StarEnhancer: Learning Real-Time and Style-Aware Image Enhancement—Appendix

A. Network

In low-level vision tasks, an increasing number of networks (*e.g.* EDSR [12] and CBDNet [5]) remove the batch normalization (BN) layers [10] commonly used in high-level vision tasks. Meanwhile, the instance normalization (IN) layers [16] used in StarGAN [2, 3] are also rarely used in low-level vision tasks. Our experiments do show that employing the image classifiers' backbone networks with a large number of normalization layers as the style encoder and the curve encoder leads to poor performance. We suppose that employing too many normalization layers can impair the network's capability to extract the distribution of image colors and illumination, which is nevertheless crucial for the image enhancement task. To this end, we tend to remove all BN and IN layers in the network. Our proposed style encoder and curve encoder are both built on shallow ResNet [7]. However, if we use the Kaiming initialization [6] to initialize each layer of ResNet without normalization layers, it can lead to exploding gradients, so we modify the residual block as follows:

- Replace the ReLU activation function [14] with the PReLU activation function [6].
- PReLU is adopted before the convolution layer that refers to pre-activation [8].
- Add a scalar multiplier that is initialized at 1 before each element-wise addition layer.
- Add a scalar bias that is initialized at 0 before each convolutional layer, fully-connected (FC) layer, and PReLU activation layer.
- Initialize the 1st convolution layer of the residual block using the kaiming initialization, and scale the weight by $L^{-\frac{1}{2}}$ that refers to the Fixup initialization [18], where L is the total number of convolution layers.
- Initialize the 2nd convolution layer of the residual block and the FC layer to 0.

For the curve encoder in StarEnhancer, a Dual AdaIN is further inserted between the two convolution blocks in the residual block. Specifically, the style encoder extracts each style's latent code $\{\tilde{\mathbf{f}}_d\}$, and the mapping network then maps the latent code to style codes $\{\mu_{d,j}, \sigma_{d,j}\}$. Because the style encoder is inferred on the server, it can be deeper and wider than the curve encoder. The mapping network consists of two components: an FC Block further extracts the features from the latent codes; output branches map

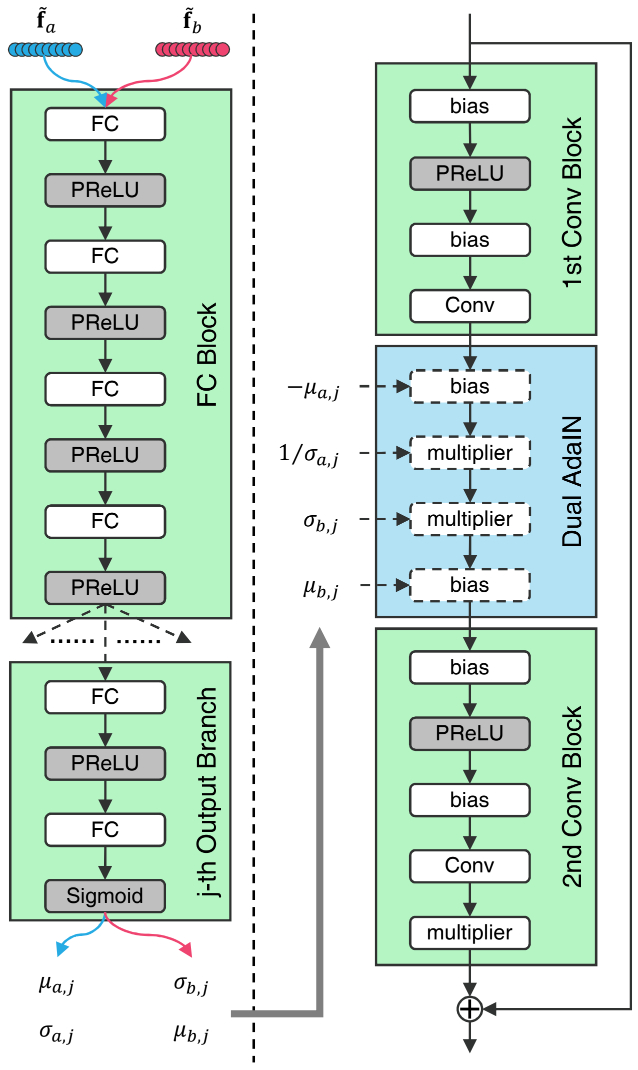


Figure 1: Details of the residual block with Dual AdaIN.

the features to the style codes. Note that the style codes are updated only when the source and target styles change. The number of output branches is equal to the number of Dual AdaINs inserted in the curve encoder, which means that each output branch corresponds to one Dual AdaIN. And the output dimension of the output branch is twice the channel number of the corresponding Dual AdaIN's feature maps so that the output vector can be split into $\mu_{d,j}$ and $\sigma_{d,j}$. The sigmoid function is used to ensure that the style code's elements are greater than zero, thereby avoiding $1/\sigma_{d,j}$ being unavailable. Finally, our proposed modified residual block is shown in Figure 1.

B. Curve-based color transformation

Using the residual blocks shown in Figure 1, we build the curve encoder to predict the curve knot points' parameter vectors. And the proposed curve-based transformation uses piecewise cubic interpolation [4] and indexing to utilize the curve knot points' parameter vectors.

