DECOUPLING SEMANTIC CONTEXT AND COLOR CORRELATION WITH MULTI-CLASS CROSS BRANCH REGULARIZATION

Vishal Keshav, Tej Pratap GVSL

Samsung Research Institute Bangalore, INDIA.
{vishal.k1, tej.gvsl}@samsung.com

ABSTRACT
This paper presents a novel design methodology for architecting a light-weight and faster DNN architecture for vision applications. The effectiveness of the architecture is demonstrated on Color Constancy (Auto White Balance) use case, an inherent block in camera and imaging pipelines. Specifically, we present a multi-branch architecture that disassembles the contextual features and color properties from an image, and later combines them to predict a global property (e.g. Global Illumination). We also propose an implicit regularization technique by designing a cross-branch regularization block that enables the network to retain high generalization accuracy. With a conservative use of best computational operators, the proposed architecture achieves state-of-the-art accuracy with 30X less model parameters and 70X faster inference time for color constancy. It is also demonstrated that the proposed architecture achieves similar efficiency in case of other low-level image restoration problems such as Low-Light enhancement.

Index Terms— Illumination estimation, Low Light photography, Multi-branch architecture, Regularization with soft-parameter sharing

1. INTRODUCTION

In the contemporary world, it has become ubiquitous to realize most of the vision-based tasks with Deep Neural Networks (DNNs) to achieve higher accuracy. This success led to the wide applicability of DNNs for the camera, image, and video applications like color constancy[1], image de-noising[2], low light enhancement[3] and image de-hazing[4]. Since color is an important cue in many such vision applications, it is essential to provide images in their true colors for better accuracy. Digital cameras that act as eyes in vision tasks are deficient in, Color Constancy, an inherent property of human visual system due to which the perceived color of the objects remains constant even under varying illumination conditions. Therefore it is implicit to state that, to achieve color constancy in digital image inputs, illumination estimation is an important problem to be addressed.

2. COLOR CONSTANCY AND RELATED WORK

2.1. Color constancy problem formulation

A sample digital image in RGB color space can be simply modeled as the product of the pixels reflectance and the illumination present in it, as shown below in eq. (1)

\[ I_{rgb} = W_{rgb} \times L_{rgb} \]  

where \( I_{rgb} \) is the (r, g, b) tuple corresponding to each pixel, \( W_{rgb} \) is the white balanced or the true color (r, g, b) tuple of each pixel and \( L_{rgb} \) is the global illumination common across all the pixels in the image.

A true colored or white balanced image can therefore be reproduced by, first estimating the illumination present in an image and then discounting it. Once the illumination is known, the white balanced image can be derived from eq. (1), as shown below in eq. (2)

\[ W_{rgb} = \frac{I_{rgb}}{L_{rgb}} \]

Hence the efficiency of any Color Constancy algorithm is a measure of, how accurately it can estimate the illumination in a given image. The most commonly used error metric to measure the efficiency of a color constancy algorithm is the “angular error”, which is defined in eq. (3)

\[ \theta = \cos^{-1}\left(\frac{\langle gt, et \rangle}{\| gt \|_2 \cdot \| et \|_2}\right) \]

where \( gt \) and \( et \) are the ground truth illumination and the estimated illumination by an algorithm respectively.

In the following subsections, we categorize and discuss existing illumination estimation methods and differentiate the proposed method against them.

2.2. Statistical methods

Most of the statistical methods[5] that model the color constancy task assume some regularity among the pixel colors or intensities under natural lighting conditions. For example, the gray world approach[6] assumes that the average surface reflectance in an image is gray and hence the color of illumination is the deviation from gray. Van De Weijer et al. in [7]
summarize most of the statistical methods in one equation. The statistical-based methods work well on data which satisfies the prior assumptions. Their performance deteriorates exponentially when those assumptions tend to fail, resulting in poor accuracy.

