Deep White-Balance Editing

Mahmoud Afifi[1,2*]   Michael S. Brown[1]

[1]Samsung AI Center (SAIC) – Toronto   [2]York University

{mafifi, mbrown}@eecs.yorku.ca

Abstract

We introduce a deep learning approach to realistically edit an sRGB image's white balance. Cameras capture sensor images that are rendered by their integrated signal processor (ISP) to a standard RGB (sRGB) color space encoding. The ISP rendering begins with a white-balance procedure that is used to remove the color cast of the scene's illumination. The ISP then applies a series of nonlinear color manipulations to enhance the visual quality of the final sRGB image. Recent work by [3] showed that sRGB images that were rendered with the incorrect white balance cannot be easily corrected due to the ISP's nonlinear rendering. The work in [3] proposed a k-nearest neighbor (KNN) solution based on tens of thousands of image pairs. We propose to solve this problem with a deep neural network (DNN) architecture trained in an end-to-end manner to learn the correct white balance. Our DNN maps an input image to two additional white-balance settings corresponding to indoor and outdoor illuminations. Our solution not only is more accurate than the KNN approach in terms of correcting a wrong white-balance setting but also provides the user the freedom to edit the white balance in the sRGB image to other illumination settings.

1. Introduction and related work

White balance (WB) is a fundamental low-level computer vision task applied to all camera images. WB is performed to ensure that scene objects appear as the same color even when imaged under different illumination conditions. Conceptually, WB is intended to normalize the effect of the captured scene's illumination such that all objects appear as if they were captured under ideal "white light". WB is one of the first color manipulation steps applied to the sensor's unprocessed raw-RGB image by the camera's onboard integrated signal processor (ISP). After WB is performed, a number of additional color rendering steps are applied by the ISP to further process the raw-RGB image to its final
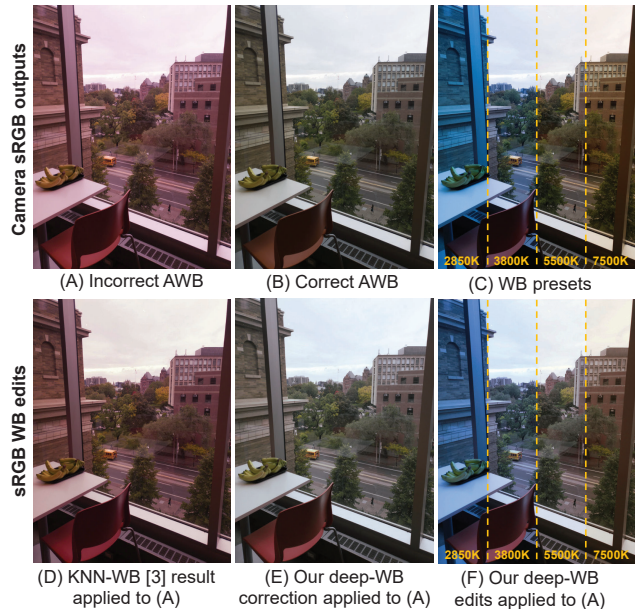

Figure 1: Top row: (A)-(C) are sRGB images produced by a camera's ISP with different WB settings. (A) An incorrect WB representing a failed AWB. (B) A correct AWB for the scene. (C) Results of the camera's manual presets. Bottom row: (D)-(F) are post-capture edits of sRGB image (A)'s WB. (D) Result from the recent KNN-WB correction method [3]. (E) Our result to correct the WB in (A). (F) Our results to produce different outputs corresponding to the camera's presets.

standard RGB (sRGB) encoding. While the goal of WB is intended to normalize the effect of the scene's illumination, ISPs often incorporate aesthetic considerations in their color rendering based on photographic preferences. Such preferences do not always conform to the white light assumption and can vary based on different factors, such as cultural preference and scene content [8, 13, 22, 31].

Most digital cameras provide an option to adjust the WB settings during image capturing. However, once the WB setting has been selected and the image is fully processed by the ISP to its final sRGB encoding it becomes challenging to perform WB editing without access to the original

---

unprocessed raw-RGB image [3]. This problem becomes even more difficult if the WB setting was wrong, which results in a strong color cast in the final sRGB image.

The ability to edit the WB of an sRGB image not only is useful from a photographic perspective but also can be beneficial for computer vision applications, such as object recognition, scene understanding, and color augmentation [2,6,19]. A recent study in [2] showed that images captured with an incorrect WB setting produce a similar effect of an untargeted adversarial attack for deep neural network (DNN) models.

In-camera WB procedure   To understand the challenge of WB editing in sRGB images it is useful to review how cameras perform WB. WB consists of two steps performed in tandem by the ISP: (1) estimate the camera sensor's response to the scene illumination in the form of a raw-RGB vector; (2) divide each R/G/B color channel in the raw-RGB image by the corresponding channel response in the raw-RGB vector. The first step of estimating the illumination vector constitutes the camera's auto-white-balance (AWB) procedure. Illumination estimation is a well-studied topic in computer vision—representative works include [1, 7–10, 14, 17, 18, 23, 28, 33]. In addition to AWB, most cameras allow the user to manually select among WB presets in which the raw-RGB vector for each preset has been determined by the camera manufacturer. These presets correspond to common scene illuminants (e.g., Daylight, Shade, Incandescent).

Once the scene's illumation raw-RGB vector is defined, a simple linear scaling is applied to each color channel independently to normalize the illumination. This scaling operation is performed using a 3×3 diagonal matrix. The white-balanced raw-RGB image is then further processed by camera-specific ISP steps, many nonlinear in nature, to render the final images in an output-referred color space—namely, the sRGB color space. These nonlinear operations make it hard to use the traditional diagonal correction to correct images rendered with strong color casts caused by camera WB errors [3].