We consider the curves that map the same color channel to be the most important, while the curves that map from pixel's coordinates only play a complementary role. Therefore, we assign more knot points to curves that map the same color channel to make them more expressive, and fewer knot points to curves that map from pixel's coordinates to make them smoother. Since the coordinates of the pixels vary monotonically, applying the transformation of the curves that map from pixel's coordinates does not require any indexing operation. Therefore, although the contribution of the curves that map from pixel's coordinates is not significant, it is cost-effective to keep them, considering its low computational complexity and capability to mimic tools such as gradient filters.

C. More implementation details

We use some tricks commonly used in face recognition to improve the performance of the style encoder. Firstly, We extend the Global Average Pooling (GAP) before the style encoder's fully connected layer to parallel GAP and Global Max Pooling (GMP) and sum the output tensor of GAP and GMP as the input of the following fully connected layer. Secondly, we fix the scaling term s (equivalent to the temperature scaling [9]) to 20 for refining the decision boundary when training the style encoder. Finally, we remove the bias of the style encoder's output layer and set its initial learning rate to 100 times that of the backbone network. The output layer's class weights are sometimes called proxies in metric learning [11, 13, 15, 17], but note that it is not equivalent to the centers of the classes.

Because the images in the MIT-Adobe-5K dataset [1] are high-resolution, the images' loading and pre-processing consume a lot of time (far more expensive than training). To this end, we train StarEnhancer on the low-resolution images and test the trained model on the full-resolution images. Note that we use random cropping to generate multiple low-resolution versions of the full-resolution image.

We can also add the single sample's latent code $\mathbf{f}_i$ as noise to train a more robust StarEnhancer:

$$\tilde{\mathbf{f}}' = \frac{(1-\alpha)\tilde{\mathbf{f}}+\alpha\mathbf{f}_i}{\left\|(1-\alpha)\tilde{\mathbf{f}}+\alpha\mathbf{f}_i\right\|_2}, \tag{1}$$

where α is a scalar that is sampled from the uniform distribution $\mathcal{U}$ on log space as follows:

$$\log(\alpha) \sim \mathcal{U}(\log(0.01), \log(1)). \tag{2}$$

D. Additional explanation

Q: Can the multi-curve enhancer be applied to other I2I translation tasks, such as image dehazing?
A: Unfortunately, it is not directly applicable to other I2I translation tasks but requires some simple modification. But we can take the encoder-decoder as the curve encoder to predict a low-resolution curve parameter map, consisting of a set of curve parameters (w/o H&W) for each cell (*i.e.*, one pixel of the parameter map). Guided by the input image, we can obtain the desired transformed image using the slice operator. In contrast to the color transformations proposed in the paper, the modified color transformation is locally smooth rather than globally shared.

Q: What is the main difference between the proposed Dual AdaIN and AdaIN?
A: The μ and σ in AdaIN are the mean and variance of the feature maps, while the two sets (source&target) of μ and σ in Dual AdaIN are all output by the mapping network. We note that the IN in AdaIN hurts our method's performance, which may be because IN makes feature maps lose important statistical properties (*e.g.*, brightness and contrast). In contrast, Dual AdaIN uses the style encoder to introduce style information and does not perform any normalization on the feature maps of the curve encoder.

Q: What is the meaning of the train set and test set for the unseen style?
A: We extend the FiveK dataset to 18 tonal styles and split the images of each style into train set and test set in the same manner. For 10 seen styles, we use the train set for training the model and collect the style codes. For 8 unseen styles, we only use the images in the train sets to collect style codes.

Q: Why use average embedding?
A: When inferring, the model cannot directly obtain the target style embedding corresponding to the input image because it can only be fetched by encoding the target style version of this image (*i.e.*, ground truth). Instead, we use the normalized average embedding to represent the target style embedding. In this way, we can use the style encoder to obtain the average embedding of all styles on the train set at once and then use the mapping network to get the corresponding μ and σ and save them. When inferring, we only need to load the saved μ and σ of the specified style without performing the style encoder and mapping network again.

E. More results

Figures 2 and 3 show some qualitative comparison results. Figures 4, 5, and 6 show some qualitative results of the transformations between multiple styles. Figures 7 and 8 show the transformations between seen styles or unseen styles. Figures 9 and 10 show some examples of manual fine-tuning.

