Deep White-Balance Editing
Supplemental Material

Mahmoud Afifi[1,2] Michael S. Brown[1]
[1]Samsung AI Center (SAIC) – Toronto [2]York University
{mafifi, mbrown}@eecs.yorku.ca

Figure S1: Our deep white-balance editing framework produces compelling results and generalizes well to images outside our training data (e.g., image above taken from an Internet photo repository). Top: input image captured with a wrong WB setting. Bottom: our framework's AWB, Incandescent WB, and Shade WB results. Photo credit: *M@tth1eu* Flickr–CC BY-NC 2.0.

We have proposed a deep learning framework for white-balance (WB) editing of sRGB images captured by a camera. Our method achieves state-of-the-art results compared with recent methods for post-capture auto WB (AWB) correction and manipulation [1, 2, 4, 8]. Our method generalizes well to unseen images that were not included in the training/validation steps. Fig. S1 shows an example of an image taken from an Internet photo repository. As shown, our method produces compelling results with the following WB settings: (1) AWB (a "correct" WB setting); (2) Incandescent WB (an indoor WB setting); (3) Shade WB (an outdoor WB setting).

The rest of our supplemental material is organized as follows. Sec. S1 provides an ablation study on the choices made in the paper regarding our DNN and color mapping function. Specifically, we study the effect of different color mapping kernels and training loss functions. Sec. S2 provides additional training details. Then, we provide additional results of our experiments in Sec. S3.

S1. Ablation study

Kernel function for color mapping In the main paper, we propose to apply a color mapping in order to generate the final output image. Our color mapping allows our method to run on an acceptable CPU inference time to process high-resolution input images.

Fig. S2 shows an example of two input images rendered with wrong WB settings. The first image has 5700×2126 pixels, while the second image has 1000×373 pixels. If our deep neural network (DNN) model processes the input images with their original resolution on a CPU, the running time is $\sim$32 and $\sim$1.1 seconds on an Intel Xeon E5-1607

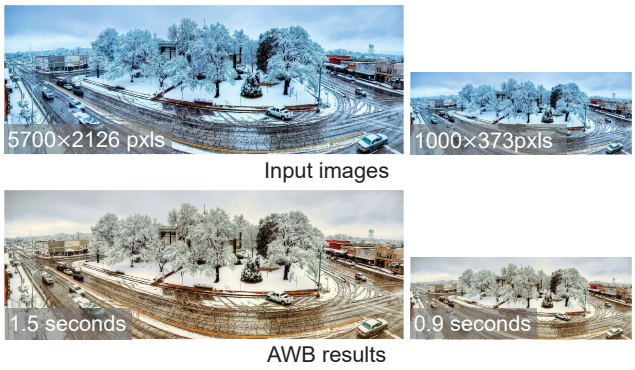


Figure S2: Running time of our algorithm for images of different sizes. Our color mapping procedure, described in Section 2.5 of the main paper, allows our method to run in an acceptable time for different-sized input images (∼1–1.5 seconds on average). Photo credit: *Wes Browning* Flickr-CC BY-NC-SA 2.0.

V4 (10M Cache, 3.10 GHz) machine with 32 GB RAM for each image, respectively. Our color mapping procedure allows our DNN framework to process a resized version of the input image, with the final output image being computed with the input image's original resolution. Accordingly, this color mapping procedure improves the runtime performance of our framework when it is deployed on limited computing resources. For example, our DNN framework using the proposed color mapping procedure takes only ∼1.5 and ∼0.9 seconds on a CPU with the same configure for the two images in Fig. S2.

In the main paper, we employed an 11-dimensional kernel function to compute our color mapping matrix. Table S1 provides results obtained by using different kernel functions. Specifically, we show results of using 9-dimensional, 11-dimensional, and 34-dimensional polynomial kernels [6, 7]. As shown, the 11-dimensional polynomial kernel achieves the best results on Set 1–*Test* of the Rendered WB dataset [2].

$\mathtt{L}_1$ vs. $\mathtt{L}_2$ training loss functions In the main paper, we showed results using the $\mathtt{L}_1$-norm loss between predicted patches and the corresponding ground truth training patches.

We also tested squared $\mathtt{L}_2$-norm loss and found that both loss functions work well for our task. Table S2 shows quantitative comparisons between our results by training using each loss function. The reported results in Table S2 are obtained by testing each model on the rendered version of Cube+ dataset [2,3] for both tasks—namely, WB correction and manipulation. Fig. S3 illustrates a qualitative comparison.

