ABSTRACT

Title of dissertation: USING CNNS TO UNDERSTAND LIGHTING WITHOUT REAL LABELED TRAINING DATA

Hao Zhou
Doctor of Philosophy, 2019

Dissertation directed by: Professor David W. Jacobs
Department of Computer Science

The task of computer vision is to make computers understand the physical word through images. Lighting is the medium through which we capture images of the physical world. Without lighting, there is no image, and different lighting leads to different images of the same physical world. In this dissertation, we study how to understand lighting from images.

With the emergence of large datasets and deep learning in recent years, learning based methods play a more and more important role in computer vision, and deep Convolutional Neural Networks (CNNs) now dominate most of the problems in computer vision. Despite their success, deep CNNs are notorious for their data hungry nature compared with traditional learning based methods. While collecting images from the internet is easy and fast, labeling those images is both time consuming and expensive, and sometimes, even impossible. In this work, we focus on understanding lighting from faces and natural scenes, in which ground truth labels of the lighting are impossible to achieve.

As a preliminary topic, we first study the capacity of deep CNNs. Designing deep CNNs with less capacity and good generalization is one way to reduce the amount of labeled data needed in training deep CNNs, and understanding the capacity of deep CNNs is the first step towards that goal. In this work, we empirically study the capacity of deep CNNs by studying the redundancy of parameters in them. More specifically, we aim at optimizing the number of neurons in a network, thus the number of parameters. To achieve that goal, we incorporate sparse constraints into the objective function and apply a forward-backward splitting method to solve this sparse constrained optimization problem efficiently. The proposed method can significantly reduce the number of parameters, showing that networks with small capacity can work well.

We then study an important problem in computer vision: inverse lighting from a single face image. Lacking massive ground truth lighting labels, we generate a large amount of synthetic data with ground truth lighting to train a deep network. However, due to the large domain gap between real and synthetic data, the network trained using synthetic data cannot generalize well to real data. We thus propose to use real data to train the deep CNN together with synthetic data. We apply an existing method to estimate lighting conditions of real face images. However, these lighting labels are noisy. We then propose a Label Denoising Adversarial Network (LDAN) to make use of these synthetic data to help train a deep CNN to regress lighting from real face images, denoising labels of real images. We have shown that the proposed method can generate more consistent lighting for faces taken under the same lighting condition.

Third, we study how to relight a face image using deep CNNs. We formulate this problem as a supervised image to image translation problem. Due to the lack of a "in the wild" face dataset that is suitable for this task, we apply a physically-based face relighting method to generate a large scale, high resolution, "in the wild" portrait relighting dataset (DPR). A deep Convolutional Neural Network (CNN) is then trained using this dataset to generate a relighted portrait image by taking a source image and a target lighting as input. We show that our training procedure can regularize the generated results, removing the artifacts caused by physically-based relighting methods.

Fourth, we study how to understand lighting from a natural scene based on an RGB image. We propose a Global-Local Spherical Harmonics (GLoSH) lighting model to improve the lighting representation, and jointly predict reflectance and surface normals. The global SH models the holistic lighting while local SHs account for the spatial variation of lighting. A novel non-negative lighting constraint is proposed to encourage the estimated SHs to be physically meaningful. To seamlessly make use of the GLoSH model, we design a coarse-to-fine network structure. Lacking labels for reflectance and lighting, we apply synthetic data for model pre-training and fine-tune the model with real data in a self-supervised way. We have shown that the proposed method outperforms state-of-the-art methods in understanding lighting, reflectance and shading of a natural scene.

USING CNNS TO UNDERSTAND LIGHTING WITHOUT REAL LABELED TRAINING DATA

by

Hao Zhou

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor David W. Jacobs, Chair/Advisor
Professor Larry S. Davis
Professor Rama Chellappa
Professor Tom Goldstein
Dr. Yaser Yacoob

In memory of my grandfather

Acknowledgments

First and foremost, I would like to express my deepest appreciation to my advisor Prof. David W. Jacobs, an excellent research mentor who helped me generously during my PhD study. David provided me the precious opportunity to do research studies in computer vision under his supervision six years ago. Ever since then, he helped me find my own research interests and helped me on how to become a qualified researcher. David was always very patient and encouraged me when I met difficulties, especially for the first two years when I struggled to find my own research interests. He was accommodating to my immature ideas and helped me to turn those ideas into mature research projects. His enthusiasm for research and curiosity in all kinds of new knowledge has greatly influenced me. It has been a great pleasure to work with and learn from him.

I would like to extend my sincere thanks to my dissertation committee members: Prof. Larry S. Davis, Rama Chellappa, Tom Goldstein and Dr. Yaser Yacoob for their insightful comments and suggestions for this dissertation. I also thank Prof. Larry S. Davis for his advice on Medifor project and his kind help in my job search.

I would like to thank Dr. Jose M. Alvarez and Prof. Fatih Porikli for their great help during my internship in NICTA. They introduced me to the filed of deep learning. I am also grateful to Dr. Quoc-Huy Tran and Prof. Manmohan Chandraker, who were my mentors in NEC Labs America. It was a great pleasure to work with them on a research project that went beyond the scope of my own research. I learned a lot from the way they work and their enthusiasm for research.

I especially thank Prof. Manmohan Chandraker for his kind suggestions during my job search. I owe my thanks to Dr. Sunil Hadap and Dr. Kalyan Sunkavalli for their invaluable advice on the deep portrait relighting project during my internship in Adobe Research.

I am thankful for the support and insightful discussions from my great lab mates: Joao Soares, Angjoo Kanazawa, Jin Sun, Soumyadip Sengupta, Abhay Yadav, Ryen Krusinga, Koutilya PNVR and Daniel Lichy.

I would like to thank all my other collaborators who are not mentioned above for their helpful contributions to my research work: Torsten Sattler, Ronen Basri, Xiang Yu, Hui Ding, Hong Wei.

Many thanks to all other friends at the University of Maryland: Shanshan Li, Jingjing Zheng, Pan Xu, Junhui Li, Hao Li, Yaming Wang, Xiyang Dai, Zebao Gao, Xing Niu, Jinfeng Rao, Xintong Han, Ruofei Du, Hongyu Xu, Peng Zhou, Xitong Yang, Zuxuan Wu and Jun-Cheng Chen. The time we spent together was unforgettable. Special thanks to Pan Xu for listening to my complaints when I met difficulties and helped me get out of bad mood.

I am grateful for my house mate: Zheng Xu, Peng Lei, Chen Zhao, Chunfeng Yang, Xue Li, Han Zhou and Zuolin Tian for their help and support in daily life.

Last but not least, I owe my deepest thanks to my family for their love and unconditional support. They always stand behind me and help me get through all those challenging times.

Table of Contents

Dedication ii

Acknowledgements iii

Table of Contents v

List of Tables viii

List of Figures ix

1 Introduction 1
1.1 Empirical study of the capacity of deep CNNs. 3
1.2 Training deep CNNs with synthetic data. 4
1.2.1 Label Denoising Adversarial Networks (LDAN) for Inverse Lighting of Faces 5
1.2.2 Deep Single-Image Portrait Relighting 7
1.2.3 GLoSH: Global-Local Spherical Harmonics for Intrinsic Image Decomposition 8
1.3 Summary 9

2 Less is More: Towards Compact CNNs 10
2.1 Introduction 10
2.2 Related Work 13
2.3 Sparse Constrained Convolutional Neural Networks 16
2.3.1 Training a Sparse Constrained CNN 16
2.3.2 Forward-Backward Splitting 18
2.3.3 Sparse Constraints 20
2.3.4 Importance of Rectified Linear Units in Sparse Constrained CNNs 22
2.4 Experiments 24
2.4.1 LeNet on MNIST 25
2.4.2 CIFAR-10 quick on CIFAR-10 33
2.4.3 AlexNet and VGG on ImageNet 35
2.5 Summary 36

3 Label Denoising Adversarial Networks (LDAN) for Inverse Lighting of Faces 38
3.1 Introduction . . . 38
3.2 Related Work . . . 42
3.3 Proposed Method . . . 44
3.3.1 Spherical Harmonics . . . 44
3.3.2 Label Denoising Adversarial Network . . . 46
3.4 Experiments . . . 49
3.4.1 Data Collection . . . 49
3.4.2 Implementation Details . . . 50
3.4.3 Evaluation Metric . . . 53
3.4.4 Experimental Results . . . 53
3.4.5 Object 2D Keypoints Detection . . . 59
3.5 Summary . . . 64

4 Deep Single-Image Portrait Relighting 65
4.1 Introduction . . . 66
4.2 Related Work . . . 69
4.3 Deep Portrait Relighting Dataset . . . 71
4.3.1 Ratio Image to Relight Faces . . . 71
4.3.2 Normal Estimation . . . 72
4.3.2.1 ARAP Based Normal Refinement . . . 73
4.3.3 Relighting Images . . . 75
4.4 Method . . . 76
4.4.1 Main architecture for portrait relighting . . . 76
4.4.2 Supervision for training the network . . . 77
4.4.3 Skip Training . . . 78
4.4.4 Implementation Details . . . 80
4.4.4.1 Network Structure . . . 80
4.4.4.2 Training Detail . . . 81
4.5 Experiments . . . 83
4.5.1 Dataset and Evaluation Metric . . . 83
4.5.2 Ablation Study . . . 84
4.5.3 Comparison with the Rendering Pipeline . . . 86
4.5.4 Comparison with State-of-the-art Methods . . . 87
4.5.5 Results on challenging images . . . 91
4.5.6 Visual Results on High Resolution Images . . . 91
4.6 Summary . . . 94

5 GLoSH: Global-Local Spherical Harmonics for Intrinsic Image Decomposition 95
5.1 Introduction . . . 96
5.2 Related Work . . . 100
5.3 Reflectance, Normal and Shading from a Single RGB Image . . . 102
5.3.1 GLoSH Lighting Modeling . . . 102
5.3.1.1 Global and local Spherical Harmonics . . . 103
5.3.1.2 Non-negative Constraints on SH . . . 103

5.3.2 Coarse-to-fine Network Structure . . . 104
5.3.3 Supervision on Training . . . 106
5.3.3.1 Reflectance . . . 106
5.3.3.2 Normal . . . 107
5.3.3.3 Shading . . . 108
5.4 Network Structure . . . 109
5.5 Implementation Details . . . 112
5.6 Experiments . . . 114
5.6.1 Datasets . . . 114
5.6.2 Spherical Harmonics Lighting Evaluation . . . 115
5.6.3 Intrinsic Image Decomposition . . . 120
5.6.4 Ablation Study . . . 122
5.7 Summary . . . 128

6 Conclusion 129

A Evaluating Local Features for Day-Night Matching 132
A.1 Introduction . . . 132
A.2 Dataset . . . 135
A.3 Evaluation . . . 136
A.3.1 Keypoint Detectors . . . 136
A.3.2 Repeatability of Detectors . . . 138
A.3.3 Matching Day-Night Image Pairs . . . 141
A.4 Potential of Improving Detectors . . . 147
A.5 Summary . . . 149

Bibliography 151

List of Tables

2.1 Neuron reduction in the first fully connected layer, the total parameter compression, reduced memory, the top-1 validation error rate (red: error rate without sparse constraints). 12
2.2 Results of adding group sparse constraints on three layers. 33
2.3 Results of adding group sparse constraints on two layers. The best compression results within 1% decrease in top 1 error rate is shown in bold. 37
2.4 Some compression results of proposed method on fc1 for vgg-B. Neuron: compression of neurons in the fc1. Parameter: compression of total parameters. 37

3.1 Accuracy of different methods. Standard deviation is shown in the parentheses for learning based methods. 49
3.2 Results of ablation study. Standard derivation is shown in the bracket. 54
3.3 Accuracy of LDAN for different scales of face images. 58
3.4 PCK value at $\alpha = 0.1$ for different methods. We notice that LDAN outperforms regression and fine-tune. 59

4.1 Details about each block of our network. 80
4.2 Ablation Study on MultiPie Dataset 85
4.3 Evaluation MultiPie Dataset 87

5.1 Details of each block in our network. 110
5.2 Details about each block in our second and third scale network. . . . 111
5.3 SH lighting Evaluation on SUNCG synthetic data. 115
5.4 Surface normal evaluation on NYUv2. 117
5.5 Reflectance evaluation on IIW and shading evaluation on SAW. . . . 121
5.6 Ablation study on loss, without synthetic SUNCG data, and the coarse-to-fine scales, evaluated on IIW reflectance, SAW shading and NYUv2 surface normal. 126

List of Figures

1.1 Illustration of LDAN. 6

2.1 Importance of Rectified Linear Units in Sparse Constrained CNNs. . 26
2.2 The role of momentum in compressing CNNs. 27
2.3 Comparison with Data-free method on MNIST. 29
2.4 Comparing with l_0 constraints. 30
2.5 Distribution of Norm of neurons. 31
2.6 Compare with Data-free method on ImageNet. 35

3.1 Domain adaptation with and without adversarial loss. 41
3.2 Structure of network for lighting regression. 52
3.3 Two baseline models compare with LDAN. 55
3.4 Visualization results on MultiPie. 56
3.5 Visualization results on CelebA. 57
3.6 Illustration of network for keypoints regression. 60
3.7 Structure of network for keypoints regression 62
3.8 PCK curve for keypoints regression. 63

4.1 Relighted images for Obama. 65
4.2 ARAP based normal refinement. 73
4.3 Visual comparison of normals estimated by 3DDFA and the proposed method. 74
4.4 Examples of relighted images using the proposed rendering pipeline. . 75
4.5 Illustration of network structure. 76
4.6 Illustration of the effect of skip layers. 79
4.7 Visual comparison of Hourglass network with and without skip training. 79
4.8 Illustration of repeating lighting feature spatially. 81
4.9 Network structure for 1024×1024 images. 81
4.10 Visual results of ablation study. 85
4.11 Visual comparison of our rendering pipeline and the proposed deep portrait relighting method. 87
4.12 Visual results of the proposed method and state-of-the-art methods on MultiPie. 89

4.13 Visual comparison of the proposed method with state-of-the-art methods. . . . 90
4.14 Some challenging examples. . . . 92
4.15 Results on flicker portrait dataset [1]. . . . 93

5.1 Visual comparison of the proposed method with [2] 95
5.2 Illustration of the effectiveness of the proposed GLoSH model. . . . 98
5.3 Our coarse-to-fine network structure. . . . 104
5.4 Network structure for the first scale. . . . 109
5.5 Network structure for the second and third scale. . . . 110
5.6 Illustration of lighting predicted with and without non-negative constraint. . . . 115
5.7 1. Comparison with [2]. . . . 117
5.8 2. Comparison with [2] 118
5.9 3. Comparison with [2]. . . . 119
5.10 1. Comparison with state-of-the-art intrinsic image decomposition method. . . . 120
5.11 2. Comparison with state-of-the-art intrinsic image decomposition method. . . . 123
5.12 3. Comparison with state-of-the-art intrinsic image decomposition method. . . . 124
5.13 4. Comparison with state-of-the-art intrinsic image decomposition method. . . . 125

A.1 Images taken from 00:00 - 23:00 in one image sequence of our dataset. 136
A.2 The number of feature points detected at different time 138
A.3 Average number of ground truth feature points and the repeatability for different detectors. . . . 140
A.4 Precision of RootSIFT for day-night image matching for different detectors and their number of correct matched feature points. . . . 143
A.5 Histogram of scales for correctly matched RootSIFT features and comparison of precision for TILDE4 and MultiscaleTILDE4. . . . 145
A.6 Recall of RootSIFT matching of day-night image pairs at different time. . . . 145
A.7 Precision and recall of matching day-night image pairs using cnn feature. 146
A.8 Precision and number of correct matches of dense RootSIFT for day-night image pairs. . . . 147
A.9 Correct matches of DoG+RootSIFT and dense RootSIFT. . . . 148
A.10 Examples of nighttime and daytime image with detected feature points using DoG and the heat map of their cosine distance of dense RootSIFT. 149

Chapter 1: Introduction

In nature, light creates the color. In the picture, color creates the light.

– Hans Hofmann

The task of computer vision is to make computers understand the physical world through images. Lighting, as the medium through which we capture images of the physical world, is a key factor to understand the physical world through images. For instance, the same world under different lighting conditions can form quite different images. As a result, the three general topics in computer vision: recognition, registration and reconstruction (known as the 3Rs) are all significantly affected by lighting.

In this dissertation, we study how to understand lighting from images. More specifically, we study how to estimate lighting conditions from faces and natural scenes through an RGB image. With the emergence of large datasets and deep learning in recent years, learning based methods play a more and more important role in computer vision, and Deep Convolutional Neural Networks (CNNs) [3] now dominate almost all problems of the 3Rs. Deep CNNs first achieved the best performance in many recognition problems [4–7]. Then, researchers began to apply deep CNNs to registration [8, 9] and reconstruction [10–12] and achieved state-of-the-art

performance.

Inspired by the success of Deep CNNs in the 3Rs, we propose to apply it to understand lighting from images. However, despite their success in the 3Rs of computer vision, deep CNNs are notorious for their data hungry nature compared with traditional learning based methods. Taking image classification as an example; to train a deep CNN that performs reasonably well, researchers need to feed hundreds of thousands of labeled images into the deep network. While collecting images from the internet is easy and fast, collecting labels for those images is both time consuming and expensive, and sometimes, even impossible (e.g. ground truth lighting of faces and natural scenes) [13].

There are several ways to save the human labor of labeling images: designing user friendly interfaces for labeling data; applying active learning to label the most informative data; using domain adaption to make use of a large dataset with labels to help the target task which has a small amount of data; unsupervised learning in which labels are not needed, to name a few. Inspired by the recent progress of rendering techniques in graphics, we propose to use synthetic data to alleviate the large demands of real labels in lighting estimation. In this dissertation, we first empirically study the capacity of current deep CNNs to get an idea of how to design deep CNNs with less capacity. We then apply deep CNNs to estimate lighting from faces with the help of synthetic face images. A portrait relighting algorithm is then proposed, in which physically based rendering is used to generate a large amount of high quality, "in the wild" face images as training data. Finally, we propose to estimate lighting together with reflectance and normal from a natural scene using an

RGB image, where synthetic images with ground truth labels are used to pre-train our network.

1.1 Empirical study of the capacity of deep CNNs.

One way to reduce the heavy demands of data for training deep CNNs is to design deep CNNs with less capacity but good generalization. Analyzing the capacity of a deep CNN is the first step towards that goal. However, theoretically analyzing the capacity of a network is difficult due to its complex structure. In Chapter 2, we empirically study the capacity of deep CNNs by studying how many parameters of deep CNNs can be removed without affecting their performance.

To attain favorable performance on large-scale datasets, a common practice is to design CNNs with a lot of parameters. However, recent studies have shown that the capacities of CNNs are much larger than necessary [14–17]. Inspired by these works, we proposed to reduce the number of parameters. More specifically, we aim at optimizing the number of neurons in a network, thus the number of parameters.

To achieve that goal, we incorporate sparse constraints into the objective function. The forward-backward splitting method [18, 19] is applied to solve this sparse constrained optimization problem efficiently. The main advantage of forward-backward splitting is that it bypasses the sparse constraint evaluations during the standard back-propagation step, making the implementation very practical. We also investigate the importance of rectified linear units (ReLU) [20] in sparse constrained CNNs, showing that using ReLU can lead to more pruned neurons. We study two

sparse constraints: tensor low rank [21] and group sparsity [22] and carry out experiments on four well-known models (LeNet [3], CIFAR-10 quick [23], AlexNet [4] and VGG [5]) using three public datasets including ImageNet. Our experiments demonstrate that the proposed method can reduce a huge number of parameters during the training stage, showing that a network with small capacity can work well.

1.2 Training deep CNNs with synthetic data.

Though getting ground truth labels for real data is difficult, generating synthetic data with ground truth labels is usually easy and cheap. As a result, synthetic data is becoming more and more popular to help train deep CNNs. [12, 24–30] are some examples of applying synthetic data to train deep CNNs. [27] and [30] use synthetic data as an intermediary to bridge between real data and their labels. [12,24,26] directly train their deep CNNs on synthetic data and apply them to real data without considering the domain gap between synthetic and real data. [29] argued that domain adaptation is necessary to better make use of synthetic data and proposed a simple strategy that consists of training synthetic and real data simultaneously. [25, 28], on the other hand, applied adversarial loss to bridge the domain gap between synthetic and real data.

1.2.1 Label Denoising Adversarial Networks (LDAN) for Inverse Lighting of Faces

In Chapter 3, we propose to train a deep CNN for inverse lighting from a single face image. Since getting ground truth lighting labels for face images in the wild is difficult, inspired by the above mentioned methods, we propose to apply synthetic data to help train a deep CNN. We notice that a deep CNN trained only on synthetic data can not generalize well to real world face images due to the big domain gap between synthetic and real data. To reduce this gap, we also apply real data while training the network. We use an existing method [31] to estimate lighting parameters for real face images, which are treated as ground truth with noise. Synthetic data with noise free ground truth labels are then used to help alleviate the effect of such noise. More specifically, we utilize the idea of Generative Adversarial Networks (GAN) [32] and propose a Label Denoising Adversarial Network (LDAN).

We design the lighting regression deep CNN to have two sub-networks: a feature net that extracts lighting related features and a lighting net that takes these features as input and predicts the lighting. Since synthetic data contains no noise, the lighting net trained with synthetic data is accurate; we thus directly apply it to real data. As a result, we only need to train a feature network for real data. As the lighting net expects lighting related features for synthetic data as input, while training the feature net for real data, we utilize the idea of Generative Adversarial Networks (GAN) [32] to map the distribution of lighting related features for real data to that of synthetic data. The training of LDAN thus has two steps: 1) Train

with synthetic data; (2) Fix the feature net for synthetic data and the lighting net, train another feature net for real data with adversarial loss and regression loss. Figure 1.1 illustrates the training procedure of LDAN.

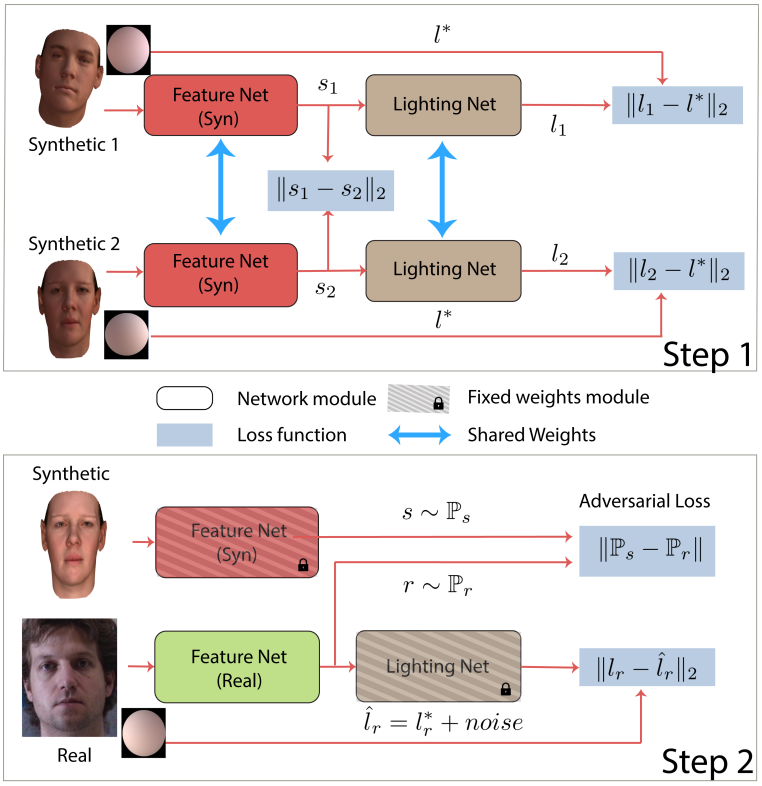


Figure 1.1: Training of a LDAN model has two steps: 1) Train the feature net and lighting net for synthetic data with two losses: Faces with similar lighting should have similar lighting related features ($||s_1 - s_2||_2$); Estimated lighting should be close to ground truth lighting ($||l_1 - l^*||_2$ and $||l_2 - l^*||_2$). 2) Train the feature net for real data while fixing both the feature net and lighting net trained in step 1. We use two losses in this step: The distribution of synthetic features and real features should be close ($||\mathbb{P}_s - \mathbb{P}_r||$); Estimated lighting should be close to noisy ground truth lighting ($||l_r - \hat{l}_r||$).

We test our proposed method on the MultiPie data set. While the lighting for each image is not known, there are groups of images that all have the same lighting. Our experiments show that our network outperforms existing methods in producing consistent lighting parameters of different faces under similar lighting conditions.

1.2.2 Deep Single-Image Portrait Relighting

In Chapter 4, we propose to train a deep CNN for single-image portrait relighting. Conventional physically-based methods need to solve an inverse rendering problem, estimating face geometry, albedo and lighting. However, the inaccurate estimation of face components can cause strong artifacts in relighting, leading to unsatisfactory results. In the proposed work, we apply a physically-based portrait relighting method to generate a large scale, high resolution, "in the wild" portrait relighting dataset (DPR). A deep Convolutional Neural Network (CNN) is then trained using the proposed data for portrait relighting. We show that the training procedure regularizes the generated results, removing the artifacts caused by physically-based relighting methods.

We design our network to have an hourglass structure [33] which takes a source image and a target lighting as input and generates a new portrait image under the target lighting condition. We notice that skip connections in an hourglass network prevent the bottleneck layers from learning good facial information, causing low quality of the generated face images. A skip training strategy is thus proposed to enforce more facial information in the bottleneck layer which can also improve the quality of the generated image. A GAN loss is further applied to alleviate the artifacts in the relighted portrait images. Our trained network can relight portrait images with resolutions as high as 1024×1024. We evaluate the proposed method on the proposed DPR datset, flickr portrait dataset and MultiPie dataset both qualitatively and quantitatively. Our experimental results demonstrate that the

proposed method achieves state-of-the-art results.

1.2.3 GLoSH: Global-Local Spherical Harmonics for Intrinsic Image Decomposition

In Chapter 5, we study how to estimate lighting from a natural scene with an RGB image. Traditional intrinsic image decomposition focuses on decomposing images into reflectance and shading, leaving surface normals and lighting entangled in shading. One challenge of estimating lighting from a natural scene is the lack of a suitable lighting model. Unlike the lighting of a single object such as faces which can be modeled using a single set of Spherical Harmonics [34, 35], the lighting of a natural scene is much more complicated due to its spatial variation caused by shadow, inter-reflection and the presence of light sources in the scene. In this work, we propose a Global-Local Spherical Harmonics (GLoSH) lighting model to better model the lighting of a natural scene. The global SH models the holistic lighting while local SHs account for the spatial variation of lighting. A novel non-negative lighting constraint is proposed to encourage the estimated SHs to be physically meaningful. To seamlessly reflect the GLoSH model, we design a coarse-to-fine network structure. The coarse network predicts global SH, reflectance and normals, and the fine network predicts their local residuals.

Similar to our previous work, ground truth labels of reflectance and lighting are not available for real data, we thus apply synthetic data for model pre-training and then fine-tune the model with real data in a self-supervised way. Compared to

the state-of-the-art methods only targeting normals or reflectance and shading, our method recovers all components and achieves consistently better results on three real dataset, IIW, SAW and NYUv2.

1.3 Summary

As the medium through which we capture images of the physical world, lighting is one of the key components to understand the physical world through images. Understand lighting from images is thus an important topics in computer vision which can help computers better understand the world. The biggest challenge of this task is the lack of ground truth lighting for real images. In this dissertation, we show the power of synthetic data in training CNNs to understand lighting from images by summarizing three research works: lighting estimation from faces, portrait relighting and intrinsic image decomposition from natural scenes. In the following chapters, we discuss these research works in detail.

Chapter 2: Less is More: Towards Compact CNNs

In this chapter, we study the capacity of current deep CNNs. Understanding the capacity of deep CNNs is important for reducing the amount of labeled data used to train a network, since it is the first step towards designing a network with less capacity and good generalization. As theoretically analyzing the capacity of a deep CNN is difficult, we do it in an empirical way by studying how many parameters of a deep CNN could be removed without affecting its performance. This work [36] is in collaboration with Jose M. Alvarez and Fatih Porikli.

2.1 Introduction

The last few years have witnessed the success of deep convolutional neural networks (CNN) in many computer vision applications. One important reason is the emergence of large annotated datasets and the development of high-performance computing hardware facilitating the training of high capacity CNNs with an exceptionally large number of parameters.

When defining the structure of a network, large networks are often preferred, and strong regularizers [17] tend to be applied to give the network as much discriminative power as possible. As such, the state-of-the-art CNNs nowadays contain

hundreds of millions of parameters [5]. Most of these parameters come from one or two layers that host a large number of neurons. Take AlexNet [4] as an example. The first and the second fully connected layers, which have 4096 neurons each, contain around 89% of all the parameters. Having to use such a large number of parameters leads to high memory requirements. Consequently, using deep CNNs obligates significant hardware resources, which hinders their practicality for mobile computing devices that have limited memory.

Several recent studies have focused on reducing the size of the network concluding that there is substantial redundancy in the number of parameters of a CNN. For instance, [37, 38] represent the filters in a CNN using a linear combination of basis filters. However, these methods can only apply to an already trained network. Other works have investigated directly reducing the number of parameters in a CNN by using sparse filters instead of its original full-size filters. To the best of our knowledge, existing deep learning toolboxes [39–41] do not support sparse data structures yet. Therefore, special structures need to be implemented. Moreover, FFT is shown to be very efficient to compute convolutions on GPUs [42]. However, a sparse matrix is usually no longer sparse in the Fourier domain, which limits the applicability (i.e., reduction of memory footprint) of these methods.

To address the shortcomings of the aforementioned works, we propose an efficient method to decimate the number of neurons. Our approach has four key advantages: (1) Neurons are assessed and reduced during the training phase. Therefore, no pre-trained network is required. (2) The reduction of parameters does not rely on the sparsity of neurons; thus, our method can be directly included in popular deep

Network	compression (%)		memory reduced[a]	Top-1 error (%)
	neurons[b]	parameters[c]		
LeNet	97.80	92.00	1.52 (MB)	0.63 (0.72)
CIFAR-10 quick	73.44	33.42	0.19 (MB)	25.25 (24.75)
AlexNet	73.73	65.42	152.14 (MB)	46.10 (45.57)
VGG-13	76.21	61.29	311.06 (MB)	39.26 (37.50)

[a]Supposing a single type is used to store the weight
[b]Results on number of neurons in the first fully connected layer
[c]Results on the number of parameters of the whole network

Table 2.1: Neuron reduction in the first fully connected layer, the total parameter compression, reduced memory, the top-1 validation error rate (red: error rate without sparse constraints).

learning toolboxes. (3) The number of filters in the Fourier domain is proportional to the number of neurons in the spatial domain. Therefore, our method can also reduce the number of parameters in the Fourier domain. (4) Reducing the number of neurons of a layer directly condenses the data dimensionality when the output of an internal layer is utilized as image features [7, 43].

Our method consists of imposing sparse constraints on the neurons of a CNN in the objective function during the training process. Furthermore, to minimize the extra computational cost of adding these constraints, we solve the sparse constrained optimization problem using the forward-backward splitting method [18, 19]. The main advantage of forward-backward splitting is to bypass the sparse constraint evaluations during the standard back-propagation step, making the implementation very practical. We also investigate the importance of rectified linear units in sparse constrained CNNs. Our experiments show that rectified linear units help to reduce the number of neurons leading to even more compact networks.

We conduct a comprehensive set of experiments to validate our method using

four well-known models (i.e., LeNet, CIFAR-10 quick, AlexNet and VGG) on three public datasets including ImageNet. Table 2.1 summarizes a representative set of our results when constraints are applied to the first fully connected layer of each model, which has the largest number of neurons and parameters. As shown, even with a large portion of neurons removed and, therefore, a significant reduction in the memory footprint of the original model, the resulting networks still perform as well as the original ones. These results are also convenient for deployment in mobile platforms where computational resources are relevant. For instance, using single float type (4 bytes) to store the network, the amount of memory saved for AlexNet in Table 2.1 is 152 MB.

To reiterate, the contribution of the proposed method in this chapter is three-fold. First, we remove neurons of a CNN during the training stage. Second, we analyze the importance of rectified linear units for sparse constraints. Finally, our experimental results on four well-known CNN architectures demonstrate a significant reduction in the number of neurons and the memory footprint of the testing phase without affecting the classification accuracy.

2.2 Related Work

Deep CNNs demand large memory and excessive computational times. This motivated studies to simplify the structure of CNNs. However, so far only one pioneering work has explored the redundancy in the number of neurons [17].

We organize the related work in three categories; network distillation, ap-

proximating parameters or neurons by memory efficient structures, and parameter pruning in large networks.

Network distillation: The work in [44] is among the first papers that tried to mimic a complicated model with a simple one. The key idea is to train a powerful ensemble model and use it to label a large amount of unlabeled data. Then a neural network is trained on these data so as to perform similarly to the ensemble model. Following the idea of [44], [45–47] present training of a simple neural network based on the soft output of a complicated one. Their results show that a much simpler network can imitate a complicated network. However, these methods require a two-step training process and the simple network must be trained after the complicated one.

Memory efficient structures: Most of the studies in this category are inspired by the work of Denil et al. [14] that demonstrated redundancy in the parameters of neural networks. It proposed to learn only 5% of the parameters and predicted the rest by choosing an appropriate dictionary. Inspired by this, [37,38,48] proposed to represent the filters in convolutional layers by a linear combination of basis filters. As a result, convolving with the original filters is equivalent to convolving with the base filters followed by a linear combination of the output of these convolutions. These methods focus on speeding up the testing phase of CNNs. In [49], the use of the canonical polyadic decomposition (CP-decomposition) to approximate the filter in each layer as a sequence of four convolutional layers with small kernels is investigated in order to speed up testing time when the network is run on a CPU. In [50] several schemes to quantize the parameters in CNNs are presented. All these

methods require a trained network and then fine tuning for the new structures. [51] applied the Tensor Train (TT) format to approximate the fully connected layers and train the TT format CNN from scratch. They can achieve a high compression rate with little loss of accuracy. However, it is not clear whether the TT format can be applied to convolutional layers.

