Neural Preset for Color Style Transfer - Supplemental

Zhanghan Ke[1] Yuhao Liu[1] Lei Zhu[1] Nanxuan Zhao[2] Rynson W.H. Lau[1]
[1]City University of Hong Kong [2]Adobe Research

Appendices

A. Optimization Problem of Neural Preset

Here we analyze (1) how to derive our training constraints from the fundamental color style transfer objective and (2) why performing color mapping via the proposed DNCM is necessary for our training strategy.

Consider the objective of color style transfer. Given an input image $\mathbf{I}_c$ and a style image $\mathbf{I}_s$ that have different content and color styles, we aim to learn a model H to transfer the color style of $\mathbf{I}_s$ to $\mathbf{I}_c$ by minimizing the objective:

$$\min\ \mathbb{E}_{\mathbf{I}_c,\mathbf{I}_s,\mathbf{G}_s\sim p_\mathbf{I}}\big[\,|\mathbf{G}_s - H(\mathbf{I}_c,\mathbf{I}_s)|\,\big], \tag{1}$$

where $p_\mathbf{I}$ represents the distribution of images, and $\mathbf{G}_s$ is the ground truth stylized image that has the same image content as $\mathbf{I}_c$ and the consistent color style as $\mathbf{I}_s$.

The idea of our two-stage pipeline is to divide H into two sub-functions, *i.e.*, two stages, to remove the original image color style of $\mathbf{I}_c$ before applying a new one, as:

$$\min\ \mathbb{E}_{\mathbf{I}_c,\mathbf{I}_s,\mathbf{G}_s\sim p_\mathbf{I}}\big[\,|\mathbf{G}_s - S_2(S_1(\mathbf{I}_c),\mathbf{I}_s)|\,\big], \tag{2}$$

where S_1 and S_2 denote the sub-functions corresponding to the two stages of our pipeline. As $\mathbf{G}_s$ is typically unavailable in practice, we generate pseudo input and style images to approximate the above optimization problem. Specifically, we add random color perturbations (*e.g.*, LUTs or filters) to each image $\mathbf{I}$ to create a set of n images $\{\mathbf{I}_1,\dots,\mathbf{I}_n\}$ with the same content but different color styles. We denote indexes by $i,j\in\{0,\dots,n\}$ in the following context. Thus, Eq. 2 can be approximated with perturbed images, as:

$$\min\ \mathbb{E}_{\mathbf{I}\sim p_\mathbf{I}}\Big[\sum_i^n\sum_j^n\Big|\mathbf{I}_j - S_2(S_1(\mathbf{I}_i),\mathbf{I}_j)\Big|\Big]. \tag{3}$$

Given any fixed S_1, the optimal S_2^* should satisfy:

$$\mathbf{I}_j = S_2^*(S_1(\mathbf{I}_j),\mathbf{I}_j) = S_2^*(S_1(\mathbf{I}_i),\mathbf{I}_j). \tag{4}$$

Eq. 4 reveals a possible optimization issue of Eq. 3 – if we model S_1 and S_2 by end-to-end CNNs like autoencoders [33], optimizing only Eq. 3 via gradient-based algorithms can easily make S_1 and S_2 converge to a trivial solution to satisfy Eq. 4: S_2^* becomes an identity function *w.r.t.* $\mathbf{I}_j$, while S_1 can be any function. Formally, for a real input and style image pair $\mathbf{I}_c$ and $\mathbf{I}_s$ with different content, the possible trivial solution is:

$$\mathbf{I}_s = S_2^*(\phi,\mathbf{I}_s),\quad \phi := S_1(\mathbf{I}_c), \tag{5}$$

where S_2^* always output $\mathbf{I}_s$ directly and ignore another input ϕ. Such a solution is undesired because the color style transfer objective (Eq. 1) requires the output image have the same content with $\mathbf{I}_c$.

To overcome the above problem, modeling S_1 and S_2 via DNCM instead of end-to-end CNNs is necessary. DNCM inputs the color style parameters $E(\tilde{\mathbf{I}}_j)$ rather than the image $\mathbf{I}_j$. Since the dimensions of $E(\tilde{\mathbf{I}}_j)$ is much lower than $\mathbf{I}_j$, it prevents S_2 from being an identity function *w.r.t.* $\mathbf{I}_j$ and forces S_1 to be involved in computing the optimal solution. Specifically, we define:

$$\begin{aligned} S_1(\mathbf{I}_i) &:= \textit{nDNCM}(\mathbf{I}_i, E(\tilde{\mathbf{I}}_i)),\\ S_2(S_1(\mathbf{I}_i),\mathbf{I}_j) &:= \textit{sDNCM}(S_1(\mathbf{I}_i), E(\tilde{\mathbf{I}}_j))). \end{aligned} \tag{6}$$

The formulas/symbols in Eq. 6 are equivalent to those in Sec. 3.2 of the paper. Benefited by DNCM, optimizing only Eq. 3 is sufficient for preventing S_1 and S_2 from converging to the trivial solution shown in Eq. 5. There are other possible approaches to avoid such a trivial solution without using DNCM, *e.g.*, modeling the optimization of our pipeline as a bi-level optimization [28, 29] problem.

