Addressing Illumination-Based Domain Shifts in Deep Learning

A Physics-Based Approach

Attila Lengyel

Addressing Illumination-Based Domain Shifts in Deep Learning

A Physics-Based Approach

by

Attila Lengyel

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday, 26 August 2019 at 10:00 AM.

Student number:	4220455
Project duration:	1 November 2018 – 26 August 2019
Thesis committee:	Prof. dr. M. J. T. Reinders, TU Delft, supervisor
	Dr. J. C. van Gemert, TU Delft, daily supervisor
	Dr. K. A. Hildebrandt, TU Delft
	Prof. M. Milford, Queensland University of Technology

TU Delft

Preface

Before you lies the MSc thesis *Addressing Illumination-Based Domain Shifts in Deep Learning - A Physics-Based Approach.* The core of this document is a scientific paper on integrating prior knowledge from physics-based reflection models into neural networks to improve their robustness to changes in illumination. Specifically, the resulting method is applied to nighttime semantic segmentation for autonomous driving. The chapters following the scientific paper include supplementary background information and additional methods and experiments. This thesis has been written to fulfill the requirements to obtain the degree of Master of Science at the Delft University of Technology and as such marks the end of an exciting 7-year journey.

The research for this thesis has been conducted at the Australian Centre for Robotic Vision in Brisbane, Australia, one of the world's leading research centers in the new field of robotic vision. First of all, I would like to thank Prof. Michael Milford for giving me the opportunity to spend the last 8 months of my studies, which in fact felt closer to 8 weeks, at such an amazing place. Australia will always hold a special place in my heart and I am sure to return one day.

I would also like to thank Dr. Jan van Gemert for proposing this research topic, which gave me a nice insight into both the practical applications of deep learning as well as the - largely still unknown - theory behind it, and his continuous guidance throughout the entire project. It has been an honest introduction to scientific research, including all of its ups and downs. Thank you for the invaluable feedback on my work and for getting me back on track again whenever I got lost. Also many thanks to the other members of the thesis committee, Prof. dr. Marcel Reinders and Dr. Klaus Hildebrandt, for taking the time to assess my work, and to Sourav and all my other wonderful and most knowledgeable colleagues at ACRV who always stood ready to tackle any challenge I presented them.

This journey would not have been the same without my friends, who made my student years in Delft, my board year with the ETV and my time abroad in both Trondheim and Brisbane unforgettable. Special thanks to my parents for their endless support and motivation, and for encouraging my interest and curiosity for science and technology from the first moment that I can remember.

Finally, I want to thank Nena, whom I was lucky enough to meet shortly after the beginning of this journey. Thank you for all the adventures we went through together and for always being there for me, both in good times and in bad.

Attila Lengyel
Delft, August 2019

Contents

1

Scientific paper

Addressing Illumination-Based Domain Shifts in Deep Learning: a Physics-Based Approach

Attila Lengyel
Delft University of Technology
Delft, The Netherlands
a.lengyel@tudelft.nl

Abstract

This work investigates how prior knowledge from physics-based reflection models can be used to improve the performance of semantic segmentation models under an illumination-based domain shift. We implement various color invariants as a preprocessing step and find that CNNs trained on these color invariants get stuck in worse local minima compared to RGB inputs, but can achieve comparable or even superior performance when applying knowledge transfer from RGB. We also find Batch Normalization to severely affect the performance of neural networks under an illumination-based domain shift and demonstrate that Instance Normalization offers a simple remedy to this issue. Additionally, we investigate different fusion models for combining color invariants with RGB. Using a combination of these methods we achieve a 14.5% performance increase on nighttime semantic segmentation without any additional training data.

1. Introduction

Recent technological advances have enabled a large variety of deep learning driven real-world applications. A commonly encountered problem in many applications is domain divergence, which occurs when the data used for training the neural network does not sufficiently resemble the test data observed during operation and consequently leads to performance degradation. Domain divergence is often caused by varying illumination conditions that alter the visual appearance of objects [47], including time of day and weather based appearance changes in outdoor robotics [48, 49]. Domain divergence is generally dealt with by collecting more and more diverse training data but this may not be practical when data is too expensive to obtain or to annotate. Domain adaptation methods typically offer a solution by training a model on a source data distribution that performs well on a similar, but different target data distribution [47]. However, these methods still require data from the target domain, either labeled or unlabeled.

A different approach for addressing an illumination-based domain shift is the use of photometric invariant features, or color invariants, which represent properties of objects irrespective of their recording conditions [15]. These features are derived from the physical nature of objects in color images and are invariant to one or more factors influencing the recording conditions. The three most important factors include the scene geometry, which affects the formation of shadows and shading on objects, the object surface properties, which defines whether or not the object exhibits interface reflections (highlights) next to Lambertian (matte) reflections, and the illuminant color, which alters the overall color of the scene. Color invariants have been used successfully in classical computer vision applications [2, 33, 15], but their use in a deep learning setting has so far remained unexplored. In this paper, we investigate the intrinsic robustness of convolutional neural networks to illumination-based domain shifts and explore several methods to incorporate the uncertain prior knowledge from these physics-based photometric invariants to improve the generalization capabilities of a CNN under such domain shift. We propose to directly feed color invariant representations of images into the input layer of a neural network. Different from other domain adaptation methods, this approach does not require any data from the target domain. Specifically, we apply these methods to perform semantic segmentation on nighttime images, which is a key computer vision task for autonomous driving.

It is well-known that photometric invariance generally comes at the loss of some discriminative power [12]. As such, RGB images inherently contain more information, enabling stronger CNNs to be trained compared to color invariants. We therefore investigate transfer learning through both knowledge distillation [17, 45] and fine-tuning to improve the performance of a color invariant segmentation network using a CNN model trained on RGB images. Motivated by the same trade-off between invariance and discriminative power, we also investigate the joint use of both RGB and color invariants to get the best of both worlds by comparing different fusion architectures.

The main contributions of this paper are (i) a thorough investigation into the illumination robustness of different CNN architectures and individual layers, (ii) a comparison of different fusion architectures for the joint use of RGB images and color invariants, and (iii) a novel day-to-night domain adaptation method using color invariants and transfer learning to improve the target domain performance of a segmentation network trained only on data from the source domain.

2. Related work

Domain Adaptation Domain adaptation aims to train a model on a source domain dataset that performs well on a different but similar target domain dataset. This alleviates the burden of annotating datasets for computer vision applications in new domains where insufficient training data is available. Popular approaches include reducing feature divergence between the source and target domain through an additional discrepancy metric [42, 31] or adversarial loss [19, 41] in the backpropagation loss function, and using generative adversarial networks to perform synthetic style transfer between the two domains [18] or to generate synthetically labelled target data [50]. [28] argues that domain-specific knowledge is stored in the batch normalization [20] layers of a model and that domain adaptation can be performed by replacing the pre-trained batch normalization statistics by the statistics of the target domain. For the specific application of nighttime semantic segmentation, Dark Model Adaptation [6] applies gradual day-to-night fine-tuning on a segmentation model pre-trained on daytime data, whereas GCMA [35] uses corresponding day-night image pairs to learn a style transfer, which is then applied to generate a dark stylized version of a daytime dataset used to train the final segmentation model. All aforementioned methods require training data from the target domain, either labelled or unlabelled, whereas our approach achieves improved performance by using only source domain data.

Color invariants The use of physics-based reflection models to achieve invariance to lighting geometry and illuminant color is a well-researched topic in classical computer vision. Early work includes invariants derived from the Dichromatic Reflection Model [36] on the one hand [11], and from the Kubelka-Munk reflection model [26] on the other. Based on a new camera sensor response model introduced in [9] many methods have been proposed for shadow removal or intrinsic image decomposition [10, 8] with applications in localization [5, 33], road detection [1, 2, 25, 23] and street image segmentation [44]. In this work, we investigate the use of a subset of these color invariants in a deep learning setting.

Physics-Guided Neural Networks Several works have investigated how physical models can be used to improve the performance or robustness of a neural network without the help of additional training data. Applications include lake temperature modeling [22] and shape from polarization [3], where the output of a physics-based model is used together with the raw data as input features for the neural network. In [39], a neural network is trained on unlabeled data to track objects in free fall and pedestrians walking across the frame by regularizing the loss function with physics laws and other domain constraints. Deep Lagrangian Networks [32] integrates prior knowledge from Lagrangian Mechanics into the network topology to learn more accurate and robust models of mechanical manipulators for trajectory tracking. This work follows the line of extracting features from the input data using physical models and using a combination of the extracted features and raw images to train a neural network.

3. Method

In this section we first give a short recap on the derivation of the relevant color invariants and their properties. We then briefly describe the transfer learning methods used to improve our color invariant segmentation networks. Finally, we discuss the fusion techniques for combining different inputs.

3.1. Color invariants

In this work we limit ourselves to color invariants originating from the Dichromatic Reflection Model (DRM) [36] and the Kubelka-Munk (KM) reflection model [26].

3.1.1 DRM based color invariants

The general model for the RGB responses of a camera is defined as

$$f^c(\vec{x}) = \int_\lambda E(\lambda, \vec{x})\rho^c(\lambda)d\lambda, \tag{1}$$

where λ denotes the wavelength of the light, $c \in \{R, G, B\}$ denotes the color channel, $E(\lambda, \vec{x})$ is the energy reflected from an object's surface and $\rho^c(\lambda)$ is the spectral sensitivity function of the sensor, tuned to the wavelength corresponding to channel c. The reflected energy is modeled using the DRM, which is defined as

$$E(\lambda, \vec{x}) = \left(m^b(\vec{x})s(\lambda, \vec{x}) + m^i(\vec{x})\right) e(\lambda, \vec{x}), \tag{2}$$

where $m^b(\vec{x})$ and $m^i(\vec{x})$ are scale factors depending on the scene geometry for Lambertian and interface reflections, respectively, $s(\lambda, \vec{x})$ denotes the surface albedo and $e(\lambda, \vec{x})$

the spectral distribution of the light source, which is often assumed to be white and spatially uniform across the scene and hence e is constant. Substituting the DRM into equation 1 and assuming only matte reflections ($m^i(\vec{x}) = 0$), the ratio of two color channels $\frac{f^i(\vec{x})}{f^j(\vec{x})}$ factors out the dependence on scene geometry such that we can define the normalized RGB color invariant as

$$\{r, g, b\} = \{R, G, B\} / (R + G + B + \epsilon), \tag{3}$$

where ϵ is a small constant added for numerical stability. The ratio of differences $\frac{f^i(\vec{x})-f^j(\vec{x})}{f^k(\vec{x})-f^j(\vec{x})}$ also factors out interface reflections, such that we can define the dichromatic color invariant as

$$\{d_R, d_G, d_B\} = \left\{(R-G)^2, (G-B)^2, (B-R)^2\right\} / I \tag{4}$$

with $I = (R-G)^2 + (G-B)^2 + (R-B)^2 + \epsilon$.

Both normalized RGB and the dichromatic color invariant assume white illumination, which is highly optimistic in any real-world scenario. Based on the von Kries model [46], which describes the relationship between sensor responses and the illuminant color, comprehensive image normalization [11] (Comp) defines an iterative process of pixel intensity normalization (3) and color channel normalization that factors out the dependency on both scene geometry and illuminant color.

3.1.2 Kubelka-Munk based color invariants

Kubelka-Munk based color invariants are derived directly from the KM reflection model without considering the sensor response model from equation 1. The KM reflection model is defined as

$$E(\lambda, \vec{x}) = e(\lambda, \vec{x})(1 - \rho_f(\vec{x}))^2 R_\infty(\lambda, \vec{x}) + e(\lambda, \vec{x})\rho_f(\vec{x}) \tag{5}$$

where $e(\lambda, \vec{x})$ denotes the spectral density of the illuminant, $\rho_f(\vec{x})$ the interface reflection coefficient of the material and $R_\infty(\lambda, \vec{x})$ the material reflectivity, which is a property similar to the albedo. Assuming a white illuminant and dropping $(\lambda, \vec{x})$ from the notation, equation 5 reduces to

$$E = e(\vec{x})\left\{\rho_f(\vec{x}) + (1 - \rho_f(\vec{x}))^2 R_\infty(\lambda, \vec{x})\right\} \tag{6}$$

with first and second order derivatives with respect to λ

$$E_\lambda = e(\vec{x})(1 - \rho_f(\vec{x}))^2 \frac{\partial R_\infty(\lambda, \vec{x})}{\partial \lambda}, \tag{7}$$

$$E_{\lambda\lambda} = e(\vec{x})(1 - \rho_f(\vec{x}))^2 \frac{\partial^2 R_\infty(\lambda, \vec{x})}{\partial \lambda^2}. \tag{8}$$

Then the ratio

$$H = \frac{E_\lambda}{E_{\lambda\lambda}} \tag{9}$$

Table 1: Overview of color invariants and their invariance properties with respect to scene geometry (G), interface reflections (I) and illumination color (C). None of the representations is invariant to all influences. The mapping functions ensure a $[0, 1]$ output range, $|\cdot|$ denotes normalization.

	Mapping	**G**	**I**	**C**
RGB	$\lvert\cdot\rvert$	✗	✗	✗
r, g, b	-	✓	✗	✗
D_R, D_G, D_B	$\lvert\cdot\rvert$	✓	✓	✗
Comp	$f(x) = \frac{1}{1+e^{-(x-1)}}$	✓	✗	✓
H	$f(x) = \frac{1}{1+e^{-x}}$	✓	✓	✗
C_λ	$f(x) = \frac{1}{1+e^{-2x}}$	✓	✗	✗
N_λ	$f(x) = \frac{11x}{1+10x}$	✓	✗	✓

is an invariant for scene geometry under both matte and interface reflections and

$$C_\lambda = \frac{E_\lambda}{E} \tag{10}$$

for matte reflections only. If the power spectrum of the illuminant is not necessarily uniform but independent of the position in the scene, then the spatial derivative

$$N_{\lambda\vec{x}} = \frac{\partial}{\partial x}\left\{\frac{E_\lambda}{E}\right\} = \frac{E_{\lambda\vec{x}}E - E_\lambda E_{\vec{x}}}{E^2} \tag{11}$$

is an invariant to both scene geometry and illuminant color for matte surfaces. The spatial derivative is calculated by convolution with Gaussian derivative filters and the gradient magnitude is simply given by $N_\lambda = \sqrt{N_{\lambda x}^2 + N_{\lambda y}^2}$. Based on the Gaussian color model [14], the linear transformation in equation 12 is used to estimate the reflected energy E and its first- and second-order derivatives E_λ and $E_{\lambda\lambda}$ from RGB camera responses.

$$\begin{pmatrix} \hat{E} \\ \hat{E}_\lambda \\ \hat{E}_{\lambda\lambda} \end{pmatrix} = \begin{pmatrix} 0.0024 & 0.0252 & 0.0108 \\ 0.0119 & 0.0017 & -0.0140 \\ 0.0137 & -0.0240 & 0.0066 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{12}$$

Mapping functions are used when necessary to ensure all output values lie in the $[0, 1]$ range. An overview of the discussed color invariants and corresponding properties and mapping functions is shown in table 1.

3.2. Transfer learning approach

We explore knowledge distillation and fine-tuning as two approaches to transfer knowledge from the more informative RGB network to the less informative but more robust color invariant network.