2.3. Learning based methods

Barron in [8] introduced a discriminative learning based method wherein the problem of illumination estimation is reduced to a 2D spatial localization task. The technique, however, is only applicable to solve color constancy. Deep learning based methods: CNN based model proposed by Binaco et al. in [9] showed that CNNs are capable of capturing the distribution of how a natural image looks like, more importantly, the global illumination present in it. But this model is proven to be very compute intensive. An advanced version of CNN based model, DS-NET proposed by Shi et al. in [10] makes use of two interactive sub-networks to solve the Color Constancy problem in a better way. Its first sub-net (Hyp-Net) generates two hypothesis for illumination estimation and the second sub-net (Sel-Net) adapts to select one among them. This design increased the accuracy but resulted in a heavier model with many parameters. Hu et al. [11] solved the problem by capturing the semantic details from the input image and introduced a novel pooling method called weighted pooling to achieve a better estimate of global illumination. It masks the estimated illuminations with the learned weight map. The weight map is learned in relative to the confidence of respective image portions contributing to the illumination. It, however, re-uses existing models like AlexNet [12] and SqueezeNet [13] that are well proven for classification tasks and hence is heavier.

In contrast, our work formulates Color Constancy task as two independent learning sub-tasks, in a multi-class learning context, that helped to achieve state-of-the-art accuracy with 30x lesser model parameters and 70x faster inference time.

The main contributions of this paper are as follows:

1. We proposed an efficient multi-branch architecture that independently learns the spatial contextual relationship of objects using depth-wise convolutions in one branch and color correlation among the pixels using point-wise convolutions in another branch. The outputs from these two branches when combined to achieve a common objective, resulted in better accuracy and faster inference.

2. We introduced an implicit regularization strategy based on soft parameter sharing between the two branches of the proposed multi-branch architecture to improve upon generalization accuracy.

3. We demonstrated the applicability of proposed architecture on a class of computer vision problems such as color constancy and low-light photography.

The claims are defended with visual and empirical results.

3. PROPOSED METHOD

This section details the proposed baseline multi-branch architecture. An overview of our approach is shown in Fig. 1. The problem of illumination estimation has been modeled as extracting color properties present at the low-level image pixels (f in Fig. 1) and masking those with contextually rich image regions (g in Fig. 1) that provides useful semantic information. The masked color representation is further transformed to match the required output dimension. We also present the implicit regularization technique used to retain high generalization accuracy.

3.1. Extracting Color property

We denote the color property extraction operation by function \( f : \mathbb{R}^{(h \times w \times 3)} \rightarrow \mathbb{R}^{(h \times w \times k)} \) where \( h < H, w < W \) and \( k >= 3 \). \( f \) transforms the input image tensor from RGB space to an intermediate embedding space. It outputs a reduced spatial dimension tensor that has per-pixel color correlation in the input image. In order to approximate \( f \), we use the point-wise convolution operator. The notation of \( f \) is formalized in the below equations.

\[
\text{PointwiseConv}(W, y)_{i,j} = \sum_{m} M \cdot y_{i+s,j+s,m} \quad (4)
\]

where \( y \) is an input tensor having spatial dimension of \((K, L)\) and depth dimension of \( M \). \( s \) is the stride used in the convolution. A choice of \( s = 2 \) reduces the spatial dimension to \((K/2, L/2)\). \( W \) contains the learn-able weights with dimension \((1, 1, M)\).

We apply \( 2 \times M \) point-wise convolution with learn-able weights \( \{W^1, W^2, ..., W^{2 \times M}\} \) that produces an output tensor with dimension \((K/2, L/2, 2 \times M)\). A unit block that extracts color properties present in the input tensor is then given by eq. (5)

\[
f_c(y) = \sigma(\text{PointwiseConv}(W^{1 \times 2M}, y)) \quad (5)
\]
where $\sigma$ is a non-linearizing unit ReLU. Using eq. (5), $f$ is then defined in eq. (6)

$$f(I) = f^N(I)$$ (6)

where $I$ is the input image and $N$ is the number of repeated units of color property extraction blocks.

