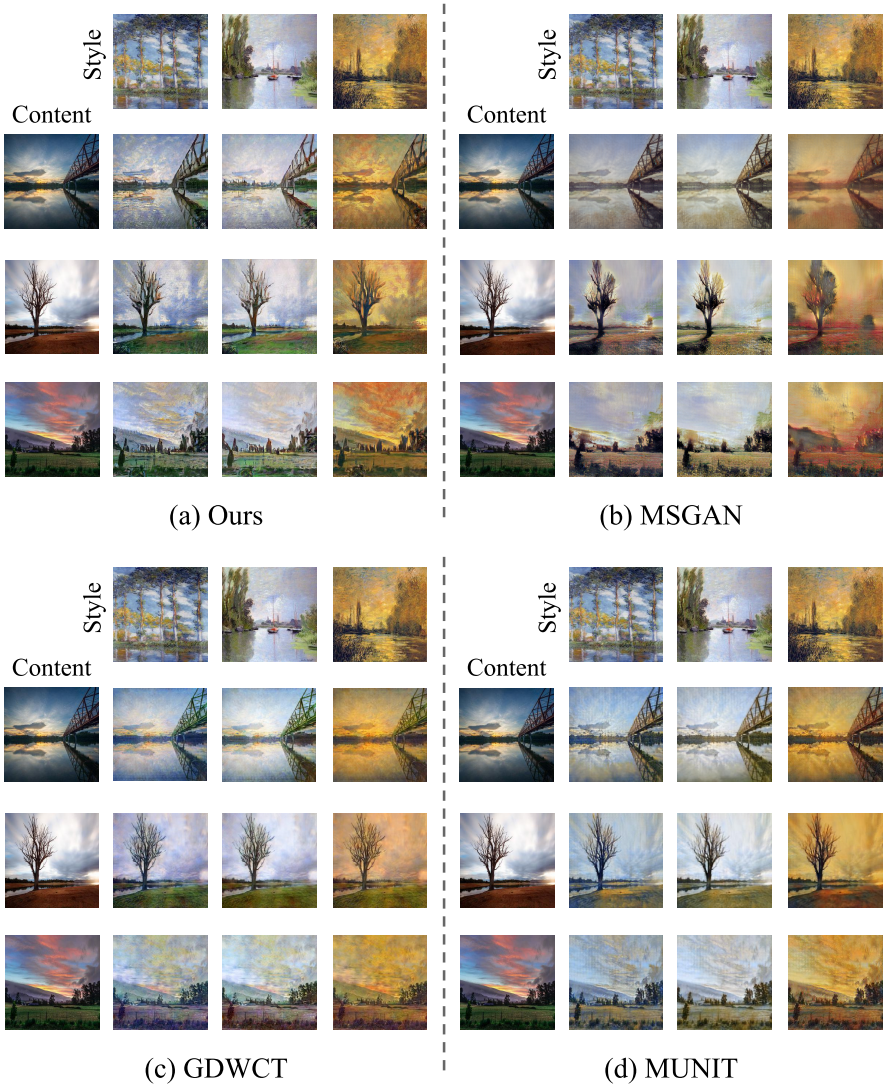


Fig. E.9: Compared results of disentanglement representation on `photo→Monet`.

F Complete quantitative results

We show the complete FID/LPIPS results with standard deviation in Table F.2 and Table F.3. The additional results of SSIM are shown in Table F.4. Notice that we calculate SSIM between the input content image and generated results; the higher score means the better structural similarity, but the score can not reflect the ability of style translation.

For the user study, we show the complete results in Fig. F.10 of each method compared to ours in the following three questions.

- *Q1: Which one preserves content information (identity, shape, semantic) better?*
- *Q2: Which one performs better style translation (in terms of color, pattern)?*
- *Q3: Which one is more likely to be a member of the domain B?*

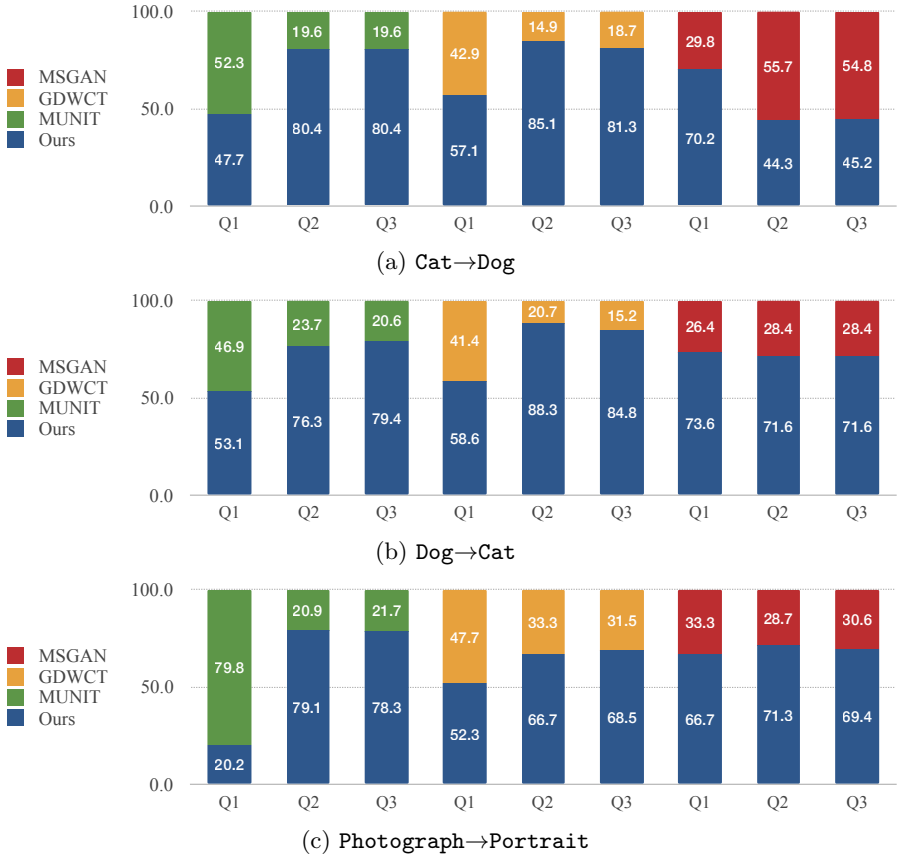


Fig. F.10: Complete user study results.

Table F.2: Results of FID score. We use the FID score (↓) to evaluate the quality of each method on six types of translation tasks. **Red** text indicates the best and blue text indicates the second best performing method.

	MUNIT	GDWCT	MSGAN	Ours
Cat → Dog	38.09 ± 0.46	91.40 ± 1.31	20.80 ± 0.49	13.60 ± 0.17
Dog → Cat	39.71 ± 0.39	59.72 ± 1.03	28.30 ± 0.58	19.69 ± 0.22
Monet → Photo	85.06 ± 0.73	113.16 ± 0.73	86.72 ± 0.91	81.61 ± 1.37
Photo → Monet	77.85 ± 0.30	71.68 ± 0.53	80.37 ± 0.54	63.94 ± 0.35
Portrait → Photograph	93.45 ± 1.12	83.69 ± 0.89	57.07 ± 0.61	62.44 ± 0.63
Photograph → Portrait	89.97 ± 0.78	75.86 ± 0.45	57.84 ± 0.33	45.81 ± 0.37
Average	70.69	82.59	55.18	47.85

Table F.3: Results of LPIPS score. We use the LPIPS score (↑) to evaluate the diversity of each method on six types of translation tasks. Red text indicates the best and blue text indicates the second best performing method.

	MUNIT	GDWCT	MSGAN	Ours
Cat → Dog	0.3501 ± 0.002	0.1804 ± 0.002	0.5051 ± 0.002	0.4149 ± 0.003
Dog → Cat	0.3167 ± 0.003	0.1573 ± 0.003	0.4334 ± 0.003	0.3174 ± 0.001
Monet → Photo	0.4282 ± 0.002	0.2478 ± 0.002	0.4229 ± 0.002	0.5379 ± 0.003
Photo → Monet	0.4128 ± 0.005	0.2097 ± 0.005	0.4306 ± 0.004	0.4340 ± 0.003
Portrait → Photograph	0.1819 ± 0.002	0.1563 ± 0.002	0.3061 ± 0.003	0.3160 ± 0.002
Photograph → Portrait	0.1929 ± 0.001	0.1785 ± 0.002	0.2917 ± 0.002	0.3699 ± 0.002
Average	0.3131	0.1881	0.3978	0.3980

Table F.4: Results of SSIM score. We use the SSIM score (↑) to evaluate the structural similarity of each method on six types of translation tasks. We calculate the value between the input content image and generated results. Red text indicates the best and blue text indicates the second best performing method.