Our knowledge distillation approach involves a typical teacher-student setup where the color invariant student network is trained using both the ground-truth labels of the dataset y and the soft targets produced by the RGB teacher network y_{soft}. The soft targets are calculated by softening the logits z_i (the outputs of the last layer before activation) using the softmax function defined as

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \tag{13}$$

where q_i denotes the output probability for class i and T is the temperature, which is normally set to 1. Higher values of T produce "softer" output probabilities that are more uniformly distributed, whereas a low temperature increases the differences between the output probabilities. The loss function is a weighted average of the cross entropy loss of the soft targets and the cross entropy loss of the ground-truth labels and is defined as

$$\mathcal{L}_{KD} = L(y_{\text{soft}}, \hat{y}_{\text{soft}}) + \lambda L(y, \hat{y}). \tag{14}$$

The predicted label $\hat{y}$ is calculated by softening the student network logits with $T = 1$, whereas for the soft prediction $\hat{y}_{\text{soft}}$ the logits are softened using the same temperature as used for generating the soft targets from the teacher network. T and λ are additional hyperparameters that need to be defined prior to training. Since the gradients with respect to the loss, as used in backpropagation to update the weights, produced by the soft targets scale with $1/T^2$, λ is often also set to a value of approximately $1/T^2$ to ensure both terms of the loss function have a comparable contribution to the total gradient.

The second approach involves fine-tuning a pre-trained RGB network on a color invariant as input. We randomly initialize the first convolutional layer in case the number of input channels deviates from 3.

3.3. Fusion of multiple modalities

We investigate four fusion methods for combining different input representations: input concatenation, feature concatenation, feature summation and prediction averaging. We modify the FC-DenseNet architecture in figure 1a to accommodate for the fusion of two different inputs at different stages of the network.

Input concatenation directly fuses different representations at the input layer and uses the same encoder to learn semantic information from both inputs. The fusion of information is thus solely dependent on the filters learned by the first convolutional layer.

Feature concatenation (figure 1b) and feature summation (figure 1c) both use separate encoders to individually extract features from the two inputs. In feature concatenation the skip connections from both encoders, in the figures shown as dashed lines, as well as the output of the last Dense Block (DB) are concatenated and fed to the decoder. In feature summation, the skip connections and DB outputs are first added together and then concatenated with the decoder. Both ways thus ensure fusion of inputs at three different semantic levels.

Finally, prediction averaging simply calculates the mean prediction of two individual networks.

Table 2: Number of rank 1 retrievals out of 1000 query images in the ALOI image retrieval benchmark (higher is better). Comp and N_λ show best average performance.

	Geometry	**Color**	**Average**
RGB	483	997	740.0
r, g, b	819	524	671.5
D^2	804	310	557.0
Comp	820	909	**864.5**
H	662	215	438.5
C_λ	738	362	550.0
N_λ	796	973	**884.5**

4. Experiments

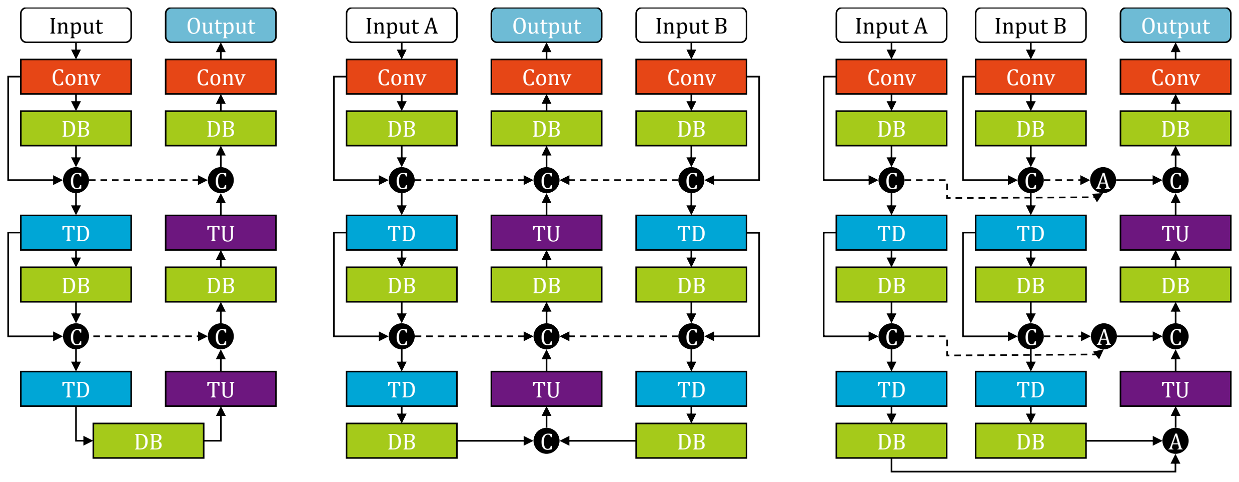
4.1. Real-world performance of color invariants

The color invariants from section 3.1 are evaluated on their robustness to changes in scene geometry and illuminant color by performing an image retrieval task on the ALOI dataset [13], which contains 1000 objects, both matte and glossy, photographed under varying lighting conditions. Given a query image the closest target image out of the 999 other images is returned, measured by the sum of absolute differences of pixel values between two images. The rank denotes after how many images the correct target image is retrieved. The overall performance is reported as the average rank over all query images. Our query image contains an object photographed under normal lighting conditions and has two corresponding target images of the same object, illuminated from a different position or with a different illuminant color. A sample query image with its two target images is shown in figure 2. The benchmark is performed on all color invariants from table 1. The results are shown in table 2.

All invariants show improved robustness to variations in scene geometry compared to the RGB color space, whereas all representations are less invariant to the illuminant color. Comp and N_λ show the best average performance and are therefore used throughout the rest of this work.

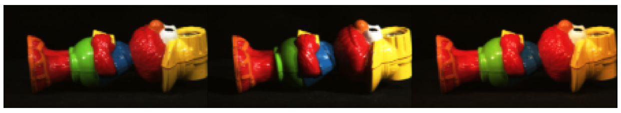
4.2. Illumination robustness of CNNs

We estimate the potential benefits of using color invariants in deep learning by performing an evaluation on the



(a) FC-DenseNet [21]. (b) Feature concatenation. (c) Feature addition.

Figure 1: FC-DenseNet-based mid-fusion CNNs. Dashed lines denote skip connections, C is concatenation, A is addition.



(a) Baseline (b) Scene gemetry (c) Illuminant color

Figure 2: Sample image from the ALOI dataset[13]. In the image retrieval task, (a) is used as the query image, whereas (b) and (c) represent the target images. (a) and (b) mostly differ in terms of shading and shadows caused by the change in illuminant position, whereas (c) has an overall yellow glow compared to (a).

intrinsic robustness of CNNs to illumination-based domain shifts using a classification toy problem. We create 3DMNIST, a 3D version of the MNIST handwritten digit dataset [27]. The data samples are rendered with POV-Ray [34], which provides a fully controlled environment in terms of illumination, and have a resolution of 96x96 pixels. Pixels from the original MNIST dataset are rendered as individual 3D blocks on a uniform background. The 3D scene is illuminated by a single light source and is subsequently captured from a fixed location, resulting again in a 2D image. A large training set of 11,000 images under normal lighting conditions, as well as multiple test sets of 1,000 images with different illumination effects have been generated, including variations in global illumination intensity, scene geometry and illuminant color. This allows both different CNNs and color invariants to be evaluated on individual illumination effects. As figure 3 shows, Comp and N_λ are invariant to nearly all changes. A clear failure case is the N_λ representation under a changed illuminant color, caused by the sharp and thin edges between object and background due to

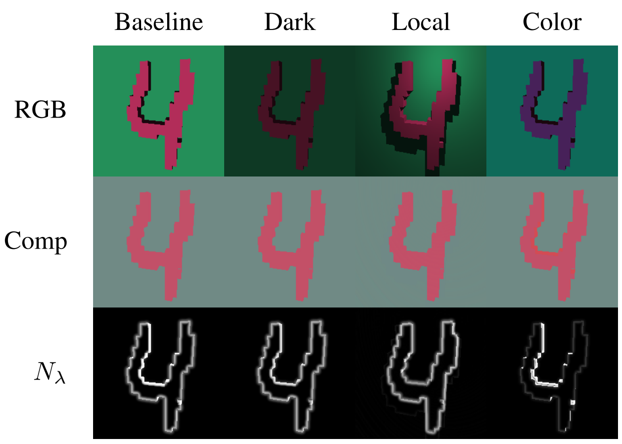


Figure 3: Sample from different test sets of the 3DMNIST dataset, shown in RGB, Comp and N_λ representations.

the artificial nature of the dataset.

4.2.1 How robust are CNNs to illumination variations?

An initial evaluation is performed on ResNet-18 [16] and VGG16 [37], two popular architectures often used as feature extractors in semantic segmentation networks. We train both networks on the baseline training set with a batch size of 50 and evaluate their performance on the various test sets. The networks are trained using the RMSProp [40] optimizer with initial learning learning rates 0.001 and 0.0001 for ResNet-18 and VGG16, respectively. The learning rate is reduced by a factor 0.1 if the training loss has not decreased for 10 epochs up to a minimum of 1e-6. Training is halted after the training loss has not improved for 20 consec-

Table 3: Classification accuracy on the different test sets of the 3DMNIST dataset (average of three models) using different input representations. Bold indicates sensitivity to the respective illumination change. ResNet-18 is highly sensitive to illumination changes and benefits from color invariants, whereas VGG16 is inherently robust to all effects.

	ResNet-18			**VGG16**		
	RGB	Comp	N_λ	RGB	Comp	N_λ
Baseline	98.5	98.6	98.2	98.6	98.6	98.4
Dark	**10.4**	98.5	97.9	98.1	98.6	98.5
Local	**21.2**	98.6	95.1	97.4	98.6	97.0
Color	**10.1**	98.5	**29.6**	96.4	98.5	89.9

utive epochs and the weights corresponding to the highest validation accuracy are stored. Each architecture is trained from scratch three times and the average performance. The same process has been repeated with the dataset converted to the Comp and N_λ representations.

The results in table 3 show that both architectures perform well on the baseline set, but respond differently to illumination changes: ResNet-18 is extremely sensitive to all sorts of changes and consequently benefits from the use of color invariant inputs, whereas VGG16 is inherently robust to intensity variations and thus the Comp and N_λ representations provide only a marginal performance gain.

4.2.2 What makes CNNs sensitive to illumination?

The ResNet-18 and VGG16 architectures mainly differ in three aspects. ResNet-18 uses both residual connections [16] and Batch Normalization (BN) layers [20], whereas VGG16 uses neither. Moreover, ResNet-18 contains a Global Average Pooling (GAP) layer [30] as opposed to a flatten layer in VGG16. To identify which of these differences causes the sensitivity of ResNet-18, we define the simple base model in table 4 and create three variations: a model with residual building blocks [16] instead of plain Conv2D layers (Res), a model with a Batch Normalization layer after each Conv2D layer (BN) and a model in which flattening has been replaced by Global Average Pooling (GAP). Following the same procedure as before, we evaluate their performance on the 3DMNIST dataset. The results are summarized in table 5.

Residual connections do not have a negative effect on the robustness of neural networks and rather increase the overall performance compared to the base model.

The model featuring Global Average Pooling performs worse on the Local and Color test sets. GAP reduces the number of trainable parameters in a model by fixing the transformation matrix between the last convolutional layer

Table 4: Base network architecture with 96x96x3 input shape. Conv2D layers are followed by ReLU activation functions and the final dense layer by softmax.

Layer	**Filter**	**Output shape**
Conv2D	7x7, 16, stride 2	48x48x16
MaxPooling2D	4x4, stride 4	12x12x16
Conv2D	3x3, 32	12x12x32
MaxPooling2D	4x4, stride 4	3x3x32
Flatten		288
Dense	288x10 fc	10

Table 5: Classification accuracy of the base model and its three variations, including residual connections (Res), Batch Normalization (BN) and Global Average Pooling (GAP), on the 3DMNIST test sets. Bold indicates sensitivity to the respective illumination change.

Model	**Base**	**Res**	**BN**	**GAP**
Parameters	9,898	33,034	10,090	7,338
Baseline	96.3	98.1	96.4	94.4
Dark	95.8	97.8	**32.0**	93.8
Local	92.3	95.2	**46.4**	**71.0**
Color	93.3	97.1	**63.0**	**42.2**

and the output layer, which generally reduces overfitting in large networks and such improves performance. Moreover, it enforces correspondence between categories and feature maps as a whole. Our hypothesis as to the performance degradation in the GAP model is two-fold. First, we argue that our network is so small that it does not contain sufficient final feature maps for such correspondences to be enforced for all classes, resulting in increased sensitivity to changes in the input data. Indeed, increasing the number of feature maps to 288 partly restores robustness. Our second hypothesis is more elaborate. Uniformly darkening an image is equivalent to multiplying all pixel values by a scale factor of < 1. Consequently, assuming linear activation functions, all activations in a neural network will be uniformly reduced by the same scale factor and the softmax layer will thus result in the same output class. Experiments showed that this argument also approximately holds for ReLU activations and as such the GAP model performs similarly well on both the Baseline and the Dark test sets. Shadows and variations in the illuminant color, on the other hand, affect feature map activations non-uniformly. Given the same input digit from the Baseline and Color test sets figure 4 shows how the activations of individual neurons in the final feature map (left) and the average activation of feature maps as a result of GAP (right) differ between the two

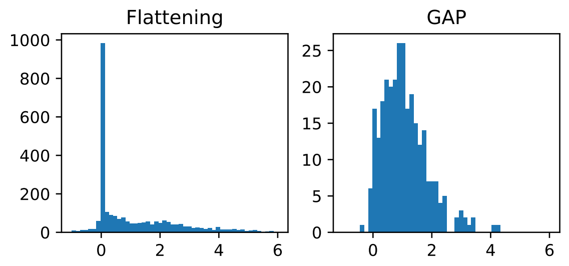


Figure 4: Differences in activations of individual neurons (left) and average activations of neurons in feature map (right) given the same input from the Baseline and Color test set. Individual neurons remain more stable.

inputs. Individual neuron activations remain more constant compared to the average activation of feature maps and thus flattening is more robust to shadows and changes in illuminant color than GAP.

The model with BN layers achieves a higher classification accuracy on the baseline test set but performs significantly worse under all types of illumination variations, indicating BN to be the main cause of the intensity sensitivity of ResNet-18. Given a mini-batch $\mathcal{B}(x_1, \ldots, x_n)$ of size n, BN normalizes, scales and shifts the output of each feature map before activation, such that the batch normalized output sample y_i of input x_i is given by

$$y_i = \gamma \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}} + \beta := \mathrm{BN}_{\gamma,\beta}(x_i), \tag{15}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ represent the mean and variance used for normalization, ϵ is a constant added for numerical stability and γ and β are the trainable scale and shift parameters. During training $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are calculated as the sample mean and variance of the given feature map of the mini-batch $\mathcal{B}$, whereas at inference time the batch statistics are replaced by the statistics of the entire train set, which are estimated during training as the moving average of the batch statistics. However, this assumes that both the training set and test set have equal statistical distributions which makes BN inherently sensitive to any domain shift that influences the low-order statistics of the data. Intensity shifts and changes in the illuminant color are in this sense an extreme form of domain shift as they directly affect the channel means and variances of an input image (e.g. the color channels of a darker image have significantly lower mean an variance). This causes a large discrepancy between the pre-computed statistics and the true mean and variance of an image affected by such domain shifts in all network layers, and the normalizations performed in the BN layers will thus result in feature maps of non-zero mean and non-unit variance. Consequently, the input to the next layer will have a different distribution than observed during training.

Table 6: Classification accuracy of the BN model on 3DMNIST using Batch Normalization, Instance Normalization, a combined normalization method and Input Standardization.

