Journal Pre-proof

Deep Reinforcement Learning based Patch Selection for Illuminant Estimation

Bolei Xu, Jingxin Liu, Xianxu Hou, Bozhi Liu, Guoping Qiu

PII: S0262-8856(19)30117-9

DOI: https://doi.org/10.1016/j.imavis.2019.08.002

Reference: IMAVIS 3798

To appear in: *Image and Vision Computing*

Received date: 15 July 2019

Accepted date: 3 August 2019

Please cite this article as: B. Xu, J. Liu, X. Hou, et al., Deep Reinforcement Learning based Patch Selection for Illuminant Estimation, *Image and Vision Computing*(2019), https://doi.org/10.1016/j.imavis.2019.08.002

Deep Reinforcement Learning based Patch Selection for Illuminant Estimation

Bolei Xu[a], Jingxin Liu[a], Xianxu Hou[a], Bozhi Liu[a], Guoping Qiu[a,b,c,d,*]

[a] *College of Information Engineering, Shenzhen University, Shenzhen, China*
[b] *Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen, China*
[c] *Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen, China*
[d] *University of Nottingham, Nottingham, United Kingdom*

Abstract

Previous deep learning based approaches to illuminant estimation either resized the raw image to lower resolution or randomly cropped image patches for the deep learning model. However, such practices would inevitably lead to information loss or the selection of noisy patches that would affect estimation accuracy. In this paper, we regard patch selection in neural network based illuminant estimation as a controlling problem of selecting image patches that could help remove noisy patches and improve estimation accuracy. To achieve this, we construct a selection network (SeNet) to learn a patch selection policy. Based on data statistics and the learning progression state of the deep illuminant estimation network (DeNet), the SeNet decides which training patches should be input to the DeNet, which in turn gives feedback to the SeNet for it to update its selection policy. To achieve such interactive and intelligent learning, we utilize a reinforcement learning approach termed policy gradient to optimize the SeNet. We show that the proposed learning strategy can enhance the illuminant estimation accuracy, speed up the convergence and improve the stability of the training process of DeNet. We evaluate our method on two public datasets and demonstrate our method outperforms state-of-the-art approaches.

Keywords: Color constancy; reinforcement learning; patch selection.

*Corresponding author
Email address: guoping.qiu@nottingham.ac.uk (Guoping Qiu)

1. Introduction

Color constancy is a kind of ability of humans to perceive the same color of a particular scene under varying illuminant. Enabling computers to have the same color constancy ability as humans has a long research history [1, 2, 3, 4, 5, 6, 7, 8, 9]. In recent years, deep learning approaches have demonstrated tremendous performance in a series of computer vision tasks [10, 11, 12], and have also been successfully applied to solve the color constancy problem [13, 14, 15, 16, 17].

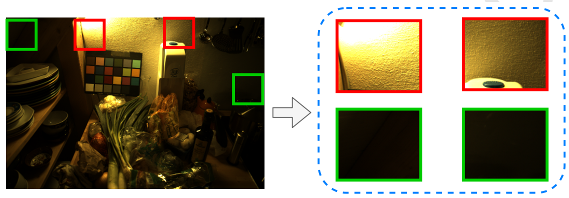


Figure 1: Two random patch cropping scenarios. **Red Bounding Box:** the illuminant in the image is not spatially uniform. **Green Bounding Box:** It is very likely the cropped patch is almost dark that can be difficult for the neural network to estimate illuminant from it.

A key step to color constancy is to estimate the color of illuminant. Although previous deep learning approaches have demonstrated significant increase in illuminant estimation accuracy, they still suffer from two main problems (examples are shown in Figure 1). Firstly, there is information loss or disturbance when constructing the inputs for the deep neural network. This is caused by either resizing the raw image to lower resolution [18] or randomly cropping (evenly partitioning) image patches [15, 19] to fit with the input size of the deep neural network. Resizing raw image would lose the detail features of the raw image, and randomly cropping image patches would has the possibility of cropping patches that could not reflect the color of the illuminant. Secondly, in the single illuminant estimation task, it is widely assumed that the illuminant is spatially uniform in the image [20] and thus all patches from the image should have the same illuminant color. However, such assumption is often violated in the real world scenario due to the occlusion of objects. Thus, it is essential to develop techniques to select optimal input patches that could increase illuminant estimation accuracy for the deep neural network.

In this paper, we regard patch selection in neural network based illuminant estimation as a controlling problem to select image patches that could increase the estimation accuracy and to remove those noisy patches that do not contribute to the estimation accuracy. This is achieved by first constructing a selection network (SeNet) to learn a patch selection policy to filter image patches in each training batch. The decision is made based on the data statistics of those patches and also the learning progress state of the deep illuminant estimation network (DeNet). In each time step, the SeNet decides which training patches should be input to the DeNet, and the DeNet then gives feedback to the SeNet to update its selection policy. To achieve such interactive training process, we utilize a reinforcement learning approach termed policy gradient to optimize the weights of SeNet. By adopting such a learning strategy, the selected patches could not only improve the estimation accuracy, but also speed up the convergent rate for the DeNet. The main contributions of our work include:

1. We present a new research perspective on illuminant estimation and highlight the importance of input selection in the illuminant estimation task, which is neglected in previous methods especially in the deep learning based approaches.
2. We propose a novel reinforcement learning based input selection mechanism for deep learning based single illuminant estimation. We show that the new method not only enhances the illuminant estimation accuracy but also accelerates the neural network's convergence speed.
3. Our approach is evaluated on the Color Checker dataset and the NUS 8-Camera datasets. On both datasets, our approach demonstrates superior estimation performances to the previous deep learning approaches.