Let us review Eq. 4 by substituting in Eq. 6, it is obvious that constraining $S_1(\mathbf{I}_i)$ and $S_1(\mathbf{I}_j)$ to be consistent will make S_2^* easier to obtain. Hence, we interpret Eq. 3 as a constrained optimization problem:

$$\begin{aligned} \min\ &\mathbb{E}_{\mathbf{I}\sim p_\mathbf{I}}\Big[\sum_i^n\sum_j^n\Big|\mathbf{I}_j - S_2(S_1(\mathbf{I}_i),\mathbf{I}_j)\Big|\Big]\\ s.t.\quad &\sum_i^n\sum_j^n\Big|S_1(\mathbf{I}_j) - S_1(\mathbf{I}_i)\Big| = 0. \end{aligned} \tag{7}$$

By using Penalty or Augmented Largangian [14] methods, Eq. 7 can be reformulated as an unconstrained optimization

problem. For example, the Penalty method produces:

$$\begin{aligned} &\min\ \mathbb{E}_{\mathbf{I}\sim p_{\mathbf{I}}}[\,\mathcal{L}_{rec} + \lambda\,\mathcal{L}_{con}\,], \\ &\mathcal{L}_{rec} := \sum_{i}^{n}\sum_{j}^{n}\Big|\mathbf{I}_j - S_2(S_1(\mathbf{I}_i), \mathbf{I}_j)\Big|, \\ &\mathcal{L}_{con} := \sum_{i}^{n}\sum_{j}^{n}\Big|S_1(\mathbf{I}_j) - S_1(\mathbf{I}_i)\Big|, \end{aligned} \tag{8}$$

where λ is a penalty coefficient. If we set $n = 2$, this new optimization objective is formally equivalent to our training constraint defined in Sec. 3.3 of the paper.

B. Details of Our Improved Quantitative Metrics

Below we describe the implementation details of our improved quantitative metrics for color style transfer.

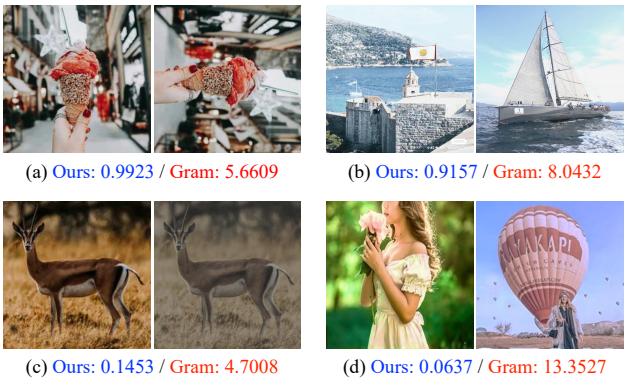
Style Similarity Metric. We first build a dataset consists of 700+ color style categories, each containing 6-10 images with the same color style retouched by human experts (see Fig. 1). We then train a discriminator model D on this dataset as our style similarity metric. Specifically, for an image $\mathbf{I}$ in the dataset, we use $\mathbf{I}_p$ and $\mathbf{I}_n$ to denote a positive sample from the same category as $\mathbf{I}$ and a negative sample from any other category, respectively. We optimize D to distinguish different color styles via minimizing the following loss from LS-GAN [31]:

$$\mathcal{L}_{dis} = ||D(\mathbf{I},\,\mathbf{I}_p) - 1||_2 + ||D(\mathbf{I},\,\mathbf{I}_n) - 0||_2. \tag{9}$$

The trained D will output a score between $[0, 1]$, which represents the style similarity between two input images. The score tends to be 1 if the two input images have similar color styles and 0 otherwise. The architecture of D is adapted from [18]. We use the Adam [20] optimizer to train D for 120 epochs. The initial learning rate is $1e^{-4}$ and is multiplied by 0.1 after every 50 epochs.

Our metric, *i.e.*, the trained D focuses more on comparing image color styles and ignores other irrelevant information, such as the photorealism of the image content. Besides, D predicts the style similarity score based on not only on the statistic of color values but also on image properties, *e.g.*, whether the two images have the same global contrast. To demonstrate that our newly proposed metric is more meaningful than the Gram metric used by prior works, we compare the style similarity predicted by our metric and the previously used Gram metric in Fig. 2. Note that a lower Gram value means a higher similarity. Gram < 5 usually indicates very similar, while Gram > 8 indicates a huge difference. We can see that both metrics work well if two images have exactly the same (Fig. 2 (a)) or very different (Fig. 2 (d)) content and color style. However, for two images with the same content but different color styles, the Gram metric may consider they have similar color styles (Fig. 2 (c)). Meanwhile, for two images with different content but similar color styles, the Gram metric may consider they are different in color style (Fig. 2 (b)). Instead, our metric gives reasonable results in these cases.

Figure 1. **Data Used to Train Our Color Style Discriminator.** We display some samples from the dataset. The four samples in each column belong to the same color style category.



(a) Ours: 0.9923 / Gram: 5.6609 (b) Ours: 0.9157 / Gram: 8.0432

(c) Ours: 0.1453 / Gram: 4.7008 (d) Ours: 0.0637 / Gram: 13.3527

Figure 2. **Predicted Style Similarity.** The blue number below each image pair is the style similarity (between $[0, 1]$) predicted by our metric (*i.e.*, the color style discriminator), and a higher value means that the two images have more similar color styles. The red number below each image pair is the style similarity (larger than 0) predicted by the Gram metric, and a lower value means that the two images have more similar color styles. We compare two metrics in four cases: (a) two images with the same content and color style; (b) two images with different content but similar color styles; (c) two images with the same content but different color styles; (d) two image with different content and color styles.

Content Similarity Metric. We test the edge detection method HED [40] used by prior works [1, 26, 42], and we observe that HED fails to detect fine edges and often predict inaccurate edges, leading to an unreliable content similarity evaluation. To alleviate this problem, we suggest replacing HED with a state-of-the-art edge detection method LDC [34] to provide more precise edges. Fig. 3 compares the visual results of HED and LDC, which shows the advantages of LDC. When computing the metric, we set the long side of the HED/LDC input images to 2048 to preserve im-

Figure 3. **Predicted Edges for Content Similarity Calculation.** For a more precise quantitative evaluation of content similarity, we replace HED [40] (used by prior works [1,26,42]) with LDC [34]. LDC can provide finer (see blue arrows) and more accurate edges (see red arrows) than HED.

age textures. After extracting edges, we compute the SSIM metric between them as the content similarity score of the two input images.