3.2. Extracting Semantic Map

We define the function $g : \mathbb{R}^{(H \times W \times 3)} \rightarrow \mathbb{R}^{(h \times w \times k)}$ that generates the contextual mask based on the input image. It can be observed that to extract the contextual information, depth dimension is not required. Instead, we focus only on spatial dimension and hence utilize depth-wise convolution operator for extracting the spatial contextual information. The notion of $g$ is formalized in the below equations.

$$DepthwiseConv(W, y)_{i,j,k} = \sum_{p,q} W_{p,q,k'} y_{i+p,j+q,k'}$$ (7)

where $y$ is the input tensor and $W$ contains the learn-able weights with dimension $(P, Q)$. $k$ varies from 1 to $2 \times K$. $k' = [k/2], k'' = k_{mod}(k, 2)$ or in other words, we have set the depth-multiplier of depth-wise convolution to 2. $h_s$ denotes a unit block that extracts semantic information from the input tensor, defined in eq. (8)

$$h_s(y) = \lambda(\sigma(\text{DepthwiseConv}(W^{1:2M}, y)))$$ (8)

where $\lambda$ is the pooling unit AVGPOOL. The output dimension of the embedding is $(K/2, L/2, 2 \times M)$. The semantic map generation function is then defined by eq. (9)

$$g(I) = h^N_s(I)$$ (9)

3.3. Channel-wise weighted pooling

In order to apply the semantic map on the color embedding, we propose a novel pooling technique called channel-wise weighted pooling. Like weighted pooling in [11], the proposed channel-wise weighted pooling does not constraint the number of weighting masks to one, instead it provides a mask for each channel. Also the number of color channels can be more than three, which is later reduced to match the required output channel dimension. Intuitively, this gives each of the learned color properties more flexibility to select image regions before aggregating for a global property. The masked output $O$ is the element-wise product of the signals from $f$ and $g$ respectively and is shown in eq. (10).

$$O^k_{ij} = S^k_{ij} \ast C^k_{ij}$$ (10)

$S = f(I)$ and $C = g(I)$ are the respective output embedding tensors with spatial dimension along $(i, j)$ and depth dimension along $k$.

The output embedding $O$ from eq. (10) is transformed to the required output dimension with a spatial reduction followed by a depth reduction. The reduced output is then normalized to obtain illumination estimation as shown in eq. (13)

$$O^k = \sum_{i=1}^{N} \sum_{j=1}^{M} O^k_{ij}$$ (11)

$$O^i_{\text{reduced}} = \sum_{k=0}^{(i+1) \times K/3} O^k, i = 0, 1, 2$$ (12)

$$(I_r, I_g, I_b) = \text{normalization}(O_{\text{reduced}})$$ (13)

The proposed baseline architecture is shown in Fig. 2 below. The architecture is end-to-end trainable and does not require supervisory signals for $S$ and $C$ independently. The output $(I_r, I_g, I_b)$ is optimized with respect to the ground truth. With the above formulation, each of the branches learns a specific function, namely semantic feature in branch-1 and color correlation information in branch-2.

![Fig. 2](image_url)

Overview of the Baseline dual branch architecture. Input image is convolved with depth-wise convolution along branch-1 and point-wise convolution along branch-2. The shape of intermediate tensors is kept same in both branches to retain Fully Convolution Property (FCN). Spatial dimension is halved and depth dimension is doubled at every layer.

3.4. Implicit regularization with soft parameter sharing

In the formulation above, $g$ in eq. (9) can be thought of as an auxiliary task in a multi-class learning problem [14] that aids in selecting the best regions for each color representation extracted by $f$ as shown in eq. (6). Intuitively, the weighted mapping helps in focusing on regions that have rich contextual information while obscuring irrelevant regions before estimating for a global property. However, the two signals produced from $f$ and $g$ are not completely independent, they are loosely dependent, as they are required to learn for a common objective jointly. Having these two tasks learn their specific signals independently result in overfitting problems since the available datasets are not large enough.

In general, to combat the overfitting problem, regularization techniques such as $L1$ regularization or $L2$ regularization, among others can be used. These are explicit regulariza-
tion techniques that add a regularizer term in the optimization function.

In contrast, we present a novel micro-architecture design for regularizing baseline multi-branch architecture. The proposed design blocks are shown in Fig. 3. Sharing parameters between two signal extractors $f_S$ and $f_C$ is a result of our observation that imposing assumptions of task dependencies in the architecture provide some inductive bias.