Parameter pruning: A straightforward way to reduce the size of CNNs is directly removing some of the unimportant parameters. Back to 1990, LeCun et al. [52] introduced computing the second derivatives of the objective function with respect to the parameters as the saliency, and removing the parameters with low saliency value. Later, Hassibi and Stork [53] followed their work and proposed an optimal brain surgeon, which achieved better results. These two methods, nevertheless, require computation of the second derivatives, which is computationally costly for large-scale neural networks [54]. In [15], directly adding sparse constraints on the parameters was discussed. This method, different from ours, cannot reduce the number of neurons.

Another approach [16] combines the memory efficient structures and parameter pruning. First, unimportant parameters are removed, and then, after a fine-tuning process, the remaining parameters are quantized for further compression. This work is complementary to ours and can be used to further compress the network trained using our method.

Directly related to our proposed method is [17]. Given a pretrained CNN, it proposes a theoretically sound approach to combine similar neurons together. Yet, their method is mainly for pruning a trained network whereas our method

directly removes neurons during training. Although [17] is *data-free* when pruning, it requires a pretrained network, which is data dependent. Moreover, our results show that, without degrading the performance, we can remove more neurons than the suggested cut-off value reported in [17].

2.3 Sparse Constrained Convolutional Neural Networks

Notation: In the following discussion in this chapter, if not otherwise specified, $|\cdot|$ is the ℓ_1 norm, $||\mathbf{x}|| = \sqrt{\sum_i x_i^2}$ is the ℓ_2 norm for a vector (Frobenius norm for a matrix). We use $\mathbf{W}$ and $\mathbf{b}$ to denote all the parameters in filters and bias terms in a CNN, respectively. $\mathbf{w}_l$ and $\mathbf{b}_l$ represent the filter and bias parameters in the l-th layer. $\mathbf{w}_l$ is a tensor whose size is $w \times h \times m \times n$, where w and h are the width and the height of a 2D filter, m represents the number of channels for the input feature and n is the number of channels for the output feature. $\mathbf{b}_l$ is a n dimensional vector, i.e. each output feature of this layer has one bias term. $\mathbf{w}_{lj}$ represents a $w \times h \times m$ filter that creates the j-th channel of the output feature for layer l, b_{lj} is its corresponding bias term. $\mathbf{w}_{lj}$ and b_{lj} together form a neuron. We use $\hat{\mathbf{W}}$, $\hat{\mathbf{w}}_l$ and $\hat{\mathbf{w}}_{lj}$ to represent the augmented filters (they contain the corresponding bias term).

2.3.1 Training a Sparse Constrained CNN

Let $\{\mathbf{X}, \mathbf{Y}^*\}$ be the training samples and corresponding ground-truth label. Then, a CNN can be represented as a function $\mathbf{Y} = f(\hat{\mathbf{W}}, \mathbf{X})$, where $\mathbf{Y}$ is the output

of the network. $\hat{\mathbf{W}}$ is learned through minimizing an objective function:

$$\min_{\hat{\mathbf{W}}} \psi(f(\hat{\mathbf{W}}, \mathbf{X}), \mathbf{Y}^*). \tag{2.1}$$

We use $\psi(\hat{\mathbf{W}})$ to represent the objective function for simplicity. The objective function $\psi(\hat{\mathbf{W}})$ is usually defined as the average cross entropy of the ground truth labels with respect to the output of the network for each training image. Equation (2.1) is usually solved using a gradient descend based method such as back-propagation [3].

We add sparse constraints on the neurons of a CNN. Therefore, the optimization problem of (2.1) can be written as:

$$\min_{\hat{\mathbf{W}}} \psi(\hat{\mathbf{W}}) + g(\hat{\mathbf{W}}), \tag{2.2}$$

where $g(\hat{\mathbf{W}})$ represents the set of constraints added to $\hat{\mathbf{W}}$. Given this new optimization problem, the k-th iteration of a standard back-propagation can be defined as:

$$\hat{\mathbf{W}}^k = \hat{\mathbf{W}}^{k-1} - \tau \frac{\partial \psi(\hat{\mathbf{W}})}{\partial \hat{\mathbf{W}}}|_{\hat{\mathbf{W}}=\hat{\mathbf{W}}^{k-1}} - \tau \frac{\partial g(\hat{\mathbf{W}})}{\partial \hat{\mathbf{W}}}|_{\hat{\mathbf{W}}=\hat{\mathbf{W}}^{k-1}}, \tag{2.3}$$

where $\hat{\mathbf{W}}^k$ represents the parameters learned at k-th iteration and τ is the learning rate. Based on (2.3), a new term $\frac{\partial g(\hat{\mathbf{W}})}{\partial \hat{\mathbf{W}}}|_{\hat{\mathbf{W}}=\hat{\mathbf{W}}^{k-1}}$ must be added to the gradient of each constrained layer during back-propagation. In those cases where $g(\hat{\mathbf{W}})$ is non-differentiable at some points, sub-gradient methods of $g(\hat{\mathbf{W}})$ are usually needed. However, these methods have three main problems. First, iterates of the

sub-gradient at the points of non-differentiability hardly ever occur [18]. Second, sub-gradient methods usually cannot generate an accurate sparse solution [55]. Finally, sub-gradients of some sparse constraints are difficult to choose due to their complex form or they may not be unique [19]. To avoid these problems, and in particular with l_1 constrained optimization problems, [56] and [15] proposed to use a proximal mapping. Following their idea, we apply a proximal operator to our problem.

2.3.2 Forward-Backward Splitting

Our proposal to solve the problem (2.2) and therefore, train a constrained CNN, consists of using a forward-backward splitting algorithm [18, 19]. Forward-backward splitting provides a way to solve non-differentiable and constrained large-scale optimization problems of the generic form:

$$\min_{\mathbf{z}} f(\mathbf{z}) + h(\mathbf{z}), \tag{2.4}$$

where $\mathbf{z} \in \mathbb{R}^N$, $f(\mathbf{z})$ is differentiable and $h(\mathbf{z})$ is an arbitrary convex function [18, 19]. The algorithm consists of two stages: First, a forward gradient descent on $f(\mathbf{z})$. Then, a backward gradient step evaluating the proximal operator of $h(\mathbf{z})$, i.e. $\text{prox}_f(z) = \arg\min_u \left(h(u) + \lambda(u - z)^2\right)$, where λ is a weight to be chosen. Using this algorithm has two main advantages. First, it is usually easy to estimate the proximal operator of $h(\mathbf{z})$ or even have a closed form solution. Second, backward analysis has an important effect on the convergence of the method when $f(\mathbf{z})$ is

convex [19].

Though there is no guarantee about convergence when $f(\mathbf{z})$ is non-convex, forward-backward splitting method usually works quite well for non-convex optimization problems [19]. By treating $\psi(\hat{\mathbf{W}})$ in Equation (2.2) as $f(\mathbf{z})$, the forward gradient descent can be computed exactly as the standard back-propagation algorithm in training CNNs. As a result, using the forward-backward splitting method to solve sparse constrained CNNs has two steps in one iteration. Algorithm 1 shows how to apply this method to optimize Equation (2.2), where τ^k is the learning rate of the forward step at k-th iteration.

Algorithm 1: Forward-backward splitting for sparse constrained CNNs

---

1: **while** Not reaching maximum number of iterations **do**
2: One step back-propagation for $\psi(\hat{\mathbf{W}})$ to get $\hat{\mathbf{W}}^{k*}$
3: $\hat{\mathbf{W}}^k = \arg\min_{\hat{\mathbf{W}}} g(\hat{\mathbf{W}}) + \frac{1}{2\tau^k}||\hat{\mathbf{W}} - \hat{\mathbf{W}}^{k*}||^2$
4: **end while**

In practice, we define one step in line 2 of Algorithm 1 as one epoch instead of one iteration of the stochastic gradient descent algorithm. There are two main reasons for this. First, to minimize the computational training overhead of the algorithm as we need to estimate fewer proximal operators of $g(\hat{\mathbf{W}})$. Second, the gradient of $\psi(\hat{\mathbf{W}})$ at each iteration is an approximation to the exact gradient which is noisy [56]. Computing the gradient after a certain number of iterations makes the learned parameters more stable [55].

2.3.3 Sparse Constraints

Our goal is removing neurons $\hat{\mathbf{w}}_{lj}$, each of which is a tensor. To this end, we consider two sparse constraints for $g(\hat{\mathbf{W}})$ in Equation (2.2): tensor low rank constraints [21] and group sparsity [22].

Tensor Low Rank Constraints

Although the low-rank constraints for 2D matrices and their approximations have been extensively studied, as far as we know, there are few works considering the low-rank constraints for higher dimensional tensors. In [21], the authors proposed to minimize the average rank of different unfoldings of a tensor matrix. To relax the problem to convex, they proposed to approximate the average rank using the average of trace norms, which is called the tensor trace norm, for different unfoldings [21]. We use this formulation as our tensor low rank constraints.

The tensor trace norm of a neuron $\hat{\mathbf{w}}_{lj}$ is $||\hat{\mathbf{w}}_{lj}||_{tr} = \frac{1}{n}\sum_{i=1}^{n}||\hat{\mathbf{w}}_{lj(i)}||_{tr}$, where n is the order of tensor $\hat{\mathbf{w}}_{lj}$, and $\hat{\mathbf{w}}_{lj(i)}$ is the result of unfolding the tensor $\hat{\mathbf{w}}_{lj}$ along the i-th mode. Under this definition, function $g(\hat{\mathbf{W}})$ can be defined as:

$$g(\hat{\mathbf{W}}) = \lambda \sum_{(j,l)\in\Omega} \frac{1}{n}\sum_{i=1}^{n}||\hat{\mathbf{w}}_{lj(i)}||_{tr}, \tag{2.5}$$

where Ω is a set containing all the neurons to be constrained and λ is the weight for the sparse constraint.

As a result, the backward step in the forward-backward splitting is given by:

$$\begin{aligned}\hat{\mathbf{w}}_{lj}^{k} &= \arg\min_{\hat{\mathbf{w}}_{lj}} \frac{1}{n}\sum_{i=1}^{n}||\hat{\mathbf{w}}_{lj(i)}||_{tr} + \frac{1}{2\tau\lambda}||\hat{\mathbf{w}}_{lj} - \hat{\mathbf{w}}_{lj}^{k*}||^2 \\ &= \arg\min_{\hat{\mathbf{w}}_{lj}} \frac{1}{n}\sum_{i=1}^{n}||\hat{\mathbf{w}}_{lj(i)}||_{tr} + \frac{1}{2\tau\lambda n}\sum_{i=1}^{n}||\hat{\mathbf{w}}_{lj(i)} - \hat{\mathbf{w}}_{lj(i)}^{k*}||^2. \end{aligned} \tag{2.6}$$

This problem can be solved using the Low Rank Tensor Completion (LRTC) algorithm proposed in [21]. Suppose we have a 2D matrix X, let $X = U\Sigma V$ be the singular value decomposition of X, where Σ is a diagonal matrix of the singular values of X. Now let us suppose the i-th singular value is σ_i, then define $\Sigma_\tau = diag(\max(\sigma_i - \tau), 0)$. Then the shrinkage operator is defined as:

$$D_t(X) = U\Sigma_t V^T, \tag{2.7}$$

Algorithm 2 shows how to use LRTC to optimize Equation (2.6).

Algorithm 2: Backward step for tensor low rank constraints

1: Initialize $\hat{\mathbf{w}}_{lj}^{k} = \hat{\mathbf{w}}_{lj}^{k*}$
2: **while** not converged **do**
3: **for** i = 1 to n **do**
4: $M_i = D_{\tau\lambda}(\frac{1}{2}(\hat{\mathbf{w}}_{lj(i)}^{k} + \hat{\mathbf{w}}_{lj(i)}^{k*}))$
5: **end for**
6: $\hat{\mathbf{w}}_{lj}^{k} = \frac{1}{n}\sum_{i=1}^{n} M_i$
7: **end while**

Group Sparse Constraints

Group sparse constraints are defined as $l_{2,1}$ regularizer. Applying $l_{2,1}$ to our

objective function, we have:

$$g(\hat{\mathbf{W}}) = \lambda \sum_{(j,l)\in\Omega} ||\hat{\mathbf{w}}_{lj}||, \tag{2.8}$$

where λ and Ω are the same as in Equation (2.5). $||.||$ is defined in Section 2.3. According to [22], the backward step in the forward-backward splitting method at the k-th iteration now becomes:

$$\hat{\mathbf{w}}_{lj}^{k} = \max\{||\hat{\mathbf{w}}_{lj}^{k*}|| - \tau\lambda, 0\} \frac{\hat{\mathbf{w}}_{lj}^{k*}}{||\hat{\mathbf{w}}_{lj}^{k*}||}, \tag{2.9}$$

where τ is the learning rate and $\hat{\mathbf{w}}_{lj}^{k*}$ is the optimized neuron from the forward step in forward-backward splitting.

2.3.4 Importance of Rectified Linear Units in Sparse Constrained CNNs

Convolutional layers in a CNN are usually followed by a nonlinear activation function. Rectified Linear Units (ReLU) [4] have been heavily used in CNNs for computer vision tasks. Besides the advantages of ReLU discussed in [20, 57, 58], we show that $\hat{\mathbf{w}}_{lj} = \mathbf{0}$ is a local minimum in sparse constrained CNNs, of which the non-linear function is ReLU. This can explain our findings that ReLU can help in removing more neurons and inspires us to set momentum to 0 for neurons which reach their sparse local minimum during training as discussed in Section 2.4.1.

The ReLU function is defined as $ReLU(x) = max(0, x)$, which is non-differentiable

at 0. This creates difficulty in analyzing the local minimum of sparse constrained CNNs. Based on the observation that, in practical implementations, the gradient of the ReLU function at 0 is set to 0 [39–41]. We consider the following practical definition of ReLU:

$$ReLU(x) = \begin{cases} x & \text{if } x > \epsilon \\ 0 & \text{if } x \leq \epsilon. \end{cases} \tag{2.10}$$

Where ϵ is chosen such that for any real number x that a computer can represent, if $x > 0$, then $x > \epsilon$; if $x \leq 0$, then $x < \epsilon$. The non-differentiable point of the ReLU function is now at ϵ which will never appear in practice. Under this definition, the ReLU function is differentiable and continuous at point 0, the gradient of $ReLU(x)$ at 0 is now 0. As a result, this practical definition of ReLU is consistent with the implementations of a ReLU function in practice [39–41].

We now show that a particular neuron $\hat{\mathbf{w}}_{lj} = \mathbf{0}$ lies in a flat region of $\psi(\hat{\mathbf{w}}_{lj})$ under the above definition of ReLU. Take a particular neuron $\hat{\mathbf{w}}_{lj}$ as an example and suppose all other neurons are fixed. Also, suppose $\mathbf{x}$ is one of the inputs that will go through $\hat{\mathbf{w}}_{lj}$ in the l-th layer. The output for $\mathbf{x}$, assuming the convolution layer is followed by a ReLU function, is $z_{\hat{\mathbf{x}}} = ReLU(\hat{\mathbf{w}}_{lj}\hat{\mathbf{x}})$, where $\hat{\mathbf{x}}$ is $\mathbf{x}$ augmented by 1 to account for the bias term. Next, we show that $\hat{\mathbf{w}}_{lj} = \mathbf{0}$ is the local minimum of Equation (2.2) along the dimensions of $\hat{\mathbf{w}}_{lj}$, referred to as a local minimum of a CNN for simplicity.

Given any $\mathbf{x} \in \Omega$, define $\varphi(\Delta\hat{\mathbf{w}}_{lj}) = \Delta\hat{\mathbf{w}}_{lj}\hat{\mathbf{x}}$ as a function of $\Delta\hat{\mathbf{w}}_{lj}$. It is easy to

see that $\varphi(\Delta\hat{\mathbf{w}}_{lj})$ is a continuous function. As a result, we can always find a $\delta_{\mathbf{x}}$ such that $|\varphi(\Delta\hat{\mathbf{w}}_{lj}) - \varphi(\mathbf{0})| < \epsilon$, $\forall ||\Delta\hat{\mathbf{w}}_{lj} - \mathbf{0}|| < \delta_{\mathbf{x}}$. According to Equation (2.10), for those $\hat{\mathbf{w}}_{lj}$, $ReLU(\Delta\hat{\mathbf{w}}_{lj}\hat{\mathbf{x}}) = 0$. Denoting δ as the smallest one among all $\delta_{\mathbf{x}}$, we know that for any $||\Delta\hat{\mathbf{w}}_{lj} - \mathbf{0}|| < \delta$, $ReLU(\Delta\hat{\mathbf{w}}_{lj}\mathbf{x}) = 0$, $\forall \mathbf{x} \in \Omega$. As all other neurons are fixed, we know that for any $||\Delta\hat{\mathbf{w}}_{lj} - \mathbf{0}|| < \delta$, $\psi(\Delta\hat{\mathbf{w}}_{lj}) = c$, where ψ is the first part in Equation (2.2) and c is a scalar. Since $g(\hat{\mathbf{W}})$ contains a sparse constraint on $\hat{\mathbf{w}}_{lj}$, $g(\hat{\mathbf{w}}_{lj} = \mathbf{0}) \leq g(\Delta\hat{\mathbf{w}}_{lj})$ and the equality holds if and only if $\Delta\hat{\mathbf{w}}_{lj} = \mathbf{0}$. As a result, for any $||\Delta\hat{\mathbf{w}}_{lj} - \mathbf{0}|| < \delta$, $\psi(\hat{\mathbf{w}}_{li} = \mathbf{0}) + g(\hat{\mathbf{w}}_{li} = \mathbf{0}) \leq \phi(\Delta\hat{\mathbf{w}}_{li}) + g(\Delta\hat{\mathbf{w}}_{li})$, i.e. $\hat{\mathbf{w}}_{lj} = \mathbf{0}$ is the local minimum of the objective function $\psi(\hat{\mathbf{w}}_{lj}) + g(\hat{\mathbf{w}}_{lj})$.

The fact that $\hat{\mathbf{w}}_{lj} = \mathbf{0}$ is a local minimum for sparse constrained CNNs using ReLU as a nonlinear activation function can explain the improvement in the number of neurons removed as discussed in Section 2.4.1. Importantly, we find that using momentum during the optimization may push a zero neuron away from being $\mathbf{0}$ since momentum memorizes the gradient of this neuron in previous steps. As a consequence, using momentum would lead to more non-zero neurons without performance improvement. In practice, once a neuron reaches $\mathbf{0}$, we set its momentum to zero, forcing the neuron to maintain its value. As we will demonstrate in Section 2.4.1, this results in more zero neurons without affecting the performance.

2.4 Experiments

We test our method on four well-known convolutional neural networks on three well-known datasets: LeNet on MNIST [3], CIFAR10-quick [23] on CIFAR-10 [59]

and AlexNet [4] and VGG [5] on ILSVRC-2012 [60]. We use LeNet and CIFAR10-quick provided by Matconvnet [41] and AlexNet and VGG provided by [61] to carry out all our experiments.

All the structures of the networks are provided by Matconvnet [41] or [61], the only change we made is to add a ReLU function after each convolutional layer in LeNet, which we will discuss in detail later. For all our experiments, data augmentation and pre-processing strategies are provided by [41] and [61].

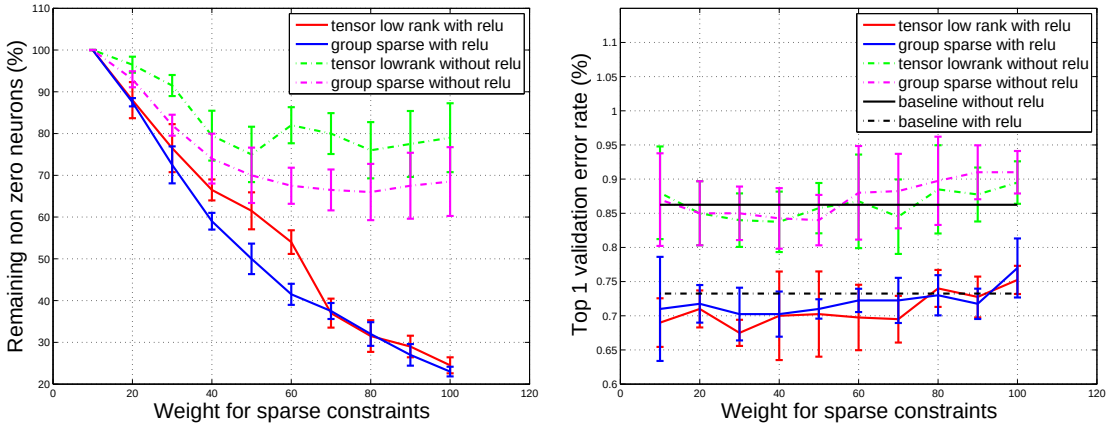
2.4.1 LeNet on MNIST

MNIST is a well-known dataset for machine learning and computer vision. It contains $60,000$ handwritten digits for training and $10,000$ for testing. All these digits are images of 28×28 with a single channel. The mean of each set of data will be subtracted, as suggested in [41].

The average top 1 validation error rate of LeNet on MNIST adding a ReLU layer is 0.73%. As training a LeNet on MNIST is fast, we use this experiment as a sanity check. Results are computed as the average over four runs of each experiment using different random seeds.

ReLU helps removing more neurons:

The LeNet structure provided by Matconvnet has two convolutional layers and two fully connected layers. The two convolutional layers are each followed by a max pooling layer. Under this structure, the sparse solution may not be a local minimum. Based on the discussion in Section 2.3.4, we add a ReLU layer after each



(a) (b)

Figure 2.1: (a) Percentage of nonzero neurons on the second convolutional layer for LeNet under different weights for sparse constraints with and without adding a ReLU layer. (b) Corresponding top 1 validation error rate. Baseline in (b) shows the top 1 validation error rate without adding any sparse constraints. Error bars represent the standard deviation over the four experiments.

of these two convolutional layers so that $\hat{\mathbf{w}}_{lj} = \mathbf{0}$ is a local minimum. We compare these two structures by using different weights for sparse constraints added to the second convolutional layer.

As shown in Figure 2.1, adding ReLU improves the performance of LeNet regardless of whether we add sparse constraints or not. Comparing these two structures, adding a ReLU layer always leads to more zero neurons compared with the original structure. What is more interesting is that, with a ReLU layer, the top 1 validation error rate of LeNet with sparse constraints is more close to, if not better than, the performance without sparse constraints. This may be explained by the fact the $\hat{\mathbf{w}}_{lj} = \mathbf{0}$ is a local minimum of the structure with ReLU and this local minimum is usually a good one. In the following discussion, when LeNet is mentioned, we mean LeNet with a ReLU function after each of its convolutional layers.

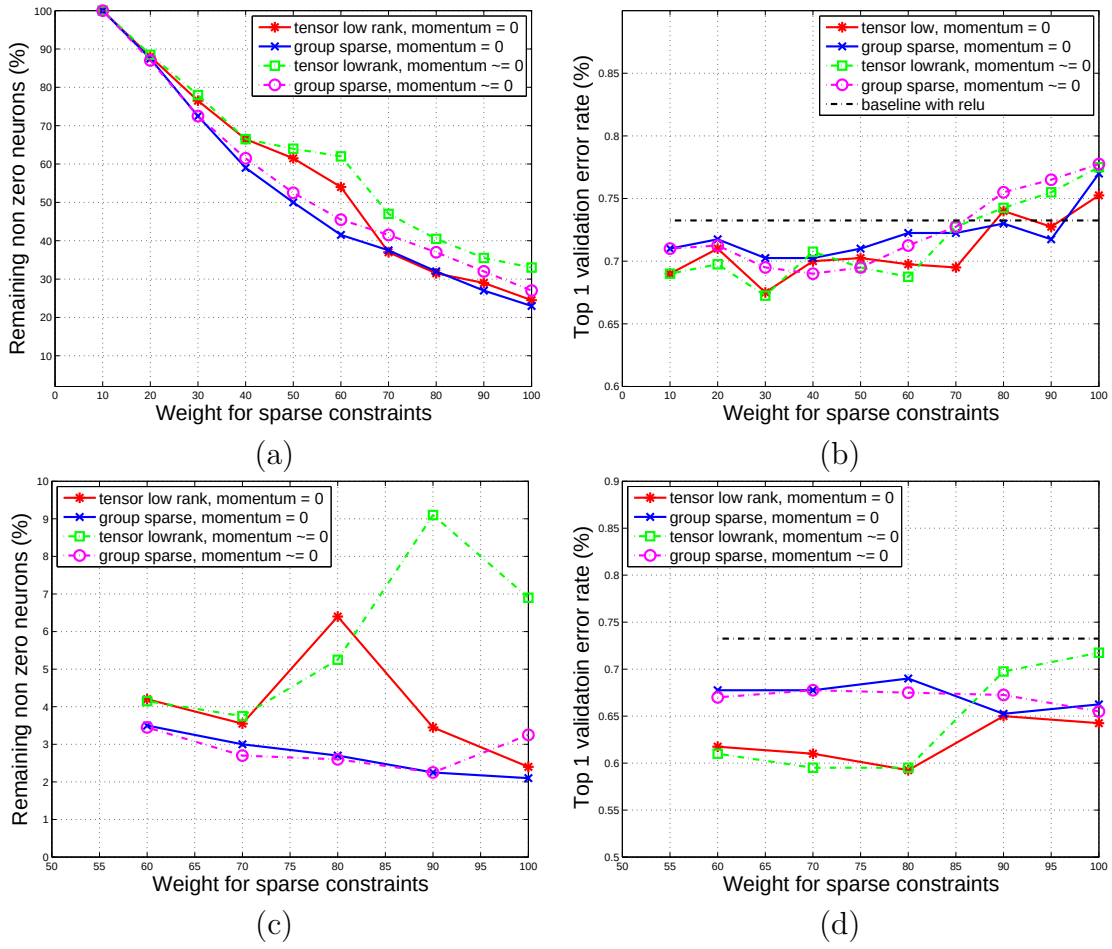


Figure 2.2: Comparison of the proposed results with setting momentum to be zero. (a) and (c) show the percentage of non-zero neurons for second convolutional layer and first fully connected layer under different weights for sparse constraints respectively. (b) and (d) show their corresponding top 1 validation error rate.

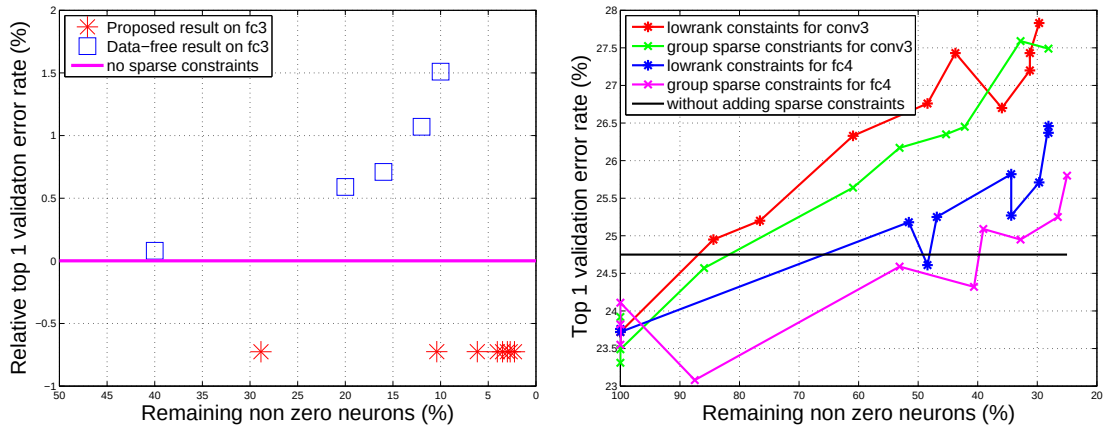
Momentum for sparse local minimum:

Momentum plays a significant role in training CNNs. It can be treated as a memory of the gradient computed in previous iterations and has been proved to accelerate the convergence of SGD. However, this memory of gradient effect may push a neuron away from its sparse local minimum, leading to results with more non-zero neurons with no improvement in performance. To avoid this problem we can directly set the momentum of the neurons to be zero for those sparse local minima that are reached.

In Figure 2.2, we compare the number of non-zero neurons with and without setting the momentum to be zero for LeNet on the MNIST dataset. We add the sparse constraints on the second convolutional layer and the first fully connected layer since they have most of the filters. As shown in Figure 2.2, the performance does not drop when momentum is set to be zero for local sparse minima. Additionally, we sporadically achieve better performance. Moreover, for the first fully connected layer, setting momentum to be zero leads to results with more zero neurons under large sparse weights.[1]

From Figure 2.1 and Figure 2.2, we find that the top-1 error rates of using tensor low-rank constraints are a little better than using group sparse constraints. Group sparse constraints, however, can produce more zero neurons, which can result in more compact networks. These differences are very small, so we conclude that both of these methods can help removing neurons without hurting the perfor-

[1]For a clear comparison, we only show results under large weights for the first fully connected layer in the figure.



(a) (b)

Figure 2.3: a) **MNIST** Comparison between single layer results of the proposed method and the data-free method presented in [17] using LeNet on MNIST. b) **CIFAR-10** Top 1 validation error rate versus percentage of remaining non-zero neurons on conv3 and fc4 using CIFAR10-quick.

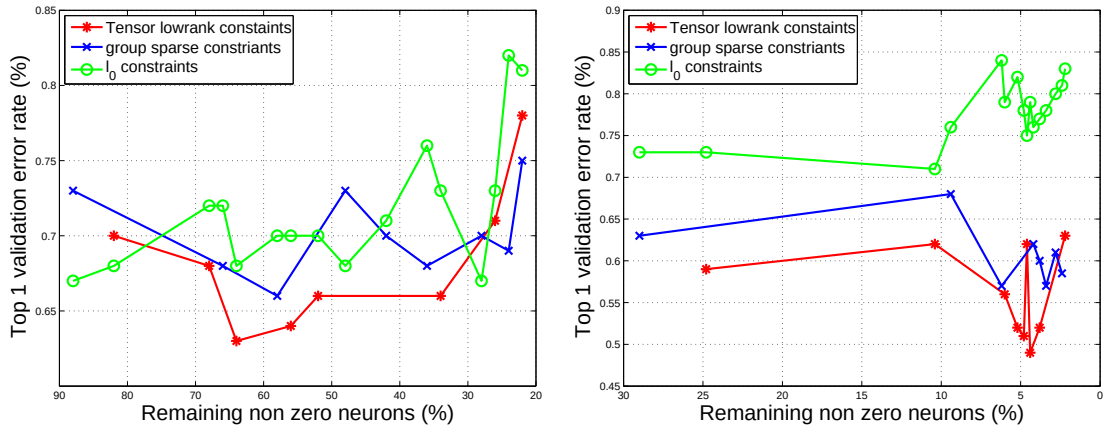
mance of the network. In the following discussion, only results from group sparse constraints are shown.

Comparison with l_0 constraints on LeNet

Both [55] and [15] proposed to add l_0 constraints to the parameters of a neural network. They show that this performs quite well in the sense of zeroing out parameters. We apply this idea to remove neurons on LeNet and compare it with adding tensor low rank and group sparse constraints. We use exactly the same setup as in the paper and test the idea using the MNIST data set [3].

Following the idea of [55] and [15], given a parameter t, neurons whose magnitudes are among the largest t will be kept, other neurons will be set to zero during the training process.

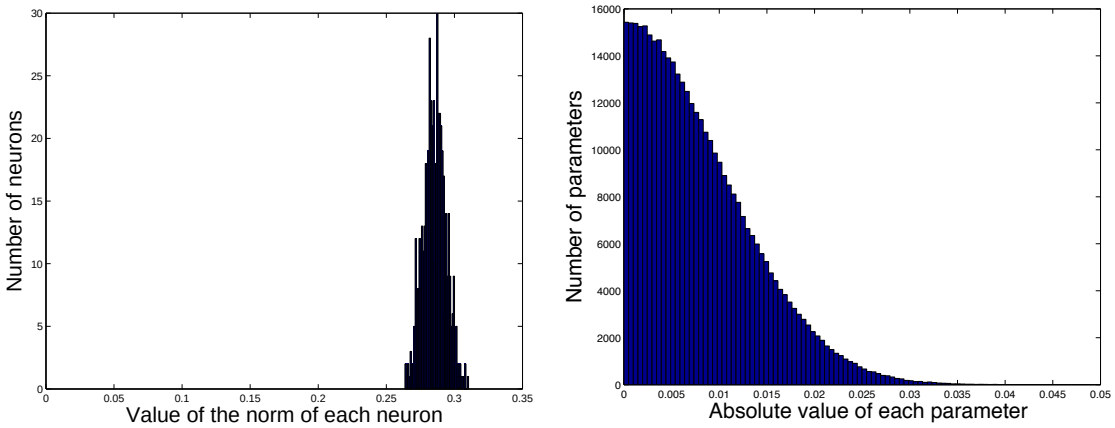
Figure 2.4 compares results of l_0 constraints with the proposed tensor low rank and group sparsity. For the experiment, t is chosen based on the number of non-



(a) (b)

Figure 2.4: (a) and (b) show the top 1 validation error rate versus the percentage of non-zero neurons for the second convolutional layer and first fully connected layer for LeNet respectively. We compare results of low rank constraints, group sparse constraints and directly using l_0 constraints.

zero neurons we got by using proposed sparse constraints under different weights. Generally speaking, the proposed constraints perform better than the l_0 constraints for training the network. This is more obvious when we try to get more zero filters for the first fully connected layer. One reason is that, without using any sparse constraints, the norm of the neurons learned are distributed more compactly and far from 0. As a result, directly setting some of the neurons to be zero is risky. The distribution of absolute values of the parameters, even without adding any sparse constraints, are more biased to 0, which explains why l_0 constraint works well in [55] and [15]. This can be seen form Figure 2.5, which shows the distribution of the norm of the neurons and absolute values of parameters.