	BN	**IN**	**IN+BN**	**IS**
Baseline	**96.4**	93.8	95.4	93.4
Dark	32.1	94.2	**96.1**	93.4
Local	46.4	64.6	**66.0**	11.9
Color	63.0	**93.5**	93.3	93.2

4.2.3 Addressing BN-induced illumination sensitivity

The assumption that BN-induced illumination sensitivity is caused by a statistical distribution shift between the source and target domain suggests that this problem can be easily overcome by replacing the training set statistics with the target domain statistics. AdaBN [28] does exactly so and estimates the target domain statistics using an online algorithm. However, this technique requires access to the target domain dataset.

Instance Normalization [43] (IN) normalizes the output of a feature map using the mean and standard deviation of a single input sample instead of a batch of samples, thus being equivalent to BN with a batch size of 1. However, as opposed to BN, IN performs the same calculation during both training and inference and so it does not rely on the statistics of the training set. We again evaluate the simple model, first using BN layers and then using IN layers. Since the input to deeper layers in a network depends on the output of its earlier layers, a distribution mismatch in the early layers is propagated through the entire network leading to erroneous predictions. To this end, similarly to AdaBN, we also explore a combined method in which IN is used in the earlier layers and BN in the deeper layers. Table 6 shows the results for all three normalization approaches (three leftmost columns).

Both the IN and IN+BN methods show a similar performance improvement compared to BN, which relies entirely on the training set statistics. This raises the question whether the distribution shift can be addressed outside the network by directly normalizing the input data itself, which is often referred to as input standardization (IS). We train an additional model with standardized inputs by subtracting the channel means and dividing by the channel standard deviations calculated from a single input image and evaluate its performance using BN layers. The results are shown in the rightmost column of table 6. Although input standardization improves robustness to global illumination changes, the resulting model becomes more sensitive to local changes such as shadows. This indicates that normalization of all feature maps throughout the network is essential for improving the robustness of a model.

4.3. Nighttime Segmentation

Experimental setup We use the FC-DenseNet [21] semantic segmentation network for our experiments, which has a conventional encoder-decoder architecture and is relatively inexpensive to train as it does not require a pre-trained frontend. We trained our models on the training set of the CityScapes [4] dataset containing 2975 densely annotated daytime street images and used the 50 coarsely annotated nighttime street images from the Nighttime Driving [6] dataset for evaluation. Results are evaluated by the mean Intersection-over-Union (mIoU) metric. Considering our findings regarding the Batch Normalization layers, we report performance both using BN layers as well as using IN layers.

Training is performed using the Adam [24] optimizer, using an initial learning rate of 0.001 which is lowered by a factor 0.1 after the training loss has not improved for 10 consecutive epochs. All input images are resized to 256x512 pixels and randomly cropped to 224x224 pixels, allowing us to use a batch size of 4 for the single-encoder models and 3 for larger fusion models. Data augmentation in the form of random scaling, zooming, rotation, brightness variation and horizontal flipping has been applied in order to reduce overfitting. For knowledge distillation we use a temperature of $T = 5$ and a relative weight of $\lambda = 0.03$ based on initial experiments. The Comp and N_λ models used in the prediction averaging fusion approache have been pre-trained on RGB data and fine-tuned on the respective color invariant.

Conversion from RGB to the color invariants is implemented as a pre-processing step. We perform 5 iterations of comprehensive normalization, which is sufficient for the pixels to be converged to fixed values, and apply the mapping function from table 1. Similarly, we convert the images to the N_λ representation from RGB-space using linear transformation 12 and equation 11, and again apply the corresponding mapping function. A standard deviation of $\sigma = 2$ is used for the Gaussian derivative filters.

The pre-processed input images are shown in figure 5. Both color invariant representations yield a visually more similar day-night representation compared to RGB. However, comprehensive normalization produces low-contrast daytime images which are still affected by the illuminant color. The daytime and nighttime images in N_λ representation are nearly indistinguishable, although the nighttime image is slightly affected by noise artefacts.

Results and analysis The segmentation results using a single input source are shown in table 7. We consider RGB, denoted in italic, to be our baseline for all experiments. Both Comp and N_λ perform worse than the baseline

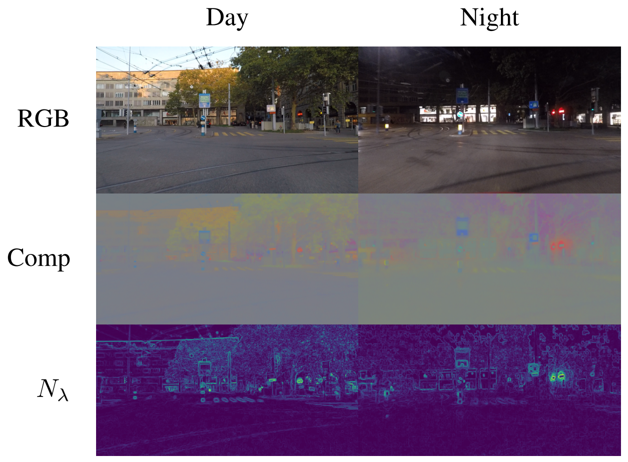


Figure 5: Samples from the Dark Zurich [35] datasets in RGB, Comp and N_λ representations.

Table 7: Day and night segmentation performance using RGB and color invariant inputs measured by the mean IoU (%). N_λ is most robust to illumination changes when using Batch Normalization, but RGB performs best with Instance Normalization.

Normalization	Batch			Instance		
Dataset	**Day**	**Night**	**Avg.**	**Day**	**Night**	**Avg.**
RGB (baseline)	***43.0***	*16.2*	*29.6*	**49.8**	**26.9**	**38.4**
Comp	42.8	8.4	25.6	44.3	21.1	32.7
N_λ	40.7	**21.7**	**31.2**	41.2	22.2	31.7

on the daytime set. As the training sets used for all three experiments are content-wise equal and only differ in their representation, it is natural to conclude that the networks trained on the color invariants got stuck in worse local minima. This is unsurprising considering the trade-off between invariance and discriminative power, as discussed earlier. However, N_λ is more robust to illumination changes and outperforms the baseline on nighttime data when using BN.

Instance Normalization (three rightmost columns) provides a significant performance improvement on both daytime and nighttime domains compared to Batch Normalization (three leftmost columns). In fact, using IN our RGB network even outperforms N_λ on nighttime data. The performance difference between BN and IN on daytime data is somewhat surprising as the test set should have the same statistical distribution as the training set. We argue that the pre-trained statistics calculated over the entire training set are too diverse and therefore not representative for individual samples.

The performance of color invariant networks can easily be improved through both knowledge distillation as well as fine-tuning, as shown in table 8. By fine-tuning our pre-

Table 8: Day and night segmentation performance using transfer learning from RGB to color invariant inputs, measured by the mean IoU (%). The fine-tuned N_λ model performs best on nighttime data.

Normalization	Batch			Instance		
Dataset	**Day**	**Night**	**Avg.**	**Day**	**Night**	**Avg.**
RGB (baseline)	*43.0*	*16.2*	*29.6*	**49.8**	26.9	**38.4**
Knowledge distillation						
RGB $\rightarrow$ Comp	40.9	9.5	25.2	41.8	23.6	32.7
RGB $\rightarrow N_\lambda$	39.4	24.6	32.0	40.4	25.6	33.0
Fine-tuning						
RGB $\rightarrow$ Comp	**44.7**	13.0	28.9	46.2	25.5	35.9
RGB $\rightarrow N_\lambda$	42.5	**25.5**	**34.0**	43.5	**27.5**	35.5

trained RGB baseline model on N_λ input, we have trained a color invariant network that not only outperform the color invariant networks trained from scratch, but also the RGB baseline model itself. Knowledge distillation by means of a teacher-student set-up also significantly improves performance but remains inferior to fine-tuning as erroneous predictions from the teacher network propagate through to the student network.

The segmentation results using various fusion methods are shown in 9. Our early- and mid-fusion approaches provide some performance improvement in the daytime domain but perform significantly worse on the nighttime dataset. Training a network on two inputs simultaneously causes the network to be dependent on both inputs, rather than using each one of them to complement the other. As a result, if the representation of an object deviates from the source domain in either one of the inputs, this object is likely to be misclassified. We verified this hypothesis by adding zero-mean Gaussian noise with a variance of 0.05 to the N_λ input and re-running the evaluation on the RGB + N_λ feature concatenation model, which resulted in a performance loss of $44.9\% \rightarrow 38.7\%$ mIoU and $49.2\% \rightarrow 49.1\%$ mIoU for the daytime domain, using BN and IN layers, respectively. Late fusion by way of averaging of two individual segmentation networks is able to successfully combine complementary information from both inputs. Using a combination of RGB and N_λ, we achieve a 14.5% performance increase on nighttime data compared to using only RGB inputs.

Ablation study: RGB-RGB fusion To verify that the performance improvement shown by the RGB and N_λ late fusion approach is not simply a result of model ensembling, we train a second segmentation model on RGB data and evaluate an RGB+RGB late fusion network. The results, shown in table 10, confirm that the improved illumination robustness is due to the use of color invariants.

Table 9: Day and night segmentation performance using fusion networks to combine RGB and color invariant inputs, measured by the mean IoU (%). Averaging of RGB and N_λ predictions significantly outperforms the baseline in both source and target domain.

Normalization	Batch			Instance		
Dataset	**Day**	**Night**	**Avg.**	**Day**	**Night**	**Avg.**
RGB (baseline)	*43.0*	*16.2*	*29.6*	49.8	26.9	38.4
Input concatenation - early fusion						
RGB + Comp	48.1	12.8	30.5	50.0	24.7	37.4
RGB + N_λ	45.9	8.7	27.3	48.9	24.7	36.8
Comp + N_λ	45.7	12.3	29.0	45.3	22.3	33.8
Feature concatenation - mid fusion						
RGB + Comp	43.3	11.5	27.4	51.1	21.5	36.3
RGB + N_λ	43.5	10.5	27.0	50.2	25.0	37.6
Comp + N_λ	43.9	14.4	29.2	42.9	21.4	32.2
Feature summation - mid fusion						
RGB + Comp	42.1	9.9	26.0	48.5	24.8	36.7
RGB + N_λ	44.9	11.3	28.1	49.2	25.6	37.4
Comp + N_λ	38.7	13.1	25.9	44.4	21.0	32.7
Prediction averaging - late fusion						
RGB + Comp	**57.1**	16.9	37.0	**63.7**	30.4	**47.1**
RGB + N_λ	55.9	**24.8**	**40.4**	63.4	**30.8**	**47.1**
Comp + N_λ	55.6	22.4	39.0	58.8	**30.8**	44.8

Table 10: Day and night segmentation performance using late fusion of two RGB networks versus an RGB and a color invariant network, measured by the mean IoU (%). Improved illumination robustness is due to the use of color invariants and not simply a result of model ensembling.

Normalization	Batch			Instance		
Dataset	**Day**	**Night**	**Avg.**	**Day**	**Night**	**Avg.**
RGB + N_λ	**55.9**	**24.8**	**40.4**	63.4	**30.8**	**47.1**
RGB + RGB	54.6	14.9	34.8	**64.3**	28.9	46.6

State-of-the-art We repeat the best performing method, prediction averaging of RGB and N_λ using Instance Normalization, using the more powerful RefineNet [29] architecture. The ResNet frontend has been pre-trained on the ImageNet [7] dataset in N_λ representation. A performance comparison on the Nighttime Driving dataset is shown in table 11. Although our method does not achieve state-of-the-art performance, it significantly outperforms the RefineNet baseline. Moreover, it does so without the use of target domain data, while [6] requires extensive fine-tuning on intermediate and target domains and [35] uses corresponding day-night image pairs. Qualitative results are shown in table 12.

Table 11: Quantitative evaluation on the Nighttime Driving (Night) datasets using the RefineNet architecture. Color invariants help achieve better-than-baseline performance by only using source domain data.

Method	Trained on	mIoU
RefineNet [29]	Source	34.0
Ours	Source	**37.5**
GCMA [35]	Source + Target	40.9
DarkModel adaptation [6]	Source + Target	**41.6**

5. Discussion

In this paper, we have presented a novel method for improving the illumination robustness of convolutional neural networks using prior knowledge derived from physics models. Additionally, through a thorough examination of different network architectures we have pinpointed Batch Normalization to be the main cause of performance degradation under an illumination-based domain shift and showed that Instance Normalization can be applied instead to largely restore performance.

We found that CNNs trained on color invariants get stuck in worse local minima compared to RGB inputs, but can achieve similar or even superior performance after applying knowledge transfer from RGB by either finetuning or knowledge distillation. We also found that the hyperparameters T and λ used in knowledge distillation have a significant effect on the achieved performance. We leave it to future work to investigate the exact effect of these hyperparameters on the final performance, possibly by performing Bayesian hyperparameter optimization [38].

The color invariants used in this work have been implemented as a separate preprocessing step. It would be interesting to investigate other methods to exploit this physics-based knowledge in deep learning, e.g. by explicitly incorporating reflection models into the neural network architecture or using a regularization term in the backpropagation loss function.

Using our method, we were able to significantly improve segmentation performance on nighttime data while only using training data from the daytime domain. On the Nighttime Driving dataset our method demonstrates competitive performance against domain adaptation methods that do require target domain data.