![Fig. 3. Proposed two designs to achieve regularization. Parameters between two task specific branches are shared as shown in designs A and B. Both the design blocks A and B has input specific depth-wise and point-wise operators. In design A, convolution operator is used for parameter sharing, whereas in design B, the multiply-broadcast operator is used. Unlike the convolution unit in design A, the scalar weights in design B imposes strict constraints. Inputs of shape $(H, W, D)$ is transformed to $(H/A, W/4, A*D)$ by each such regularizing units. For better depiction, non-linearization and pooling are not shown.](image)

In other words, these architectural changes work because our re-constructed hypothesis space $H$ puts constraints on the estimators $f$ and $g$. The constraint is soft parameter sharing between two branches, and thus it effectively shrinks the class $H$. The network is trained in an end-to-end manner and thus the optimizer searches automatically through $H$ for good estimators $f$ and $g$, or identifies how much to share at which layer. We have formalized the regularization with soft parameter sharing In eq. (14) through (18).

$$\begin{bmatrix} o'_s \\ o'_c \end{bmatrix} = \begin{bmatrix} w_s & w_{sc} \\ w_{sc} & w_c \end{bmatrix} \odot \begin{bmatrix} i_s \\ i_c \end{bmatrix}$$

(14)

where $i_s$ and $i_c$ are input signals of tasks specific to spatial contextual information and color representation respectively. $w_s$ and $w_c$ are learnable parameter corresponding to input signals $i_s$ and $i_c$ respectively. $w_{sc}$ is a shared parameter that combines $i_s$ and $i_c$ signals and $o'_s$ represents the combined output signal. $\odot$ denotes a convolution operator in Design-A or a scalar broadcasting product in Design-B. Additions are performed with element-wise addition operator. Branch specific computations are specified by (15) and (16).

$$o'_s = h^2_s(i_s)$$

(15)

$$o'_c = f^2_c(i_c)$$

(16)

where $o'_s$ represents output signals corresponding to the branch specific tasks $h_s$ and $f_c$ as defined in eqs. (9) and (6) respectively.

Output of the design block (A or B) $o_s$ and $o_c$ is then given by eqs. (17) and (18).

$$o_s = o'_s \odot o''_s$$

(17)

$$o_c = o'_c \odot o''_c$$

(18)

where $\odot$ is a concatenation operator along the depth dimension. Our different design strategies are discussed below:

- **Parameter sharing with a convolution operator (Design A):** Convolution operators linearly transform the two signals. If there is no sharing of parameters between two input signals, then $w_{sc}$ will be a zero tensor. With this design, we achieve state-of-the-art accuracy on color constancy task with far lesser computation time.

- **Parameter sharing with scalar weights (Design B):** The scalars scales the input signals. If there is no sharing of parameters, then $w_{sc}$ will be a zero scalar. This design achieves an accuracy which is under visual acceptable limits, but is the lightest and fastest.

In our two designs A and B, we observe the below two architectural characteristics:

1. Soft parameter sharing that helps the architecture in reducing generalization error.

2. Merging of task specific signals helps in easy flow of gradients. They mimic the behavior of residual connections[15].

As shown by He. et. al. [15], a residual connection breaks the output $H(x)$ to $F(x) + x$ or $F(x) + Wx$. Under this settings, eqs. (19) and (20) formalize the outputs from both designs.

$$\text{out} = F(\text{in}) + \text{in}$$

(19)

$$\text{out} = F(\text{in}) + W \cdot \text{in}$$

(20)

where $F(\text{in}) = \text{DepthwiseConv}(W, h_s(\text{in}))$ if the input is a spatial signal, else $F(\text{in}) = \text{PointwiseConv}(W, f_c(\text{in}))$.

Eq. (19) and (20) are two special cases which may arise in eq. (14). We observe that when when $w_{sc}$ is a zero kernel (Design A) or a zero weight scalar (Design B), the case is covered by the eq. (20). Over the previous condition, when $w_s$ and $w_c$ are identity kernel in Design-A or a unit scalar in Design-B, then this case is shown by eq. (19).

Next section details the improvement results on color constancy and low-light photography with the above discussed design choices.
4. EXPERIMENTS

4.1. Experiments and evaluation on color constancy

This section evaluates the proposed method in terms of efficiency and accuracy for the color constancy task on Cube [16] and NUS-8 [17] data sets. Our baseline architecture and its regularized versions as shown in Fig. 2 and Fig. 3 are implemented in the TensorFlow framework. The details of our architecture are described in the Table 1.