(a) (b)

Figure 2.5: (a) shows the distribution of norm of the neurons in first fully connected layer for LeNet. (b) shows the distribution of the absolute value of the parameters in the first fully connected layer for LeNet. The network is trained with one epoch without adding sparse constraints.

Compression for LeNet:

Figure 2.3 (a) shows the comparison of the proposed method with [17]. Since [17] only adds sparse constraints on the first fully connected layer of LeNet (fc3), we only show our results using group sparse constraints on fc3. The compression rate and top 1 validation error are averaged over four results with different random seeds. Since the two networks may be trained differently, we show the relative top 1 validation error rate, which is defined as the top 1 validation error rate of the proposed method minus that without sparse constraints. A smaller value means better performance. As shown in this figure, our method outperforms the one proposed in [17] in both compression and accuracy. Please note that [17] predicts the cut-off number of neurons for fc3 to be 440 and a drop of performance of 1.07%. The proposed method, on the contrary, can remove 489 neurons while maintaining the performance of the original one.

To improve the compression rate, we add group sparse constraints to the layers containing most of the parameters in LeNet: the second convolutional layer (conv2) and the first fully connected layer (fc3). We compare two strategies. First, adding sparse constraints in a layer by layer fashion and second, jointly constraining both layers. Empirically we found that the first strategy performs better. Thus, we first add sparse constraints on fc3 and, after the number of zero-neurons in this layer is stable, we add sparse constraints on conv2. We report the results of using group sparse constraints. The weight used for fc3 is set to 100, and the weights for conv2 are set to $60, 80, 100$. Table 2.3 summarizes the average results over the four runs of the experiment. As shown, our method not only reduces significantly the number of

τ			Non-zero Neurons			Number of	Top-1 error (%)
conv1	conv2	fc3	conv1	conv2	fc3	Parameters	
100	80	90	7	23	20	11820	0.72
100	120	80	7	13	21	7079	0.76
120	120	90	6	14	20	6980	0.81

Table 2.2: Results of adding group sparse constraints on three layers.

neurons, which leads to significant reduction in the number of parameters in these two layers but also compresses the total number of parameters of the network for more than 90%, leading to a memory footprint reduction larger than 1 MB.

We further try to add group sparse constraints on all three layers of LeNet to check whether our method can work on more layers. To introduce more redundancy on conv1 and conv2, we initialize the number of non-zero neurons for conv1 and conv2 to be 100. Similar to adding sparse constraints on two layers, we add sparse constraints layer by layer. We show some of our results in Table 2.2. To compare, the best compression result of adding sparse constraints on conv2 and fc3 leads to a model with 13062 parameters (third row of LeNet in Table 2.3). We find that by adding sparse constraints on three layers, a more compact network can be achieved though we initialize the network with more neurons.

2.4.2 CIFAR-10 quick on CIFAR-10

CIFAR-10 [59] is a database consisting of $50,000$ training and $10,000$ testing RGB images with a resolution of 32×32 pixels split into 10 classes. As suggested in [41], data is standardized using zero-mean unit length normalization followed by a whitening process. Furthermore, we use random flips as data augmentation. For a

fair comparison, we train the original network, CIFAR10-quick, without any sparse constraints using the same training set-up and achieve a top 1 validation error rate of 24.75%.

Since the third convolutional layer (conv3) and the first fully connected layer (fc4) contain most of the parameters, we add sparse constraints on these two layers independently. Figure 2.3 (b) shows the top 1 validation error rate versus the percentage of remaining non-zero neurons. As shown, 20% and 70% of neurons for conv3 and fc4 can be removed without a noticeable drop in performance.

To obtain the best compression results, we jointly constrained conv3 and fc4 as we did with LeNet. To this end, we first add the constraints on fc4, and then, we include the same ones on conv3. We run the experiment for more epochs compared to the default values in Matconvnet [41]. For a fair comparison, we train the baseline (i.e., CIFAR-10 quick without sparse constraints) for the same number of epochs. As a result, we obtain a top 1 validation error of 22.33%. We use this result to compute relative top 1 validation error rate. The weight of group sparse constraints for fc4 is fixed to 280 while the three different weights for conv3 are 220, 240, 280. A summary of results is listed in the second part of Table 2.3. Through this experiment, it can be seen that even for this simple network, we can remove a large number of neurons, which leads to a great compression in the number of parameters. Considering that there are only 64 neurons in each of these two layers, the compression results are significant.

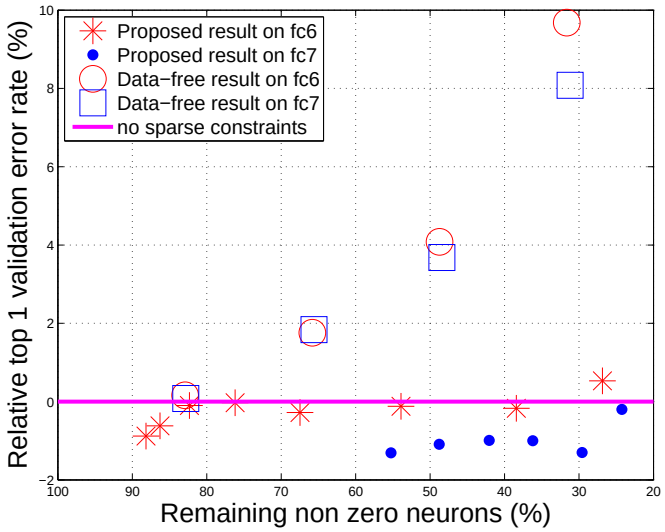


Figure 2.6: **ImageNet.** Comparison between single layer results on fc6 and fc7 and the data-free method in [17] using AlexNet on ImageNet. Data-free results are from [17].

2.4.3 AlexNet and VGG on ImageNet

ImageNet [62] is a dataset with over 15 million labeled images split into 22,000 categories. AlexNet [4] was proposed to be trained on ILSVRC-2012 [60] which is a subset of ImageNet with 1.2 million training images and 50,000 validation images. We use the implementation of AlexNet and VGG-13 provided by [61] in order to test ImageNet on our cluster. Random flipping is applied to augment the data. Quantitative results are reported using a single crop in the center of the image. For comparison, we consider the network trained without adding any sparse constraints. The top 1 validation error for this baseline is 45.57% and 37.50% for AlexNet and VGG-13 respectively. For the rest of experiments we only report results using group sparse constraints.

Figure 2.6 shows the top 1 validation error rate versus the percentage of non-zero neurons for AlexNet. Group sparse constraints are added on the first and

second fully connected layers (fc6 and fc7) independently. Results from [17] are copied from their paper and shown in this figure for comparison. We show the relative top 1 validation error rates because the training of two methods may be different. This figure clearly shows that compared with [17], the proposed method can remove a large number of neurons without decreasing the performance. For instance, the best compression results of the proposed method can eliminate 76.76% of all the parameters of AlexNet with a negligible drop in performance (0.57% in top 1 validation error). The best performance model in [17], on the other hand, can only remove 34.89% of the parameters with top 1 validation error rate decreased by 2.24%. A representative set of compression results obtained using sparse constraints on two layers is shown in the third part of Table 2.3.

We test the proposed method on VGG-13 on the first fully connected layer as it contains most of the parameters of the network. Table 2.4 summarizes the outcomes of the experiment for different group sparsity weights. As shown, for this state-of-the-art network structure, our method can eliminate nearly half of the parameters and significantly reduce the memory footprint at the expense of a slight drop in performance.

2.5 Summary

We proposed an algorithm to significantly reduce of the number of neurons in a convolutional neural network by adding sparse constraints during the training step. The forward-backward splitting method is applied to solve the sparse constrained

	τ		Neurons pruned(%)		Top-1 error (%)		parameter	memory
	conv2	fc3	conv2	fc3	absolute	relative	reduction (%)	reduced (MB)
LeNet	60	100	45.5	97.75	0.73	0.00	95.35	1.57
	80	100	56.5	97.75	0.77	0.04	96.31	1.58
	100	100	63.0	97.75	0.76	0.03	**96.79**	**1.59**
	τ		Neurons pruned (%)		Top 1 error (%)		parameters	memory
	conv3	fc4	conv3	fc4	absolute	relative	reduction(%)	reduced (KB)
cifar10-quick	220	280	31.25	70.31	22.21	-0.12	47.17	268.24
	240	280	46.88	71.86	22.73	0.4	**55.15**	**313.62**
	280	280	54.69	70.31	23.78	1.45	58.56	333.01
	τ		Neurons pruned (%)		Top 1 error (%)		parameters	memory
	fc6	fc7	fc6	fc7	absolute	relative	reduction (%)	reduced(MB)
AlexNet	40	35	48.46	56.49	44.58	-0.98	55.15	128.26
	45	30	77.05	60.21	46.14	0.57	**76.76**	**178.52**
	45	35	73.39	65.80	45.88	0.31	74.88	174.14

Table 2.3: Results of adding group sparse constraints on two layers. The best compression results within 1% decrease in top 1 error rate is shown in bold.

layer	τ	compression %		memory	top 1 error (%)	
		neurons	parameters	reduced (MB)	absolute	relative
fc1	5	39.04	35.08	178.02	38.30	0.80
fc1	10	49.27	44.28	224.67	38.54	1.04
fc1	20	76.21	61.30	311.06	39.26	1.76

Table 2.4: Some compression results of proposed method on fc1 for vgg-B. Neuron: compression of neurons in the fc1. Parameter: compression of total parameters.

problem. We also analyze the benefits of using rectified linear units as the non-linear activation function to remove a larger number of neurons.

Experiments using four popular CNNs including AlexNet and VGG-13 demonstrate the capability of the proposed method to reduce the number of neurons, and therefore, the number of parameters and memory footprint, with a negligible loss in performance.

Chapter 3: Label Denoising Adversarial Networks (LDAN) for Inverse Lighting of Faces

Obtaining ground truth labels for real data is sometimes not just difficult, but impossible. In this situation, synthetic data can be very helpful in training deep CNNs, since they are easy to generate and ground truth labels are easy to get. In this chapter, we study one of these problems: inverse lighting from a single face image. This work [13] is in collaboration with Jin Sun, Yaser Yacoob and David W Jacobs

3.1 Introduction

Estimating lighting sources from an image is a fundamental problem in computer vision. In general, this is a particularly difficult task when the scene has unknown shape and reflectance properties. On the other hand, estimating the lighting of a human face, one of the most popular and well studied objects, is easier due to its approximately known geometry and near Lambertian reflectance. Lighting estimation can be used in applications such as image editing, 3D structure estimation, and image forgery detection. This chapter focuses on estimating lighting from a single face image. We consider the most common face image type: near frontal

pose. The same idea can be applied to face images with other poses.

There exist many approaches for lighting estimation from a single face image [31, 63–65], however they are not learning-based and rely on complicated optimization during testing, making the process inefficient. Moreover, the performance of these methods (e.g., [31]) depends on the resolution of face images, and cannot give accurate predictions for low resolution images.

Witnessing the dominant success of neural network models in other computer vision problems such as image classification, we are interested in a supervised learning approach that directly regresses lighting parameters from a single face image. Given an input face image, the approach outputs low dimensional Spherical Harmonics (SH) coefficients [34,35] of its environment lighting condition. This is a very difficult problem, especially due to the scarcity of accurate ground truth lighting labels for real face images in the wild. In fact, building a dataset with realistic images and ground truth lighting parameters is extremely hard and currently there exists no such dataset.

Lacking ground truth labels, we propose to use synthetic data to help train a deep CNN for real data. However, the network trained using synthetic data cannot generalize well to real data due to the domain gap between synthetic and real data. As a result, we also include real data to train a deep CNN together with the synthetic data. We apply an existing method [31] to estimate lighting parameters of real face images. However, these lighting parameters are not the real "ground truth" as they contain unknown noise. Synthetic face images, on the other hand, have noise-free ground truth lighting labels. In this work, we show that this synthetic data with

accurate labels can help train a deep CNN to regress lighting of real face images: "denoising" the unreliable labels.

The proposed method is based on two assumptions: (1) A deep CNN trained with synthetic data is accurate, i.e., it is not affected by any noise; (2) Ground truth labels for real data are noisy, but still contain useful information. We design the lighting regression deep CNN, which consists of two sub-networks: a feature net that extracts lighting related features and a lighting net that takes these features as input and predicts the Spherical Harmonics parameters. Based on the first assumption, the lighting net trained with synthetic data is accurate. However, this lighting net expects lighting related features for synthetic data as input. To make it work for real data, the lighting related features for real data should be mapped to the same space. For that purpose, we utilize the idea of Generative Adversarial Networks (GAN) [32]. Specifically, a discriminator is trained to distinguish between lighting related features from synthetic data and real data, while the feature net (instead of a generator in the standard GAN) is trained to fool the discriminator. The discriminator and our feature net play a minmax two player game, with the objective of pulling the distribution of lighting related features of real data towards that of the synthetic data. Under the second assumption, we have an additional objective of reducing regression loss between predicted lightings and ground truth labels. Moreover, we design the network to take 64×64 RGB face images so that it will work for low resolution face images.

Figure 1.1 illustrates the proposed LDAN model. It consists of two steps during training: (1) Train with synthetic data; (2) Fix the feature net for synthetic

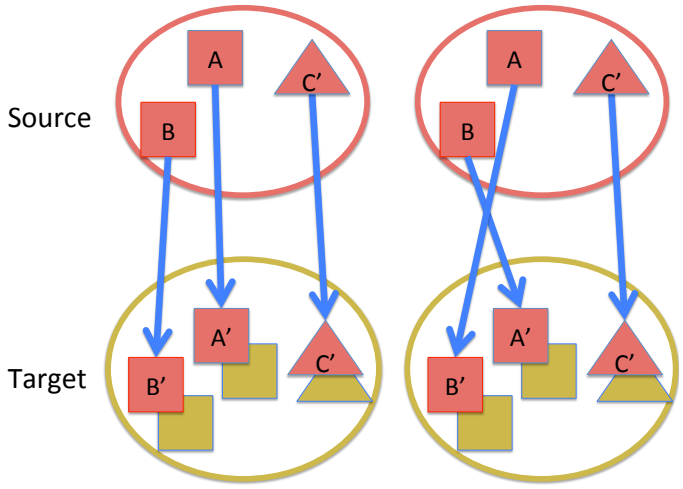


Figure 3.1: Two different functions that map data from the source domain to target domain with similar adversarial loss. With additional regression loss, our model is encouraged to learn a better behaved mapping function.

data and the lighting net, train another feature net for real data with adversarial loss and regression loss. Eric et al. [66] proposed similar ideas, applying adversarial loss to map the distribution of features from the source domain to the target domain. However, they only use adversarial loss. We argue that such a mapping can be unpredictably arbitrary. As illustrated by Figure 3.1, both mapping A to A', B to B' and mapping A to B' and B to A' make the source and target data have similar distributions. This may be correct for classification tasks if A and B belong to the same class. However, for regression, mapping A to B' takes the mapped feature far away from where it should be. As a result, using the regression loss for real data is critical in our regression problem: it regularizes the domain mapping function to have reasonable behavior. At the same time, the noise in real data labels are suppressed by training with the adversarial loss.

Since real ground truth labels for SH do not exist, we propose to use a clas-

sification based method to evaluate the consistency of estimated SH. However, this is still an indirect approach. To further evaluate the effectiveness of the proposed method, and show its potential to other applications, we apply it to an object key point regression problem where the ground truth labels are available. Similar to lighting regression from faces, we apply an existing method [27] to get the noisy ground truth and use synthetic data to help train an object key point regression network. Evaluated using the real ground truth, we demonstrate that LDAN works better than directly training a network with these noisy ground truth labels.

The main contributions of our work are: 1) We propose a lighting regression network for face images; 2) We propose a novel method, LDAN, to utilize accurate synthetic image lighting labels in training real face images with noisy labels; 3) The proposed method: increases the accuracy by 9% compared to [31] on quantitative evaluation, is robust to low resolution images, and is thousands of times faster.

3.2 Related Work

Lighting Estimation from A Single Face Image. Estimating lighting conditions from a single face image is an interesting but difficult problem. Blanz and Vetter [67] proposed to estimate the ambient and directional light as a byproduct of fitting 3D Morphable Models (3DMM) to a single face image. Since then, several 3DMM based methods were proposed [63–65, 68, 69]. The performance of these methods rely on a good 3DMM of faces. However, existing 3DMMs are usually built with face images taken in a controlled environment, so their expressive power

(especially the texture model) for faces in the wild is limited [70]. Barron and Malik proposed an optimization based method for estimating shape, albedo and lighting for general objects [31]. To solve such an underconstrained problem, their method heavily relies on prior knowledge about shape, albedo and lighting of general objects. Though they achieved promising results, their method is slow and may fail to give reasonable results in some cases due to the non-convexity of the objective function. [71] proposed to use deep learning to disentangle representations about pose, lighting and identity of a face image. The authors only show the effectiveness of their method on synthetic images, its performance on real face images is unclear. Recently, there is a trend to disentangle real faces using deep CNNs [72–74]. These methods, however, mainly focus on evaluating their performance on shape and albedo estimation. It is not clear whether the lighting estimated by these methods are accurate.

Learning with Noisy Labels. Learning with noisy labels has attracted the interest of researchers for a long time. [75] gives a comprehensive introduction to this problem. With the development of deep learning, many research studies have now focused on how to train deep neural networks with noisy labels [76–81]. [76,77,80,81] assume the probability of a noisy label only depends on the noise-free label but not on the input data, and try to model the conditional probability explicitly. [79] models the type of noise as a hidden variable and proposes a novel probabilistic model to infer the true labels. [78] proposes to use CNNs pre-trained with noise-free data to help select data with noisy labels in order to better handle the noise. All the above mentioned methods focus on classification problems and a considerable portion of the data are assumed to have noise-free labels. However, estimating lighting from

face images is a regression problem, and the translation probability from noise-free label to noisy label is much more difficult to model. Moreover, the labels of our real data are noisy. As a result, we are dealing with a much harder problem than the methods mentioned above.

GAN for Domain Adaption. Since Goodfellow et al. [32] first proposed Generative Adversarial Networks, several works have been using this idea for unsupervised Domain Adaption [66, 82–84]. All these methods solve a problem in which the labels in the target domain are not enough to train a deep neural network. However, the problem we try to solve is intrinsically different from theirs in that the labels in the target domain are sufficient, but all these labels are noisy. Moreover, all these methods apply domain adaption to classification tasks where adversarial loss is enough to achieve a good performance. On the contrary, adversarial loss alone cannot work in our regression task. Though adversarial loss could map the distribution of data in the target domain to that of the source domain, for a single point in the target domain, the mapping is arbitrary which is problematic as every data point has its unique label in a regression task.

3.3 Proposed Method

3.3.1 Spherical Harmonics

[34, 35] have shown that for convex objects with Lambertian reflectance and distant light sources, the lighting of the environment can be well estimated by 9 (gray scale) or 27 (color) dimensions of Spherical Harmonics (SH). In this chapter,

we use SH as the lighting representation as it has been widely used to represent the environmental lighting in face related applications as suggested in [31,65,69,85–87].

All dimensions of SH can be fully recovered from an image if the pixels are equally distributed over a sphere. However, the pixels of a face image, loosely speaking, are distributed over a hemisphere. The SH that can be recovered from a face image, as discussed in [88], lie in a lower dimensional subspace, and the SH for faces under different poses lie in different subspaces. As a result, we consider regressing the SH in a lower dimensional subspace instead of the original 27 dimensional SH and focus on near frontal faces since most face images are taken under this pose.

Taking the red color channel as an example, we now show how to get the lower dimensional subspace of SH for near frontal faces. Let $\mathbf{I}_r$ be a column vector: each element represents one pixel value of a face image for the red channel, then $\mathbf{I}_r = \Lambda_r Y \mathbf{l}_r$. Λ_r is a $n \times n$ diagonal matrix, each element of which is the albedo of the corresponding pixel, $\mathbf{l}_r$ is a 9 dimensional SH parameters vector, Y is a $n \times 9$ matrix and n is the number of pixels in the image. Each column of Y corresponds to one SH base image whose elements are determined by the normal of the corresponding pixel (see [34]). By applying SVD on Y, we get $Y = UDV^T$, then $\mathbf{I}_r = \Lambda_r UDV^T \mathbf{l}_r$. V is a 9×9 matrix that spans the entire 9 dimensions of SH. We use synthetic data to get V since we know the ground truth normal of every pixel and thus Y is known. We then only keep the first 6 columns of V, denoted as V_6, corresponding to the largest 6 singular values since they capture 99% energy of the singular values. With V_6, we project all the SH to their 18 dimensional subspace throughout the experiments.

3.3.2 Label Denoising Adversarial Network

Training a regression deep CNN needs a lot of data with ground truth labels. However, getting the ground truth lighting parameters from a realistic face image is extremely difficult. It usually needs a mirror ball or panorama camera which is carefully set up to record an environment map relative to the position of the face. Instead, we adapted [31] to predict lighting parameters from a large number of face images. These parameters are then projected to a lower dimensional subspace using V_6 discussed above. We use these projected lighting parameters as noisy ground truth labels and denote them as $\hat{\mathbf{y}}_r$. Together with real face images $\mathbf{r}$, ($\mathbf{r}$, $\hat{\mathbf{y}}_r$) will be used as (data, label) pairs to train a deep CNN to regress lighting parameters. Because these labels are noisy, directly training a deep CNN with these data cannot give the best performance.

We propose to use synthetic face images, whose ground truth lighting parameters are known, to help train a deep CNN. The proposed deep CNN has two sub-networks: a feature network that is used to extract lighting related features and a lighting network that takes lighting related features as input and predicts SH for the face images. For synthetic data $\mathbf{s}$, we denote its feature network as $\mathcal{S}$ and its lighting network as $\mathcal{L}$. Then the predicted SH is represented as $\mathbf{y}_s = \mathcal{L}(\mathcal{S}(\mathbf{s}))$. Since $\mathcal{S}$ and $\mathcal{L}$ are trained using synthetic data whose ground truth labels are known, they are accurate. Feature network $\mathcal{R}$ and Lighting network $\mathcal{L}_r$ for real data, on the other hand, will both be affected by the noisy labels if directly trained using the noisy ground truth of real data. To alleviate the effect of noisy labels, we propose to

use $\mathcal{L}$ as the lighting net for real data, i.e., $\mathcal{L}_r = \mathcal{L}$, since it is not affected by noise. However, since $\mathcal{L}$ is trained using synthetic data, it only works if the input is from the space of lighting related features of synthetic data. As a result, $\mathcal{R}$ needs to be trained such that the lighting related features for real data will be mapped into the same space as synthetic data.

Given a set of synthetic images $\mathbf{s}$ and their ground truth labels $\mathbf{y}_s^*$, we train feature net $\mathcal{S}$ and lighting net $\mathcal{L}$ through the following loss function:

$$\min_{\mathcal{S},\mathcal{L}} \sum_{(i,j)\in\Omega} \underbrace{[(\mathcal{L}(\mathcal{S}(\mathbf{s}_i)) - \mathbf{y}_{si}^*)^2 + (\mathcal{L}(\mathcal{S}(\mathbf{s}_j)) - \mathbf{y}_{si}^*)^2]}_{\text{regression loss for synthetic}} + \underbrace{\lambda(\mathcal{S}(\mathbf{s}_i) - \mathcal{S}(\mathbf{s}_j))^2}_{\text{feature loss}}, \tag{3.1}$$

where $\mathbf{s}_i$ and $\mathbf{s}_j$ are a pair of synthetic images with the same SH lighting, different IDs, and different small random deviations from frontal pose. $\mathbf{y}_{si}^*$ represents their ground truth label. Ω is a set containing all such pairs. λ is the weight for feature loss. Besides the regression loss, we also add a MSE feature loss that enforces the lighting related features of face images with the same SH to be the same. This encourages the lighting related features to contain no information about face ID and pose.

With trained $\mathcal{S}$ and $\mathcal{L}$, we need to train the feature net $\mathcal{R}$ for real face images $\mathbf{r}$ so that the lighting related features for real data ($\mathbf{f}_r = \mathcal{R}(\mathbf{r})$) lie in the same space as that of synthetic data ($\mathbf{f}_s = \mathcal{S}(\mathbf{s})$). Our idea is inspired by recently proposed Generative Adversarial Networks (GAN) [32], which have proved to be very effective to synthesize realistic images. In our setting, a discriminator $\mathcal{D}$ is trained to distinguish $\mathbf{f}_r$ and $\mathbf{f}_s$, while $\mathcal{R}$ is trained so that $\mathbf{f}_r$ would make $\mathcal{D}$ fail. By playing this

minmax game, the distribution of $\mathbf{f}_r$ will be close to that of $\mathbf{f}_s$. Wasserstein GAN (WGAN) [89] is used as our training strategy since it can alleviate the "mode dropping" problem and generate more realistic samples for image synthesis. However, making the distribution of $\mathbf{f}_r$ and that of $\mathbf{f}_s$ similar is not enough for our regression problem since the mapping can be arbitrary to some extent. As shown in Figure 3.1, both these two mappings would make two sets of points have similar distributions, but not both of them are correct since every point has its unique label. To deal with this problem, we use the noisy ground truth of real data as "anchor points" during training. As a result, the final loss function for our problem is defined as follows:

$$\min_{\mathcal{R}} \max_{\mathcal{D}} \underbrace{\sum_{i} (\mathcal{L}(\mathcal{R}(\mathbf{r}_i)) - \hat{\mathbf{y}}_{ri})^2}_{\text{regression loss for real}} + \mu \underbrace{(\mathbb{E}_{\mathcal{S}(\mathbf{s})\sim\mathbb{P}_s}[\mathcal{D}(\mathcal{S}(\mathbf{s})] - \mathbb{E}_{\mathcal{R}(\mathbf{r})\sim\mathbb{P}_r}[\mathcal{D}(\mathcal{R}(\mathbf{r}))])}_{\text{adversarial loss}} \quad (3.2)$$

where $\mathbb{P}_s$ and $\mathbb{P}_r$ are the distributions of lighting related features for synthetic and real images respectively.

Following [32, 89], the discriminator $\mathcal{D}$ and feature net $\mathcal{R}$ are trained in an alternating fashion. While training $\mathcal{D}$, RMSProp [90] is applied and Adadelta [91] is used to train $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{L}$ as discussed in [89]. The details on how to train the whole model are illustrated in Algorithm 3.

Algorithm 3: Training procedure for LDAN

Train $\mathcal{S}$ and $\mathcal{L}$ for synthetic data using loss function in Equation 3.1 by Adadelta.
Compute lighting related features for synthetic images using $\mathbf{f}_{si} = \mathcal{S}(\mathbf{s}_i)$.
for number of training epochs **do**
    for k=1 to 1 iterations **do**
        Sample 128 $\mathbf{f}_s$ and $\mathbf{r}$. Train discriminator $\mathcal{D}$ through the following loss using RMSProp:

$$\max_{\mathcal{D}} \mathbb{E}_{\mathbf{f}_s \sim \mathbb{P}_s}[\mathcal{D}(\mathbf{f}_s)] - \mathbb{E}_{\mathcal{R}(\mathbf{r}) \sim \mathbb{P}_r}[\mathcal{D}(\mathcal{R}(\mathbf{r}))]$$

    end for
    for k=1 to 4 iterations **do**
        Sample 128 $\mathbf{r}$ and train $\mathcal{R}$ through the following loss using Adadelta:

$$\min_{\mathcal{R}} \sum_i (\mathcal{L}(\mathcal{R}(\mathbf{r}_i)) - \hat{\mathbf{y}}_{ri})^2 - \mu \mathbb{E}_{\mathcal{R}(\mathbf{r}) \sim \mathbb{P}_r}[\mathcal{D}(\mathcal{R}(\mathbf{r}))]$$

    end for
end for

	SIRFS log	SIRFS SH	REAL	LDAN	Model B	Model C
top-1 (%)	60.72	56.04	61.29 (±1.8)	**65.73** (±1.78)	56.62 (±3.86)	63.03 (±0.91)
top-2 (%)	79.65	74.39	81.95 (±1.3)	**84.57** (±1.35)	76.94 (±4.10)	82.79 (±0.35)
top-3 (%)	87.27	83.74	90.59 (±0.7)	**92.43** (±0.59)	86.69 (±3.39)	91.21 (±0.47)

Table 3.1: Accuracy of different methods. Standard deviation is shown in the parentheses for learning based methods.

3.4 Experiments

3.4.1 Data Collection

Real Face Images: The proposed LDAN requires a large number of both synthetic and real face images for training. To collect the real face images, we download images with faces from the Internet. The SIRFS method proposed by Barron and Malik [31] is then applied to these face images to get the noisy ground truth SH for lighting. Since SIRFS was proposed to estimate lighting for general

objects, the prior they use is not face-specific. To get a better constraint for a face shape, we apply Discriminative Response Map Fitting (DRMF) [92] to estimate the facial landmarks and pose. Then, a 3DMM [67] is fitted to get an estimation of the face depth map which is used as a prior to constrain the face shape estimation of SIRFS. We collected $40,000$ faces with noisy ground truth SH for training.

Synthetic Face Images: We apply the 3D face model provided by [93] to generate $40,000$ pairs of faces. Each pair of these faces are under the same lighting but with different identities and a small random variation with respect to frontal pose.

MultiPie: The MultiPie dataset [94] contains a large number of face images of different IDs taken under different poses and illumination conditions. From this data set, $4,980$ face images are chosen, which contain 250 IDs in frontal pose under 19 lighting conditions. Though the ground truth lighting parameters are not provided for each of these face images, the lighting condition group under which a face image is taken is given. This data is used only for evaluation in our experiments.

3.4.2 Implementation Details

We use the same network structure for feature net $\mathcal{S}$ and $\mathcal{R}$. We apply the ResNet structure [6] to define a feature net. It takes a 64×64 RGB face image as input and outputs a 128 dimensional feature vector. We define our lighting net $\mathcal{L}$ and discriminator $\mathcal{D}$ to be 2 and 3 fully connected layers respectively. The lighting net outputs 18 dimensional lighting parameters and $\mathcal{D}$ outputs the score for being

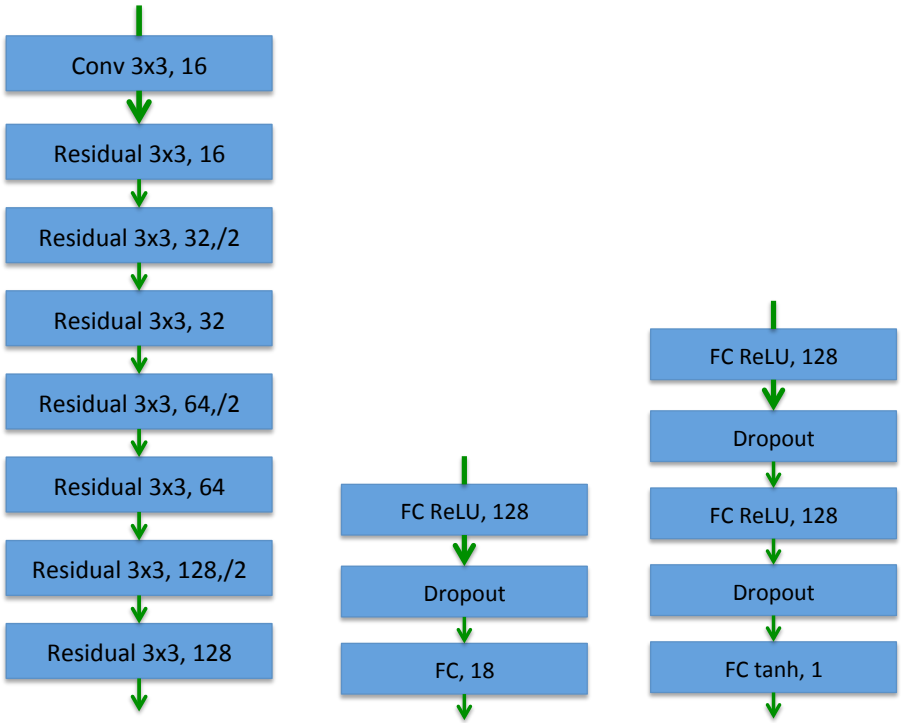
a lighting related feature of real data.

We show the structure of our networks in this section. As mentioned above, we borrow the structure of ResNet [6] to define our feature net. Figure 4.5 (a) shows the details. A block like "Conv 3×3, 16" means a convolutional layer with 16 filters, the size of each filter is $3 \times 3 \times n$ where n is the number of input channels. This convolutional layer is followed by a batch normalization layer and a ReLU layer. A block like "Residual 3×3 32" means a residual block of two 3×3 convolutional layers with skip connections: each of the two convolutional layers has 32 filters and is followed by batch normalization and ReLU layer. ",/2" means the stride of the first convolutional layer in residual block is 2. The output of the feature net is a 128 dimensional feature.

Figure 4.5 (b) shows the structure of the lighting net. "FC ReLU 128" means a fully connected layer whose number of outputs is 128 followed by a ReLU layer. "Dropout" means a dropout layer with dropout ratio being 0.5. "FC, 18" means a fully connected layer with 18 outputs.

Figure 4.5 (c) shows the structure of the discriminator. "FC tanh, 1" means a fully connected layer with one output followed by a tanh layer.

While training the proposed model, we first train discriminator $\mathcal{D}$ for 1 iteration and then train feature net $\mathcal{R}$ for 4 iterations. We alternate these two steps for 25 epochs. We choose $\mu = 0.01$, and $\lambda = 0.01$. Our algorithm is implemented using Keras [95] with Tensorflow [96] as backend.