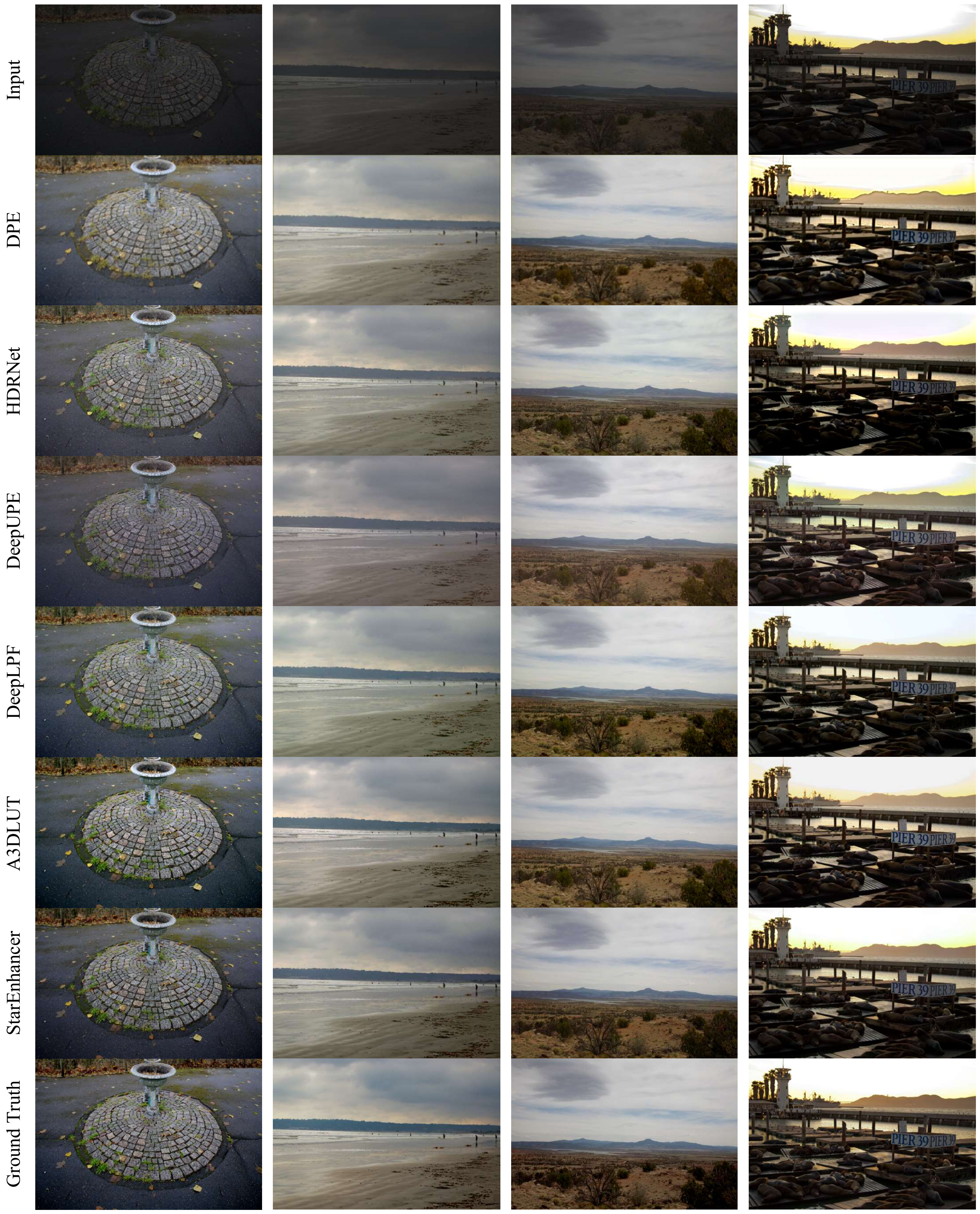


Figure 2: Qualitative comparison results of single style transformation with contemporaneous methods.

