A Convolutional Framework for Color Constancy

Marco Buzzelli and Simone Bianco

***Abstract*— We introduce a convolutional framework (CF) for computational color constancy, building upon the established low-level image feature-based framework, which utilized simple image statistics for illuminant estimation. Our framework expands upon this through an end-to-end learnable neural architecture. This adaptation enables the learning and usage of advanced filters that are not restricted to Gaussian kernels operating on individual color channels, thus generalizing the capabilities of the original framework. Additionally, our general framework supports deeper convolutional architectures, thus increasing its computational power. It can also be efficiently applied to estimate multiple spatially varying illuminants within a single scene. Our experimental results on standard datasets demonstrate that the CF outperforms the best methods in the low-level framework, improving the illuminant estimation accuracy by up to 34% for single illuminant estimation and 30% for multiple illuminants estimation. Additionally, our framework exhibits superior performance even when the number of training images is reduced. Finally, we document the inference speedup of our implementation reaching up to 30×, making the CF especially suitable for applications where efficiency is critical. Source code and trained models available at: https://github.com/MarcoBauzz/convolutional-color-constancy**

***Index Terms*— Automatic white balance, color constancy, convolutional neural networks, framework, illuminant estimation.**

I. Introduction

AMONG the fundamental problems in computer vision is that of estimating the real color of the objects in the acquired scene: the observed color in fact depends on the surface spectral reflectance of the object, on the illumination, on their relative positions, and on the observer's characteristics. This problem is called (computational) color constancy and many computer vision tasks, both in the image and the video domains, can exploit it as a preprocessing step in order to guarantee that the observed color of the objects in the scene does not change in different illumination conditions.

Color constancy often consists of two successive steps: 1) estimating the color of the illuminant in the scene and 2) rendering the colors of the objects in the scene as if they were viewed under a white light source. A common choice for the latter step is the diagonal von Kries model [1]: an independent scaling of the color components of the pixels by the corresponding components of the estimate. The former step, on the other hand, is usually where methods in the state of the art differ.

Despite its apparent simplicity, the problem of color constancy is very challenging for both human and computer vision systems [2], [3] as it is inherently underdetermined. This is the reason why many color constancy algorithms have been proposed in the state of the art, each based on different assumptions. A common taxonomy of color constancy methods divides them into statistics-based (or parametric) and learning-based methods.

Historically, the first proposed methods belong to the category of statistics-based solutions. These are based on simple assumptions, such as the white patch (WP) (or max-RGB) [4] that estimates the light source color from the maximum response of the different color channels, or the gray world (GW) [5] that assumes that the average reflectance in the scene is achromatic. These assumptions were later merged in the shades of gray (SoG) hypothesis [6], which assumes that the L^p Minkowski norm of the image is achromatic, and further refined by the gray-edge hypothesis [7], which assumes that the L^p Minkowski norm of the nth order derivative of the image is achromatic. Other methods in the parametric or statistics-based category are for example the one proposed by [8], which exploits bright pixels in the image to estimate the illuminant color, or the one proposed by [9], which selects bright and dark pixels using a projection distance in the color distribution and then applies a dimensionality reduction technique to estimate the illuminant color.

The second category of methods is represented by learning-based solutions. Early learning-based methods were based on handcrafted features extracted from the images. Representatives of this type of method are for example the one proposed by [10], where the Weibull parameterization (e.g., grain size and contrast) is used to capture the image characteristics, or the one proposed by [11], where a set of general purpose low-level visual properties taken from the pattern recognition and image analysis fields are used to train a classifier to select which algorithm to apply to the particular input image. Other notable examples of this type of solution are for example the works of [12], [13], [14], [15], and [16]. Later methods in this category are based on deep learning, and range from those fine-tuning neural architectures designed for image recognition [17], to those designing ad hoc architectures [18], [19] or more elaborated training schemes (e.g., [20], [21]), to those exploiting generative models (e.g., [22], [23], [24], [25]). Approaches like sensor-independent frameworks [26] and discriminative feature learning [27], [28] have improved robustness and generalization. Cross-camera adaptation techniques [29] and contrastive learning strategies [30] further refine real-time

Manuscript received 27 October 2023; revised 28 June 2024; accepted 28 August 2024. This work was supported in part by the Ministero dell'Università e della Ricerca (MUR) through the Department of Informatics, Systems and Communication, University of Milano - Bicocca (Dipartimenti di Eccellenza 2023–2027), Italy. *(Corresponding author: Marco Buzzelli.)*

The authors are with the Department of Informatics Systems and Communication, University of Milano - Bicocca, 20126 Milan, Italy (e-mail: marco.buzzelli@unimib.it; simone.bianco@unimib.it).

Digital Object Identifier 10.1109/TNNLS.2024.3454484

applications and feature specificity. Further improvements have been reached, thanks to transfer and adversarial learning [31], [32], along with ranking-based mechanisms [33] and chromaticity representations [34]. Learning-based approaches tend to achieve the best accuracy in terms of illuminant estimation on standard datasets, but we have reached the point in which improvements are below the threshold of what can be detected by a human observer [35]. To reach higher illuminant estimation accuracy researchers tend to use wider and deeper models, requiring large labeled datasets for training, and making them unfeasible for practical uses.

Alongside the proposal of new methods, there are also works in the state of the art that unify multiple color constancy algorithms, redefining them as instantiations of a single, more general, framework. With this aim, Finlayson et al. [12] presented the correlation framework. They propose to estimate the illuminant color by correlating the image data and the prior knowledge about which colors appear under a certain light. They further show that the correlation framework they developed is general and can be used to describe several existing algorithms. In particular, they show that GW [5], gamut mapping [36], [37], and a neural network approach [38] relate to different definitions of color, likelihood, and correlation. Finlayson and Trezzi [6] introduced a new color constancy framework and showed that the max-RGB and the GW methods can be interpreted as different instantiations of the error function of the same algorithm, which they call SoG. In particular, they showed that the max-RGB method is equal to applying the L^{∞} Minkowski norm to each color channel of the input image independently, and GW is equal to using the L^1 norm. van de Weijer et al. [7] further extended the color constancy framework of Finlayson and Trezzi [6] to also include color constancy methods derived from the gray-edge hypothesis. They thus introduced a framework of color constancy based on low-level image features which includes the algorithms already included in [6] (i.e., GW , max-RGB, and SoG) and their newly proposed gray-edge and higher order gray-edge algorithms. We will refer to this formulation as the low-level framework in the rest of the article.

After 2007, no other unifying frameworks have been proposed, but in the meanwhile computer vision has entered the era of deep learning and convolutional neural networks, leading to a drastic paradigm shift in how different tasks and challenges are approached. The aim of this work is to introduce a new unifying framework for color constancy: the convolutional framework (CF).[1] The proposed framework extends the low-level framework by [7] combining its simplicity with the power of deep learning. This allows to bridge the gap in illuminant estimation accuracy between simple statistics-based algorithms and recent deep learning-based algorithms, while keeping a small model size and fast inference time to target applications where efficiency is critical.

The main contribution of this article is the CF for color constancy, and its characteristics are listed below.

1) CF replicates the results of the low-level framework [7], originally written in MATLAB, with an efficient Python/PyTorch implementation. The source code is made available for public download.
2) It is capable of learning network parameters that must be otherwise set by hand in the original low-level framework. Furthermore, the proposed framework has the ability to learn filter kernels that are not Gaussian, and it can learn to exploit cross-channel information at the same computational cost.
3) It provides a search space that can be explored by neural architecture search (NAS) techniques to design the best color constancy method for each dataset. The search space is shown to include recent color constancy algorithms (e.g., [39], [40]).

Additionally, our convolutional-based implementation allows for a seamless spatial extension of the framework, thus producing spatially varying illuminant estimation. In terms of execution time, our experiments document an inference speedup gain for the most computationally intensive method of more than 3× compared to the original low-level framework, reaching about 30× when exploiting multibatch processing.

The rest of the article is organized as follows: Section II formalizes the color constancy problem and introduces the CF. Section III describes how the CF is made learnable. The CF is extended in Section IV to include deeper architectures and estimate local, spatially varying scene illuminants. Sections V and VI respectively introduce the experimental setup and results. Finally, Section VIII concludes the article.

II. Convolutional Framework

In the image acquisition process, the values for a Lambertian surface located at the pixel with position x can be seen as a function $f(x)$ that mainly depends on three physical factors

$$f(x) = \int I(x,\lambda)S(x,\lambda)C(\lambda)d\lambda \tag{1}$$

where λ is the wavelength, $I(x,\lambda)$ is the illuminant spectral power distribution, $S(x,\lambda)$ is the surface spectral reflectance, $C(\lambda)$ are the sensor spectral sensitivities, and the integration is performed over the visible spectrum. With this notation the goal of color constancy is to estimate the color of the scene illuminant, i.e., the projection of $I(x,\lambda)$ on the sensor spectral sensitivities $C(\lambda)$

$$e(x) = \int I(x,\lambda)C(\lambda)d\lambda. \tag{2}$$

Usually, the illuminant color is estimated up to a scale factor [i.e., $k\cdot e(x)$] as it is more important to estimate the chromaticity of the scene illuminant rather than its overall intensity [41].

Since the only information available is the sensor response $f(x)$ across the image, color constancy is an underdetermined problem [42] and thus further assumptions and/or knowledge are needed to solve it. Several computational color constancy algorithms have been proposed in the state of the art, each based on different assumptions. van de Weijer et al. [7] proposed a framework from which several algorithms can be instantiated: the low-level framework. The aim of this section

[1] Source code and trained models available at: https://github.com/MarcoBauzz/convolutional-color-constancy.

is to reformulate the low-level framework with an equivalent definition based on convolutional operations, laying the basis for its eventual extension as later described in Section IV. The general hypothesis of the original low-level framework is described as follows:

$$\left(\int \left|\frac{\partial^n f^\sigma(x)}{\partial x^n}\right|^p dx\right)^{1/p} = k \cdot e^{n,p,\sigma} \tag{3}$$

where n identifies the derivative order, σ is the standard deviation for a Gaussian filter such that $f^\sigma(x) = G_\sigma * f(x)$, and p is the order of the Minkowski norm. A further assumption made by the low-level framework, which is also the most common assumption made by other algorithms, is that the color of the light source is uniform across the scene. This is reflected in the fact that the spatial position (x) in the right-hand side of (3) is dropped. Such behavior is generalized by our framework, thus enabling spatially varying multi-illuminant estimation as described in Section IV-B.

The framework described in (3) includes six color constancy algorithms that can be generated with different combinations of n, p, and σ.

1) GW, by setting $[n, p, \sigma] = [0, 1, 0]$.
2) WP, by setting $[n, p, \sigma] = [0, \infty, 0]$.
3) SoG, by setting $[n, p, \sigma] = [0, p, 0]$.
4) General GW (gGW), by setting $[n, p, \sigma] = [0, p, \sigma]$.
5) Gray edge first order (GE1), by setting $[n, p, \sigma] = [1, p, \sigma]$.
6) Gray edge second order (GE2), by setting $[n, p, \sigma] = [2, p, \sigma]$.

The appropriate values for these settings have been in the past optimized with a grid search approach on given datasets [18], whereas the framework presented in this article allows their generalization and optimization via gradient backpropagation.

The framework described by (3) can be directly translated in the convolutional neural network architecture described in the following, and is reported in Table I. Let us assume that the input image has size $H \times W \times 3$. The first layer of the CNN is a convolutional layer (`Conv2d-1`) with size $h \times w \times 3 \times 9$ with $[(h-1)/2, (w-1)/2]$ padding transforming the three-channel input into a $H \times W \times 9$ tensor. The second layer is an exponentiation layer (`Pow-1`) that takes the power of each element in input with exponent α_1 and returns a tensor with the result. The third layer is a pointwise convolution layer (`Conv2d-2`) with size $1 \times 1 \times 9 \times 3$ that linearly combines the activations across channels to obtain as output a $H \times W \times 3$ tensor. The fourth layer is another element-wise exponentiation layer (`Pow-2`) with exponent α_2 which returns a tensor having the same size as the input. The last layer is a 2-D power-average pooling layer (`LpPool2D`) with norm type p and having kernel size $H \times W$ to generate the final illuminant estimate with size $1 \times 1 \times 3$. Varying p the behavior of the layer changes from Max pooling ($p = \infty$) to Sum pooling ($p = 1$, which is an unscaled version of the Average pooling).