(a) Feature net (b) Lighting net (c) Discriminator

Figure 3.2: (a), (b) and (c) show the structure of feature net, lighting net, and discriminator used in our paper.

3.4.3 Evaluation Metric

Since ground truth lighting parameters for real face images are not available, it is difficult to evaluate the accuracy of regressed lighting quantitatively. We propose an "indirect" quantitative evaluation metric based on classification, and test our method on the MultiPie data set, which contains face images taken under 19 lighting conditions. More specifically, after regressing the SH for each test face image, 90% of them are used to compute the mean SH for each lighting condition. Then, each of the rest of the face images are assigned to the 19 lighting conditions based on the Euclidean distance between its estimated SH and the mean SH. We carry out 10 cross validations for this classification measurement to make use of all the data.

3.4.4 Experimental Results

To show the effectiveness of the proposed method, we compare our results with the SIRFS [31] based method in this section. In SIRFS, the shading of a face is formulated in logarithm space, i.e. $\log\{s_i\} = Y_i\mathbf{l}$ where s_i is the shading at the i-th pixel, Y_i is the i-th row of Y and $\mathbf{l}$ represents the SH. $\mathbf{l}$ estimated in this way is not the correct SH. To estimate the correct SH lighting, we assume that the normal of each pixel estimated by SIRFS is in Euclidean space instead of logarithm space. This assumption is reasonable since we adapted the SIRFS method by estimating a face depth map using 3DMM in Euclidean space, and constrained the estimated face shape to be consistent with it. Supposing $\hat{\mathbf{l}}$ is the correct SH, the shading can

	LDAN	LDAN w/o Adversarial	LDAN w/o Regression	LDAN w/o Fixed Lighting Net
top-1 (%)	**65.73** (±1.78)	63.63 (±2.12)	30.72 (±0.63)	63.95 (±0.60)
top-2 (%)	**84.75** (±1.35)	83.44 (±1.57)	49.12 (±0.85)	83.97 (±0.25)
top-3 (%)	**92.43** (±0.59)	91.48 (±1.09)	61.58 (±1.07)	92.07 (±0.46)

Table 3.2: Results of ablation study. Standard derivation is shown in the bracket.

be found by $s_i = Y_i\hat{\mathbf{l}}$. Then $\hat{\mathbf{l}}$ can be found by solving the following equation:

$$Y\hat{\mathbf{l}} = \exp\{Y\mathbf{l}\}. \tag{3.3}$$

This is an over complete linear equation as the number of pixels is larger than the dimension of SH.

Comparison with baselines. Table 3.1 compares the proposed method with the SIRFS based method using the classification measurement on the MultiPie data set. We denote the original output of the SIRFS method as SIRFS log, and SIRFS SH is used to denote the corrected SH using Equation (3.3). We test these two methods on the original resolution of the MultiPie data which is roughly 220×270 after cropping the faces. REAL in Table 3.1 represents our baseline method which uses SIRFS SH as ground truth to train a deep CNN without synthetic data. REAL and LDAN are trained 5 times and the mean accuracies are shown in Table 3.1. We notice that SIRFS SH, which solves Equation (3.3) based on SIRFS log, performs worse than SIRFS log. According to Equation (3.3), the accuracy of SIRFS SH depends not only on the accuracy of SIRFS log, but also on the accuracy of estimated normals. The noisy estimation of normals would make the performance of estimated SIRFS SH more noisy. The performance of REAL is better than SIRFS SH, though

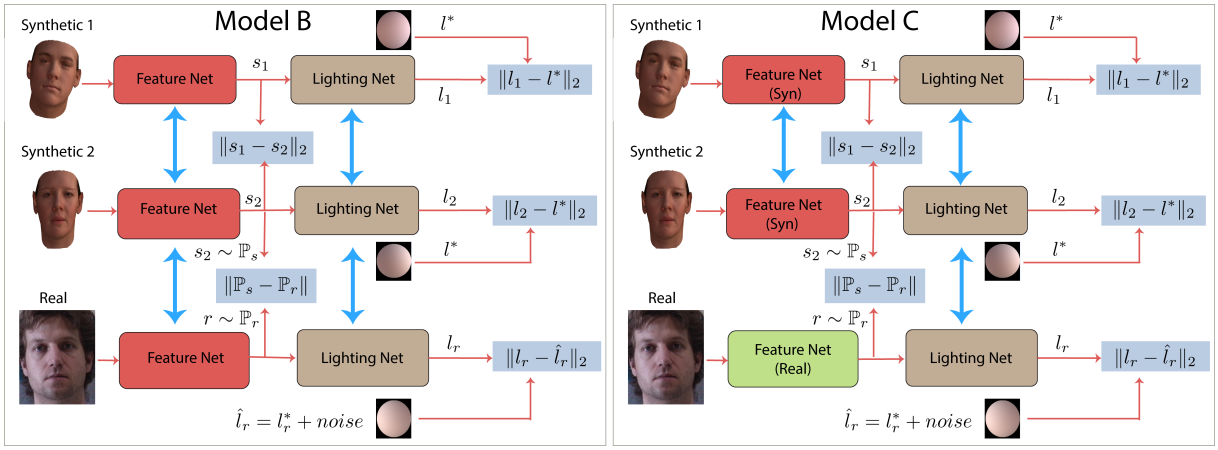
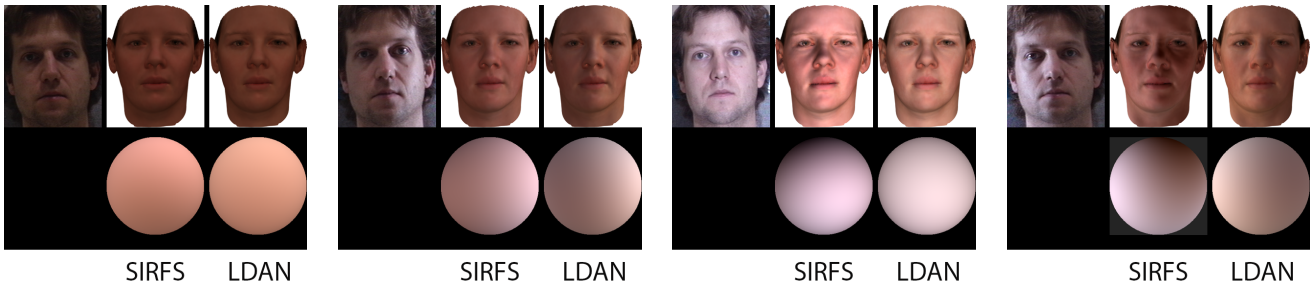


Figure 3.3: Two models we use to compare with the proposed LDAN. Different from LDAN, Model B uses the same feature net for synthetic and real data; Model C trains feature nets for synthetic and real data together.

it is trained directly using the output of SIRFS SH as the ground truth label. This shows that by observing a large amount of data, the deep CNN itself can be robust to noise to some extent. This is an advantage for learning based methods compared with optimization based algorithms. LDAN outperforms REAL by more than 4% and SIRFS SH by more than 9% for top-1 accuracy, showing the effectiveness of the proposed method.

We further propose two other baselines to compare with LDAN as shown in Figure 3.3. Different from LDAN, Model B and Model C learn the feature nets for synthetic and real data simultaneously and map the lighting related features of them to the same space. These two models are inspired by [82] and [84]. For Model B, synthetic and real data share the same feature net. Since synthetic data and real data are quite different from each other, using a single feature net it is difficult to make their lighting features have the same distribution, and we do not expect good performance. Model C defines different feature nets for synthetic and real data. The



(a) (b) (c) (d)

Figure 3.4: First row: MultiPie face image, rendered synthetic face with SIRFS estimated lighting and rendered synthetic face with LDAN estimated lighting. Second row: the hemisphere visualization of the corresponding estimated lightings. Images are best viewed on screen.

difference between Model C and LDAN is that Model C tries to map lighting related features for synthetic and real data to the same space, which might be different from that learned with synthetic data alone, whereas LDAN tries to directly map lighting related features of real data to the space of synthetic data. Intuitively, compared with LDAN, Model C is more easily affected by the noisy labels of real data since the training of the feature net for synthetic data is affected by the real data.

Model B and C are also trained 5 times and their mean accuracies are shown in Table 3.1 for comparison. We notice that Model B performs even worse than REAL, which shows that a single feature net for both synthetic and real data is not enough. LDAN and Model C outperform all other methods in Table 3.1. Moreover, LDAN performs better than Model C, showing that it is more robust to the noise in the labels of real data.

Ablation Study. To investigate the effectiveness of adversarial loss and regression loss, we carry out ablation studies for LDAN. We train the feature net 5 times for real data without adversarial loss or regression loss respectively and com-

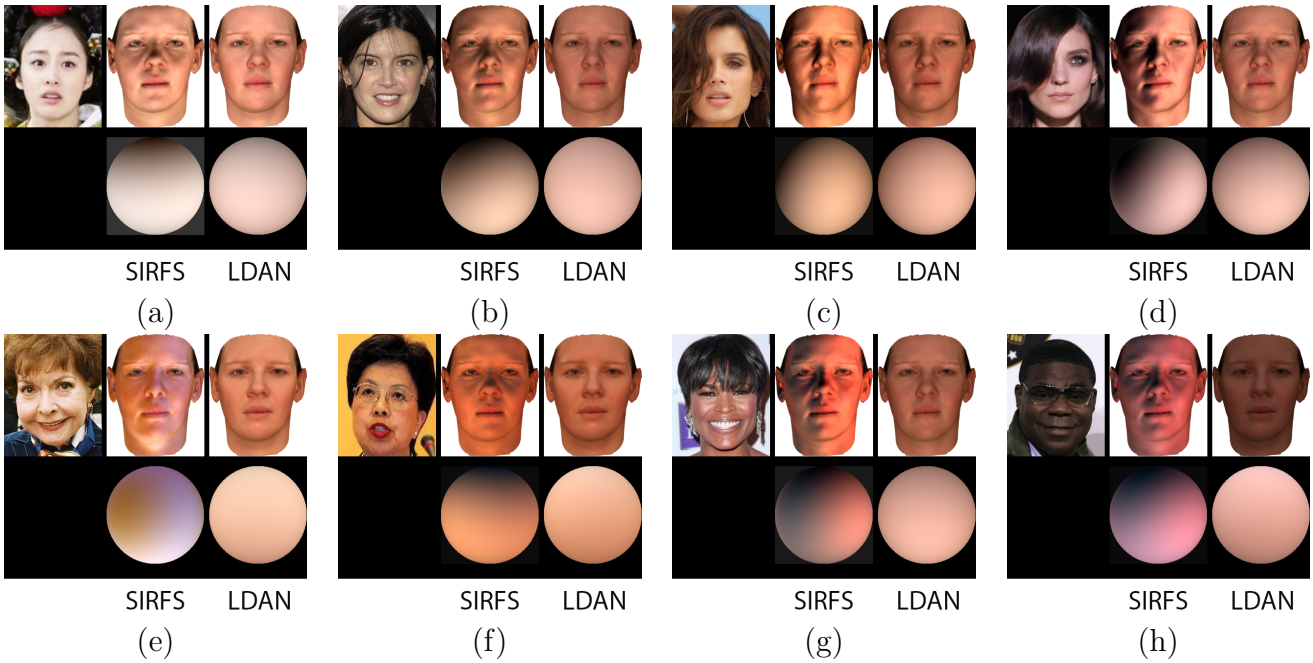


Figure 3.5: First row: CelebA face image, rendered synthetic face with SIRFS estimated lighting, and rendered synthetic face with LDAN estimated lighting. Second row: the hemisphere visualization of the corresponding estimated lightings. Images are best viewed on screen.

pare the results with LDAN in Table 3.2. Without adversarial loss, the performance of LDAN is better than REAL in Table 3.1, which means that synthetic data can help to regress lighting in this case. Without regression loss, on the other hand, the performance of LDAN drops dramatically. This is because the mapping of the distribution of lighting related features of real data to that of synthetic data is arbitrary as shown in Figure 3.1. This is problematic for a regression task where each data has its unique label. Having noisy ground truth as "anchor points", as we do in LDAN, can alleviate this problem and give much better results. We also train LDAN without fixing the lighting net and show the results in Table 3.2. We notice that the performance is similar to training LDAN without adversarial loss. This is expected since the lighting net will be trained to adapt to the noisy labels, reducing

Method	LDAN			SIRFS SH
Resolution	64×64	32×32	16×16	64×64
top-1 (%)	**65.73** ($\pm$1.78)	64.89 ($\pm$1.65)	61.72 ($\pm$1.72)	42.17
top-2 (%)	**84.75** ($\pm$1.35)	84.39 ($\pm$1.33)	82.17 ($\pm$1.01)	61.94
top-3 (%)	**92.43** ($\pm$0.59)	92.10 ($\pm$0.74)	90.94 ($\pm$0.81)	74.51

Table 3.3: Accuracy of LDAN for different scales of face images.

the impact of the adversarial loss.

Visualizing Estimated Lighting. Figure 3.4 and Figure 3.5 visualize the SH parameters estimated by SIRFS, and LDAN from MultiPie images and the CelebA [97] data set respectively. Note that for visualization purposes, we show the images cropped by bounding boxes instead of images cropped by facial landmarks. Though there are few images with strong side light effects, we notice that LDAN can still work reasonably well for such images as shown in Figure 3.4 (b) and (d). However, the predicted lightings are not as sharp as those by SIRFS. This is mainly because the performance of learning based methods are heavily dependent on the training data. Without sufficient face images with strong side light for training, the performance of LDAN on those images may not be optimal. We notice that the lighting predicted by SIRFS can have incorrect directions (Figure 3.5 (a) (b) (c) and (d) as well as Figure 3.4 (c)). One of the reasons is the effect of the hair. Since the facial landmark detection method is not perfect, some of the hair regions are included in the cropped face images, and this confuses SIRFS. Moreover, some lighting predicted by SIRFS have the incorrect color tone, especially for faces with dark reflectance, as shown in Figure 3.5 (e) (f) (g) and (h). On the other hand, our learning based method, LDAN, is not affected by these two issues.

Robustness to Low Resolution Images. To investigate the robustness of the proposed method for low resolution images, we downsample face images of MultiPie to 32×32 and 16×16 and then resize them to 64×64 and evaluate the lighting classification accuracy using our trained LDAN model. Table 3.3 shows the performance of LDAN on low resolution face images. We notice that the trained model is quite robust to low resolution images, even for face images with size 16×16, the top-1 accuracy only drops 4% compared with the original resolution (64×64). To compare, we also run SIRFS on 64×64 face images. Since we cannot run 3DMM on lower resolution images to get a good initialization directly, we fit the 3D model on the original resolution and then resize it accordingly. We notice that the performance of SIRFS drops a lot (14%) even on 64×64 images.

Running Time. We run experiments on a workstation with 4 Intel Xeon CPUs and 80 GB memory. While running on a GPU, we use one NVIDIA GeForce TITAN X. For a 64×64 RGB face image, SIRFS [31] takes 47 second to predict the lighting parameters. The proposed deep CNN can predict 390 such face images on the CPU and $2,400$ face images on the GPU per second, so it is potentially $100,000$ times faster.

3.4.5 Object 2D Keypoints Detection

synthetic	regression	fine-tune	LDAN	3DINN [27]	regression_gt
31.90%	69.61%	76.12%	**79.66**%	81.95%	90.57%

Table 3.4: PCK value at $\alpha = 0.1$ for different methods. We notice that LDAN outperforms regression and fine-tune.

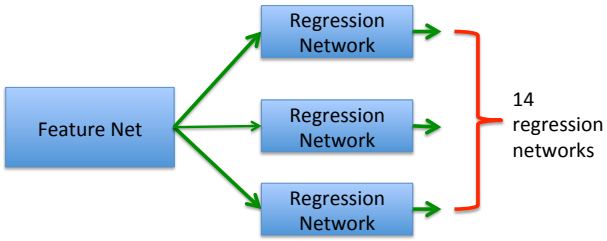


Figure 3.6: Illustration of the network for keypoints regression.

Since ground truth lighting is hard to obtain, to better quantitatively check the effectiveness of the proposed method, we further apply LDAN to object 2D keypoint detection, which has ground truth labels. The keypoint-5 dataset provided by 3DINN [27] has ground truth labels for 2D keypoints of sofa, chair, bed and swivel chair. [26] provided synthetic images of sofa and chair and their corresponding labels. Since 3DINN has achieved very high accuracy on the chair data set, we focus on using sofa to test our method.

Proposed Method for 2D Keypoint Regression

Similar to the lighting regression formulation, let us denote $\mathcal{S}$ as the feature network for synthetic data, $\mathcal{R}$ as the feature network for real data, and $\mathcal{L}$ as the regression network. Specifically, we use $\mathcal{L}_j$ to represent the regression network for the j-th keypoint, where $j = 1, 2, ..., 14$. Using $(\mathbf{s}_i, \mathbf{y}^*_{si})$ as (data, ground truth label) pair for synthetic data, and y^{j*}_{si} to represent the ground truth location of the j-th keypoint. We train the feature net $\mathcal{S}$ and regression net $\mathcal{L}$ using the following loss function:

$$\min_{\mathcal{S},\mathcal{L}} \sum_{i,j} (\mathcal{L}_j(\mathcal{S}(\mathbf{s}_i)) - \mathbf{y}^{j*}_{si})^2. \tag{3.4}$$

Note that since we do not have two sofa images that have exactly the same 2D keypoints, we ignore the feature loss.

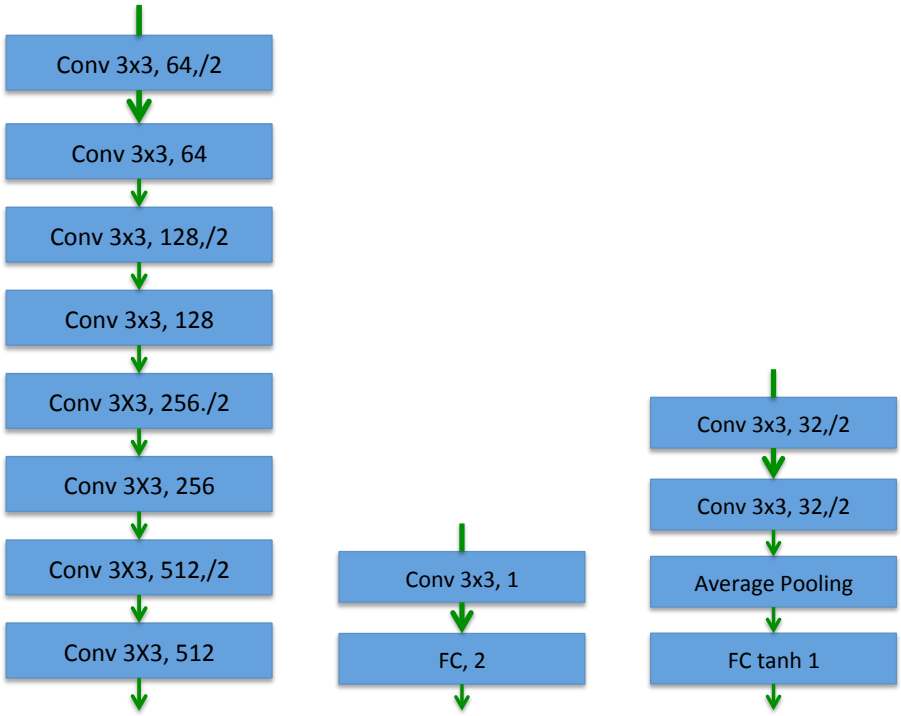
After training $\mathcal{S}$ and $\mathcal{L}$, we fix $\mathcal{S}$ and $\mathcal{L}$ and train the feature net $\mathcal{R}$ with real data together with a discriminator $\mathcal{D}$. Suppose $\mathbf{r}_i$ represents the i-th real data and $\hat{\mathbf{y}}_{ri}$ represents its corresponding noisy label. More specifically, letting $\hat{\mathbf{y}}_{ri}^j$ represent the location of j-th keypoint, we train $\mathcal{R}$ and $\mathcal{D}$ using the following loss function:

$$\min_{\mathcal{R}} \max_{\mathcal{D}} \underbrace{\sum_{i,j} (\mathcal{L}_j(\mathcal{R}(\mathbf{r}_i)) - \hat{\mathbf{y}}_{ri}^j)^2}_{\text{regression loss for real}} + \mu \underbrace{(\mathbb{E}_{\mathcal{S}(\mathbf{s})\sim\mathbb{P}_s}[\mathcal{D}(\mathcal{S}(\mathbf{s})] - \mathbb{E}_{\mathcal{R}(\mathbf{r})\sim\mathbb{P}_r}[\mathcal{D}(\mathcal{R}(\mathbf{r}))])}_{\text{adversarial loss}} \quad (3.5)$$

where $\mu = 0.05$ and $\mathbb{P}_s$ and $\mathbb{P}_r$ are the distributions of keypoints related features for synthetic and real images respectively.

Training Details.

To mimic noisy labels, we apply the code provided by 3DINN to predict the keypoints of sofa, and then double the noise of these labels so they contain more noise. Supposing $\mathbf{l}^*$ is a ground truth location of a keypoint (a 2 dimensional vector) and $\mathbf{l}'$ is the keypoint location predicted by 3DINN, we double the noise in the label $\mathbf{l}'$ and get $\hat{\mathbf{l}} = 2\mathbf{l}' - \mathbf{l}^*$. Unless otherwise specified, **we use $\hat{\mathbf{l}}$ as noisy labels to train networks**. Different from 3DINN, we formulate keypoint detection as a regression problem. Similar to our LDAN model, the network is designed to have a feature network and regression network. Inspired by [27], we define a separate regression network for each keypoint, resulting in 14 different regression networks as illustrated in Figure 3.6. Following [26], we normalize the keypoint location by the width and



(a) Feature net (b) Regression net (c) Discriminator

Figure 3.7: (a), (b) and (c) show the structure of feature net, keypoint regression net, and discriminator used in our system.

height of the image, so that the (x, y) coordinates of each 2D keypoint are within $[0, 1]$. We resize each of the images from the dataset to be 256×256. Our feature net takes a 256×256 image as input and outputs a $16 \times 16 \times 512$ tensor as the feature vector. Our regression network takes this feature as input and predicts the 2D location of the corresponding keypoint. Figure 3.7 (a), (b), and (c) shows the structure of the feature net, regression net and discriminator separately. The notion of each block is the same as those in Figure 4.5.

Experiments We use the Percentage of Correct Keypoints (PCK) metric [98] to evaluate the accuracy. A 2D keypoint prediction is correct if it lies within a radius $\alpha * L$ of the ground truth, where L is the diagonal of the image with $0 < \alpha < 1$. Following [27], we show the PCK curve with the value of α between

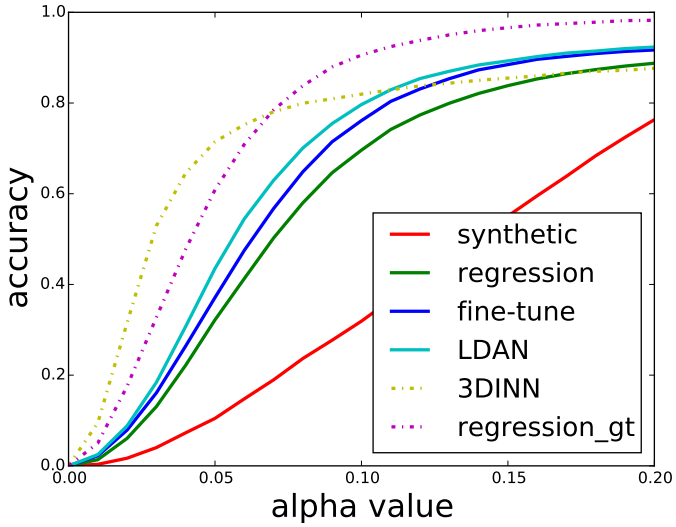


Figure 3.8: PCK curve of real sofa images for different methods.

0.0 and 0.2 in Figure 3.8. We also show the PCK value at $\alpha = 0.1$ (as in [26]) in Table 3.4 and compare it with several reasonable baselines: (1) Training the network using real data with the noisy label ("regression"); (2) Training the network using synthetic data and testing it on real data ("synthetic"); (3) Fine tune the network trained on synthetic data using real images with the noisy labels ("fine-tune"). We also show the performance of 3DINN and performance of training the network using real data with real ground truth labels as supervision as references ("3DINN" and "regression_gt", both are trained with real ground truth). These results show that the proposed method works better than other methods that train with noisy labels, including directly training the network and fine-tuning on it. This shows the effectiveness of the proposed method.

3.5 Summary

In this chapter, we propose a lighting regression network to predict Spherical Harmonics of environment lighting from face images. Lacking the ground truth labels for real face images, we applied an existing method to get noisy ground truth. To alleviate the effect of noise, we propose to apply the idea of adversarial networks and use synthetic face images with known ground truth to help train a deep CNN for lighting regression. Compared with existing methods, the proposed method is more efficient and can predict more consistent Spherical Harmonics from different faces taken under the same lighting. Due to a lack of direct measurement of estimated Spherical Harmonics, we further apply the proposed method to regress 2D keypoints, for which ground truth labels are provided. Our experiments further demonstrate the effectiveness of the proposed method.

Chapter 4: Deep Single-Image Portrait Relighting

In this chapter, we introduce a single image portrait relighting algorithm. Given a source portrait and a target lighting condition, the proposed method can generate a new portrait image under the target lighting condition. Due to the lack of an "in the wild" dataset that is suitable for this task, we apply physical based rendering to generate a large scale, high resolution, "in the wild" dataset. A deep CNN is then trained using the proposed dataset. Our experiments demonstrate that the proposed method can generate high resolution relighted images accurately.

This work [99] is in collaboration with Sunil Hadap, Kalyan Sunkavalli and David W. Jacobs.

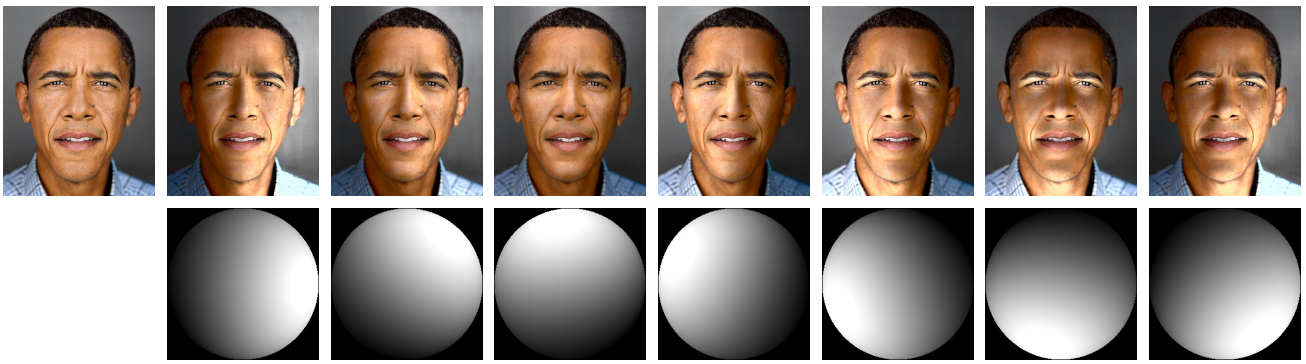


Figure 4.1: Our algorithm takes a portrait image and a target lighting as input and generates a new portrait image.

4.1 Introduction

Portrait relighting is one of the most interesting and important applications of image editing. The goal of the this work is to design an automatic single-image portrait relighting algorithm, which takes a portrait image and a target lighting as input and generates a new portrait image under the target lighting condition.

There are physically-based relighting methods that explicitly reconstruct the face geometry, reflectance, and lighting and then re-render this reconstruction using a novel lighting [72, 100–104]. However, single image face reconstruction is still an open problem, and even the state-of-the-art methods have strong errors, e.g., inaccurate estimation of face geometry. These errors can propagate into the relighting and lead to poor results. As a result, while these relighting methods are generally good at capturing lighting variations, they may contain artifacts that prevent them from looking realistic. In this work we leverage this property: we use a physically-based relighting method to generate a large-scale training dataset, and then use it to train a generative network to reproduce them while imposing an adversarial loss based only on real photographs. The supervised reconstruction loss allows the network to learn how to relight, while the adversarial loss ensures that the results are on the manifold of real photographs and do not have the errors due to the physically-based relighting method.

We first propose a ratio image [105] (RI) based rendering algorithm to generate a large scale, high resolution, "in the wild" deep portrait relighting dataset (DPR). In this algorithm, an image under a target lighting condition can be represented as

the ratio of the target shading and source shading, multiplied by the source image. Face normals and SH lighting of the source image are estimated by 3DDFA [106] and SfSNet [104] respectively. A novel As-Rigid-As-Possible [107] based warping method is then proposed to accurately align the estimated face normal to the portrait image. Spherical Harmonics (SH) [34,35] lighting is then randomly sampled from a lighting prior dataset [102] to relight the portrait image. We apply our proposed RI based algorithm to the high resolution CelebA dataset (CelebA-HQ) [108] and generate 138,135 relighted 1024×1024 portrait images with known SH lighting.

An hourglass network [33] is trained using the proposed DPR dataset for the portrait relighting task. It takes a source image and a target lighting as input and generates the relighted image. It also predicts the SH lighting for the source image using the features from the bottleneck layer to disentangle lighting information from the source image. We notice that the skip connections in the hourglass network prevent the bottle neck layer from learning meaningful facial information. A simple skip training strategy is then proposed to enforce facial information in the bottleneck layer, which can improve the quality of the generated images. Our network is first trained on 512×512 images and then fine tuned on 1024×1024 images. To the best of our knowledge, the proposed method can generate relighted images at the highest resolution among all deep learning based algorithms. We test the proposed method qualitatively on our proposed DPR dataset and flicker portrait dataset [1] and quantitatively on the MultiPie dataset [94]. All these experiments demonstrate that the proposed method can achieve state-of-the-art results both qualitatively and quantitatively.

To reiterate, the contributions of the proposed method are threefold. First, we propose a ratio image based algorithm to generate a large scale, high resolution "in the wild" deep portrait relighting dataset. A novel As-Rigid-As-Possible based warping method is proposed to align the face normals accurately with the face image. Second, we design an automatic single-image portrait relighting algorithm which takes a source image and target SH lighting as input and generates a face image under the target lighting. Third, our trained network can generate 1024×1024 relighted portrait images, which, to the best of our knowledge, is the highest resolution among all deep learning based portrait relighting methods.

4.2 Related Work

Quotient Image for Portrait Relighting Shashua and Riklin-raviv [105] proposed to use the quotient (ratio) image for portrait relighting. They require multiple reference images as input and assume all these images are in frontal view. Stoschek [109] extended the ratio image to arbitrary pose by aligning facial landmarks of the source and target image. Wen *et al* [110] proposed to render a new image using the ratio of the radiance environment map. [111] proposed to apply ratio images to real time portrait illumination editing. However, their method requires capturing images of a static subject using a Light Stage apparatus. Due to the success of ratio images in portrait relighting applications, we apply this technique in our data preparation pipeline.

Inverse Rendering of Portrait Images Starting with the 3D Morphable Model (3DMM) [67], many inverse rendering methods for portrait images have been proposed [72, 100–104, 112, 113]. These methods decompose a portrait image into reflectance, normal and lighting. A relit portrait image can then be rendered by changing the lighting and keeping the normal and reflectance fixed. [100–102] are optimization based method, which are time consuming. [72,103,104,112,113] are all deep learning based methods. Compared with optimization based methods, they are more time efficient. However, due to the complexity of inverse rendering, all these methods can only work on low resolution images. On the contrary, our proposed method, focusing on portrait relighting, can be designed to generate very high resolution (1024×1024) images.

Photo and Portrait Style Transfer Photo and portrait style transfer [1,114–116] takes a source image and a reference image as input and transfers the style of the reference image to the source image. Since lighting can be treated as a kind of style, these methods can also be applied in portrait relighting applications. To generate a high quality portrait image, these methods usually require a high quality, non-occluded reference image that contains the desired lighting with a different subject as input, which limits the possible application scenarios. Different from these methods, our proposed method is a single-image based algorithm, lighting is specified as input and no reference image is required.

4.3 Deep Portrait Relighting Dataset

In this section, we introduce the Deep Portrait Relighting (DPR) dataset, which is a large scale, high resolution, "in the wild" image dataset generated for portrait relighting purposes. DPR is built on the high resolution CelebA dataset (CelebA-HQ) published by [108], which contains 30,000 face images from the CelebA [97] dataset with 1024×1024 resolution. We remove images on which the landmark detector [117] fails to detect landmarks, resulting in 27,627 images in the DPR dataset. For each of these images, we randomly select 5 lighting conditions from a lighting prior dataset [102] to generate relighted face images, leading to 138,135 relighted images.