References

[1] J. A. E. Alvarez, A. Lopez, and R. Baldrich. Illuminant-invariant model-based road segmentation. *2008 IEEE Intelligent Vehicles Symposium*, pages 1175–1180, 2008. 2

[2] J. M. . Alvarez and A. M. Lopez. Road detection based on illuminant invariance. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):184–193, March 2011. 1, 2

[3] Y. Ba, R. Chen, Y. Wang, L. Yan, B. Shi, and A. Kadambi. Physics-based neural networks for shape from polarization. *CoRR*, abs/1903.10210, 2019. 2

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 8

[5] P. I. Corke, R. Paul, W. Churchill, and P. Newman. Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2085–2092, 2013. 2

[6] D. Dai and L. Van Gool. Dark model adaptation: Semantic image segmentation from daytime to nighttime. pages 3819–3824, 11 2018. 2, 8, 9, 10, 11

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 9

[8] G. D. Finlayson, M. S. Drew, and C. Lu. Entropy minimization for shadow removal. *International Journal of Computer Vision*, 85:35–57, 2009. 2

[9] G. D. Finlayson and S. D. Hordley. Color constancy at a pixel. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 18 2:253–64, 2001. 2

[10] G. D. Finlayson, S. D. Hordley, C. Lu, and M. S. Drew. On the removal of shadows from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:59–68, 2006. 2

[11] G. D. Finlayson, B. Schiele, and J. L. Crowley. Comprehensive colour image normalization. In H. Burkhardt and B. Neumann, editors, *Computer Vision — ECCV'98*, pages 475–490, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. 2, 3

[12] J. . Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and H. Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, Dec 2001. 1

[13] J. Geusebroek and G. Burghouts. Amsterdam library of object images (ALOI). `aloi.science.uva.nl/`, 2004. Accessed: 2019-05-23. 4, 5

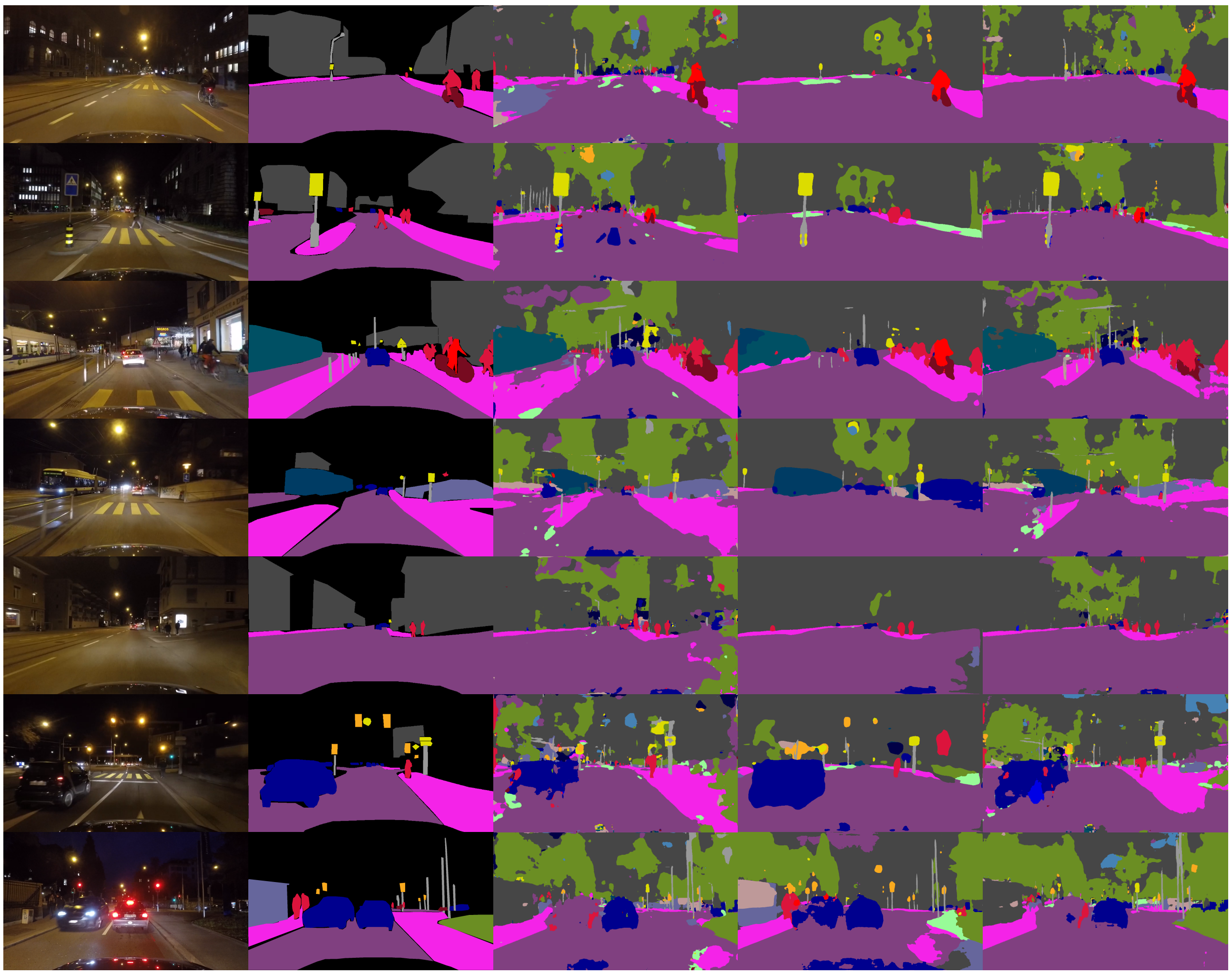
[14] J. M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and A. Dev. Color and scale: The spatial structure of color images. In *European Conference on Computer Vision*, volume 1, pages 331–341, 2000. 3

[15] T. Gevers, A. Gijsenij, J. van de Weijer, and J.-M. Geusebroek. *Color in Computer Vision: Fundamentals and Applications*. Wiley Publishing, 1st edition, 2012. 1

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 5, 6

[17] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 1

Table 12: Qualitative semantic segmentation results on the Nighttime Driving dataset. 'Ours' denotes prediction averaging of RGB and N_λ using RefineNet with Instance Normalization.


(a) Input (b) Ground-truth (c) RefineNet [29] (d) DarkModel [6] (e) Ours

[18] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. 2

[19] J. Hoffman, D. Wang, F. Yu, and T. Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR*, abs/1612.02649, 2016. 2

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 2, 6

[21] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *CoRR*, abs/1611.09326, 2016. 5, 8

[22] A. Karpatne, W. Watkins, J. S. Read, and V. Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *CoRR*, abs/1710.11431, 2017. 2

[23] T. Kim, Y.-W. Tai, and S. eui Yoon. Pca based computation of illumination-invariant space for road detection. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 632–640, 2017. 2

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 8

[25] T. Krajnk, J. Blazicek, and J. Santos. Visual road following using intrinsic images. pages 1–6, 09 2015. 2

[26] P. Kubelka and F. Munk. Ein beitrag zur optik der farbanstriche. In *Zeitung fur Technische Physik*, volume 12, page 593, 1999. 2

[27] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. 5

[28] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. *CoRR*, abs/1603.04779, 2016. 2, 7

[29] G. Lin, A. Milan, C. Shen, and I. D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. *CoRR*, abs/1611.06612, 2016. 9, 10, 11

[30] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2014. 6

[31] M. Long and J. Wang. Learning transferable features with deep adaptation networks. *CoRR*, abs/1502.02791, 2015. 2

[32] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019. 2

[33] W. P. Maddern, A. D. Stewart, C. McManus, B. Upcroft, W. Churchill, and P. Newman. Illumination invariant imaging : Applications in robust vision-based localisation , mapping and classification for autonomous vehicles. 2014. 1, 2

[34] Persistence of Vision Pty. Ltd. Persistence of vision raytracer (version 3.6). http://www.povray.org/download/, 2004. Accessed: 2019-05-23. 5

[35] C. Sakaridis, D. Dai, and L. V. Gool. Semantic nighttime image segmentation with synthetic stylized data, gradual adaptation and uncertainty-aware evaluation. *CoRR*, abs/1901.05946, 2019. 2, 8, 9, 10

[36] S. A. Shafer. Using color to separate reflection components. *Color Research & Application*, 10(4):210–218, 1985. 2

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 5

[38] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012. 10

[39] R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. *CoRR*, abs/1609.05566, 2016. 2

[40] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. 5

[41] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464, 2017. 2

[42] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014. 2

[43] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. 7

[44] B. Upcroft, C. McManus, W. Churchill, W. P. Maddern, and P. Newman. Lighting invariant urban street classification. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1712–1718, 2014. 2

[45] G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson. Do deep convolutional nets really need to be deep (or even convolutional)? *Arxiv*, 521, 03 2016. 1

[46] J. Von Kries. Influence of adaptation on the effects produced by luminous stimuli. In D. L. MacAdam, editor, *Sources of Color Vision*, pages 109–119. MIT Press, Cambridge (MA), 1970. 3

[47] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018. 1

[48] M. Wulfmeier, A. Bewley, and I. Posner. Addressing appearance change in outdoor robotics with adversarial domain adaptation. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1551–1558, 2017. 1

[49] M. Wulfmeier, A. Bewley, and I. Posner. Incremental adversarial domain adaptation for continually changing environments. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2017. 1

[50] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. Kweon. Pixel-level domain transfer. *CoRR*, abs/1603.07442, 2016. 2

2 Introduction

Over the past several years deep learning has shown unprecedented progress in various computer vision tasks, including image classification, object detection and semantic segmentation [1, 2, 3], mainly powered by the increasing availability of cheap computational resources and easily accessible large-scale datasets. Whereas classical machine learning and computer vision methods relied on carefully engineered hand-crafted features [4, 5, 6], the current deep learning paradigm dictates that all information should be learned from the available dataset rather than be included a priori. Although this approach has proven to be successful, it has two general shortcomings. Firstly, without any prior knowledge a deep learning model requires a sufficiently large and diverse dataset to achieve satisfactory performance and to be able to generalize well to unseen data. Secondly, CNNs are often bound to the specific domain of the training dataset and perform significantly worse under any form of domain divergence. A possible solution to both shortcomings is to collect and label more data and to include additional domains in the training set. However, this solution is often extremely labour-intensive and thus costly, if possible at all. The field of domain adaptation [7] offers a more elegant solution to this problem by modifying models trained on data from a certain source domain to perform well on a different but related target domain. However, very few domain adaptation techniques exist that do not require any date from the target domain, which leaves us again with the burden of data collection.

This work investigates a different approach to improve the generalization capabilities of neural networks by moving away from the data-driven paradigm and including prior knowledge in deep learning methods by design. This work investigates the use of physics-based reflection models in a convolutional neural networks to improve the robustness of a model to domain shifts caused by illumination changes in the environment. Specifically, we apply this method in semantic segmentation for autonomous driving.

2.1. Motivation

Illumination changes are omnipresent in all outdoor computer vision applications and are known to significantly affect the performance of neural networks [8]. Color invariants are derived from physical models describing light reflections and represent properties of objects irrespective of their recording conditions [9]. Color invariants have been successfully used in classical computer vision applications [10, 11] and implementing them in a deep learning setting thus seems like an obvious next step.

Color invariants achieve invariance to shadows and shading by factoring out dependence on the scene geometry from an RGB image. Consequently, information is lost in the process and as such invariance to illumination changes comes at a loss of discriminative power [12]. Part of this research therefore focuses on how both RGB and color invariants can be optimally used to complement each other. This leads us to investigating different architectures for the fusion of multiple modalities and transfer learning methods for distilling knowledge between deep learning models trained on different input representations.

2.2. Research questions

The main research question is defined as:

> *What are the physical color invariance accuracy versus generalizability trade-offs under an illumination-varying domain shift?*

This is further divided into the following sub-questions:

Q1. *Is the illumination-robustness of a neural network dependent on its architecture?*

Q2. *Can the robustness of neural networks to illumination-based domain shifts be improved through the use of physics-based color invariants?*

The first question aims to explore the relation between the intrinsic illumination-robustness of a convolutional neural network and the specific layer types used in its architecture, whereas the second question concerns the use of physics-based models to include prior information about the formation of color images in order to improve the illumination-robustness of a CNN.

2.3. Outline

The chapters that follow contain supplementary materials to the scientific paper presented before. Chapter 3 includes background information about both the physics-based image formation models that are used in this work, as well as a general introduction to deep learning in computer vision. Chapter 4 contains additional experiments to support claims in the scientific paper. Finally, chapter 5 introduces a novel method to deep-learn a color invariant mapping from data that did not end up within the scope of the paper.

Bibliography

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

[2] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL `http://arxiv.org/abs/1506.02640`.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015. URL `http://arxiv.org/abs/1511.00561`.

[4] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL `https://doi.org/10.1023/B:VISI.0000029664.99615.94`.

[5] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851.

[6] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[7] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312: 135–153, 2018.

[8] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Addressing appearance change in outdoor robotics with adversarial domain adaptation. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1551–1558, 2017.

[9] Theo Gevers, Arjan Gijsenij, Joost van de Weijer, and Jan-Mark Geusebroek. *Color in Computer Vision: Fundamentals and Applications*. Wiley Publishing, 1st edition, 2012. ISBN 0470890843, 9780470890844.

[10] J. M. Á. Alvarez and A. M. Lopez. Road detection based on illuminant invariance. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):184–193, March 2011. ISSN 1524-9050. doi: 10.1109/TITS.2010.2076349.

[11] William P. Maddern, Alexander D. Stewart, Colin McManus, Ben Upcroft, Winston Churchill, and Paul Newman. Illumination invariant imaging : Applications in robust vision-based localisation , mapping and classification for autonomous vehicles. 2014.

[12] J. . Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and H. Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, Dec 2001. ISSN 0162-8828. doi: 10.1109/34.977559.

3 Preliminaries

3.1. Color Invariants

Color invariants are derived from the physical nature of objects in color images and are invariant to one or more factors influencing the recording conditions [1]. The three most important factors include the scene geometry, which affects the formation of shadows and shading on objects, the object surface properties, which defines whether or not the object exhibits interface reflections (highlights) next to Lambertian (matte) reflections, and the illuminant color, which alters the observed color of the overall scene. In order to derive possible color invariants from RGB images, we should first study how images are rendered in the first place. This process is defined as the color image formation process.

3.1.1. Color image formation

Light is physically described as electromagnetic radiation within a certain range of the electromagnetic spectrum. The visible spectrum for humans ranges from wavelengths of approximately 380 nm (violet) to 740 nm (infrared). *Color image formation* describes the process of how visual images are formed by an observer, be it either an electronic device such as a visual light- or hyperspectral camera, or the eyes of humans and animals. The process consists of three stages: *illumination*, *material reflection* and *detection*.

Illumination

The process begins with a light source emitting photons, thereby illuminating the environment. The light source is characterized by its power spectral density $e(\lambda)$. Light with a uniform spectral density is called white and can be notation-wise simplified to $e(\lambda) = e$. Moreover, we distinguish between an isotropic light source which radiates light in all directions with the same power spectral density, and a directed light source where the power spectrum also depends on the relative location x.

Material reflection

Material reflection is modelled using a so-called Bidirectional Reflectance Distribution Function (BRDF). Incident light interacts with an object by partly being reflected and partly being absorbed. A material property called the *surface albedo* defines which part of the spectrum of the light is absorbed and consequently defines the object color, e.g. a red object absorbs light rays of all wavelengths except those corresponding to the red color and is therefore observed red. When the intensity of the reflected light is independent of the viewing angle, i.e. the reflection is isotropic, we speak of *Lambertian reflection.* This is the case for matte materials such as fabric, unfinished wood and paper. The reflected energy from a surface E under the assumption of Lambertian reflection is given by the *Lambertian Reflection*

Model

$$E(\lambda, \vec{x}) = m^b(\vec{x}) s(\lambda, \vec{x}) e(\lambda, \vec{x}), \tag{3.1}$$

where $m^b(\vec{x})$ is a scale factor depending on the scene geometry (e.g. the angular difference between incoming light and surface normal), $s(\lambda, \vec{x})$ denotes the surface albedo and $e(\lambda, \vec{x})$ the power spectrum of the light source.

In addition to Lambertian reflection, glossy materials also exhibit *interface reflections*, which introduce highlights on the object. In case of interface reflections, the incident light is reflected directly from the object surface without interacting with the albedo and therefore the spectral density of the reflected light beam is not affected. Shafer's *Dicromatic Reflection Model* (DRM) accounts for both Lambertian and interface reflections: [2].

$$E(\lambda, \vec{x}) = \left(m^b(\vec{x}) s(\lambda, \vec{x}) + m^i(\vec{x})\right) e(\lambda, \vec{x}), \tag{3.2}$$

with $m_i(\vec{x})$ being the geometric factor for interface reflections. It is easy to see that the DRM is essentially an extension of the Lambertian reflection model with a second term accounting for interface reflections.

Both aforementioned reflection models assume a single, direct light source. However, in most realistic scenes, an additional ambient light source of lower intensity can be observed. Indeed, an outdoor environment contains both a direct light source, the sun, and the diffuse, blue-ish illumination of the sky. While the illuminated parts of the scene are dominated by the direct light source, in the shaded areas the ambient illuminant has a non-negligible effect. This highlights an important shortcoming of the DRM. Alternative BRDFs have been proposed to account for such ambient light source, including the *Bi-illuminant Dichromatic Reflection Model* [3]. However, the color invariants derived from such models require additional scene-specific calibration parameters and therefore are not suitable for the purposes of this work.