TABLE I
CNN ARCHITECTURE OF THE CF

Layer	Size	Padding	Value	Output size
`Input`				H×W×3
`Conv2D-1`	h×w×3×9	[(h-1)/2,(w-1)/2]		H×W×9
`Pow-1`			α_1	H×W×9
`Conv2D-2`	1×1×9×3			H×W×3
`Pow-2`			α_2	H×W×3
`LpPool2D`	[H W]		p	1×1×3

A. Initialization

The six color constancy algorithms described above can be replicated in the equivalent CF by properly initializing the available parameters and hyperparameters.

We will identify the convolutional filters using the following syntax, under a base-1 indexing system:

$$C_N\big[\text{row, column, input channel, output channel}\big]. \tag{4}$$

All filters are initialized with zeroes, unless otherwise specified.

1) Gray World: The `Conv2D-1` layer is initialized so that it applies the identity function

$$C_1[:, :, 1, 1] = C_1[:, :, 2, 2] = C_1[:, :, 3, 3] = \lim_{\sigma \to 0} G_\sigma \tag{5}$$

i.e.,

$$C_1\left[\left\lceil\frac{h}{2}\right\rceil, \left\lceil\frac{w}{2}\right\rceil, 1, 1\right] = C_1\left[\left\lceil\frac{h}{2}\right\rceil, \left\lceil\frac{w}{2}\right\rceil, 2, 2\right] \tag{6}$$

$$= C_1\left[\left\lceil\frac{h}{2}\right\rceil, \left\lceil\frac{w}{2}\right\rceil, 3, 3\right] = 1. \tag{7}$$

The `Pow-1` and `Pow-2` layers are initialized with $\alpha_1 = \alpha_2 = 1$. The `Conv2D-2` layer is initialized so that it applies the identity function

$$C_2[1, 1, 1, 1] = C_2[1, 1, 2, 2] = C_2[1, 1, 3, 3] = 1. \tag{8}$$

Finally, the `LpPool2D` layer is initialized with $p = 1$.

2) White Patch: It can be replicated with the same initialization of the GW with the only difference of the `LpPool2D` layer, to be initialized with $p = \infty$.

3) Shades of Gray: It can be replicated with the same initialization of the GW, with the only difference of the `LpPool2D` layer, to be initialized with the corresponding p value of the nonconvolutional version that we want to replicate.

4) General GW: The `Conv2D-1` layer is initialized so that it applies the Gaussian smoothing G_σ independently to the three input channels

$$C_1[:, :, 1, 1] = C_1[:, :, 2, 2] = C_1[:, :, 3, 3] = G_\sigma. \tag{9}$$

The `Pow-1` and `Pow-2` layers are initialized with $\alpha_1 = \alpha_2 = 1$. The `Conv2D-2` layer is initialized so that it applies the identity function

$$C_2[1, 1, 1, 1] = C_2[1, 1, 2, 2] = C_2[1, 1, 3, 3] = 1. \tag{10}$$

Finally, the `LpPool2D` layer is initialized with the corresponding p value of the nonconvolutional version that we want to replicate.

TABLE II
SUMMARY OF THE INITIALIZATIONS NEEDED TO REPLICATE THE BEHAVIOR OF THE CORRESPONDING NONCONVOLUTIONAL COLOR CONSTANCY ALGORITHMS INCLUDED IN THE FRAMEWORK

Method	`Conv2D-1`	`Pow-1`	`Conv2D-2`	`Pow-2`	`LpPool2D`
GW	$C_1[:,:,1,1] = C_1[:,:,2,2] = C_1[:,:,3,3] = \lim_{\sigma\to 0} G_\sigma$	1	$C_2[1,1,1,1] = C_2[1,1,2,2] = C_2[1,1,3,3] = 1$	1	1
WP	$C_1[:,:,1,1] = C_1[:,:,2,2] = C_1[:,:,3,3] = \lim_{\sigma\to 0} G_\sigma$	1	$C_2[1,1,1,1] = C_2[1,1,2,2] = C_2[1,1,3,3] = 1$	1	∞
SoG	$C_1[:,:,1,1] = C_1[:,:,2,2] = C_1[:,:,3,3] = \lim_{\sigma\to 0} G_\sigma$	1	$C_2[1,1,1,1] = C_2[1,1,2,2] = C_2[1,1,3,3] = 1$	1	p
gGW	$C_1[:,:,1,1] = C_1[:,:,2,2] = C_1[:,:,3,3] = G_\sigma$	1	$C_2[1,1,1,1] = C_2[1,1,2,2] = C_2[1,1,3,3] = 1$	1	p
GE1	$C_1[:,:,1,1] = C_1[:,:,2,2] = C_1[:,:,3,3] = \frac{\partial G_\sigma}{\partial x}$ $C_1[:,:,1,4] = C_1[:,:,2,5] = C_1[:,:,3,6] = \frac{\partial G_\sigma}{\partial y}$	2	$C_2[1,1,1,1] = C_2[1,1,4,1] = 1$ $C_2[1,1,2,2] = C_2[1,1,5,2] = 1$ $C_2[1,1,3,3] = C_2[1,1,6,3] = 1$	$\frac{1}{2}$	p
GE2	$C_1[:,:,1,1] = C_1[:,:,2,2] = C_1[:,:,3,3] = \frac{\partial^2 G_\sigma}{\partial x^2}$ $C_1[:,:,1,4] = C_1[:,:,2,5] = C_1[:,:,3,6] = \frac{\partial^2 G_\sigma}{\partial y^2}$ $C_1[:,:,1,7] = C_1[:,:,2,8] = C_1[:,:,3,9] = \frac{\partial^2 G_\sigma}{\partial xy}$	2	$C_2[1,1,1,1] = C_2[1,1,2,2] = C_2[1,1,3,3] = 1$ $C_2[1,1,4,1] = C_2[1,1,5,2] = C_2[1,1,6,3] = 1$ $C_2[1,1,7,1] = C_2[1,1,8,2] = C_2[1,1,9,3] = 2$	$\frac{1}{2}$	p

5) Gray-Edge First Order: The `Conv2D-1` layer is initialized so that it independently applies the first-order partial derivative along rows and columns of the three Gaussian smoothed input channels

$$C_1[:,:,1,1] = C_1[:,:,2,2] = C_1[:,:,3,3] = \frac{\partial G_\sigma}{\partial x} \tag{11}$$

$$C_1[:,:,1,4] = C_1[:,:,2,5] = C_1[:,:,3,6] = \frac{\partial G_\sigma}{\partial y}. \tag{12}$$

The `Pow-1` layer is initialized with $\alpha_1 = 2$. The `Conv2D-2` layer is initialized so that it computes the gradient of each input color channel

$$C_2[1,1,1,1] = C_2[1,1,4,1] = 1 \tag{13}$$

$$C_2[1,1,2,2] = C_2[1,1,5,2] = 1 \tag{14}$$

$$C_2[1,1,3,3] = C_2[1,1,6,3] = 1. \tag{15}$$

This ensures that in the first channel of the output tensor we compute the gradient of the red channel, in the second one the gradient of the green channel, and in the third one the gradient of the blue channel. The `Pow-2` layer is initialized with $\alpha_2 = 1/\alpha_1 = 1/2$. Finally, the `LpPool2D` layer is initialized with the corresponding p value of the nonconvolutional version that we want to replicate.

6) Gray-Edge Section Order: The `Conv2D-1` layer is initialized so that it independently applies the second-order partial derivative of the three Gaussian-smoothed input channels along rows, columns, and the mixed second-order partial derivative

$$C_1[:,:,1,1] = C_1[:,:,2,2] = C_1[:,:,3,3] = \frac{\partial^2 G_\sigma}{\partial x^2} \tag{16}$$

$$C_1[:,:,1,4] = C_1[:,:,2,5] = C_1[:,:,3,6] = \frac{\partial^2 G_\sigma}{\partial y^2} \tag{17}$$

$$C_1[:,:,1,7] = C_1[:,:,2,8] = C_1[:,:,3,9] = \frac{\partial^2 G_\sigma}{\partial xy}. \tag{18}$$

The `Pow-1` layer is initialized with $\alpha_1 = 2$. The `Conv2D-2` layer is initialized so that it computes the gradient of each input color channel

$$C_2[1,1,1,1] = C_2[1,1,2,2] = C_2[1,1,3,3] = 1 \tag{19}$$

$$C_2[1,1,4,1] = C_2[1,1,5,2] = C_2[1,1,6,3] = 1 \tag{20}$$

$$C_2[1,1,7,1] = C_2[1,1,8,2] = C_2[1,1,9,3] = 2. \tag{21}$$

This ensures that in the first channel of the output tensor we compute the gradient of the red channel, in the second one the gradient of the green channel, and in the third one the gradient of the blue channel. Note that the original implementation of the low-level framework mistakenly uses the equivalent of a factor 4 in (21), instead of 2. The `Pow-2` layer is initialized with $\alpha_2 = 1/\alpha_1 = 1/2$. Finally, the `LpPool2D` layer is initialized with the corresponding p value of the nonconvolutional version that we want to replicate.

A summary of the initializations needed to replicate the behavior of the corresponding nonconvolutional color constancy algorithms included in the framework is reported in Table II.

In addition to the algorithms included in the original low-level framework, the proposed CF also includes the truncated version of the same algorithms [43]: in the original low-level framework, the spatial size of `Conv2D-1` (i.e., $h = w$) is guided by the Gaussian filter standard deviation σ and the break-off value B: $h = w = (\lfloor B \cdot \sigma + 0.5 \rfloor \cdot 2) + 1$, where the default is $B = 3$. The truncated low-level framework algorithms can be replicated in our CF by specifying the required filter size needed $h \times w$, which corresponds to setting a new value for the break-off sigma, that can be even different in the two directions.

III. LEARNABLE CF

In Section II, we defined a CF that is equivalent to the low-level framework by [7], and as such it includes the same family of color constancy algorithms. In this section, we describe how to render a learnable CF (LCF), providing access to a larger number of algorithms not covered by the original framework. For example, we can learn smoothing and derivative filters that are not Gaussian, or we can learn to exploit cross-channel information, e.g., color derivatives. These can be obtained with properly tuned filter weights

TABLE III
CNN ARCHITECTURE OF THE LCF, WITH PROTECTIONS FOR ASSISTED BACKPROPAGATION

Layer	Size	Padding	Value	Output size
`Input`				H×W×3
`Conv2D-1`	h×w×3×9	[(h-1)/2,(w-1)/2]		H×W×9
`PReLU`			init: y = x	H×W×9
`Pow-1`			α_1	H×W×9
`Conv2D-2`	1×1×9×3		init: bias = 1	H×W×3
`PosReLU`				H×W×3
`Pow-2`			α_2	H×W×3
`Abs`				H×W×3
`LpPool2D`	[H W]		$p \geq 1$	1×1×3
Total number of learnable parameters:				27·h·w+41

in `Conv2D-1` and `Conv2D-2`, as we experimentally show and discuss in Section VI-B1. Additionally, we formulate the `LpPool2D` layer so that the exponent p can be learned as well.

A. Assisting Gradient Backpropagation

In order to enable the learning process, we introduce a number of modifications to the CF, aimed at ensuring proper gradient backpropagation. These modifications are also designed to guarantee the possibility of the CF to learn all the methods from the low-level framework. The detailed modifications are summarized in Table III and presented in the following.