4.3.1 Ratio Image to Relight Faces

We propose a ratio image based algorithm for data generation. Optimization based inverse rendering methods are either too slow [101, 102] or cannot generate high resolution images [72, 104]. Portrait style transfer methods, such as [1, 116], require a reference image as target light source, which is not flexible. To render a face image $\mathbf{I}$, we need the reflectance $\mathbf{R}$, normal $\mathbf{N}$ and lighting $\mathbf{L}$. We further assume that the reflectance of human face is Lambertian. A face image $\mathbf{I}$ can thus be represented as:

$$\mathbf{I} = \mathbf{R} \odot f(\mathbf{N}, \mathbf{L}), \tag{4.1}$$

where $\odot$ represents the element-wise product and f is the rendering function. To relight a face image, we apply the ratio image trick proposed in [105]. According to Eq 4.1, the same face under two different lighting conditions $\mathbf{L}$ and $\mathbf{L}^*$ can be represented as $\mathbf{I} = \mathbf{R} \odot f(\mathbf{N}, \mathbf{L})$ and $\mathbf{I}^* = \mathbf{R} \odot f(\mathbf{N}, \mathbf{L}^*)$. We know that

$$\begin{aligned} \mathbf{I}^* &= \mathbf{R} \odot f(\mathbf{N}, \mathbf{L}^*) \\ &= \frac{\mathbf{R} \odot f(\mathbf{N}, \mathbf{L}^*)}{\mathbf{R} \odot f(\mathbf{N}, \mathbf{L})} (\mathbf{R} \odot f(\mathbf{N}, \mathbf{L})) \\ &= \frac{f(\mathbf{N}, \mathbf{L}^*)}{f(\mathbf{N}, \mathbf{L})} \mathbf{I}. \end{aligned} \tag{4.2}$$

As a result, a portrait image $\mathbf{I}^*$ under lighting $\mathbf{L}^*$ can be generated given portrait image $\mathbf{I}$ and its normal and lighting.

4.3.2 Normal Estimation

There are many research studies target at estimating normals from portrait images. We use 3DDFA [118] [1] since it outputs the shape parameters of a 3DMM, which can be used to generate portrait normal images at arbitrary resolution. Although 3DDFA takes facial expression into consideration while fitting 3DMM, the normals estimated still cannot be acccurately aligned with the the portrait image. We believe this is due to the limited power of the 3DMM to model variations of face geometry, as 3DMM is built on a limited number of faces. To better align the estimated normals with the portrait image, so as to avoid possible artifacts in the relighted images, we propose a As-Rigid-As-Possible (ARAP) based normal

[1] Code provided by [106]

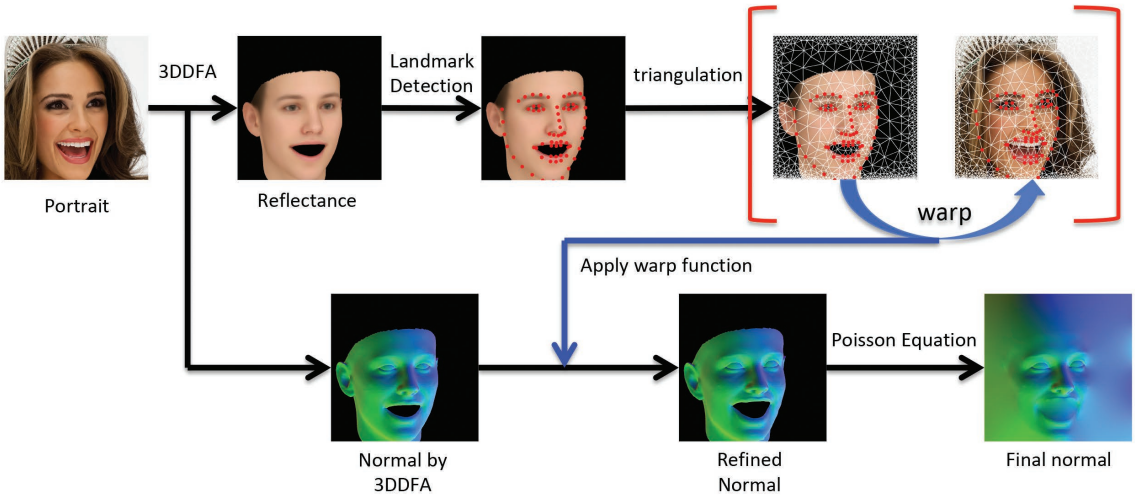


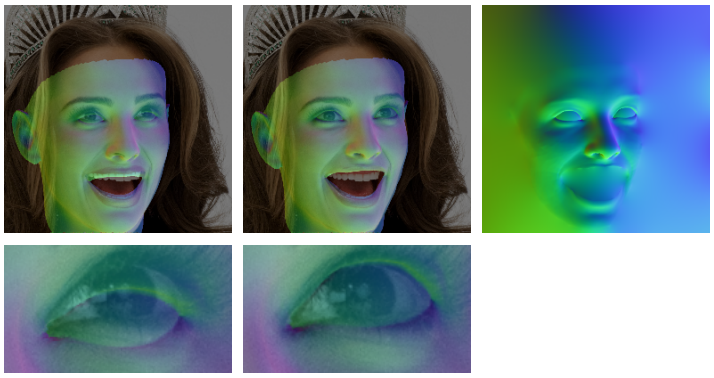
Figure 4.2: ARAP based normal refinement.

refinement algorithm.

4.3.2.1 ARAP Based Normal Refinement

Figure 4.2 illustrates the procedure of the ARAP based normal refinement algorithm. Using the 3DMM parameters predicted by 3DDFA [106], a mesh can be created. The "reflectance" image of the portrait can be obtained by projecting the generic reflectance map of the 3DMM model onto this mesh. [2] We then apply [117] to detect 68 facial landmarks on this "reflectance" image. These 68 detected facial landmarks, together with evenly sampled 198 points along the boundaries of the image are combined as "anchor points" and are used to create a triangle mesh on the reflectance image using Delauny Triangulation. Similarly, a triangle mesh is created for the portrait image. An As-Rigid-As-Possible transformation [107] (ARAP) is then applied to warp the triangle mesh of the "reflectance" image to the portrait image. The estimated warp function by ARAP is then applied to the face

[2]Note that this is not the real reflectance image of the portrait, and it is just used to help refine face normals.



(a) (b) (c)

Figure 4.3: We show the original face overlaid with normals estimated by 3DDFA [106] in (a) and our refined normals in (b). The second row of (a) and (b) show the right eye region.

normals estimated by 3DDFA to get refined normals as illustrated in Figure 4.2. To demonstrate the effectivenss of the proposed normal refinement method, we overlay the normals estimated by 3DDFA [106] and our refined normals with the original image, and show them in Figure 4.3 (a) and (b) respectively. It is clear that the quality of the alignment of normals w.r.t. portrait image at the eye and mouth has been improved significantly through our proposed normal refinement method.

We notice that our proposed ARAP normal refinement method cannot improve the misalignment of the ear and neck regions. This is because 3DMM cannot model the deformation of ear and neck well, and, to the best of our knowledge, there is no landmark detection algorithm for ears and necks. As a result, we remove the ear and neck regions from the refined normals to avoid possible artifacts in relighted images. In order to get a full normal image, we solve a Poisson equation to fill in the missing normals for ear, neck, mouth and background region as suggested by [72]. Figure 4.3 (c) shows the normals after filling the missing region.

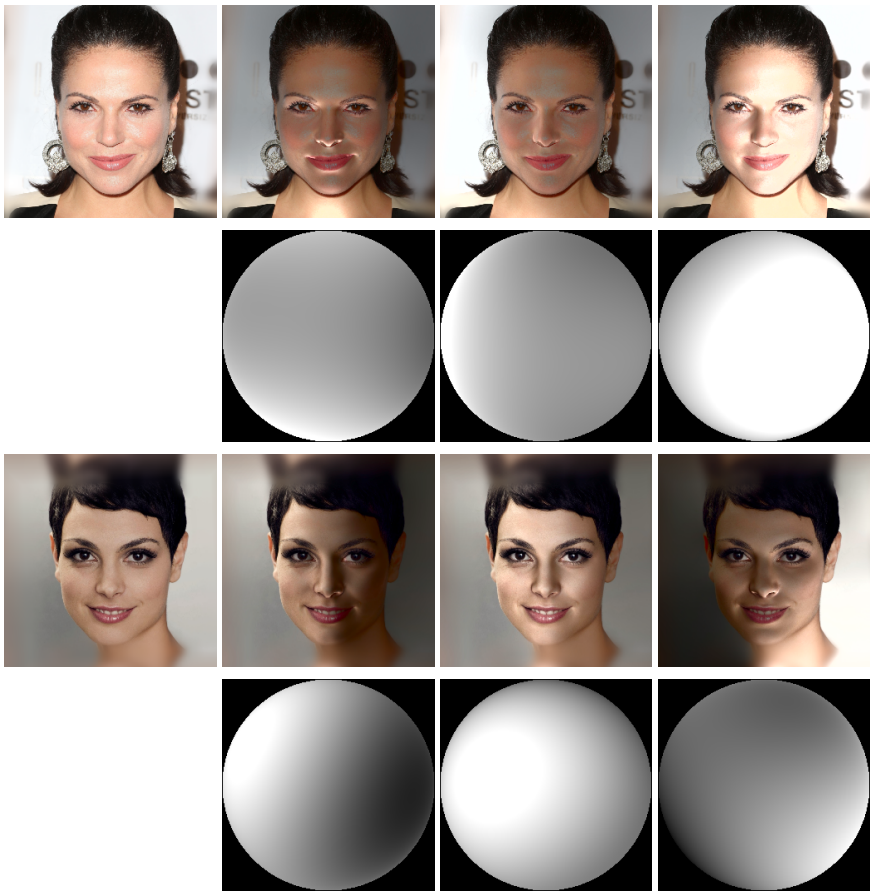


Figure 4.4: First column is the original image, second to forth columns in the first row are relighted images generated by our rendering pipeline, the second row shows the half sphere rendered using the corresponding SH lighting.

4.3.3 Relighting Images

For a portrait image $\mathbf{I}^*$, we apply our method to estimate normals $\mathbf{N}$ and use SfSNet [104] to estimate SH lighting $\mathbf{L}^*$. Then a target SH lighting $\mathbf{L}$ is randomly sampled from the lighting prior dataset [102]. Eq 4.3 is then used to generate the relighted face image $\mathbf{I}$. Due to the ambiguity of the color between lighting and reflectance, we apply the rendering pipeline to the luminance channel and keep the color of the portrait image unchanged. We show some examples of relighted face image in Figure 4.4.

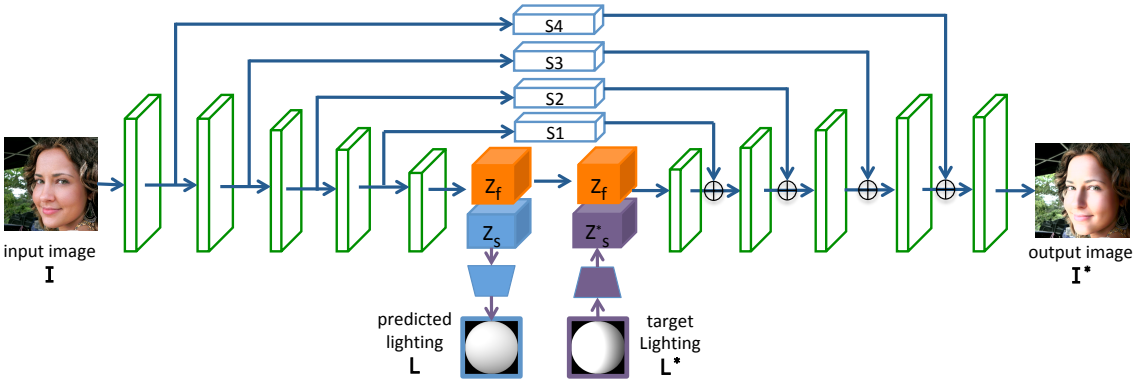


Figure 4.5: Our network takes a portrait image and a target SH lighting as input and outputs the SH lighting of the input portrait image and generates a new, relit portrait image using the target SH lighting.

4.4 Method

In this section, we introduce our proposed deep learning based single-image portrait relighting algorithm. We design an hourglass network for this task and use the DPR dataset created in the section 4.3 to train the network.

4.4.1 Main architecture for portrait relighting

Figure 4.5 shows the structure of our proposed hourglass network [33]. It has an encoder and a decoder part. Four skip connections are used to connect the features at different scales in the encoder part to their corresponding scale in the decoder part. To relight a face, our network takes a face image $\mathbf{I}$ and a target lighting $\mathbf{L}^*$ as input. The encoder extracts features $\mathbf{Z}$ which is divided into two parts: face feature $\mathbf{Z_f}$ which is independent of lighting; and lighting feature $\mathbf{Z_s}$. $\mathbf{Z_s}$ is then fed into a lighting regression network to predict the lighting $\mathbf{L}$ of the input face image $\mathbf{I}$. The target lighting $\mathbf{L}^*$ is then mapped to the lighting feature $\mathbf{Z}^*_\mathbf{s}$. $\mathbf{Z_f}$ and $\mathbf{Z}^*_\mathbf{s}$ are

concatenated together and fed into the decoder part to generate the relighted face image.

4.4.2 Supervision for training the network

As discussed in Section 4.3, our data preparation process generated 5 relighted images with known ground truth lighting for each image in CelebA-HQ dataset. To generate one training data, we randomly select one source image $\mathbf{I}_s$ and one target image $\mathbf{I}_t$ and their corresponding ground truth SH lighting $\mathbf{L}_s$ and $\mathbf{L}_t$ from these 5 relighted images. Our network then takes source image $\mathbf{I}_s$ and target lighting $\mathbf{L}_t$ as input and generates $\mathbf{L}_s^*$ and $\mathbf{I}_t^*$. $\mathbf{L}_s$ and $\mathbf{I}_t$ are used as ground truth to supervise the training. We apply an L_1 loss for generated portrait image $\mathbf{I}_t^*$ and an L_2 loss for the predicted lighting $\mathbf{L}_s^*$. An L_1 loss is further applied to the gradient of $\mathbf{I}_t^*$ to preserve edges and avoid possible blurry effect:

$$\mathcal{L}_I = \frac{1}{N_I}\left(||\mathbf{I}_t - \mathbf{I}_t^*||_1 + ||\nabla\mathbf{I}_t - \nabla\mathbf{I}_t^*||_1\right) + (\mathbf{L}_s - \mathbf{L}_s^*)^2, \tag{4.3}$$

N_I is the number of pixels in the image.

Since our "ground truth" images are generated using the ratio image trick, they may contain some artifacts due to inaccurate estimation of face normal or lighting. We thus propose to use a GAN loss to improve the quality of the generated images. As these artifacts mostly appear locally, we use a patch GAN [119] to force the distribution of local image patches to be close to that of a natural image. We follow the implementation of [119] (lsgan [120]) and use an MSE criterion for our GAN

loss:

$$\mathcal{L}_{GAN} = \mathbb{E}_{\mathbf{I}}(1 - D(\mathbf{I}))^2 + \mathbb{E}_{\mathbf{I_s}} D(G(\mathbf{I}_s, \mathbf{L}_t))^2, \tag{4.4}$$

where $\mathbf{I}$ is the real image, and G and D represent our relighting network and discriminator respectively. We use 1 as a label for real images and 0 as a label for fake images. While training, we use the images from FFHQ dataset [121] as real images in our GAN loss since images in this dataset contain more lighting variations.

A feature matching loss is further proposed to increase the the accuracy of the relighted portrait image. More specifically, same images under different lighting conditions should have the same face features, we thus define a feature loss as:

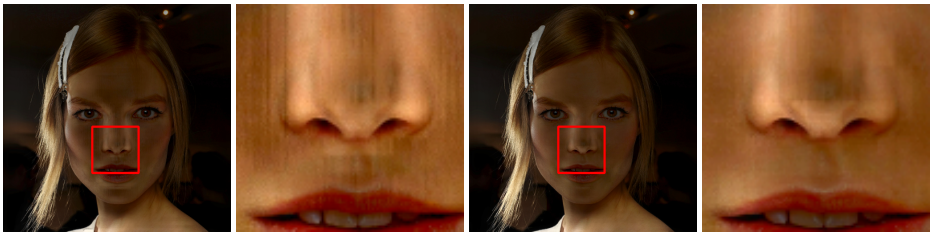
$$\mathcal{L}_F = \frac{1}{N_F}(\mathbf{Z_{f1}} - \mathbf{Z^*_{f2}})^2, \tag{4.5}$$

where $\mathbf{Z_{f1}}$ and $\mathbf{Z^*_{f2}}$ are face features of $\mathbf{I}_{s1}$ and $\mathbf{I}_{s2}$, and N_F is the number of elements in feature $\mathbf{Z}_f$.

4.4.3 Skip Training

When the Hourglass network is trained end-to-end (denoted as vanilla Hourglass), we notice that most of the facial information is passed through skip layers. Our facial feature $\mathbf{Z_f}$, on the other hand, contains little facial information. We thus propose a skip training strategy in which we train our network without skip connections first, then add skip layers one by one during subsequent training. We denote this as skip training. Figure 4.6 compares the relighted images generated by remov-

Figure 4.6: From left to right: output without skip layer S4, output without S4/S3, output without S4/S3/S2, output without S4/S3/S2/S1. Top row: vanilla Hourglass network, bottom row: Hourglass network with skip training.



(a) (b) (c) (d)

Figure 4.7: (a) output of vanilla Hourglass network, (b) rectangle region of (a), (c) output of Hourglass network with skip training, (d) rectangle region of (c). We increase the pixel intensity of (b) and (c) for visualization purpose.

ing the skip layers of vanilla Hourglass network and Hourglass network with skip training. We can see that with the skip training strategy, more facial information is kept in the feature layer. Figure 4.7 further demonstrates that skip training can help improve the quality of the generated results by removing artifacts around the nose. In the following discussion, unless otherwise specified, our network is trained with skip training.

4.4.4 Implementation Details

We discuss the implementation details of the proposed method in this section.

4.4.4.1 Network Structure

h1, h2, h3, h4 are down-sampling layers followed by residual blocks defined in [6]. h5, h6, h7 and h8 are designed as residual blocks [6] followed by upsampling layers. s1, s2, s3 and s4 are defined to be residual blocks [6]. For convenience, we defined one convolutional block as one convolutional layer followed by a batch normalization layer and ReLU activation. c1 is designed to have one convolutional block. c2 is designed to have three convolutional blocks (denoted as c2_1, c2_2, c2_3) followed by one convolutional layer (denoted as c2_o). More details of these blocks are shown in Table 4.1. Note that the output of h4 has 155 channels, from which 128 channels belong to face features $\mathbf{Z_f}$ and 27 channels belong to lighting feature $\mathbf{Z_s}$.

	h1	h2	h3	h4	h5	h6	h7	h8	
input channel number	16	16	32	64	155	64	32	16	
output channel number	16	32	64	155	64	32	16	16	
filter size	3	3	3	3	3	3	3	3	
	s1	s2	s3	s4	c1	c2_1	c2_2	c2_3	c2_o
input channel number	64	32	16	16	1	16	16	16	16
output channel number	64	32	16	16	16	16	16	16	1
filter size	3	3	3	3	5	3	1	1	1

Table 4.1: Details about each block of our network.

The lighting prediction network, which takes $\mathbf{Z_s}$ as input and predicts $\mathbf{L}$, is

defined as an average pooling layer followed by two fully connected layers whose number of channels are 128 and 9 respectively. The network that maps target lighting $\mathbf{L}^*$ to lighting features $\mathbf{Z}_\mathbf{s}^*$ is defined as two fully connected layers whose number of channels are 128 and 27. The 27 dimensional lighting feature is then repeated spatially so it has the same spatial resolution as $\mathbf{Z}_\mathbf{s}$ as illustrated in Figure 4.8.

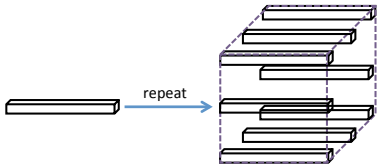


Figure 4.8: Illustration of repeating lighting feature spatially.

4.4.4.2 Training Detail

The overall loss for our network is a linear combination of the losses mentioned in Sec. 4.4.2:

$$\mathcal{L} = \mathcal{L}_I + \mathcal{L}_{GAN} + \lambda \mathcal{L}_F, \tag{4.6}$$

where $\lambda = 0.5$. Our network is trained for 14 epochs. We add our feature loss $\mathcal{L}_F$ after 10 epochs. For skip training, we train our network without any skip

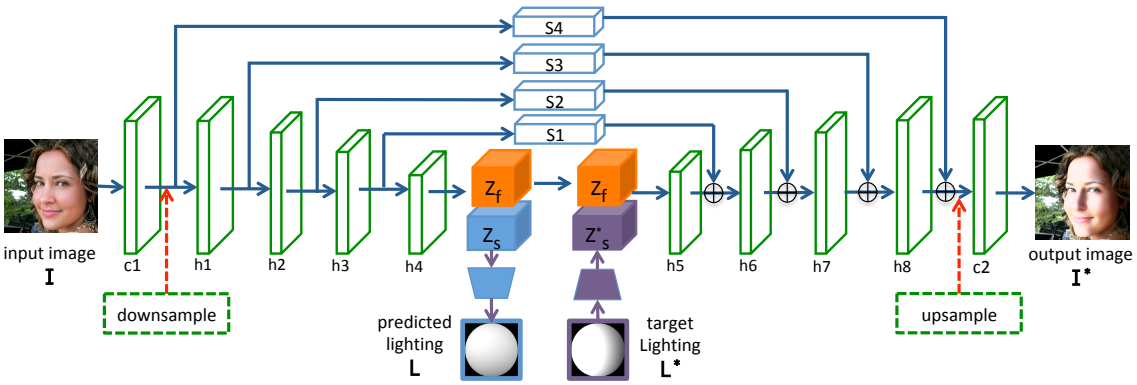


Figure 4.9: Network structure for 1024×1024 images.

connections for 5 epochs, and add skip connections one at each epoch after the fifth epoch, until all skip layers are added. We first train our network with images of resolution 512×512; most of our experiments are carried out under this resolution. Finally, we fine tune our trained network using images with resolution 1024×1024 with a simple modification. More specifically, an additional down sampling and up sampling layer is added to make our network compatible with 1024×1024 images as shown in 4.9. We train our network from scratch using the Adam optimizer [122] with default parameters.

4.5 Experiments

In this section, we evaluate our proposed method both quantitatively and qualitatively and compare it with some of the state-of-the-art methods. Since our network can predict lighting, it can be used in two ways for portrait relighting: (A) Given a source image $\mathbf{I}_s$ and a SH lighting $\mathbf{L}_t$, generate an image $\mathbf{I}_t$ (denoted as **SH-based way**). (B) Given a source image $\mathbf{I}_s$ and a reference image $\mathbf{I}_f$, extracting SH lighting $\mathbf{L}_t$ from $\mathbf{I}_f$ and use it to relight $\mathbf{I}_s$ to get $\mathbf{I}_t$ (denoted as **image-based way**). Our network is designed to be used as (A). Due to that reason, when the target SH lighting $\mathbf{I}_t$ is known (e.g. our DPR dataset) we use (A) for our relighting task. For datasets such as MultiPie [94], in which ground truth SH lighting is unknown, we use (B) for relighting.

4.5.1 Dataset and Evaluation Metric

Dataset: We demonstrate the effectiveness of the proposed method on the test set of our proposed DPR dataset. However, due to lack of real ground truth, we cannot evaluate the accuracy of the relighted images using this dataset. We thus propose to use the MultiPie dataset [94] for quantitative evaluation. The MultiPie dataset contains images of the same person under different lighting conditions, which can be used as source and target image pair. Each MultiPie image is formed by lighting with a dominant point light source, while the lighting conditions of most "in the wild" portrait images are diffuse. We thus generate images under 7 lighting conditions by averaging 3 to 4 original face images from MultiPie, so as to generate images under

more realistic, diffuse lighting conditions. We created 440 groups of images from our generated face images, each of which contains a source image $\mathbf{I}_s$, a target image $\mathbf{I}_t$ and a reference image $\mathbf{I}_f$. $\mathbf{I}_s$ and $\mathbf{I}_t$ are images with the same identity but with different lighting conditions, $\mathbf{I}_t$ and $\mathbf{I}_r$ are images of different identities but with the same lighting condition. When evaluating, a relighting algorithm takes $\mathbf{I}_s$ and $\mathbf{I}_f$ as input and predicts $\mathbf{I}_t$.

Evaluation metric: Since lighting is ambiguous up to a scale (e.g., longer exposure time may lead to a SH with high energy under the same lighting conditions), we propose to use a scale invariant Mean Squared Error (Si-MSE) [31] to evaluate the error between the generated image $\mathbf{I}_t^*$ and the ground truth image $\mathbf{I}_t$.

$$\text{Si-MSE} = \frac{1}{N_I} \min_{\alpha} (\mathbf{I}_t - \alpha * \mathbf{I}_t^*)^2, \tag{4.7}$$

where α is a scalar and N_I is the number of pixels in the image. To further check whether the generated image portrays the target lighting, we run SfSNet [104] to extract the lighting $\mathbf{L}_t$ and $\mathbf{L}_t^*$ from $\mathbf{I}_t$ and $\mathbf{I}_t^*$ respectively, and compute the scale invariant L_2 (Si-L_2) distance between $\mathbf{L}_t$ and $\mathbf{L}_t^*$. We choose to use SfSNet [104] since it is proven to work well at predicting consistent lighting for face images under the same lighting condition.

4.5.2 Ablation Study

To demonstrate the effectiveness of the GAN loss and feature loss, we show the quantitative and qualitative results of our network trained using $\mathcal{L}_I$, $\mathcal{L}_I + \mathcal{L}_{GAN}$ and

	Si-MSE	Si-L_2
$\mathcal{L}_I$	**0.00504**	**0.1307**
$\mathcal{L}_I + \mathcal{L}_{GAN}$	0.00658	0.1686
$\mathcal{L}_I + \mathcal{L}_{GAN} + \mathcal{L}_f$	0.00590	0.1444

Table 4.2: Ablation Study on MultiPie Dataset

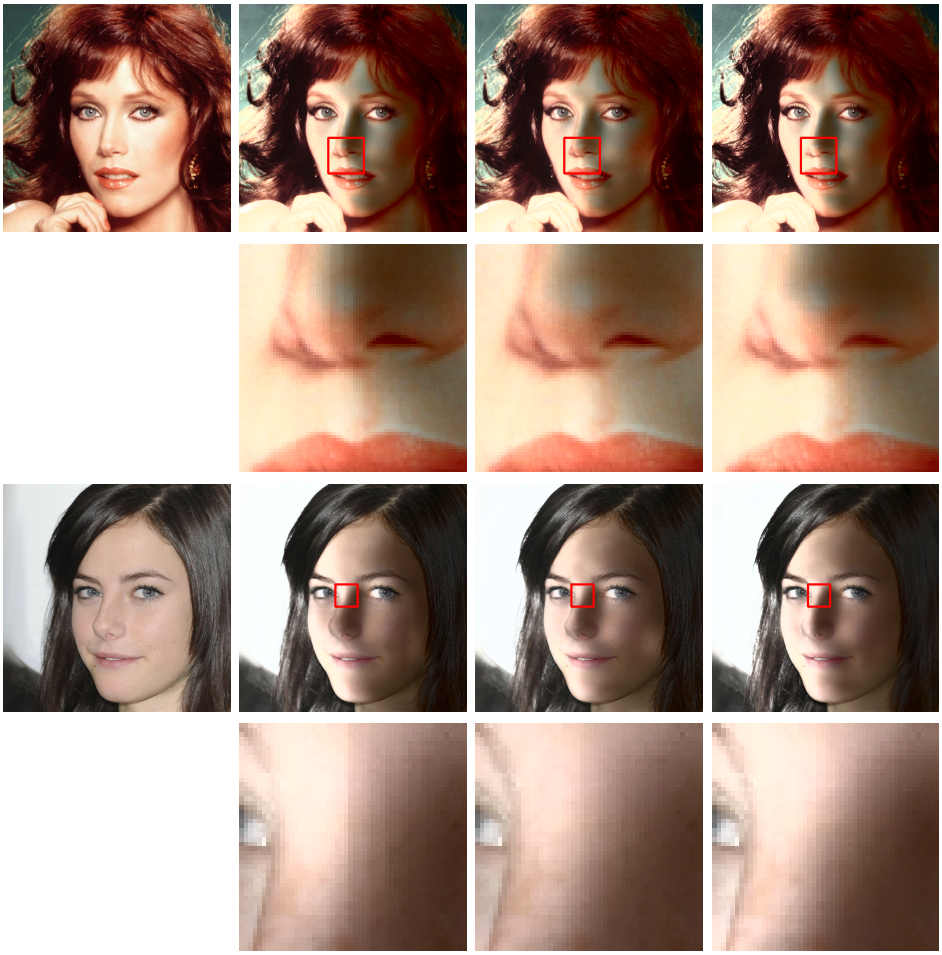
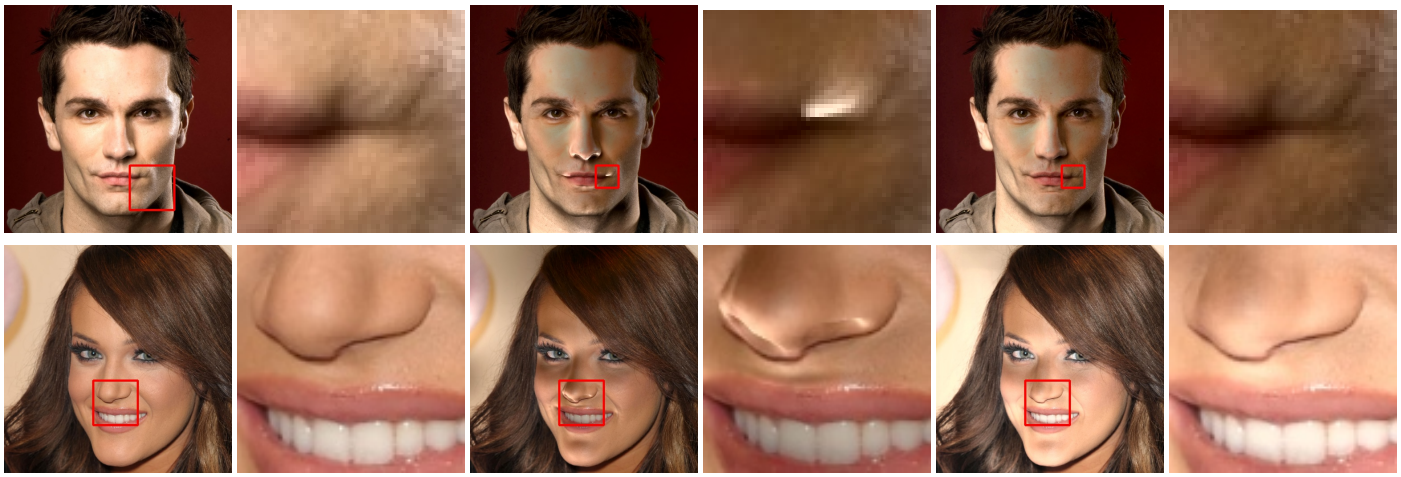


Figure 4.10: From left to right: first row: the input image, images generated using $\mathcal{L}_I$, $\mathcal{L}_I + \mathcal{L}_{GAN}$ and $\mathcal{L}_I + \mathcal{L}_{GAN} + \mathcal{L}_f$; second row: red rectangle region of the corresponding image in the first row. Note the edge in the middle of the noise generated using $\mathcal{L}_I$.

$\mathcal{L}_I + \mathcal{L}_{GAN} + \mathcal{L}_f$ (i.e. full model) in Table 4.2 and Figure 4.10. We notice that with GAN loss, the accuracy of our trained network is worse than the network trained without GAN loss. This is because the GAN loss is used to make the distribution of the generated images closer to that of the real images, i.e. improve the visual quality of the generated images. Adding the GAN loss may distract the network training process from being closer to the "ground truth" images. However, Figure 4.10 shows that with GAN loss, the artifacts on the nose part are alleviated compared with the network trained without GAN loss. This demonstrates the effectiveness the GAN loss in improving the visual quality. Adding a feature loss $\mathcal{L}_f$ significantly improves the accuracy of the images generated by our model, as shown in Table 4.2. We believe this is because our feature loss can force the generated images of the same identity to have similar latent features, thus, better preserving the identity information in the generated images. Moreover, Figure 4.10 shows that feature loss does not affect the quality of the generated images. As a result, we conclude that our full model can achieve a good balance between the accuracy and quality of the generated images.

4.5.3 Comparison with the Rendering Pipeline

Our proposed ARAP based normal refinement method improves the misalignments of face normals as discussed in Section 4.3. However, there are still cases in which the face normals cannot perfectly align with the face image, especially at the nose and the mouth region. These misalignments can cause ghost effects on the nose



(a) (b) (c) (d) (e) (f)

Figure 4.11: (a) original image, (c) results RI based rendering, (e) our results. (b), (d) and (f) show the red rectangle region of (a), (c) and (d) respectively. Note that the proposed method removes the ghost effect and artificial highlights.

	Si-MSE	Si-L_2
Li *et al* [115]	0.01322	0.3939
Shih *et al* [1]	0.01513	0.3415
Shu *et al* [116]	0.01384	0.3908
SfSNet [104]	0.00659	0.1593
Proposed Method	**0.00590**	**0.1444**

Table 4.3: Evaluation MultiPie Dataset

and artificial highlights at the corner of the mouth as shown in Figure 4.11. Though our training data contains images with these artifacts, Figure 4.11 shows that these artifacts can be avoided by the proposed method. This is because a deep learning based method can regularize the results, avoiding outlier effects.

4.5.4 Comparison with State-of-the-art Methods

In this section, we compare the proposed method with [1, 104, 115, 116], which can do portrait relighting. Since there is no ground truth lighting for images in the MultiPie dataset, we use an **image-based method** to evaluate our proposed

method and SfSNet [104] on this dataset, i.e., target lighting is extracted from the reference image and used for relighting. Both the proposed method and SfSNet [104] use their own lighting estimation method to extract the target lighting. [116] and [1] are two state-of-the-art portrait style transfer methods. They take two images $\mathbf{I}_s$ and $\mathbf{I}_f$ as input and transfer the style of $\mathbf{I}_s$ to $\mathbf{I}_f$. To get the relighted image using these two methods, we transfer $\mathbf{I}_s$ and $\mathbf{I}_f$ from *RGB* image to *Lab* image, and only apply their algorithm on the L channel. [115] is designed for general photo style transfer, and similar to [116] and [1], we use the L channel for portrait relighting.