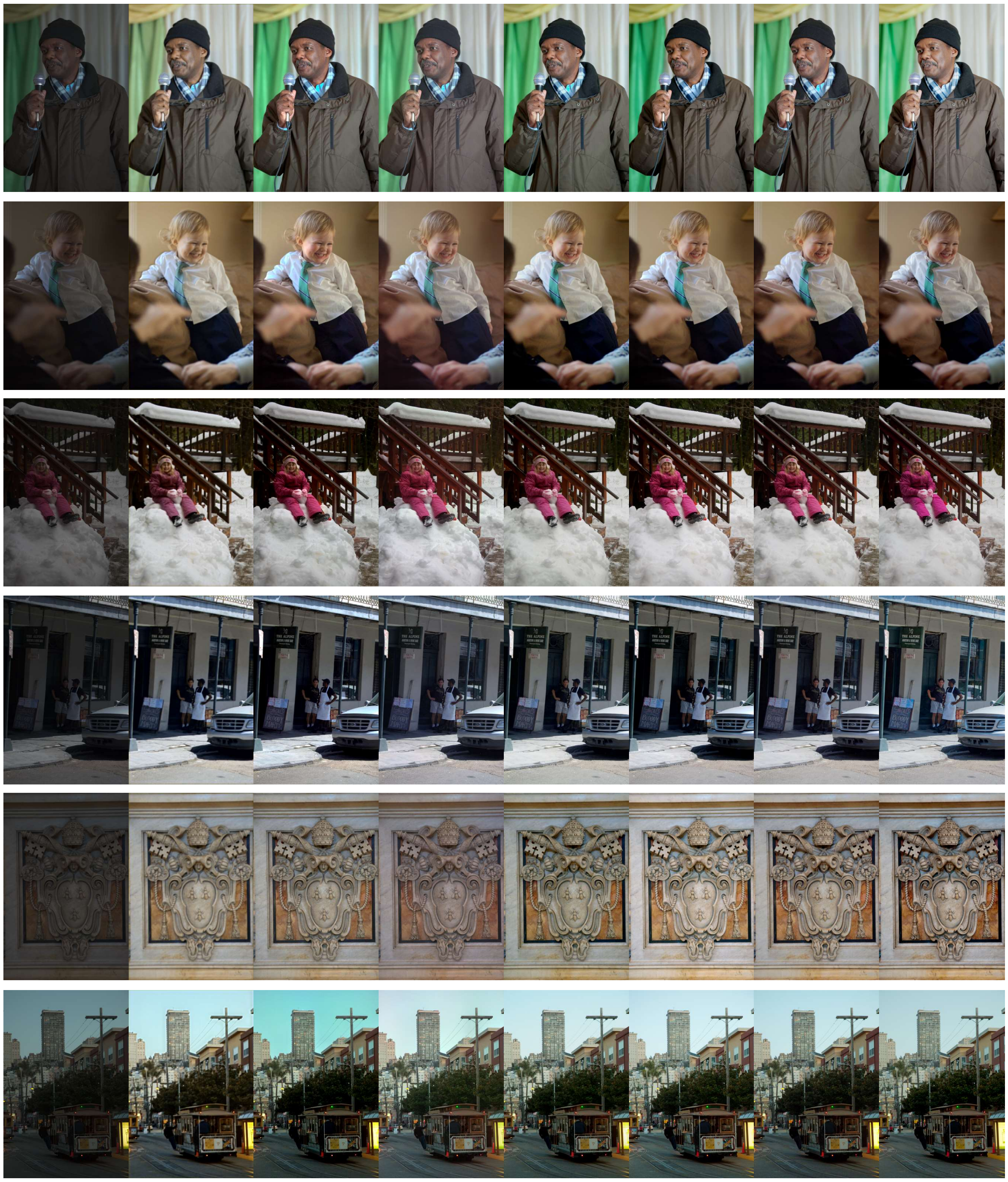


Figure 3: Qualitative comparison results of single style transformation with contemporaneous methods.

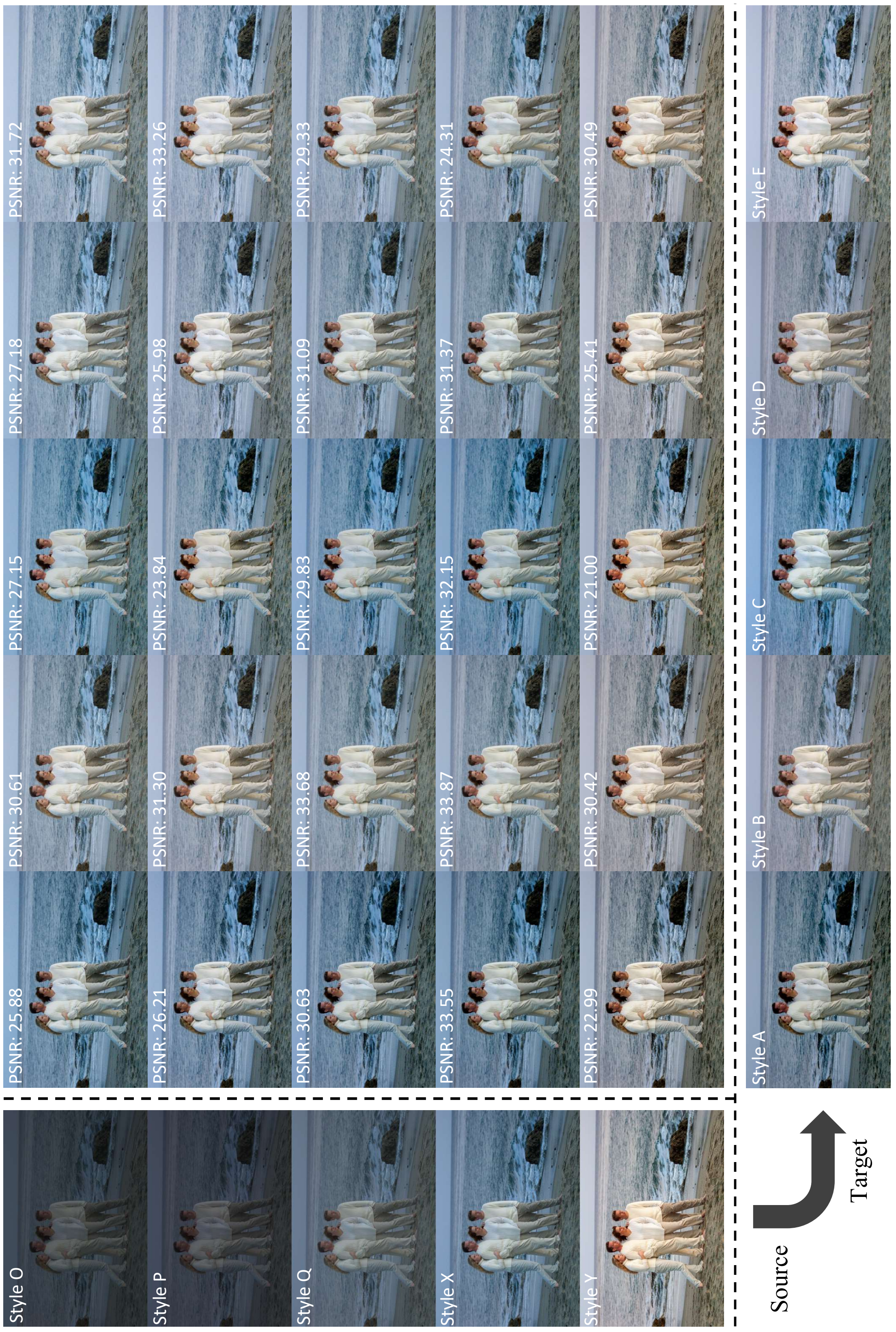


Figure 4: An example of mappings between multiple styles.

