Cross-Camera Convolutional Color Constancy

Mahmoud Afifi[1,2*] Jonathan T. Barron[1] Chloe LeGendre[1] Yun-Ta Tsai[1] Francois Bleibel[1]

[1]Google Research [2]York University

Abstract

We present "Cross-Camera Convolutional Color Constancy" (C5), a learning-based method, trained on images from multiple cameras, that accurately estimates a scene's illuminant color from raw images captured by a new camera previously unseen during training. C5 is a hypernetwork-like extension of the convolutional color constancy (CCC) approach: C5 learns to generate the weights of a CCC model that is then evaluated on the input image, with the CCC weights dynamically adapted to different input content. Unlike prior cross-camera color constancy models, which are usually designed to be agnostic to the spectral properties of test-set images from unobserved cameras, C5 approaches this problem through the lens of transductive inference: additional unlabeled images are provided as input to the model at test time, which allows the model to calibrate itself to the spectral properties of the test-set camera during inference. C5 achieves state-of-the-art accuracy for cross-camera color constancy on several datasets, is fast to evaluate (∼7 and ∼90 ms per image on a GPU or CPU, respectively), and requires little memory (∼2 MB), and thus is a practical solution to the problem of calibration-free automatic white balance for mobile photography.

1. Introduction

The goal of computational color constancy is to emulate the human visual system's ability to constantly perceive object colors even when they are observed under different illumination conditions. In many contexts, this problem is equivalent to the practical problem of automatic white balance—removing an undesirable global color cast caused by the illumination in the scene, thereby, making it appear to have been imaged under a white light (see Figure 1). White balance does not only affect the quality of photographs but also has an impact on the accuracy of different computer vision tasks [3]. On modern digital cameras, automatic white balance is performed for all captured images as an essential part of the camera's imaging pipeline.

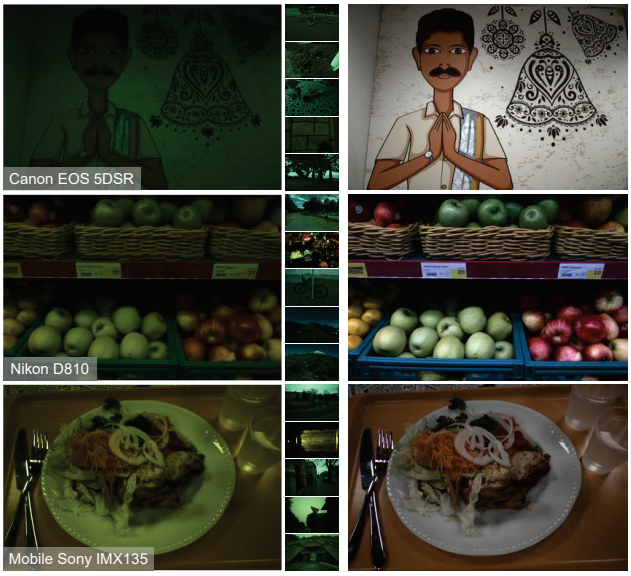

Figure 1: Our C5 model exploits the colors of unlabeled additional images captured by the new camera model to generate a specific color constancy model for the input image. These additional images can be *randomly* loaded from the photographer's "camera roll", or they could be a fixed set taken once by the camera manufacturer. The shown images were captured by *unseen* DSLR and smartphone camera models [38] that were not included in the training stage.

Color constancy is a challenging problem, because it is fundamentally under-constrained: an infinite family of white-balanced images and global color casts can explain the same observed image. Color constancy is, therefore, often framed in terms of inferring the most likely illuminant color given some observed image and some prior knowledge of the spectral properties of the camera's sensor.

One simple heuristic applied to the color constancy problem is the "gray-world" assumption: that colors in the world tend to be neutral gray and that the color of the illuminant can, therefore, be estimated as the average color of the input image [14]. This gray-world method and its related tech-

*This work was done while Mahmoud was an intern at Google.

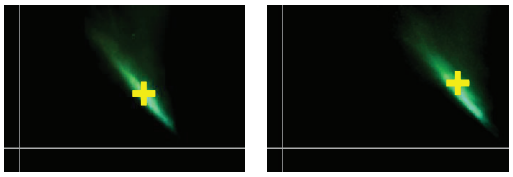

Figure 2: A visualization of uv log-chroma histograms ($u = \log(g/r)$, $v = \log(g/b)$) of images from two different cameras averaged over many images of the same scene set in the NUS dataset [15] (shown in green), as well as the uv coordinate of the mean of ground-truth illuminants over the entire scene set (shown in yellow). The "positions" of these histograms change significantly across the two camera sensors because of their different spectral sensitivities, which is why many color constancy models generalize poorly across cameras.

niques have the convenient property that they are *invariant* to much of the spectral sensitivity differences among camera sensors and, therefore, very well-suited to the cross-camera task. If camera A's red channel is twice as sensitive as camera B's red channel, then a scene captured by camera A will have an average red intensity that is twice that of the scene captured by camera B, and so gray-world will produce identical output images (though this assumes that the spectral response of A and B are identical up to a scale factor, which is rarely the case in practice). However, current state-of-the-art learning-based methods for color constancy rarely exhibit this property, because they often learn things like the precise distribution of likely illuminant colors (a consequence of black-body illumination and other scene lighting regularities) and are, therefore, sensitive to any mismatch between the spectral sensitivity of the camera used during training and that of the camera used at test time [2].

Because there is often significant spectral variation across camera models (as shown in Figure 2), this sensitivity of existing methods is problematic when designing practical white-balance solutions. Training a learning-based algorithm for a new camera requires collecting hundreds, or thousands, of images with ground-truth illuminant color labels (in practice: images containing a color chart), a burdensome task for a camera manufacturer or platform that may need to support hundreds of different camera models. However, the gray-world assumption still holds surprisingly well across sensors—if given several images from a particular camera, one can do a reasonable job of estimating the range of likely illuminant colors (as can also be seen in Figure 2).