An alternative description of photometric reflections is provided by the *Kubelka-Munk reflection model* [4]

$$E(\lambda, \vec{x}) = e(\lambda, \vec{x})(1 - \rho_f(\vec{x}))^2 R_\infty(\lambda, \vec{x}) + e(\lambda, \vec{x}) \rho_f(\vec{x}) \tag{3.3}$$

where $e(\lambda, \vec{x})$ again denotes the power spectrum of the illuminant, $\rho_f(\vec{x})$ the interface reflection coefficient of the material and $R_\infty(\lambda, \vec{x})$ the material reflectivity, which is a property similar to the albedo. Substitution of

$$\begin{aligned} s(\lambda, \vec{x}) &= R_\infty(\lambda, \vec{x}), \\ m^b(\vec{x}) &= (1 - \rho_f(\vec{x}))^2, \\ m^i(\vec{x}) &= \rho_f(\vec{x}) \end{aligned}$$

shows the similarity with the Dichromatic Reflection Model in equation 3.2.

Detection

Finally, reflected light is captured by integrating the energy of the photons over a certain bandwidth ω, spatial area and period of time. According to the trichromatic theory [5], three independent detectors, each tuned to a specific wavelength, are required to record the full color space observed by humans. In a camera, each pixel is recorded by capturing the intensity of long, medium and short wavelength light using three separate sensors corresponding to the three cones in the retina of an eye. The RGB responses of a camera are therefore given by

$$\begin{aligned} f^R(\vec{x}) &= \int_\omega E(\lambda, \vec{x}) \rho^R(\lambda) d\lambda, \\ f^G(\vec{x}) &= \int_\omega E(\lambda, \vec{x}) \rho^G(\lambda) d\lambda, \\ f^B(\vec{x}) &= \int_\omega E(\lambda, \vec{x}) \rho^B(\lambda) d\lambda, \end{aligned} \tag{3.4}$$

where $\rho^c(\lambda)$ with $c \in R, G, B$ is the spectral sensitivity function of the sensor, tuned to the wavelength corresponding to the respective color channel.

3.1.2. Derivation of color invariants

Based on this color image formation process and some simplifying assumptions, different color invariants can be derived that factor out the dependencies on either geometry, illuminant color, or both. Since these color invariants will be used as inputs to CNNs, their outputs should be well-defined and within a reasonable range as infinity and NaN values cause unpredictable behaviour.

DRM-based color invariants

The sensor response function for color i based on the Lambertian reflection model assuming white illumination ($e(\lambda, \vec{x}) = e(\vec{x})$) is defined as

$$f^i(\vec{x}) = m^b(\vec{x})e(\vec{x}) \int_\omega s(\lambda, \vec{x})\rho^i(\lambda)d\lambda. \tag{3.5}$$

It is clear that the dependence on geometry and intensity can easily be factored out by division and the resulting representation is only dependent on the properties of the object and the camera sensors.

$$\frac{f^i}{f^j} = \frac{m^b(\vec{x})e(\vec{x}) \int_\omega s(\lambda, \vec{x})\rho^i(\lambda)d\lambda}{m^b(\vec{x})e(\vec{x}) \int_\omega s(\lambda, \vec{x})\rho^j(\lambda)d\lambda} = \frac{\int_\omega s(\lambda, \vec{x})\rho^i(\lambda)d\lambda}{\int_\omega s(\lambda, \vec{x})\rho^j(\lambda)d\lambda} \tag{3.6}$$

Considering the RGB channels, a complete set of Lambertian color invariants is given by [1]

$$L_{RGB} = \frac{\sum_i a_i R_i^p G_i^q B_i^r}{\sum_j b_j R_j^s G_j^t B_j^u}, \tag{3.7}$$

where $p + q + r = s + t + u$ denotes the order of the color invariant. A well-known first-order color invariant is normalized RGB, where each of the color channels is divided by the total pixel intensity.

$$\{r, g, b\} = \{R, G, B\} / (R + G + B + \epsilon) \tag{3.8}$$

The color space is undefined for low intensity, e.g. for $R = G = B \approx 0$, and therefore a small number ϵ is added for numerical stability. Normalized RGB has output range $\{r, g, b\} \in [0, 1]$.

Color invariants for interface reflections can be derived in a similar fashion. The sensor response function given by the full DRM is defined as

$$f^i(\vec{x}) = \int_\omega m^b(\vec{x})s(\lambda, \vec{x})e(\lambda, \vec{x})\rho^i(\lambda)d\lambda + \int_\omega m^i(\vec{x})e(\lambda, \vec{x})\rho^i(\lambda)d\lambda. \tag{3.9}$$

Subtraction and division factors out the geometric scale factor for both Lambertian and interface reflections and the resulting representation is again only dependent on the albedo en sensor sensitivities.

$$\frac{f^i - f^j}{f^k - f^m} = \frac{\int_\omega s(\lambda, \vec{x})\rho^i(\lambda)d\lambda - \int_\omega s(\lambda, \vec{x})\rho^j(\lambda)d\lambda}{\int_\omega s(\lambda, \vec{x})\rho^k(\lambda)d\lambda - \int_\omega s(\lambda, \vec{x})\rho^m(\lambda)d\lambda}, \quad k \neq m. \tag{3.10}$$

The complete set of invariants is thus given by

$$D_{RGB} = \frac{\sum_i a_i (R-G)_i^p (R-B)_i^q (G-B)_i^r}{\sum_j b_j (R-G)_j^s (R-B)_j^t (G-B)_j^u}. \tag{3.11}$$

Let us define the second-order Dichromatic invariant

$$\{D_R, D_G, D_B\} = \frac{\left\{(R-G)^2, (G-B)^2, (B-R)^2\right\}}{(R-G)^2 + (G-B)^2 + (R-B)^2} \tag{3.12}$$

which is well-defined for $R \neq G \neq B$ with an output range $[0, \frac{2}{3}]$.

When we drop the assumption of white illumination, it is no longer possible to take the term $e(\lambda, \vec{x})$ out of the integration in equation 3.9 and deriving color invariants becomes again non-trivial. However, we can use the *von Kries Model* [6] to predict the change in camera sensor responses under two different illuminants a and b:

$$\begin{pmatrix} f_b^R \\ f_b^G \\ f_b^B \end{pmatrix} = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{pmatrix} \begin{pmatrix} f_a^R \\ f_a^G \\ f_a^B \end{pmatrix} \tag{3.13}$$

This transformation reveals that a ratio of two pixel values of the same channel factors out the effect of the illuminant. More specifically, dividing each pixel value of an image by the mean of all pixel values of the same channel results in an illuminant invariant representation, as is shown for the R channel in equation 3.14.

$$\frac{f_{b,i}^R}{\frac{1}{N}\sum_{i=1}^N f_{b,i}^R} = \frac{\alpha f_{a,i}^R}{\frac{1}{N}\sum_{i=1}^N \alpha f_{a,i}^R} = \frac{f_{a,i}^R}{\frac{1}{N}\sum_{i=1}^N f_{a,i}^R} \tag{3.14}$$

Invariance to both illuminant color and lighting geometry can be achieved through *comprehensive image normalization* [7], an iterative process of normalizing for illuminant color (equation 3.14) and lighting geometry (equation 3.8), which converges to a fixed point.

Being an iterative process, it is less straightforward to calculate the output range of comprehensive normalization. A good estimate can be obtained by producing a histogram of pixel values from a diverse dataset, such as STL-10 [7], as shown in the leftmost image in figure 3.1. Based on this histogram, a logistic function with midpoint 1 is defined as the mapping function: $f(x) = \frac{1}{1+e^{-(x-1)}}$.

Kubelka-Munk-based color invariants

Rather than considering the complete color image formation process, Kubelka-Munk-based color invariants are derived directly from the reflection model. Let us assume white illumination and let us drop $(\lambda, \vec{x})$ from $E(\lambda, \vec{x})$ for notational convenience. The Kubelka-Munk reflection model from equation 3.3 now simplifies to

$$E = e(\vec{x})\left\{\rho_f(\vec{x}) + (1-\rho_f(\vec{x}))^2 R_\infty(\lambda, \vec{x})\right\}, \tag{3.15}$$

with its first- and second-order derivatives with respect to λ defined as

$$E_\lambda = e(\vec{x})(1-\rho_f(\vec{x}))^2 \frac{\partial R_\infty(\lambda, \vec{x})}{\partial \lambda}, \tag{3.16}$$

$$E_{\lambda\lambda} = e(\vec{x})(1-\rho_f(\vec{x}))^2 \frac{\partial^2 R_\infty(\lambda, \vec{x})}{\partial \lambda^2}. \tag{3.17}$$

Division between these two derivatives factors out all terms except the material reflectance and thus

$$H = \frac{E_\lambda}{E_{\lambda\lambda}} = \frac{\frac{\partial R_\infty(\lambda,\vec{x})}{\partial \lambda}}{\frac{\partial^2 R_\infty(\lambda,\vec{x})}{\partial \lambda^2}} \tag{3.18}$$

is an invariant for lighting geometry, for both Lambertian and interface reflections. If we assume matte surfaces with low interface reflectance ($\rho_f(\vec{x}) \approx 0$), then the ratio

$$C_\lambda = \frac{E_\lambda}{E} = \frac{\frac{\partial R_\infty(\lambda,\vec{x})}{\partial \lambda}}{\frac{\partial^2 R_\infty(\lambda,\vec{x})}{\partial \lambda^2}} \tag{3.19}$$

is also an invariant.

Let us now relax the assumption on white illumination. Let the spectral density of the illuminant be not necessarily uniform but independent on the scene geometry. Then $e(\lambda, \vec{x})$ can be decomposed into a separate spectral component $e(\lambda)$ and a spatial component $i(\vec{x})$ and thus for matte surfaces, equation 3.15 reduces to

$$E = e(\lambda)i(\vec{x})R_\infty(\lambda, \vec{x}) \tag{3.20}$$

with the first order derivative with respect to λ being

$$E_\lambda = i(\vec{x})R_\infty(\lambda,\vec{x})\frac{\partial e(\lambda)}{\partial\lambda} + e(\lambda)i(\vec{x})\frac{\partial R_\infty(\lambda,\vec{x})}{\partial\lambda}. \tag{3.21}$$

Dividing the two terms results in

$$\frac{E_\lambda}{E} = \frac{1}{e(\lambda)}\frac{\partial e(\lambda)}{\partial\lambda} + \frac{1}{R_\infty(\lambda,\vec{x})}\frac{\partial R_\infty(\lambda,\vec{x})}{\partial\lambda}, \tag{3.22}$$

which consists of a term related to the illuminant $e(\lambda)$ and a term related to the material reflectivity $R_\infty(\lambda,\vec{x})$. As the illuminant is independent of the scene, the first term can easily be removed by differentiating with respect to $\vec{x}$

$$\frac{\partial}{\partial x}\left\{\frac{E_\lambda}{E}\right\} = \frac{\partial}{\partial x}\left\{\frac{1}{R_\infty(\lambda,\vec{x})}\frac{\partial R_\infty(\lambda,\vec{x})}{\partial\lambda}\right\}, \tag{3.23}$$

and as such

$$N_{\lambda\vec{x}} = \frac{\partial}{\partial x}\left\{\frac{E_\lambda}{E}\right\} = \frac{E_{\lambda\vec{x}}E - E_\lambda E_{\vec{x}}}{E^2} \tag{3.24}$$

is an invariant to the scene geometry and illuminant color. The spatial derivative of E and E_λ is calculated by convolution with Gaussian derivative filters and the gradient magnitude is simply given by $N_\lambda = \sqrt{N_{\lambda x}^2 + N_{\lambda y}^2}$.

Based on the Gaussian color model[8], a linear transformation can be derived to estimate the reflected energy E and its first- and second-order derivatives E_λ and $E_{\lambda\lambda}$ from RGB camera responses.

$$\begin{pmatrix}\hat{E}\\ \hat{E}_\lambda\\ \hat{E}_{\lambda\lambda}\end{pmatrix} = \begin{pmatrix}0.0024 & 0.0252 & 0.0108\\ 0.0119 & 0.0017 & -0.0140\\ 0.0137 & -0.0240 & 0.0066\end{pmatrix}\begin{pmatrix}R\\ G\\ B\end{pmatrix} \tag{3.25}$$

The output ranges and the corresponding mapping functions are derived in the same way as for the comprehensive normalization. The resulting histograms are shown in figure 3.1.

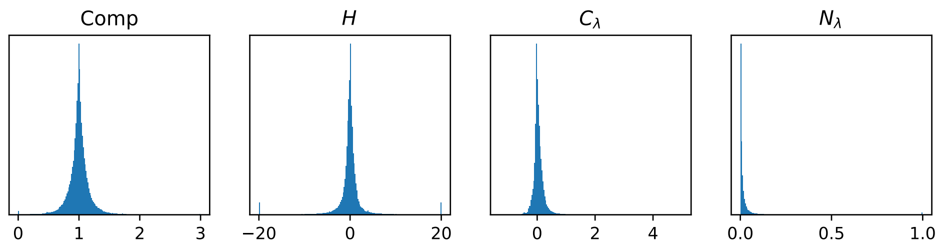


Figure 3.1: Histogram of pixel values of the STL-10 dataset in the Comp, H, C_λ and N_λ Gaussian color spaces.

3.1.3. Overview of color invariants

Figure 3.2 shows a visualization of the color invariants mentioned throughout this section. An overview of the invariance properties can be found in table 1 of the scientific paper.

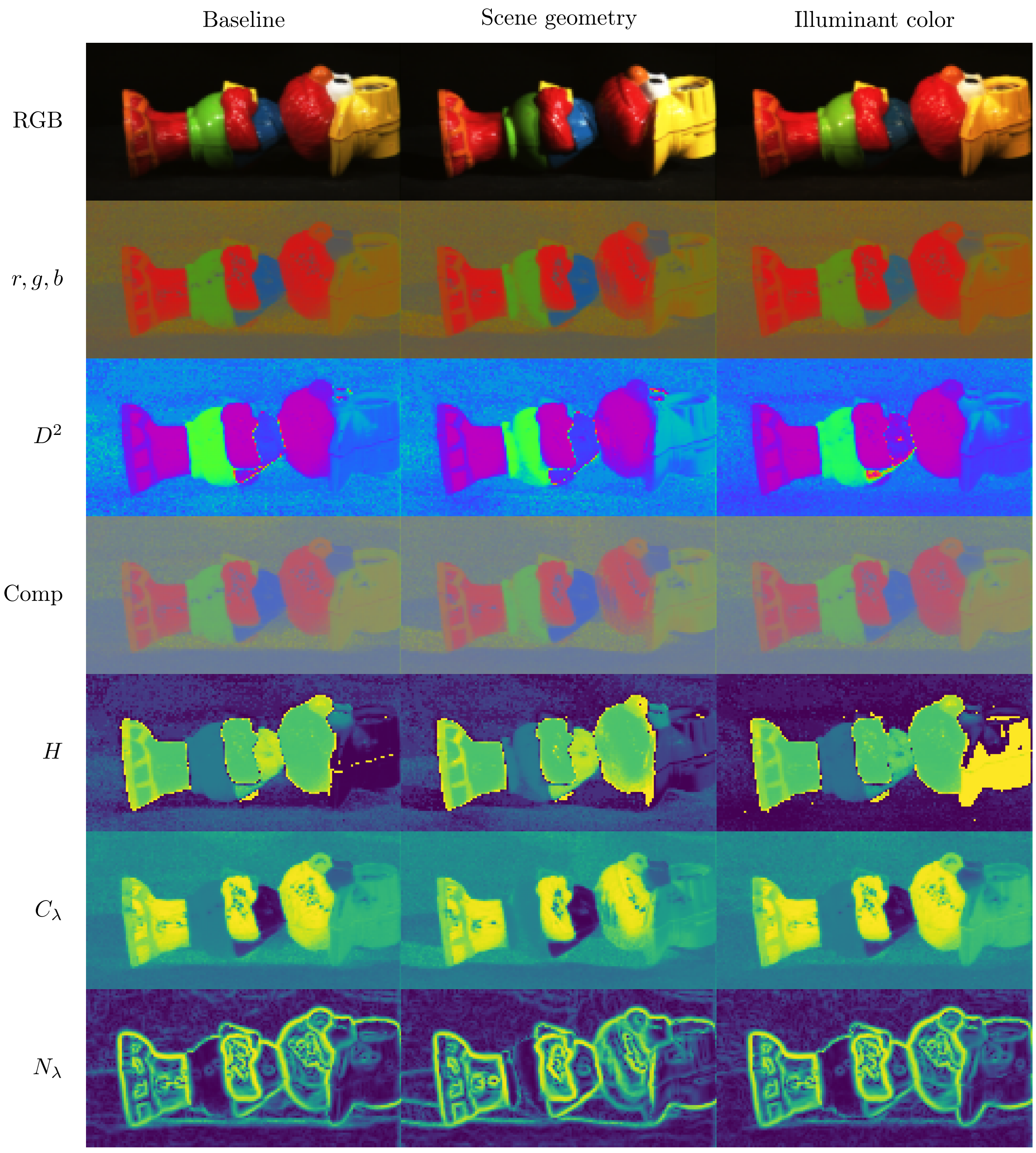


Figure 3.2: Visualization of color invariants under normal "baseline" lighting conditions, a change in scene geometry and a change in illuminant color. Most color invariants are noise-sensitive in low-intensity parts of the image. Comp and N_λ are the only color invariants that are, in theory, invariant to both types of changes. The images are taken from the Amsterdam Library of Object Images (ALOI) dataset [9].