1) A parametric rectifying linear unit (PReLU) [44] is introduced after the first convolutional layer. This is done to enforce a sharper nonlinearity than `Pow-1` (assuming that $\alpha_1 > 1$), since it has been proven to better guide the learning process. Specifically, a PReLU is used in place of a ReLU, since the latter would destroy the negative values returned by the derivative filter implemented with `Conv2D-1`, thus breaking the generalization of low-level framework. It is initialized with a slope coefficient set to 1, which corresponds to the identity function.
2) The bias in `Conv2D-2` is initialized to a vector of 1 s. A traditional He-based initialization [44] with null bias was found to occasionally produce all-negative activations for some of the channels. These would be destroyed by the PosReLU introduced afterward (and by any other ReLU-like nonlinearity that can be included for extension), preventing the gradient from correctly flowing during the backpropagation, and from guiding the learning process. By enforcing a reasonably high initialization bias, this can be avoided.
3) A strictly positive ReLU (PosReLU) is introduced before the `Pow-2` layer. This is defined as follows:

$$\text{PosReLU}(x) = \max(\epsilon, x) \tag{22}$$

for a sufficiently small ϵ. This layer is devised as a protection when $0 < \alpha_2 < 1$ (such as for the GE1 and GE2 initializations), since in this case the derivative of the power of a zero base is not defined, and returns ∞ in the adopted framework (PyTorch 1.7).

4) The learnable p in `LpPool2D` is forced to be ≥ 1. Recall the definition of the pooling layer as follows:

$$\text{LpPool2D}(x) = \sqrt[p]{\sum_{x \in X} x^p}. \tag{23}$$

An unconstrained p was found to hinder the loss minimization when $0 < p < 1$, since $\sqrt[p]{\cdot}$ tends to ∞ for $p \to 0$. The value of p was therefore forced to be ≥ 1 by reprocessing it as $|p-1|+1$. Furthermore, it was initialized to $1+\epsilon$ to avoid starting with a null gradient and thus to bootstrap its optimization.

IV. EXTENDED LCF

In this section, we present two extensions of the previously introduced LCF: a generalization of the model architecture to also include deeper models, and the estimation of multiple spatially varying illuminants.

A. Deeper Architecture

The CNN architecture of the extended CF (ECF) here described is reported in Table IV. An analysis of the architecture shows a deeper model with respect to the one reported in Table III. In particular, the depth is increased by including an intermediate convolutional layer `Conv2D-i` that can be repeated $N\times$, and the use of multiple max pooling layers (`Mpool-1` and `Mpool-2`). A dropout layer is also added to reduce the risk of overfitting when several intermediate convolutional layers are used.

The LCF can be instantiated from ECF by setting $c_1 = 9$ and $h_{m_1} = w_{m_1} = 1$ in the first block; excluding the second block (i.e., $N = 0$); setting $p_d = 0$, $h_{c_2} = w_{c_2} = 1$, $c_i = 9$, and $h_{m_2} = w_{m_2} = 1$ in the third block; and setting the kernel size of `LpPool2D` equal to the spatial size of the output of the third block, i.e., $[h_p \ w_p] = [H_4 \ W_4]$ so that the size of the output is $H_5 \times W_5 \times 3 = 1 \times 1 \times 3$.

The extended LCF also includes two recent color constancy algorithms: convolutional mean [39] and OneNet [40].

The convolutional mean algorithm can be instantiated by setting $h_{c_1} = w_{c_1} = 3$, $c_1 = 7$, $h_{m_1} = w_{m_1} = 2$, and $\alpha_1 = 1$ in the first block; setting $N = 1$, $h_{c_i} = w_{c_i} = 3$, and $c_i = 14$ in the second, intermediate, block; setting $p_d = 0$, $h_{c_2} = w_{c_2} = 1$, $h_{m_2} = w_{m_2} = 1$, and $\alpha_2 = 1$ in the third block; setting the kernel size of `LpPool2D` equal to the spatial size of the output of the third block, i.e., $[h_p \ w_p] = [H_4 \ W_4]$ so that the size of the output is $H_5 \times W_5 \times 3 = 1 \times 1 \times 3$, and setting its norm $p = 1$ in order to implement an average pooling.

OneNet can be instantiated by setting $h_{c_1} = w_{c_1} = 1$, $c_1 = 64$, $h_{m_1} = w_{m_1} = 8$, and $\alpha_1 = 1$ in the first block; considering $N = 3$ intermediate blocks: `Conv2D-i`$_1$ with $h_{c_{i1}} = w_{c_{i1}} = 1$, $c_{i1} = 64$, and `Mpool-i`$_1$ with $h_{m_{i1}} = w_{m_{i1}} = 8$; `Conv2D-i`$_2$ with $h_{c_{i2}} = w_{c_{i2}} = 1$, $c_{i2} = 128$, and `Mpool-i`$_2$ with $h_{m_{i2}} = w_{m_{i2}} = 1$; `Conv2D-i`$_3$ with $h_{c_{i3}} = w_{c_{i3}} = 1$, $c_{i3} = 64$, and `Mpool-i`$_3$ with $h_{m_{i3}} = w_{m_{i3}} = 1$; setting $p_d = 0.5$, $h_{c_2} = w_{c_2} = 1$, $c_2 = 3$, $h_{m_1} = w_{m_1} = 1$, and $\alpha_2 = 1$ in the third block; and setting $h_p = w_p = 6$ and $p = 1$ in the final `LpPool2D` layer so that it results in a $H_5 \times W_5 \times 3 = 1 \times 1 \times 3$ output size for the suggested $H \times W \times 3 = 384 \times 384 \times 3$ input size.

TABLE IV
CNN ARCHITECTURE OF THE ECF (WITH PROTECTIONS)

	Layer	Size	Stride	Padding	Value	Output size
	Input					$H \times W \times 3$
	Conv2D-1	$h_{c1} \times w_{c1} \times 3 \times c_1$		$[(h_{c1}-1)/2, (w_{c1}-1)/2]$		$H_1 \times W_1 \times c_1$
	PReLU				init: y = x	$H_1 \times W_1 \times c_1$
	Mpool-1	$[h_{m1}\ w_{m1}]$	s_{m1}			$H_2 \times W_2 \times c_1$
	Pow-1				α_1	$H_2 \times W_2 \times c_1$
N×	Conv2D-i	$h_{ci} \times w_{ci} \times c_1 \times c_i^{(\dagger)}$		$[(h_{ci}-1)/2, (w_{ci}-1)/2]$	init: bias = 1	$H_2 \times W_2 \times c_i$
N×	PosReLU					$H_2 \times W_2 \times c_i$
N×	Mpool-i	$[h_{mi}\ w_{mi}]$	s_{mi}			$H_3 \times W_3 \times c_i$
	Dropout				p_d	$H_3 \times W_3 \times c_i^{(\ddagger)}$
	Conv2D-2	$h_{c2} \times w_{c2} \times c_i \times 3^{(\ddagger)}$		$[(h_{c2}-1)/2, (w_{c2}-1)/2]$		$H_3 \times W_3 \times 3$
	PosReLU					$H_3 \times W_3 \times 3$
	Mpool-2	$[h_{m2}\ w_{m2}]$	s_{m2}			$H_4 \times W_4 \times 3$
	Pow-2				α_2	$H_4 \times W_4 \times 3$
	Abs					$H_4 \times W_4 \times 3$
	LpPool2D	$[h_p\ w_p]$	s_p		$p \geq 1$	$H_5 \times W_5 \times 3$

Total number of learnable params: $3h_{c1}w_{c1}c_1 + (c_1 \min(\mathrm{N}, 1) + \mathrm{c_i} \max(\mathrm{N}-1, 0)(\mathrm{h_{ci}w_{ci}c_i}) + 3\mathrm{h_{c2}w_{c2}c_i} + \mathrm{c_1} + \mathrm{Nc_i}+5$

$^{(\dagger)}$ if $\mathrm{N} > 1$ we set $c_1 = c_i$ for the occurrence(s) $n = 2, \ldots, \mathrm{N}$ of the convolutional block
$^{(\ddagger)}$ if $\mathrm{N} = 0$ we set $c_i = c_1$

B. Spatially Varying Illuminant Estimation

In all the instantiations of the CF introduced so far, the output consisted of a single illuminant estimate for the whole image. The framework, however, has the flexibility to also output a local, spatially varying illuminant estimate. This can be achieved by setting the kernel of the `LpPool2D` layer in Table IV to a value smaller than the spatial dimension of the output of the `Pow-2` layer, i.e., $h_p < H_4$ and $w_p < W_4$. The granularity of the spatially varying estimate can be therefore controlled with the kernel size $[h_p w_p]$, its smoothness can be controlled with the stride s_p.

V. EXPERIMENTAL SETUP

In this section, we introduce the image datasets used for the experiments, as well as the training details.

To test the performance of the proposed CF for the global illuminant estimation task, two standard datasets of RAW images including a known color target in each scene are used. The first one is the ColorChecker dataset [45]. Images in the dataset were captured using high-quality digital single-lens reflex (SLR) cameras in RAW format, and are therefore free of any color correction. The dataset consists of a total of 568 images. The Macbeth ColorChecker (MCC) chart is included in every scene: this allows to accurately estimate the actual illuminant of each acquired image; however, it is masked to black pixels for training and testing, in order to prevent the color constancy algorithm from directly exploiting this information. Multiple versions of the raw images and ground truth of this dataset have been proposed through the years: in this work, we use the "recommended" version by Hemrit et al. [46], [47]. All experiments were run using the original threefold cross-validation split: for each experiment we train three instances of our framework using the training and validation folds for learning, and the test fold for inference. We collect the estimations on the three test folds and consequently compute statistics on the whole dataset.

The second dataset is the NUS-8 dataset [9], which is similar to the previous one: it has been captured using digital SLR cameras in RAW format with an MCC included in every scene. Differently from the previous one it has been captured by eight different cameras and it contains a larger number of images, i.e., 1853 with around 200 images for each camera.

To test the performance of the proposed CF for the local illuminant estimation task, we select the MIRF dataset by [48]. The dataset was acquired using a high-quality digital SLR camera, it is available in RAW format, and it comes with pixel-wise ground-truth information. The dataset consists of two parts: the first one is taken in controlled laboratory settings for a total of ten scenes acquired under six distinct illumination conditions; the second one is taken in real-world settings for a total of 20 indoor and outdoor scenes. The experiments on this dataset are performed using a twofold cross-validation for each of the two acquisition settings.

Our CF for color constancy was written in PyTorch 1.7.0. The model was trained using the Adam optimizer [49]. Default learning hyperparameters have been selected as follows, although a search for optimal values is later presented in Section VI-D: learning rate 0.05, weight decay 0.0, batch size 32, and a total number of 1000 training epochs.

In order to assist the learning procedure with a limited-cardinality training set, we rely on data augmentation operations, determined after preliminary experimentation.

1) Random rotations between $-20°$and $+20°$.
2) Random translation by $\pm 10\%$ of the image size.
3) Random shear between $-10°$and $+10°$.
4) Random crop between 80% and 125% of the original image size, with aspect ratio varying between 0.75 and 1/0.75.

Images are then resized to a fixed size: a hyperparameter which is also explored in this section.