S2. Additional Training Details

As discussed in the main paper, our goal is not to reconstruct unprocessed raw-RGB images. Instead, our goal is to model the functionality of the F and G functions (Eq. 1 in the main paper) in the sense that the function F maps a given image rendered with any WB setting to the same starting point that could be used afterwards by G to re-render the image with different WB settings. Our encoder f maps the given input image into a common latent representation that is used afterwards by each decoder g to generate images with target WB settings.

We trained our approach end-to-end, as discussed in Sec. 2.4 of the main paper. During the training phase, each mini-batch contains random patches selected from training images that may contain different WB settings. During training, each decoder receives the generated latent representations by our single encoder and generates corresponding patches with the target WB setting. The loss function is computed using the result of each decoder and is followed by gradient backpropagation from all decoders aggregated back to our single encoder via the skip-layer connections. Thus, the encoder is trained to map the images into an intermediate latent space that is beneficial for generating the target WB setting by each decoder.

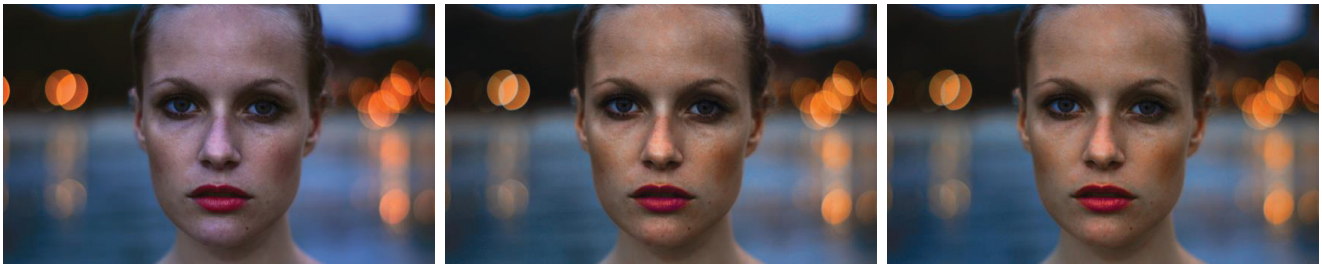
S3. Additional Results

In the main paper, we compared our framework against a vanilla U-Net architecture [9]. Table S3 reports additional details to those reported results in the main paper. Specifically, we show the mean and the first, second (median), and third quartile of mean square error, mean angular error, and $\triangle\mathbf{E}$ 2000 error [10] obtained by our framework and the multi-U-Net model. The results were obtained after training each model for 88,000 iterations on the same training dataset and settings.

In Table S4 and Table S5, we provide results for the evaluation of AWB correction and WB manipulation using the $\triangle\mathbf{E}$ 76 metric.