WB editing in sRGB   In order to perform accurate post-capture WB editing, the rendered sRGB values should be properly reversed to obtain the corresponding unprocessed raw-RGB values and then re-rendered. This can be achieved only by accurate radiometric calibration methods (e.g., [12,24,34]) that compute the necessary metadata for such color de-rendering. Recent work by Afifi et al. [3] proposed a method to directly correct sRGB images that were captured with the wrong WB setting. This work proposed an exemplar-based framework using a large dataset of over 65,000 sRGB images rendered by a software camera pipeline with the wrong WB setting. Each of these sRGB

images had a corresponding sRGB image that was rendered with the correct WB setting. Given an input image, their approach used a KNN strategy to find similar images in their dataset and computed a mapping function to the corresponding correct WB images. The work in [3] showed that this computed color mapping constructed from exemplars was effective in correcting an input image. Later Afifi and Brown [2] extended their KNN idea to map a correct WB image to appear incorrect for the purpose of image augmentation for training deep neural networks. Our work is inspired by [2,3] in their effort to directly edit the WB in an sRGB image. However, in contrast to the KNN frameworks in [2, 3], we cast the problem within a single deep learning framework that can achieve both tasks—namely, WB correction and WB manipulation as shown in Fig. 1.

Contribution   We present a novel deep learning framework that allows realistic post-capture WB editing of sRGB images. Our framework consists of a single encoder network that is coupled with three decoders targeting the following WB settings: (1) a "correct" AWB setting; (2) an indoor WB setting; (3) an outdoor WB setting. The first decoder allows an sRGB image that has been incorrectly white-balanced image to be edited to have the correct WB. This is useful for the task of post-capture WB correction. The additional indoor and outdoor decoders provide users the ability to produce a wide range of different WB appearances by blending between the two outputs. This supports photographic editing tasks to adjust an image's aesthetic WB properties. We provide extensive experiments to demonstrate that our method generalizes well to images outside our training data and achieves state-of-the-art results for both tasks.

2. Deep white-balance editing

2.1. Problem formulation

Given an sRGB image, $\mathbf{I}_{\mathrm{WB}^{(\mathrm{in})}}$, rendered through an unknown camera ISP with an arbitrary WB setting $\mathrm{WB}^{(\mathrm{in})}$, our goal is to edit its colors to appear as if it were re-rendered with a target WB setting $\mathrm{WB}^{(t)}$.

As mentioned in Sec. 1, our task can be accomplished accurately if the original unprocessed raw-RGB image is available. If we could recover the unprocessed raw-RGB values, we can change the WB setting $\mathrm{WB}^{(\mathrm{in})}$ to $\mathrm{WB}^{(t)}$, and then re-render the image back to the sRGB color space with a software-based ISP. This ideal process can be described by the following equation:

$$\mathbf{I}_{\mathrm{WB}^{(t)}} = G\left(F\left(\mathbf{I}_{\mathrm{WB}^{(\mathrm{in})}}\right)\right), \qquad (1)$$

where $F : \mathbf{I}_{\mathrm{WB}^{(\mathrm{in})}} \to \mathbf{D}_{\mathrm{WB}^{(\mathrm{in})}}$ is an unknown reconstruction function that reverses the camera-rendered sRGB image $\mathbf{I}$
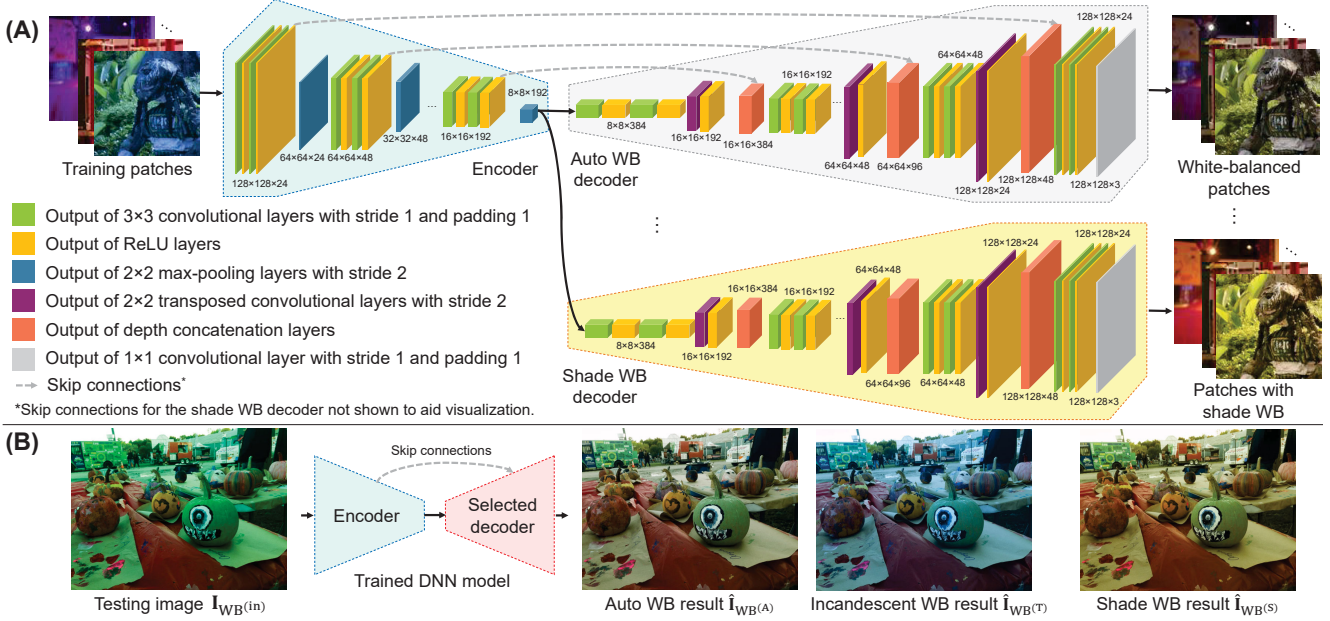
Figure 2: Proposed multi-decoder framework for sRGB WB editing. (A) Our proposed framework consists of a single encoder and multiple decoders. The training process is performed in an end-to-end manner, such that each decoder "re-renders" the given training patch with a specific WB setting, including AWB. For training, we randomly select image patches from the Rendered WB dataset [3]. (B) Given a testing image, we produce the targeted WB setting by using the corresponding trained decoder.

back to its corresponding raw-RGB image $\mathbf{D}$ with the current $\mathrm{WB^{(in)}}$ setting applied and $G : \mathbf{D}_{\mathrm{WB^{(in)}}} \rightarrow \mathbf{I}_{\mathrm{WB^{(t)}}}$ is an unknown camera rendering function that is responsible for editing the WB setting and re-rendering the final image.