C. More Results

C.1 Visual Results of Neural Preset

We provide more visual results in Fig. 6 and Fig. 8.

C.2 Video Stylization Results of Neural Preset

We show frames of video results in Fig. 4 and provide videos in our project page. By creating *nDNCM/sDNCM* from the first frame and using them to process subsequent frames, Neural Preset can provide consistent results across frames. In contrast, prior methods often cause flickering artifacts and post-processing like DVP [22] should be applied.

C.3 Stylize Various Images with the Same Color Style

Fig. 5 shows the results of applying the same color style to different images through Neural Preset.

C.4 Comparison on User Study Results

In Fig. 7, we calculate the ratios of each method being ranked as 1st Best, 2nd Best, and 3rd Best. Neural Preset is selected as 1st Best (*i.e.*, Top1) in 61.28% of cases, significantly surpassing 16.25% obtained by the second-ranked WCT^2. In addition, Neural Preset achieves a Top3 (*i.e.*, (1st + 2nd + 3rd) Best) ratio of 93.02%.

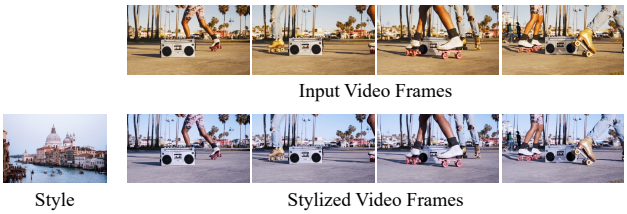


Figure 4. **Our Video Color Stylization Results.** Neural Preset provides consistent color style transfer results across video frames.

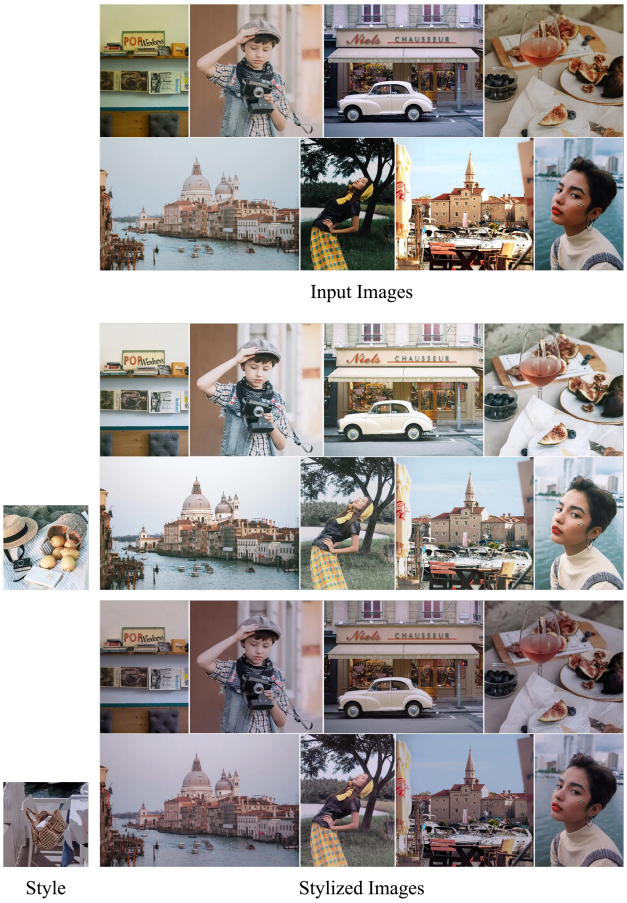


Figure 5. **Our Image Stylization Results.** Neural Preset can convert images with diverse color styles to the same color style.

C.5 Comparison on Applied to Other Tasks

In Fig. 9, we provide visual results of our Neural Preset (without fine-tuning) and other color style transfer methods on low-light image enhancement [24], underwater image color correction [41], image dehazing [11], and image harmonization [32]. Neural Preset outperforms other methods by a large margin. However, since our model is only trained in a self-supervised manner and not fine-tuned on task-specific datasets, it may fail on these tasks, as shown and discussed in Fig. 10.