2. Related Work

2.1. Traditional Approaches

There are mainly two lines of traditional approaches to addressing the illuminant estimation problem including statistics-based approaches and the learning-based approaches.

Statistics-based approaches are based on the assumption that the statistics of reflectance in the scene should be achromatic. A number of well-known approaches including Grey-World [2], White-Patch [3, 4] , Shades of Grey [5] and Grey-Edge [6] are based on the assumption of the scene color to be gray. The advantage of the statistics-based approaches is that they do not heavily rely on the illuminant label and also those approaches are efficient to estimate illuminations. However, the estimation accuracy of these methods is not comparable with the learning-based approaches.

The learning-based approaches usually employ labeled training data to estimate illumination. There are mainly two kinds of learning-based methods including combinatorial methods and direct methods. Combinatorial methods try to find a optimal combination of several statistics-based methods according to the scene contents of the input images. One work [21] trained a neural network to estimate illumination based on the binarized rg-chromaticity as the input. The work of [22] applies low level properties of images to select the best combination of algorithms.

Direct approaches manage to train a learning model and estimate the illumination from the training dataset. The Gamut Mapping methods assume one observes only a limited gamut of colors for a given illuminant [23]. [1, 7] first find the canonical gamut from the training data and then map the gamut of each input image into the canonical gamut. Other learning approaches such as SVR-based algorithm [24], neural networks [25], Bayesian model [8, 9] and the exemplar-based algorithm [26], usually employ hand-crafted features and the learning models are also shallow.

2.2. Deep Learning Approaches

However, with the emergence of deep learning, the deep learning approaches [10, 11, 12] are shown to achieve superior performance to the traditional hand-crafted features on a number of computer vision tasks.

There are also a number of deep learning work trying to solve the color constancy problem. One problem with the deep learning approaches is that the size of dataset is usually small, and it would lead to the over-fitting problem when training a deep neural network. To overcome this problem, [13] uses ImageNet dataset to pre-train a CNN whose ground-truths are obtained by the existing method such as Gray-of-shades. In the work of [14], they regard the color constancy as a classification problem and try to compute illuminant by finding their nearest neighbor in the training dataset. In their work, they also try to cluster the image to make the classification easier for the CNN. In those approaches, the raw images are usually resized to fit in the deep neural network to predict illuminant. However, resizing raw image would inevitable lead to information loss and thus lower the estimation accuracy.

Another way to augment the dataset size is to partition the raw image into patches. One pioneer work [15] takes raw image patches as input and directly predict illumination from a CNN. In their further work [16], they develop a multiple illuminate detector to decide whether to aggregate the local outputs into the single estimate. The author of [18] develop a fully convolutional network architecture that can take any size of input patches. In the work of [17], they also apply image patches to the network and they propose a selection network to choose an estimate from illumination hypotheses. [19] constructs a recurrent neural network to take a sequence of input image patches to estimate illuminant.

In these work, they usually randomly crop patches or evenly partition raw image into many patches. By using such patch extraction strategy, it is possible to select noisy patches that could have influence on the network's estimation ability. Thus, it is desired to develop an intelligent patch selection mechanism to select appropriate training patches that will help boost the deep neural network

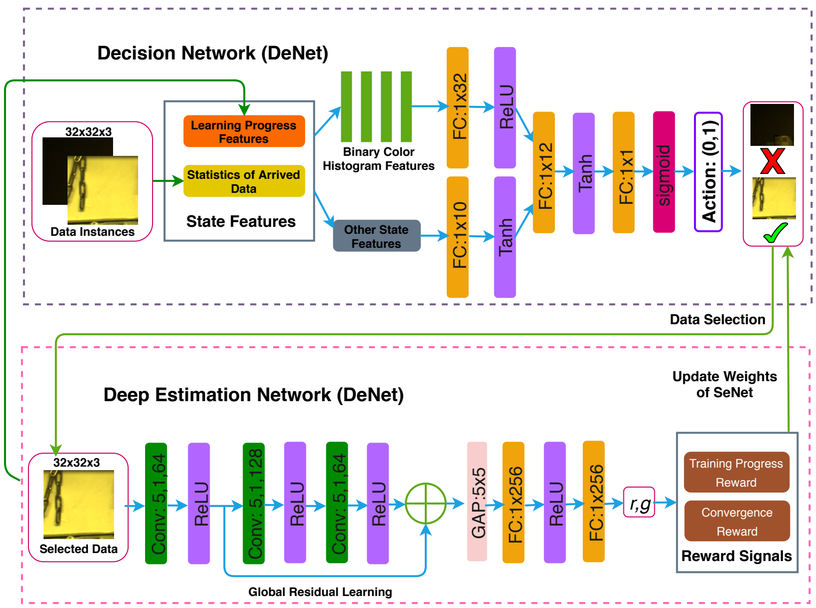


Figure 2: The overview of our patch selection framework. It contains two networks including a selection network to select appropriate patches and a deep estimation network takes these patches to estimate the illuminant value, and the deep estimation network gives reward feedback to selection network for updating its weights in each time step. In this figure, "Conv(7,1,64)" means there are 64 convolutional filters with kernel size of 7×7 and stride of 1. "FC" refers to fully-connected layers. "GAP" denotes the global average pooling layer.

ability to accurately estimate the illuminant of an image.