In this paper, we propose a camera-independent color constancy method. Our method achieves high-accuracy cross-camera color constancy through the use of two concepts: First, our system is constructed to take as input not just a single test-set image, but also a small set of additional images from the test set, which are: (i) arbitrarily-selected, (ii) unlabeled, (iii) and not white balanced. This allows the model to calibrate itself to the spectral properties of the test-time camera during inference. We make *no assumptions* about these additional images except that they come from the same camera as the "target" test set image and they contain some content (not all black or white images). In practice, these images could simply be *randomly* chosen images from the photographer's "camera roll", or they could be a fixed set of ad hoc images of natural scenes taken once by the camera manufacturer—because these images do not need to be annotated, they are abundantly available. Second, our system is constructed as a *hypernetwork* [28] around an existing color constancy model. The target image and the additional images are used as input to a deep neural network whose output is the weights of a smaller color constancy model, and those generated weights are then used to estimate the illuminant color of the target image.

Our system is trained using labeled (and unlabeled) images from multiple cameras, but at test time our model is able to look at a set of (unlabeled) test set images from a new camera. Our hypernetwork is able to infer the likely spectral properties of the new camera that produced the test set images (much as the reader can infer the likely illuminant colors of a camera from only looking at aggregate statistics, as in Figure 2) and produce a small model that has been dynamically adapted to produce accurate illuminant estimates when applied to the target image. Our method is computationally fast and requires a low memory footprint while achieving state-of-the-art results compared to other camera-independent color constancy methods.

2. Prior Work

There is a large body of literature proposed for illuminant color estimation, which can be categorized into statistical-based methods (e.g., [13–15,20,26,34,47,51,54]) and learning-based methods (e.g., [8,9,11,12,19,21,24,25, 31, 42, 44, 45, 49, 52, 60]). The former rely on statistical-based hypotheses to estimate scene illuminant colors based on the color distribution and/or spatial layout of the input raw image. Such methods are usually simple and efficient, but they are less accurate than the learning-based alternatives.

Learning-based methods, on the other hand, are typically trained for a single target camera model in order to learn the distribution of illuminant colors produced by the target camera's particular sensor [2, 23, 37]. The learning-based methods are typically constrained to the specific, single camera use-case, as the spectral sensitivity of each camera sensor significantly alters the recorded illuminant and scene colors, and different sensor spectral sensitivities change the illuminant color distribution for the same set of scenes [32, 58]. Such camera-specific methods cannot accurately extrapo-

late beyond the learned distribution of the training camera model's illuminant colors [2, 47] without tuning/re-training or pre-calibration [39].

Recently, few-shot and multi-domain learning techniques [44, 59] have been proposed to reduce the effort of re-training camera-specific learned color constancy models. These methods require only a small set of labeled images for a new camera unseen during training. In contrast, our technique requires no ground-truth labels for the unseen camera, and is essentially calibration-free for this new sensor.

Another strategy has been proposed to white balance the input image with several illuminant color candidates and learn the likelihood of properly white-balanced images [29]. Such a Bayesian framework requires prior knowledge of the target camera model's illuminant colors to build the illuminant candidate set. Despite promising results, these methods, however, all require labeled training examples from the target camera model: raw images paired with ground-truth illuminant colors. Collecting such training examples is a tedious process, as certain conditions must be satisfied—i.e., for each image to have a single uniform lighting and a calibration object to be present in the scene [15].

An additional class of work has sought to learn *sensor-independent* color constancy models, circumventing the need to re-train or calibrate to a specific camera model. A recent quasi-unsupervised approach to color constancy has been proposed, which learns the semantic features of achromatic objects to help build a model robust to differing camera sensor spectral sensitivities [10]. Another technique proposes to learn an intermediate "device independent" space before the illuminant estimation process [2]. The goal of our method is similar, in that we also propose to learn a color constancy model that works for all cameras, but neither of these previous sensor-independent approaches leverages multiple test images to reason about the spectral properties of the unseen camera model. This enables our method to outperform these state-of-the-art sensor-independent methods across diverse test sets.

Though not commonly applied in color constancy techniques, our proposal to use multiple test-set images at inference-time to improve performance is a well-explored approach across machine learning. The task of classifying an entire test set as accurately as possible was first described by Vapnik as "transductive inference" [33, 55]. Our approach is also closely related to the work on domain adaptation [17, 50] and transfer learning [46], both of which attempt to enable learning-based models to cope with differences between training and test data. Multiple sRGB camera-rendered images of the same scene have been used to estimate the response function of a given camera in the radiometric calibration literature [27, 35]. In our method, however, we employ additional images to learn to extract informative cues about the spectral sensitivity of the camera capturing the input test image, without needing to capture the same scene multiple times.

3. Method

We call our system "cross-camera convolutional color constancy" (C5), because it builds upon the existing "convolutional color constancy" (CCC) model [8] and its successor "fast Fourier color constancy" (FFCC) [9], but embeds them in a multi-input hypernetwork to enable accurate cross-camera performance. These CCC/FFCC models work by learning to perform localization within a log-chroma histogram space, such as those shown in Figure 2.

Here, we present a convolutional color constancy model that is a simplification of those presented in the original work [8] and its FFCC follow-up [9]. This simple convolutional model will be a fundamental building block that we will use in our larger neural network. The image formation model behind CCC/FFCC (and most color constancy models) is that each pixel of the observed image is assumed to be the element-wise product of some "true" white-balanced image (or equivalently, the observed image if it were imaged under a white illuminant) and some illuminant color:

$$\forall_k \, \mathbf{c}^{(k)} = \mathbf{w}^{(k)} \circ \boldsymbol{\ell} \,, \tag{1}$$

where $\mathbf{c}^{(k)}$ is the observed color of pixel k, $\mathbf{w}^{(k)}$ is the true color of the pixel, and $\boldsymbol{\ell}$ is the color of the illuminant, all of which are 3-vectors of RGB values. Color constancy algorithms traditionally use the input image $\{\mathbf{c}^{(k)}\}$ to produce an estimate of the illuminant $\hat{\boldsymbol{\ell}}$ that is then divided (element-wise) into each observed color to produce an estimate of the true color of each pixel, $\{\hat{\mathbf{w}}^{(k)}\}$.