	MUNIT	GDWCT	MSGAN	Ours
Cat → Dog	0.192	0.103	0.112	0.215
Dog → Cat	0.188	0.048	0.094	0.196
Monet → Photo	0.212	0.183	0.159	0.108
Photo → Monet	0.202	0.224	0.131	0.128
Portrait → Photograph	0.470	0.280	0.232	0.239
Photograph → Portrait	0.510	0.277	0.228	0.248
Average	0.296	0.186	0.160	0.189

G Additional qualitative comparisons

We first show the `Summer→Winter` results of different style transfer and I2I methods in Fig. G.11. Then we show more multi-modal results with randomly sampled styles from a Gaussian distribution in Fig. G.12. Fig. G.13 shows the simple case in the `Cat→Dog` and `Dog→Cat` translation tasks. We can observe that all methods work reasonably well in simple cases, but some generate poor results in more difficult cases, as shown in the following.

Fig. G.14–Fig. G.19 shows the results when given the dog as content and cat as style. Fig. G.20–Fig. G.23 shows the results when given the cat as content and dog as style. Fig. G.24–Fig. G.28 shows the results when given the photograph as content and portrait as style. Fig. G.29–Fig. G.32 shows the results when given the photo as content and monet as style.

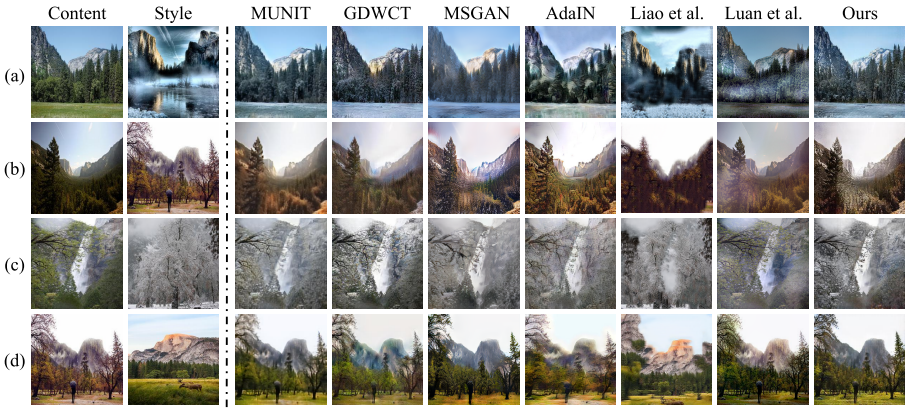
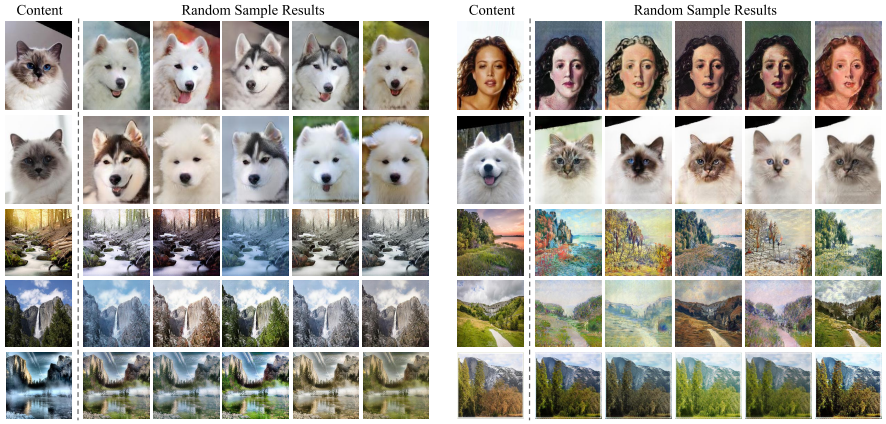


Fig. G.11: Comparisons with other methods on the Yosemite dataset.



(a) For top to bottom:
`Cat`→`Dog`,
`Cat`→`Dog`,
`Summer`→`Winter`,
`Summer`→`Winter`,
`Winter`→`Summer`.

(b) For top to bottom:
`Photo`→`Portrait`,
`Dog`→`Cat`,
`Photo`→`Monet`,
`Photo`→`Monet`,
`Winter`→`Summer`.

Fig. G.12: More multi-modal results.

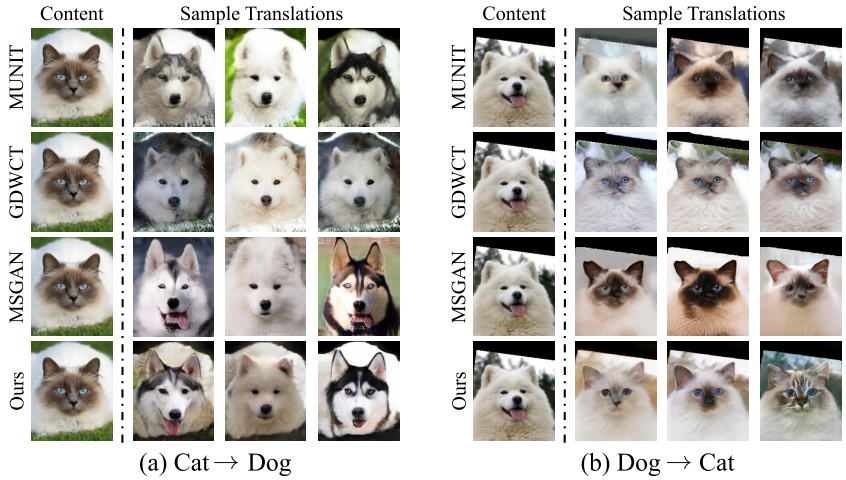


Fig. G.13: Results of different methods on the same **simple** content image with randomly selected style images. (a) `Cat`→`Dog` and (b) `Dog`→`Cat`.

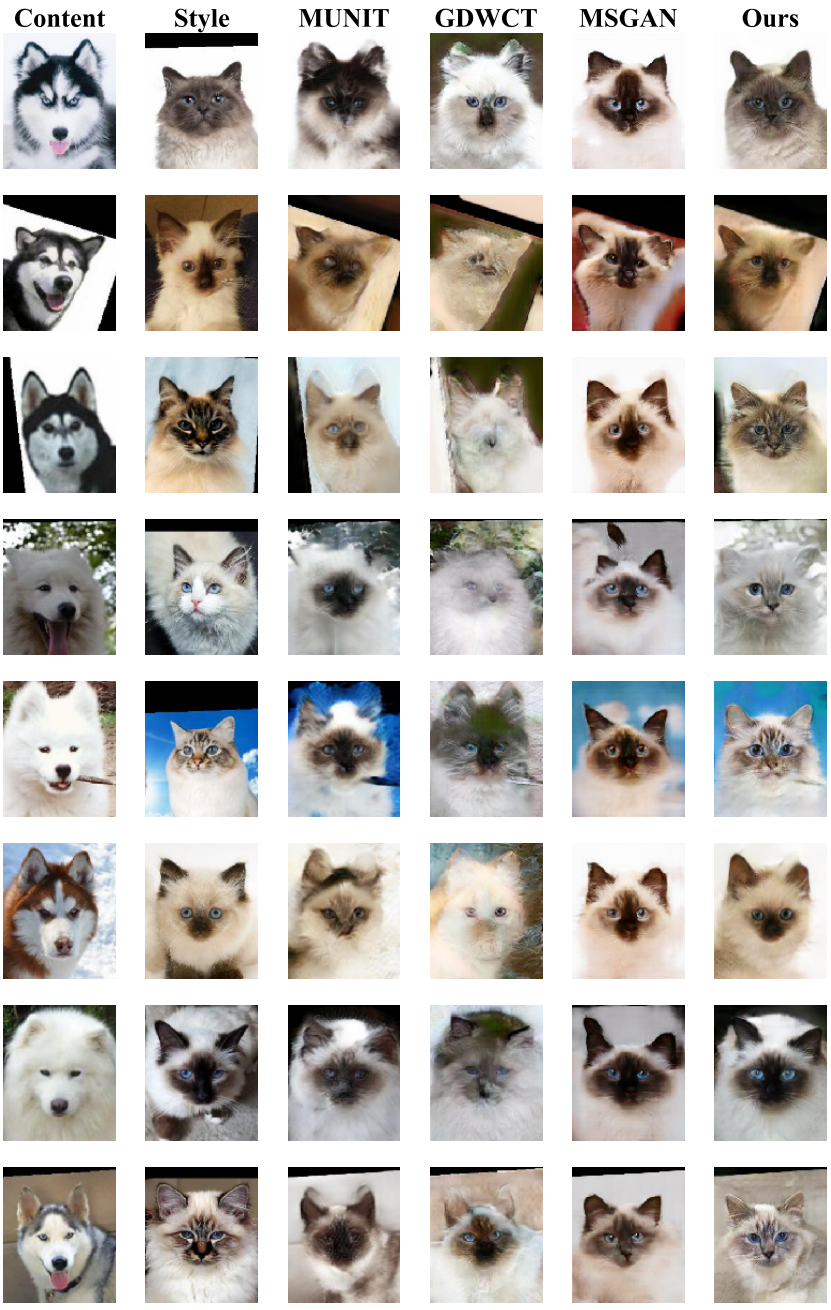


Fig. G.14: More compared results on `dog→cat`.