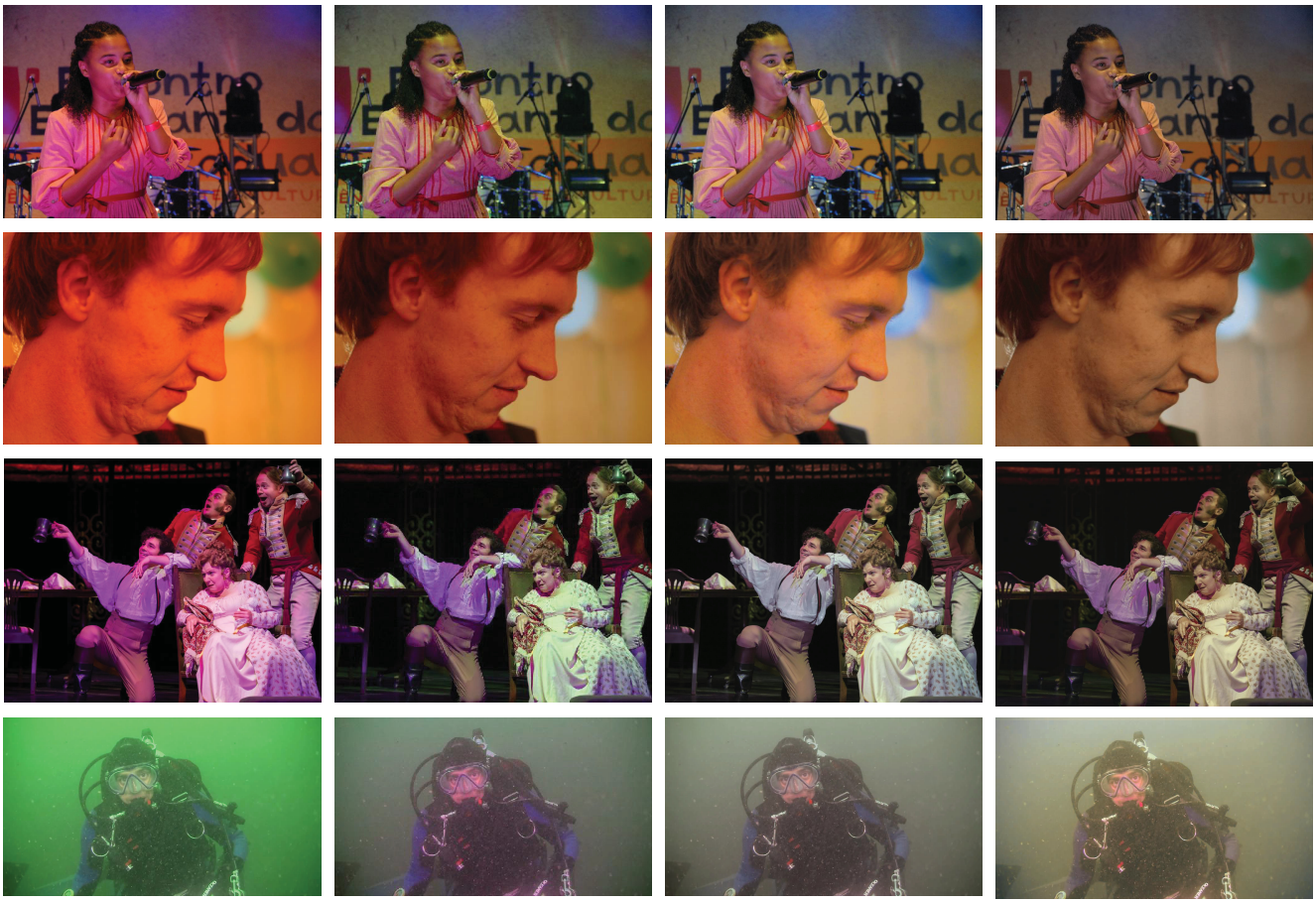
Fig. S4 shows example images randomly selected from the Internet with strong color casts. As demonstrated, our method generalizes well and produces better results compared to other methods for WB correction.

Our method can accept input images with any WB settings, including white-balanced images, and accurately produces the target WB settings. Fig. S5 shows an example of white-balanced input image. Our AWB result is almost identical to the input image, as the input image is correctly white-balanced. Additionally, other output WB settings properly work as shown in Fig. S5-(C). Moreover, our method can produce consistent results for the same input image rendered with different WB settings. Fig. S6 shows a scene image rendered with two different WB settings from



(A) Input image | (B) Our AWB trained with L_1-norm loss | (B) Our AWB trained with L_2-norm loss

Figure S3: (A) Input image. (B) AWB result of our deep-WB model trained using the $\mathtt{L}_1$-norm loss. (C) AWB result of our deep-WB model trained using the squared $\mathtt{L}_2$-norm loss. Photo credit: *Mark Jenkinson* Flickr–CC BY-NC-SA 2.0.



(A) Input image | (B) Quasi-U CC | (C) KNN-WB | (D) Our AWB

Figure S4: (A) Input images. (B) Results of quasi-U CC [4]. (C) Results of KNN-WB [2]. (D) Our deep-WB results. Photo credits (from top to bottom): *Educao Bahia* Flickr–CC BY-NC-SA 2.0, *Duncan Yoyos* Flickr–CC BY-NC 2.0, *The Lowry* Flickr–CC BY-NC-SA 2.0, and *SCUBATOO* Flickr–CC BY 2.0, respectively.

the rendered Cube+ dataset [2, 3]. As shown, our method can produce consistent results regardless of the input image WB setting.

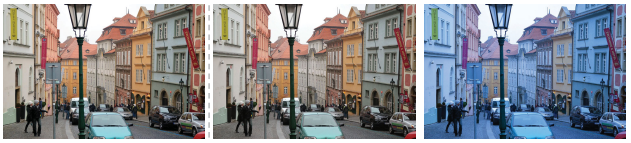
Our method fails in some scenarios, as demonstrated in Fig. S7. In particular, our method is not designed to deal with multi-illuminant scenes, as is the case in the first and second rows in Fig. S7. The third row in Fig. S7 shows a classical failure example in the color constancy literature,

Table S1: This table shows AWB results of different mapping kernels on the Set 1–*Test* images of the Rendered WB dataset [2]. The shown images were not used in the training process of our model. We report the mean, first, second (median), and third quartile (Q1, Q2, and Q3) of mean square error (MSE), mean angular error (MAE), and △**E** 2000 [10]. The top results are indicated with yellow.