2.2. Method overview

Our goal is to model the functionality of $G\left(F\left(\cdot\right)\right)$ to generate $\mathbf{I}_{\mathrm{WB^{(t)}}}$. We first analyze how the functions G and F cooperate to produce $\mathbf{I}_{\mathrm{WB^{(t)}}}$. From Eq. 1, we see that the function F transforms the input image $\mathbf{I}_{\mathrm{WB^{(in)}}}$ into an intermediate representation (i.e., the raw-RGB image with the captured WB setting), while the function G accepts this intermediate representation and renders it with the target WB setting to an sRGB color space encoding.

Due to the nonlinearities applied by the ISP's rendering chain, we can think of G as a hybrid function that consists of a set of sub-functions, where each sub-function is responsible for rendering the intermediate representation with a specific WB setting.

Our ultimate goal is *not* to reconstruct/re-render the original raw-RGB values, but rather to generate the final sRGB image with the target WB setting $\mathrm{WB^{(t)}}$. Therefore, we can model the functionality of $G\left(F\left(\cdot\right)\right)$ as an encoder/decoder scheme. Our encoder f transfers the input image into a latent representation, while each of our decoders (g_1, g_2, ...) generates the final images with a different WB setting. Similar to Eq. 1, we can formulate our framework as follows:

$$\hat{\mathbf{I}}_{\mathrm{WB^{(t)}}} = g_t\left(f\left(\mathbf{I}_{\mathrm{WB^{(in)}}}\right)\right), \qquad (2)$$

where $f : \mathbf{I}_{\mathrm{WB^{(in)}}} \rightarrow \mathcal{Z}$, $g_t : \mathcal{Z} \rightarrow \hat{\mathbf{I}}_{\mathrm{WB^{(t)}}}$, and $\mathcal{Z}$ is an intermediate representation (i.e., latent representation) of the original input image $\mathbf{I}_{\mathrm{WB^{(in)}}}$.

Our goal is to make the functions f and g_t independent, such that changing g_t with a new function g_y that targets a different WB y does not require any modification in f, as is the case in Eq. 1.

In our work, we target three different WB settings: (i) $\mathrm{WB^{(A)}}$: AWB—representing the correct lighting of the captured image's scene; (ii) $\mathrm{WB^{(T)}}$: Tungsten/Incandescent—representing WB for indoor lighting; and (iii) $\mathrm{WB^{(S)}}$: Shade—representing WB for outdoor lighting. This gives rise to three different decoders (g_A, g_T, and g_S) that are responsible for generating output images that correspond to AWB, Incandescent WB, and Shade WB.

The Incandescent and Shade WB are specifically selected based on the color properties. This can be understood when considering the illuminations in terms of their correlated color temperatures. For example, Incandescent and Shade WB settings are correlated to 2850 Kelvin (K) and $7500K$ color temperatures, respectively. This wide range of illumination color temperatures considers the range of pleasing illuminations [26, 27]. Moreover, the wide color temperature range between Incandescent and Shade allows the approximation of images with color temperatures within this range by interpolation. The details of this interpolation

Figure 3: We consider the runtime performance of our method to be able to run on limited computing resources (∼1.5 seconds on a single CPU to process a 12-megapixel image). First, our DNN processes a downsampled version of the input image, and then we apply a global color mapping to produce the output image in its original resolution. The shown input image is rendered from the MIT-Adobe FiveK dataset [11].

process are explained in Sec. 2.5. Note that there is no fixed correlated color temperature for the AWB mode, as it changes based on the input image's lighting conditions.

2.3. Multi-decoder architecture

An overview of our DNN's architecture is shown in Fig. 2. We use a U-Net architecture [29] with multi-scale skip connections between the encoder and decoders. Our framework consists of two main units: the first is a 4-level encoder unit that is responsible for extracting a multi-scale latent representation of our input image; the second unit includes three 4-level decoders. Each unit has a different bottleneck and transposed convolutional (conv) layers. At the first level of our encoder and each decoder, the conv layers have 24 channels. For each subsequent level, the number of channels is doubled (i.e., the fourth level has 192 channels for each conv layer).

2.4. Training phase

Training data We adopt the Rendered WB dataset produced by [3] to train and validate our model. This dataset includes ∼65K sRGB images rendered by different camera models and with different WB settings, including the Shade and Incandescent settings. For each image, there is also a corresponding ground truth image rendered with the correct WB setting (considered to be the correct AWB result). This dataset consists of two subsets: Set 1 (62,535 images taken by seven different DSLR cameras) and Set 2 (2,881 images taken by a DSLR camera and four mobile phone cameras). The first set (i.e., Set 1) is divided into three equal partitions by [3]. We randomly selected 12,000 training images from the first two partitions of Set 1 to train our model. For each training image, we have three ground truth images rendered with: (i) the correct WB (denoted as AWB), (ii) Shade WB, and (iii) Incandescent WB. The final partition of Set 1 (21,046 images) is used for testing. We refer to this partition as Set 1–*Test*. Images of Set 2 are not used in training and the entire set is used for testing.