CCC defines two log-chroma measures for each pixel, which are simply the log of the ratio of two color channels:

$$u^{(k)} = \log\left(c_g^{(k)}/c_r^{(k)}\right), \quad v^{(k)} = \log\left(c_g^{(k)}/c_b^{(k)}\right). \tag{2}$$

As noted by Finlayson, this log-chrominance representation of color means that illuminant changes (i.e. element-wise scaling by $\boldsymbol{\ell}$) can be modeled simply as additive offsets to this uv representation [18]. We then construct a 2D histogram of the log-chroma values of all pixels:

$$N_0(u,v) = \sum_k ||\mathbf{c}^{(k)}||_2 \left[\left|u^{(k)} - u\right| \leq \epsilon \wedge \left|v^{(k)} - v\right| \leq \epsilon\right]. \tag{3}$$

This is simply a histogram over all uv coordinates of size (64×64) written out using Iverson brackets, where ϵ is the width of a histogram bin, and where each pixel is weighted by its overall brightness under the assumption that bright pixels provide more actionable signal than dark pixels. As was done in FFCC, we construct two histograms: one of

pixel intensities, N_0, and one of gradient intensities, N_1. The latter is constructed analogously to Equation 3.

These histograms of log-chroma values exhibit a useful property: element-wise multiplication of the RGB values of an image by a constant results in a *translation* of the resulting log-chrominance histograms. The core insight of CCC is that this property allows color constancy to be framed as the problem of "localizing" a log-chroma histogram in this uv histogram-space [8]—because every uv location in N corresponds to a (normalized) illuminant color, $\boldsymbol{\ell}$, the problem of estimating $\boldsymbol{\ell}$ is *reducible* (in a computability sense) to the problem of estimating a uv coordinate. This can be done by discriminatively training a "sliding window" classifier much as one might train, say, a face-detection system: the histogram is convolved with a (learned) filter and the location of the argmax is extracted from the filter response, and that argmax corresponds to uv value that is (the inverse of) an estimated illumination location.

We adopt a simplification of the convolutional structure used by FFCC [9]:

$$P = \text{softmax}\left(B + \sum_i \left(N_i * F_i\right)\right), \tag{4}$$

where $\{F_i\}$ and B are filters and a bias, respectively, which have the same shape as N_i. Each histogram, N_i, is convolved with each filter, F_i, and summed across channels (a "conv" layer). Then, the bias, B, is added to that summation, which collectively biases inference towards uv coordinates that correspond to common illuminants, such as black body radiation.

As was done in FFCC, this convolution is accelerated through the use of FFTs, though, unlike FFCC, we use a non-wrapped histogram, and thus non-wrapped filters and bias. This avoids the need for the complicated "de-aliasing" scheme used by FFCC which is not compatible with the convolutional neural network structure that we will later introduce.

The output of the softmax, P, is effectively a "heat map" of what illuminants are likely, given the distribution of pixel and gradient intensities reflected in N and in the prior B, from which, we extract a "soft argmax" by taking the expectation of u and v with respect to P:

$$\hat{\ell}_u = \sum_{u,v} uP(u,v)\,, \quad \hat{\ell}_v = \sum_{u,v} vP(u,v). \tag{5}$$

Equation 5 is equivalent to estimating the mean of a fitted Gaussian, in the uv space, weighted by P. Because the absolute scale of $\boldsymbol{\ell}$ is assumed to be irrelevant or unrecoverable in the context of color constancy, after estimating $(\hat{\ell}_u, \hat{\ell}_v)$, we produce an RGB illuminant estimate, $\hat{\boldsymbol{\ell}}$, that is simply the unit vector whose log-chroma values match

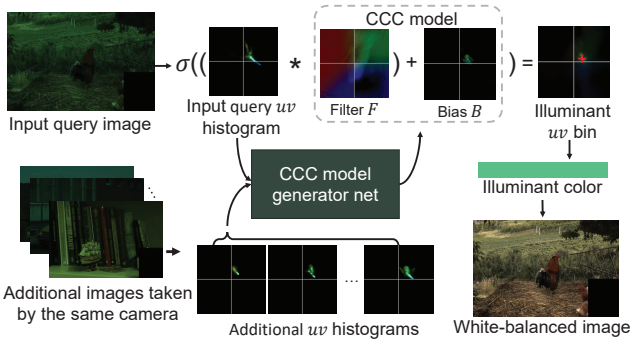

Figure 3: An overview of our C5 model. The uv histograms for the input query image and a variable number of additional input images taken from the same sensor as the query are used as input to our neural network, which generates a filter bank $\{F_i\}$ (here shown as one filter) and a bias B, which are the parameters of a conventional CCC model [8]. The query uv histogram is then convolved by the generated filter and shifted by the generated bias to produce a heat map, whose argmax is the estimated illuminant [8].

our estimate:

$$\hat{\boldsymbol{\ell}} = \left(\exp\left(-\hat{\ell}_u\right)/z,\, 1/z,\, \exp\left(-\hat{\ell}_v\right)/z\right), \tag{6}$$

$$z = \sqrt{\exp\left(-\hat{\ell}_u\right)^2 + \exp\left(-\hat{\ell}_v\right)^2 + 1}. \tag{7}$$

A convolutional color constancy model is then trained by setting $\{F_i\}$ and B to be free parameters which are then optimized to minimize the difference between the predicted illuminant, $\hat{\boldsymbol{\ell}}$, and the ground-truth illuminant, $\boldsymbol{\ell}^*$.