3. Methodology

3.1. Overview

In this paper, we formulate single illuminant estimation task as a Markov Decision Process (MDP) to find optimal input patches for the estimation task. Essentially, the MDP is constructed based on state $s \in \mathcal{S}$, action $a \in \mathcal{A}$, and reward function $R = r(s, a)$. In the problem of single illuminant estimation, the state s_t at time step t refers to the information of learning progress of deep estimation network (DeNet) and also the statistics of arrived mini-batch data $D_t = (d_1, \ldots, d_N)$. Given the state s_t, the action $a_t = \{a_t^n\}_{n=1}^N \in \{0, 1\}^N$ has to decide which patches in the arrived mini-batch data D_t are the appropriate inputs to the DeNet, where $a_t^n = 1$ means to keep the $n-$th instance in the mini-batch, and $a_t^n = 0$ means to remove the corresponding data in the mini-batch. We could further define $\pi_\theta(s_t) = a_t$ as the selection policy and we call π_θ the selection network (SeNet) that is parameterized by θ. Then, the DeNet will be trained by the mini-batch data provided by the SeNet. Finally, we will receive a reward R by executing action a_t, which will be the feedback for the SeNet to update its policy for time step $t + 1$. By formulating the illuminat estimation task as a MDP, we are able to optimize the patch selection process through a reinforcement learning approach termed *policy gradient*, which is illustrated in Section 3.6. The overview of our approach is shown in Figure 2.

3.2. State Feature Construction

Here we introduce the details of how to construct the state features in each time step. In our framework, the state features $\mathcal{S} = (\mathcal{F}_L, \mathcal{F}_D)$ is constructed to represent the learning progress of deep estimation network $\mathcal{F}_L$ and the statistics of the arrived mini-batch $\mathcal{F}_D$. In Table 1, we summarize the details of each feature.

The *learning progress features* $\mathcal{F}_L$ of DeNet are represented as: (1) the average historical training loss; (2) the best estimation accuracy so far on the

	Feature name	Length
Learning progress features	Average historical training loss	1
	Best estimation accuracy	1
	Passed iteration number	1
Statistics of arrived data	VGG-16 features	4096
	Estimated illuminant value $(\hat{r}, \hat{g})$	2
	Ground-truth value (r, g)	2

Table 1: Summary of state features to represent learning progress and statistics of arrived data.

validation dataset; (3) the passed iteration number. These are the effective features to represent the progress of DeNet. The *statistics features* $\mathcal{F}_D$ of the arrived mini-batch data are consisted of: (1) patch features extracted by the VGG-16 FC1 layer; (2) the estimated illuminant value on each data instance.; (3) its ground-truth illuminat value.

3.3. Selection Network (SeNet)

The SeNet parameterized by θ models the action policy $\pi_\theta(s_t)$ to sample actions based on the state features in each time step. As shown in Table 1, the length of VGG-16 features is much larger than the sum of the rest of the features (4096 versus 7). Thus, directly combining all the features together will cause unbalanced representation of state features. To address this unbalanced representation problem, we first re-separate state features into two groups including VGG features G and the rest state features Z: $\mathcal{S} = G \cup Z$. We then embed the VGG features through a fully-connected (FC) layer to a low dimension feature representation:

$$G'_i = \phi(\mathrm{W}_{\boldsymbol{g}}(G_i) + \mathrm{b}_g) \tag{1}$$

where $\mathrm{W}_g \in \mathbb{R}^{m \times v}$, $\mathrm{b}_g \in \mathbb{R}^v$, m is the length of VGG features, v is the length of embedded VGG features and $\phi(\cdot)$ represents *ReLU* activation function. The rest state features Z are also embedded through a FC layer with *tanh* activation function $\sigma(\cdot)$ to produce embedding features Z':

$$Z'_i = \sigma(\mathrm{W}_{\boldsymbol{z}}(Z_i) + \mathrm{b}_z) \tag{2}$$

where $W_z \in \mathbb{R}^{d \times d'}$, $b_z \in \mathbb{R}^{d'}$, d is the length of rest state features Z and d' is the length of embedding features of Z. We then combine the embedded VGG feature C' with the embedded features of the rest state features Z' to form the final embedded state features. We thus could sample actions from the final embedded state features:

$$a_i = sigmoid(W_{\boldsymbol{a}}(\phi(W_{\boldsymbol{s}}(Z'_i||C'_i) + b_s)) + b_a) \tag{3}$$

where $W_s \in \mathbb{R}^{(v+d') \times u}$, $W_a \in \mathbb{R}^{u \times 1}$, $b_s \in \mathbb{R}^u$, $b_a \in \mathbb{R}^1$, u is the length of final embedded state features and "$||$" represents the concatenation operation. The output layer uses a sigmoid function to sample actions to decide whether to keep the data instance in the mini-batch for training the deep estimation network.