Data augmentation We also augment the training images by rendering an additional 1,029 raw-RGB images, of the same scenes included in the Rendered WB dataset [3], but
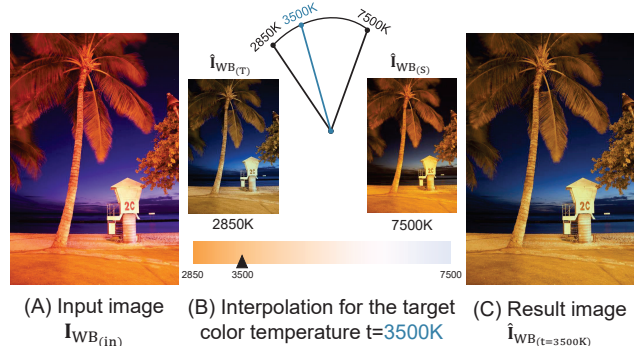

Figure 4: In addition to our AWB correction, we train our framework to produce two different color temperatures (i.e., Incandescent and Shade WB settings). We interpolate between these settings to produce images with other color temperatures. (A) Input image. (B) Interpolation process. (C) Final result. The shown input image is taken from the rendered version of the MIT-Adobe FiveK dataset [3, 11].

with random color temperatures. At each epoch, we randomly select four 128×128 patches from each training image and their corresponding ground truth images for each decoder and apply geometric augmentation (rotation and flipping) as an additional data augmentation to avoid overfitting.

Loss function We trained our model to minimize the L_1-norm loss function between the reconstructed and ground truth patches:

$$\sum_i \sum_{p=1}^{3hw} |\mathbf{P}_{\text{WB}^{(i)}}(p) - \mathbf{C}_{\text{WB}^{(i)}}(p)|, \qquad (3)$$

where h and w denote the patch's height and width, and p indexes into each pixel of the training patch $\mathbf{P}$ and the ground truth camera-rendered patch $\mathbf{C}$, respectively. The index $i \in \{A, T, S\}$ refers to the three target WB settings. We also have examined the squared L_2-norm loss function and found that both loss functions work well for our task. See supplemental materials for an ablation study.

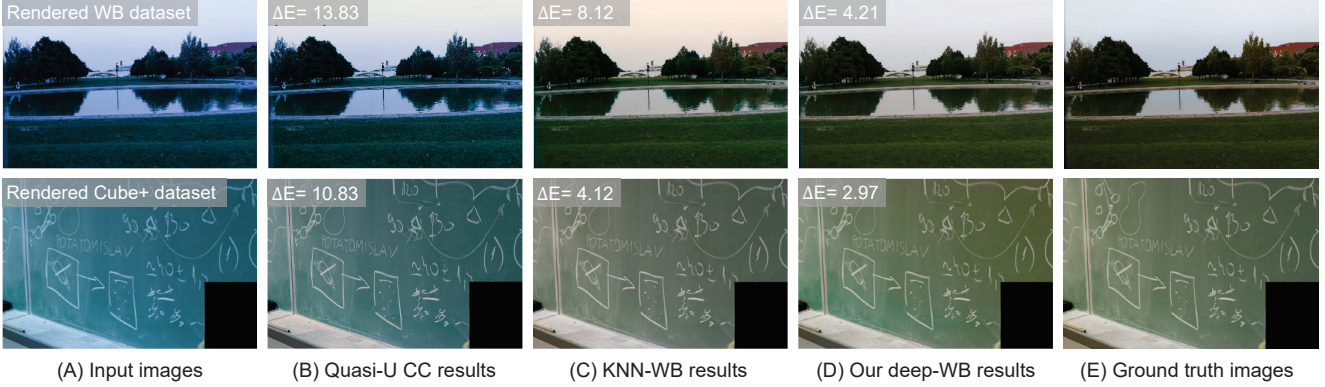| Rendered WB dataset | ΔE= 13.83 | ΔE= 8.12 | ΔE= 4.21 | |
| Rendered Cube+ dataset | ΔE= 10.83 | ΔE= 4.12 | ΔE= 2.97 | |
| (A) Input images | (B) Quasi-U CC results | (C) KNN-WB results | (D) Our deep-WB results | (E) Ground truth images |

Figure 5: Qualitative comparison of AWB correction. (A) Input images. (B) Results of quasi-U CC [9]. (C) Results of KNN-WB [3]. (D) Our results. (E) Ground truth images. Shown input images are taken from the Rendered WB dataset [3] and the rendered version of Cube+ dataset [3, 5].

Training parameters We initialized the weights of the conv layers using He's initialization [20]. The training process is performed for 165,000 iterations using the adaptive moment estimation (Adam) optimizer [25], with a decay rate of gradient moving average $\beta_1 = 0.9$ and a decay rate of squared gradient moving average $\beta_2 = 0.999$. We used a learning rate of 10^{-4} and reduced it by 0.5 every 25 epochs. The mini-batch size was 32 training patches per iteration.

2.5. Testing phase

Color mapping procedure Our DNN model is a fully convolutional network and is able to process input images in their original dimensions with the restriction that the dimensions should be multiples of 2^4, as we use 4-level encoder/decoders with 2×2 max-pooling and transposed conv layers. However, to ensure a consistent run time for any sized input images, we resize all input images to a maximum dimension of 656 pixels. Our DNN is applied on this resized image to produce image $\hat{\mathbf{I}}_{\text{WB}(i)\downarrow}$ with the target WB setting $i \in \{A, T, S\}$.

We then compute a color mapping function between our resized input and output image. Work in [16, 21] evaluated several types of polynomial mapping functions and showed their effectiveness to achieve nonlinear color mapping. Accordingly, we computed a polynomial mapping matrix $\mathbf{M}$ that globally maps values of $\psi\left(\mathbf{I}_{\text{WB}(\text{in})\downarrow}\right)$ to the colors of our generated image $\hat{\mathbf{I}}_{\text{WB}(i)\downarrow}$, where $\psi(\cdot)$ is a polynomial kernel function that maps the image's RGB vectors to a higher 11-dimensional space—see supplemental materials for an evaluation of different kernel functions. This mapping matrix $\mathbf{M}$ can be computed in a closed-form solution, as demonstrated in [2, 3].