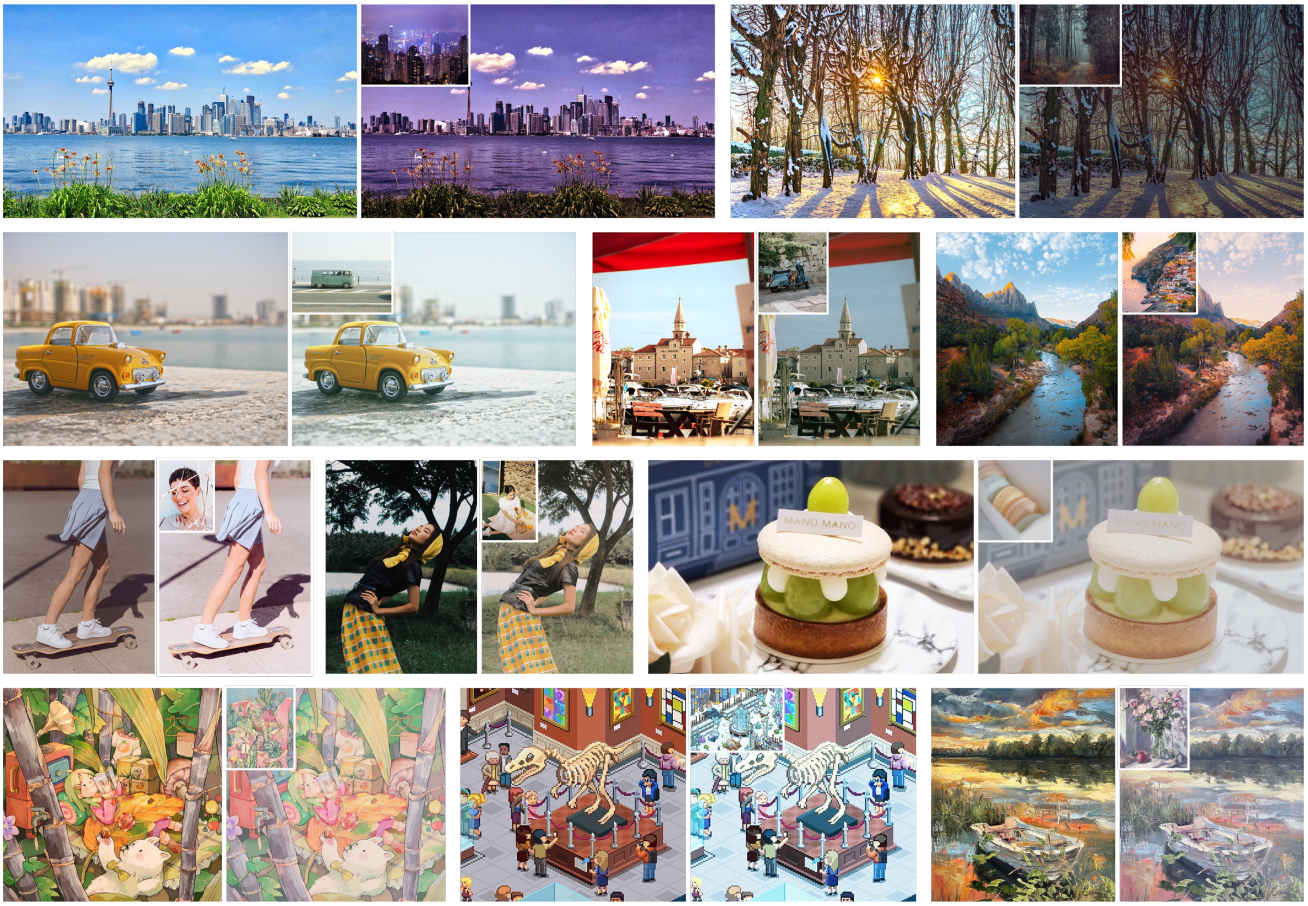


Figure 6. **Image Color Style Transfer Results of Neural Preset.** For each image pair, the left is the input image, while the right is our stylized result. The reference style image is displayed in the top-left corner. Our method is robust when generalizing to different types of images, *e.g.*, illustrations, pixelated images, oil paintings (see the last row).

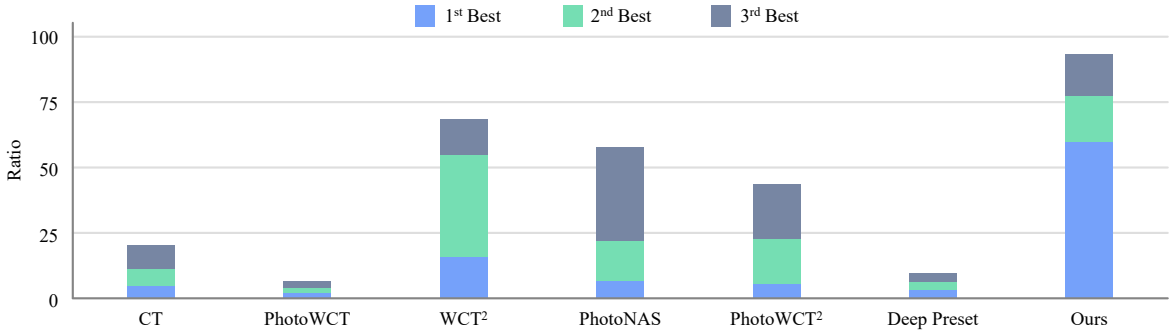


Figure 7. **Comparison on User Study Results.** We display the ratios of each method being ranked as 1st, 2nd and 3rd Best. Our Neural Preset is ranked as the Top1, Top2, and Top3 results over 61%, 78%, and 93% cases, respectively.

C.6 Comparison on CPU Inference Time

Table 1 shows that Neural Preset is much faster on CPU. Remarkably, it takes only 0.686 seconds to process a 4K image, but recent state-of-the-art methods either have the out-of-memory issue or take about 1 minute for processing.

D. DNCM for Image Harmonization/Enhancement

Here we describe how to train DNCM with pairwise data for image harmonization and image color enhancement.