3.4. Deep Estimation Network (DeNet)

After the selection process, the SeNet is able to provide a batch of selected data for the deep estimation network to estimate the illuminant. Deep estimation network formulates the illuminant estimation problem as a regression problem:

$$\hat{y}_i = \delta(d_i; \theta_r) \tag{4}$$

where $\hat{y} = \{\hat{r}, \hat{g}\}$ is the estimated illuminant color, and it is predicted by the DeNet with parameters θ_r.

The detailed structure of deep estimation network is shown in Figure 2. Based on the previous experience to construct suitable network architecture for illuminant estimation [17], we use large size of convolutional kernel (7×7 and 5×5) to capture more spatial information. Inspired by the success of ResNet [10] on feature learning, we also construct a residual block to learn more representative patch features as shown in Figure 2. The selected patches are trained by DeNet based on the mean-squared-error loss:

$$\arg\min_{\theta_r} \frac{1}{N} \sum_{i}^{N} \|y_i - \delta(d_i)\|_2^2 \tag{5}$$

where y_i is the ground-truth illuminant label, d_i is the input data.

It should be noticed that we did not apply very deep network architecture such as ResNet [10] in DeNet. Such shallow network architectures are also applied in previous deep learning based illuminant estimation work [13, 14, 15, 16, 17]. It is mainly due to two reasons: (1) the size of current color constancy datasets are usually small, and thus using very deep network would lead to the over-fitting problem; (2) although very deep networks have powerful discriminative ability, they are usually illuminant-insensitive which is not a suitable property for the illuminant estimation problem [18]. Thus we did not choose those networks with very deep structure in this work.

3.5. Reward Signal

After executing the actions, the DeNet will be trained by the selected data which are provided by the SeNet. We will then have new observation of state $\mathcal{S}_{t+1}$ and also will receive a reward signal R_t from last time step. The goal of our patch selection mechanism is to maximize the sum of the reward signal: $R = \sum_{t=1}^{T}(r_p^t + r_c^t)$, where the reward signal is designed to reflect two things: (1) the training performance r_p^t of DeNet; (2) the convergence rate r_c^t of DeNet. In this paper, we set both r_p^t and r_c^t as the terminal reward, which is calculated at time step T and set to 0 in other time steps. Specifically, they are computed in the following ways: the training performance is estimated by the average angular error on the validation set:

$$r_p^T = -\frac{1}{N}\sum_{i=1}^{N}\arccos(\frac{\hat{Y}_i \cdot Y_i}{\|\hat{Y}_i\| \cdot \|Y_i\|}). \tag{6}$$

The convergence rate estimation is set to the mini-batch index i_Γ that the validation loss is lower than a threshold Γ according to work [27]:

$$r_c^T = -\log(\frac{i_\Gamma}{T'}). \tag{7}$$

where T' is denoted as the pre-defined maximum iteration number. The final reward signal is then calculated by:

$$R = r_p^T + r_c^T \tag{8}$$

3.6. Training SeNet with Policy Gradient

We train the SeNet to learn optimal selection policy $\pi_\theta(a_t|s_{1:t})$ based on the strategy of policy gradient [28] by maximizing the expected value:

$$J(\theta) = \mathbb{E}_{p(s_{1:T};\theta)}[\sum_{t=1}^{T}(r_p^T + r_c^T)] = \mathbb{E}_{p(s_{1:T};\theta)}[R]. \tag{9}$$

To maximize J, the gradient of J is approximated by:

$$\begin{aligned} \nabla_\theta J &= \sum_{t=1}^{T}[\nabla_\theta \log \pi_\theta(a_t|s_{1:t})R] \\ &\approx \frac{1}{M}\sum_{j=1}^{M}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^j|s_{1:t}^j)R^j \end{aligned} \tag{10}$$

Although the this gradient estimator (Equation 10) can provide an unbiased estimate of gradient, it might cause high variance and unstable the training progress. One way to address this problem is to subtract a baseline from it [29]:

$$\frac{1}{M}\sum_{j=1}^{M}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^j|s_{1:t}^j)(R^j - b_t) \tag{11}$$

where b_t is the average historical reward in previous episodes. The estimation of Equation 11 has the same expectation value as Equation 10 but could has lower variance to achieve stable training performance.

3.7. Illuminant Value Inference

In the testing phase, both SeNet and DeNet are no longer updated and thus their weights are fixed. Each image in the testing dataset is evenly partitioned to form the image patches as $X_i = (d_1, \ldots, d_N)$ which is discussed in Section 4.6. Those patches are filtered by the SeNet and the selected patches will be estimated by the DeNet. The final estimated illuminant value is calculated by taking the average illuminant value over all the selected patches from a particular testing image.

4. Experiment

4.1. Experimental Setup

The following settings were used in the DeNet:

1. Data preprocessing: we crop image patches through sliding window without overlapping and padding and cropped patch size is set to 32×32.
2. The learning rate is set to 0.0003 and the batch size is set to 24.
3. We use *RMSprop* optimizer for training the estimation network.
4. The ground-truth label is converted to the normalized rg chromaticity space as: $r = R/(R + G + B), g = G/(R + G + B)$.