Once $\mathbf{M}$ is computed, we obtain our final result in the same input image resolution using the following equation [3]:

$$\hat{\mathbf{I}}_{\text{WB}(i)} = \mathbf{M}\psi\left(\mathbf{I}_{\text{WB}(\text{in})}\right). \qquad (4)$$

Fig. 3 illustrates our color mapping procedure. Our method requires ~1.5 seconds on an Intel Xeon E5-1607 @ 3.10GHz machine with 32 GB RAM to process a 12-megapixel image for a selected WB setting.

We note that an alternative strategy is to compute the color polynomial mapping matrix directly [30]. We conducted preliminary experiments and found that estimating the polynomial matrix directly was less robust than generating the image itself followed by fitting a global polynomial function. The reason is that having small errors in the estimated polynomial coefficients can lead to noticeable color errors (e.g., out-of-gamut values), whereas small errors in the estimated image were ameliorated by the global fitting.

Editing by user manipulation Our framework allows the user to choose between generating result images with the three available WB settings (i.e., AWB, Shade WB, and Incandescent WB). Using the Shade and Incandescent WB, the user can edit the image to a specific WB setting in terms of color temperature, as explained in the following.

To produce the effect of a *new* target WB setting with a color temperature t that is not produced by our decoders, we can interpolate between our generated images with the Incandescent and Shade WB settings. We found that a simple linear interpolation was sufficient for this purpose. This operation is described by the following equation:

$$\hat{\mathbf{I}}_{\text{WB}(t)} = b\,\hat{\mathbf{I}}_{\text{WB}(T)} + (1-b)\,\hat{\mathbf{I}}_{\text{WB}(S)}, \qquad (5)$$

where $\hat{\mathbf{I}}_{\text{WB}(T)}$ and $\hat{\mathbf{I}}_{\text{WB}(S)}$ are our produced images with Incandescent and Shade WB settings, respectively, and b is the interpolation ratio that is given by $\frac{1/t - 1/t(S)}{1/t(T) - 1/t(S)}$. Fig. 4 shows an example.

3. Results

Our method targets two different tasks: post-capture WB correction and manipulation of the sRGB rendered images

Table 1: AWB results using the Rendered WB dataset [3] and the rendered version of the Cube+ dataset [3,5]. 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 [32]. For all diagonal-based methods, gamma linearization [4,15] is applied. The top results are indicated with yellow and boldface.

Method	MSE				MAE				△E 2000			
	Mean	Q1	Q2	Q3	Mean	Q1	Q2	Q3	Mean	Q1	Q2	Q3
Rendered WB dataset: Set 1–*Test* (21,046 images) [3]												
FC4 [23]	179.55	33.89	100.09	246.50	6.14°	2.62°	4.73°	8.40°	6.55	3.54	5.90	8.94
Quasi-U CC [9]	172.43	33.53	97.9	237.26	6.00°	2.79°	4.85°	8.15°	6.04	3.24	5.27	8.11
KNN-WB [3]	**77.79**	13.74	**39.62**	**94.01**	**3.06°**	**1.74°**	**2.54°**	**3.76°**	**3.58**	**2.07**	**3.09**	**4.55**
Ours	82.55	**13.19**	42.77	102.09	3.12°	1.88°	2.70°	3.84°	3.77	2.16	3.30	4.86
Rendered WB dataset: Set 2 (2,881 images) [3]												
FC4 [23]	505.30	142.46	307.77	635.35	10.37°	5.31°	9.26°	14.15°	10.82	7.39	10.64	13.77
Quasi-U CC [9]	553.54	146.85	332.42	717.61	10.47°	5.94°	9.42°	14.04°	10.66	7.03	10.52	13.94
KNN-WB [3]	171.09	37.04	87.04	190.88	4.48°	2.26°	3.64°	5.95°	5.60	3.43	4.90	7.06
Ours	**124.97**	**30.13**	**76.32**	**154.44**	**3.75°**	**2.02°**	**3.08°**	**4.72°**	**4.90**	**3.13**	**4.35**	**6.08**
Rendered Cube+ dataset with different WB settings (10,242 images) [3,5]												
FC4 [23]	371.9	79.15	213.41	467.33	6.49°	3.34°	5.59°	8.59°	10.38	6.6	9.76	13.26
Quasi-U CC [9]	292.18	15.57	55.41	261.58	6.12°	1.95°	3.88°	8.83°	7.25	2.89	5.21	10.37
KNN-WB [3]	194.98	27.43	57.08	118.21	4.12°	1.96°	3.17°	5.04°	5.68	3.22	4.61	6.70
Ours	**80.46**	**15.43**	**33.88**	**74.42**	**3.45°**	**1.87°**	**2.82°**	**4.26°**	**4.59**	**2.68**	**3.81**	**5.53**

to a specific WB color temperature. We achieve state-of-the-art results for several different datasets for both tasks. We first describe the datasets used to evaluate our method in Sec. 3.1. We then discuss our quantitative and qualitative results in Sec. 3.2 and Sec. 3.3, respectively. We also perform an ablation study to validate our problem formulation and the proposed framework.