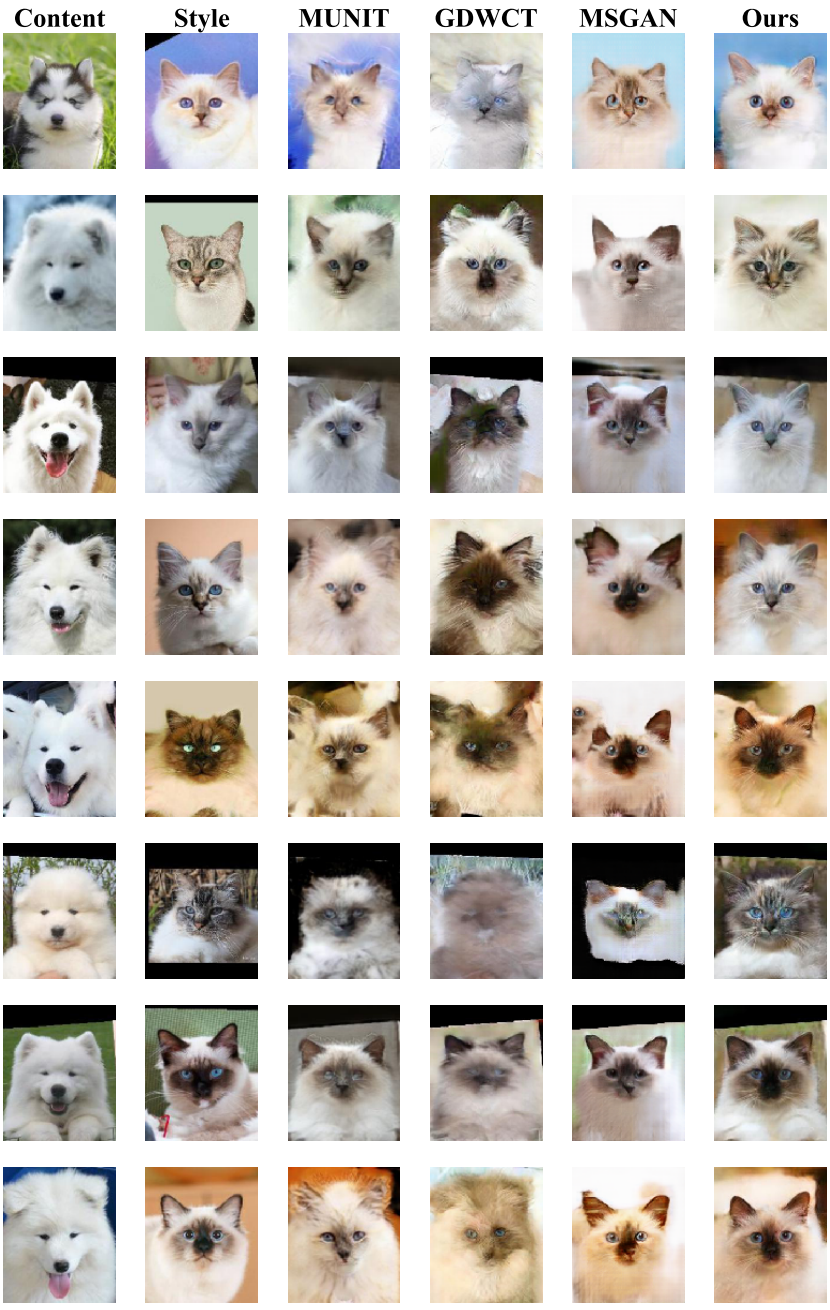


Fig. G.15: More compared results on dog→cat

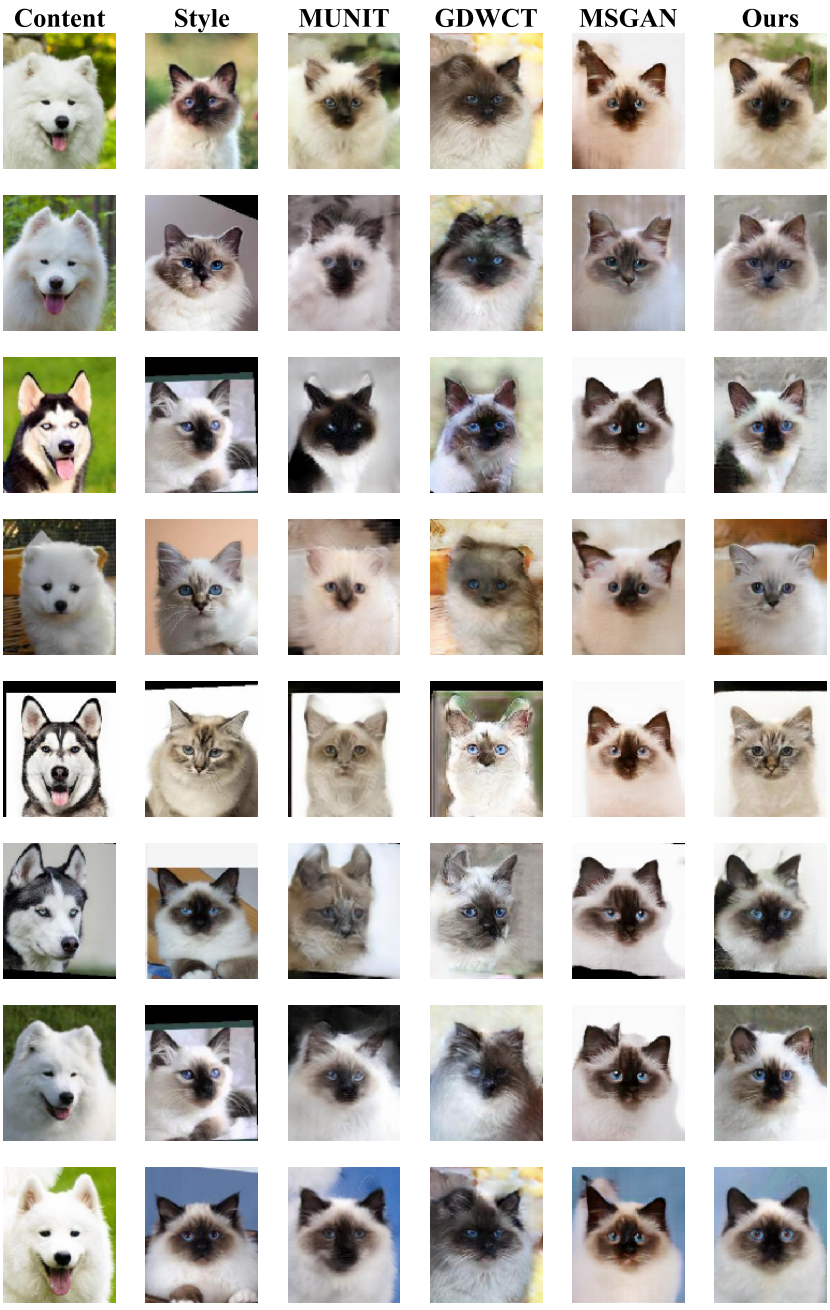


Fig. G.16: More compared results on `dog→cat`

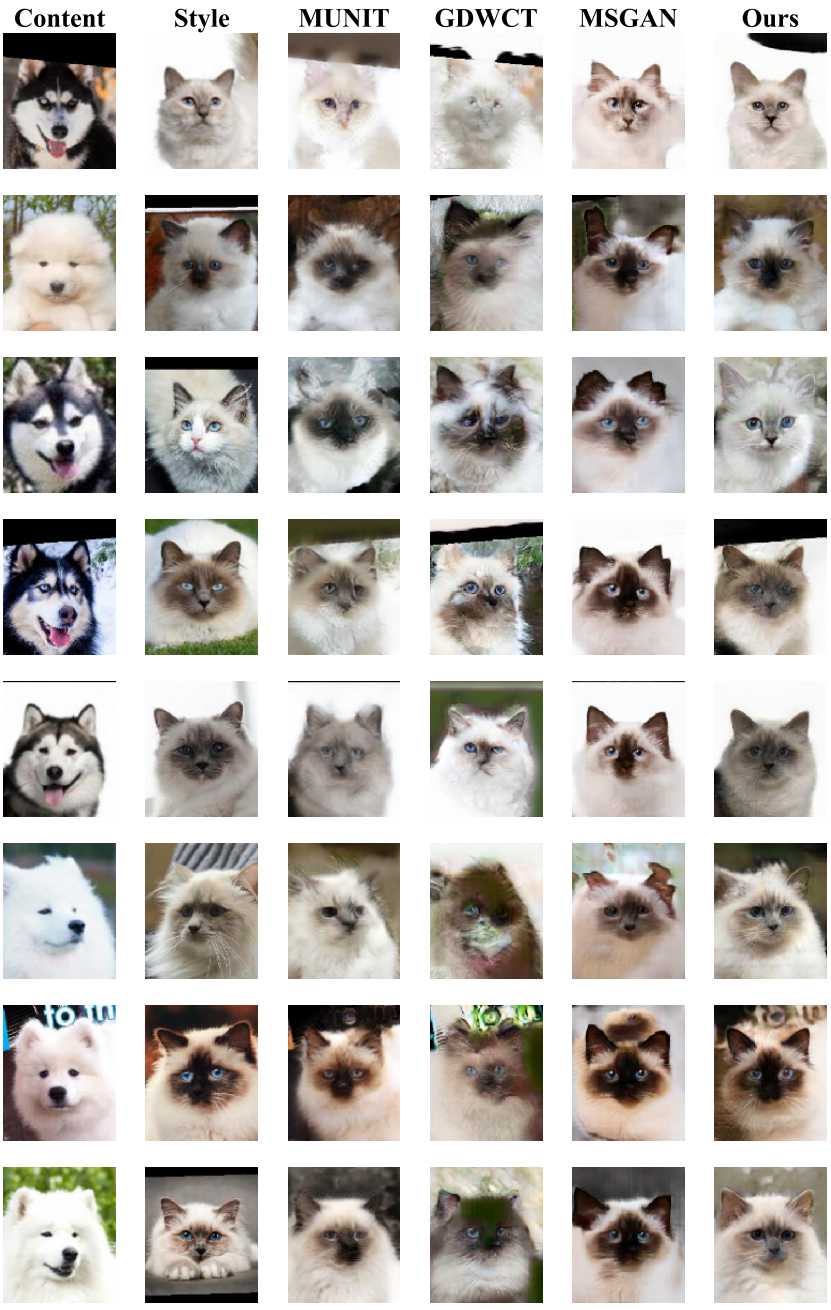


Fig. G.17: More compared results on dog→cat