Color mapping	**MSE**				**MAE**				**△E 2000**			
	Mean	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**
Kernel-9 [6]	75.2	12.80	44.35	98.27	3.23°	1.99°	2.77°	3.92°	3.89	2.26	3.48	4.94
Kernel-11 [7]	74.76	12.80	43.97	95.98	3.20°	1.97°	2.74°	3.85°	3.90	2.28	3.47	4.91
Kernel-34 [6]	75.78	13.58	45.2	96.27	3.21°	1.98°	2.72°	3.82°	3.90	2.32	3.43	4.91

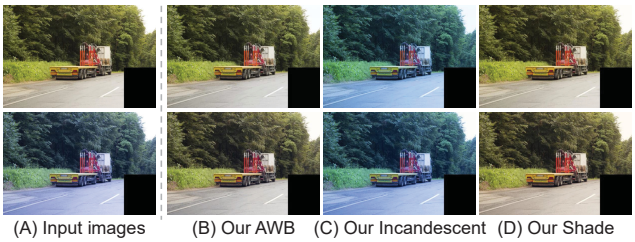
Table S2: Comparison between results of using $\mathtt{L}_1$ and $\mathtt{L}_2$ loss functions to train our framework. Shown results were obtained using the rendered version of Cube+ dataset [2, 3]. We report the mean, first, second (median), and third quartile (Q1, Q2, and Q3) of mean square error (MSE), mean angular error (MAE), and △**E** 2000 [10]. The top results are indicated with boldface.

Method	**MSE**				**MAE**				**△E 2000**			
	Mean	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**
AWB correction												
$\mathtt{L}_1$ loss	**80.46**	**15.43**	**33.88**	**74.42**	3.45°	1.87°	2.82°	4.26°	4.59	2.68	3.81	5.53
$\mathtt{L}_2$ loss	83.35	16.01	34.99	75.34	**3.24°**	**1.75°**	**2.65°**	**3.96°**	**4.39**	**2.54**	**3.63**	**5.31**
WB manipulation												
$\mathtt{L}_1$ loss	199.38	**32.30**	**63.34**	**142.76**	**5.40°**	**2.67°**	**4.04°**	**6.36°**	**5.98**	**3.44**	**4.78**	**7.29**
$\mathtt{L}_2$ loss	**196.58**	36.76	72.08	166.24	5.50°	2.79°	4.25°	6.79°	6.06	3.46	5.03	7.62



(A) Input image (B) Our AWB (C) Our Incandescent WB

Figure S5: (A) Input image is correctly white-balanced. (B) Our AWB result. (D) Our Incandescent WB result. Photo credit: *Tom Magliery* Flickr–CC BY-NC-SA 2.0.



(A) Input images (B) Our AWB (C) Our Incandescent WB (D) Our Shade WB

Figure S6: Our method produces consistent results from input images rendered with different WB settings. (A) Input images from the rendered version of the Cube+ dataset [2,3] with different WB settings. (B)-(D) Our results.

where the input image does not have enough semantic information to distinguish between original object colors and the scene illuminant colors. As we can see this failure case

(A) Input images (B) Quasi-U CC (C) KNN-WB (D) Our AWB

Figure S7: Failure examples. (A) Input images. (B) Results of quasi-U CC [4]. (C) Results of KNN-WB [2]. (D) Our deep-WB results. Photo credits (from top to bottom): Martina Flickr–CC BY-NC-SA 2.0, *Steve Tannock* Flickr–CC BY-NC-SA 2.0 and *Travis Nep Smith* Flickr–CC BY-NC 2.0, respectively.

also is challenging other recent methods [2, 4]. Lastly, Fig. S8 provides additional qualitative results of our method.

References

[1] Mahmoud Afifi and Michael S Brown. What else can fool deep learning? Addressing color constancy errors on deep neural network performance. In *ICCV*, 2019. 1, 6

Table S3: Comparison between results of our framework and the traditional U-Net architecture [9]. For the AWB task, we trained a single U-Net model, while we trained two U-Net models for the WB manipulation task. We show results of trained models using the same training settings for 88,000 iterations. Shown results are reported using Set 2 of the Rendered WB dataset [2] and the rendered version of the Cube+ dataset [2,3]. We report the mean, first, second (median), and third quartile (Q1, Q2, and Q3) of mean square error (MSE), mean angular error (MAE), and $\triangle$**E** 2000 [10]. The top results are indicated with yellow and boldface.