We quantitatively compare the performance of our proposed method with these methods on the MultiPie dataset and show the results in Table 4.3. Our proposed method achieves the state-of-the-art results on both Si-MSE and Si-L_2 metric. This demonstrates that the proposed method can accurately generate relighted images under the target lighting condition. [116] and [1] both require accurately detected facial landmarks. The built-in facial landmark detector [123] of [116] fail to detect landmarks of 90 testing face images, thus we exclude those results when computing the Si-MSE and Si-L_2 for [116] in Table 4.3. We show some examples of relighted faces in the MultiPie dataset in figure 4.12.

We visually compare the proposed method with these state-of-the-art methods on DPR dataset and show results in Figure 4.5.4. Since the target lighting is known in this dataset, we apply a **SH-based method** to evaluate the proposed method and SfSNet [104]. We see that although SfSNet [104] can generate images under the correct lighting condition, their results are of low quality. Also, SfSNet [104] works on 128×128 images, which is too small for portrait relighting applications.

Figure 4.12: Visual results of the proposed method and state-of-the-art methods on MultiPie.

Furthermore, SfSNet cannot deal with the background correctly, making the results visually unpleasant. [1], [116] and [115] do not generate images under the correct lighting in these examples. These three methods are all reference image based, when the reference image is of low quality (e.g. occluded by hair region or sunglasses), they fail to understand the lighting correctly. As a result, they cannot generate images with accurate lighting conditions. We believe this is the common drawback for all methods which require a reference image as input. On the contrary, the proposed method and SfSNet [104] can directly take the target lighting as input, no referene image is required. Moreover, we notice that [1] and [116] cannot generate attached shadows on the nose, whereas the proposed method can generate very natural attached shadows.

From these experiments, we conclude that the proposed method outperforms the state-of-the-art methods both quantitatively and qualitatively, demonstrating the effectiveness of the proposed method.

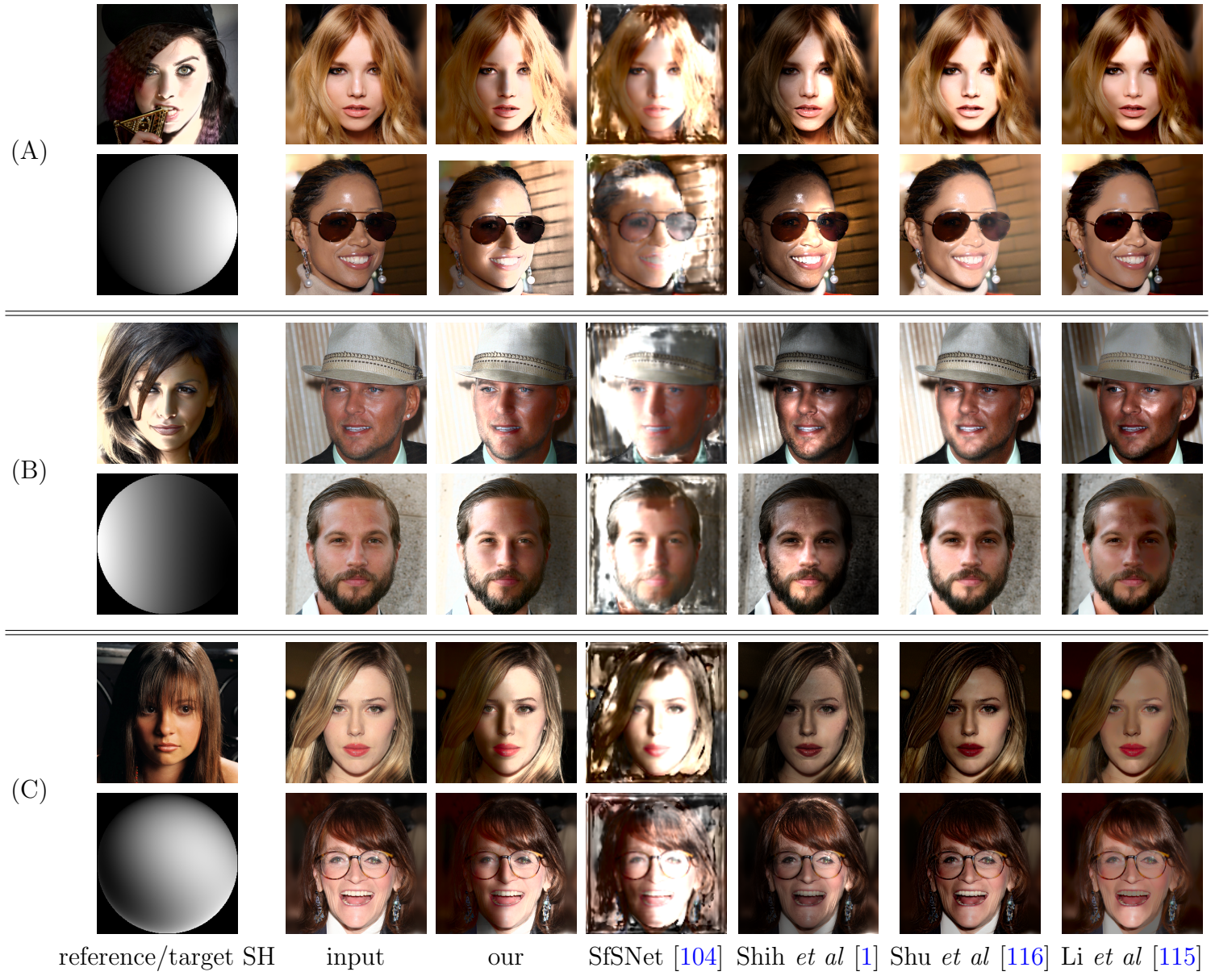


Figure 4.13: Qualitative comparison of the proposed method with state-of-the-art methods. The first column in (A) (B) and (C): first row is the reference image, second row is the target SH. The second column in (A), (B) and (C) shows the input image, third to seventh columns show the results of our method, SfSNet [104], Shih *et al* [1], Shu *et al* [116] and Li *et al* [115].

4.5.5 Results on challenging images

In Figure 4.5.5 we show our results on some challenging images. We notice that the proposed algorithm performs well on images with non-frontal faces, faces with occlusions and even faces with makeup.

4.5.6 Visual Results on High Resolution Images

We fine tune our network on 1024×1024 images and test the trained model on the flicker portrait dataset [1]. Figure 4.15 shows some of the results.

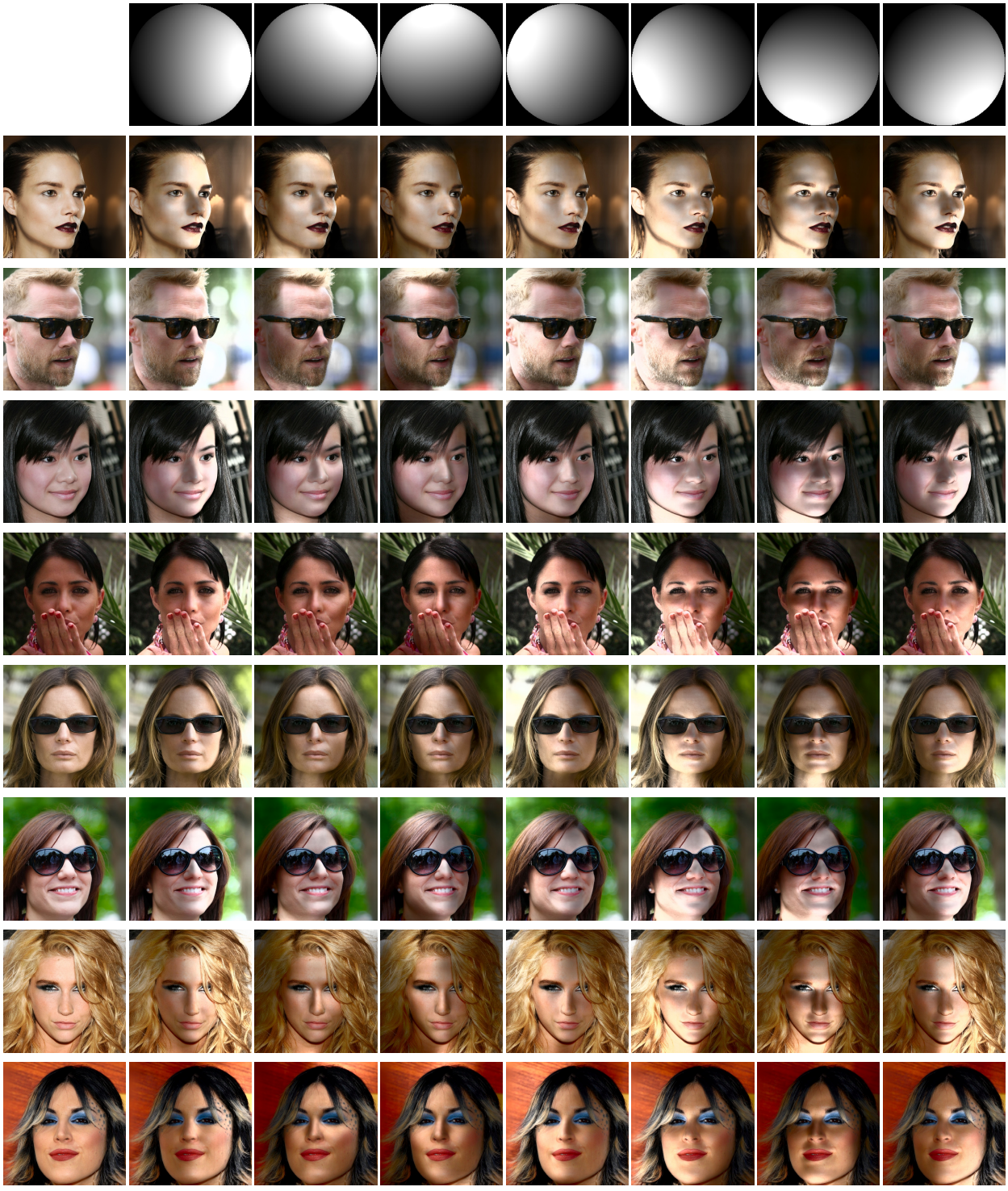


Figure 4.14: Some challenging examples. The first row shows the target SH lighting. The first column of the remaining rows shows the input image, the other columns show relighted images by the proposed method under target SH lighting. Our proposed method can deal with non-frontal faces, faces with occlusions and faces with makeup well.

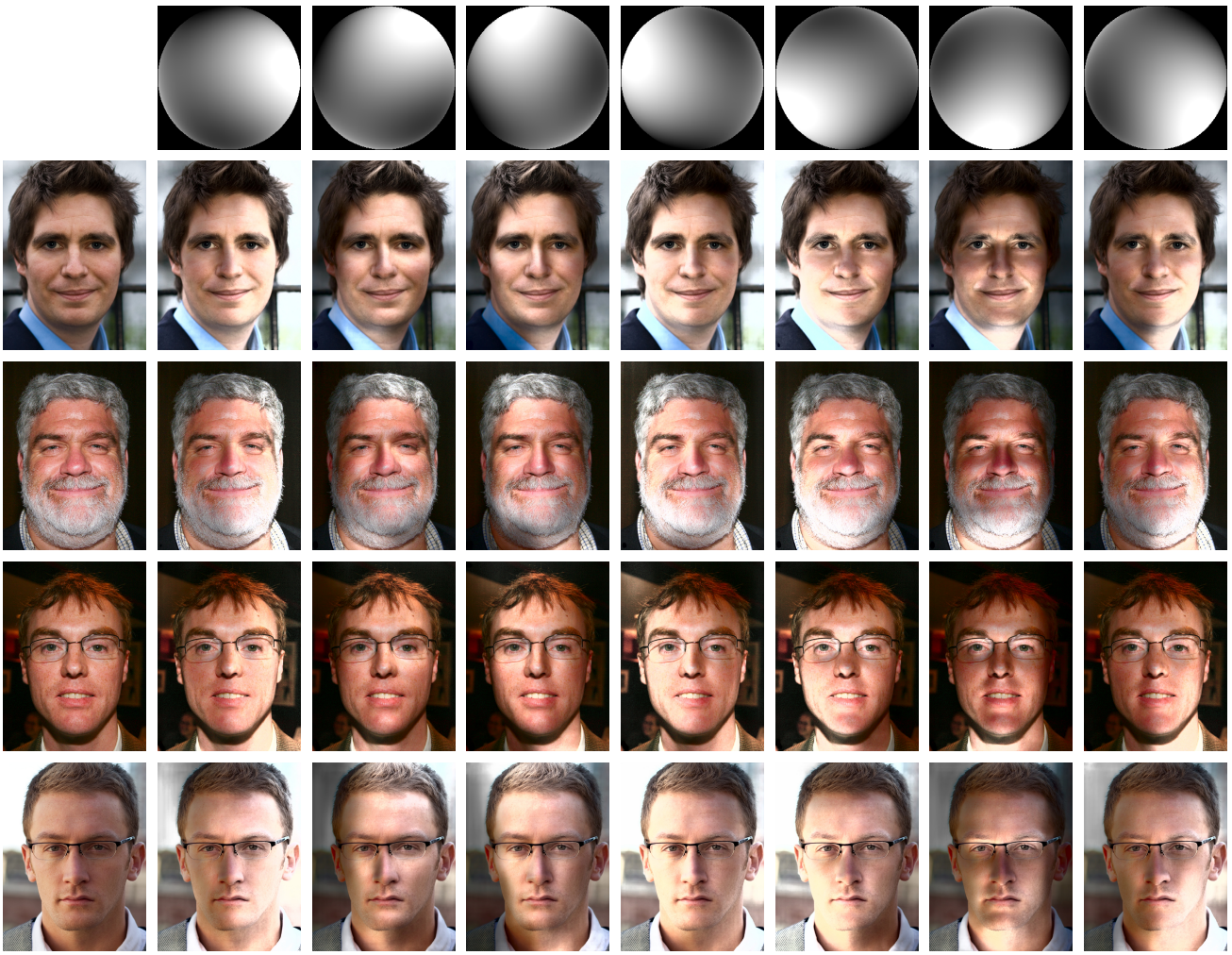
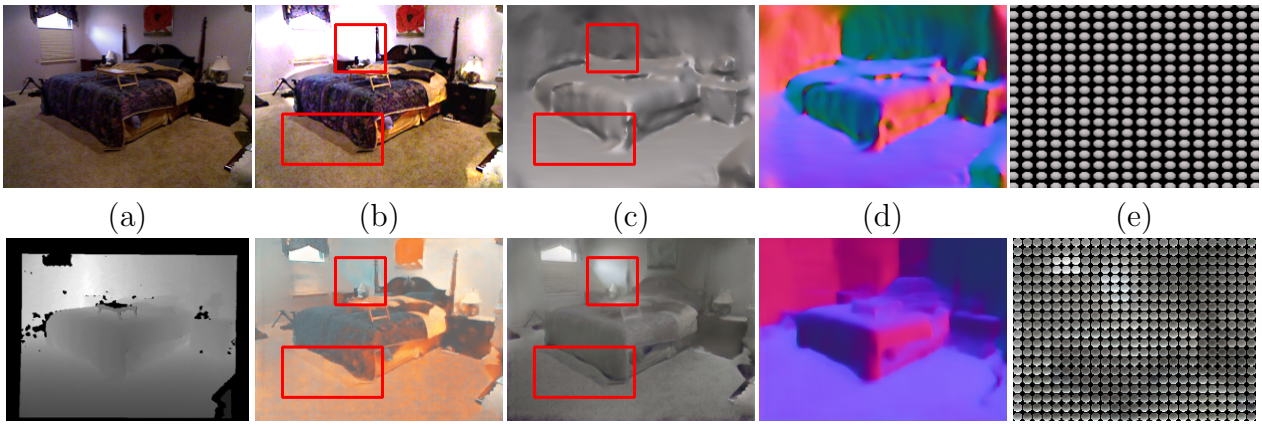


Figure 4.15: Results on flicker portrait dataset [1]. The first column shows the input image, the remaining columns show our relighted images using the corresponding target lighting.

4.6 Summary

In this chapter, we proposed an automatic single-image portrait relighting algorithm. We first apply a ratio image based relighting algorithm to generate a large scale, high quality, "in the wild" deep portrait relighting dataset. We then design a hourglass network that takes a source portrait image and a target lighting as input and generate a relighted portrait image. We train our network on the proposed DPR dataset, and find that deep network training can regularize the results, removing the artifacts in relighted images generated by ratio image based relighting. Moreover, our network can generate images with resolution 1024×1024. Extensive experiments demonstrate the effectiveness of the proposed method.

Chapter 5: GLoSH: Global-Local Spherical Harmonics for Intrinsic Image Decomposition



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 5.1: Top row: result of [2]. [2] takes an **RGB-D** image as input and predicts: (b) reflectance, (c) shading, (d) normal, (e) lighting. Bottom row: result of our method. It takes an **RGB** image as input and predicts: (g) reflectance, (h) shading, (i) normal and (j) lighting. The red boxes show that our algorithm correctly predicts cast shadows and highlights to shading while [2] incorrectly predict them to reflectance. Our lighting (j) captures local lighting variation better than (e) from [2].

In this chapter, we introduce our research work: intrinsic image decomposition from a natural scene. Different from conventional methods that decompose an image into reflectance and shading, the proposed method further decomposes shading into geometry (normal) and lighting. A Global-Local Spherical Harmonics is proposed to model the lighting of a natural scene. Due to the lack of real labels for reflectance,

shading and lighting, synthetic data is used for this problem. We demonstrate the power of our proposed lighting model and synthetic data through experiments.

This work [124] is in collaboration with Xiang Yu and David W. Jacobs.

5.1 Introduction

Understanding the physical world that produces an image is a core problem in computer vision. [125] first proposed to estimate the intrinsic scene characteristics from images, including range, orientation, reflectance and incident lighting. This is a notoriously difficult inverse problem as it is highly under-constrained. Moreover, we lack models of the physical components of the problem, such as lighting, that are both accurate and easy to use. Early works start with investigating the reflectance, shape and illumination of a single object [31,126], as the lighting for a single object is easier to model, for instance, by using a single set of low dimensional Spherical Harmonics [34,35]. The lighting of a natural scene, however, is much more complicated due to its spatial variation caused by shadow, inter-reflection and the presence of light sources in the scene. As a result, most works that address scenes have lumped normal and lighting together as shading, and try to recover that, known as intrinsic image decomposition.

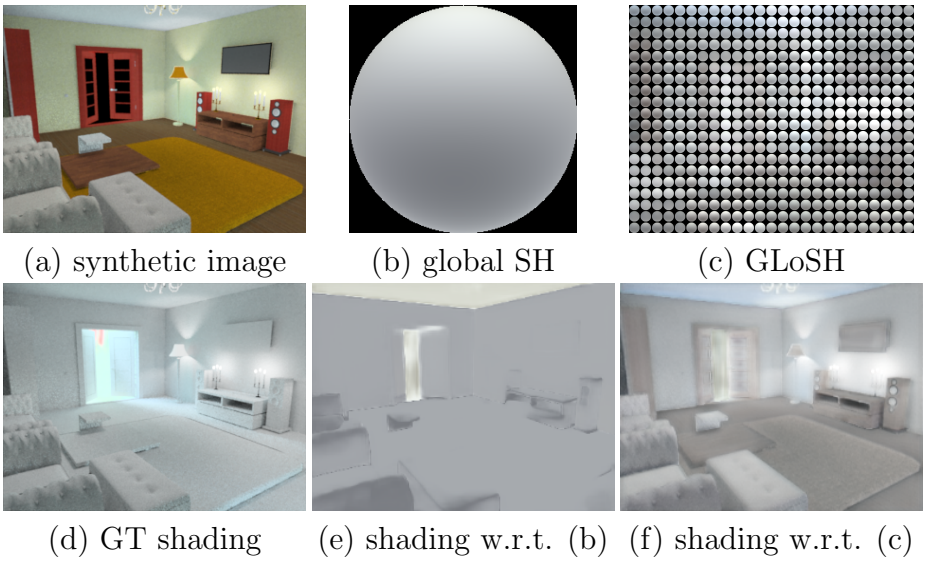
In this work, we propose a new representation of lighting for scenes, which allows us to disentangle lighting and surface normals, while also recovering reflectance. One way to model lighting is Spherical Harmonics (SHs) [34,35], which approximates the lighting with 9 low frequency components. While this works well for modeling

the lighting of small objects, such as faces [34], such a global lighting cannot capture the spatially varying lighting in a complex scene, as shown in Figure 2 (e). Allowing independent lighting in each pixel, however, creates too many degrees of freedom and would allow lighting variation alone to explain the image.

To overcome the problem, we propose a Global-Local SHs (GLoSH) lighting model. Our global SH represents the holistic lighting of the entire scene. On top of it, the local SHs, produced by the sum of global SH and local residual SHs, account for the spatial variation of the lighting. An L_2 regularization on the local residual SHs limits the effects of over-parameterization. Figure 5.2 (c) shows our GLoSH and Figure 5.2 (f) shows the reconstructed shading, which is much closer to ground truth than only using global SH.

Spherical Harmonics with arbitrary coefficients would represent lighting in a physically unrealistic way, if the lighting is negative in some directions. Nevertheless, enforcing non-negative SH lighting is not trivial. Existing methods either introduce many more parameters to constrain non-negative lighting [34] or require solving a semi-definite programming problem [127], which is difficult to directly incorporate with deep networks. In this work, we propose to sample the intensity of the lighting uniformly distributed on a sphere generated from the predicted SH. A non-negative loss is then defined on the sampled lighting. Our non-negative constraint is only applied to global SH, because practically the local residual SHs regularized by L_2 are not likely to change the sign of the lighting.

We apply a CNN to achieve an end-to-end coarse-to-fine solution. Training deep CNNs requires huge amounts of data and ground truth labels, and labeling



(a) synthetic image (b) global SH (c) GLoSH

(d) GT shading (e) shading w.r.t. (b) (f) shading w.r.t. (c)

Figure 5.2: Visualization of global SH modeling (b) and its reconstructed shading (e), comparing to our GLoSH (c) and its reconstructed shading (f). With GLoSH, clearly our method generates the shading much closer to ground truth.

images for reflectance and lighting is extremely difficult. Intrinsic Images in the Wild (IIW) [128] labels the relative darkness of the reflectance from pairs of pixels. Shading Annotations in the Wild (SAW) [129] labels constant shading regions, shadow boundaries and depth/normal discontinuities. However, these datasets only provide sparse labels and a limited number of images. Inspired by the recent success of synthetic data on computer vision applications, we propose to use the synthesized SUNCG dataset [130], in which ground truth reflectance, normal and shading can be easily determined, to pre-train the models. The pre-trained model is then further trained with real data in a self-supervised way.

To sum up, we propose a Global-Local Spherical Harmonics (GLoSH) lighting model, and apply a coarse-to-fine CNN structure to predict GLoSH together with reflectance and normal. The synthetic data pre-training and self-supervised training with real data lead to state-of-the-art performance across three real scene datasets,

IIW, SAW and NYUv2. The contributions of our work are the following.

1. We propose a GLoSH lighting model with global and local SHs, and a novel non-negative constraint to estimate physically realistic lighting.

2. To the best of our knowledge, under a single RGB image setting, we are the first to apply CNNs to jointly estimate reflectance, normal and lighting.

3. We propose a coarse-to-fine network that is compatible with our proposed global-local lighting model.

4. Our method achieves the best results on IIW reflectance, the second best on SAW shading, and strongly competitive performance on NYUv2 normal. Notice that the state-of-the-art methods only focus on one or two components, while our method jointly estimates reflectance, normal and lighting.

5.2 Related Work

Intrinsic Image Characteristics. We categorize the literature into two main streams: single object based and natural scene based methods. Researchers have long been studying the estimation of intrinsic image characteristics for a single object. For example, shape from shading [131, 132] focuses on recovering the shape assuming illumination and reflectance are known. Photometric stereo [133] estimates geometry from multiple images assuming known lighting. Recent progress in photometric stereo [126] can estimate geometry and lighting up to a bas-relief transformation [134]. [135–137] proposed to decompose a single object image into its reflectance and shading. [31] and [138] proposed to jointly estimate reflectance, shape and lighting from a single object image.

Estimating a natural scene is more difficult due to more complicated geometry and lighting. Recent studies [139, 140] show the capability of accurately estimating the scene geometry thanks to large scale training data and the success of deep learning. Some advanced methods [2, 128, 141–144] proposed optimization based approaches to decompose an image into reflectance and shading, where [2, 141, 142] require depth to be known. Most recent work [145–154] applies deep Convolutional Neural Networks (CNN) on this task and achieves impressive performance. [153] proposed to render realistic synthetic data and then use it to train the deep models and adapt to the real dataset. Our work follows a similar idea. However, we not only estimate reflectance and shading, but also further decompose the shading into normal and lighting.

Barron and Malik [2] first proposed to estimate reflectance, depth/normal and lighting with RGB-D images. They model the lighting at each pixel as a linear combination of eight sets of Spherical Harmonics. In contrast, we jointly estimate reflectance, surface normals and lighting from a single RGB image without depth, which is a much harder problem. Moreover, we propose our Global-Local SHs (GloSH) with a coarse-to-fine neural network to represent the lighting for each pixel, which accounts for not only the holistic lighting but also the local lighting variations.

Non-negative Spherical Harmonics. While using Spherical Harmonics lighting, one challenge is how to enforce lighting to be non-negative. [34] proposed to represent lighting using a non-negative linear combination of delta functions to solve this problem. One drawback of this method is that to have an accurate representation, a lot of delta functions are needed. [127] proved that the Toeplitz matrix of a non-negative SH is positive semi-definite. They proposed to solve a semi-definite programming (SDP) problem to enforce non-negative lighting. However, the SDP constraint is not obviously tractable to incorporate with deep training. In contrast, we formulate a non-negative lighting loss by sampling hundreds of points on a predicted lighting sphere, which is computationally efficient and fits into the network training smoothly.

5.3 Reflectance, Normal and Shading from a Single RGB Image

Intrinsic image decomposition assumes an image $\mathbf{I}$ to be the product of reflectance $\mathbf{R}$ and shading $\mathbf{S}$, *i.e*, $\mathbf{I} = \mathbf{R} \odot \mathbf{S}$, where $\odot$ represents an element-wise product. Most research studies focus on decomposing an image $\mathbf{I}$ into $\mathbf{R}$ and $\mathbf{S}$, where geometry and lighting remain entangled in shading. In our work, we propose to further decompose shading $\mathbf{S}$ into surface normal (*i.e*, geometry) $\mathbf{N}$ and lighting $\mathbf{L}$. Assuming $\mathbf{S} = \Psi(\mathbf{N}, \mathbf{L})$, an image $\mathbf{I}$ can be represented as

$$\mathbf{I} = \mathbf{R} \odot \Psi(\mathbf{N}, \mathbf{L}), \tag{5.1}$$

Ψ is a rendering function. Our target is to estimate $\mathbf{R}$, $\mathbf{N}$ and $\mathbf{L}$ given a single image $\mathbf{I}$.

5.3.1 GLoSH Lighting Modeling

While a single, global set of low-dimensional SH have been used to represent lighting of objects, this would be unable to capture the complex lighting conditions of a scene. On the other hand, estimating SHs for each pixel easily falls into over-parameterization. We propose a neural network based Global-Local Spherical Harmonics (GLoSH) model, where global SHs serve as the low frequency approximation of the lighting, and local SHs residuals count for spatial variation. A coarse-to-fine neural structure is designed to exactly execute the global and local lighting modeling.

5.3.1.1 Global and local Spherical Harmonics

Following [34, 35], we propose to use SH up to the second order, resulting in a 9 dimensional SH for each color channel. Denote the global SH as $\mathbf{L}_c \in \mathbf{R}^9$. $\mathbf{L}_c$ is predicted from our coarse level network

As revealed in Figure 5.2, based only on the global SH $\mathbf{L}_c$, the shading is far from satisfactory, lacking much spatial variation. To better model the spatial variation of the lighting, we predict local SH residuals for each pixel in a fine level network. Our local SH is then formulated as global SH with the local SH residuals:

$$\mathbf{L}_f = \mathbf{L}_c + \delta\mathbf{L}_f, \tag{5.2}$$

where $\delta\mathbf{L}_f$ represent the local residual SH predicted by a fine scale network.

5.3.1.2 Non-negative Constraints on SH

Physically realistic lighting requires non-negative SH lighting, which previous work [2, 31] do not properly consider. To enforce the non-negative SH lighting, we propose a simple yet effective constraint on SH. According to [34], given a SH coefficient $\mathbf{L}_c$, the lighting intensity at a direction (θ, ϕ) is a function of $\mathbf{L}_c$, *i.e*, $f_{\mathbf{L}}(\mathbf{L}_c, \theta, \phi)$. A non-negative lighting means $f_{\mathbf{L}}(\mathbf{L}_c, \theta, \phi) \geq 0, \forall 0 \leq \theta \leq \pi, 0 \leq \phi \leq 2\pi$. Based on this, we uniformly sample the value of the function $f_{\mathbf{L}}$ on a unit sphere and constrain all the sampled values to be non-negative. The non-negative

loss function is thus defined as

$$\mathcal{L}_{Lc} = \frac{1}{K}\sum_{i=1}^{K} \min(0, f_{\mathbf{L}}(\mathbf{L}_c, \theta_i, \phi_i))^2, \tag{5.3}$$

K is the number of directions sampled from the sphere. We apply the above non-negative constraint to global SH. We further apply the L_2 regularization over the local residual SH:

$$\mathcal{L}_{Lf} = \|\delta \mathbf{L}_f\|_2^2. \tag{5.4}$$

This regularization penalizes their L2 norm, encouraging the local lighting to not vary too much from the global lighting.

Our experiments demonstrate that Equation 5.4 together with non-negative constraint on global SH is sufficient to guarantee the non-negative lighting for local SH.

5.3.2 Coarse-to-fine Network Structure

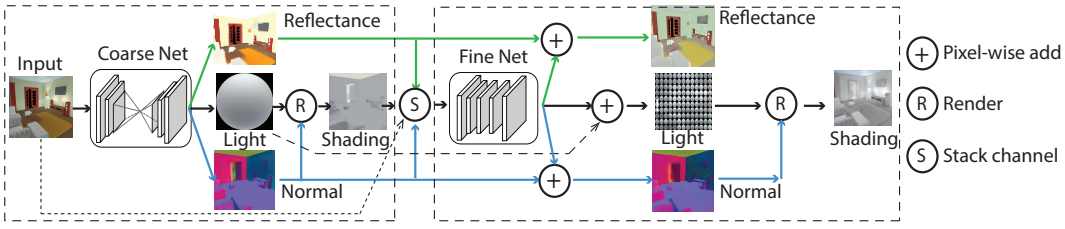


Figure 5.3: Our coarse-to-fine network structure. The coarse net predicts the first level reflectance, lighting and surface normal, of which the latter two form the shading. The fine net takes the previous stacked output as input and predicts the residual of reflectance, lighting and surface normal. The final reflectance, lighting and normal are recovered by adding the predicted residual with the first level results.

To exactly match the proposed GLoSH lighting modeling, we design a coarse-to-fine network structure shown in Figure 5.3. The coarse network is defined as a hourglass network [155]. It takes an image $\mathbf{x} \in \mathbb{R}^{64\times64\times3}$ as input and predicts reflectance $\mathbf{R}_c \in \mathbb{R}^{64\times64\times3}$, normal $\mathbf{N}_c \in \mathbb{R}^{64\times64}$, and global SH $\mathbf{L}_c \in \mathbb{R}^9$. Shading $\mathbf{S}_c \in \mathbb{R}^{64\times64\times3}$ can be constructed by a simple rendering function.

$$\mathbf{S}_c = \Psi(\mathbf{N}_c, \mathbf{L}_c) \tag{5.5}$$

The fine scale network is designed with fully convolutional structures. It takes $\mathbf{x} \in \mathbb{R}^{128\times128\times3}$, upsampled $\mathbf{R}_c \in \mathbb{R}^{128\times128\times3}, \mathbf{N}_c \in \mathbb{R}^{128\times128}, \mathbf{S}_c \in \mathbb{R}^{128\times128\times3}$ as input and predicts residual maps. The recovered local reflectance, normal and local SHs are:

$$\begin{aligned}
\mathbf{R}_f &= \mathbf{R}_c + \Phi_f^{\mathbf{R}_f}(\mathbf{R}_c, \mathbf{N}_c, \mathbf{L}_c), \\
\mathbf{N}_f &= \mathbf{N}_c + \Phi_f^{\mathbf{N}_f}(\mathbf{R}_c, \mathbf{N}_c, \mathbf{L}_c), \\
\mathbf{L}_f &= \mathbf{L}_c + \Phi_f^{\mathbf{L}_f}(\mathbf{R}_c, \mathbf{N}_c, \mathbf{L}_c),
\end{aligned} \tag{5.6}$$

where, $\Phi_f^{\mathbf{R}_f}$, $\Phi_f^{\mathbf{N}_f}$ and $\Phi_f^{\mathbf{L}_f}$ represent the fine level network for reflectance, normal and lighting respectively. The fine scale shading is calculated by Equation 5.5 $\mathbf{S}_f = \Psi(\mathbf{N}_f, \mathbf{L}_f)$. The fine scale network structure can be recurrently applied to a finer scale. Our full model is defined to have three scales, which can predict reflectance, normal and lighting with resolution 256×256. Please refer to Section 5.4 for more details of the network structure.

5.3.3 Supervision on Training

It is difficult to obtain dense accurate ground truth annotation for reflectance, normal and lighting. We thus leverage the rendered synthetic data for the supervised pre-training. The pre-trained network is then fine-tuned using sparsely annotated real data (IIW [128], SAW [129] and NYUv2 [156]) in a self-supervised way, *i.e*, applying the pre-trained model to provide pseudo ground truth labels for fine-tuning.