3.1. Architecture

With our baseline CCC/FFCC-like model in place, we can now construct our cross-camera convolutional color constancy model (C5), which is a deep architecture in which CCC is a component. Both CCC and FFCC operate by learning a single fixed set of parameters consisting of a single filter bank $\{F_i\}$ and bias B. In contrast, in C5 the filters and bias are parameterized as the output of a deep neural network (parameterized by weights θ) that takes as input not just log-chrominance histograms for the image being color-corrected (which we will refer to as the "query" image), but also log-chrominance histograms from several other randomly selected input images (but with no ground-truth illuminant labels) from the test set.

By using a generated filter and bias from additional images taken from the query image's camera (instead of using a fixed filter and bias as was done in previous work) our model is able to automatically "calibrate" its CCC model to the specific sensor properties of the query image. This can be thought of as a hypernetwork [28], wherein a deep neural network emits the "weights" of a CCC model, which is itself a shallow neural network. This approach also bears

some similarity to a Transformer approach, as a CCC model can be thought of as "attending" to certain parts of a log-chroma histogram, and so our neural network can be viewed as a sort of self-attention mechanism [56]. See Figure 3 for a visualization of this data flow.

At the core of our model is the deep neural network that takes as input a set of log-chroma histograms and must produce as output a CCC filter bank and bias map. For this we use a multi-encoder-multi-decoder U-Net-like architecture [48]. The first encoder is dedicated to the "query" input image's histogram, while the rest of the encoders take as input the histograms corresponding to the additional input images. To allow the network to reason about the set of additional input images in a way that is insensitive to their ordering, we adopt the permutation invariant pooling approach of Aittala *et al*. [4]: we use max pooling *across* the set of activations of each branch of the encoder. This "cross-pooling" gives us a single set of activations that are reflective of the set of additional input images, but are agnostic to the particular ordering of those input images. At inference time, these additional images are needed to allow the network to reason about how to use them in challenging cases. The cross-pooled features of the last layer of all encoders are then fed into two decoder blocks. Each decoder produces one component of our CCC model: a bias map, B, and two filters, $\{F_0, F_1\}$ (which correspond to pixel and edge histograms, $\{N_0, N_1\}$, respectively).

As per the traditional U-Net structure, we use skip connections between each level of the decoder and its corresponding level of the encoder with the same spatial resolution, but only for the encoder branch corresponding to the query input image's histogram. Each block of our encoder consists of a set of interleaved 3×3 conv layers, leaky ReLU activation, batch normalization, and 2×2 max pooling, and each block of our decoder consists of $2\times$ bilinear upsampling followed by interleaved 3×3 conv layers, leaky ReLU activation, and instance normalization.

When passing our 2-channel (pixel and gradient) log-chroma histograms to our network, we augment each histogram with two extra "channels" comprising of only the u and v coordinates of each histogram, as in CoordConv [40]. This augmentation allows a convolutional architecture on top of log-chroma histograms to reason about the absolute "spatial" information associated with each uv coordinate, thereby allowing a convolutional model to be aware of the absolute color of each histogram bin (see supplementary materials for an ablation study). Figure 4 shows a detailed visualization of our architecture.

3.2. Training

Our model is trained by minimizing the angular error [30] between the predicted unit-norm illuminant color, $\hat{\boldsymbol{\ell}}$, and the ground-truth illuminant color, $\boldsymbol{\ell}^*$, as well as an additional loss that regularizes the CCC models emitted by our network. Our loss function $\mathcal{L}(\cdot)$ is:

$$\mathcal{L}\left(\boldsymbol{\ell}^*, \hat{\boldsymbol{\ell}}\right) = \cos^{-1}\left(\frac{\boldsymbol{\ell}^* \cdot \hat{\boldsymbol{\ell}}}{\|\boldsymbol{\ell}^*\|}\right) + S\left(\{F_i(\theta)\}, B(\theta)\right), \quad (8)$$

where $S(\cdot)$ is a regularizer that encourage the network to generate smooth filters and biases, which reduces over-fitting and improves generalization:

$$\begin{aligned} S\left(\{F_i\}, B\right) = \lambda_B(\|B * \nabla_u\|^2 + \|B * \nabla_v\|^2) \\ +\lambda_F \sum_i (\|F_i * \nabla_u\|^2 + \|F_i * \nabla_v\|^2), \end{aligned} \quad (9)$$

where ∇_u and ∇_v are 3×3 horizontal and vertical Sobel filters, respectively, and λ_F and λ_B are multipliers that control the strength of the smoothness for the filters and the bias, respectively. This regularization is similar to the total variation smoothness prior used by FFCC [9], though here we are imposing it on the filters and bias generated by a neural network, rather than on a single filter bank and bias map. We set the multiplier hyperparameters λ_F and λ_B to 0.15 and 0.02, respectively (see supplementary materials for an ablation study).

In addition to regularizing the CCC model emitted by our network, we additionally regularize the weights of our network themselves, θ, using L2 regularization (i.e., "weight decay") with a multiplier of 5×10^{-4}. This regularization of our network serves a different purpose than the regularization of the CCC models emitted by our network—regularizing $\{F_i(\theta)\}$ and $B(\theta)$ prevents over-fitting by the CCC model emitted by our network, while regularizing θ prevents over-fitting by the model *generating* those CCC models.

Training is performed using the Adam optimizer [36] with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, for 60 epochs. We use a learning rate of 5×10^{-4} with a cosine annealing schedule [41] and increasing batch-size (from 16 to 64) [43, 53] which improve the stability of training (see the supplementary materials for an ablation study).

When training our model for a particular camera model, at each iteration we randomly select a batch of training images (and their corresponding ground-truth illuminants) for use as query input images, and then randomly select *eight* additional input images for each query image from the training set for use as additional input images. See the supplementary materials for results of multiple versions of our model in which we vary the number of additional images used.