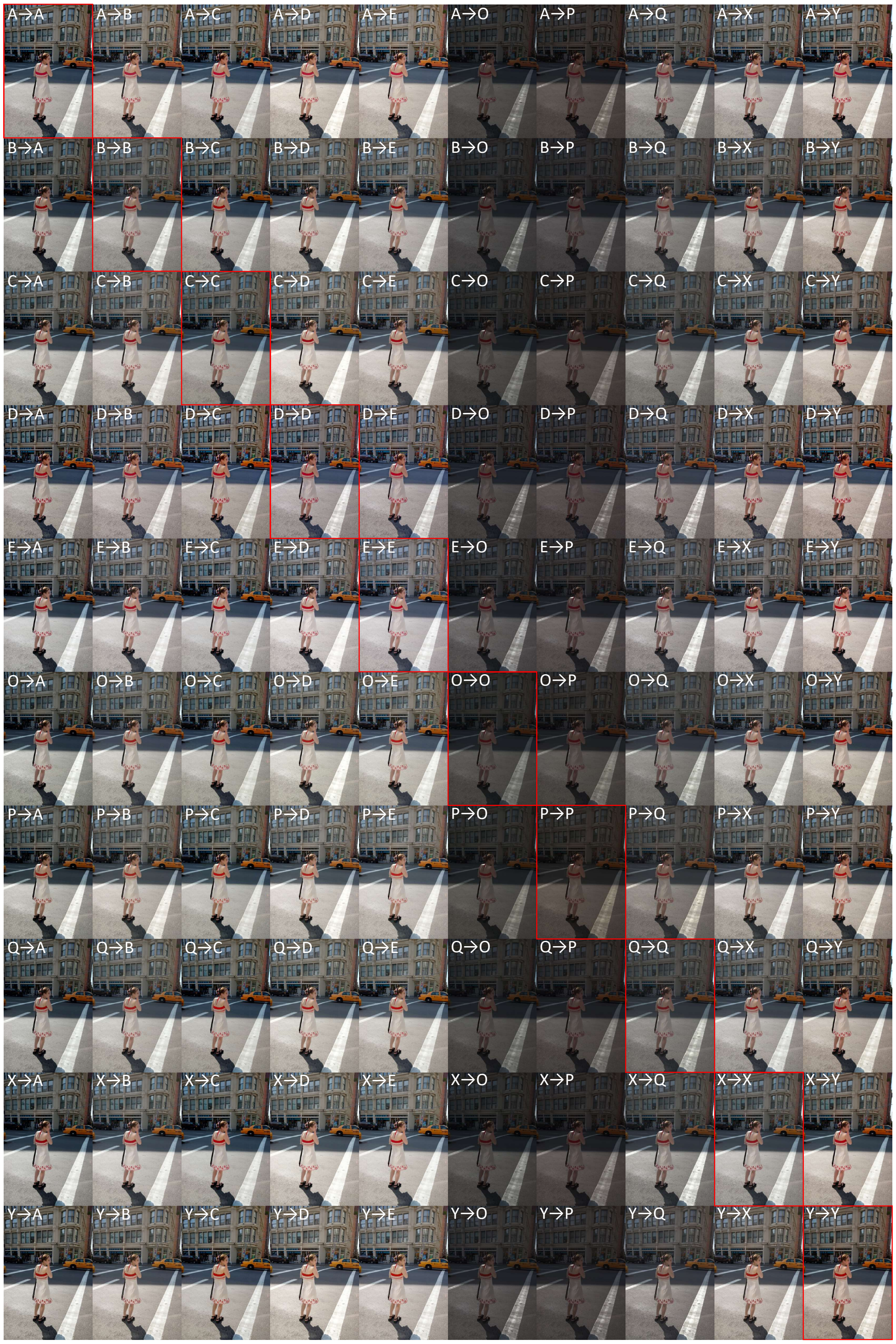


Figure 5: An example of mappings between all training styles. The images in the red boxes are ground truth.