DNCM for Image Harmonization. Extracting the fore-

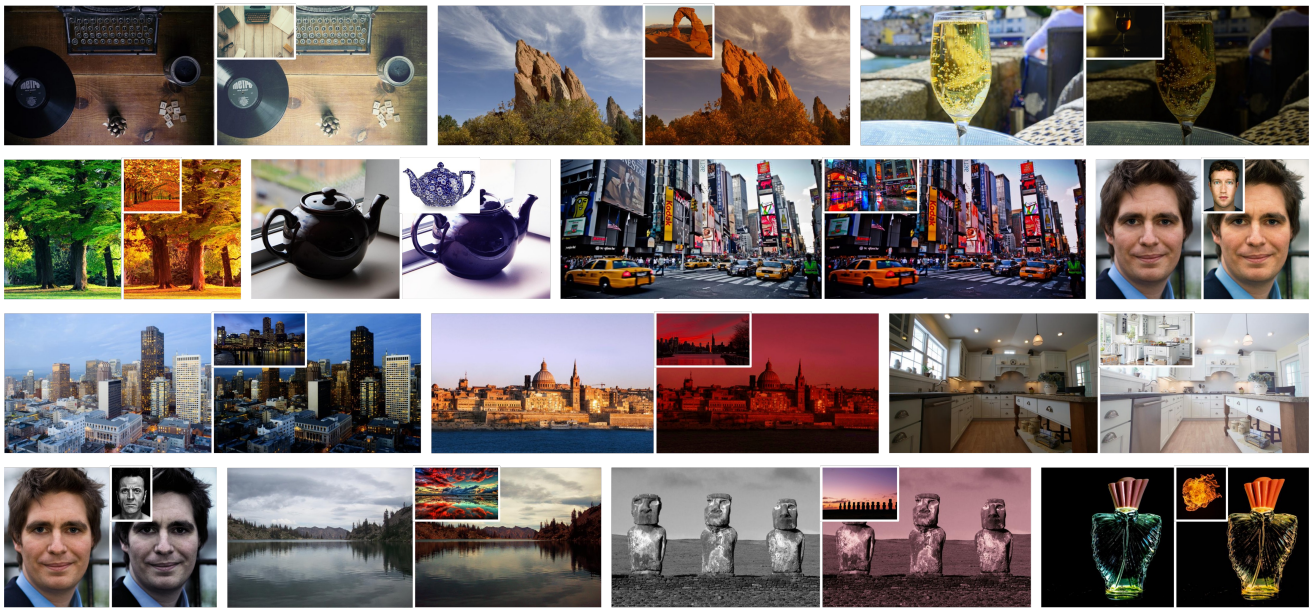


Figure 8. **Image Color Style Transfer Results of Neural Preset.** For each image pair, the left is the input image, while the right is our stylized result. The reference style image is displayed in the top-left corner. All samples we show here are from the test set provided by Luan *et al.* [30]. Neural Preset works well on most cases (see the first three rows), except for cases similar to the ones we have discussed in the limitations (see the last row).

Method	CPU Inference Time ↓			
	FHD (1920 × 1080)	2K (2560 × 1440)	4K (3840 × 2160)	8K (7680 × 4320)
PhotoWCT [26]	14.591 s	25.686 s	OOM	OOM
WCT^2 [42]	24.204 s	42.669 s	OOM	OOM
PhotoNAS [1]	14.227 s	24.826 s	OOM	OOM
Deep Preset [15]	14.354 s	25.173 s	58.030 s	OOM
$PhotoWCT^2$ [3]	3.111 s	4.588 s	OOM	OOM
Ours	**0.215 s**	**0.346 s**	**0.686 s**	**2.290 s**

Table 1. **Comparison on CPU Inference Time.** Evaluations are conducted on an Intel i9-11900KF CPU with 32GB PC memory. All models are in Float32 precision. "OOM" means having the out-of-memory issue.

ground from one image and compositing it onto a background image is a common operation in image editing. In order to make the composite image more realistic, the image harmonization task is introduced to remove the inconsistent appearances between the foreground and the background. Recently, many image harmonization methods [4–7,12,13,19,27,36] based on deep learning have been proposed with notable successes.

Since image harmonization can be regarded as a color mapping process from the background to the foreground inside an image, we attempt to solve it using the proposed DNCM. We adapt DNCM to image harmonization with two modifications. First, we downsample the composite image $\mathbf{I}$ and the foreground mask $\mathbf{M}$ to obtain thumbnails $\tilde{\mathbf{I}}$ and $\tilde{\mathbf{M}}$, which are concatenated as the input of the encoder E. Second, we only use DNCM to alter the color of foreground pixels (marked by $\mathbf{M}$) in $\mathbf{I}$, *i.e.*, all background pixels in $\mathbf{I}$ are not changed.

We follow existing works to conduct experiments on the iHarmony4 [6] benchmark. We evaluate the image harmonization performance by MSE and PSNR. The encoder E is set to EfficientNet-B0 [35], and the DNCM hyperparameter k is set to 16. With the training loss from Tsai *et al.* [36], our model is optimized by the Adam [20] optimizer for 50 epochs. We set the learning rate to $3e^{-4}$ (with a batch size of 16) and multiply it by 0.1 after every 20 epochs. Table 2 compares our model with state-of-the-art image harmonization methods. Without specific modules/constraints designed for the image harmonization task, our model achieves top-level performance in terms of MSE