For the SeNet, the following settings are applied:

1. The weights are uniformly initialized between $(-0.01, 0.01)$.
2. The bias value are set to 0 in hidden layers and set to 1.5 in the output layer.
3. We use l_2 normalization to normalize input state features.
4. The batch size is set to 36, learning rate is set to 0.001 and *Adam* optimizer is applied to train SeNet.
5. It should be noticed that in some situation the SeNet might select fewer number of data instance than the batch size of DeNet. In such cases, we will wait the SeNet for gathering enough number of data to train the DeNet. It is to ensure the result of DeNet is not affected by different input batch sizes, but only influenced by the input quality.

The threshold Γ in Equation 7 is set to 0.02 which is discussed in Section 4.8 and we set pre-defined iteration number $T' = 200$. We train our approach on four Nvidia 1080 Ti GPUs and the model is implemented based on the PyTorch.

4.2. Datasets and Evaluation Metric

We evaluate the proposed patch selection framework on two widely used color constancy datasets including the reprocessed [30] Color Checker Dataset [9] and NUS 8-camera dataset [31]. The Color Checker dataset contains 568 raw images.

For the NUS 8-camera dataset, it contains 1,736 images from 8 different cameras and the experiment is done independently on each sub-dataset. Both datasets apply a Macbeth Color Checker (MCC) to obtain the ground truth illumination color. When doing experiment, we masked out the MCCs in both training and testing phase. The evaluation is done through a 3-fold cross validation for both datasets. For the NUS dataset, we calculate the performance metric by taking geometric mean over the eight image subsets as done in previous works.

A widely applied evaluation metric for illuminant estimation is through angular error computation:

$$err_{angle} = \arccos\left(\frac{\hat{Y}_i \cdot Y_i}{\|\hat{Y}_i\| \cdot \|Y_i\|}\right). \tag{12}$$

4.3. Quantitative Comparisons

In Table 2 and 3, we compared our approaches with a number of traditional approaches [2, 6, 32, 1, 5, 22, 26, 35, 38, 36], and also a series of state-of-the-art deep learning approaches [16, 17, 37, 18, 40, 39]. As shown in both tables, our approach outperformed all the other algorithms, which proved the effectiveness of the new approach. Especially when comparing with the state-of-the-art deep learning approaches such as FC4 [18], our approach demonstrates low angular error not only on the mean and media value, but also on the best and worst value. It shows that our approach is more stable to handle different scene in the dataset, which results in much lower the best and worst angular error than the previous work. It proves that our solution to more carefully select suitable input patches for the DeNet is effective.

4.4. Ablation study

We have also performed a series of ablation study to investigate the contribution of each component of our framework. We built five kinds of baseline models:

1. Ours (w/o SeNet): SeNet is removed and we use all the patches to train the DeNet.

Method	Mean	Med	Best-25%	Worst-25%
Gray World [2]	6.36	6.28	2.33	10.58
General Gray World [6]	4.66	3.48	1.00	10.09
White Patch [32]	7.55	5.68	1.45	16.12
Shades-of-Gray [5]	4.93	4.01	1.14	10.20
Spatio-spectral (GenPrior) [33]	3.59	2.96	0.95	7.61
Cheng *et al.* [31]	3.52	2.14	0.50	8.74
NIS [22]	4.19	3.13	1.00	9.22
Corrected-Moment (Edge) [34]	3.12	2.38	0.90	6.46
Corrected-Moment (Color) [34]	2.96	2.15	0.64	6.69
Exemplar [26]	3.10	2.30	-	-
Regression Tree [35]	2.42	1.65	0.38	5.87
GreyPixel [36]	3.07	1.87	0.43	7.62
CNN [16]	2.36	1.98	-	-
CCC (dist+ext) [37]	1.95	1.22	0.35	4.76
DS-Net (HypNet+SelNet) [17]	1.90	1.12	0.31	4.84
FFCC-4 channels [38]	1.78	0.96	0.29	4.29
SqueezeNet-FC4 [18]	1.65	1.18	0.38	3.78
AlexNet-FC4 [18]	1.77	1.11	0.34	4.29
FPCNet [39]	2.06	1.46	0.46	4.66
Ours (w/o SeNet)	2.26	1.57	0.46	5.72
Ours (w/o r_p^t)	2.16	1.48	0.44	5.20
Ours (w/o r_c^t)	1.72	1.06	0.33	4.18
Ours ($\mathcal{S} = \mathcal{F}_D$)	2.10	1.40	0.42	5.31
Ours ($\mathcal{S} = \mathcal{F}_L$)	1.94	1.22	0.39	5.15
Ours ($\mathcal{S} = (\mathcal{F}_L, \mathcal{F}_D)$)	**1.62**	**0.94**	**0.28**	**4.14**

Table 2: Performance of various methods on the Color Checker dataset. For metric values not reported in the literature, their entries are left blank.