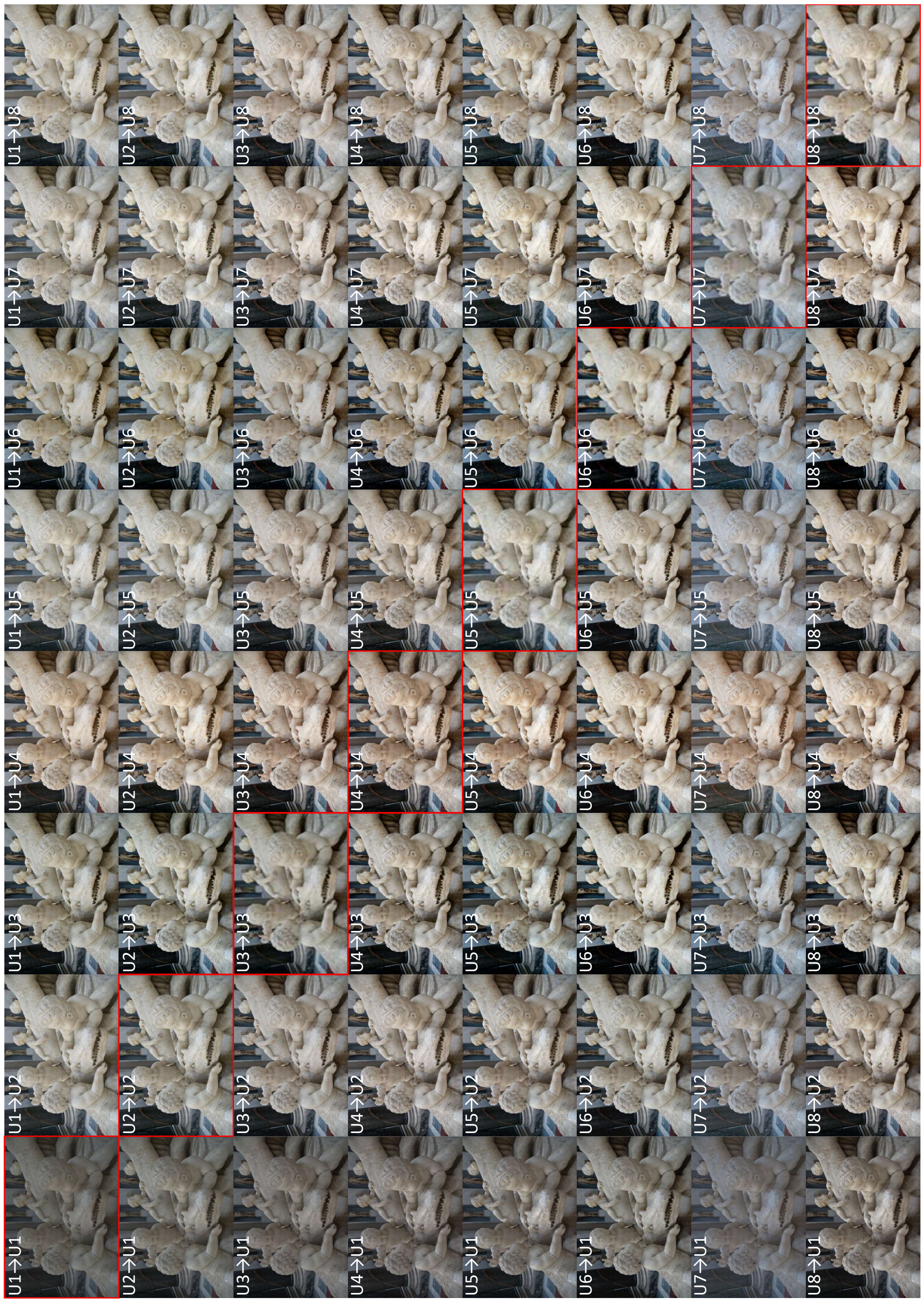
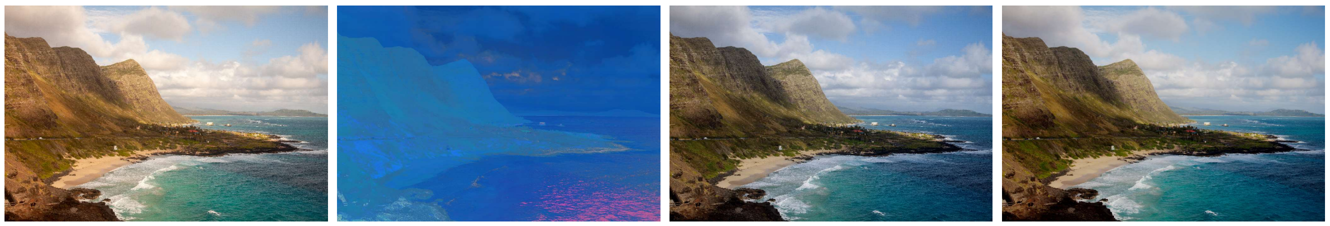


Figure 6: An example of mappings between multiple unseen styles. The images in the red boxes are ground truth.