(A) Input images   (B) KNN-WB emulator results   (C) Our deep-WB results   (D) Target camera WB
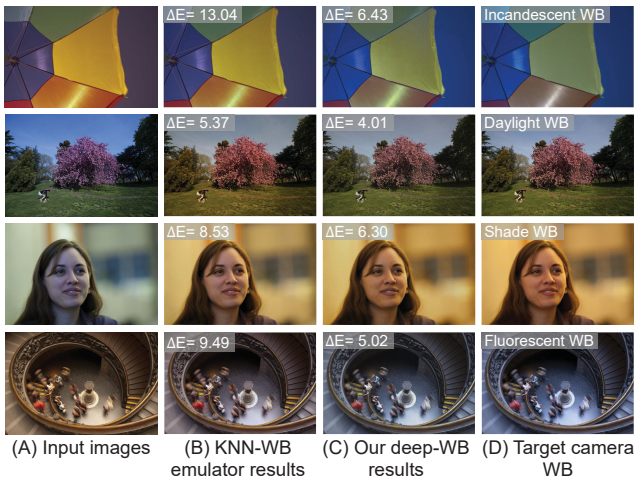
Figure 6: Qualitative comparison of WB manipulation. (A) Input images. (B) Results of KNN-WB emulator [2]. (C) Our results. (D) Ground truth camera-rendered images with the target WB settings. In this figure, the target WB settings are Incandescent, Daylight, Shade, and Fluorescent. Shown input images are taken from the rendered version of the MIT-Adobe FiveK dataset [3,11].

3.1. Datasets

As previously mentioned, we used randomly selected images from the two partitions of Set 1 in the Rendered WB dataset [3] for training. For testing, we used the third partition of Set 1, termed Set 1-*Test*, and three additional datasets not part of training. Two of these additional datasets are as follows: (1) Set 2 of the Rendered WB dataset (2,881 images) [3], and (2) the sRGB rendered version of the Cube+ dataset (10,242 images) [5]. Datasets (1) and (2) are used to evaluate the task of AWB correction. For the WB manipulation task, we used the rendered Cube+ dataset and (3) a rendered version of the MIT-Adobe FiveK dataset (29,980 images) [11]. The rendered version of each dataset of these datasets is available from the project page associated with [3]. These latter datasets represent raw-RGB images that have been rendered to the sRGB color space with *different* WB settings. This allows us to evaluate how well we can mimic different WB settings.

3.2. Quantitative results

For both tasks, we follow the same evaluation metrics used by the most recent work in [3]. Specifically, we used the following metrics to evaluate our results: mean square error (MSE), mean angular error (MAE), and △E 2000 [32]. We also report △E 76 in the supplemental materials. For each evaluation metric, we report the mean, lower quartile (Q1), median (Q2), and the upper quartile (Q3) of the error.

WB correction   We compared the proposed method with the KNN-WB approach in [3]. We also compared our

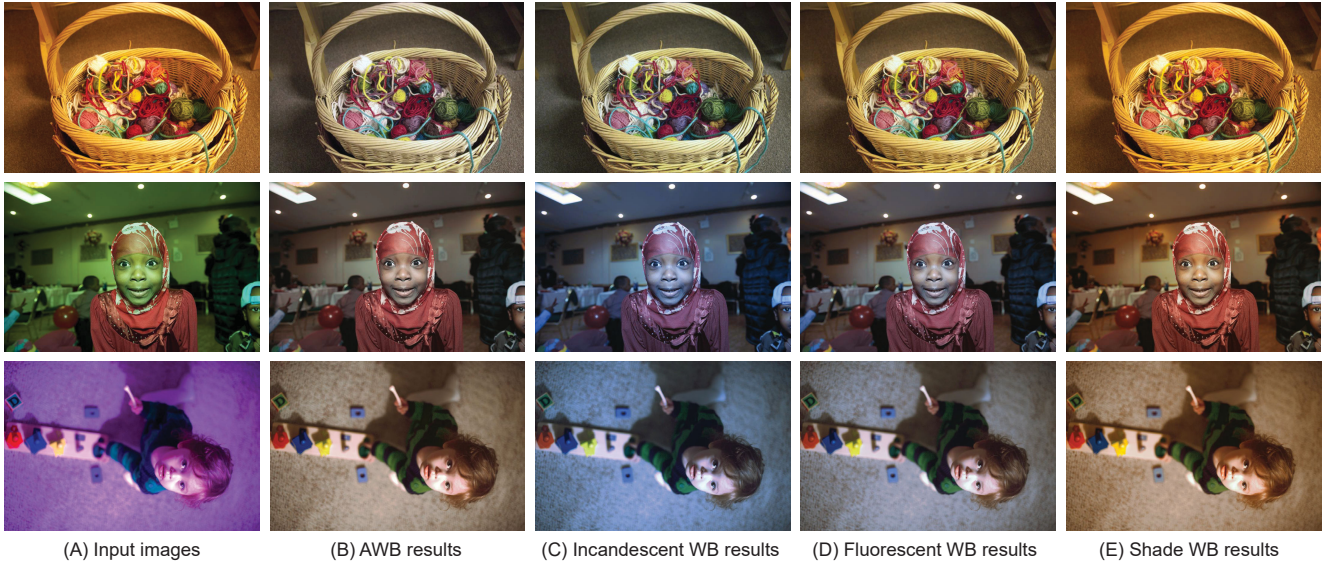| (A) Input images | (B) AWB results | (C) Incandescent WB results | (D) Fluorescent WB results | (E) Shade WB results |

Figure 7: Qualitative results of our method. (A) Input images. (B) AWB results. (C) Incandescent WB results. (D) Fluorescent WB results. (E) Shade WB Results. Shown input images are rendered from the MIT-Adobe FiveK dataset [11].

| (A) Input image | (B) Quasi-U CC | (C) KNN-WB | (D) Our AWB correction | (E) Our Incandescent WB | (F) Our Shade WB |

Figure 8: (A) Input image. (B) Result of quasi-U CC [9]. (C) Result of KNN-WB [3]. (D)-(F) Our deep-WB editing results. Photo credit: *Duncan Yoyos* Flickr–CC BY-NC 2.0.

results against the traditional WB diagonal-correction using recent illuminant estimation methods [9, 23]. We note that methods [9, 23] were not designed to correct nonlinear sRGB images. These methods are included, because it is often purported that such methods are effective when the sRGB image has been "linearized" using a decoding gamma.