4. Experiments and Discussion

In all experiments we used 384×256 raw images after applying the black-level normalization and masking out the

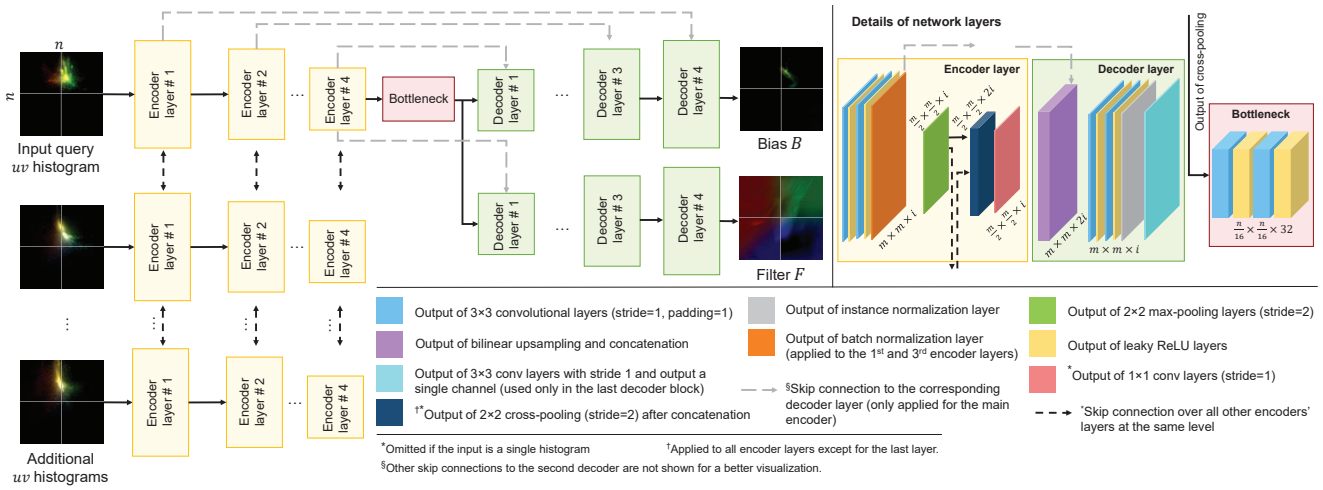

Figure 4: An overview of neural network architecture that emits CCC model weights. The uv histogram of the query image along with additional input histograms taken from the same camera are provided as input to a set of multiple encoders. The activations of each encoder are shared with the other encoders by performing max-pooling across encoders after each block. The cross-pooled features at the last encoder layer are then fed into two decoder blocks to generate a bias and filter bank of an CCC model for the query histogram. Each scale of the decoder is connected to the corresponding scale of the encoder for query histogram with skip connections. The structure of encoder and decoder blocks is shown at the upper right corner.

Figure 5: An example of the image mapping used to augment training data. From left to right: a raw image captured by a Fujifilm X-M1 camera; the same image after white-balancing in CIE XYZ; the same image mapped into the Nikon D40 sensor space; and a real image captured by a Nikon D40 of the same scene for comparison [15].

calibration object to avoid any "leakage" during the evaluation. Excluding histogram computation time (which is difficult to profile accurately due to the expensive nature of scatter-type operations in deep learning frameworks), our method runs in ∼7 milliseconds per image on a NVIDIA GeForce GTX 1080, and ∼90 milliseconds on an Intel Xeon CPU Processor E5-1607 v4 (10M Cache, 3.10 GHz). Because our model exists in log-chroma histogram space, the uncompressed size of our entire model is ∼2 MB, small enough to easily fit within the narrow constraints of limited compute environments such as mobile phones.

4.1. Data Augmentation

Many of the datasets we use contain only a few images per distinct camera model (e.g. the NUS dataset [15]) and this poses a problem for our approach as neural networks generally require significant amounts of training data. To address this, we use a data augmentation procedure in which images taken from a "source" camera model are mapped into the color space of a "target" camera.

To perform this mapping, we first white balance each raw source image using its ground-truth illuminant color, and then transform that white-balanced raw image into the device-independent CIE XYZ color space [16] using the color space transformation matrix (CST) provided in each DNG file [1]. Then, we transform the CIE XYZ image into the target sensor space by inverting the CST of an image taken from the target camera dataset.

Instead of randomly selecting an image from the target dataset, we use the correlated color temperature of each image and the capture exposure setting to match source and target images that were captured under roughly the same conditions. This means that "daytime" source images get warped into the color space of "daytime" target images, etc., and this significantly increases the realism of our synthesized data. After mapping the source image to the target white-balanced sensor space, we randomly sample from a cubic curve that has been fit to the rg chromaticity of illuminant colors in the target sensor.