Method	**MSE**				**MAE**				**$\triangle$E 2000**			
	Mean	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**
AWB on Rendered WB dataset: Set 2 (2,881 images) [2]												
U-Net [9]	187.25	58.87	119.67	227.26	4.85°	2.79°	4.17°	6.12°	6.23	4.41	5.62	7.46
Ours	**124.47**	**30.81**	**75.87**	**153.18**	**3.81°**	**2.11°**	**3.14°**	**4.80°**	**4.99**	**3.30**	**4.50**	**6.07**
WB manipulation on Rendered Cube+ dataset with different WB settings (10,242 images) [2, 3]												
Multi-U-Net [9]	234.77	66.24	107.11	194.30	5.78°	3.22°	4.54°	7.13°	6.87	4.47	5.85	8.18
Ours	**206.81**	**39.43**	**78.78**	**177.78**	**5.68°**	**2.90°**	**4.37°**	**6.97°**	**6.23**	**3.53**	**5.18**	**7.86**

(A) Input images (B) Our AWB (C) Our Incandescent WB (D) Our Fluorescent WB (E) Our Shade WB

Figure S8: Additional qualitative results of our method. (A) Input images. (B) AWB results. (C) Incandescent WB results. (D) Fluorescent WB results. (E) Shade WB results. Photo credits (from top to bottom): *Santiago Borthwick* Flickr–CC BY-NC-SA 2.0, *Roland Tanglao* Flickr–CC0 1.0, *sg.sam* Flickr–CC BY-NC-SA 2.0, and *Biggleswade Blue* Flickr–CC BY-NC-SA 2.0.

[2] Mahmoud Afifi, Brian Price, Scott Cohen, and Michael S Brown. When color constancy goes wrong: Correcting improperly white-balanced images. In *CVPR*, 2019. 1, 2, 3, 4, 5, 6

[3] Nikola Banić and Sven Lončarić. Unsupervised learning for color constancy. *arXiv preprint arXiv:1712.00436*, 2017. 2, 3, 4, 5, 6

[4] Simone Bianco and Claudio Cusano. Quasi-unsupervised color constancy. In *CVPR*, 2019. 1, 3, 4, 6

[5] 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. 6

[6] Graham D Finlayson, Michal Mackiewicz, and Anya Hurlbert. Color correction using root-polynomial regression. *IEEE Transactions on Image Processing*, 24(5):1460–1470, 2015. 2, 4

[7] Guowei Hong, M Ronnier Luo, and Peter A Rhodes. A study of digital camera colorimetric characterisation based on polynomial modelling. *Color Research & Application*, 26(1):76–84, 2001. 2, 4

[8] Yuanming Hu, Baoyuan Wang, and Stephen Lin. FC4: Fully convolutional color constancy with confidence-weighted pooling. In *CVPR*, 2017. 1, 6

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015. 2, 5

[10] Gaurav Sharma, Wencheng Wu, and Edul N Dalal. The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30(1):21–30, 2005. 2, 4, 5

Table S4: $\triangle$**E** 76 results: complementary results for Table 1 in the main paper. The top results are indicated with yellow and boldface.

Method	**$\triangle$E 76**			
	Mean	**Q1**	**Q2**	**Q3**
Rendered WB dataset: Set 1–*Test* [2]				
FC4 [8]	8.73	4.25	7.49	11.83
Quasi-U CC [4]	8.08	4.00	6.88	10.77
KNN-WB [2]	**4.56**	**2.51**	**3.92**	**5.87**
Ours	4.78	2.62	4.17	6.27
Rendered WB dataset: Set 2 [2]				
FC4 [8]	15.89	9.49	14.35	20.46
Quasi-U CC [4]	15.70	9.35	14.46	20.47
KNN-WB [2]	7.46	4.19	6.04	9.38
Ours	**6.60**	**3.91**	**5.61**	**8.04**
Rendered Cube+ dataset [2, 3]				
FC4 [8]	13.42	7.4	11.98	17.39
Quasi-U CC [4]	9.78	3.45	6.69	14.18
KNN-WB [2]	7.23	3.74	5.59	8.56
Ours	**6.00**	**3.20**	**4.76**	**7.34**

Table S5: $\triangle$**E** 76 results: complementary results for Table 2 in the main paper. The top results are indicated with yellow and boldface.

Method	**$\triangle$E 76**			
	Mean	**Q1**	**Q2**	**Q3**
Rendered Cube+ dataset [2, 3]				
KNN-WB emulator [1]	13.20	6.33	11.01	18.19
Ours	**9.73**	**4.58**	**7.07**	**12.03**
Rendered MIT-Adobe FiveK dataset [2, 5]				
KNN-WB emulator [1]	10.48	5.18	8.38	13.64
Ours	**7.44**	**4.23**	**6.23**	**9.20**