Table 1. Architecture details for color constancy problem.

| Input-Output Tensor | Tensor shape | Operation          | Strides | Filters |
|---------------------|--------------|--------------------|---------|---------|
| Input               | (224,224,3)  | Depthwise, Pointwise | 2       | 12      |
| Tensor1a, Tensor1b | (112,112,12) | Design A or B       | —       | —       |
| Tensor2a, Tensor2b | (28,28,48)   | Design A or B       | —       | —       |
| Tensor3a, Tensor3b | (7,7,192)    | Channel-wise weighting | —     | —       |
| Tensor4             | (7,7,192)    | Spatial reduction   | —       | —       |
| Tensor5             | (1,1,192)    | Depth reduction     | —       | —       |
| Output tensor       | (1,1,3)      | —                  | —       | —       |

The training was done in an end-to-end manner on a workstation with Nvidia GeForce GTX-1080Ti GPUs. We use Adam optimizer with a batch size of 32, and a learning rate of $5 \times 10^{-4}$ for all of the designs. No explicit regularization techniques such as $L_1$ or $L_2$ have been used. For optimization, we used Mean Squared Error (MSE) of normalized ground truth illumination and the estimated illumination predicted by the network, while for performance evaluation, we use angular error as given by eq. (3).

We augment data on both Cube and NUS-8 datasets, by randomly cropping and flipping the images along horizontal and vertical axis. All images in test and training sets are gamma corrected with a $\gamma$ of $1/2.2$. The Cube portion and color checker from Cube and NUS-8 datasets respectively are masked, for both training and testing.

Table 2. Results on NUS-8 data set. Number of parameters are given in Millions. Floating point operations are given in Giga-Flops. DS-Net* is evaluated with an input shape of $(47, 47, 2)$ (result from DS-Net [10]) while others are evaluated with an input shape of $(224, 224, 3)$.

| Models            | Mean | Median | Tri-mean | Best 25% | Worst 25% | Params | Flops |
|-------------------|------|--------|----------|----------|-----------|--------|-------|
| Gray-world        | 3.75 | 2.91   | 3.15     | 0.69     | 8.18      | —      | —     |
| Color Tiger       | 2.94 | 2.59   | 2.66     | 0.61     | 5.88      | —      | —     |
| Restricted Color Tiger | 1.64 | 0.82   | 1.05     | 0.24     | 4.37      | —      | —     |
| Baseline          | 1.701 | 1.111 | 1.276    | 0.345    | 4.003     | —      | —     |
| Design A          | 1.616 | 1.09   | 1.242    | 0.318    | 3.76      | —      | —     |

Table 3. Comparison study of Color Constancy on Cube data set. As compared to baseline multi-branch architecture, regularization using Design A improves the accuracy significantly.

| Models            | Mean | Median | Tri-mean | Best 25% | Worst 25% |
|-------------------|------|--------|----------|----------|-----------|
| Gray-world        | 4.14 | 3.2    | 3.39     | 0.9      | 9         | —       |
| DS-Net            | 2.24 | 1.46   | 1.68     | 0.48     | 5.28      | 2.64    | 0.031* |
| FC4-alex          | 2.12 | 1.53   | 1.67     | 0.48     | 4.78      | 2.9     | 1.2    |
| FC4-squeeze       | 2.23 | 1.57   | 1.72     | 0.47     | 5.15      | 1.23    | 0.77   |
| Design A          | 2.102 | 1.654 | 1.72     | 0.576    | 4.469     | 0.13    | 0.11   |
| Design B          | 2.442 | 1.871 | 1.956    | 0.67     | 5.283     | 0.04    | 0.04   |

With three-fold cross validation, we compare our results with other methods on standard metrics such as mean, median, tri-mean, mean of the lowest 25%, and mean of highest 25%. Results with NUS-8 dataset are presented in Table 2. It is to be noted that, as compared to state of the art learning based methods such as FC4 [11], accuracy is comparable but the number of computation cycles is reduced by 70 times. Visual comparison results are shown in Fig. 4.

Table 3. Comparison study of Color Constancy on Cube data set. As compared to baseline multi-branch architecture, regularization using Design A improves the accuracy significantly.