Lastly, we apply a chromatic adaptation to generate the

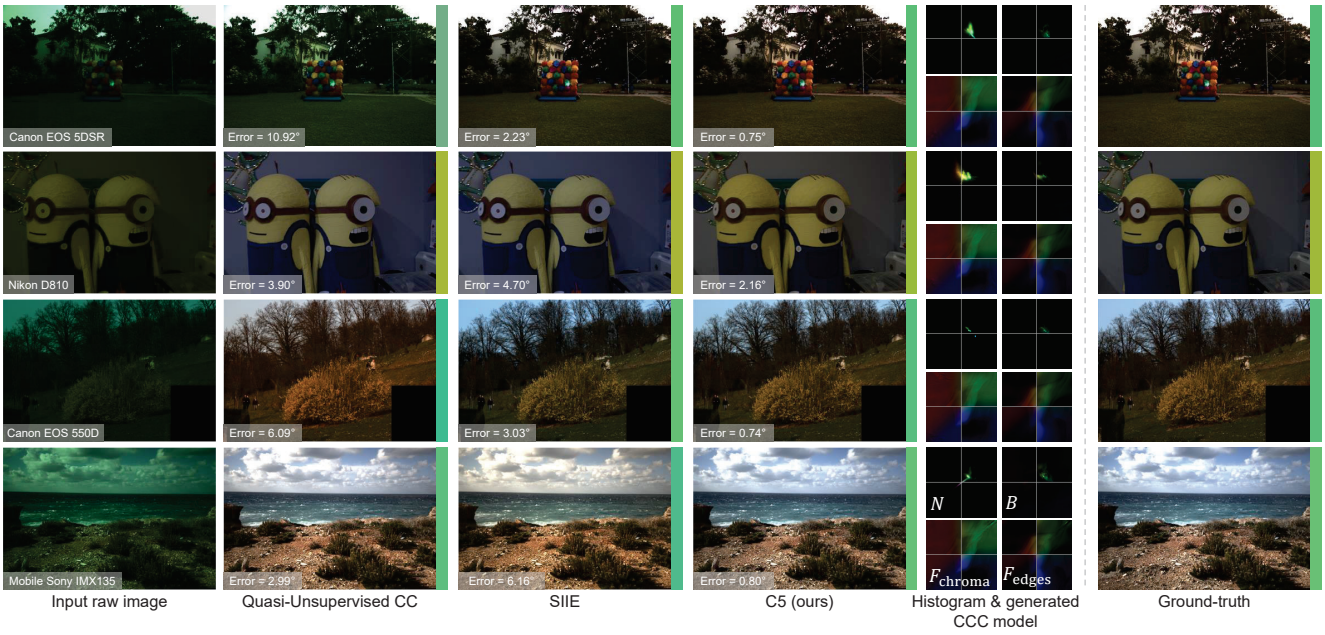

Figure 6: Here we visualize the performance of our C5 model alongside other camera-independent models: "quasi-unsupervised CC" [10] and SIIE [2]. Despite not having seen any images from the test-set camera during training, C5 is able to produce accurate illuminant estimates. The intermediate CCC filters and biases produced by C5 are also visualized.

augmented image in the target sensor space. This chromatic adaptation is performed by multiplying each color channel of the white-balanced raw image, mapped to the target sensor space, with the corresponding sampled illuminant color channel value; see Figure 5 for an example. Additional details can be found in the supplementary materials. This augmentation allows us to generate additional training examples to improve the generalization of our model. More details are provided in Sec. 4.2.

4.2. Results and Comparisons

We validate our model using four public datasets consisting of images taken from one or more camera models: the Gehler-Shi dataset (568 images, two cameras) [24], the NUS dataset (1,736 images, eight cameras) [15], the INTEL-TAU dataset (7,022 images, three cameras) [38], and the Cube+ dataset (2,070 images, one camera) [7] which has a separate 2019 "Challenge" test set [6]. We measure performance by reporting the error statistics commonly used by the community: the mean, median, trimean, and arithmetic means of the first and third quartiles ("best 25%" and "worst 25%") of the angular error between the estimated illuminant and the true illuminant. As our method randomly selects the additional images, each experiment is repeated ten times and we reported the arithmetic mean of each error metric (the supplementary materials contain standard deviations).

To evaluate our model's performance at generalizing to new camera models not seen during training, we adopt a leave-one-out cross-validation evaluation approach: for each dataset, we exclude all scenes and cameras used by the test set from our training images. For a fair comparison with FFCC [9], we trained FFCC using the same leave-one-out cross-validation evaluation approach. Results can be seen in Table 1 and qualitative comparisons are shown in Figures 6 and 7. Even when compared with prior sensor-independent techniques [2, 10], we achieve state-of-the-art performance, as demonstrated in Table 1.

When evaluating on the two Cube+ [6, 7] test sets and the INTEL-TAU [38] dataset in Table 1, we train our model on the NUS [15] and Gehler-Shi [24] datasets. When evaluating on the Gehler-Shi [24] and the NUS [15] datasets in Table 1, we train C5 using the INTEL-TAU dataset [38], the Cube+ dataset [7], and one of the Gehler-Shi [24] and the NUS [15] datasets after excluding the testing dataset. The one deviation from this procedure is for the NUS result labeled "CS", where for a fair comparison with the recent SIIE method [2], we report our results with their cross-sensor (CS) evaluation in Table 1, in which we only excluded images of the test camera, and repeated this process over all cameras in the dataset.

We augmented the data used to train the model, adding 5,000 augmented examples generated as described in Sec. 4.1. In this process, we used only cameras of the training sets of each experiment as "target" cameras for augmentation, which has the effect of mixing the sensors and scene content from the training sets only. For instance, when evaluating on the INTEL-TAU [38] dataset, our augmented im-

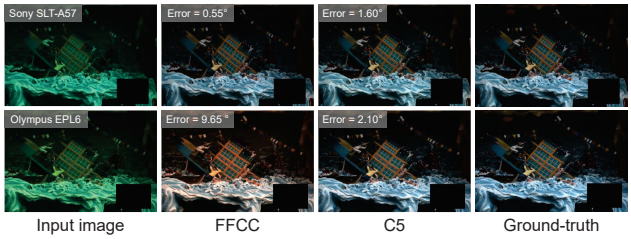

Figure 7: Here we compare our C5 model against FFCC [9] on cross-sensor generalization using test-set Sony SLT-A57 images from the NUS dataset [15]. If FFCC is trained and tested on images from the same camera it performs well, as does C5 (top row). But if FFCC is instead tested on a different camera, such as the Olympus EPL6, it generalizes poorly, while C5 retains its performance (bottom row).

ages simulate the scene content of the NUS [15] dataset as observed by sensors of the Gehler-Shi [24] dataset, and vice-versa.

Characteristics of Additional Images Unless otherwise stated, the additional input images are randomly selected, but from the same camera model as the test image. This setting is meant to be equivalent to the real-world use case in which the additional images provided as input are, say, a photographer's previously-captured images that are already present on the camera during inference. However, for the "Cube+ Challenge" table, we provide an additional set of experiments in Table 1, in which the set of additional images are chosen according to some heuristic, rather than randomly. We identified the 20 test-set images with the lowest variation of uv chroma values ("dull images"), the 20 test-set images with the highest variation of uv chroma values ("vivid images"), and we show that using vivid images produces lower error rates than randomly-chosen or dull images. This makes intuitive sense, as one might expect colorful images to be a more informative signal as to the spectral properties of previously-unobserved camera. We also show results in Table 1 where the additional images are taken from a different camera than the test-set camera, and show that this results in error rates that are higher than using additional images from the same test-set camera, as one might expect.