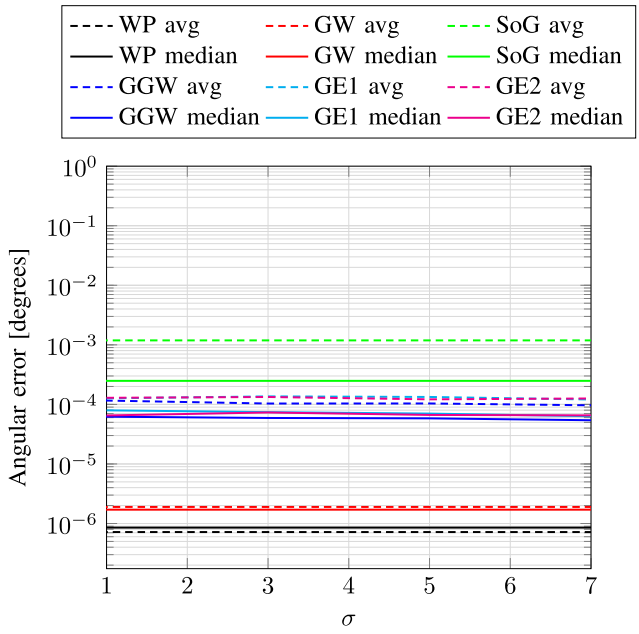


Fig. 1. Angular error between the estimates given by the low-level framework and the CF on the ColorChecker dataset.

VI. Results and Discussion

In this section, different experiments are presented. First, we perform a thorough comparison of the CF and the low-level framework in terms of the difference in the estimated illuminants, and the difference in inference speed. Then we exploit the end-to-end learning ability of the CF to perform a comparison with the state of the art for single global illuminant estimation, and multiple spatially varying illuminant estimations. In the subsequent experiment, we explore the hyperparameter space defined by the CF by a sweeping procedure, and we assess its robustness when reducing the size of the training set. Finally, we analyze the change in performance when a model ensembling technique is used.

A. Comparison With Low-Level Framework

In this section, we compare the proposed CF with the low-level framework in its original MATLAB implementation. In particular, we are interested in two aspects: first, we want to ensure that when one instance of the original low-level framework is instantiated in our CF, it produces the same illuminant estimates. Second, we want to compare the computational speed of our CF and the original low-level framework.

In order to assess the difference in the illuminant estimation provided by the low-level framework and the CF, we consider each of the six algorithms: WP, GW, SOG, GGW, GE1, and GE2, using Minkowski norm $p = 2$ and varying standard deviation σ. For the purpose of these experiments, each algorithm is executed on the whole ColorChecker dataset, downscaled with the longest side at 200 pixels, without using any mask for the exclusion of saturated pixels, and using the weights in (19)–(21) for the computation of the second-order derivative. For each image the angular error between the two implementations is computed: the plot of the average and median angular error between the corresponding estimates is reported in Fig. 1 for different values of σ.

From the reported plot it is possible to see that the difference among the estimates given by the low-level framework and the CF is always lower than 0.001°, mainly due to variations in numerical precision, and independent of the level of σ. For reference, a literature survey by [35] described a deviation of 1° in angular error with the reference illuminant to be below the threshold of what can be detected by a human being [42], and the range between 2° and 3° is considered perceivable but acceptable [50], [51].

In the same experimental configuration, we also compared the inference speed of our CF and the original low-level framework. All tests were executed on the same machine, equipped with an Intel i7-7700 CPU at 3.60 GHz and 15.6 GB of RAM, as well as an NVIDIA TITAN X (Pascal) GPU with 12 GB of dedicated memory. The low-level framework relies on optimized CPU computation and was run on MATLAB 2019a, whereas the CF exploits GPU hardware acceleration, thanks to its PyTorch-based implementation. For our experiments, we consider both single image and batch processing to provide a further layer of analysis. The results of this evaluation are plotted in Fig. 2, concerning both absolute comparison and relative speedup.

The batch-based CF is shown to outperform the low-level framework for all methods under all tested values of σ. It is only suboptimal for the simpler (and less accurate) methods WP, GW, and SoG when operating in single-image processing. Overall, the increment of inference speed consequently ranges from 2× up to 30×. Another noteworthy observation is relative to the difference in dependency on the method complexity: once convolutional operations are involved, the speed of the CF implementations of GGW, GE1, and GE2 is essentially identical, whereas the inference time of the original MATLAB implementation increases in pair with the derivative order.

A final comparison of inference speed is presented by varying the image size, ranging from 100 pixels for the longest side up to 2000 pixels per side, and setting the standard deviation σ to an intermediate value of 4. The results, which are presented in Fig. 3, corroborate the observation on the stability of the CF when varying the input image size, as opposed to the exponential behavior displayed by the original low-level framework implementation.

B. Global Illuminant Estimation Results

In this section, we compare the performance of the different versions of the CF with algorithms in the state of the art on the ColorChecker and the NUS-8 datasets. In particular, we compare the following.

1) *LCF Model A (LCF-A):* the parameters and hyperparameters are set to allow the replication of the GE2 method: $h = w = 3$ and $\alpha_1 = 1/\alpha_2 = 2$. More precisely, this allows the replication of the truncated GE2 method [43] with settings $[n, p, \sigma, t] = [n, p, \sigma, 1]$.
2) *ECF Model A (ECF-A):* the parameters and hyperparameters are set to include the previous model and extend it with a further intermediate convolutional layer:

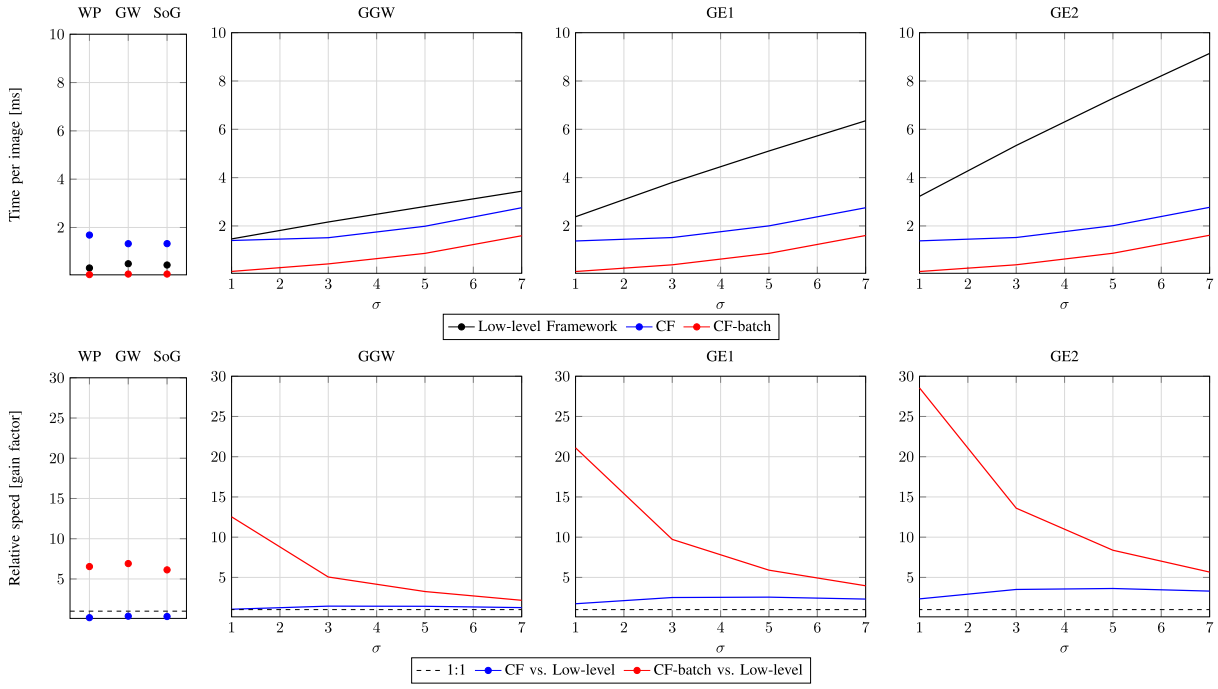


Fig. 2. Inference speed comparison between the original low-level framework and the CF by varying the adopted Gaussian standard deviation σ. Top: absolute inference time per image. Bottom: relative speed gain factor of CF over low-level framework.

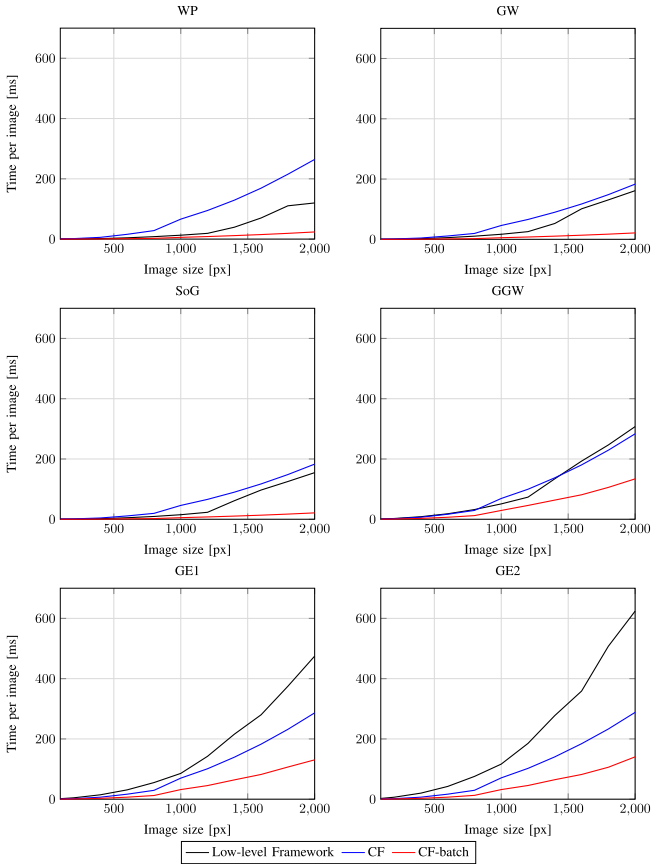


Fig. 3. Inference speed comparison between the original low-level framework and the CF by varying the image size, expressed as the longest side in pixels.

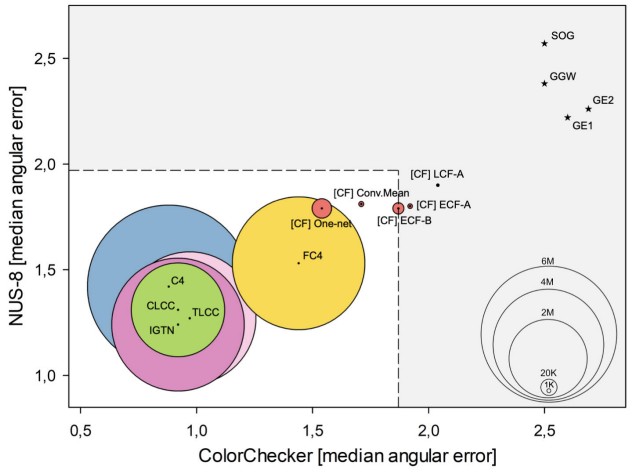


Fig. 4. Bubble plot of the median illuminant estimation error on the ColorChecker and NUS-8 datasets of instantiations of the CF (red circles), the best supervised learning methods (colored circles), and instantiations of the low-level framework (black stars). The light gray area shows the performance limit on the two datasets obtained by algorithms not based on deep learning.

$h_{c1} = w_{c1} = 3$, $c_1 = 9$, $h_{m1} = w_{m1} = 1$, $s_{m1} = 1$, and $\alpha_1 = 2$ in the first block; $N = 1$, $h_{ci} = w_{ci} = 3$, $c_i = 9$, $h_{mi} = w_{mi} = 1$, and $s_{mi} = 1$ in the second block; $p_d = 0$, $h_{c2} = w_{c2} = 1$, $h_{m2} = w_{m2} = 1$, $s_{m2} = 1$, and $\alpha_2 = 1/2$ in the third block.