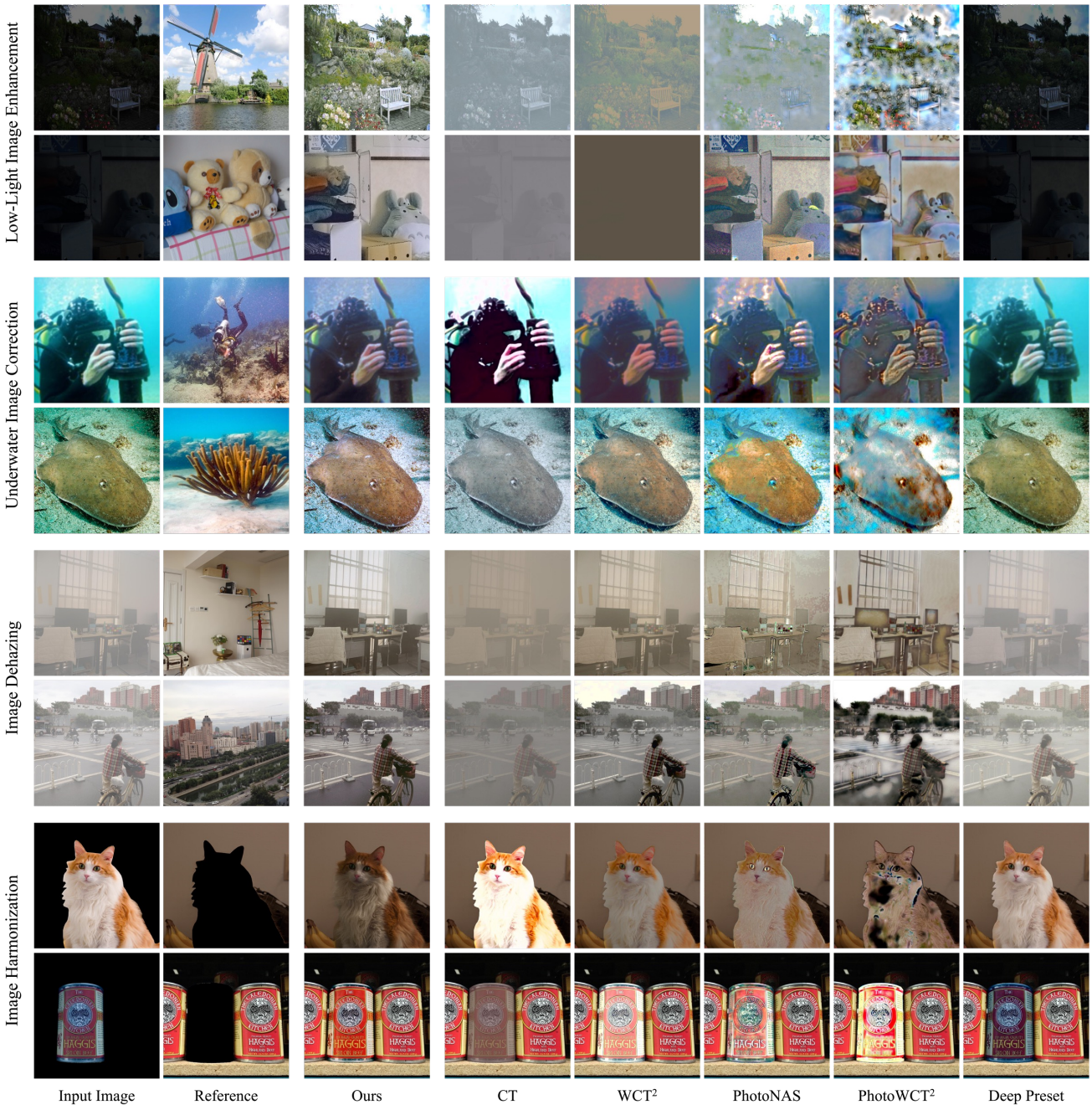
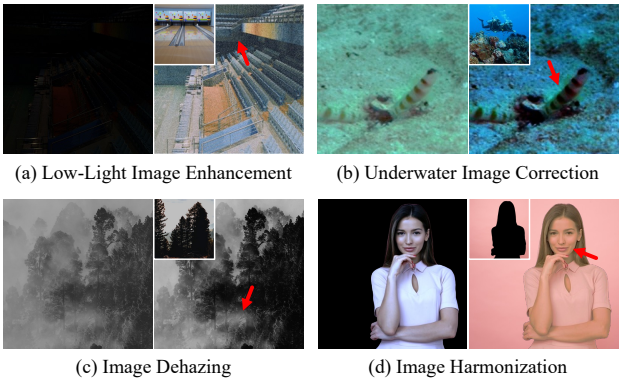


Figure 9. **Applying Color Style Transfer Methods to Other Tasks without Fine-tuning.** Our Neural Preset robustly generalizes to other color mapping tasks and surpasses previous color style transfer methods by a large margin. The datasets we used are listed in the Acknowledgments at the end of the paper.

and PSNR. Notably, our model outperforms other methods in terms of inference time and memory footprint.

DNCM for Image Color Enhancement. Image color enhancement aims to improve the visual quality of images captured in different scenes, such as underexposed or overexposed scenes. Recently, deep learning based methods [10, 16, 37, 38, 43] have dominated this field. Since we can formulate image color enhancement as a many-to-one color mapping from diverse degraded domains (*e.g.*, underexposed and overexposed domains) to an enhanced domain, we experiment with applying DNCM to this task.

We set the DNCM hyper-parameter k to 24. We train DNCM for 200 epochs using the Adam [20] optimizer



(a) Low-Light Image Enhancement (b) Underwater Image Correction

(c) Image Dehazing (d) Image Harmonization

Figure 10. **Limitations of Applying Our Model to Other Tasks.** For each image pair, the left is the input image while the right is the output image. The top-left corner of the output image shows the reference image. As indicated by red arrows: (a) heavy noise may be introduced if the input low-light image is too dark; (b) incorrect colors may be left over if the input underwater image is blurry; (c) non-uniform distributed haze in the input image may not be removed; (d) the output foreground may overfit the reference background with a solid color.

(with a learning rate of $3e^{-4}$ and a batch size of 1). Our training loss is adopted from Zeng *et al.* [43]. We follow Wang *et al.* [38] to use PSNR, SSIM, and LPIPS as performance metrics. The results on the MIT-Adobe FiveK benchmark [2] (Table 3) demonstrate that our model performs on par with the state-of-the-art methods. The inference speed of our model is also comparable to the methods designed to run in real time [38, 43].