5. Conclusion

We have presented C5, a cross-camera convolutional color constancy method. By embedding the existing state-of-the-art convolutional color constancy model (CCC) [8, 9] into a multi-input hypernetwork approach, C5 can be trained on images from multiple cameras, but at test time synthesize weights for a CCC-like model that is dynamically calibrated to the spectral properties of the previously-unseen camera of the test-set image. Extensive experimentation demonstrates that C5 achieves state-of-the-art performance on cross-camera color constancy for several datasets. By enabling accurate illuminant estimation without requiring the tedious collection of labeled training data for every particular camera, we hope that C5 will accelerate the widespread adoption of learning-based white balance by the camera industry.

Table 1: Angular errors on the Cube+ dataset [7], the Cube+ challenge [6], the INTEL-TAU dataset [38], the Gehler-Shi dataset [24], and the NUS dataset [15]. The term "CS" refers to cross-sensor as used in [2]. See the text for additional details. Lowest errors are highlighted in yellow.

Cube+ Dataset	**Mean**	**Med.**	**B. 25%**	**W. 25%**	**Tri.**	**Size (MB)**
Gray-world [14]	3.52	2.55	0.60	7.98	2.82	-
Shades-of-Gray [20]	3.22	2.12	0.43	7.77	2.44	-
Cross-dataset CC [37]	2.47	1.94	-	-	-	-
Quasi-Unsupervised CC [10]	2.69	1.76	0.49	6.45	2.00	622
SIIE [2]	2.14	1.44	0.44	5.06	-	10.3
FFCC [9]	2.69	1.89	0.46	6.31	2.08	0.22
C5	1.92	1.32	0.44	4.44	1.46	2.09

Cube+ Challenge	**Mean**	**Med.**	**B. 25%**	**W. 25%**	**Tri.**
Gray-world [14]	4.44	3.50	0.77	9.64	-
1st-order Gray-Edge [54]	3.51	2.30	0.56	8.53	-
Quasi-Unsupervised CC [10]	3.12	2.19	0.60	7.28	2.40
SIIE [2]	2.89	1.72	0.71	7.06	-
FFCC [9]	3.25	2.04	0.64	8.22	2.09
C5	2.24	1.48	0.47	5.39	1.62
C5 (another camera model)	2.97	2.47	0.78	6.11	2.52
C5 (dull images)	2.35	1.58	0.46	5.57	1.70
C5 (vivid images)	2.19	1.39	0.43	5.44	1.54

INTEL-TAU	**Mean**	**Med.**	**B. 25%**	**W. 25%**	**Tri.**
Gray-world [14]	4.7	3.7	0.9	10.0	4.0
Shades-of-Gray [20]	4.0	2.9	0.7	9.0	3.2
PCA-based B/W Colors [15]	4.6	3.4	0.7	10.3	3.7
Weighted Gray-Edge [26]	6.0	4.2	0.9	14.2	4.8
Quasi-Unsupervised CC [10]	3.12	2.19	0.60	7.28	2.40
SIIE [2]	3.42	2.42	0.73	7.80	2.64
FFCC [9]	3.42	2.38	0.70	7.96	2.61
C5	2.52	1.70	0.52	5.96	1.86

Gehler-Shi Dataset	**Mean**	**Med.**	**B. 25%**	**W. 25%**	**Tri.**
Shades-of-Gray [20]	4.93	4.01	1.14	10.20	4.23
PCA-based B/W Colors [15]	3.52	2.14	0.50	8.74	2.47
ASM [5]	3.80	2.40	-	-	2.70
Woo *et al.* [57]	4.30	2.86	0.71	10.14	3.31
Grayness Index [47]	3.07	1.87	0.43	7.62	2.16
Cross-dataset CC [37]	2.87	2.21	-	-	-
Quasi-Unsupervised CC [10]	3.46	2.23	-	-	-
SIIE [2]	2.77	1.93	0.55	6.53	-
FFCC [9]	2.95	2.19	0.57	6.75	2.35
CS	2.50	1.99	0.53	5.46	2.03

NUS Dataset	**Mean**	**Med.**	**B. 25%**	**W. 25%**	**Tri.**
Gray-world [14]	4.59	3.46	1.16	9.85	3.81
Shades-of-Gray [20]	3.67	2.94	0.98	7.75	3.03
Local Surface Reflectance [22]	3.45	2.51	0.98	7.32	2.70
PCA-based B/W Colors [15]	2.93	2.33	0.78	6.13	2.42
Grayness Index [47]	2.91	1.97	0.56	6.67	2.13
Cross-dataset CC [37]	3.08	2.24	-	-	-
Quasi-Unsupervised CC [10]	3.00	2.25	-	-	-
SIIE (CS) [2]	2.05	1.50	0.52	4.48	
FFCC [9]	2.87	2.14	0.71	6.23	2.30
CS	2.54	1.90	0.61	5.61	2.02
C5 (CS)	1.77	1.37	0.48	3.75	1.46

References

[1] Digital negative (DNG) specification. Technical report, Adobe Systems Incorporated, 2012. Version 1.4.0.0.

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

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

[4] Miika Aittala and Frédo Durand. Burst image deblurring using permutation invariant convolutional neural networks. *ECCV*, 2018.

[5] Arash Akbarinia and C Alejandro Parraga. Colour constancy beyond the classical receptive field. *TPAMI*, 2017.

[6] Nikola Banić and Karlo Koščević. Illumination estimation challenge. `https://www.isispa.org/illumination-estimation-challenge`. Accessed: 2021-03-07.