We also anticipate results of the ECF model B (ECF-B) based on the selection of optimal hyperparameters as described in Section VI-D. The results of the comparison on the ColorChecker dataset are reported in Table V in terms of average and median angular error. The results of the methods in the state of the art are taken from their respective papers and may have used a different ground truth [47]. In order to allow for an easier comparison, the methods are categorized into four groups: unsupervised, parametric, supervised, and instantiations of the CF. Contrary to the common mispractice

TABLE V

GLOBAL ILLUMINANT ESTIMATION RESULTS: AVERAGE AND MEDIAN ANGULAR ERRORS OBTAINED ON THE COLORCHECKER DATASET [45]

	Algorithm	Avg	Med.
Unsuperv./S.I.	WP [4]	5.71	3.46
	GW [5]	4.72	3.57
	Buzzelli et al. (gl. norm) [52]	4.84	4.12
	Buzzelli et al. (ch. norm) [52]	5.48	4.81
	QU [53]	3.46	2.23
	SIIE [26]	2.77	1.93
	C5 [29]	2.50	1.99
Parametric	SOG $[n, p, \sigma] = [0, 4, 0]$ [6]	4.21	2.50
	GGW $[n, p, \sigma] = [0, 4, 0.1]$ [7]	4.21	2.50
	GE1 $[n, p, \sigma] = [1, 2, 0.9]$ [7]	4.03	2.60
	GE2 $[n, p, \sigma] = [2, 2, 9]$ [7]	4.23	2.69
	Truncated GGW $[n, p, \sigma, t] = [0, 4, 13, 1]$ [43]	4.17	2.49
	BP [8]	3.98	2.61
	Cheng et al. [9]	3.52	2.14
	Gray Pixel (edge) [54]	4.60	3.10
	GI [55]	3.07	1.87
Supervised	Bayesian [45]	4.70	3.44
	Spatio-Spectral (ML) [56]	3.55	2.93
	Spatio-Spectral (GP) [56]	3.47	2.90
	Natural Image Statistics [10]	4.09	3.13
	Exemplar-based [57]	2.89	2.27
	Chakrabarti (Empirical) [58]	2.89	1.89
	Chakrabarti (End-to-end) [58]	2.56	1.67
	Cheng et al. [16]	2.42	1.65
	Color Dog [59]		1.49
	Bianco et al. [19]	2.36	1.44
	FFCC (Model M) [60]	1.78	0.96
	FFCC (Model J) [60]	1.80	0.95
	FFCC (Model J)* [60]	2.23	1.45
	Oh and Kim [61]	2.16	1.47
	CCC (dist+ext) [62]	1.95	1.22
	FC^4(AlexNet) [20]	1.77	1.11
	FC^4(AlexNet)* [20]	2.14	1.44
	DS-Net (HypNet+SelNet) [21]	1.90	1.12
	QU + Fine Tuning [53]	2.91	1.98
	Convolutional Mean [39]	2.48	1.61
	APAP (GW) [63]	2.76	2.02
	Corrected-Moment (19 edge moments) [64]	2.86	2.04
	One-net (no noise) [40]	2.47	1.86
	C4 [27]	1.35	**0.88**
	IGTN (full) [28]	1.58	0.92
	CLCC (w/ Full-Aug) [30]	1.44	0.92
	TLCC [31]	**1.31**	0.97
	RCC_C [33]	2.37	1.20
	LCC (v5) [34]	2.12	1.24
Convol. Fram.	LCF-A (this work)	2.94	2.04
	ECF (this work, replica of Conv. Mean [39])	2.57	1.71
	ECF (this work, replica of One-net [40])	2.29	1.54
	ECF-A (this work)	2.74	1.92
	ECF-B (this work)	2.68	1.87
	LCF-A (this work, best run)	2.86	1.94
	ECF-A (this work, best run)	2.68	1.85
	ECF-B (this work, best run)	2.68	1.77

* models retrained according to the fair procedure [65]

TABLE VI

RESULTS FOR GLOBAL ILLUMINANT ESTIMATION: AVERAGE AND MEDIAN ANGULAR ERRORS OBTAINED ON THE NUS-8 DATASET [9]

	Algorithm	Avg	Med.
Unsuperv./S.I.	WP [4]	10.62	10.58
	GW [5]	4.14	3.20
	Buzzelli et al. (gl. norm) [52]	4.88	4.17
	Buzzelli et al. (ch. norm) [52]	4.32	3.37
	QU [53]	3.00	2.25
	SIIE [26]	2.05	1.50
	C5 [29]	1.77	1.37
Parametric	SOG [6]	3.40	2.57
	GGW [7]	3.21	2.38
	GE1 [7]	3.20	2.22
	GE2 [7]	3.20	2.26
	Truncated GE2 [43]	3.16	2.15
	BP [8]	3.17	2.41
	Cheng et al. [9]	2.92	2.04
	Gray Pixel (edge) [54]	3.15	2.20
	GI [55]	2.91	1.97
Supervised	Bayesian [45]	3.67	2.73
	Spatio-Spectral (ML) [56]	3.11	2.49
	Spatio-Spectral (GP) [56]	2.96	2.33
	Natural Image Statistics [10]	3.71	2.60
	Cheng et al. [16]	2.36	1.59
	Color Dog [59]	2.83	1.77
	Bianco et al. [19]		1.76
	FFCC (Model M) [60]	1.99	1.31
	Oh and Kim [61]	2.41	2.15
	CCC (dist+ext) [62]	2.38	1.48
	FC^4(AlexNet) [20]	2.12	1.53
	DS-Net (HypNet+SelNet) [21]	2.24	1.46
	QU + Fine Tuning [53]	1.97	1.41
	Convolutional Mean [39]	2.25	1.59
	APAP (GW) [63]	2.40	1.76
	Corrected-Moment (19 edge moments) [64]	3.03	2.11
	One-net (no noise) [40]	2.16	1.57
	C4 [27]	1.96	1.42
	IGTN (full) [28]	1.85	1.24
	CLCC (w/ Full-Aug) [30]	1.84	1.31
	TLCC [31]	1.60	1.27
	DALCC [32]	**1.42**	**1.06**
	RCC_C [33]	2.62	1.43
Convol. Fram.	LCF-A (this work)	2.84	1.90
	ECF (this work, replica of Conv. Mean [39])	2.73	1.81
	ECF (this work, replica of One-net [40])	2.48	1.79
	ECF-A (this work)	2.65	1.80
	ECF-B (this work)	2.62	1.79
	LCF-A (this work, best run)	2.75	1.80
	ECF-A (this work, best run)	2.56	1.79
	ECF-B (this work, best run)	2.50	1.73

used in the state of the art, for the CF instantiations we follow the fair comparison procedure [65] and we report the average result over five independent runs. Furthermore, we also report the best result (the only one that is commonly reported). For the sake of comparison, for some supervised methods we also report the performance obtained when they are trained using the fair comparison procedure [65].

First, we can observe that, on average, LCF-A is able to improve the best average angular error obtained by the methods included in the original low-level framework (i.e., among WP, GW, SOG, GGW, GE1, and GE2) by about 27.0%, and the best median error by 18.4%. The improvement given by ECF-A is respectively of about 32.0% and 23.2%, while the improvement given by ECF-B is of about 33.5% and 25.2%. The improvements are even higher if, instead of the average result, we consider the best run.

The experimental results on the NUS-8 dataset (Table VI) further confirm these results: on average, LCF-A is able to improve the best average angular error obtained by the methods included in the original low-level framework by about 11.3%, and the best median error by 14.4%. The improvement given by ECF-A is respectively of about 17.2% and 18.9%, while the improvement given by ECF-B is of about 18.1% and 19.4%. Also on this dataset the improvements are even higher if instead of the average result we consider the best run.

TABLE VII
RESULTS FOR LOCAL ILLUMINANT ESTIMATION: AVERAGE AND MEDIAN ANGULAR ERRORS OBTAINED ON THE TWO PARTS OF THE MIRF DATASET [48]: LABORATORY (LEFT) AND REAL WORLD (RIGHT)

Algorithm	Avg	Med	Avg	Med
DN	10.6	10.5	8.8	8.9
GW [5]	3.2	2.9	5.2	4.2
WP [4]	7.8	7.6	6.8	5.6
SOG $[n, p, \sigma] = [0, 1, 1]$ [6]	3.2	2.9	5.2	4.2
GE1 $[n, p, \sigma] = [1, 1, 1]$ [7]	3.1	2.8	5.3	3.9
GE2 $[n, p, \sigma] = [2, 1, 1]$ [7]	3.2	2.9	6.9	4.7
IEbv [48]	8.5	8.3	6.0	4.9
LSAC [68]	6.2	5.4	5.3	5.2
RETINEX [69]	6.3	5.4	5.2	5.2
LRS RETINEX [70]	5.8	4.8	5.6	4.0
Fusion Grad. Tree Boost. [71]	6.4	5.7	5.5	5.4
Fusion Rand. Forest Regr. [71]	5.0	3.9	4.1	3.5
MLS (with GE1) [72]	4.8	4.2	9.1	9.2
MIRF (with GE2) [48]	2.6	2.6	4.9	4.5
Bianco et al. [19]	**2.2**	**2.0**	**3.1**	3.0
CF-SV (GE1 $[n, p, \sigma] = [1, 1, 1]$)	2.4	2.1	5.3	3.5
LCF-A-SV (this work)	2.3	2.2	3.7	3.7
ECF-A-SV (this work)	2.3	2.1	3.4	**2.4**

Compared to supervised methods, on average the considered instantiations of the CF produce results close to or better than half of the methods in this category on both datasets, even if they are not trained using the fair comparison procedure [65]. It is worth noting that LCF-A and ECF-A have a very low number of learnable parameters, i.e., 284 and 1022 respectively. As a reference, the top five supervised methods reaching the lowest average angular errors are all based on AlexNet or SqueezeNet architectures with a total number of parameters ranging from 1.9 to 5.7 millions. The convolutional mean method [39] is also replicated within our framework, producing results comparable to the ones officially reported by the authors (the slight variation can be attributed to a different training configuration, including loss function and data augmentation). In Fig. 4, we report a bubble plot comparing the instantiations of the CF (red circles) and the best supervised learning methods (colored circles) in terms of median angular errors on the two datasets considered, with the bubble size representing the number of learnable parameters. As a further comparison we also report the performance of instantiations of the original low-level framework (black stars). We also represent with a dashed line the performance limit on the two datasets obtained by algorithms not based on deep learning. From the plot we can see how the instantiations of the CF bridge the gap in illuminant estimation accuracy between simple statistics-based algorithms and recent deep learning-based algorithms while keeping a very small model size. Qualitatively, besides the total number of learnable parameters, one of the main differences between the tested instantiations of the CF and the neural backbones used by the best supervised methods is the network receptive field [66] and its growth across the different network layers, as plotted in Fig. 5. We can notice how the backbones used by the best supervised methods show a more gradual growth of the receptive field, helping the network to build a more detailed and hierarchical representation of features.

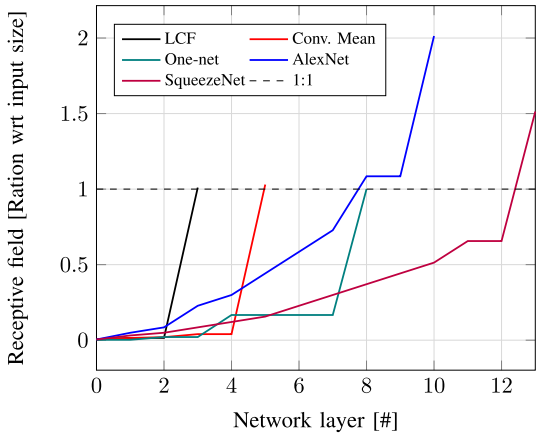


Fig. 5. Receptive field growth across the different layers of the networks considered. The receptive field is normalized with respect to the respective input size.