E. On-Device Deployment of Neural Preset

On-device [8] (*e.g.*, mobile or browser) applications expect a small computational overhead in the client to support real-time UI responses and a small amount of data exchanges between the client/server to save network bandwidth. However, existing color style transfer models are not suitable for such applications: deploying them in the client requires too much memory and is computationally expensive, while deploying them in the server will significantly increase the network bandwidth since high-resolution images must be transmitted over the internet. Instead, our Neural Preset supports distributed deployment to alleviate this problem: we can deploy the encoder E in the server and *nDNCM*/*sDNCM* in the client. In this way, the client-side calculation can be fast. Besides, only thumbnails and the DNCM parameters are transmitted over the internet.

As illustrated in Fig. 11, when transferring color style from a style image $\mathbf{I}_s$ to an input image $\mathbf{I}_c$, the client first downsamples the two images and transmits their thumbnails to the server. Note that the file size of a thumbnail with a resolution of 256×256 is only about 30KB, but the file size of a 4K resolution image can be up to 20MB. Then,

Method	Performance		GPU Inference
	MSE ↓	PSNR ↑	Time ↓ / Memory ↓
S^2AM [7]	59.67	34.35	0.148 s / 6.3 GB
DoveNet [6]	52.36	34.75	0.072 s / 6.5 GB
BargainNet [4]	37.82	35.88	0.086 s / 3.7 GB
IntrinsicIH [13]	38.71	35.90	0.833 s / 16.5 GB
IHT [12]	37.07	36.71	0.196 s / 18.5 GB
Harmonizer [19]	24.26	37.84	0.017 s / 2.3 GB
CDTNet [5]	**23.75**	**38.23**	0.023 s / 8.1 GB
DNCM (Ours)	24.31	37.97	**0.006 s / 1.1 GB**

Table 2. **Image Harmonization Results on iHarmony4.** The performance metrics (MSE and PSNR) are computed at 256×256 resolution, while the inference time and memory footprint are measured at Full HD resolution on a Nvidia RTX3090 GPU.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
UPE [37]	20.03	0.7841	0.2523
RSGUNet [16]	21.37	0.7998	0.1861
HDRNet [10]	22.15	0.8403	0.1823
Adaptive 3D LUT [43]	22.27	0.8368	0.1832
Learnable 3D LUT [38]	**23.17**	0.8636	0.1451
DNCM (Ours)	23.12	**0.8697**	**0.1439**

Table 3. **Image Color Enhancement Results on FiveK.** All performance metrics are calculated at the Full resolution, *i.e.*, the original resolution of the test samples.

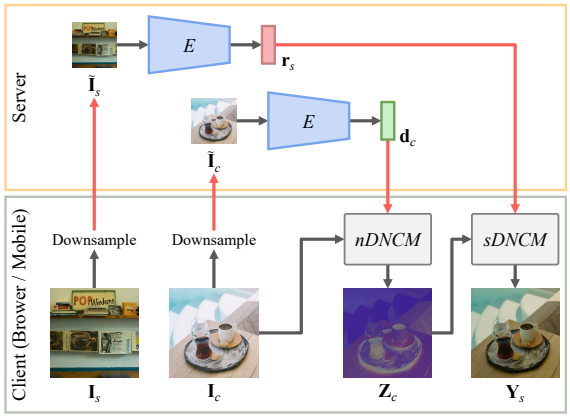


Figure 11. **On-Device Deployment of Neural Preset.** Black arrows represent server or client processing flow, while red arrows represent network transmission between server and client.

the server calculates the color style parameters $\mathbf{d}_c/\mathbf{r}_s$ (the data size is about 3KB) from the uploaded thumbnails and transmits $\mathbf{d}_c/\mathbf{r}_s$ back to the client. Finally, the client performs *nDNCM*/*sDNCM* on the high-resolution $\mathbf{I}_c$ to complete color style transfer, which is fast and require only a small memory footprint.

Acknowledgments

Most of the images we displayed in this paper are from the *pexels.com* and *flickr.com* websites, which are under the Creative Commons license. The rest of the images we displayed are from publicly available datasets [2, 6, 9, 17, 21, 23, 25, 39, 44]. We thank the artists and photographers for sharing their amazing works online, and we thank the researchers who constructed the datasets. Besides, we thank the project contributors (Weiwei Chen, Jing Li, and Xiaojun Zheng) for their help in developing demos. We also appreciate all user study participants and anonymous peer reviewers.

References

[1] Jie An, Haoyi Xiong, Jun Huan, and Jiebo Luo. Ultrafast photorealistic style transfer via neural architecture search. In *AAAI*, 2020. 2, 3, 5

[2] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR*, 2011. 7, 8

[3] Tai-Yin Chiu and Danna Gurari. Photowct2: Compact autoencoder for photorealistic style transfer resulting from blockwise training and skip connections of high-frequency residuals. In *WACV*, 2022. 5

[4] Wenyan Cong, Li Niu, Jianfu Zhang, Jing Liang, and Liqing Zhang. Bargainnet: Background-guided domain translation for image harmonization. In *ICME*, 2021. 5, 7

[5] Wenyan Cong, Xinhao Tao, Li Niu, Jing Liang, Xuesong Gao, Qihao Sun, and Liqing Zhang. High-resolution image harmonization via collaborative dual transformations. In *CVPR*, 2022. 5, 7

[6] Wenyan Cong, Jianfu Zhang, Li Niu, Liu Liu, Zhixin Ling, Weiyuan Li, and Liqing Zhang. Dovenet: Deep image harmonization via domain verification. In *CVPR*, 2020. 5, 7, 8

[7] Xiaodong Cun and Chi-Man Pun. Improving the harmony of the composite image by spatial-separated attention module. *IEEE TIP*, 2020. 5, 7