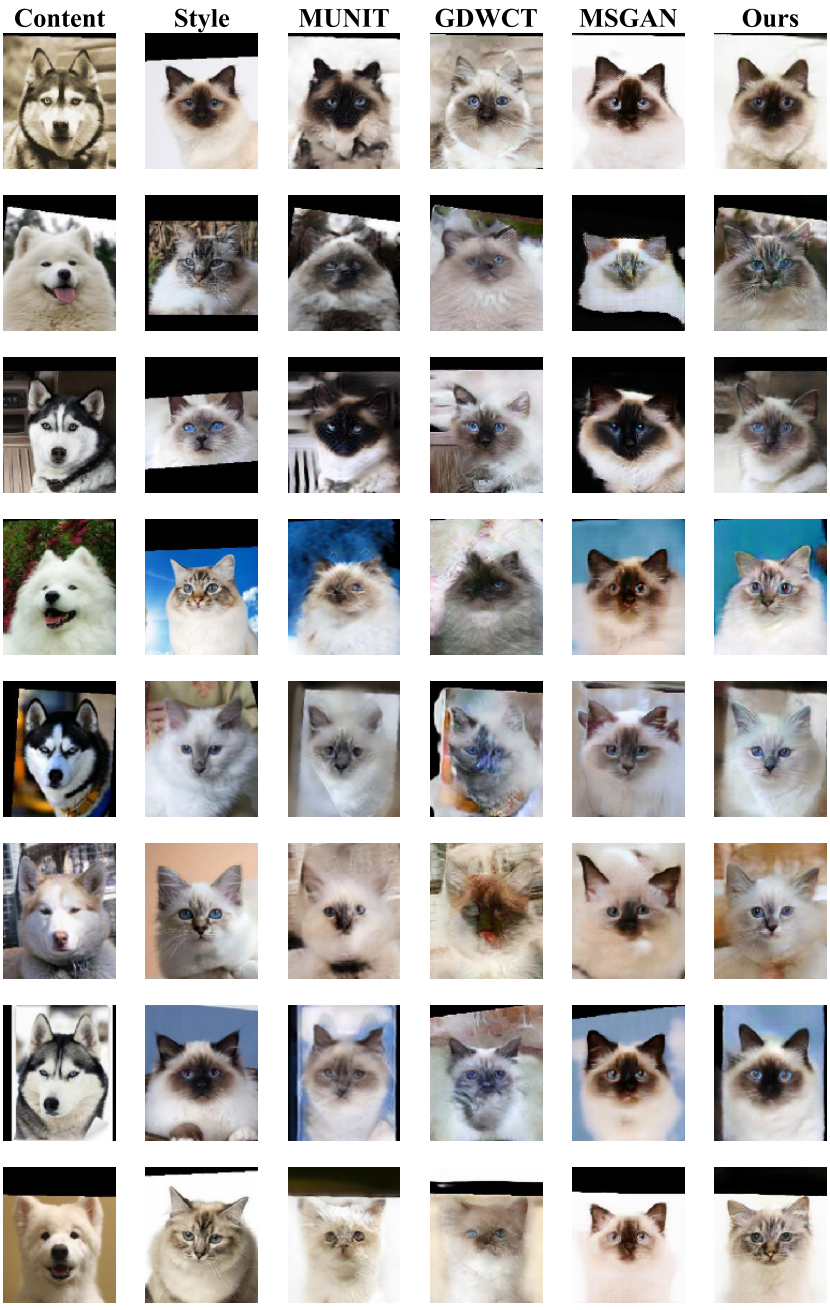


Fig. G.18: More compared results on `dog→cat`

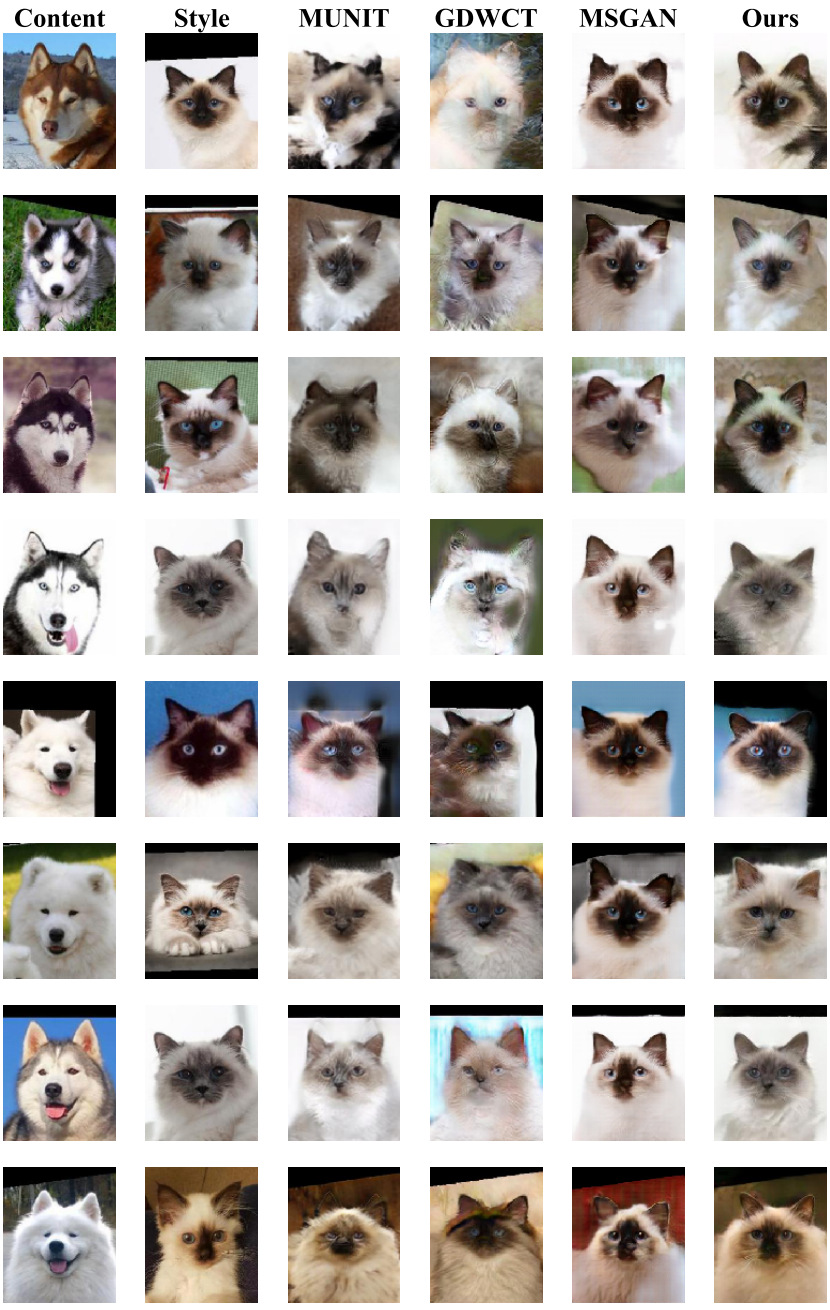


Fig. G.19: More compared results on dog→cat

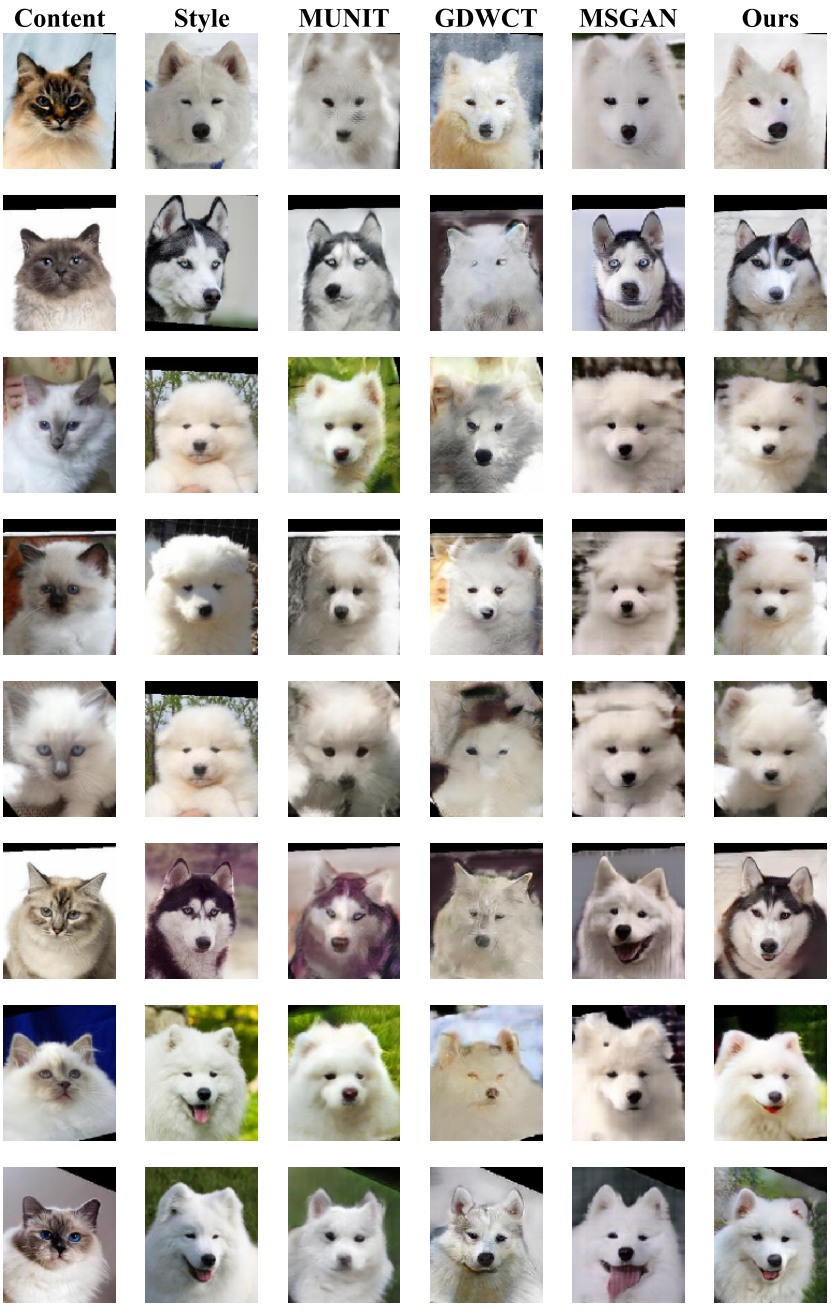


Fig. G.20: More compared results on `cat→dog`