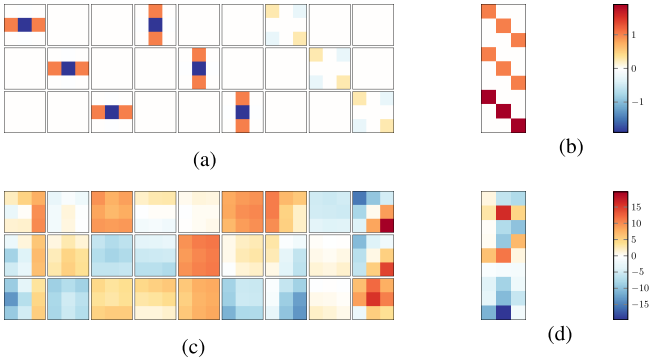
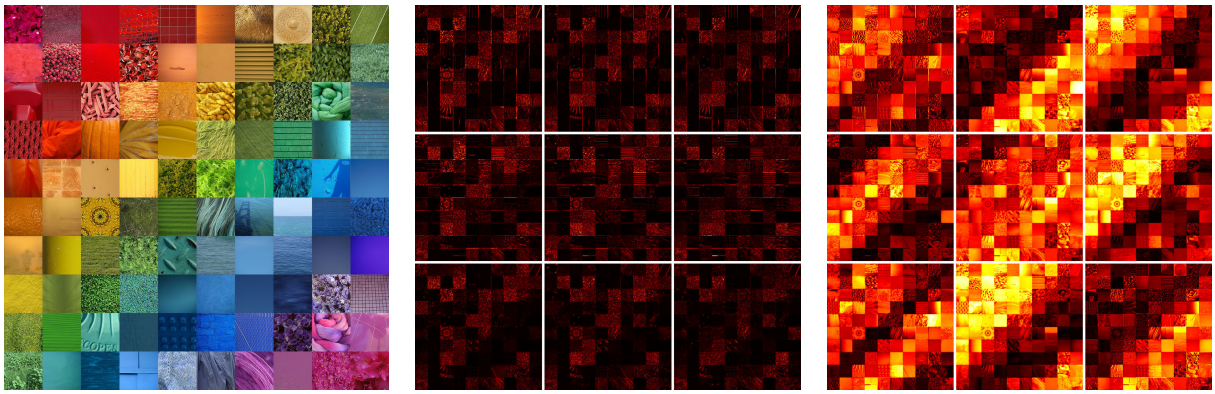


Fig. 6. Visualization of the weights of GE2 $[n, p, \sigma] = [2, 1, 0.33]$ of (a) `Conv2D-1` layer and (b) `Conv2D-2` layer; weights learned by LCF-A for (c) `Conv2D-1` layer and (d) `Conv2D-2` layer. The size of `Conv2D-1` is $3 \times 3 \times 3 \times 9$: for visualization purposes input channels are reported as rows, while output channels are reported as as columns. The size of `Conv2D-2` is $1 \times 1 \times 9 \times 3$: for better visualization it is reported as a 9×3 matrix.

Instead, the tested methods in the CF show a rapid growth of the receptive field exploiting a global pooling of fine-grained local features. We can also observe how the backbones of the best supervised methods result in a bigger receptive field, which has been observed to have a logarithmic relationship with the performance on other computer vision tasks, e.g., classification [66].

1) Model Interpretation: In this section, we analyze what is learned by a sample instance of the CF. Specifically, we inspect the weights of layers `Conv2D-1` and `Conv2D-2` of LCF-A, and we compare them with a CF replica of the GE2 algorithm with settings $[n, p, \sigma] = [2, 1, 0.33]$. The corresponding weights are reported in Fig. 6. From the `Conv2D-1` weights it is possible to verify that, by definition, GE2 has several kernels filled with zeros, and it performs the partial derivatives in each color channel independently. LCF-A instead fully exploits the possibility of performing cross-channel operations, and in fact none of its kernels are completely filled with zeroes. We can also observe how the kernels in most of the cases represent wideband filters, resembling what can be achieved with truncated filters [43]. A similar behavior can be observed in the weights of the `Conv2d-2` layer: GE2 is very sparse,



(a) (b) (c)

Fig. 7. Visualization of the activation maps of filters from the `Conv2D-1` layer on an input image. (a) Input image considered ("Rainbow" by jakerome is licensed under CC BY-ND 2.0). Activation maps, normalized in the [0, 1] range and visualized with a hot color map, of the `Conv2D-1` layer of: (b) GE2 and (c) LCF-A.

and mixes with positive weights only the entries corresponding to one color channel at a time. LCF-A instead is dense: for each of the three outputs it contains both positive and negative weights, and it exploits the information contained in all nine input channels. In Fig. 7, we report the activation maps of the `Conv2d-1` layer for both GE2 and LCF-A on a composite of multiple natural images. From the reported maps it is possible to see how `Conv2d-1` GE2, as expected, activates on image edges. LCF-A instead performs more complex operations: first, it is possible to see how the different filters perform a form of color segmentation, activating more on specific tint ranges. On top of this, some filters apply a form of blur (e.g., filters #3, #5, and #8, left-to-right, top-to-bottom), while others identify edges (e.g., vertical edges for filter #1, horizontal edges for filter #4, and both for filter #7), and others apply a form of edge enhancement (e.g., filter #9).

A visual comparison of the error distributions can also provide additional insights into the different behavior of color constancy models. By computing the ratio between ground truth and estimated illuminant, and projecting the corresponding points in a chromaticity diagram such as the angle-retaining chromaticity (ARC) [67], it is possible to observe the bias of such models in terms of their deviation from a perfect estimation (the diagram center). Fig. 8 shows that the error distribution of GGW is considerably more spread out than LCF-A. Both methods present a bias for natural lights, which is manifested in the observations being distributed along the blue-to-yellow diagonal. However, GGW is more influenced by the greenish tint of the ColorChecker RAW images, whereas LCF-A was trained to better disregard the sensor transmittance bias. Considerations on the ECF-B model are provided along with its introduction in Section VI-D.

C. Spatially Varying Illuminant Estimation Results

The performance of the spatially varying variants of the proposed framework is reported in Table VII on the MIRF dataset. To facilitate the comparison, the results are organized in three blocks: global methods, spatially varying methods, and instantiations of the CF.

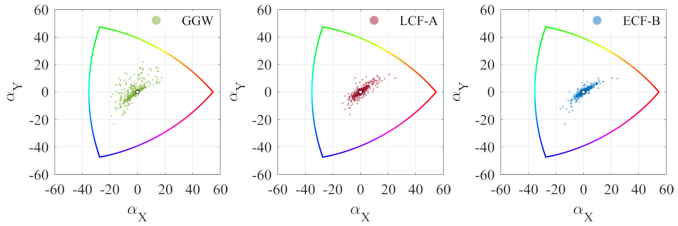


Fig. 8. Error distributions on the ColorChecker dataset for color constancy methods GGW (left), LCF-A (center), and ECF-B (right), visualized in ARC diagram. An ideal method would produce all observations in the center of the diagram.

From the reported results we can observe that replicating the GE1 algorithm in our CF and extending it to work in a spatially varying way (CF-SV) allows us to obtain results very close to the best method in the state of the art on the laboratory partition of the dataset. Taking GE1 as reference, the improvements in terms of average and median errors for the spatially varying version are respectively 22.6% and 25.0% on the laboratory partition, and 0.0% and 10.3% on the real-world partition. The kernel size of the `LpPool2D` layer is set to $h_p = w_p = 161$ with stride $s_p = 35$ for the laboratory partition of the dataset, and $h_p = w_p = 241$ with stride $s_p = 30$ for the real-world partition of the dataset. These hyperparameters were chosen by grid search using a twofold cross-validation on each partition of the dataset, and are also kept for the next experiments.

The spatially varying version of the LCF (i.e., LCF-A-SV) obtains an improvement in terms of average and median errors with respect to the standard GE1 of respectively 25.8% and 21.4% on the laboratory partition, and 30.2% and 5.1% on the real-world partition.

As expected, the spatially varying version of the ECF (i.e., ECF-A-SV) is the one that obtains the highest improvement on both dataset partitions. The gain in terms of average and median errors with respect to the standard GE1 is respectively 25.8% and 25.0% on the laboratory partition, and 35.8% and 38.5% on the real-world partition. In particular, we can observe that the median error on the real-world partition is 20.0% lower than the best method in the state of the art.

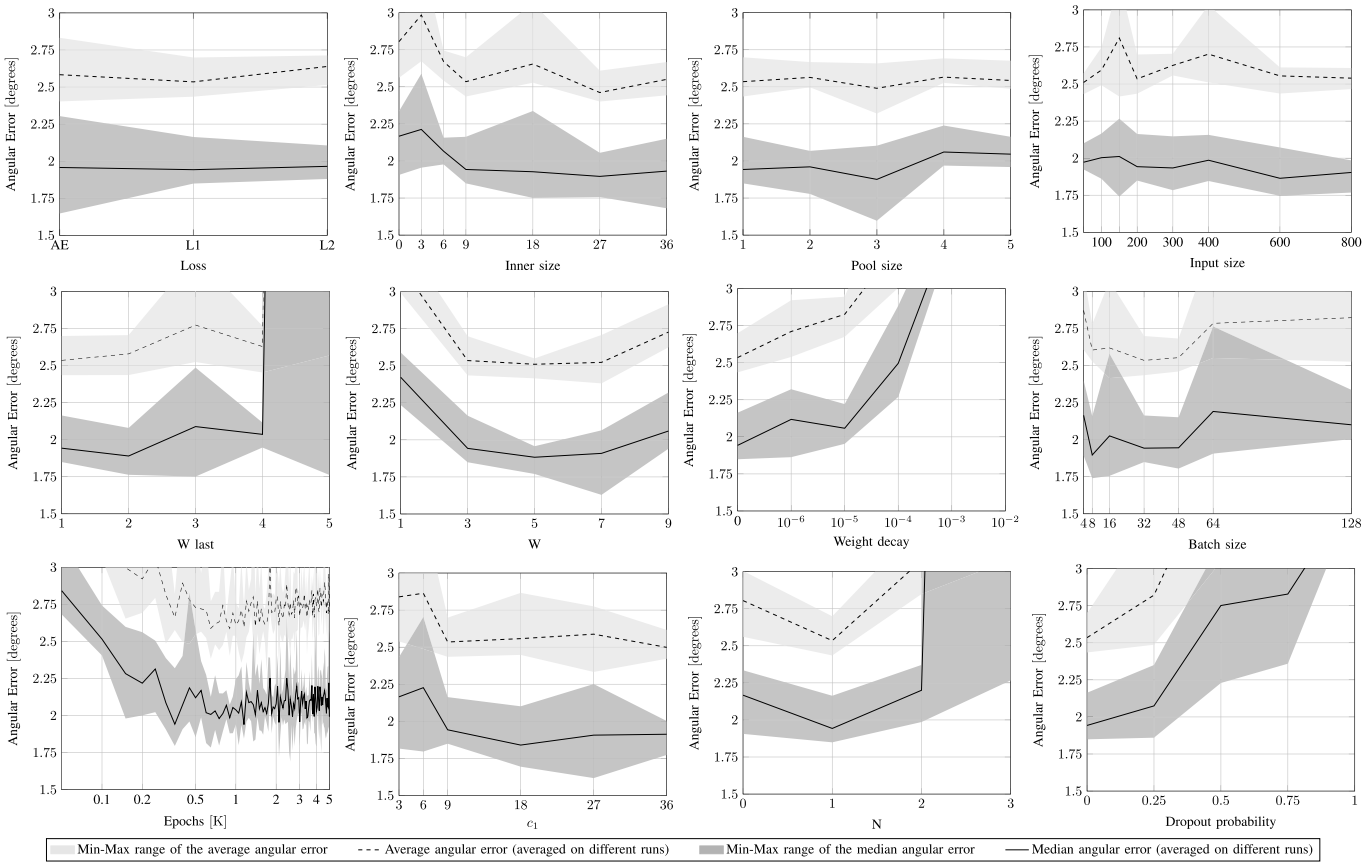


Fig. 9. Hyperparameters sweep of the ECF. For each value of each hyperparameter, we report the average, the minimum and maximum values of the average, and median angular error over five independent runs.