5.3.3.1 Reflectance

In the pre-training stage, we directly apply the ground truth reflectance to guide the training in a fully supervised way, where L_1 loss is applied as shown in Equation 5.7.

$$\mathcal{L}_{R1} = \|\mathbf{R} - \mathbf{R}^*\|_1 + \|\nabla\mathbf{R} - \nabla\mathbf{R}^*\|_1. \tag{5.7}$$

$\mathbf{R}$ is the predicted reflectance and $\mathbf{R}^*$ is the corresponding ground truth. Moreover, similar to [153], we add supervision to the gradient of the reflectance to encourage the predicted reflectance to be piece-wise smooth.

For real data, there is no dense annotation for either reflectance, normal or lighting. Instead, IIW [128] provides sparse ordinal reflectance judgments. Given a pair of reflectances $\mathbf{R}_1$ and $\mathbf{R}_2$, the label indicates whether $\mathbf{R}_1$ is darker than (lighter than or equal to) $\mathbf{R}_2$ (demoted as $J = 1$, $J = -1$ and $J = 0$ respectively) with a confidence score w. We use the WHDR hinge loss proposed in [146] as the loss for

reflectance in real images:

$$\mathcal{L}_R(\mathbf{R}_1, \mathbf{R}_2, J) = \tag{5.8}$$

$$w \begin{cases} \max\left(0, \frac{R_1}{R_2} - \frac{1}{1+\delta+\xi}\right) & \text{if} \quad J = 1 \\ \max\left(0, \begin{cases} \frac{1}{1+\delta-\xi} - \frac{R_1}{R_2} \\ \frac{R_1}{R_2} - (1+\delta-\xi) \end{cases}\right) & \text{if} \quad J = 0 \\ \max\left(0, (1+\delta+\xi) - \frac{R_1}{R_2}\right) & \text{if} \quad J = -1 \end{cases}$$

We set $\delta = 0.12$ and $\xi = 0.08$ during training as in [146]. Notice that the above loss is not symmetric, *i.e*, $\mathcal{L}_{R_i}(\mathbf{R}_1, \mathbf{R}_2, J) \neq \mathcal{L}_R(\mathbf{R}_2, \mathbf{R}_1, -J)$. We thus adapt the above loss and define the modified WHDR loss as:

$$\mathcal{L}_{R2} = \mathcal{L}_R(\mathbf{R}_1, \mathbf{R}_2, J) + \mathcal{L}_R(\mathbf{R}_2, \mathbf{R}_1, -J) \tag{5.9}$$

5.3.3.2 Normal

The ground truth normal for synthetic data and part of the real data (NYUv2) are available. For those data in which ground truth normals are available, we define the loss as

$$\mathcal{L}_N = -\mathbf{N}^T\mathbf{N}^* + \|\nabla\mathbf{N} - \nabla\mathbf{N}^*\|_1 \tag{5.10}$$

Similar to reflectance regularization in Equation 5.7, we further apply the first order derivative smoothness term to encourage the normal to be piece-wise continuous.

5.3.3.3 Shading

There is no supervision for lighting. The non-negative constraint and the L_2 regularization are all unsupervised losses. Applying rendering to generate shading $\mathbf{S} = \Psi(\mathbf{N}, \mathbf{L})$ from normal and lighting, we use the supervision on shading and normal discussed in Sec. 5.3.3.2 to indirectly supervise the lighting. The supervised signal for shading is similar to that of reflectance:

$$\mathcal{L}_{S1} = \|\mathbf{S} - \mathbf{S}^*\|_1 + \|\nabla\mathbf{S} - \nabla\mathbf{S}^*\|_1 \tag{5.11}$$

where $\mathbf{S}$ and $\mathbf{S}^*$ are predicted shading and its ground truth.

For real images, SAW [129] provides annotation for smooth shading regions and shadow boundaries. We thus apply the same loss as in [153] for the shading:

$$\mathcal{L}_{S2} = \lambda_{cs}\mathcal{L}_{constant-shading} + \mathcal{L}_{shadow} \tag{5.12}$$

where $\lambda_{cs} = 10$ and $\mathcal{L}_{constant-shading}$ and $\mathcal{L}_{shadow}$ are the loss for constant shading region and shadow boundary defined in [153].

5.4 Network Structure

Overall our framework can be divided into three scales. We employ a hourglass network [155] for the first scale network and a fully convolutional structure for second and third scale networks. Details are discussed in the following for the three scale network structures.

Network of First Scale. The first scale network takes a 64×64 image as input and predicts 64×64 reflectance $\mathbf{R}_c$, normal $\mathbf{N}_c$, and a single global Lighting $\mathbf{L}_c$. Shading $\mathbf{S}_c$ can be constructed based on $\mathbf{N}_c$ and $\mathbf{L}_c$. The branches used to predict reflectance and normal have the same hourglass network structure [155], which is illustrated in Figure 5.4 (a). The branch to predict global SH is shown in Figure 5.4 (b). The green blocks are shared by all the three branches.

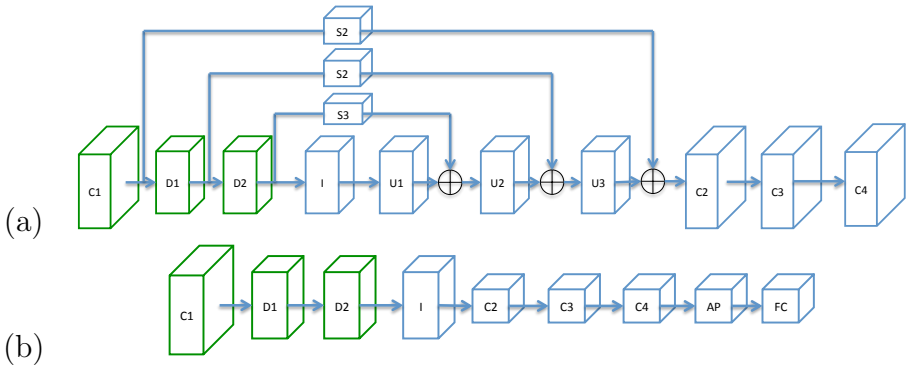


Figure 5.4: (a) shows the network structure to predict reflectance and normal and (b) shows the network structure for predicting global SH.

In Figure 5.4, C1, C2, C3, and C4 represent convolutional layers. D1, D2, I, U1, U2, U3, S1, S2, S3 are residual blocks defined in [6]. AP is an average pooling layer and FC represents a fully-connected layer. Each of the convolution layers is followed by batch normalization [157] and ReLU except for the output layer.

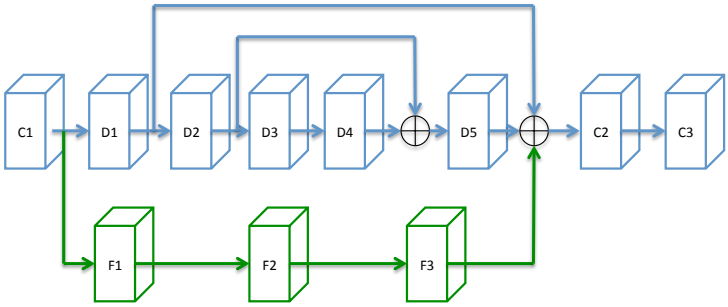


Figure 5.5: Network structure to predict reflectance, normal and lighting at second and third scale.

Table 5.4 shows the detailed definition of the block in Figure 5.4. Since we predict 9 Spherical Harmonics for each channel of the global SH, the number of output channels for global SH is 27.

		C1	C2	C3	C4	D1	D2	I	U1	U2	U3	S1	S2	S3	FC
(a)	input channel number	3	64	32	3	64	128	256	248	256	128	64	128	256	-
	output channel number	64	32	3	3	128	256	248	256	128	64	64	128	256	-
	filter size	5	3	3	3	3	3	3	3	3	3	3	3	3	-
	input feature size	64	64	64	64	64	32	16	8	16	32	64	32	16	-
	output feature size	64	64	64	64	32	16	8	16	32	64	64	32	16	-
(a)	input channel number	3	16	64	-	64	128	256	-	-	-	-	-	-	128
	output channel number	64	64	128	-	128	256	16	-	-	-	-	-	-	27
	filter size	5	3	3	-	3	3	3	-	-	-	-	-	-	-
	input feature size	64	8	4	-	64	32	16	-	-	-	-	-	-	-
	output feature size	64	4	2	-	32	16	8	-	-	-	-	-	-	-

Table 5.1: Details of each block in our network. (a) shows the details about each block in Figure 5.4 (a) and (b) shows the details about each block in Figure 5.4 (b).

Network of Second and Third Scale. Our second and third scale network has the same network structure. Our second network works on images with resolution 128×128 and our third network works on images with resolution 256×256. We define the network to predict residuals of reflectance, normal and lighting using separate networks with no shared layers.

The network structure is illustrated in Figure 5.5. C1, C2, C3, F1, F2, F3

are convolutional layers, D1, D2, D3, D4, D5 are residual blocks defined in [6]. Each convolutional layer is followed by a Batch Normalization layer [157] and ReLU except for the output layer.

Table 5.4 (a) shows the detailed definition of each block for networks used to predict residual reflectance and normal. Table 5.4 (b) shows the detailed definition of each block for networks used to determine local residual SHs. For reflectance, we concatenate the image and the upsampled reflectance from the coarse network as input, so the number of input channels is 6. Similarly, we concatenate the image and upsampled normals from the coarse network as input for the network for normals. The number of input channels is also 6. For the network used to predict local SHs, we concatenate the image, upsampled normal, reflectance and shading as input. As a result, the number of input channels for local SHs prediction is 12. Since we predict the color SH for each pixel, the number of output channels is 27.

		C1	C2	C3	D1	D2	D3	D4	D5	F1	F2	F3
(a)	input channel number	6	32	3	32	32	32	64	32	16	32	32
	output channel number	16	3	3	32	32	64	32	32	32	32	32
	filter size	5	3	3	3	3	3	32	32	1	1	1
	dilation size	1	1	1	1	2	4	1	1	1	1	1
(b)	input channel number	12	32	3	32	32	32	64	32	16	32	32
	output channel number	16	3	27	32	32	64	32	32	32	32	32
	filter size	5	3	3	3	3	3	32	32	1	1	1
	dilation size	1	1	1	1	2	4	1	1	1	1	1

Table 5.2: Details about each block in our second and third scale network shown in Figure 5.5. (a) shows the detailed structure for the network used to predict reflectance and normal. (b) shows the detailed structure for the network used to predict lighting.

5.5 Implementation Details

Pre-training on Synthetic Data: We first train our network using the SUNCG dataset with synthesized ground truth normal, reflectance, and shading. The loss to train our network on synthetic data is

$$\mathcal{L}_s = \lambda_{sR}\mathcal{L}_{R1} + \lambda_{sS}\mathcal{L}_{S1} + \lambda_{sN}\mathcal{L}_N + \lambda_{Lc}\mathcal{L}_{Lc} + \lambda_{Lf}\mathcal{L}_{Lf} \tag{5.13}$$

where $\mathcal{L}_{R1}$, $\mathcal{L}_{S1}$, $\mathcal{L}_N$, $\mathcal{L}_{Lc}$ and $\mathcal{L}_{Lf}$ are losses for reflectance, shading, normal, global and local residual lighting defined above, and λ_{sR}, λ_{sS}, λ_{sN}, λ_{Lc} and λ_{Lf} are their corresponding weights. We set $\lambda_{sR} = \lambda_{sS} = \lambda_{sN} = \lambda_{Lc} = 1$ and $\lambda_{Lf} = 0.2$. Our coarse-to-fine network is trained step by step using the Adam [122] optimizer with initial learning rate 0.001 and weight decay 0.

Fine-tuning on Real Data: Due to the lack of annotation from real datasets, we use the rendered SUNCG dataset as supervision, with the loss denoted as $\mathcal{L}_r^{cg}$. In addition, we apply our network trained on synthetic data to predict reflectance, shading and normal of real images and use the results as pseudo supervision (self-supervision), with the loss denoted as $\mathcal{L}_r^{ss}$.

$$\begin{aligned}
\mathcal{L}_r^{cg} &= \lambda_{sR}^{cg}\mathcal{L}_{R1} + \lambda_{sS}^{cg}\mathcal{L}_{S1} + \lambda_{sN}^{cg}\mathcal{L}_N + \lambda_{Lc}^{cg}\mathcal{L}_{Lc} + \lambda_{Lf}^{cg}\mathcal{L}_{Lf}, \\
\mathcal{L}_r^{ss} &= \lambda_{rR}^{ss}\mathcal{L}_{R1} + \lambda_{rS}^{ss}\mathcal{L}_{S1} + \lambda_{rN}^{ss}\mathcal{L}_N + \lambda_{Lc}^{ss}\mathcal{L}_{Lc} + \lambda_{Lf}^{ss}\mathcal{L}_{Lf}
\end{aligned} \tag{5.14}$$

where we set $\lambda_{sR}^{cg} = \lambda_{sS}^{cg} = \lambda_{Lc}^{cg} = \lambda_{Lf}^{cg} = 1$, $\lambda_{sN}^{cg} = 10$, $\lambda_{rS}^{ss} = \lambda_{rN}^{ss} = 5$, $\lambda_{Lf}^{ss} = \lambda_{Lc}^{ss} = 1$

and $\lambda_{rR}^{ss} = 0.1$. Our loss defined on the annotation and ground truth of IIW, SAW and NYUv2 is:

$$\mathcal{L}_r^o = \lambda_{rR}^o \mathcal{L}_{R2} + \lambda_{rS}^o \mathcal{L}_{S2} + \lambda_{rN}^o \mathcal{L}_N \tag{5.15}$$

where $\lambda_{Lc}^o = \lambda_{rN}^o = 10$, $\lambda_{rS}^o = 1$. Inspired by [128], we introduce the L_2 regularization to achieve a reasonable color for reflectance.

$$\mathcal{L}_r^c = \| \frac{\mathbf{R}}{\frac{1}{3}\sum_c \mathbf{R}^c} - \frac{\mathbf{I}}{\frac{1}{3}\sum_c \mathbf{I}^c} \|_1 \tag{5.16}$$

where $\mathbf{R}$ and $\mathbf{I}$ are predicted reflectance and input image, and $\mathbf{R}^c$ and $\mathbf{I}^c$, $c \in \{R, G, B\}$ denote the color channel of $\mathbf{R}$ and $\mathbf{I}$. Importantly, a reconstruction loss is further introduced to guarantee the predicted reflectance, normal and lighting preserve the input's characteristics.

$$\mathcal{L}_r^{rc} = \|\mathbf{I}_i - \mathbf{R}_i \odot \mathbf{S}_i\|_2 \tag{5.17}$$

The overall loss that we apply to fine-tune our network on real images is:

$$\mathcal{L}_r = \mathcal{L}_r^{cg} + \mathcal{L}_r^{ss} + \mathcal{L}_r^o + \mathcal{L}_r^c + \lambda_r^{rc} \mathcal{L}_r^{rc} \tag{5.18}$$

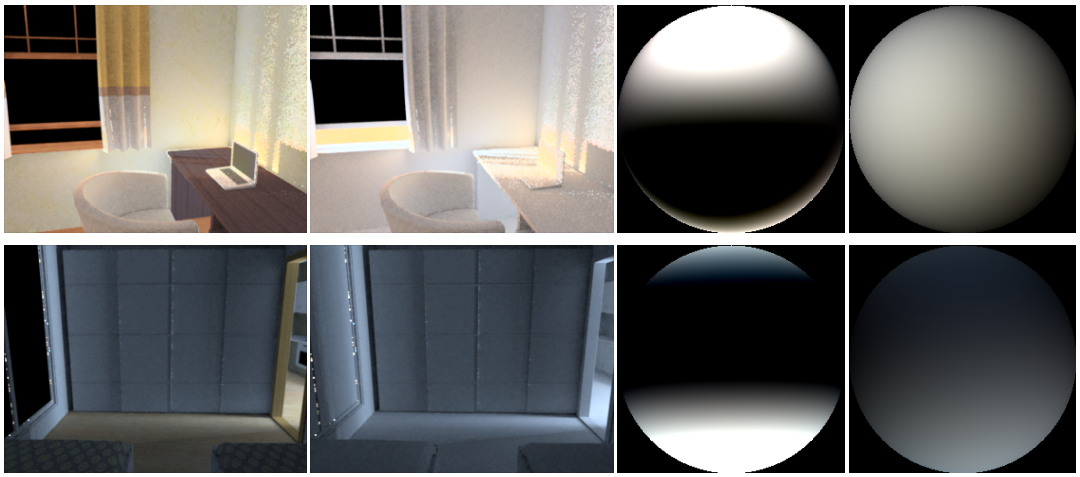
where $\lambda_r^{rc} = 0.1$. The coarse-to-fine network is fine tuned scale by scale. The Adam optimizer with learning rate 0.0005 and weight decay 0.00001 is used for fine-tuning.

5.6 Experiments

In this section, we introduce the synthetic dataset that we create for pre-training and the public real datasets. Then we compare to Barron and Malik [2], who first proposed to predict reflectance, normal and lighting from an RGB-D image. Further, we compare to the state-of-the-art intrinsic image decomposition methods to indicate the overall advantage of our method. An ablation study is then carried out to demonstrate the contribution of each of our proposed modules.

5.6.1 Datasets

Synthetic Dataset: we make use of the SUNCG dataset [158] to generate synthetic data. It contains $568,793$ images rendered using Mitsuba [159] and their corresponding ground truth surface normals, depths, semantic labels and object boundaries. Since our task also requires ground truth reflectance and shading, we re-render $58,949$ images of SUNCG using the multi-channel renderer of Mitsuba. We further split the images into a set of 51,507 training images and a set of 7,442 validation images. Instead of directly rendering images, we render shading by setting all the materials to diffuse and the reflectance to be 1. Then the final image is $\mathbf{I} = \mathbf{R} \odot \mathbf{S}$. We believe rendering in this way has two main advantages: (1) The generated images strictly follow the assumption of intrinsic image decomposition. (2) The pixel value of ground truth shading has bounded range which makes data preparation easier. Though the rendered images do not contain non-diffuse effects of the material, our experiments show that this does not degrade the performance.



(a) (b) (c) (d)

Figure 5.6: (a) synthetic images, (b) shading images, (c) and (d) are lighting predicted by training the network without and with non-negative constraint respectively.

	[2]	GLoSH SUNCG	GLoSH SUNCG + real
MSE	0.098	0.038	**0.032**

Table 5.3: SH lighting Evaluation on SUNCG synthetic data.

Public Real Datasets: we use IIW [128], SAW [129] and NYUv2 [156] as real data for training and testing. More specifically, SAW is a combination of IIW and NYUv2 (3761 images from IIW and 381 images from NYUv2 with ground truth normals). The real dataset we use is the same as [153] in addition to ground truth normals from NYUv2. We strictly follow the train/val/test splitting strategy of [153].

5.6.2 Spherical Harmonics Lighting Evaluation

Quantitative comparison to [2]. We compare to [2] as they also propose a lighting model to jointly predict reflectance, normal and lighting of a natural scene. Notice that [2] uses RGB-D images, which simplifies the problem.

Lighting for real data is hard to obtain. We instead evaluate the shading from

the SUNCG synthetic data by fixing the surface normal from ground truth, at which we can indirectly evaluate the SH lighting. We calculate the per-pixel Mean squared error (MSE) of the reconstructed shading w.r.t. ground truth shading and show the results in Table 5.3. Our method shows a significant advantage over [2] and the real data self-supervision provides a further performance boost. We also evaluate the shading of [2] on NYUv2 dataset using the AP challenge metric proposed by [153]. They achieve **90.38**% shading accuracy, while under the same setup, our method achieves **95.36**%. We believe all these evidences prove the proposed method can predict much more accurate lighting than [2].

Qualitative comparison to [2]. Figure 5.7, 5.8 and 5.9 compare their visual results with ours. The red rectangles in reflectance and shading images show that [2] mistakenly decomposes cast shadow into reflectance instead of shading. We believe the limited number of SH basis in their method prevents them from modeling the spatial variation of the lighting well, resulting in a lack of ability to model cast shadows.

Non-negative lighting: [127] proved that a SH represents non-negative lighting if its Toeplitz matrix is positive semi-definite. We use their proposed method to evaluate the effectiveness of our non-negative constraint. We train our coarse scale network with and without the proposed non-negative constraint, *i.e*, Equation (5.3), and then test on the validation set of our synthetic SUNCG data. Without the proposed non-negative constraint, the percentage of global SH that represents negative lighting is **13.39**%. It decreases drastically to **1.09**% with this constraint. Figure 5.6 visualizes the predicted lighting with and without the non-negative constraint. After

Method	Avg. (°)↓	Med. (°)↓	11.25° ↑	22.5° ↑	30° ↑
[158]	**27.90**	21.29	26.76	52.21	**63.75**
Ours	28.63	**21.05**	**27.68**	**52.42**	62.87

Table 5.4: Surface normal evaluation on NYUv2. Average (Avg.) and Median (Med.) show the average and median angular error, smaller values are the better. 11.25°, 22.5° and 30° shows the percentage of normals with angular error smaller than 11.25°, 22.5° and 30°, higher values are better.

fine-tuning on real data, the global SH that represents negative lighting is reduced to **0**% and there is only one image that contains negative local lighting.

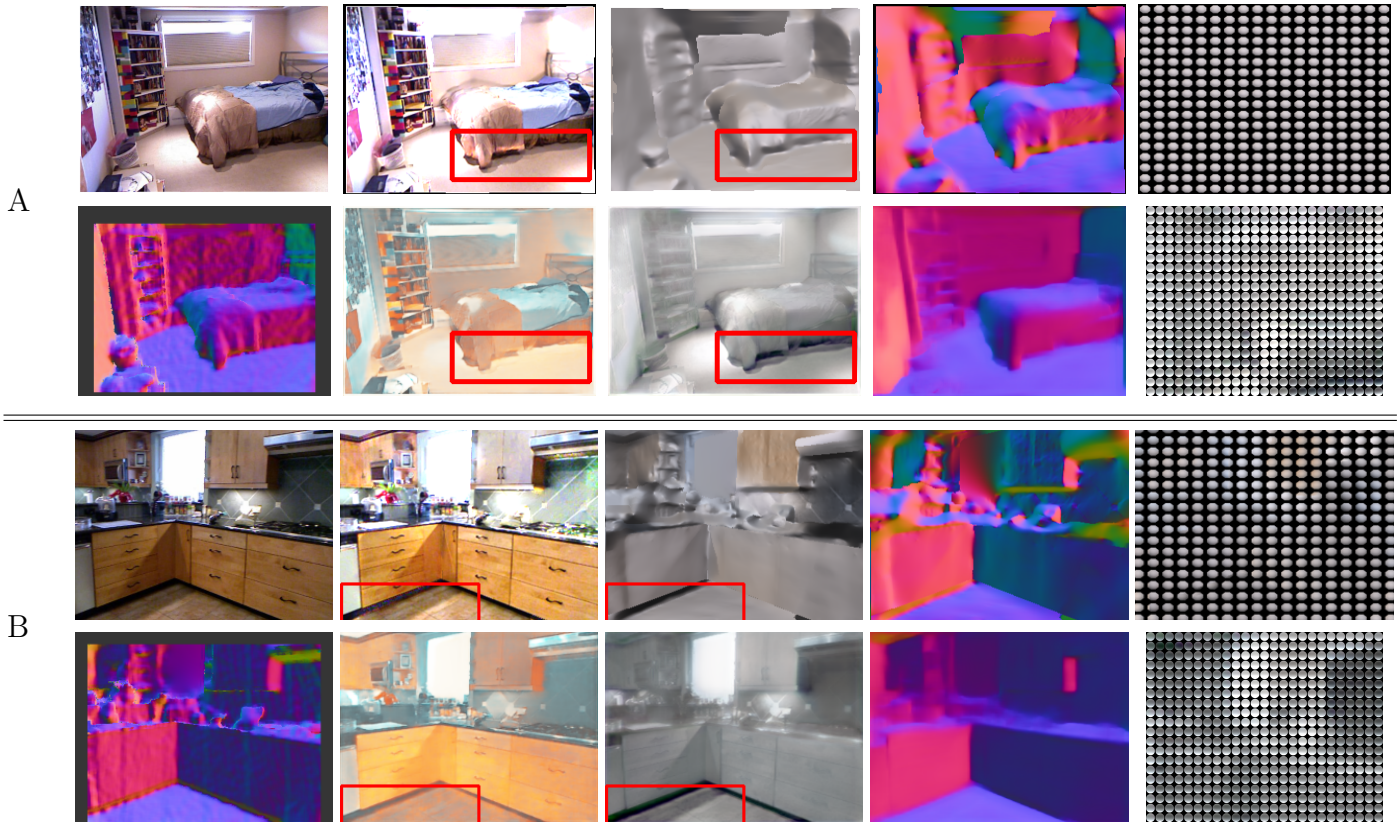


Figure 5.7: Comparison with [2]. First row of A and B from left to right: input image, reflectance by [2], shading by [2], normal by [2], lighting by [2]. Second row of A and B from left to right: ground truth normal; reflectance, shading, normal, global SH and local SHs by the proposed method.

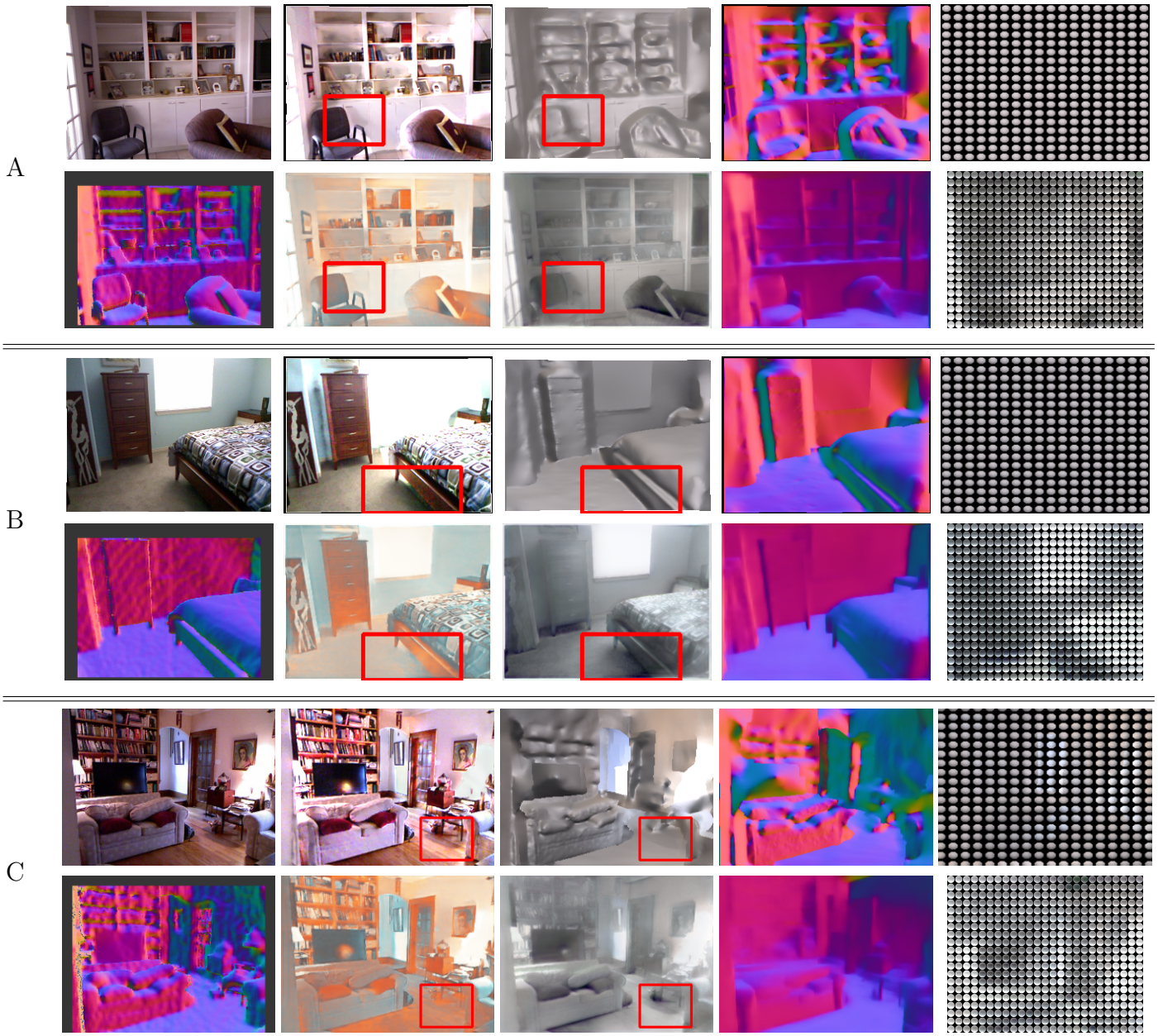


Figure 5.8: Comparison with [2]. First row of A, B, C and D from left to right: input image, reflectance by [2], shading by [2], normal by [2], lighting by [2]. Second row of A, B, C and D from left to right: ground truth normal; reflectance, shading, normal, global SH and local SHs by the proposed method.

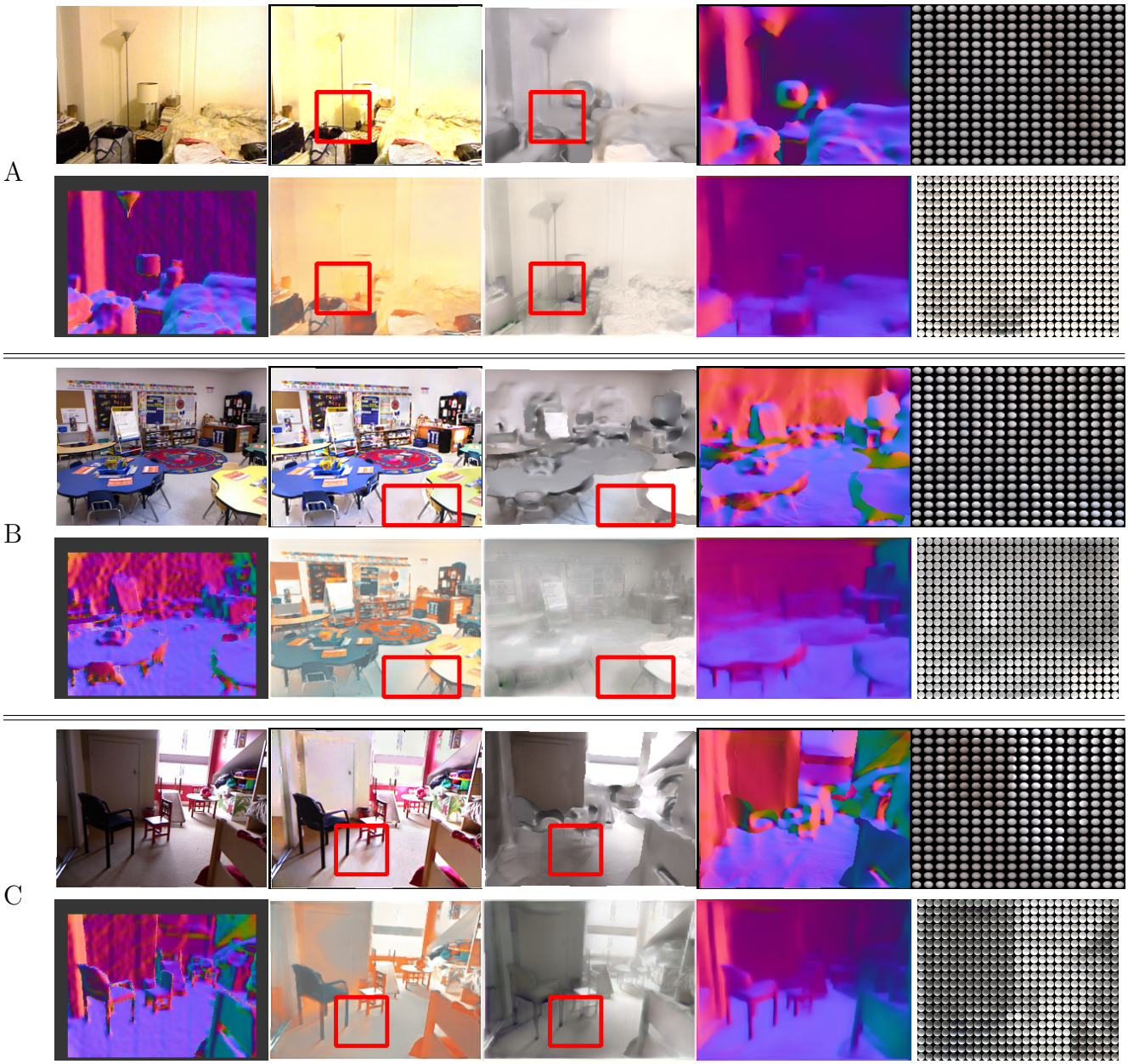
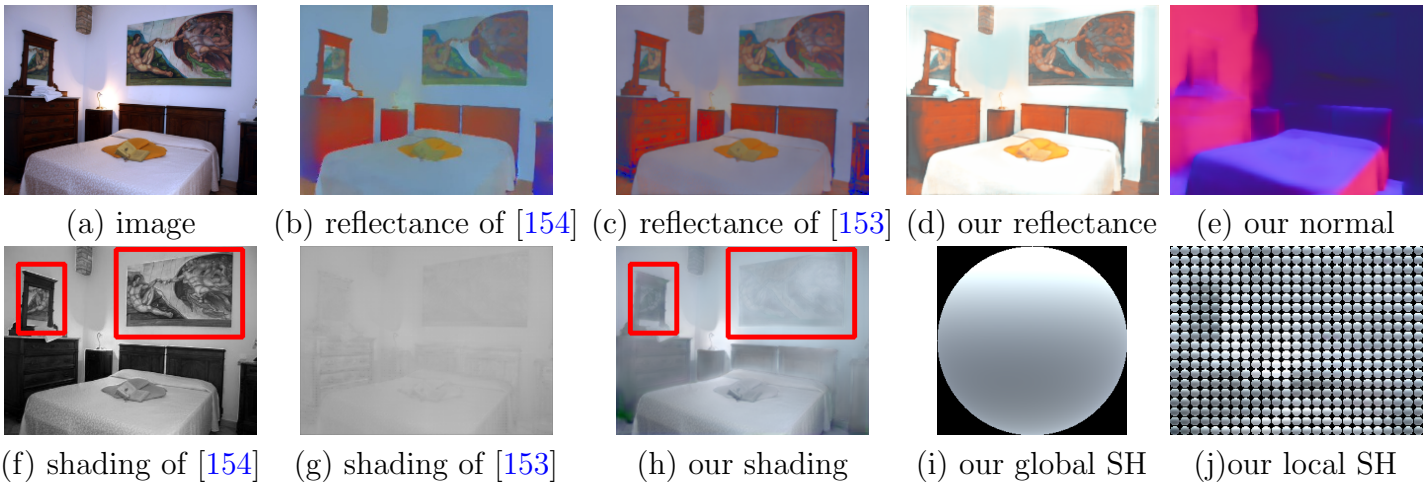


Figure 5.9: Comparison with [2]. First row of A, B, C and D from left to right: input image, reflectance by [2], shading by [2], normal by [2], lighting by [2]. Second row of A, B, C and D from left to right: ground truth normal; reflectance, shading, normal, global SH and local SHs by the proposed method.