Method	Mean	Med	Best-25%	Worst-25%
White-Patch [32]	10.62	10.58	1.86	19.45
Edge-based Gamut [1]	8.43	7.05	2.41	16.08
Pixel-based Gamut [1]	7.70	6.71	2.51	14.05
Intersection-based Gamut [1]	7.20	5.96	2.20	13.61
Gray-World [2]	4.14	3.20	0.90	9.00
Bayesian [9]	3.67	2.73	0.82	8.21
NIS [22]	3.71	2.60	0.79	8.47
Shades-of-Gray [5]	3.40	2.57	0.77	7.41
1st-order Gray-Edge [6]	3.20	2.22	0.72	7.36
2nd-order Gray-Edge [6]	3.20	2.26	0.75	7.27
Spatio-spectral (GenPrior) [33]	2.96	2.33	0.80	6.18
Corrected-Moment (Edge) [34]	3.03	2.11	0.68	7.08
Corrected-Moment (Color) [34]	3.05	1.90	0.65	7.41
Cheng *et al.* [31]	2.92	2.04	0.62	6.61
CCC (dist+ext) [37]	2.38	1.48	0.45	5.85
Regression Tree [35]	2.36	1.59	0.49	5.54
GreyPixel [36]	1.99	1.31	0.56	6.67
DS-Net (HypNet+SelNet) [17]	2.24	1.46	0.48	6.08
FFCC-4 channels [38]	2.12	1.53	0.48	4.78
AlexNet-FC4 [18]	2.12	1.53	0.48	4.78
SqueezeNet-FC4 [18]	2.23	1.57	0.47	5.15
DPN + DMBEN [40]	2.21	1.47	0.48	5.42
FPCNet [39]	2.17	1.57	0.51	4.88
Ours (w/o SeNet)	2.47	1.65	0.53	5.98
Ours (w/o r_p^t)	2.38	1.59	0.49	5.89
Ours (w/o r_c^t)	1.98	1.35	0.45	4.82
Ours ($\mathcal{S} = \mathcal{F}_D$)	2.43	1.52	0.49	5.76
Ours ($\mathcal{S} = \mathcal{F}_L$)	2.24	1.38	0.47	5.32
Ours ($\mathcal{S} = (\mathcal{F}_L, \mathcal{F}_D)$)	**1.94**	**1.29**	**0.43**	**4.68**

Table 3: Performance of various methods on the NUS 8-Camera dataset.

2. Ours (w/o r_p^t): r_p^t is removed from the reward function (Eq. 8).
3. Ours (w/o r_c^t): r_c^t is removed from the reward function (Eq. 8).
4. Ours ($\mathcal{S} = \mathcal{F}_L$): only use learning progress state features for the SeNet.
5. Ours ($\mathcal{S} = \mathcal{F}_D$): only use arrived data statistics state features for the SeNet.

The results are shown in Table 2 and 3. Firstly, we could see that the estimation performance drops dramatically when no selection mechanism is involved, which is caused by a number of noisy patches that are used to train the DeNet. Secondly, when evaluating the contribution of each reward signal, we could see that r_p^t is more important to lower down the estimation error, since r_p^t reflects estimation ability of DeNet. When r_p^t is absent from the reward function, the SeNet is not sure which image patches are the suitable training samples for the DeNet. In comparison, r_c^t contributes less to the estimation performance, since it only denotes the convergence rate of DeNet. However, it is still able to improve estimation accuracy, since the redundant and noisy patches could be removed from training dataset to achieve fast convergence. When comparing the contribution of different state features, it can be seen that the arrived data statistics $\mathcal{F}_D$ contributes slightly more than the learning progress features $\mathcal{F}_L$, which means the features of incoming data is crucial for SeNet to find out optimal selection policy.

4.5. Patch Size Analysis

We also investigate how the size of cropping patch would affect the estimation accuracy. The results is shown in Fig. 3. It can be seen that the best performance is reached by setting patch size to 32×32. When setting it to lower resolution (16×16), there is small increasing on the mean angular error. It is caused by its small receptive field that makes the DeNet unable to capture its detailed patch features for estimation. However, it does not mean it is always better to increase the patch size. When setting it to higher resolution (64×64 and 128×128), there is also a reduction on the estimation accuracy. The main reason is that redundant and possible noisy features could be included in the

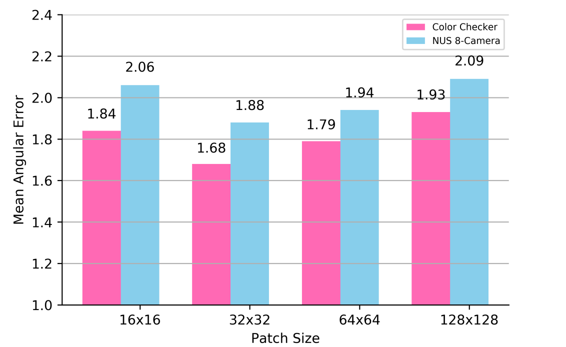


Figure 3: Comparative results of different patch sizes on mean angular error of both datasets.

	Original number	Convergence stage number
Color Checker	1,001,952	664,788
NUS 8-Camera	3,648,652	2,740,579

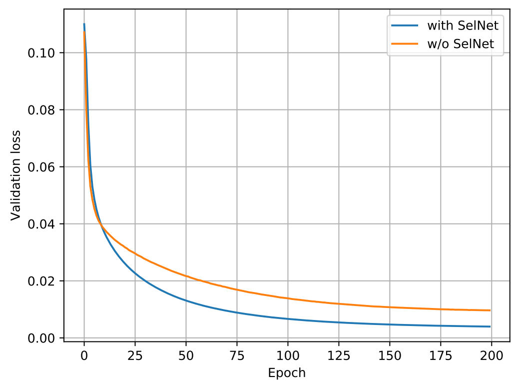
Table 4: Comparing the patch number at the beginning and at the end of convergence stage.

larger receptive field patch when there is already enough feature information for illuminant estimation.