D. Hyperparameter Sweep

In this section, we perform an exploration of the hyperparameter space of the ECF. Starting from the model ECF-A, we sweep each of the following hyperparameters independently: loss function (angular error, L1, L2), number of output channels in the first convolutional layer (c_1), inner size (c_i), pool size ($h_{m_1} = w_{m_1} = h_{m_i} = h_{m_i} = h_{m_2} = h_{m_2}$), input size ($H = W$), W last ($h_{c_2} = w_{c_2}$), W ($h_{c_1} = w_{c_1} = h_{c_i} = w_{c_i}$), weight decay, batch size, number of epochs, and dropout probability (p_d). The values of the power in `Pow-1` and `Pow-2` are set $\alpha_1 = 1/\alpha_2 = 2$. For each hyperparameter value we perform five different runs on the ColorChecker dataset using the first fold as the test set and the second and third ones as the training set. The result of this analysis is reported in Fig. 9, where we plot the average, minimum and maximum values over the runs of the average, and median angular error.

As a proof of concept, we run a further experiment selecting the most promising values of the hyperparameters, i.e., L1 loss, $H = W = 200$, $c_1 = 9$, $c_i = 27$, $h_{c_1} = w_{c_1} = h_{c_i} = w_{c_i} = 5$, $N = 1$, $h_{m_1} = w_{m_1} = h_{m_i} = h_{m_i} = h_{m_2} = h_{m_2} = 3$, $h_{c_2} = w_{c_2} = 2$, no weight decay, batch size 8, 1000 epochs, and no dropout (i.e., $p_d = 0$). The instantiated model is named ECF-B and has a total of 7115 learnable parameters; its corresponding performance on the ColorChecker dataset is reported in Table V. From the results it is possible to notice that ECF-B is able to improve the performance of ECF-A both in terms of average and median angular error. A visual comparison among ECF-B, LCF-A, and GGW is also presented in Fig. 8: the superior performance of ECF-B is reflected in its error distribution being more compact and close to the diagram center.

As can be observed, our replicas of convolutional mean [39] and OneNet [40] achieve even better performance than our hyperparameter sweep on the ColorChecker dataset (ECF-B). This highlights how the space of possible solutions reachable by the CF has not been completely covered by our greedy optimization, suggesting the application of NAS for a more thorough optimization. This also proves the elevated potential of our framework to further improve upon its original baseline.

E. Performance Dependence on Training Set Size

To further analyze the performance of the proposed CF, we trained LCF-A and ECF-A with reduced versions of the training set of the ColorChecker dataset. The considered size ratios range from 1 corresponding to the original training set size (i.e., about 378 images, averaged over the three cross-validation folds), down to 0.01 (i.e., about four images). For each training set size, five different random selections (runs) are performed: in Fig. 10, we plot the average and median angular errors, averaged over the different runs. In the same plot, we report the best average and median errors obtained by instantiations of methods within the low-level framework, i.e., GE1 and GGW respectively. From the reported plots it is possible to see that LCF-A outperforms GGW in terms of

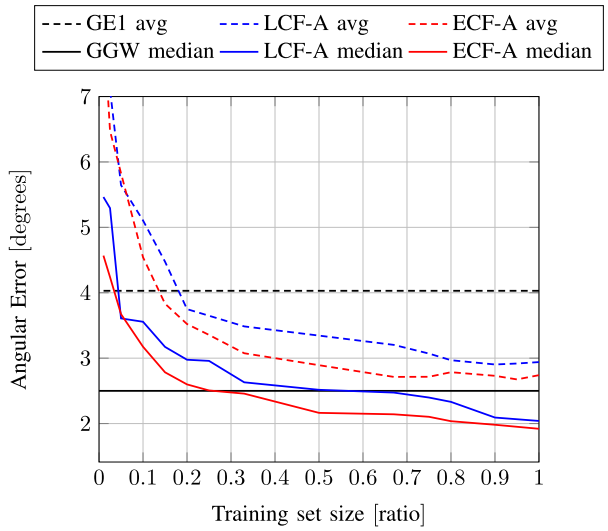


Fig. 10. Average and median angular error for LCF-A and ECF-A reducing the cardinality of the training set: 100% corresponds to about 378 images (depending on the current fold), 50% to about 189 images, and 25% to about 95 images.

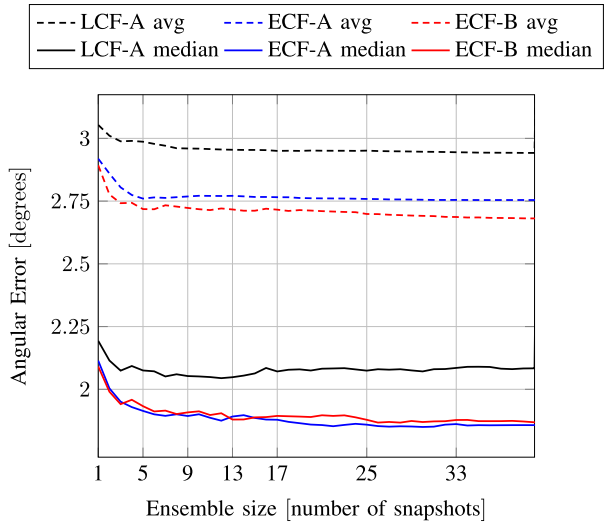


Fig. 11. Average and median angular error obtained with snapshot ensemble on LCF, ECF, and ECF. Each point represents the result obtained by averaging the predictions of the model snapshots obtained at the last epoch (i.e., 1000) with those obtained at the previous epochs.

median error even if we reduce the training set to 50% of the original size (i.e., about 189 images), and ECF-A outperforms GGW even if we reduce the training set to 25% of the original size (i.e., about 95 images). Considering the average error, LCF-A and ECF-A outperform GE1 even reducing the training set respectively to less than 20% and 15% of the original size (i.e., about 76 and 57 images respectively).

F. Model Ensembling

It is known that ensembles of neural networks are more robust and accurate than individual networks [73], but training multiple deep networks for model ensembling is computationally expensive. Inspired by snapshot ensembling [74], in this section, we aim to understand if we can obtain a more robust and accurate model by averaging the predictions of the models saved at different training epochs.

We start from the model from the last training epoch (i.e., 1000) and we average its estimations with those of the saved models back to a given epoch. During training, snapshots are created every 25 epochs and the results are averaged over five independent runs. The results of this analysis for LCF-A, LCF-A, and ECF-B are reported in Fig. 11. From the reported plots it is possible to see that for all three considered models, the angular error decreases when snapshot ensembling is performed. The largest improvement in performance occurs when the last five snapshots are used (corresponding to epochs from 1000 back to 900), after which the improvement starts to diminish. In particular, both the average and median errors converge to the respective best run values reported in Table V.

VII. Discussion and Limitations

It is possible to highlight limitations of the proposed CF for computational color constancy.

1) *Hyperparameter Optimization*: Extensive hyperparameter sweeps did not yield configurations superior to those manually determined by designers of established networks like nonvolutional mean and OneNet.
2) *Search Space Limitations*: Preliminary experiments revealed an upper bound in performance improvements when replicating the repeatable convolutional block multiple times, indicating inherent constraints within our current search space.
3) *Architectural Support*: The current framework does not account for some architectural features, such as residual blocks, known for enhancing performance in deep networks.
4) *Input Diversity*: Our framework only processes RGB images as input. However, studies have shown that incorporating chromaticity histograms [34] or edge information [60] can enhance illuminant estimation performance.

Addressing these areas will be crucial for advancing the framework's capabilities and broadening its application scope.

VIII. Conclusion

We presented a new unifying framework for color constancy that exploits the convolution operation at its core: the CF. In particular, the CF includes, improves, and extends a framework for illuminant estimation based on low-level image features.

The contribution is threefold: 1) the CF translates in terms of efficiently implemented convolution operations a low-level framework for illuminant estimation; 2) the CF is written so that it is end-to-end learnable, exploiting non-Gaussian filter kernels as well as cross-channel information, such as color derivatives; and 3) the CF is designed in such a way that it includes deeper convolutional architectures, and it can also estimate multiple spatially varying illuminants. Experiments are performed on standard datasets, and show that the CF improves the illuminant estimation accuracy in terms of average angular error of the best methods included in the low-level

framework up to about 34% for single illuminant estimation and 30% for multiple illuminant estimation, comparing favorably to other more complex supervised methods. Furthermore, the CF is able to outperform the best methods in the low-level framework even when the number of available training images is drastically reduced.

The proposed framework allows bridging the gap in illuminant estimation accuracy between simple statistics-based algorithms and recent deep learning-based algorithms while keeping a small model size and fast inference time, making it a good candidate for real-time applications such as video color constancy [75], [76]. The inference speedup gain for the most computationally intensive method in the low-level framework is up to about 30× in GPU using multibatch processing.

Due to the existing relationship among optimal filter kernel size and image input size, as future work we plan to investigate the use of a learnable resizing module, e.g., Shape Adaptor [77] or DiffStride [78]. Although we have performed a hyperparameter sweep analysis of our framework, we also plan to perform an NAS within the proposed framework to identify the best instantiations under different constraints, e.g., model size, inference speed, or illuminant estimation accuracy.

REFERENCES

[1] H. Von Helmholtz, *Physiological Optics—The Sensations of Vision, 1866, as Translated in Sources of Colour Science*, D. L. MacAdam, Ed., Cambridge, MA, USA: MIT Press, 1970.

[2] A. Gijsenij, T. Gevers, and J. van de Weijer, "Computational color constancy: Survey and experiments," *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2475–2489, Sep. 2011.

[3] D. H. Foster, "Color constancy," *Vis. Res.*, vol. 51, no. 7, pp. 674–700, 2011.

[4] E. Land and J. McCann, "Lightness and Retinex theory," *J. Opt. Soc. Amer.*, vol. 61, no. 1, pp. 1–11, 1971.

[5] G. Buchsbaum, "A spatial processor model for object colour perception," *J. Franklin Inst.*, vol. 310, no. 1, pp. 1–26, Jul. 1980.

[6] G. D. Finlayson and E. Trezzi, "Shades of gray and colour constancy," in *Proc. 12th Color Imag. Conf., Color Sci. Eng. Syst., Technol., Appl.*, 2004, pp. 37–41.

[7] J. van de Weijer, T. Gevers, and A. Gijsenij, "Edge-based color constancy," *IEEE Trans. Image Process.*, vol. 16, no. 9, pp. 2207–2214, Sep. 2007.

[8] H. R. V. Joze, M. S. Drew, G. D. Finlayson, and P. A. T. Rey, "The role of bright pixels in illumination estimation," in *Proc. Color Imag. Conf.* Springfield, VA, USA: Society for Imaging Science and Technology, 2012, pp. 41–46.

[9] D. Cheng, D. K. Prasad, and M. S. Brown, "Illuminant estimation for color constancy: Why spatial-domain methods work and the role of the color distribution," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 31, no. 5, pp. 1049–1058, 2014.

[10] A. Gijsenij and T. Gevers, "Color constancy using natural image statistics and scene semantics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 4, pp. 687–698, Apr. 2011.

[11] S. Bianco, G. Ciocca, C. Cusano, and R. Schettini, "Automatic color constancy algorithm selection and combination," *Pattern Recognit.*, vol. 43, no. 3, pp. 695–705, Mar. 2010.

[12] G. D. Finlayson, S. D. Hordley, and P. M. HubeL, "Color by correlation: A simple, unifying framework for color constancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1209–1221, Nov. 2001.

[13] V. C. Cardei, B. Funt, and K. Barnard, "Estimating the scene illumination chromaticity by using a neural network," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 19, no. 12, pp. 2374–2386, 2002.