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

[8] Jonathan T Barron. Convolutional color constancy. *ICCV*, 2015.

[9] Jonathan T Barron and Yun-Ta Tsai. Fast Fourier color constancy. *CVPR*, 2017.

[10] Simone Bianco and Claudio Cusano. Quasi-Unsupervised color constancy. *CVPR*, 2019.

[11] Simone Bianco, Claudio Cusano, and Raimondo Schettini. Color constancy using cnns. *CVPR Workshops*, 2015.

[12] David H Brainard and William T Freeman. Bayesian color constancy. *JOSA A*, 1997.

[13] David H Brainard and Brian A Wandell. Analysis of the retinex theory of color vision. *JOSA A*, 1986.

[14] Gershon Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin Institute*, 1980.

[15] 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*, 2014.

[16] C CIE. Commission internationale de l'eclairage proceedings, 1931. *Cambridge University, Cambridge*, 1932.

[17] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *JAIR*, 2006.

[18] Graham D Finlayson and Steven D Hordley. Color constancy at a pixel. *JOSA A*, 2001.

[19] Graham D Finlayson, Steven D Hordley, and Ingeborg Tastl. Gamut constrained illuminant estimation. *IJCV*, 2006.

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

[21] David A Forsyth. A novel algorithm for color constancy. *IJCV*, 1990.

[22] Shaobing Gao, Wangwang Han, Kaifu Yang, Chaoyi Li, and Yongjie Li. Efficient color constancy with local surface reflectance statistics. *ECCV*, 2014.

[23] Shao-Bing Gao, Ming Zhang, Chao-Yi Li, and Yong-Jie Li. Improving color constancy by discounting the variation of camera spectral sensitivity. *JOSA A*, 2017.

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

[25] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Generalized gamut mapping using image derivative structures for color constancy. *IJCV*, 2010.

[26] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Improving color constancy by photometric edge weighting. *TPAMI*, 2012.

[27] Michael D Grossberg and Shree K Nayar. Modeling the space of camera response functions. *TPAMI*, 2004.

[28] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[29] Daniel Hernandez-Juarez, Sarah Parisot, Benjamin Busam, Ales Leonardis, Gregory Slabaugh, and Steven McDonagh. A multi-hypothesis approach to color constancy. *CVPR*, 2020.

[30] Steven D Hordley and Graham D Finlayson. Re-evaluating colour constancy algorithms. In *ICPR*, 2004.

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

[32] Jun Jiang, Dengyu Liu, Jinwei Gu, and Sabine Süsstrunk. What is the space of spectral sensitivity functions for digital color cameras? *WACV*, 2013.

[33] Thorsten Joachims. Learning to classify text using support vector machines. *ICML*, 1999.

[34] Hamid Reza Vaezi Joze, Mark S Drew, Graham D Finlayson, and Perla Aurora Troncoso Rey. The role of bright pixels in illumination estimation. *Color and Imaging Conference*, 2012.

[35] Seon Joo Kim, Jan-Michael Frahm, and Marc Pollefeys. Radiometric calibration with illumination change for outdoor scene analysis. *CVPR*, 2008.

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

[37] Samu Koskinen12, Dan Yang, and Joni-Kristian Kämäräinen. Cross-dataset color constancy revisited using sensor-to-sensor transfer. *BMVC*, 2020.

[38] Firas Laakom, Jenni Raitoharju, Alexandros Iosifidis, Jarno Nikkanen, and Moncef Gabbouj. Intel-TAU: A color constancy dataset. *arXiv preprint arXiv:1910.10404*, 2019.

[39] Orly Liba, Kiran Murthy, Yun-Ta Tsai, Tim Brooks, Tianfan Xue, Nikhil Karnad, Qiurui He, Jonathan T Barron, Dillon Sharlet, Ryan Geiss, et al. Handheld mobile photography in very low light. *ACM TOG*, 2019.

[40] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *NeurIPS*, 2018.

[41] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[42] Zhongyu Lou, Theo Gevers, Ninghang Hu, Marcel P Lucassen, et al. Color constancy by deep learning. *BMVC*, 2015.

[43] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[44] Steven McDonagh, Sarah Parisot, Fengwei Zhou, Xing Zhang, Ales Leonardis, Zhenguo Li, and Gregory Slabaugh. Formulating camera-adaptive color constancy as a few-shot meta-learning problem. *arXiv preprint arXiv:1811.11788*, 2018.

[45] Seoung Wug Oh and Seon Joo Kim. Approaching the computational color constancy as a classification problem through deep learning. *Pattern Recognition*, 2017.

[46] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE TKDE*, 2009.

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

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. *MICCAI*, 2015.

[49] Charles Rosenberg, Martial Hebert, and Sebastian Thrun. Color constancy using KL-divergence. *ICCV*, 2001.

[50] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. *ECCV*, 2010.

[51] Lilong Shi and Brian Funt. MaxRGB reconsidered. *Journal of Imaging Science and Technology*, 2012.

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

[53] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[54] Joost Van De Weijer, Theo Gevers, and Arjan Gijsenij. Edge-based color constancy. *IEEE TIP*, 2007.

[55] Vladimir Vapnik. Statistical learning theory. *Wiley*, 1998.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[57] S. Woo, S. Lee, J. Yoo, and J. Kim. Improving color constancy in an ambient light environment using the phong reflection model. *IEEE TIP*, 2018.

[58] Seoung Wug Oh, Michael S Brown, Marc Pollefeys, and Seon Joo Kim. Do it yourself hyperspectral imaging with everyday digital cameras. *CVPR*, 2016.

[59] Jin Xiao, Shuhang Gu, and Lei Zhang. Multi-domain learning for accurate and few-shot color constancy. *CVPR*, 2020.

[60] Bolei Xu, Jingxin Liu, Xianxu Hou, Bozhi Liu, and Guoping Qiu. End-to-end illuminant estimation based on deep metric learning. *CVPR*, 2020.