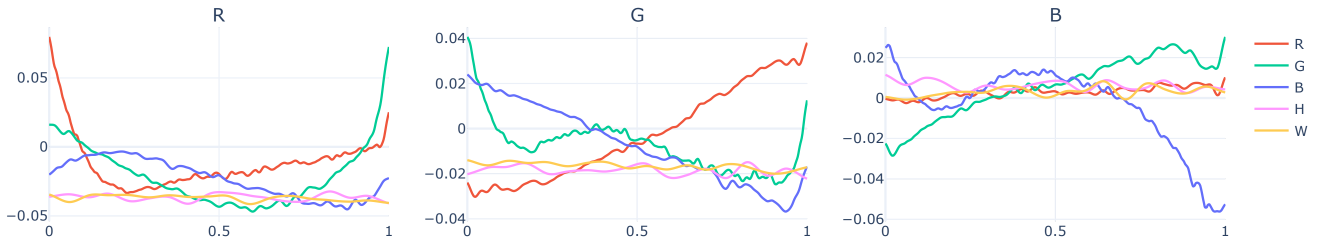
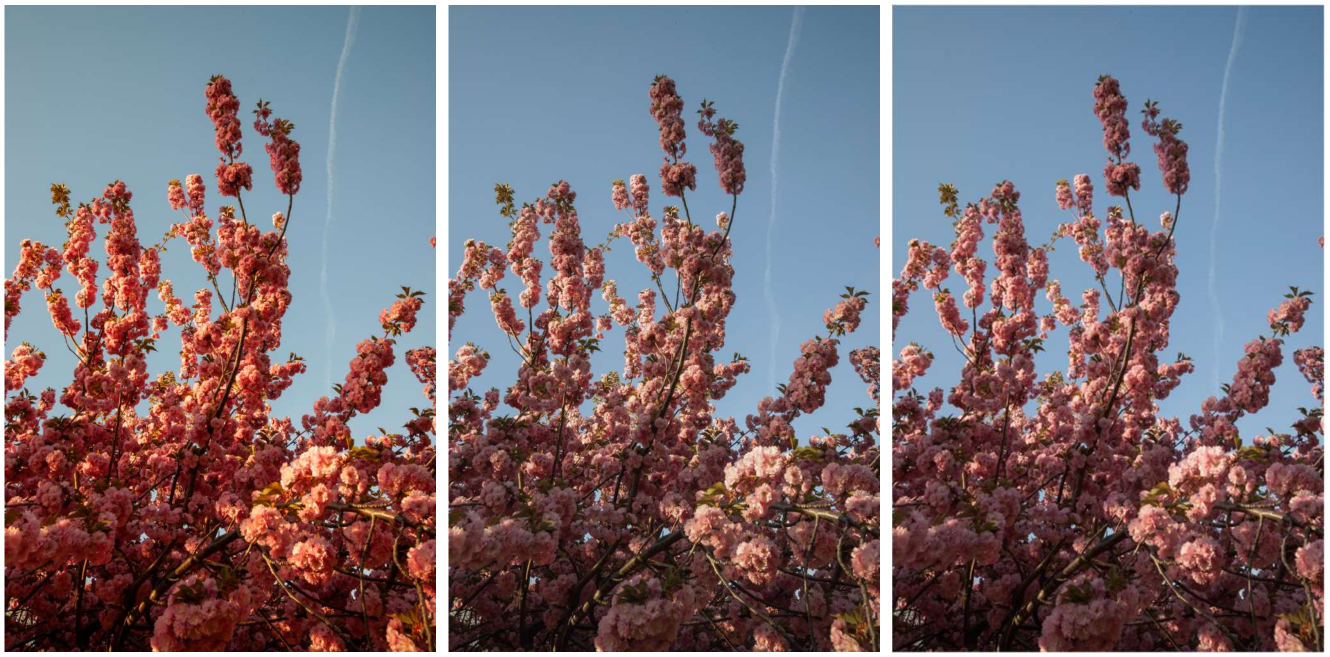


Figure 7: An example of the color transformation from style Y to style C. The normalized residual image corresponds to the bottom color transformation curves predicted by the network.



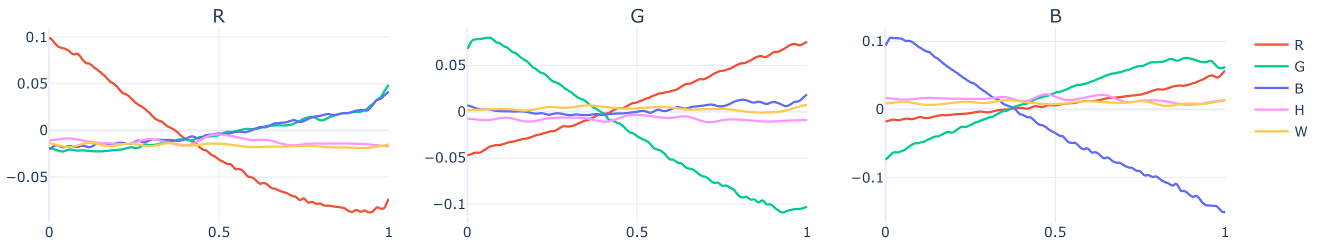
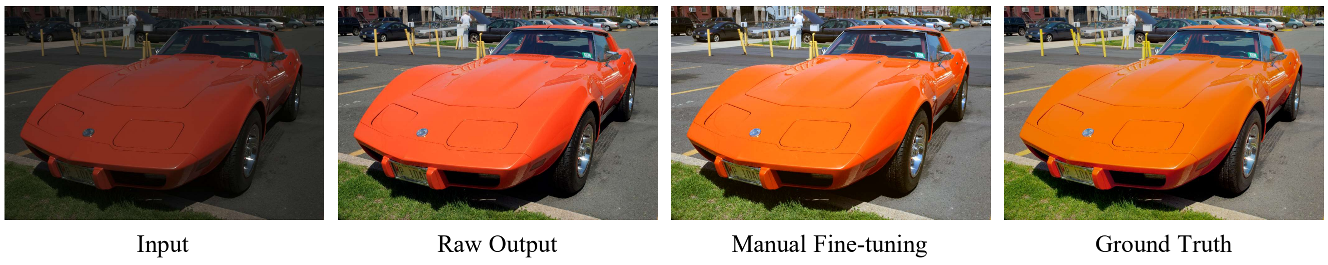


Figure 8: An example of the color transformation between two unseen styles.