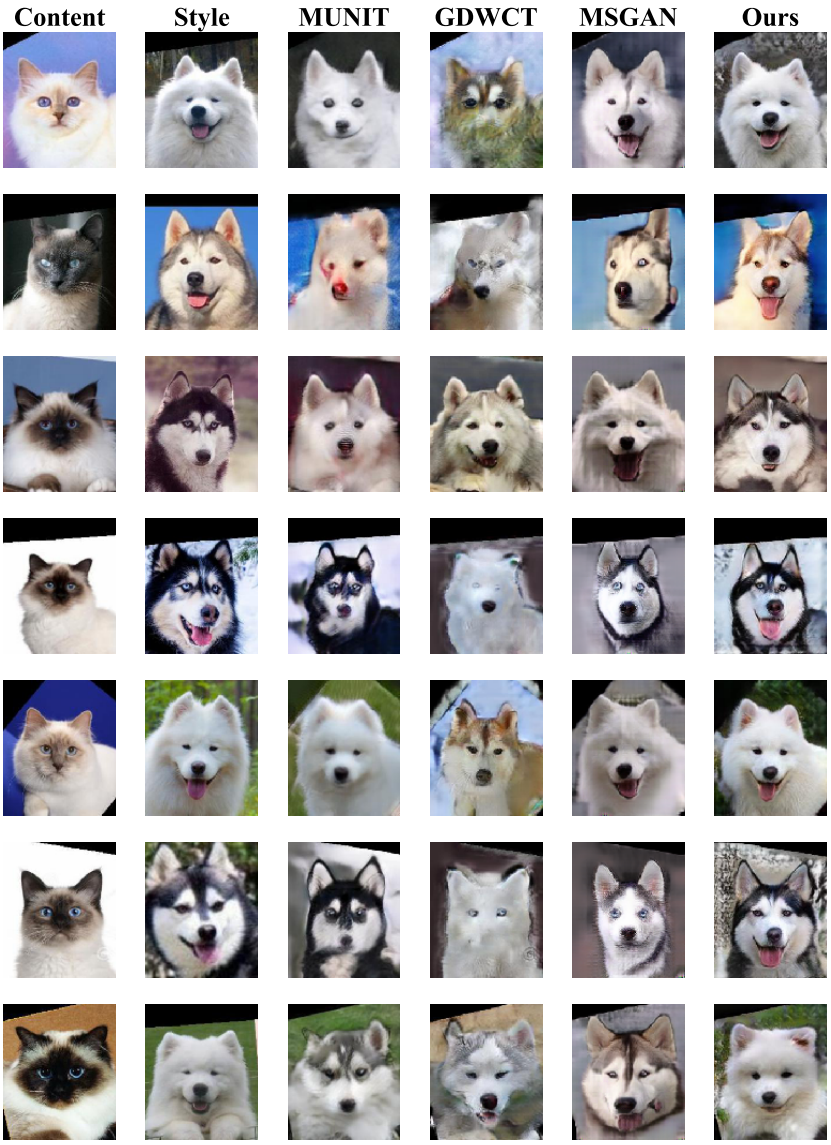


Fig. G.21: More compared results on `cat→dog`

Fig. G.22: More compared results on `cat→dog`

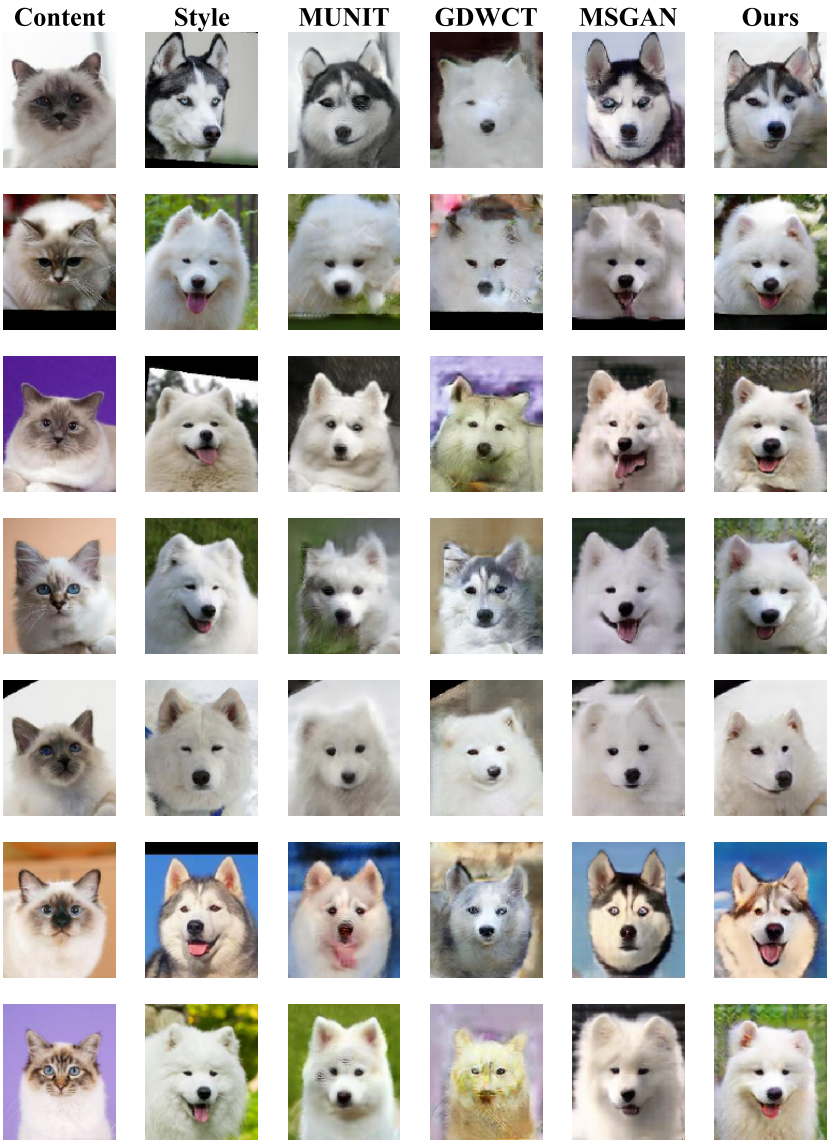


Fig. G.23: More compared results on `cat→dog`

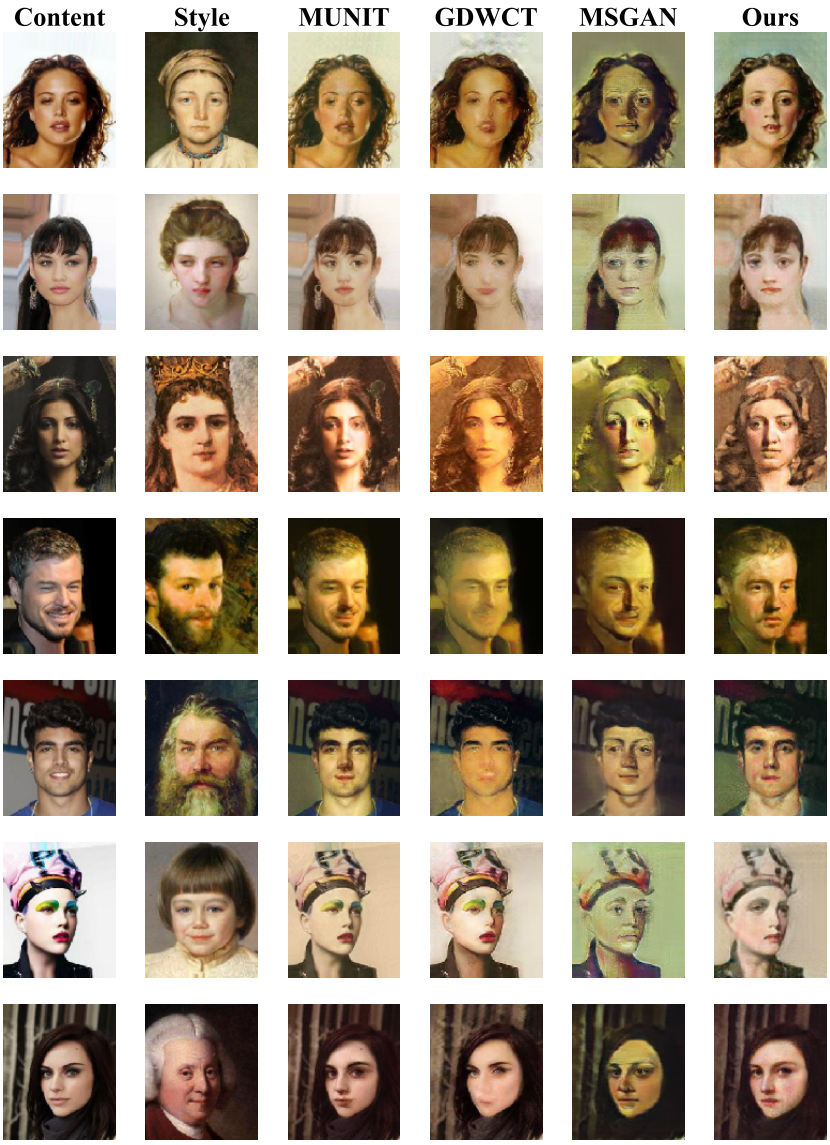