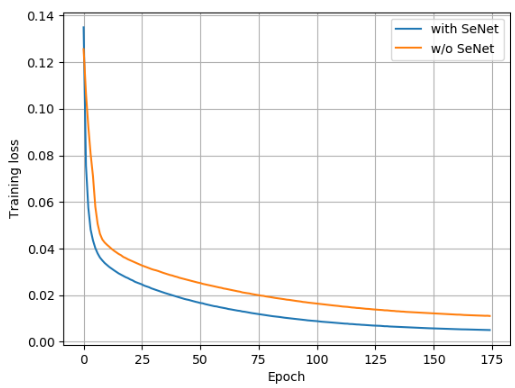
4.6. Patch Number Comparison

We extract image patches by sliding window without overlapping and padding. It results in 2,646 patches per image in the Color Checker dataset and 3,176 patches per image in the NUS 8-Camera dataset. In Table 4, we present the training size of the original patches and the number of patches selected in the convergence stage. It can be seen that the SeNet filtered out around 34% patches of the Color Checker dataset in the convergence stage, while only 25% of the patches from the NUS 8-Camera dataset are filtered out. It is caused by more severe occlusion problem that leading to the dark regions in the indoor scene in the Color Checker dataset. Those dark patches usually contains minor information about illuminant. They are considered as valueless patches in the selection process and are filtered out by SeNet in the training stage.

4.7. Convergence Analysis



(a) The convergene analysis on Color Checker dataset.



(b) The convergene analysis on NUS 8-camera dataset

Figure 4: We evaluate the effectiveness of SeNet on both datasets by calculating MSE loss on the training dataset. It can be seen that by using the SeNet to select appropriate training sets, the DeNet could achieve faster convergence than not using SeNet.

We then evaluate the effectiveness of SeNet to achieve fast convergence. We calculate the MSE loss by the end of each epoch. The results are shown in Figure 4a and 4b. We observed that on both datasets the SeNet could contribute to achieve faster convergence and lower loss. In specific, the training loss of utilizing SeNet is almost always lower than that of not using SeNet. It can also be seen that the network is able to reach convergence in around 150 epochs when SeNet is applied, while it takes more than 175 epochs when SeNet is not available.We can also observe that, in the early stage of training, the loss value is almost the same when the SeNet is applied or not, since the DeNet is not well-trained. Thus DeNet could not give accurate reward feedback to SeNet for updating its selection policy. After around 5 training epochs, the SeNet demonstrates superior ability on providing suitable training batches to increase the estimation ability for the DeNet. Finally, we can see that the DeNet is able to achieve faster convergence and lower loss value by co-operating with SeNet.

4.8. Threshold analysis

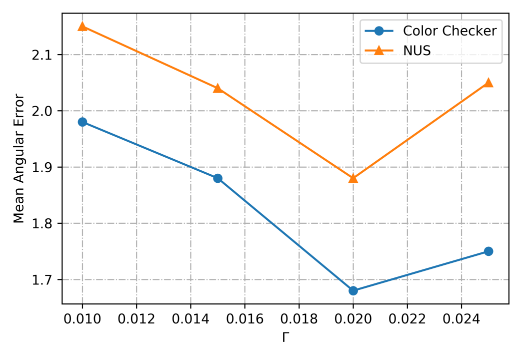


Figure 5: Comparing different mean angular errors on two datasets by setting different value of threshold in Equation 7.

We then evaluate the influence of threshold Γ in Equation 7 on the estimation performance. The result is shown in Figure 5. It can be seen that the best performance is achieved when setting $\Gamma = 0.02$ on both datasets. When setting Γ to higher value, the estimation error increases dramatically. The reason is that it is easy to achieve such high loss at the early stage of training (see Figure 4a and 4b), which reduces the incentive of DeNet to reach lower loss value. When setting Γ to smaller value, there is slightly increase on the estimation error. The reason is that it is difficult for the DeNet to achieve such low validation loss, and thus improvement between each epoch is relatively smaller. This makes it ambiguous for the SeNet to update its selection policy with such tiny changes in the reward feedback.

5. Conclusion

In this paper, we provide a new research perspective on the single illuminant estimation task, where the inputs to the deep learning model should be carefully chosen to prevent noisy inputs and information loss. To solve this problem, we propose a patch selection mechanism based on a reinforcement learning framework. A selection network uses the state features of learning progress and statistics of the arrived data to formulate a decision policy to provide the optimal inputs to the deep estimation network. By applying such patch selection strategy, the deep estimation network could achieve better estimation results and faster convergence rate than the random cropping approach in previous deep learning based solutions to this problem.

[1] K. Barnard, Improvements to gamut mapping colour constancy algorithms, in: European conference on computer vision, Springer, 2000, pp. 390–403.

[2] G. Buchsbaum, A spatial processor model for object colour perception, Journal of the Franklin institute 310 (1) (1980) 1–26.

[3] B. Funt, L. Shi, The rehabilitation of maxrgb, in: Color and Imaging Conference, Vol. 2010, Society for Imaging Science and Technology, 2010, pp. 256–259.

[4] E. H. Land, J. J. McCann, Lightness and retinex theory, Josa 61 (1) (1971) 1–11.

[5] G. D. Finlayson, E. Trezzi, Shades of gray and colour constancy, in: Color and Imaging Conference, Vol. 2004, Society for Imaging Science and Technology, 2004, pp. 37–41.