[14] B. Funt and W. Xiong, "Estimating illumination chromaticity via support vector regression," in *Proc. Color Imag. Conf.* Springfield, VA, USA: Society for Imaging Science and Technology, Jan. 2004, vol. 12, no. 1, pp. 47–52.

[15] C. Rosenberg, M. Hebert, and S. Thrun, "Color constancy using KL-divergence," in *Proc. 8th IEEE Int. Conf. Comput. Vision. (ICCV)*, vol. 1, Jul. 2001, pp. 239–246.

[16] D. Cheng, B. Price, S. Cohen, and M. S. Brown, "Effective learning-based illuminant estimation using simple features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1000–1008.

[17] Z. Lou, T. Gevers, N. Hu, and M. P. Lucassen, "Color constancy by deep learning," in *Proc. Brit. Mach. Vis. Conf.*, 2015, pp. 76.1–76.12.

[18] S. Bianco, C. Cusano, and R. Schettini, "Color constancy using CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2015, pp. 81–89.

[19] S. Bianco, C. Cusano, and R. Schettini, "Single and multiple illuminant estimation using convolutional neural networks," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4347–4362, Sep. 2017.

[20] Y. Hu, B. Wang, and S. Lin, "FC4: Fully convolutional color constancy with confidence-weighted pooling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4085–4094.

[21] W. Shi, C. C. Loy, and X. Tang, "Deep specialized network for illuminant estimation," in *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, 2016, pp. 371–387.

[22] P. Das, A. S. Baslamisli, Y. Liu, S. Karaoglu, and T. Gevers, "Color constancy by GANs: An experimental survey," 2018, *arXiv:1812.03085*.

[23] O. Sidorov, "Conditional GANs for multi-illuminant color constancy: Revolution or yet another approach?" in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1748–1758.

[24] P. Das, Y. Liu, S. Karaoglu, and T. Gevers, "Generative models for multi-illumination color constancy," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 1194–1203.

[25] M. Buzzelli, R. Riva, S. Bianco, and R. Schettini, "Consensus-driven illuminant estimation with GANs," *Proc. SPIE*, vol. 11605, Jan. 2021, Art. no. 1160520.

[26] M. Afifi and M. S. Brown, "Sensor-independent illumination estimation for DNN models," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2019.

[27] H. Yu, K. Chen, K. Wang, Y. Qian, Z. Zhang, and K. Jia, "Cascading convolutional color constancy," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 12725–12732.

[28] B. Xu, J. Liu, X. Hou, B. Liu, and G. Qiu, "End-to-end illuminant estimation based on deep metric learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3613–3622.

[29] M. Afifi, J. T. Barron, C. LeGendre, Y.-T. Tsai, and F. Bleibel, "Cross-camera convolutional color constancy," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 1961–1970.

[30] Y.-C. Lo et al., "CLCC: Contrastive learning for color constancy," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8049–8059.

[31] Y. Tang, X. Kang, C. Li, Z. Lin, and A. Ming, "Transfer learning for color constancy via statistic perspective," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 2, pp. 2361–2369.

[32] Z. Zhang, X. Kang, and A. Ming, "Domain adversarial learning for color constancy," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 1693–1699.

[33] B. Li et al., "Ranking-based color constancy with limited training samples," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 12304–12320, 2023.

[34] M. Buzzelli, R. Schettini, and S. Bianco, "Learning color constancy: 30 years later," in *Proc. Color Imag. Conf.*, vol. 31, 2023, pp. 91–95.

[35] A. Gijsenij, T. Gevers, and M. P. Lucassen, "Perceptual analysis of distance measures for color constancy algorithms," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 26, no. 10, pp. 2243–2256, 2009.

[36] D. A. Forsyth, "A novel algorithm for color constancy," *Int. J. Comput. Vis.*, vol. 5, no. 1, pp. 5–35, Aug. 1990.

[37] G. D. Finlayson, "Color in perspective," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 10, pp. 1034–1038, Oct. 1996.

[38] B. Funt, V. Cardei, and K. Barnard, "Learning color constancy," in *Proc. Color Imag. Conf.*, 1996, pp. 58–60.

[39] H. Gong, "Convolutional mean: A simple convolutional neural network for illuminant estimation," in *Proc. 30th Brit. Mach. Vis. Conf. (BMVC)*, 2019.

[40] I. Domislović, D. Vršnak, M. Subašić, and S. Lončarić, "One-Net: Convolutional color constancy simplified," *Pattern Recognit. Lett.*, vol. 159, pp. 31–37, Jul. 2022.

[41] S. D. Hordley and G. D. Finlayson, "Re-evaluating colour constancy algorithms," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, Oct. 2004, pp. 76–79.

[42] B. Funt, K. Barnard, and L. Martin, "Is machine colour constancy good enough?" in *Computer Vision—ECCV'98: 5th European Conference on Computer Vision Freiburg, Germany, June, 2–6, 1998 Proceedings, Volume I*. Springer, 1998, pp. 445–459.

[43] S. Bianco and M. Buzzelli, "Truncated edge-based color constancy," in *Proc. IEEE 12th Int. Conf. Consum. Electron. (ICCE-Berlin)*, Sep. 2022, pp. 1–5.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[45] P. V. Gehler, C. Rother, A. Blake, T. Minka, and T. Sharp, "Bayesian color constancy revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[46] G. Hemrit et al., "Rehabilitating the ColorChecker dataset for illuminant estimation," in *Proc. Color Imag. Conf.*, 2018, pp. 350–353.

[47] G. Hemrit et al., "Providing a single ground-truth for illuminant estimation for the ColorChecker dataset," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 5, pp. 1286–1287, May 2020.

[48] S. Beigpour, C. Riess, J. van de Weijer, and E. Angelopoulou, "Multi-illuminant estimation with conditional random fields," *IEEE Trans. Image Process.*, vol. 23, no. 1, pp. 83–96, Jan. 2014.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[50] G. D. Finlayson, S. D. Hordley, and P. Morovic, "Colour constancy using the chromagenic constraint," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Oct. 2005, pp. 1079–1086.

[51] C. Fredembach and G. Finlayson, "Bright chromagenic algorithm for illuminant estimation," *J. Imag. Sci. Technol.*, vol. 52, no. 4, pp. 40906-1–40906-11, Jul. 2008.

[52] M. Buzzelli, J. van de Weijer, and R. Schettini, "Learning illuminant estimation from object recognition," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 3234–3238.

[53] S. Bianco and C. Cusano, "Quasi-unsupervised color constancy," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12204–12213.

[54] K.-F. Yang, S.-B. Gao, and Y.-J. Li, "Efficient illuminant estimation for color constancy using grey pixels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2254–2263.

[55] Y. Qian, J.-K. Kämäräinen, J. Nikkanen, and J. Matas, "On finding gray pixels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8054–8062.

[56] A. Chakrabarti, K. Hirakawa, and T. Zickler, "Color constancy with spatio-spectral statistics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 8, pp. 1509–1519, Aug. 2012.

[57] H. R. V. Joze and M. S. Drew, "Exemplar-based color constancy and multiple illumination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 860–873, May 2014.

[58] A. Chakrabarti, "Color constancy by learning to predict chromaticity from luminance," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 163–171.

[59] N. Banic and S. Loncaric, "Color dog—Guiding the global illumination estimation to better accuracy," in *Proc. 10th Int. Conf. Comput. Vis. Theory Appl.*, 2015, pp. 129–135.

[60] J. T. Barron and Y.-T. Tsai, "Fast Fourier color constancy," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 886–894.

[61] S. W. Oh and S. J. Kim, "Approaching the computational color constancy as a classification problem through deep learning," *Pattern Recognit.*, vol. 61, pp. 405–416, Jan. 2017.

[62] J. T. Barron, "Convolutional color constancy," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 379–387.

[63] M. Afifi, A. Punnappurath, G. Finlayson, and M. S. Brown, "As-projective-as-possible bias correction for illumination estimation algorithms," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 36, no. 1, pp. 71–78, 2019.

[64] G. D. Finlayson, "Corrected-moment illuminant estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1904–1911.

[65] M. Buzzelli et al., "On the fair use of the ColorChecker dataset for illuminant estimation," *IEEE Signal Process. Lett.*, to be published.

[66] A. Araujo, W. Norris, and J. Sim, "Computing receptive fields of convolutional neural networks," *Distill*, vol. 4, no. 11, Nov. 2019. [Online]. Available: https://distill.pub/2019/computing-receptive-fields

[67] M. Buzzelli, S. Bianco, and R. Schettini, "ARC: Angle-retaining chromaticity diagram for color constancy error analysis," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 37, no. 11, pp. 1721–1730, 2020.

[68] M. Ebner, "Color constancy based on local space average color," *Mach. Vis. Appl.*, vol. 20, no. 5, pp. 283–301, Jul. 2009.

[69] E. H. Land, "The Retinex theory of color vision," *Sci. Amer.*, vol. 237, no. 6, pp. 108–129, Dec. 1977.

[70] N. Banic and S. Loncaric, "Light random sprays Retinex: Exploiting the noisy illumination estimation," *IEEE Signal Process. Lett.*, vol. 20, no. 12, pp. 1240–1243, Dec. 2013.

[71] M. Bleier et al., "Color constancy and non-uniform illumination: Can existing algorithms work?" in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCV Workshops)*, Nov. 2011, pp. 774–781.

[72] A. Gijsenij, R. Lu, and T. Gevers, "Color constancy for multiple light sources," *IEEE Trans. Image Process.*, vol. 21, no. 2, pp. 697–707, Feb. 2012.

[73] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 1–18.

[74] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free," 2017, *arXiv:1704.00109*.

[75] Y. Qian, K. Chen, J. Nikkanen, J.-K. Kämäräinen, and J. Matas, "Recurrent color constancy," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5459–5467.

[76] Y. Qian, J. Käpylä, J.-K. Kämäräinen, S. Koskinen, and J. Matas, "A benchmark for burst color constancy," in *Computer Vision—ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 2020, pp. 359–375.

[77] S. Liu, Z. Lin, Y. Wang, J. Zhang, F. Perazzi, and E. Johns, "Shape adaptor: A learnable resizing module," in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*. Springer, 2020, pp. 661–677.

[78] R. Riad, O. Teboul, D. Grangier, and N. Zeghidour, "Learning strides in convolutional neural networks," 2022, *arXiv:2202.01653*.

Marco Buzzelli received the Ph.D. degree in computer science from the University of Milano-Bicocca, Milan, Italy, in 2019.

He is currently an Assistant Professor at the Department of Informatics, Systems and Communication, University of Milano-Bicocca. He is actively engaged in conducting cutting-edge research in the field of signal/image/video processing and understanding, using machine learning techniques. He is particularly passionate about color imaging. He has actively collaborated with European institutions, including but not limited to the Universitat Autònoma de Barcelona, Barcelona, Spain; the Universidade Nova de Lisboa, Lisbon, Portugal; the Université Jean Monnet, Saint-Étienne, France; and the Universidad de Granada, Granada, Spain. These collaborations have allowed him to contribute to the European AI landscape as an active ELLIS member, while also gaining valuable insights and exposure to diverse perspectives.

Simone Bianco is an Associate Professor of computer science at the University of Milano-Bicocca, Milan, Italy, holder of Italian National Academic Qualification as a Full Professor of computer engineering (09/H1) and computer science (01/B1). He is on Stanford University's World Ranking Scientists List for his achievements in Artificial Intelligence and Image Processing. He is Research and Development Manager of the University of Milano-Bicocca spin off "Imaging and Vision Solutions" and a Member of European Laboratory for Learning and Intelligent Systems (ELLIS). His teaching and research interests include computer vision, artificial intelligence, machine learning, and optimization algorithms applied in multimodal and multimedia applications.