Table 1 reports the error between corrected images obtained by each method and the corresponding ground truth images. Table 1 shows results on the Set 1-*Test*, Set 2, and Cube+ dataset described earlier. This represents a total of 34,169 unseen sRGB images by our DNN-model, each rendered with different camera models and WB settings. For the diagonal-correction results, we pre-processed each testing image by first applying the 2.2 gamma linearization [4, 15], and then we applied the gamma encoding after correction. We have results that are on par with the state-of-the-art method [3] on the Set 1–*Test*. We achieve state-of-the-art results in all evaluation metrics for additional testing sets (Set 2 and Cube+).

WB manipulation The goal of this task is to change the input image's colors to appear as they were rendered using a target WB setting. We compare our result with the most

recent work in [2] that proposed a KNN-WB emulator that mimics WB effects in the sRGB space. We used the same WB settings produced by the KNN-WB emulator. Specifically, we selected the following target WB settings: Incandescent ($2850K$), Fluorescent ($3800K$), Daylight ($5500K$), Cloudy ($6500K$), and Shade ($7500K$). As our decoders were trained to generate only Incandescent and Shade WB settings, we used Eq. 5 to produce the other WB settings (i.e., Fluorescent, Daylight, and Cloudy WB settings).

Table 2 shows the obtained results using our method and the KNN-WB emulator. Table 2 demonstrates that our method outperforms the KNN-WB emulator [2] over a total of 40,222 testing images captured with different camera models and WB settings using all evaluation metrics.

3.3. Qualitative results

In Fig. 5 and Fig. 6, we provide a visual comparison of our results against the most recent work proposed for WB correction [3, 9] and WB manipulation [2], respectively. On top of each example, we show the $\triangle$E 2000 error between the result image and the corresponding ground truth image (i.e., rendered by the camera using the target setting). It is clear that our results have the lower $\triangle$E 2000 and are the most similar to the ground truth images.

Table 2: Results of WB manipulation using the rendered version of the Cube+ dataset [3, 5] and the rendered version of the MIT-Adobe FiveK dataset [3, 11]. 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 [32]. The top results are indicated with yellow and boldface.

Method	MSE				MAE				△E 2000			
	Mean	Q1	Q2	Q3	Mean	Q1	Q2	Q3	Mean	Q1	Q2	Q3
Rendered Cube+ dataset (10,242 images) [3, 5]												
KNN-WB emulator [2]	317.25	50.47	153.33	428.32	7.6°	3.56°	6.15°	10.63°	7.86	4.00	6.56	10.46
Ours	**199.38**	**32.30**	**63.34**	**142.76**	**5.40°**	**2.67°**	**4.04°**	**6.36°**	**5.98**	**3.44**	**4.78**	**7.29**
Rendered MIT-Adobe FiveK dataset (29,980 images) [3, 11]												
KNN-WB emulator [2]	249.95	41.79	109.69	283.42	7.46°	3.71°	6.09°	9.92°	6.83	3.80	5.76	8.89
Ours	**135.71**	**31.21**	**68.63**	**151.49**	**5.41°**	**2.96°**	**4.45°**	**6.83°**	**5.24**	**3.32**	**4.57**	**6.41**

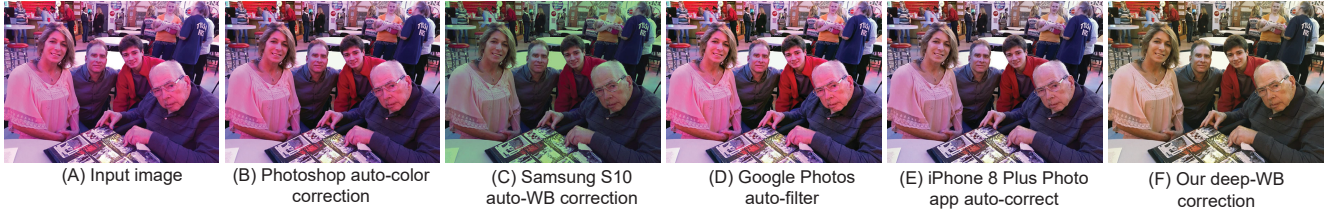| (A) Input image | (B) Photoshop auto-color correction | (C) Samsung S10 auto-WB correction | (D) Google Photos auto-filter | (E) iPhone 8 Plus Photo app auto-correct | (F) Our deep-WB correction |

Figure 9: Strong color casts due to WB errors are hard to correct. (A) Input image rendered with an incorrect WB setting. (B) Result of Photoshop auto-color correction. (C) Result of Samsung S10 auto-WB correction. (D) Result of Google Photos auto-filter. (E) Result of iPhone 8 Plus built-in Photo app auto-correction. (F) Our AWB result using the proposed deep-WB editing framework. Photo credit: *OakleyOriginals* Flickr–CC BY 2.0.

Fig. 7 shows additional examples of our results. As shown, our framework accepts input images with arbitrary WB settings and re-renders them with the target WB settings, including the AWB correction.

We tested our method with several images taken from the Internet to check its ability to generalize to images typically found online. Fig. 8 and Fig. 9 show examples. As is shown, our method produces compelling results compared with other methods and commercial software packages for photo editing, even when input images have strong color casts. We provide additional examples in supplemental materials.

3.4. Comparison with a vanilla U-Net

As explained earlier, our framework employs a single encoder to encode input images, while each decoder is responsible for producing a specific WB setting. Our architecture aims to model Eq. 1 in the same way cameras would produce colors for different WB settings from the same raw-RGB captured image.

Intuitively, we can re-implement our framework using a multi-U-Net architecture [29], such that each encoder/decoder model will be trained for a single target of the WB settings.