3.2. Deep Learning in Computer Vision

Supervised machine learning is concerned with learning a function $f(x) = y$ which maps an input x to a desired output y given a large collection of input-output pairs. This function is then used to perform a classification or regression task on new, previously unseen data. Deep learning is a subset of machine learning that uses artificial neural networks, consisting of multiple interconnected layers of neurons, to define the structure of the mapping function. In recent years, deep learning gave rise to significant improvements in computer vision tasks such as image classification, object detection and semantic segmentation [10, 11, 12], as illustrated in figure 3.3.

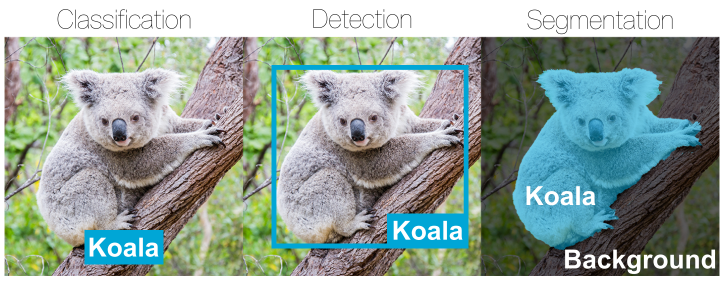


Figure 3.3: Classification, detection and semantic segmentation: three common computer vision tasks.

3.2.1. Neural networks

An artificial neural network (ANN) generally consists of an input layer, several hidden layers and an output layer, where every neuron in a layer is connected to every neuron in the neighbouring layers through weighted connections. The input layer is where input data enters the network - in the case of computer vision applications in the form of pixel intensities - and the output layer gives the desired classification or regression output. The hidden layers and the weights corresponding to their neurons define the exact mapping function $f(x)$. Figure 3.4a denotes an ANN with an input layer and two hidden layers of three neurons each, and an output layer of two neurons.

Inside each neuron a weighted summation of all incoming activations is performed and is offset with a bias term, resulting in the following linear function:

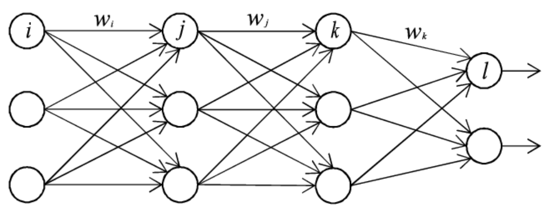
$$a_j = \sum_{i=1}^{n} x_i w_i + b. \tag{3.26}$$

In order to allow the ANN to learn non-linear mapping functions, the output resulting from this summation is passed through an activation function, such that the activation of neuron j as a function of its n inputs is given by

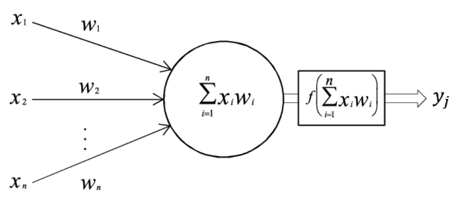
$$y_j = f\left(\sum_{i=1}^{n} x_i w_{ij} + b_j\right), \tag{3.27}$$

as is schematically shown in figure 3.4b. Popular activation functions include the rectified linear unit [13] (ReLU) for hidden layers, given by $f(x) = \max(0, x)$ and the softmax or normalized exponential function for representing output probabilities in the final layer of a network, given by $x_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$.

Based on the outputs from the output layer, a task-specific loss function is defined which is used together with the correct data labels during training to find the values of the weight and bias terms w and b that minimize the loss. Typical loss functions include the mean squared error loss for regression and the categorical cross-entropy loss for classification. Training of the network is performed by computing the gradient of the loss function with respect to the weights w and biases b and performing optimization through gradient descent. This process is known as backpropagation, as the error in the output is propagated back through the network. The loss and gradients are calculated as an average over a randomly chosen subset of the training set called a *mini-batch*, and are then used to update the weights and biases in the network. This is repeated until an *epoch* has elapsed, i.e. all training data in the training set has been used once, after which training either stops or continues with the next epoch. The



(a) Neural network containing two hidden layers of three neurons each.



(b) Schematic representation of a neuron.

Figure 3.4: Schematic representation of an artificial neural network.

mini-batch size and the number of epochs after which to halt training are hyperparameters that need to be defined before training. Due to its random nature, this optimization process is called Stochastic Gradient Descent (SGD). Other popular optimization algorithms that use this same basic principle include RMSProp [14] and Adam [15].

The layers in figure 3.4a are known as *fully-connected* or *dense* layers since all neurons in a layer are connected to all neurons in neighbouring layers. Due to the inherent nature of fully-connected layers, image data needs to be flattened to a vector representation before it can be used in an ANN, a process in which spatial information in the image is lost.

3.2.2. Convolutional neural networks

Convolutional layers

Convolutional layers instead learn fixed-size filters, which are then used to perform convolution operations on the original image or the outputs of the previous layer, as shown in figure 3.5. Filters in early layers represent edges, corners and other low-level features, whereas deeper layer filters learn to recognize increasingly more abstract features, such as pointy ears and whiskers when learning to identify a cat. The output of a convolutional layer can thus be seen as a feature map. The use of convolutions not only preserves spatial information in the image, but also reduces the number of parameters in the network as only the filter values need to be learned rather than individual connections between all neurons, which in turn reduces overfitting.

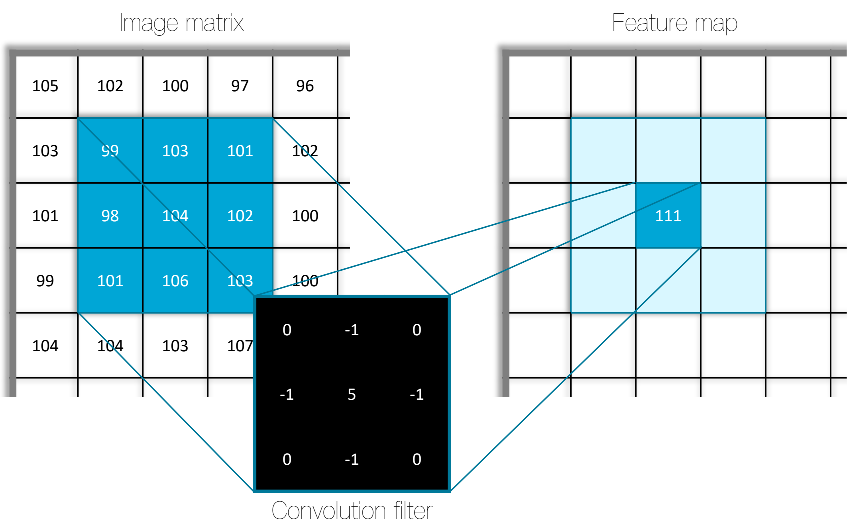


Figure 3.5: Basic principle of a convolutional layer in a neural network.

Pooling layers

Deeper layers of convolutional neural networks (CNNs) generally not only contain more abstract features, but each feature map activation also corresponds to an increasingly large area covering multiple pixels in the input image. This is necessary for abstract features to be detectable as a single pixel does not contain any semantic information; an image patch is needed to detect cat ears, and an even larger patch is needed to detect cats. Feature map reduction can be implemented by either performing convolutions with a stride of larger than 1 or by downsampling the feature map using pooling layers, where the latter is generally preferred. An $n \times n$ pooling layer has a default stride of n and as such reduces the feature map size by a factor n. The most widely used pooling method is *max pooling*, where the output is simply defined as the maximum of the neuron activations in the input patch. Alternatively, *average pooling* can be applied, where the average of the input patch is calculated.

Flattening and Global Average Pooling

Data in CNNs is stored as a four-dimensional tensor of size `(mini-batch size,height,width,channels)`. However, the output of an image classification network is a two-dimensional tensor of class probabilities of size `(mini-batch size,no.classes)`, which requires that at some point in the network convolutional layers are converted to dense layers. This can be achieved through either flattening the last convolutional layer, i.e. reshaping it to a tensor of size `(mini-batch,height*width*channels)`, or by applying Global Average Pooling [16] (GAP), where the output tensor is calculated by averaging the activations in each feature map in the last convolutional layer and has size `(mini-batch size,channels)`. This way, GAP enforces correspondences between classes and feature maps and significantly reduces the number of learnable parameters in the network as it results in a smaller dense layer. Consequently, most state-of-the-art CNNs employ GAP instead of flattening.

3.2.3. Batch normalization

Very deep neural networks tend to be difficult to train and often get stuck in local minima. Batch Normalization [17] (BN) accelerates the training of neural networks by making the optimization landscape smoother and more predictable, allowing the optimizer to use larger steps [18]. BN is implemented as a separate network layer, generally placed between a feature map and its activation function. Given a mini-batch $\mathcal{B}(x_1, \ldots, x_n)$ of size n, BN normalizes, scales and shifts the output of each feature map, such that the batch normalized output sample y_i of input x_i is given by

$$y_i = \gamma \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}} + \beta := \mathrm{BN}_{\gamma,\beta}(x_i), \tag{3.28}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ represent the mean and variance used for normalization, ϵ is a constant added for numerical stability and γ and β are trainable scale and shift parameters. During training $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are calculated as the sample mean and variance of the given feature map of the mini-batch $\mathcal{B}$, whereas at inference time the batch statistics are replaced by the statistics of the entire train set, which are estimated during training as the moving average of the batch statistics. Due to its benefits, BN is used in almost all state-of-the-art ANNs.

3.2.4. Fine-tuning

As a result of the large number of parameters involved, deep neural networks easily overfit when trained on a not sufficiently large training set. This can be problematic, as training data can be scarce for specific applications. However, the filters learned in early layers of CNNs represent generic features and textures, such as edges, corners and various patterns. The intuition behind fine-tuning is that these universal features do not need to be learned from the task-specific dataset, but can rather be trained using any large and diverse collection of image-label pairs. Fine-tuning involves pre-training a cumbersome network on such dataset, which then functions as a so-called feature extractor and is placed before an optional task-specific part of the network. Subsequently, the weights in the first n layers are frozen such that they do not get updated during training and the rest of the network is fine-tuned on the smaller, task-specific dataset. Fine-tuning is a well proven and often applied method in deep learning.

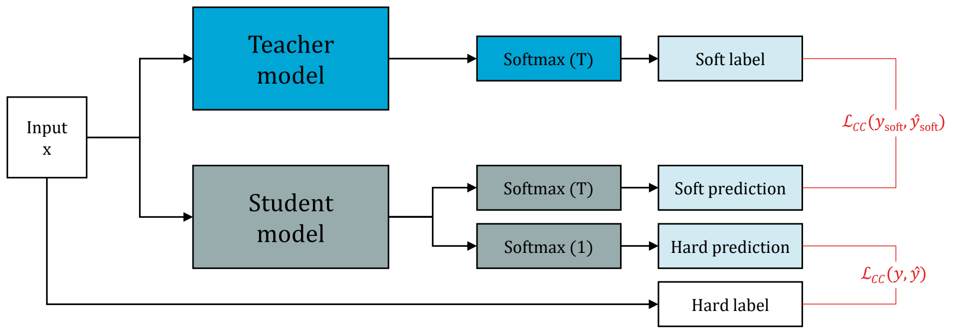


Figure 3.6: Schematic representation of knowledge distillation. $\mathcal{L}_{CC}$ denotes the categorical cross-entropy loss function.

Popular feature extractor networks include ResNet-50, ResNet-101 [10] and VGG16 [19], and are often pre-trained on the ImageNet dataset [20] containing over 1.5 million training images.

3.2.5. Knowledge distillation

ANNs are generally trained on datasets containing hard labels, e.g. when learning to recognize handwritten digits, an image of an '8' in the training set is labelled with 100% certainty to be an '8'. However, it is found that models trained on additional 'soft targets' achieve superior performance by using knowledge about resemblances between classes. The soft targets of said '8' might show that it is with 80% certainty an '8', but also displays a 10% resemblance with a '6' and a '9'. Knowledge distillation [21] is a method to improve the performance of a simple 'student' network by using the output probabilities produced by an expert 'teacher' network as soft targets. A schematic representation of this set-up is shown in figure 3.6. As expert networks are usually very cumbersome - they often contain an ensemble of large ANNs that average their predictions - the main benefit of knowledge distillation is that a significant reduction in model size can be achieved with only a minor performance loss.

The softmax activation function normalizes a layer such that the highest activation is very close to 1 and all other activations are very close to 0. As such, the output probabilities produced by a common softmax layer contain very little knowledge about class resemblances. The soft targets are therefore calculated by softening the logits (z_i), the outputs of the last layer before activation, using the softmax function defined as

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \tag{3.29}$$

where q_i denotes the output probability for class i and T is the temperature, which is normally set to 1. Higher values of T produce "softer" output probabilities that are more uniformly distributed, whereas a low temperature increases the differences between the output probabilities. As the expert network can still produce faulty predictions, the total loss function is a weighted average of the cross entropy loss of the soft targets and the cross entropy loss of the ground-truth labels and is defined as

$$\mathcal{L}_{KD} = \mathcal{L}(y_{\text{soft}}, \hat{y}_{\text{soft}}) + \lambda\mathcal{L}(y, \hat{y}). \tag{3.30}$$

The predicted label $\hat{y}$ is calculated by softening the student network logits with $T = 1$, whereas for the soft prediction $\hat{y}_{\text{soft}}$ the logits are softened using the same temperature as used for generating the soft targets from the teacher network. T and λ are additional hyperparameters that need to be defined prior to training. Since the gradients with respect to the loss, as used in backpropagation to update the weights, produced by the soft targets scale with $1/T^2$, λ is often also set to a value of approximately $1/T^2$ to ensure both terms of the loss function have a comparable contribution to the total gradient.