[6] J. Van De Weijer, T. Gevers, A. Gijsenij, Edge-based color constancy, IEEE Transactions on image processing 16 (9) (2007) 2207–2214.

[7] A. Gijsenij, T. Gevers, J. Van De Weijer, Generalized gamut mapping using image derivative structures for color constancy, International Journal of Computer Vision 86 (2-3) (2010) 127–139.

[8] C. Rosenberg, A. Ladsariya, T. Minka, Bayesian color constancy with non-gaussian models, in: Advances in neural information processing systems, 2004, pp. 1595–1602.

[9] P. V. Gehler, C. Rother, A. Blake, T. Minka, T. Sharp, Bayesian color constancy revisited, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.

[10] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[11] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[12] Y. Sun, X. Wang, X. Tang, Deep learning face representation from predicting 10,000 classes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1891–1898.

[13] Z. Lou, T. Gevers, N. Hu, M. P. Lucassen, et al., Color constancy by deep learning., in: BMVC, 2015, pp. 76–1.

[14] S. W. Oh, S. J. Kim, Approaching the computational color constancy as a classification problem through deep learning, Pattern Recognition 61 (2017) 405–416.

[15] S. Bianco, C. Cusano, R. Schettini, Color constancy using cnns, arXiv preprint arXiv:1504.04548.

[16] S. Bianco, C. Cusano, R. Schettini, Single and multiple illuminant estimation using convolutional neural networks, IEEE Transactions on Image Processing 26 (9) (2017) 4347–4362.

[17] W. Shi, C. C. Loy, X. Tang, Deep specialized network for illuminant estimation, in: European Conference on Computer Vision, Springer, 2016, pp. 371–387.

[18] Y. Hu, B. Wang, S. Lin, Fc 4: Fully convolutional color constancy with confidence-weighted pooling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4085–4094.

[19] Y. Qian, K. Chen, J. Nikkanen, J.-K. Kämäräinen, J. Matas, Recurrent color constancy, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5458–5466.

[20] A. Gijsenij, T. Gevers, J. Van De Weijer, et al., Computational color constancy: Survey and experiments, IEEE Transactions on Image Processing 20 (9) (2011) 2475–2489.

[21] B. Funt, V. Cardei, K. Barnard, Learning color constancy, in: Color and Imaging Conference, Vol. 1996, Society for Imaging Science and Technology, 1996, pp. 58–60.

[22] A. Gijsenij, T. Gevers, Color constancy using natural image statistics and scene semantics, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (4) (2011) 687–698.

[23] D. A. Forsyth, A novel algorithm for color constancy, International Journal of Computer Vision 5 (1) (1990) 5–35.

[24] B. Funt, W. Xiong, Estimating illumination chromaticity via support vector regression, in: Color and Imaging Conference, Vol. 2004, Society for Imaging Science and Technology, 2004, pp. 47–52.

[25] R. Stanikunas, H. Vaitkevicius, J. J. Kulikowski, Investigation of color constancy with a neural network, Neural Networks 17 (3) (2004) 327–337.

[26] H. R. V. Joze, M. S. Drew, Exemplar-based colour constancy., in: BMVC, 2012, pp. 1–12.

[27] Y. Fan, F. Tian, T. Qin, X.-Y. Li, T.-Y. Liu, Learning to teach, arXiv preprint arXiv:1805.03643.

[28] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning 8 (3-4) (1992) 229–256.

[29] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in neural information processing systems, 2000, pp. 1057–1063.

[30] L. Shi, Re-processed version of the gehler color constancy dataset of 568 images, http://www. cs. sfu. ca/˜ color/data/.

[31] D. Cheng, D. K. Prasad, M. S. Brown, Illuminant estimation for color constancy: why spatial-domain methods work and the role of the color distribution, JOSA A 31 (5) (2014) 1049–1058.

[32] D. H. Brainard, B. A. Wandell, Analysis of the retinex theory of color vision, JOSA A 3 (10) (1986) 1651–1661.

[33] A. Chakrabarti, K. Hirakawa, T. Zickler, Color constancy with spatio-spectral statistics, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (8) (2012) 1509–1519.

[34] G. D. Finlayson, Corrected-moment illuminant estimation, in: Computer Vision (ICCV), 2013 IEEE International Conference on, IEEE, 2013, pp. 1904–1911.

[35] D. Cheng, B. Price, S. Cohen, M. S. Brown, Effective learning-based illuminant estimation using simple features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1000–1008.

[36] Y. Qian, J.-K. Kamarainen, J. Nikkanen, J. Matas, On finding gray pixels, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 8062–8070.

[37] J. T. Barron, Convolutional color constancy, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 379–387.

[38] J. T. Barron, Y.-T. Tsai, Fast fourier color constancy, in: IEEE Conf. Comput. Vis. Pattern Recognit, 2017.

[39] J. Zhang, Y. Cao, Y. Wang, C. Wen, C. W. Chen, Fully point-wise convolutional neural network for modeling statistical regularities in natural images, arXiv preprint arXiv:1801.06302.

[40] F. Wang, W. Wang, Z. Qiu, J. Fang, J. Xue, J. Zhang, Color constancy via multibranch deep probability network, Journal of Electronic Imaging 27 (4) (2018) 043010.