Fig. G.24: More compared results on `photograph→portrait`

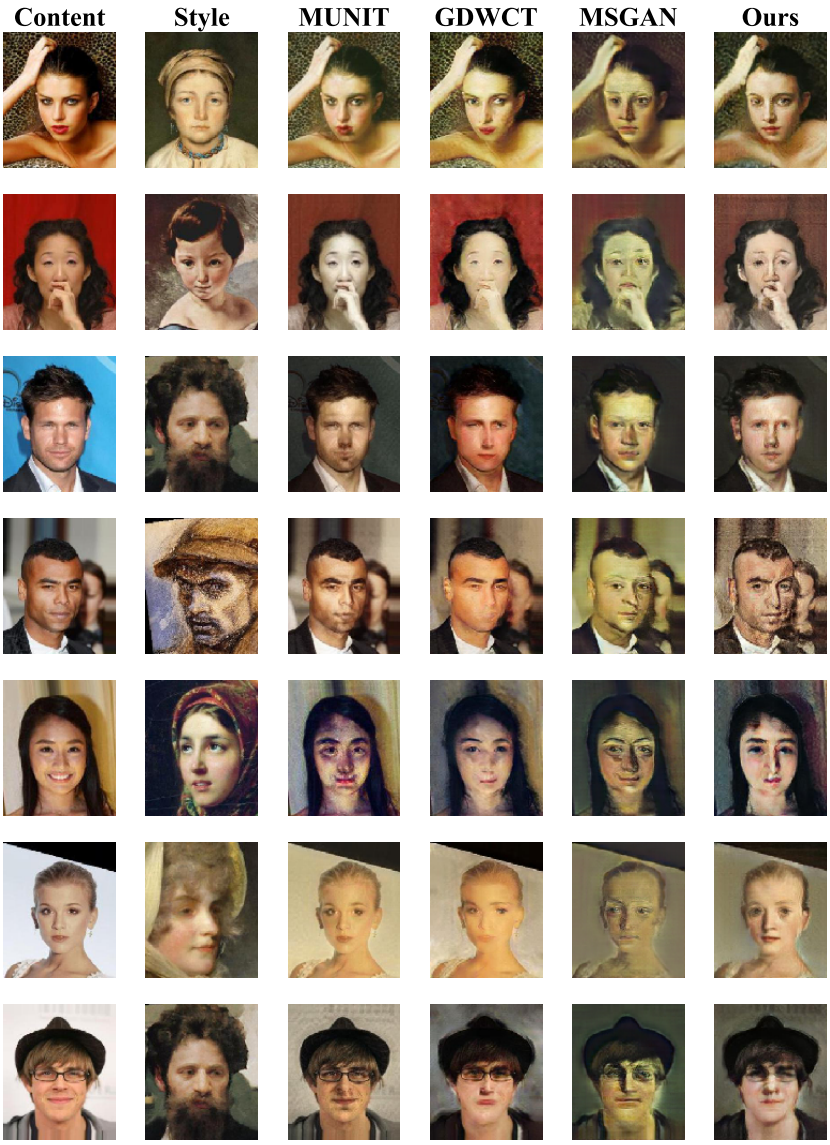


Fig. G.25: More compared results on `photograph→portrait`

Fig. G.26: More compared results on `photograph→portrait`

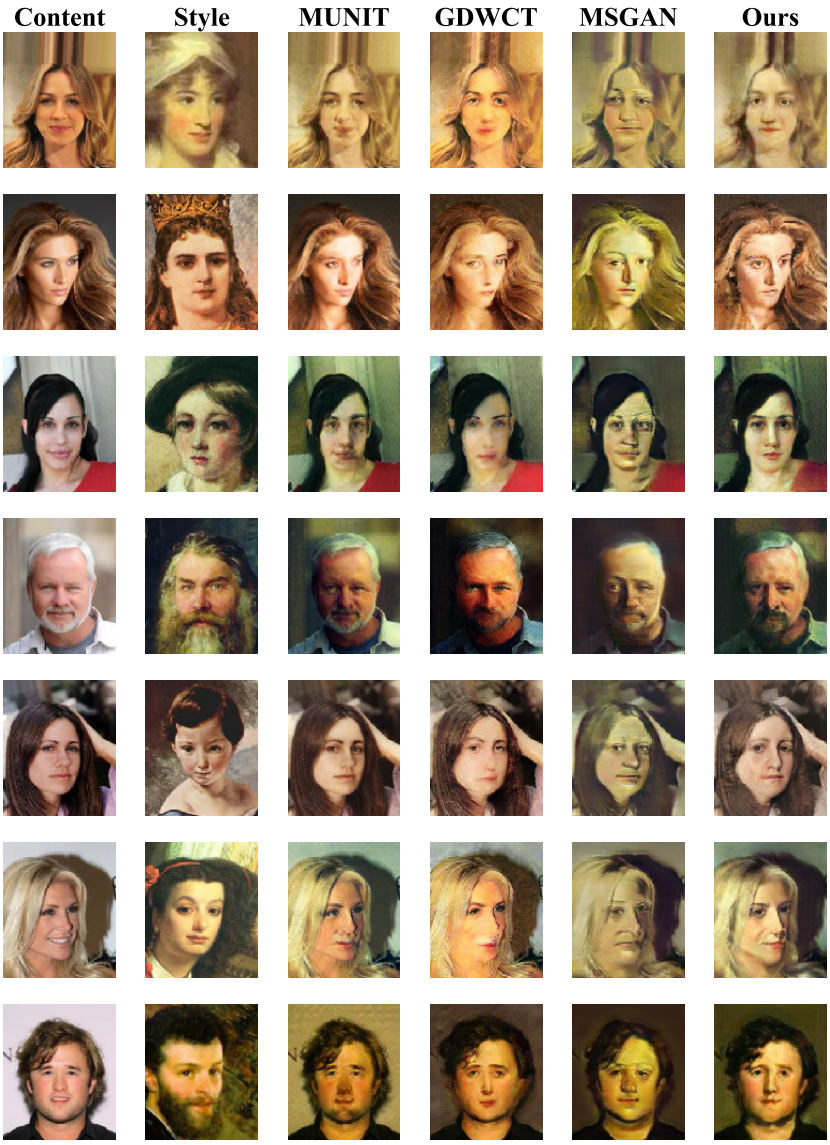