[8] Sauptik Dhar, Junyao Guo, Jiayi (Jason) Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM TOIT*, 2021. 7

[9] Cameron Fabbri, Md Jahidul Islam, and Junaed Sattar. Enhancing underwater imagery using generative adversarial networks. In *ICRA*, 2018. 8

[10] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *TOG*, 2017. 6, 7

[11] Jie Gui, Xiaofeng Cong, Yuan Cao, Wenqi Ren, Jun Zhang, Jing Zhang, and Dacheng Tao. A comprehensive survey on image dehazing based on deep learning. In *IJCAI*, 2021. 3

[12] Zonghui Guo, Dongsheng Guo, Haiyong Zheng, Zhaorui Gu, Bing Zheng, and Junyu Dong. Image harmonization with transformer. In *ICCV*, 2021. 5, 7

[13] Zonghui Guo, Haiyong Zheng, Yufeng Jiang, Zhaorui Gu, and Bing Zheng. Intrinsic image harmonization. In *CVPR*, 2021. 5, 7

[14] Magnus R. Hestenes. Multiplier and gradient methods. *JOTA*, 1969. 1

[15] Man M. Ho and Jinjia Zhou. Deep preset: Blending and retouching photos with color style transfer. In *WACV*, 2021. 5

[16] Jie Huang, Peng Fei Zhu, Mingrui Geng, Jie Ran, Xingguang Zhou, Chen Xing, Pengfei Wan, and Xiangyang Ji. Range scaling global u-net for perceptual image enhancement on mobile devices. In *ECCVW*, 2018. 6, 7

[17] Md Jahidul Islam, Youya Xia, and Junaed Sattar. Fast underwater image enhancement for improved visual perception. *IEEE RA-L*, 2020. 8

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2

[19] Zhanghan Ke, Chunyi Sun, Lei Zhu, Ke Xu, and Rynson W.H. Lau. Harmonizer: Learning to perform white-box image and video harmonization. In *ECCV*, 2022. 5, 7

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2015. 2, 5, 6

[21] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2018. 8

[22] Chenyang Lei, Yazhou Xing, Hao Ouyang, and Qifeng Chen. Deep video prior for video consistency and propagation. *IEEE TPAMI*, 2022. 3

[23] Boyi Li, Wenqi Ren, Dengpan Fu, Dacheng Tao, Dan Feng, Wenjun Zeng, and Zhangyang Wang. Benchmarking single-image dehazing and beyond. *IEEE TIP*, 2019. 8

[24] Chongyi Li, Chunle Guo, Ling-Hao Han, Jun Jiang, Ming-Ming Cheng, Jinwei Gu, and Chen Change Loy. Low-light image and video enhancement using deep learning: A survey. *IEEE TPAMI*, 2021. 3

[25] Chongyi Li, Chunle Guo, Wenqi Ren, Runmin Cong, Junhui Hou, Sam Kwong, and Dacheng Tao. An underwater image enhancement benchmark dataset and beyond. *IEEE TIP*, 2020. 8

[26] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *ECCV*, 2018. 2, 3, 5

[27] Jun Ling, Han Xue, Li Song, Rong Xie, and Xiao Gu. Region-aware adaptive instance normalization for image harmonization. In *CVPR*, 2021. 5

[28] Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE TPAMI*, 2021. 1

[29] Risheng Liu, Pan Mu, Xiaoming Yuan, and Shangzhi Zeng. A general descent aggregation framework for gradient-based bi-level optimization. *IEEE TPAMI*, 2021. 1

[30] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *CVPR*, 2017. 5

[31] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017. 2

[32] Li Niu, Wenyan Cong, Liu Liu, Yan Hong, Bo Zhang, Jing Liang, and Liqing Zhang. Making images real again: A comprehensive survey on deep image composition. *Preprint*, 2021. 3

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[34] Xavier Soria, Gonzalo Pomboza-Junez, and Angel Domingo Sappa. Ldc: Lightweight dense cnn for edge detection. *IEEE Access*, 2022. 2, 3

[35] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 5

[36] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, Xin Lu, and Ming-Hsuan Yang. Deep image harmonization. In *CVPR*, 2017. 5

[37] Ruixing Wang, Qing Zhang, Chi-Wing Fu, Xiaoyong Shen, Wei-Shi Zheng, and Jiaya Jia. Underexposed photo enhancement using deep illumination estimation. In *CVPR*, 2019. 6, 7

[38] Tao Wang, Yong Li, Jingyang Peng, Yipeng Ma, Xian Wang, Fenglong Song, and Youliang Yan. Real-time image enhancer via learnable spatial-aware 3d lookup tables. In *ICCV*, 2021. 6, 7

[39] Chen Wei, Weijing Wang, Wenhan Yang, and Jiaying Liu. Deep retinex decomposition for low-light enhancement. In *BMVC*, 2018. 8

[40] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, 2015. 2, 3

[41] Miao Yang, Jintong Hu, Chongyi Li, Gustavo Rohde, Yixiang Du, and Ke Hu. An in-depth survey of underwater image enhancement and restoration. *IEEE Access*, 2019. 3

[42] Jaejun Yoo, Youngjung Uh, Sanghyuk Chun, Byeongkyu Kang, and Jung-Woo Ha. Photorealistic style transfer via wavelet transforms. In *ICCV*, 2019. 2, 3, 5

[43] Hui Zeng, Jianrui Cai, Lida Li, Zisheng Cao, and Lei Zhang. Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time. *IEEE TPAMI*, 2020. 6, 7

[44] Xinyi Zhang, Hang Dong, Jinshan Pan, Chao Zhu, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Fei Wang. Learning to restore hazy video: A new real-world dataset and a new method. In *CVPR*, 2021. 8