In Table 3, we provide a comparison between our proposed framework against vanilla U-Net models. We train our proposed architecture and three U-Net models (each U-Net model targets one of our WB settings) for 88,000 iterations. The results validate our design and make evident that

Table 3: Average of mean square error and △E 2000 [32] obtained by our framework and the traditional U-Net architecture [29]. Shown results on Set 2 of the Rendered WB dataset [3] for AWB and the rendered version of the Cube+ dataset [3, 5] for WB manipulation. The top results are indicated with yellow and boldface.

Method	AWB [3]		WB editing [3, 5]	
	MSE	△E 2000	MSE	△E 2000
Multi-U-Net [29]	187.25	6.23	234.77	6.87
Ours	**124.47**	**4.99**	**206.81**	**6.23**

our shared encoder not only reduces the required number of parameters but also gives better results.

4. Conclusion

We have presented a deep learning framework for editing the WB of sRGB camera-rendered images. Specifically, we have proposed a DNN architecture that uses a single encoder and multiple decoders, which are trained in an end-to-end manner. Our framework allows the direct correction of images captured with wrong WB settings. Additionally, our framework produces output images that allow users to manually adjust the sRGB image to appear as if it was rendered with a wide range of WB color temperatures. Quantitative and qualitative results demonstrate the effectiveness of our framework against recent data-driven methods.

References

[1] Mahmoud Afifi and Michael S Brown. Sensor-independent illumination estimation for dnn models. In *BMVC*, 2019. 2

[2] Mahmoud Afifi and Michael S Brown. What else can fool deep learning? Addressing color constancy errors on deep neural network performance. In *ICCV*, 2019. 2, 5, 6, 7, 8

[3] 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, 7, 8

[4] Matthew Anderson, Ricardo Motta, Srinivasan Chandrasekar, and Michael Stokes. Proposal for a standard default color space for the Internet - sRGB. In *Color and Imaging Conference*, pages 238–245, 1996. 6, 7

[5] Nikola Banić and Sven Lončarić. Unsupervised learning for color constancy. *arXiv preprint arXiv:1712.00436*, 2017. 5, 6, 8

[6] Kobus Barnard, Vlad Cardei, and Brian Funt. A comparison of computational color constancy algorithms: Methodology and experiments with synthesized data. *IEEE Transactions on Image Processing*, 11(9):972–984, 2002. 2

[7] Jonathan T Barron. Convolutional color constancy. In *ICCV*, 2015. 2

[8] Jonathan T Barron and Yun-Ta Tsai. Fast Fourier color constancy. In *CVPR*, 2017. 1, 2

[9] Simone Bianco and Claudio Cusano. Quasi-unsupervised color constancy. In *CVPR*, 2019. 2, 5, 6, 7

[10] Gershon Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin Institute*, 310(1):1–26, 1980. 2

[11] 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. 4, 6, 7, 8

[12] Ayan Chakrabarti, Ying Xiong, Baochen Sun, Trevor Darrell, Daneil Scharstein, Todd Zickler, and Kate Saenko. Modeling radiometric uncertainty for vision with tone-mapped color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2185–2198, 2014. 2

[13] Dongliang Cheng, Abdelrahman Kamel, Brian Price, Scott Cohen, and Michael S Brown. Two illuminant estimation and user correction preference. In *CVPR*, 2016. 1

[14] Dongliang Cheng, Dilip K Prasad, and Michael S Brown. Illuminant estimation for color constancy: Why spatial-domain methods work and the role of the color distribution. *JOSA A*, 31(5):1049–1058, 2014. 2

[15] Marc Ebner. *Color Constancy*, volume 6. John Wiley & Sons, 2007. 6, 7

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

[17] Graham D Finlayson and Elisabetta Trezzi. Shades of gray and colour constancy. In *Color and Imaging Conference*, 2004. 2

[18] Peter V Gehler, Carsten Rother, Andrew Blake, Tom Minka, and Toby Sharp. Bayesian color constancy revisited. In *CVPR*, 2008. 2

[19] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Computational color constancy: Survey and experiments. *IEEE Transactions on Image Processing*, 20(9):2475–2489, 2011. 2

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 5

[21] 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. 5

[22] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG)*, 37(2):26:1–26:17, 2018. 1

[23] Yuanming Hu, Baoyuan Wang, and Stephen Lin. FC4: Fully convolutional color constancy with confidence-weighted pooling. In *CVPR*, 2017. 2, 6, 7

[24] Seon Joo Kim, Hai Ting Lin, Zheng Lu, Sabine Süsstrunk, Stephen Lin, and Michael S Brown. A new in-camera imaging model for color computer vision and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2289–2302, 2012. 2

[25] Diederik P Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[26] Arie Andries Kruithof. Tubular luminescence lamps for general illumination. *Philips Technical Review*, 6:65–96, 1941. 3

[27] Andrius Petrulis, Linas Petkevičius, Pranciškus Vitta, Rimantas Vaicekauskas, and Artūras Žukauskas. Exploring preferred correlated color temperature in outdoor environments using a smart solid-state light engine. *The Journal of the Illuminating Engineering Society*, 14(2):95–106, 2018. 3

[28] Yanlin Qian, Joni-Kristian Kamarainen, Jarno Nikkanen, and Jiri Matas. On finding gray pixels. In *CVPR*, 2019. 2

[29] 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. 4, 8

[30] Eli Schwartz, Raja Giryes, and Alex M Bronstein. DeepISP: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, 28(2):912–923, 2018. 5

[31] Michael Scuello, Israel Abramov, James Gordon, and Steven Weintraub. Museum lighting: Why are some illuminants preferred? *JOSA A*, 21(2):306–311, 2004. 1

[32] 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. 6, 8

[33] Wu Shi, Chen Change Loy, and Xiaoou Tang. Deep specialized network for illuminant estimation. In *ECCV*, 2016. 2

[34] Ying Xiong, Kate Saenko, Trevor Darrell, and Todd Zickler. From pixels to physics: Probabilistic color de-rendering. In *CVPR*, 2012. 2