Fig. G.27: More compared results on `photograph→portrait`

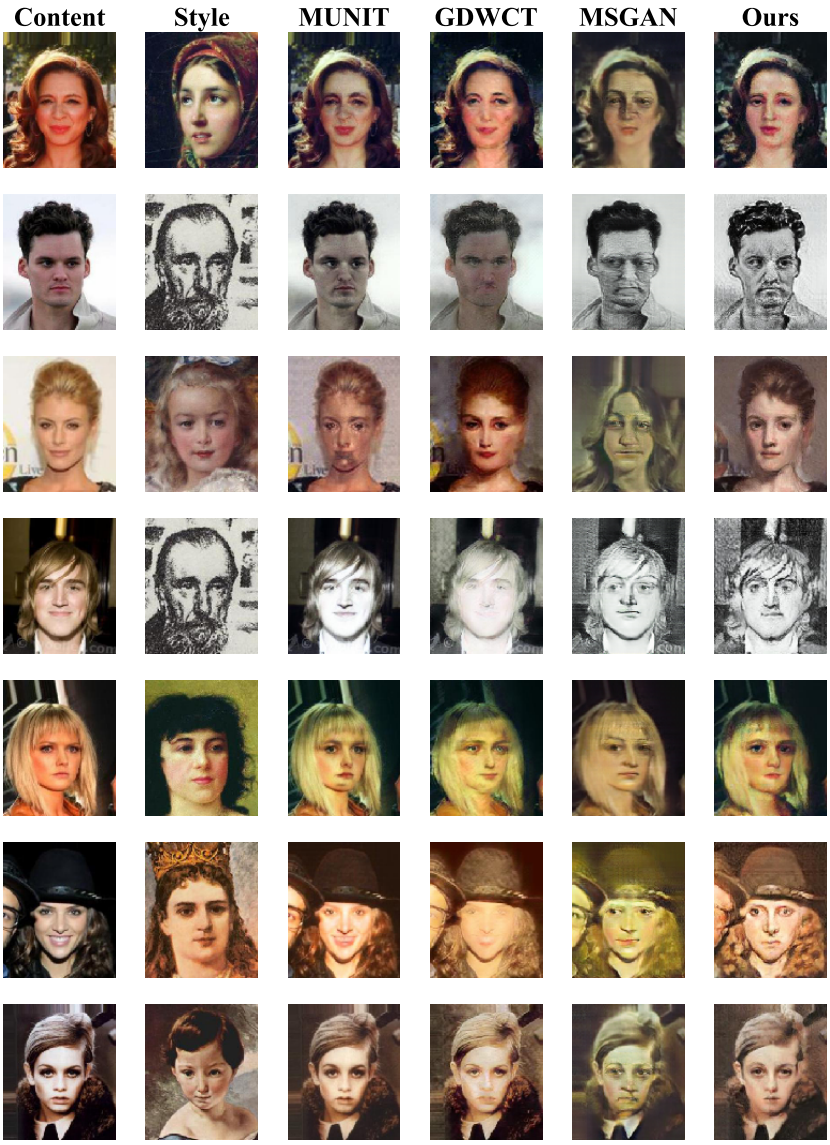


Fig. G.28: More compared results on `photograph→portrait`

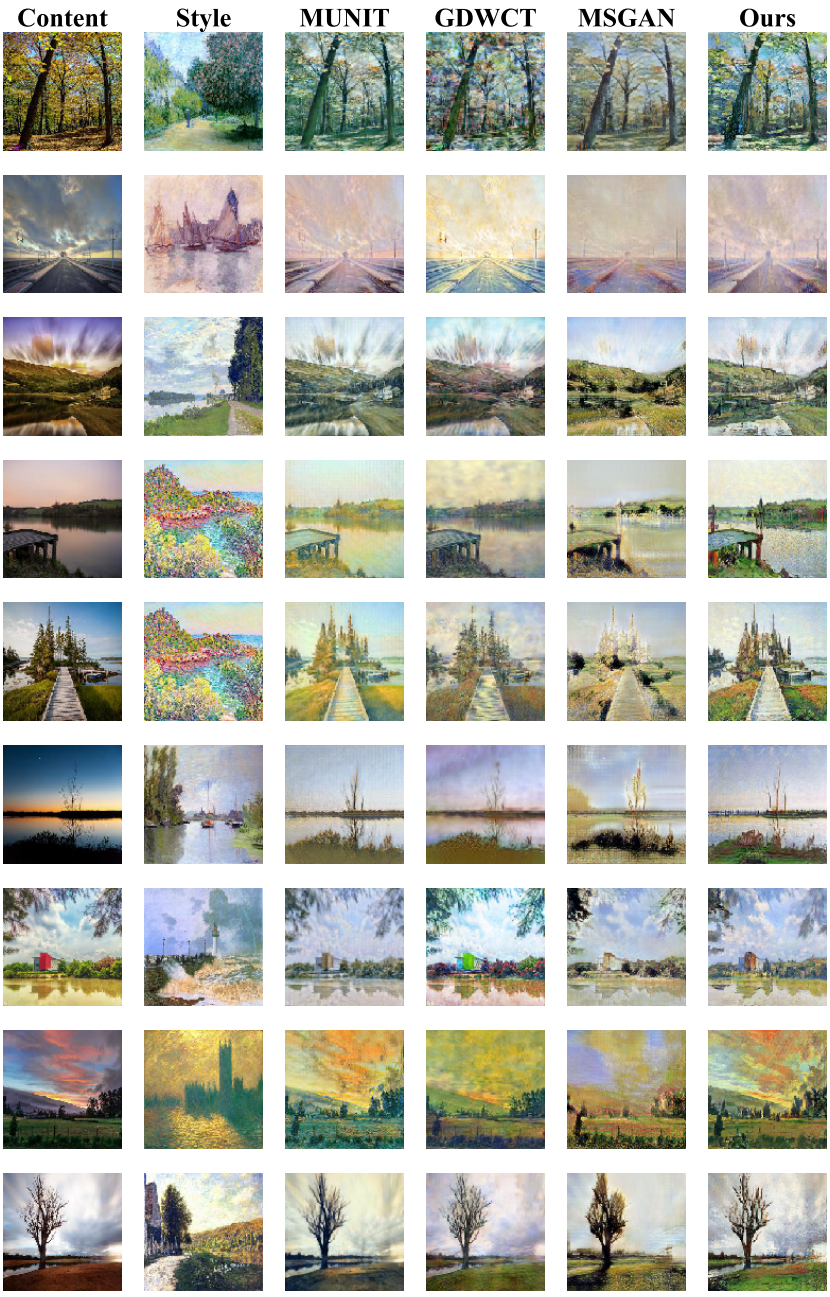


Fig. G.29: More compared results on `photo→Monet`