3.2.6. Performance metrics

The performance of an ANN can be evaluated using different metrics, depending on the computer vision task at hand. The two performance metrics that will be used throughout this thesis are Accuracy and Intersection over Union.

Accuracy

Accuracy is calculated as the percentage of correctly classified samples and is generally used for evaluating a classification task. Semantic segmentation methods can be evaluated by the pixel accuracy, which denotes the percentage of correctly classified pixels in an image.

Intersection over Union

Intersection over Union (IoU) is a popular metric used in both object detection and semantic segmentation. Given the ground-truth label GT and a prediction P, the IoU is defined as

$$IoU = \frac{GT \cap P}{GT \cup P}, \tag{3.31}$$

where a score of 1 denotes perfect overlap and 0 denotes no overlap at all. In object detection the object's bounding box is used, whereas in semantic segmentation the IoU is calculated using the area under the object expressed in the number of corresponding pixels. For multiclass detection or segmentation the IoU score is first calculated per object class and then averaged over all classes to arrive at the mean Intersection over Union (mIoU) score.

Bibliography

[1] Theo Gevers, Arjan Gijsenij, Joost van de Weijer, and Jan-Mark Geusebroek. *Color in Computer Vision: Fundamentals and Applications*. Wiley Publishing, 1st edition, 2012. ISBN 0470890843, 9780470890844.

[2] Steven A. Shafer. Using color to separate reflection components. *Color Research & Application*, 10 (4):210–218, 1985. doi: 10.1002/col.5080100409. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/col.5080100409`.

[3] Bruce Maxwell, Richard M. Friedhoff, and Casey A. Smith. A bi-illuminant dichromatic reflection model for understanding images. 06 2008. doi: 10.1109/CVPR.2008.4587491.

[4] P. Kubelka and F. Munk. Ein beitrag zur optik der farbanstriche. In *Zeitung fur Technische Physik*, volume 12, page 593, 1999.

[5] Thomas Young. *On the theory of light and colors [microform] / by Thomas Young ; read November 12, 1801.* The Society [London, 1802.

[6] J Von Kries. Influence of adaptation on the effects produced by luminous stimuli. In D. L. MacAdam, editor, *Sources of Color Vision*, pages 109–119. MIT Press, Cambridge (MA), 1970.

[7] Graham D. Finlayson, Bernt Schiele, and James L. Crowley. Comprehensive colour image normalization. In Hans Burkhardt and Bernd Neumann, editors, *Computer Vision — ECCV'98*, pages 475–490, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-69354-3.

[8] J. . Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and H. Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, Dec 2001. ISSN 0162-8828. doi: 10.1109/34.977559.

[9] J.M. Geusebroek and G.J. Burghouts. Amsterdam library of object images (ALOI). `aloi.science.uva.nl/`, 2004. Accessed: 2019-05-23.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

[11] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL `http://arxiv.org/abs/1506.02640`.

[12] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015. URL `http://arxiv.org/abs/1511.00561`.

[13] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. URL `http://dl.acm.org/citation.cfm?id=3104322.3104425`.

[14] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[16] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2014.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL `http://arxiv.org/abs/1502.03167`.

[18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 2488–2498, USA, 2018. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=3327144.3327174`.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. In *CVPR09*, 2009.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL `http://arxiv.org/abs/1503.02531`.

4 Experiments

This chapter contains both additional details to experiments already discussed in the scientific paper, as well as experiments to back certain claims that were made in the paper. It also includes methods that have been explored but did not result in the desired performance improvement.

4.1. Intensity robustness of CNNs

This section corresponds to section 4.2 of the scientific paper - *Illumination robustness of CNNs.*

Network architectures

The architectures used in experiment 4.2.2 - *What makes CNNs sensitive to illumination?* - have been briefly outlined in the scientific paper. This section gives a more detailed description and motivation for the variations on the base network (table 4.1a).

Residual connections Deeper ANNs are increasingly difficult to train. It is found that adding more layers to an already sufficiently deep model also leads to an increased training error [1], and as such the observed performance degradation is not caused by overfitting but simply by the model failing to converge. The motivation behind the residual learning block [2] is that given an input x and a desired mapping $\mathcal{H}(x)$, to be learnt within a subset of layers in the neural network, it is easier to fit the residual function of the mapping with respect to its inputs $\mathcal{F}(x) := \mathcal{H}(x) - x$, such that the final mapping is defined as $\mathcal{F}(x)+x$. Residual connections indeed enable deeper networks to be trained and are therefore used in most state-of-the-art architectures. A residual learning block including two convolutional layers is shown in figure 4.1. The network architecture in table 4.1b includes two such residual learning blocks (ResBlocks).

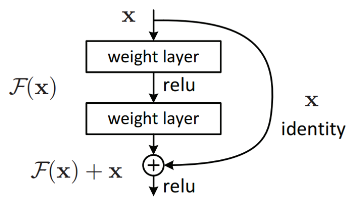

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4.1: A residual learning block [2].

Batch Normalization BN has been added between every Conv2D layer and its ReLU activation in the base network, resulting in the network architecture shown in table 4.1b.

Global Average Pooling In the network architecture in table 4.1d the Flatten layer has been replaced by Global Average Pooling. This resulted in a significant dimensionality reduction from 288 to 32 neurons in the final fully-connected layer and as such a parameter reduction of 28% over the entire network.

Table 4.1: Network architectures used for the 3DMNIST classification problem.

(a) Base network architecture.

Layer	Filter	Output shape
Conv2D	7x7, 16, stride 2	48x48x16
ReLU	-	48x48x16
MaxPooling2D	4x4, stride 4	12x12x16
Conv2D	3x3, 32	12x12x32
ReLU	-	12x12x32
MaxPooling2D	4x4, stride 4	3x3x32
Flatten	-	288
Dense	288x10 fc	10
Softmax	-	10
9,898 parameters		

(b) Network architecture with ResBlocks.

Layer	Filter	Output shape
Conv2D	7x7, 16, stride 2	48x48x16
ReLU	-	48x48x16
MaxPooling2D	4x4, stride 4	12x12x16
ResBlock	-	12x12x16
Conv2D	3x3, 32	12x12x32
ReLU	-	12x12x32
MaxPooling2D	4x4, stride 4	3x3x32
ResBlock	-	3x3x32
Flatten	-	288
Dense	288x10 fc	10
Softmax	-	10
33,034 parameters		

(c) Network architecture containing Batch Normalization layers.

Layer	Filter	Output shape
Conv2D	7x7, 16, stride 2	48x48x16
Batch Norm.	-	48x48x16
ReLU	-	48x48x16
MaxPooling2D	4x4, stride 4	12x12x16
Conv2D	3x3, 32	12x12x32
Batch Norm.	-	12x12x32
ReLU	-	12x12x32
MaxPooling2D	4x4, stride 4	3x3x32
Flatten	-	288
Dense	288x10 fc	10
Softmax	-	10
10,090 parameters		

(d) Network architecture using Global Average Pooling (GAP) instead of a flatten layer.

Layer	Filter	Output shape
Conv2D	7x7, 16, stride 2	48x48x16
ReLU	-	48x48x16
MaxPooling2D	4x4, stride 4	12x12x16
Conv2D	3x3, 32	12x12x32
ReLU	-	12x12x32
MaxPooling2D	4x4, stride 4	3x3x32
GAP		32
Dense	288x10 fc	10
Softmax	-	10
7,338 parameters		

Performance degradation due to Global Average Pooling

Hypothesis 1 The first hypothesis presented in section 4.2.2 of the paper as to the performance degradation in the model using Global Average Pooling argues that the network is too small for correspondences between classes and feature maps to be enforced. We therefore extended the number of filters in the second Conv2D layer of the GAP model to 288 and repeated the 3DMNIST classification experiment. The results are shown in table 4.2. The extended GAP model outperforms the GAP model on all test sets, which confirms the hypothesis.

Table 4.2: Classification accuracy of the extended GAP model on 3DMNIST. Base and GAP model performances are included for reference. Extending the number of filters significantly improved performance as indicated in bold.

Model	**Base**	**GAP**	**Ext. GAP**
Parameters	9,898	7,338	47,018
Baseline	96.3	94.4	97.1
Dark	95.8	93.8	96.2
Local	92.3	**71.0**	**83.0**
Color	93.3	**42.2**	**57.8**

Hypothesis 2 In the second hypothesis, we claimed that uniformly darkening an image by a scale factor also uniformly reduces the activations in a neural network by approximately the same scale factor, even when ReLU activation functions are used. To verify this claim, we compute the ratios between the neuron activations in the extended GAP model given different input image pairs. The first image pair contains the Baseline and Dark digits from figure 2 in the scientific paper. The relationship between the pixel values of the two images can be approximated by $\mathcal{I}_{\text{Dark}} = \alpha \mathcal{I}_{\text{Dark}}$, with $\alpha = 0.4$. The second and third image pair consists of the Baseline and Color, and the Baseline and Local digits, respectively. The results are shown as histograms in figure 4.2. The histogram corresponding to the Baseline-Dark image pair (left) clearly shows a peak around 0.4, corresponding to the scale factor α in the input image, whereas the ratios of the other input pairs are more uniformly distributed. This supports our claim.

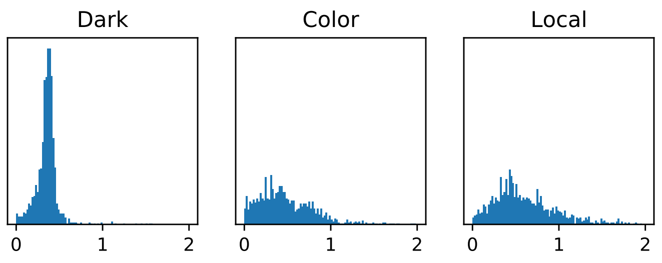


Figure 4.2: Ratios between neuron activations given different image pairs from the 3DMNIST dataset: Baseline-Dark, Baseline-Color and Baseline-Local. A uniform change in the input results in an approximately uniform change in activations.

Statistical distribution shift under changes in illumination

To support the claim that intensity shifts and changes in the illuminant color directly affect the color channel means and variances of an image, we report the statistics of the different test sets of the 3DMNIST dataset in table 4.3.

Table 4.3: First-order statistics of 3DMNIST test sets, reported as mean±std, RGB $\in [0, 1]$.

	R	**G**	**B**
Baseline	0.25 ± 0.22	0.48 ± 0.16	0.34 ± 0.04
Dark	0.10 ± 0.09	0.19 ± 0.06	0.14 ± 0.01
Local	0.14 ± 0.14	0.24 ± 0.36	0.18 ± 0.34
Color	0.10 ± 0.09	0.36 ± 0.12	0.34 ± 0.04

4.2. Nighttime Segmentation

This section corresponds to section 4.3 of the scientific paper - *Nighttime Segmentation*.

Convoluted Mixture of Deep Experts

We have evaluated the Convoluted Mixture of Deep Experts [3] (CMoDE) fusion method in addition to the fusion methods presented in the scientific paper.

CMoDE fuses the outputs of two expert networks trained on different modalities by means of class-wise adaptive weighting of their predictions. The class weights are produced by adaptive gating networks, which predict probabilistic confidence scores for the outputs of the expert networks based on the feature maps in their *conv4* layers. An additional convolutional layer followed by softmax activation finally produces the segmentation output. The CMoDE archtitecture is shown in figure 4.3.

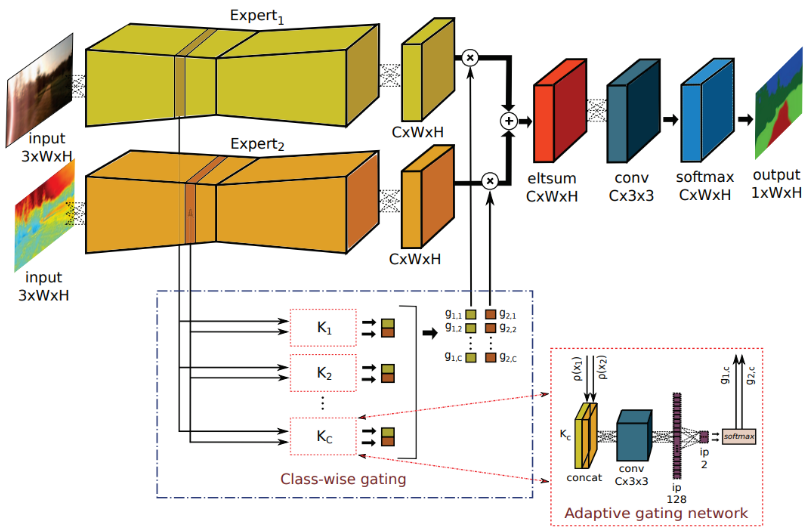


Figure 4.3: Convoluted Mixture of Deep Experts architecture [3].

The CMoDE network has been trained end-to-end on the CityScapes training set using both RGB and a color invariant as inputs. The pre-trained and fine-tuned models from table 8 in the scientific paper have been used as experts, whose weights have been frozen such that backpropagation only updates the weights of the adaptive gating networks and the final convolutional layer of the model.

The segmentation results on both the Cityscapes test set as well as on Nighttime Driving are shown in table 4.4. As CMoDE dynamically weighs the prediction outputs from the expert networks based on information from the multimodal input it is expected to outperform prediction averaging, at the least on the known source domain data. However, CMoDE only provides a marginal performance improvement compared to the RGB baseline and performs significantly worse than simple prediction averaging. This suggests that the adaptive gating networks were not successful in learning to predict correct confidence scores based on the input images. Further work is needed to assess why this is the case and how the class-wise probability predictions can be improved upon.

Bibliography

[1] K. He and J. Sun. Convolutional neural networks at constrained time cost. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5353–5360, June 2015. doi: 10.1109/CVPR.2015.7299173.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

Table 4.4: Day and night segmentation performance of CMoDE, measured by the mean IoU (%). RGB and prediction averaging are shown for reference.

Normalization	**Batch**			**Instance**		
Dataset	**Day**	**Night**	**Avg.**	**Day**	**Night**	**Avg.**
RGB (baseline)	*43.0*	*16.2*	*29.6*	49.8	26.9	38.4
CMODE - late fusion						
RGB + Comp	45.3	17.3	31.3	49.4	28.5	39.0
RGB + N_λ	44.3	19.8	32.1	49.5	28.4	39.0
Comp + N_λ	45.3	17.2	31.3	46.7	27.9	37.3
Prediction averaging - late fusion						
RGB + Comp	**57.1**	16.9	37.0	**63.7**	30.4	**47.1**
RGB + N_λ	55.9	**24.8**	**40.4**	63.4	**30.8**	**47.1**
Comp + N_λ	55.6	22.4	39.0	58.8	**30.8**	44.8

[3] A. Valada, J. Vertens, A. Dhall, and W. Burgard. Adapnet: Adaptive semantic segmentation in adverse environmental conditions. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4644–4651, May 2017. doi: 10.1109/ICRA.2017.7989540.

5 Deep-Learning Color Invariants

In the DRM-based color invariants introduced in section 3.1, dependency on scene geometry is factored out by dividing each pixel by the total pixel intensity I. As a result, these color invariants are noise-sensitive to low-intensity ($I = R + G + B \approx 0$) parts of images, as can clearly be seen in figure 3.2. While this thesis is mainly concerned with equipping ANNs with physics-based prior knowledge through the use of these color invariants, in this section we explore the exact opposite: can we learn a color invariant mapping with better noise characteristics from RGB data using deep learning as an optimization framework?