Effect of our designs on regularizing the baseline model is being reflected in the results as shown in Table 3. Better results on mean of worst 25% metric (hard to learn examples) in both the tables shows the robustness of our model due to regularization method. The difference in accuracy between baseline and regularized versions defends our analysis. For even better efficiency, quantization and sparsity reduction methods can be used. The inference time of our model on a single threaded ARM based platform running at 2.1 GHz is 30ms as compared to 100ms for FC4-SqueezeNet model. Hence, the proposed model can be very well used to realize real time mobile based applications.
4.2. Feasibility demo for low-light photography

This section demonstrates how the proposed architecture can address low-light enhancement use-case and also compares the results against state-of-the-art Learning to See in the Dark [18] (LTSID) method.

We utilize the developed design blocks A or B (in Fig. 3) to replace the convolution units in the encoder part of the UNET architecture used by LTSID. The intermediate output from convolution block (design A) or multiply-broadcast block (design B) is passed to the decoder network of the UNET architecture used by LTSID. The intermediate output of the block (design A) or multiply-broadcast block (design B) is passed to the decoder network of the UNET architecture for multi-scale context aggregation. With this, we achieved the depicted visual accuracy as shown in Fig. 5 on See-in-the-Dark (SID) dataset with 50% lesser parameters.

![Sample visual results on SID dataset](image)

**Fig. 5.** Sample visual results on SID dataset [18]. (a)-Input Image – captured in Extreme-Low-Light with exposure of 0.1sec, (b)-Ground Truth Image – captured with exposure of 10sec, (c)-LTSID method output and (d)-Proposed Method output

5. CONCLUSION

In this paper, we present an efficient multi-branch architecture for a class of low level image restoration problems such as color constancy and Low-Light-enhancement. It utilizes depth-wise convolution to learn semantic features and point-wise convolution to learn color correlation. With channel-wise weighted pooling layer, we combine the two signals in order to predict a global property. To improve the accuracy of our method, we introduce an implicit regularization technique based on multi-task soft parameter sharing. The experimental results confirm that such design choices lead to superior accuracy.

6. REFERENCES

[1] David A Forsyth, “A novel algorithm for color constancy,” *International Journal of Computer Vision*, vol. 5, no. 1, pp. 5–35, 1990.
[2] Harold C Burger, Christian J Schuler, and Stefan Harmeling, “Image denoising: Can plain neural networks compete with bm3d?,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2392–2399.
[3] David Mayer, “Learning to see in the dark,” *Nineteenth Century Theatre*, vol. 25, no. 2, pp. 92, 1997.
[4] Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao, “Dehazenet: An end-to-end system for single image haze removal,” *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5187–5198, 2016.
[5] Rafael C Gonzalez and Rechard E Woods, “Histogram processing,” *Digital Image Processing*, vol. 3, pp. 162–165, 2008.
[6] Jun-yan Huo, Yi-lin Chang, Jing Wang, and Xiao-xia Wei, “Robust automatic white balance algorithm using gray color points in images,” *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 541–546, 2006.
[7] Joost Van De Weijer, Theo Gevers, and Arjan Gijsenij, “Edge-based color constancy,” *IEEE Transactions on image processing*, vol. 16, no. 9, pp. 2207–2214, 2007.
[8] Jonathan T Barron, “Convolutional color constancy,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 379–387.
[9] Simone Bianco, Claudio Cusano, and Raimondo Schettini, “Color constancy using cnns,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 81–89.
[10] Wu Shi, Chen Change Loy, and Xiaoou Tang, “Deep specialized network for illuminant estimation,” in *European Conference on Computer Vision*. Springer, 2016, pp. 371–387.
[11] Yuanning Hu, Baoyuan Wang, and Stephen Lin, “Fc4: Fully convolutional color constancy with confidence-weighted pooling,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR17)*, 2017, pp. 4085–4094.
[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[13] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and; 0.5 mb model size,” arXiv preprint arXiv:1602.07360, 2016.
[14] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil, “Multi-task feature learning,” in *Advances in neural information processing systems*, 2007, pp. 41–48.
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[16] Nikola Banic and Sven Loncaric, “Unsupervised learning for color constancy,” in *VISIGRAV*, 2018.
[17] Dongliang Cheng, Dilip K Prasad, and Michael S Brown, “Illuminant estimation for color constancy: why spatial-domain methods work and the role of the color distribution,” *JOSA A*, vol. 31, no. 5, pp. 1049–1058, 2014.
[18] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun, “Learning to see in the dark,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.