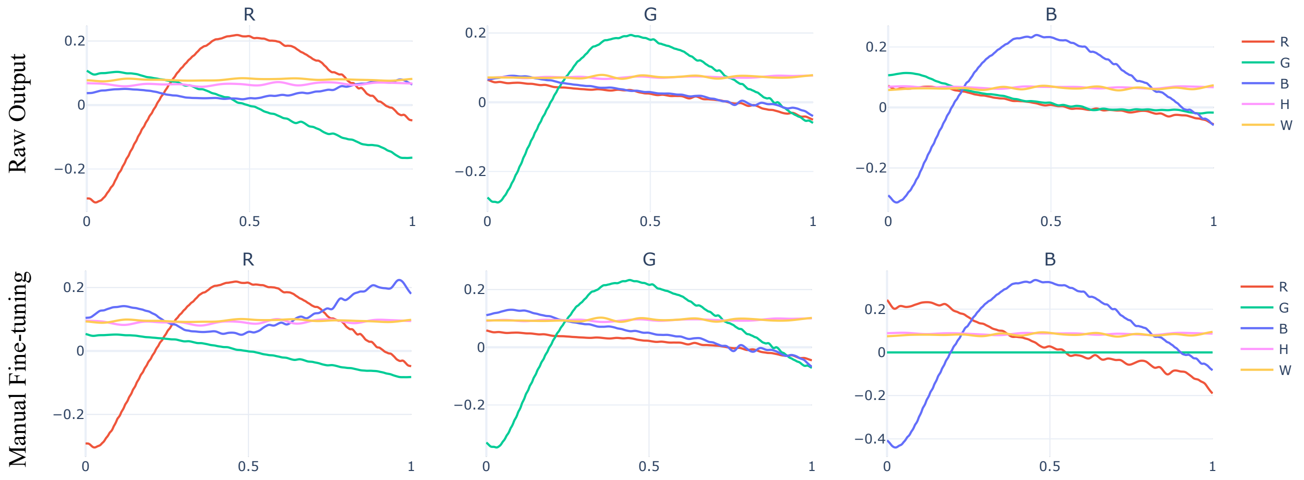
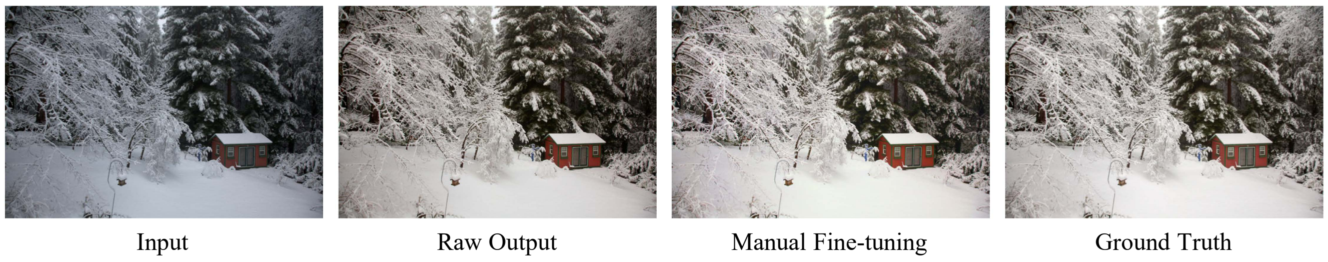


Figure 9: An example of manual fine-tuning from style O to style C.



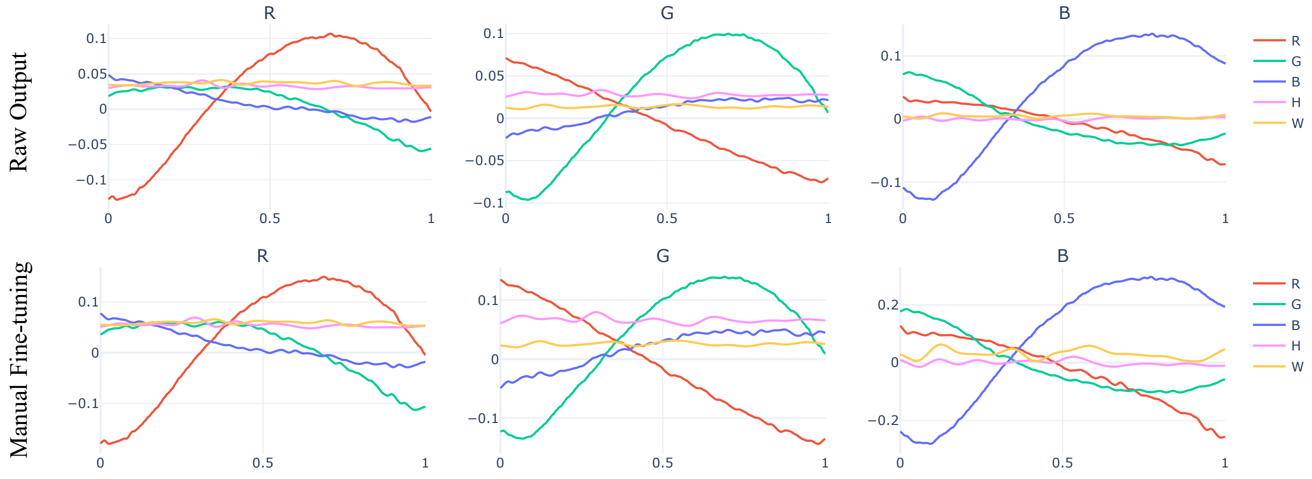


Figure 10: An example of manual fine-tuning between two unseen styles.

References

[1] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 97–104. IEEE, 2011.

[2] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8789–8797, 2018.

[3] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8188–8197, 2020.

[4] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[5] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1712–1722, 2019.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015.

[11] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3238–3247, 2020.

[12] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 136–144, 2017.

[13] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *International Conference on Computer Vision (ICCV)*, pages 360–368, 2017.

[14] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[15] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *International Conference on Computer Vision (ICCV)*, pages 6450–6458, 2019.

[16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[17] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. *British Machine Vision Conference (BMVC)*, 2018.

[18] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations (ICLR)*, 2018.