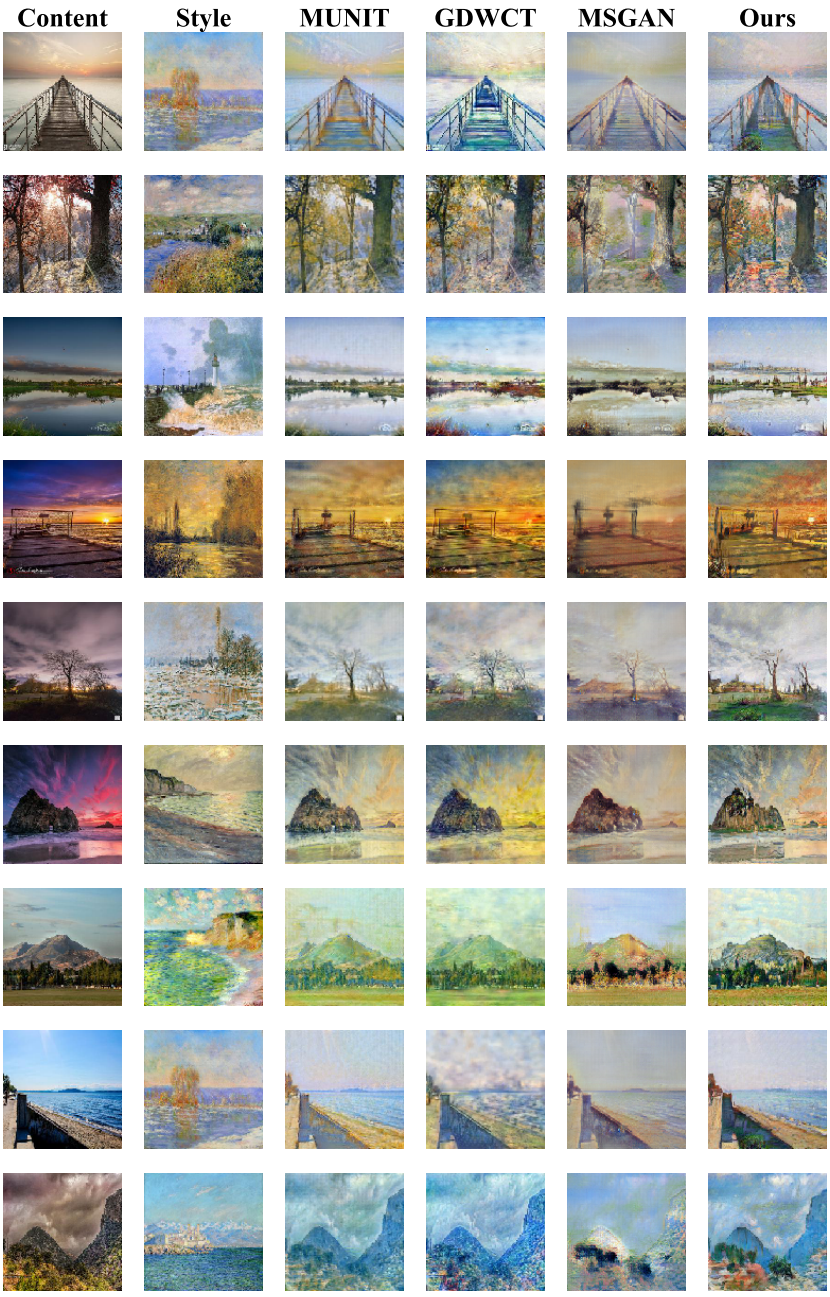


Fig. G.30: More compared results on `photo→Monet`

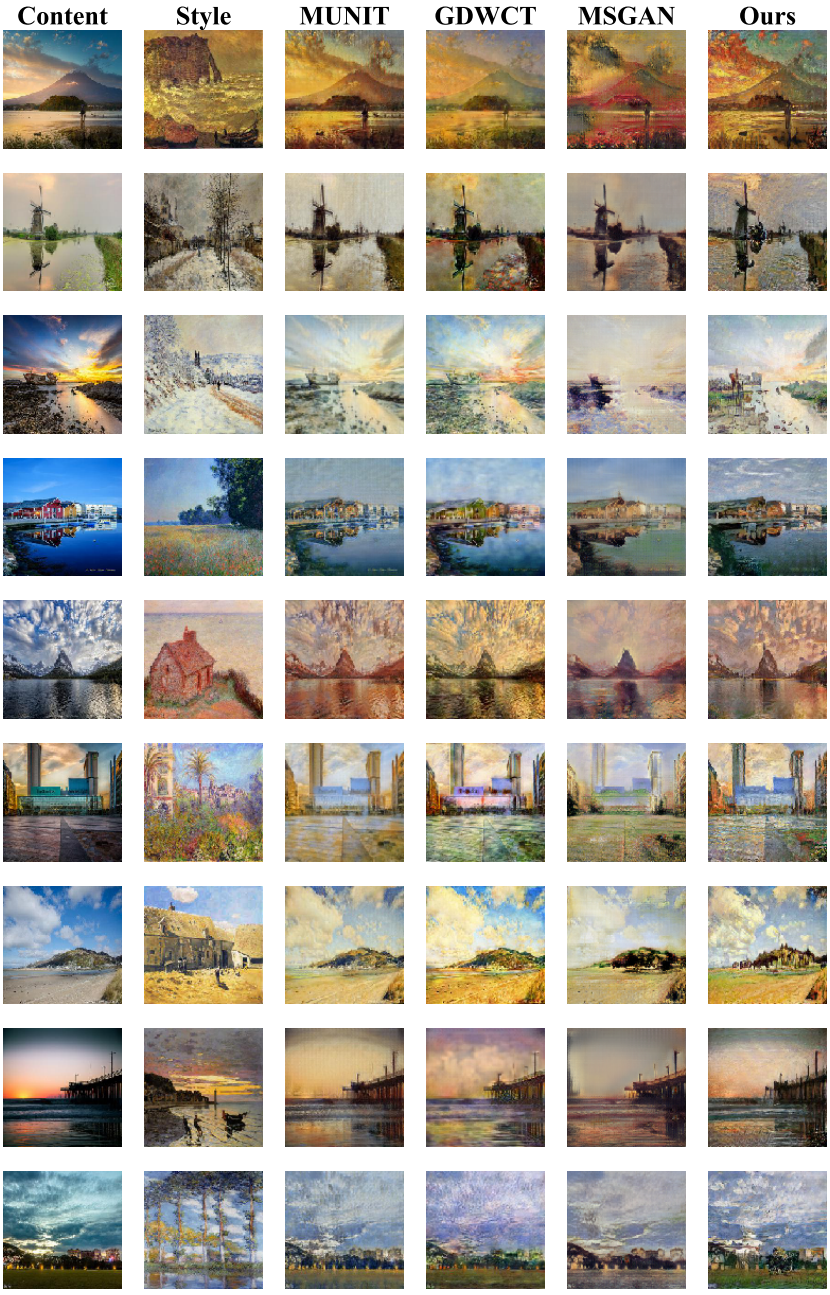


Fig. G.31: More compared results on `photo`→`Monet`

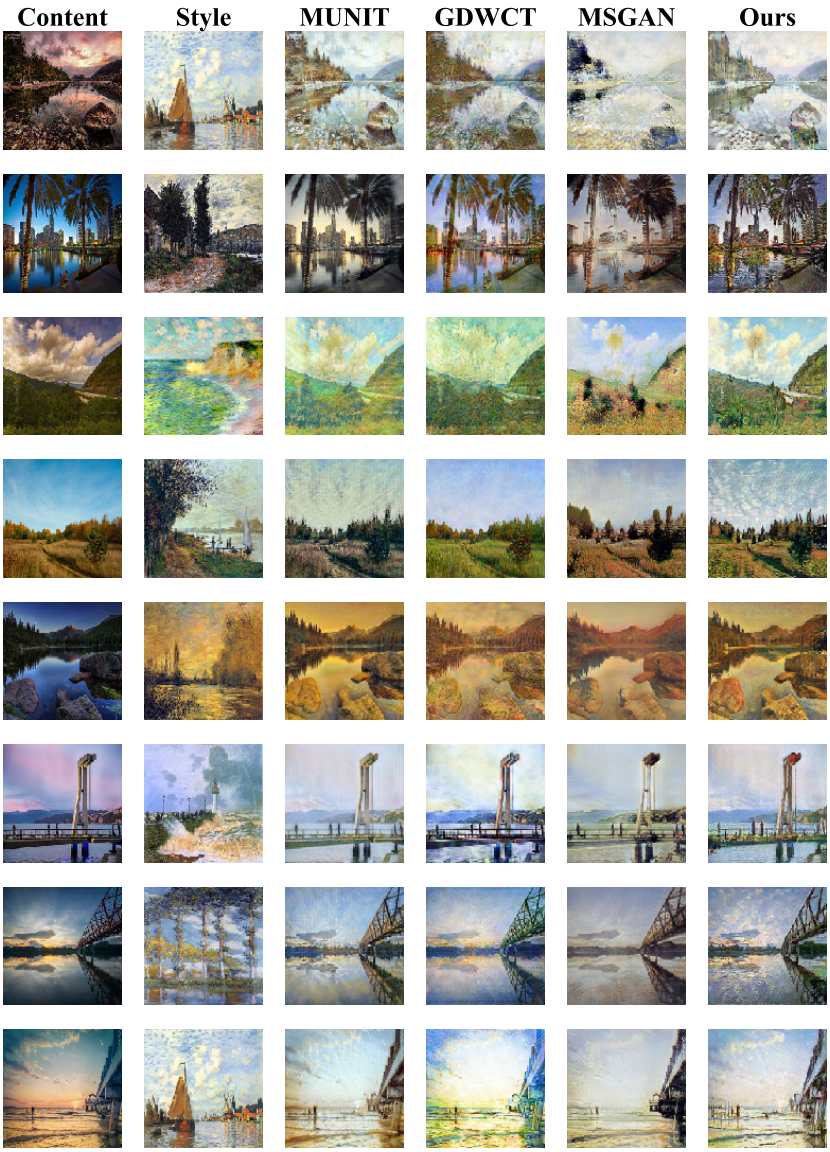


Fig. G.32: More compared results on `photo→Monet`