Inspired by recent work [1, 2] in the field of intrinsic image decomposition[1], we would like to train a CNN with two encoders to decompose an input image I into a reflectance component R and a shading component S such that the original image can be reconstructed by multiplication of the two components $I = R \times S$. As opposed to [1, 2] we would like to derive a color invariant mapping that does not use any spatial or contextual information from the input image but only depends on the RGB values of a single pixel.

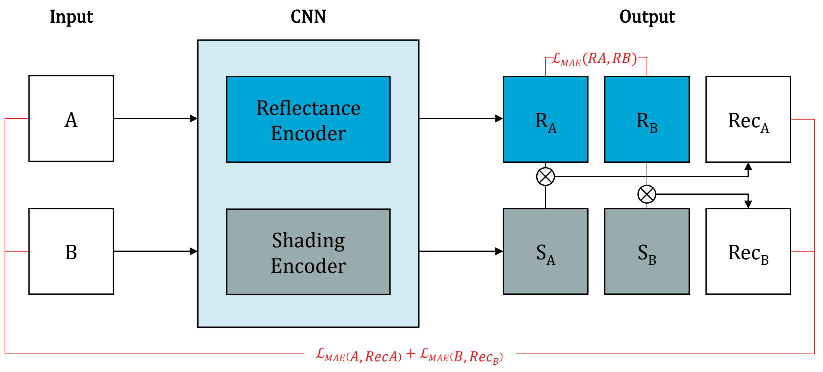
As intrinsic image decomposition is an ill-posed problem, accurate ground-truth shading and reflectance images are extremely difficult to obtain for real images. Therefore, most methods either make use of photo-realistic rendering to generate training data with corresponding ground-truth labels or rely on self-supervised training. Since we want our color invariant to approximate real-world conditions as closely as possible we employ the second strategy.

5.1. Method

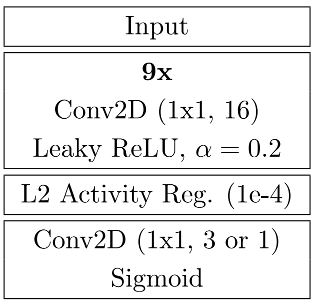
Our DLInv convolutional neural network consists of two separate encoders for the reflectance and shading components that are equivalent in architecture. To enforce the use of only pixel-level information, the entire encoder architecture solely consists of 1x1 convolutional layers, each followed by Leaky ReLU activations. L2 activity regularization is applied before the final convolutional layer. The reflectance encoder has an output of depth 3, corresponding to the RGB channels, whereas the shading component is represented in a single channel. The encoder architecture is shown in figure 5.1b. The output of the two encoders is multiplied with each other to reconstruct the original input image. The loss for training the network is defined as the mean average error (MAE) between the input image and the reconstruction.

As the decomposition problem is ill-posed - an image could be decomposed into a reflectance component containing the original image and a shading component being an all-ones matrix, resulting in a perfect reconstruction - an additional constraint is required that forces the reflectance component to represent a scene irrespective of its illumination conditions. We therefore train the encoders on two input images

[1]Intrinsic image decomposition is the computer vision task of breaking an image down into its shading and reflectance components, where the latter represents a scene irrespective of its recording conditions.


Input
CNN
Output
A
B
Reflectance Encoder
Shading Encoder
$\mathcal{L}_{MAE}(RA, RB)$
R_A
R_B
Rec_A
S_A
S_B
Rec_B
$\mathcal{L}_{MAE}(A, RecA) + \mathcal{L}_{MAE}(B, Rec_B)$

(a) DLInv decomposes an input image into a reflectance component R and a shading component S based on its individual pixel values. During training the difference between the two reflectances R_A and R_B as well as the reconstruction losses are minimized.



(b) Encoder architecture.

simultaneously, containing the same object taken from the same viewpoint but illuminated from a different position, and minimize the MAE between the two resulting reflectance components such that the illumination-related appearance changes are encoded in the shading components. The total loss function is thus given as

$$\begin{aligned}\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{MAE}}(A, R_A \times S_A) + \mathcal{L}_{\text{MAE}}(B, R_B \times S_B) + \mathcal{L}_{\text{MAE}}(R_A, R_B)\\ + \lambda\left[\mathcal{L}_2(R_A) + \mathcal{L}_2(S_A) + \mathcal{L}_2(R_B) + \mathcal{L}_2(S_B)\right], \quad (5.1)\end{aligned}$$

where A and B correspond to the two input images, R and S represent the reflectance and shading components, respectively, and $\lambda = 1e-4$ is the regularization weight parameter. Figure 5.1a denotes the CNN training setup. The reflectance and shading encoders contain 2,563 and 2,529 parameters, respectively, summing up to a total of 5,092 parameters for the complete DLInv network.

5.2. Experiments

Experimental setup

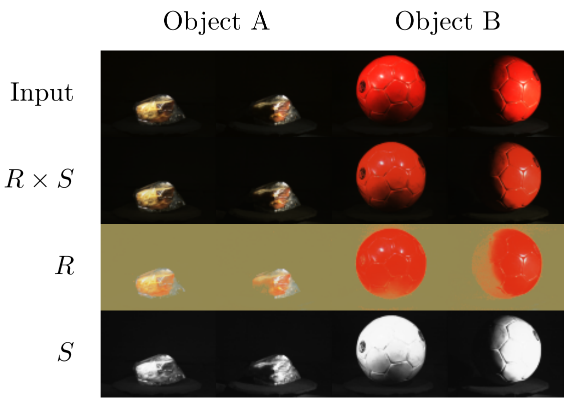
The network has been trained using the Adam [3] optimizer with an initial learning rate of 1e-3 and decay 1e-4. The training set consists of 800 image pairs from the ALOI [4] dataset, containing objects photographed under different illumination conditions. Each pair includes an image of an attribute illuminated from the viewpoint of the camera and an image showing the same object illuminated from the right side, casting heavy shadows on the left side of the object.

Results and analysis

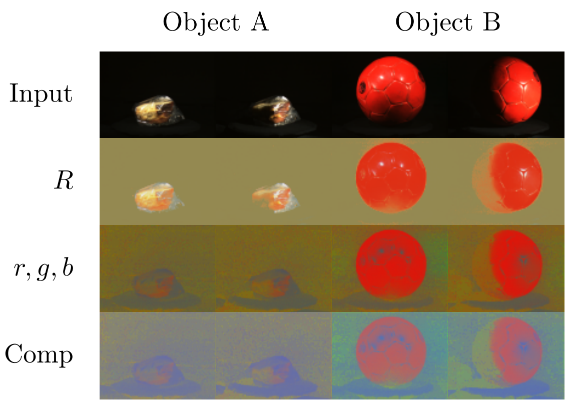
Two example input pairs and their resulting reflectance R and shading S components and reconstructions $R \times S$ are shown in figure 5.2a. Reflectance and shading have been near-perfectly decomposed in both examples and as a result the reconstructions are indistinguishable from the input images. Note that the reflectance component of the football even shows the object color in low-intensity parts of the input image (see [R, Object B] in figure 5.2a), where at the same time the amount of noise is limited.

Figure 5.2b compares the reflectance R produced by the DLInv network with the physics-based color invariants r, g, b (normalized RGB) and Comp (Comprehensive normalization). R not only exhibits more realistic color rendering, it is also significantly less sensitive to low intensity in the input image. Also note that both physics-based color invariants clearly show the shading of the turntable the attributes rest on, whereas DLInv encodes this effect in the shading component S instead.

Figure 5.3 shows the RGB encoding of hue, represented in the RGB, normalized RGB (r, g, b) and DLInv color spaces. RGB and DLInv show significant resemblances throughout almost the entire color spectrum, whereas the secondary colors yellow, magenta and cyan appear dark in normalized RGB.



(a) Input image pairs and their reflectance R and shading S components, resulting in reconstruction $R \times S$.



(b) Reflectance R compared to the more noisy physics-based color invariants r, g, b (normalized RGB) and Comp (Comprehensive normalization).

Figure 5.2: Outputs produced by the DLInv network. Both images were not part of the training set.

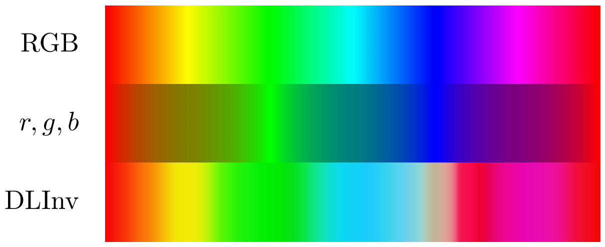


Figure 5.3: RGB encoding of hue, shown in RGB, normalized RGB (r, g, b) and DLInv. DLInv is more similar to RGB but fails to correctly map the range of blue colors.

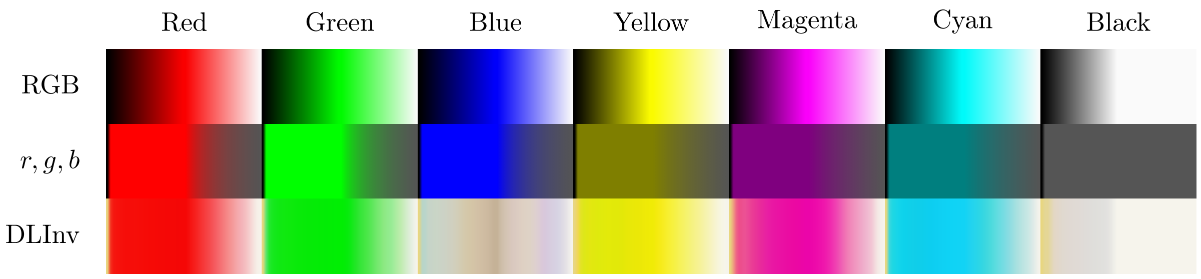


Figure 5.4: RGB, normalized RGB (r, g, b) and DLInv representations of the primary colors red, green and blue, the secondary colors yellow, magenta and cyan, and grayscales.

DLInv has failed to learn a correct mapping for the blue part of the color spectrum, which is now shown in beige. This is most likely a result of blue being underrepresented in the ALOI dataset. Figure 5.4 shows the primary and secondary colors of the RGB color space in RGB, normalized RGB (r, g, b) and DLInv representations. For the primary colors DLInv can be seen as a composition of the RGB and r, g, b color spaces: dark colors are mapped to r, g, b and light colors to RGB, with the exception of the incorrectly mapped blue color. For secondary colors, DLInv behaves like the RGB color space but removes the dark component of the color, which is now encoded in the shading component S. Finally, grayscale values are mostly removed from the DLInv mapping, while white is preserved. Note that r, g, b is unable to distinguish between any of the grayscale values as they lack saturation $(R = G = B)$.

Real-world applications

We applied the DLInv color invariant on semantic segmentation of street images. The experimental setup is identical to the experiment described in section 4.3 of the scientific paper. Figure 5.5 shows an input image in the daytime and nighttime domain. Note that the original colors are better preserved in the DLInv representation, but both color invariants are unable to completely factor out the shadow in the daytime image and are heavily affected by artificial lighting in the nighttime image.

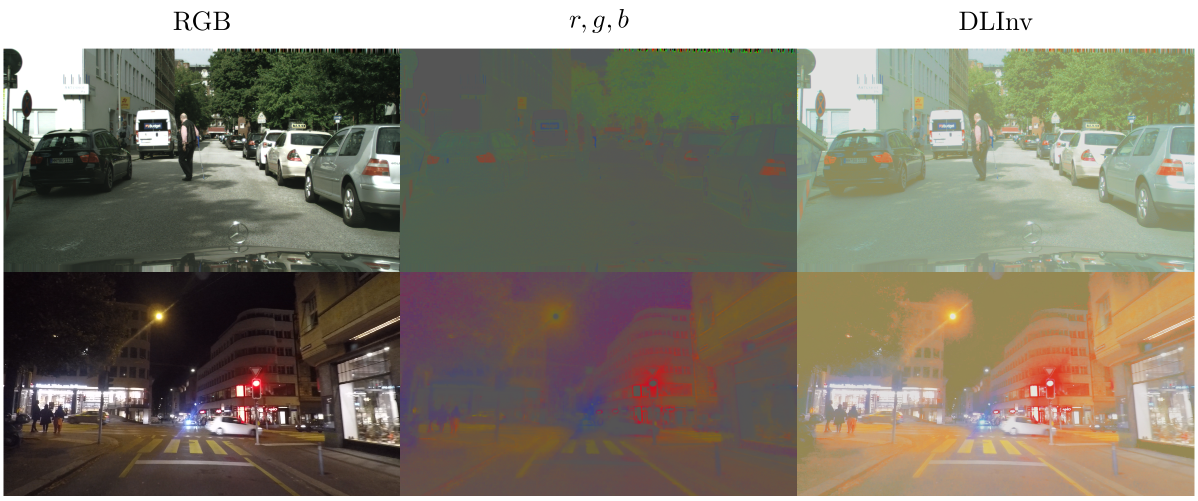


Figure 5.5: RGB, normalized RGB (r, g, b) and DLInv representations of daytime and nighttime street image data from the Cityscapes [5] and Nighttime Driving [6] datasets.

Quantitative segmentation results are given in table 5.1. DLInv only performs marginally better than comprehensive image normalization (Comp) and performs significantly worse than RGB and N_λ in the daytime domain.

Table 5.1: Day-night segmentation performance using RGB and color invariant inputs measured by the mean IoU (%). DLInv does not improve segmentation performance.

Normalization	**Batch**			**Instance**		
Dataset	**Day**	**Night**	**Avg.**	**Day**	**Night**	**Avg.**
RGB (baseline)	***43.0***	*16.2*	*29.6*	**49.8**	**26.9**	**38.4**
Comp	42.8	8.4	25.6	44.3	21.1	32.7
N_λ	40.7	**21.7**	**31.2**	41.2	22.2	31.7
DLInv	42.0	9.9	26.0	44.3	21.2	32.8

5.3. Discussion

The experiments have shown that it is possible to learn a color invariant mapping using nothing but data and self-supervised deep learning. DLInv has better noise characteristics than DRM-based color invariants, in the sense that it is able to completely remove dark components from colors, as shown in figure 5.4. However, solid black is mapped to a seemingly arbitrary neutral color in the reflectance component R. While this is self-evident, as a 0 in the shading component S results in a zero in the reconstruction for any value in R, one might argue that a black mapping in R would be more natural. It is worthwhile to explore whether this can be achieved through additional regularization.

Future work also includes further improving the color invariant mapping and investigating how to further optimize the network architecture in terms of performance and number of parameters, possibly by including physics-based reflection models in the network architecture itself as extra constraints. It is also interesting to examine the opposite, namely whether we can use a well-performing deep-learnt color invariant to improve upon its physics-based counterparts.

Bibliography

[1] Michael Janner, Jiajun Wu, Tejas D. Kulkarni, Ilker Yildirim, and Joshua B. Tenenbaum. Self-supervised intrinsic image decomposition. *CoRR*, abs/1711.03678, 2017. URL `http://arxiv.org/abs/1711.03678`.

[2] Zhengqi Li and Noah Snavely. Learning intrinsic image decomposition from watching the world. *CoRR*, abs/1804.00582, 2018. URL `http://arxiv.org/abs/1804.00582`.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[4] J.M. Geusebroek and G.J. Burghouts. Amsterdam library of object images (ALOI). `aloi.science.uva.nl/`, 2004. Accessed: 2019-05-23.

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Dengxin Dai and Luc Van Gool. Dark model adaptation: Semantic image segmentation from daytime to nighttime. pages 3819–3824, 11 2018. doi: 10.1109/ITSC.2018.8569387.