(a) image (b) reflectance of [154] (c) reflectance of [153] (d) our reflectance (e) our normal

(f) shading of [154] (g) shading of [153] (h) our shading (i) our global SH (j)our local SH

Figure 5.10: Comparison with state-of-the-art intrinsic image decomposition method. Note that although [153] achieves the best AP score on shading, the generated shading image is of very low contrast. The red rectangle shows the shading of [154] suffers seriously from the reflectance bleeding problem.

5.6.3 Intrinsic Image Decomposition

Model trained on synthetic data. We evaluate our network trained using synthetic data on IIW, SAW and NYUv2. For reflectance on IIW, we use the WHDR metric proposed in [128], which computes the weighted error of the predicted reflectance with human annotation. The challenge average precision (AP) proposed by [153] is used to evaluate the predicted shading. It computes the average precision of classification for constant shading regions and shadow boundaries. Table 5.5 (a) compares our trained network with [153] on IIW and SAW dataset. It shows that our proposed method is closely comparable to [153] on IIW and produces much better results than [153] on SAW when trained on SUNCG dataset.

[153] claimed that the dataset they provided (denoted as CGI) has a smaller domain gap with real data compared with SUNCG. For a sanity check, we train our coarse network using CGI and achieve WHDR 37.98, while the WHDR of our

			IIW	SAW
	Method	Dataset	WHDR (%)↓	AP (%)↑
a	Li [153]	SUNCG	**26.1**	87.09
	Proposed	SUNCG	26.8	**92.40**
b	Grosse [160]	-	26.9	85.26
	Garces [161]	-	24.8	92.39
	Zhao [162]	-	23.8	89.72
	Bi [144]	-	17.7	-
	Bell [128]	-	20.6	92.18
	Zhou [145]	IIW	19.9	86.34
	[146]	IIW	19.5	89.94
	Fan [154]	IIW	15.4	-
	Li [153]	CGI + *real*	15.5	**96.57**
	proposed	SUNCG + *real*	**15.2**	95.01

Table 5.5: Reflectance evaluation on IIW and shading evaluation on SAW. For WHDR, lower value (↓) is better, for AP, higher value is better(↑).

coarse network trained on SUNCG is 28.20. We do not see the advantage of using CGI data for training and thus we train our network using SUNCG dataset.

Model fine-tuned on real data. Table 5.5 (b) compares our method with some start-of-the-art methods on IIW and SAW. Our method achieves the best performance on IIW and second best on SAW.

[154] demonstrated that by incorporating the guided filter into the training of their network, they can achieve a WHDR of 14.5% which is the state-of-the-art result. By applying a guided filter to our model as suggested by [146], we can achieve 14.6% which is closely comparable to this result. However, the challenge AP for the shading of [154] on IIW dataset [1] is **85.77**%. Under the same setting, we achieve **96.81**%, a more than 10% improvement.

Besides reflectance and shading, Table 5.4 shows that the normal predicted by

[1]Images are provided by the authors.

our model achieves strongly competitive results with [158], when trained on SUNCG synthetic data and evaluated on NYUv2. We further fine-tuned the models with limited real data (381 images with surface normal ground truth), and achieved 25° average angular error, close to [158] 21.74°.

Visual comparison. We visualize the shading predicted by [154] in Figure 5.10 (f). It shows that the shading images of [154] still retain the effect of reflectance in their shading prediction. Although [153] achieves the best performance on SAW, Figure 5.10 (g) shows that their predicted shading images are of low contrast. That is, the quality of the shading image is low. Across the compared methods, our method achieves relatively better visual quality on both reflectance and shading. Figure 5.11, 5.12 and 5.13 show more visual results of [154], [153] and the proposed method.

To conclude, our GLoSH achieves consistently better results compared to state-of-the-art methods evaluated on models trained on both synthetic data and fine-tuned on real data, across the tasks of estimating reflectance, normal, shading and lighting. We believe this also indicates the effectiveness of the proposed coarse-to-fine network structure.

5.6.4 Ablation Study

Without synthetic data. Synthetic data is very important for the proposed method. Table 5.6 "w/o SUNCG" shows the WHDR on IIW, average precision (AP) on SAW and mean error on the NYUv2 data set when training our network

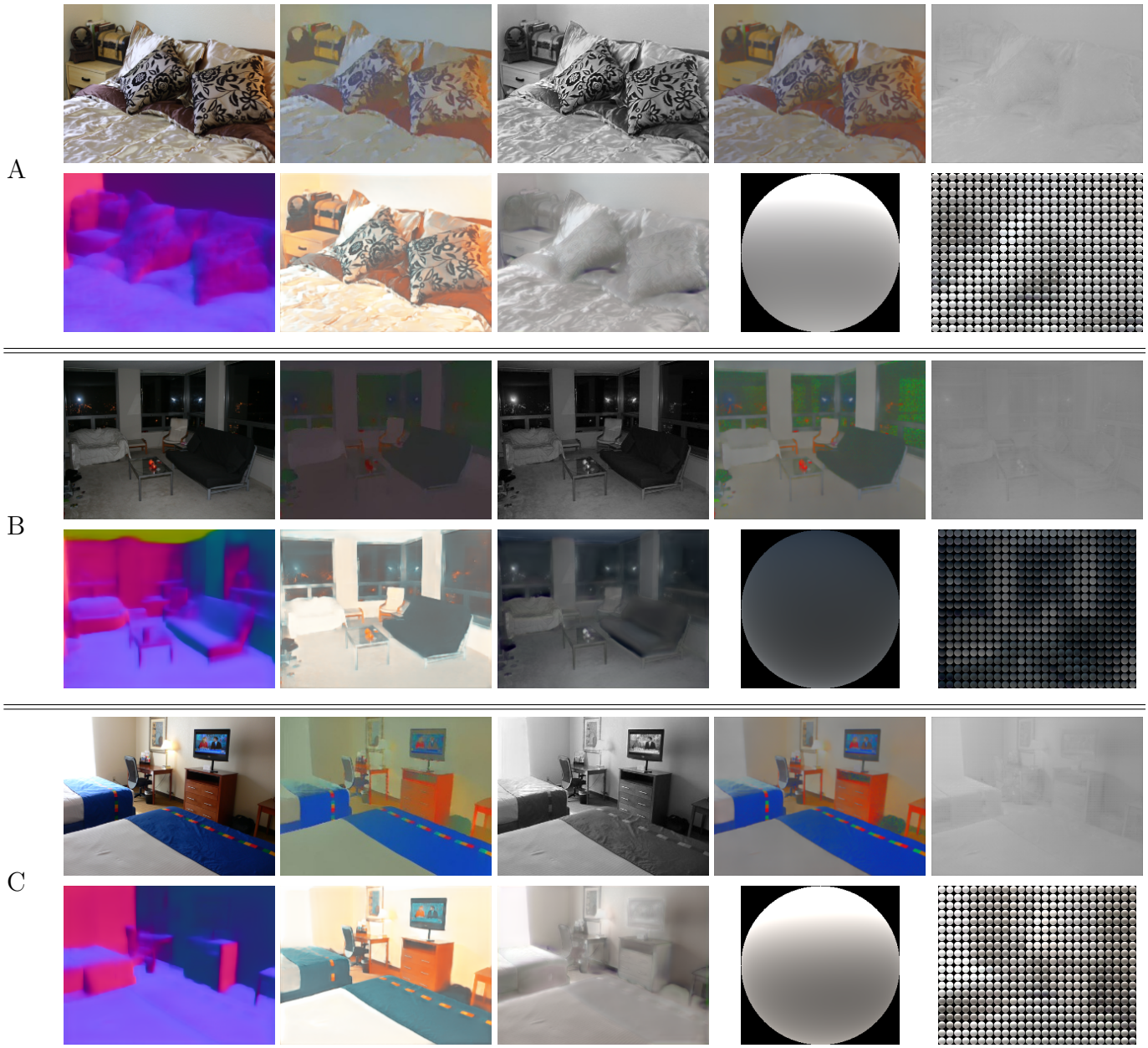


Figure 5.11: comparison with state-of-the-art intrinsic image decomposition methods. First row of A, B and C from left to right: input image, reflectance by [154], shading by [154], reflectance by [153], shading by [153]. The second row of A, B and C from left to right: reflectance, shading, normal, global SH and local SHs of the proposed method.

only using real data. It is clear that without synthetic data, the performance of our network on reflectance, shading and normal shows significant gap relative to the "full" model. This is because training a network that performs reasonably well

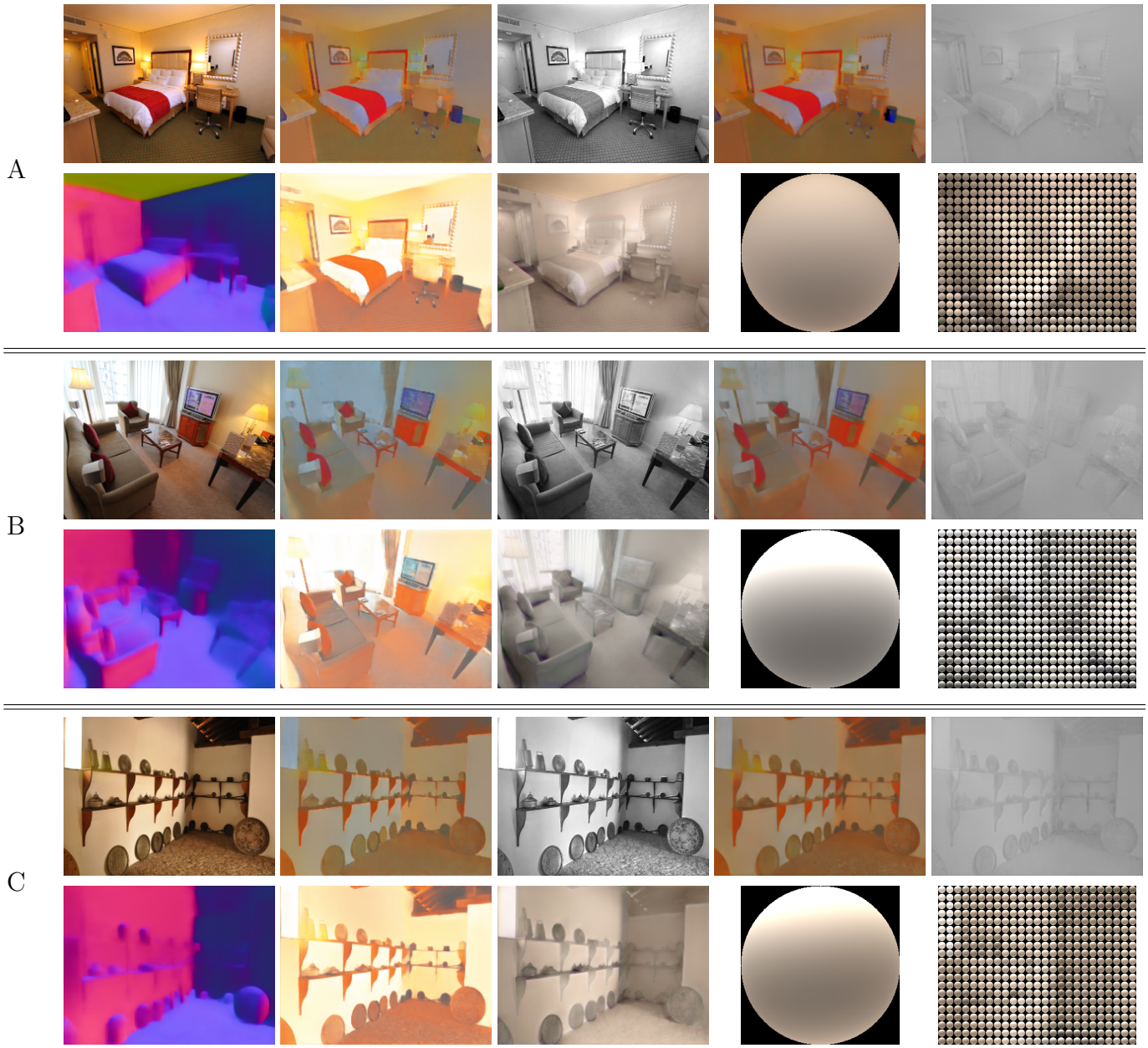


Figure 5.12: comparison with state-of-the-art intrinsic image decomposition methods. First row of A, B, C and D from left to right: input image, reflectance by [154], shading by [154], reflectance by [153], shading by [153]. The second row of A, B, C and D from left to right: reflectance, shading, normal, global SH and local SHs of the proposed method.

requires a huge amount of data. The sparsity of the annotation for reflectance and shading, and the small amount of real images makes the training intractable.

Without pseudo supervision. Table 5.6 "w/o $\mathcal{L}_r^{ss}$" shows that on IIW and

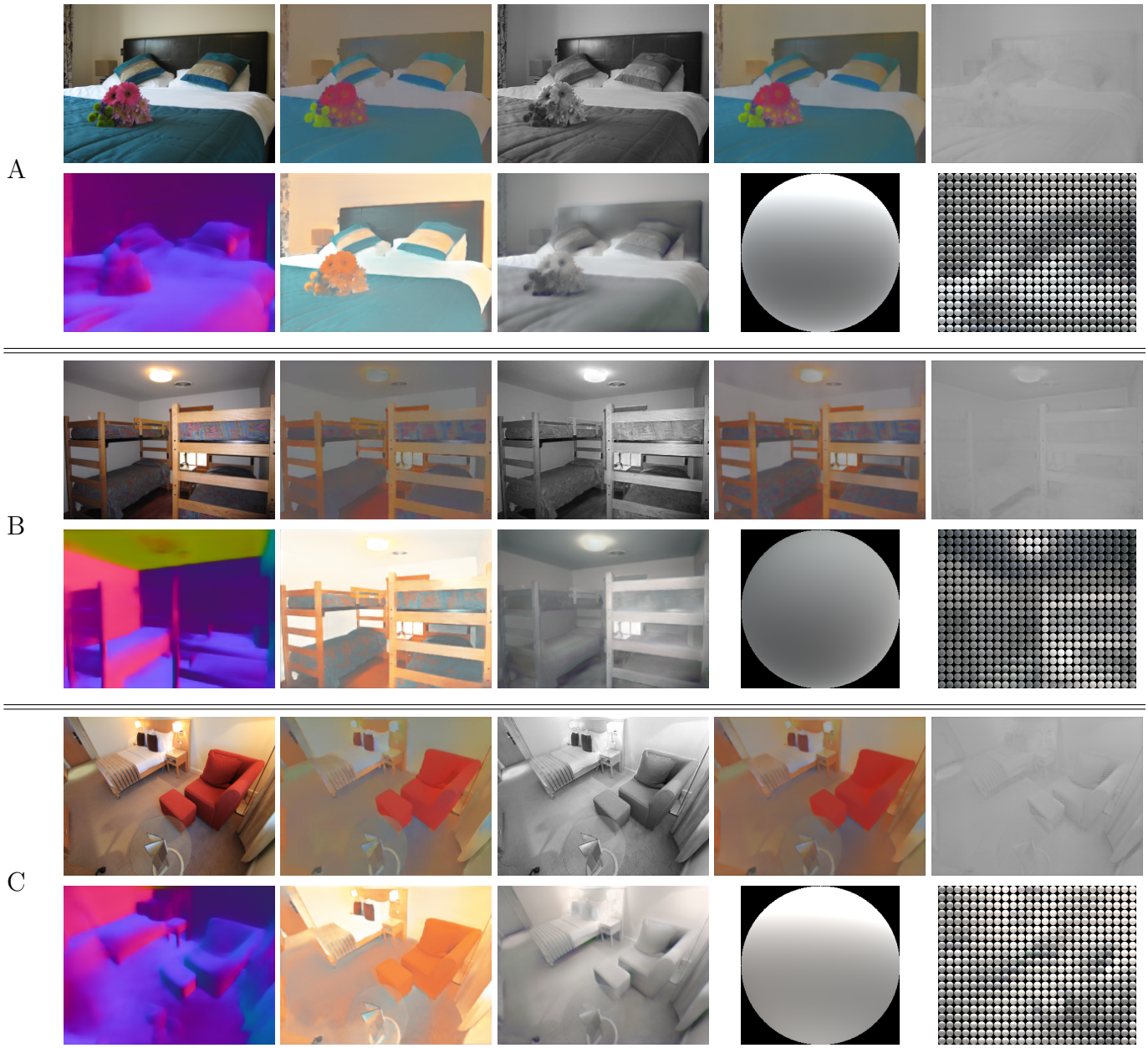


Figure 5.13: Comparison with state-of-the-art intrinsic image decomposition methods. First row of A, B, C and D from left to right: input image, reflectance by [154], shading by [154], reflectance by [153], shading by [153]. The second row of A, B, C and D from left to right: reflectance, shading, normal, global SH and local SHs of the proposed method.

NYUv2, performance degrades relative to the the "full" model, except for the AP on the SAW dataset. This shows that the self-supervision helps to provide rough guidance for the real unlabeled data on reflectance and normal. The degradation

	IIW	SAW	NYUv2
Method	WHDR (%) ↓	AP (%) ↑	Mean Error (°) ↓
w/o SUNCG	17.82	88.52	35.14
w/o $\mathcal{L}_r^{ss}$	15.50	**95.79**	25.93
w/o $\mathcal{L}_{R2}$	15.34	91.89	25.96
scale1	18.70	90.35	26.68
scale1+scale2	16.62	94.98	25.59
full	**15.20**	95.01	**25.44**

Table 5.6: Ablation study on loss, without synthetic SUNCG data, and the coarse-to-fine scales, evaluated on IIW reflectance, SAW shading and NYUv2 surface normal.

for shading is probably due to the large domain gap bwteen the lighting of synthetic data and real data. However, when compared with shading of [153] in Figure 5.10, we see even with weak supervision, our model can still predict more reasonable shading.

Contribution of multiple scales. We clearly see in Table 5.6 that "scale1+scale2" outperforms "scale1", and our "full" model further outperforms "scale1+scale2". It suggests that further adding a finer scale module indeed helps the local lighting modeling and boosts the overall performance. Worth noting that there is gradually saturation by further adding finer modules as the improvement gap from "scale1+scale2" to "full" is smaller than "scale1" to "scale1+scale2". In practice, we define our full model to have three scales, a coarse net with two cascaded finer nets, which strikes a good balance between accuracy and model complexity.

Without symmetric loss. The WHDR hinge loss proposed by [146] (Equation 5.8) is not symmetric. This leads to unequal loss when the same points are used in a different order. By adapting the WHDR to our proposed symmetric one (Equa-

tion 5.9), we observe improvement on IIW by 0.14%.

Model complexity: We calculate the model parameters of CGI [153] and our full model. There are $68,572,482$ floating numbers in CGI and only $14,665,594$ in our model, which is much smaller than CGI. Among the state-of-the-art CNN based methods, our method achieves consistently better performance with a smaller model size.

5.7 Summary

In this chapter, we propose to estimate reflectance, normal and lighting from a single image, which is a very hard problem that has not been well addressed. A global and local SHs model is proposed to model the lighting of a natural scene, which accounts for both holistic lighting and the spatial variation of the lighting. A novel non-negative constraint is proposed to force the SH lighting to be physically meaningful. A synthetic data set is applied as augmentation for real data. Extensive experiments on SAW, IIW, and NYUv2 datasets demonstrate the effectiveness of our proposed method.

Chapter 6: Conclusion

Lighting is the medium through which we capture images of the physical world. Understanding lighting from images can help the computer better understand the world. Conventional computer vision algorithms usually try to understand lighting from images through optimizing complicated objective functions with priors designed by an expert. They are usually slow and the performance relies on the quality of priors. Recently, deep CNNs have been applied to many computer vision problems and achieve great success. In this dissertation, we try to apply deep CNNs for the task of understanding lighting from images.

The biggest challenge to understanding lighting from images is the lack of labeled data. Deep CNNs are notorious for their data hungry nature. Millions of labeled data are needed in order to train a CNN that works reasonably well. However, labeling lighting from images is impossible.

In this dissertation, we study how to apply synthetic data to train a CNN for understanding lighting from images.

1. We first empirically study the capacity of CNNs by compressing them. Our sparsity-induced method can reduce a huge amount of parameters for several popular CNNs (e.g. more than 76% of parameters in AlexNet when training

on ImageNet), which shows the current CNNs contain a lot of redundancy.

2. We then designed a label denoising network to make use of synthetic data to help estimate lighting from real face images. This is a task for which it is extremely difficult to collect real labels. We demonstrate our proposed method significantly outperforms the current state-of-the-art lighting estimation methods.

3. A deep CNN based portrait relighting method is then proposed. Lacking of an existing dataset, we generate a large scale, high resolution, "in the wild" dataset for this task. Our model trained on the proposed dataset outperforms existing methods significantly.

4. At last, we proposed a novel intrinsic image decomposition algorithm for a natural scene. We are the first to decompose an RGB image of a natural scene into reflectance, normal and lighting. A novel global and local lighting model is proposed to model the complicated lighting conditions of a natural scene.

Future direction. Understanding lighting from images is an interesting but hard problem. Though our proposed methods achieve great progress, there are still many open questions in this filed. We list a few possible research directions below:

Powerful and compact lighting model. One direction is how to find a better lighting model. The Spherical Harmonics (SH) model [34,35] used in our work assumes lighting is distant and objects are convex, moreover, it cannot model

cast shadows and high frequency lighting components. Modeling lighting using environment maps requires a huge number of parameters. As a result, finding a powerful and compact lighting representation is necessary.

Realistic synthetic data with ground truth lighting Due to the difficulty of labeling ground truth lighting for real images, synthetic data is used to train deep CNNs in our work. However, the synthetic data either have a large domain gap with real data (e.g. the synthetic data we used in [13]) or the lighting is not accurate [99]. How to generate more realistic synthetic data with accurate lighting is an interesting direction to explore.

Appendix A: Evaluating Local Features for Day-Night Matching

In this chapter, we introduce our work of evaluating the performance of local features in the presence of large illumination changes that occur between day and night. Through our evaluation, we find that repeatability of detected features, as a de facto standard measure, is not sufficient in evaluating the performance of feature detectors; we must also consider the distinctiveness of the features. Moreover, we find that feature detectors are severely affected by illumination changes between day and night and that there is great potential to improve both feature detectors and descriptors.

This work [163] is in collaboration with Torsten Sattler and David W. Jacobs.

A.1 Introduction

Feature detection and matching is one of the central problems in computer vision and a key step in many applications such as Structure-from-Motion [164], 3D reconstruction [165], place recognition [166], image-based localization [167], Augmented Reality and robotics [168], and image retrieval [169]. Many of these applications require robustness under changes in viewpoint. Consequently, research on feature detectors [170–175] and descriptors [176–179] has for a long time focused

on improving their stability under viewpoint changes. Only recently has robustness against seasonal [180] and illumination changes [174,181] come into focus. Especially the latter is important for large-scale localization and place recognition applications, *e.g.*, for autonomous vehicles. In these scenarios, the underlying visual representation is often obtained by taking photos during daytime and it is infeasible to capture large-scale scenes also during nighttime.

Many popular feature detectors such as Difference of Gaussians (DoG) [176], Harris-affine [182], and Maximally Stable Extremal Regions (MSER), as well as the popular SIFT descriptor [176] are invariant against (locally) uniform changes in illumination. However, the illumination changes that can be observed between day and night are often highly non-uniform, especially in urban environments (*cf.* Fig. A.1). Recent work has shown that this causes problems for standard feature detectors: Verdie *et al.* [174] demonstrated that a detector specifically trained to handle temporal changes significantly outperforms traditional detectors in challenging conditions such as day-night illumination variations. Torii *et al.* [166] observed that foregoing the feature detection stage and densely extracting descriptors instead results in a better matching quality when comparing daytime and nighttime images. Naturally, these results lead to a set of interesting questions: (i) to what extent is the feature detection stage affected by the illumination changes between day and night? (ii) the number of repeatable features provides an upper bound on how many correspondences can be found via descriptor matching. How tight is this bound, *i.e.*, is finding repeatable feature detections the main challenge of day-night matching? (iii) how much potential is there to improve the matching performance of local de-

tectors and descriptors, *i.e.*, is it worthwhile to invest more time in the day-night matching problem?

In this work, we aim at answering these questions through extensive quantitative experiments, with the goal of stimulating further research on the topic of day-night feature matching. We are interested in analyzing the impact of day-night changes on feature detection and matching performance. Thus, we eliminate the impact of viewpoint changes by collecting a large dataset of daytime and nighttime images from publicly available webcams [183] [1]. Through our experiments on this large dataset, we find that: (i) the repeatability of feature detectors for day-night image pairs is much smaller than that for day-day and night-night image pairs, meaning that detectors are severely affected by illumination changes between day and night; (ii) for day-night image pairs, high repeatability of feature detectors does not necessarily lead to a high matching performance. For example, the TILDE [174] detector specifically learned for handling illumination changes has a very high repeatability, but the precision and recall of matching local features are very low. A low recall shows that the number of repeatable points provides a loose bound for the number of correspondences that could be found via descriptor matching. As a result, further research is necessary for improving both detectors and descriptors; (iii) through dense local feature matching, we find that there are a lot more correspondences that could be found using local descriptors than are produced by current detectors, *i.e.*, there is great potential to improve detectors for day-night feature matching.

[1] Please find the data set at `http://www.umiacs.umd.edu/~hzhou/dnim.html`

A.2 Dataset

Illumination and viewpoint changes are two main factors that would affect the performance of feature detectors and descriptors. Ideally, both detectors and descriptors should be robust to both type of changes. However, obtaining a large dataset with both types of changes with ground truth transformations is difficult. In this work, we thus focus on pure illumination changes and collect data that does not contain any viewpoint changes. Our results show that already this simpler version of the day-night matching problem is very hard.

The AMOS dataset [183], which contains a huge number of images taken (usually) every half an hour by outdoor webcams with fixed positions and orientations, satisfies our requirements perfectly. [174] has collected 6 sequences of images taken at different times of the day for training illumination robust detectors from the AMOS dataset. However, the dataset has no time stamps and some of the sequences have no nighttime images. As a consequence, we collect our own dataset from AMOS. 17 image sequences with relatively high resolution containing 1722 images are selected. Since the time stamps of the images provided by AMOS are usually not correct, we choose image sequences with time stamp watermarks. The time of the images will be decided by the watermarks which are removed afterwards. For each image sequence, images taken in one or two days are collected. Fig. A.1 gives an example of images we collected.

Figure A.1: Images taken from 00:00 - 23:00 in one image sequence of our dataset.

A.3 Evaluation

A.3.1 Keypoint Detectors

For evaluation, we focus on the keypoint detectors most commonly used in practice. We choose DoG [176], Hessian, HessianLaplace, MultiscaleHessian, HarrisLaplace and MultiscaleHarris [182] implemented by vlfeat [184] for evaluation. Their default parameters are used to determine how well these commonly used settings perform under strong illumination changes. DoG detects feature points as the extrema of the difference of Gaussian functions. By considering the extrema of the difference of two images, DoG detections are invariant against additive or multiplicative (affine) changes in illumination. Hessian, HessianLaplace and MultiscaleHessian are based on the Hessian matrix $\begin{pmatrix} L_{xx}(\sigma) & L_{xy}(\sigma) \\ L_{yx}(\sigma) & L_{yy}(\sigma) \end{pmatrix}$, where L represents the image smoothed by a Gaussian with standard deviation σ and L_{xx}, L_{xy}, and L_{yy} are the second-order derivatives of L. Hessian detects feature points as the local maxima

of the determinant of the Hessian matrix. HessianLaplace chooses a scale for the Hessian detector that maximizes the normalized Laplacian $|\sigma^2(L_{xx}(\sigma) + L_{yy}(\sigma))|$. MultiscaleHessian instead applies the Hessian detector on multiple scales of images and detects feature points at each scale independently. HarrisLaplace and MultiscaleHarris extended the Harris corner detector to multiple scales in a similar way to HessianLaplace and MultiscaleHessian. The Harris corner detector is based on the determinant and trace of the second moment matrix of gradient distribution. All these gradient based methods are essentially invariant to additive and multiplicative illumination changes.

We also included the learning based detector TILDE [174], since it is designed to be robust to illumination changes. We use the model trained on the St. Louis sequence as it has the highest repeatability when testing on the other image sequences [174]. TILDE detects feature points at a fixed scale. In this work, we define a multiple scale version by detecting features at multiple scales, denoted as MultiscaleTILDE. Feature points are detected from the original image and images smoothed by a Gaussian with standard deviation of 2 and 4. When TILDE detects feature points from the original image, the scale of it is set to be 10. Accordingly, the scale of detected feature points from those three images are set to be 10, 20 and 40. As suggested by [174], we keep a fixed number of feature points based on the resolution of the image. For the proposed MultiscaleTILDE, the same number of feature points as that of TILDE are selected for the first scale. For other scales, the number of feature points selected are reduced by half compared with the previous scale. In modified versions, we include 4 times as many points as suggested, naming

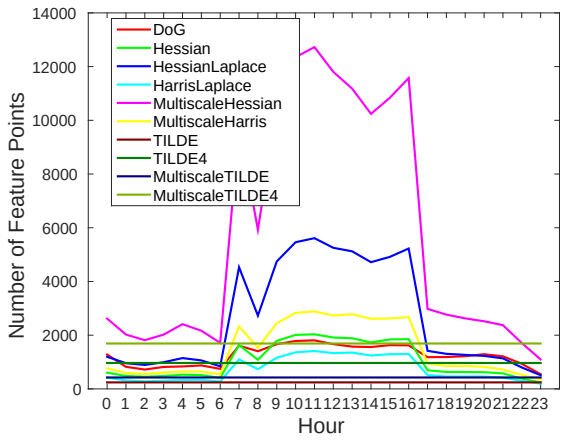


Figure A.2: The number of feature points detected at different time

these TILDE4 and MultiscaleTILDE4 respectively.

A.3.2 Repeatability of Detectors

In this section we address the question: **to what extent are feature detections affected by illumination changes between day and night?** by evaluating how many feature points are detected, and how repeatable they are. First we show the number of detected feature points at different times of the day for different detectors in Fig. A.2. The numbers are averaged from all 17 image sequences in our dataset. The number of feature points for TILDE is the same across different times, since a fixed number of feature points are extracted. For the other detectors, fewer feature points are detected at nighttime. Especially, the number of feature points detected by HessianLaplace and MultiscaleHessian are affected most by illumination changes between day and night.

We then use the repeatability of the detected feature points to evaluate the performance of detectors. According to [170], the measurement of repeatability is related to the detected region of feature points. Suppose σ_a and σ_b are the

scale of two points A and B, (x_a, y_a) and (x_b, y_b) are their locations, the detected regions μ_a and μ_b are defined as the region of $(x - x_a)^2 + (y - y_a)^2 = (3\sigma_a)^2$ and $(x-x_b)^2+(y-y_b)^2 = (3\sigma_b)^2$ respectively, where 3σ is the size of one spatial bin from which the SIFT feature is extracted. Then A and B are considered to correspond to each other if $1-\frac{\mu_a \cap \mu_b}{\mu_a \cup \mu_b} \leq 0.5$, *i.e.* the intersection of these two regions are larger than or equal to half of the union of these two regions. This overlap error is the same as the one proposed in [170] except that we do not normalize the region size. This is because if the detected regions do not overlap, we cannot extract matchable feature descriptors; normalizing the size of the region would obscure this. For example, two regions with small scales may be judged to correspond after normalization. However, the detected region from which the feature descriptor is extracted may not overlap at all, making it impossible to extract feature descriptors to match them.

Some of the images in our dataset may contain moving objects. To avoid the effect of those objects, we define "ground truth" points and compute the repeatability of detectors at different times w.r.t. them. To make the experiments comprehensive, we use daytime ground truth and nighttime ground truth. Images taken at 10:00 to 14:00 are used to get the daytime ground truth feature points (and 00:00 to 02:00 together with 21:00 to 23:00 for nighttime ground truth feature points). We select the image that has the largest number of detected feature points and match them to those in other images in that time period. A feature point is chosen as a ground truth if it appears in more than half of all the images of that time period. Fig. A.3 (a) and (b) shows the number of daytime and nighttime ground truth feature points detected for different detectors respectively. We notice that though Fig. A.2 shows

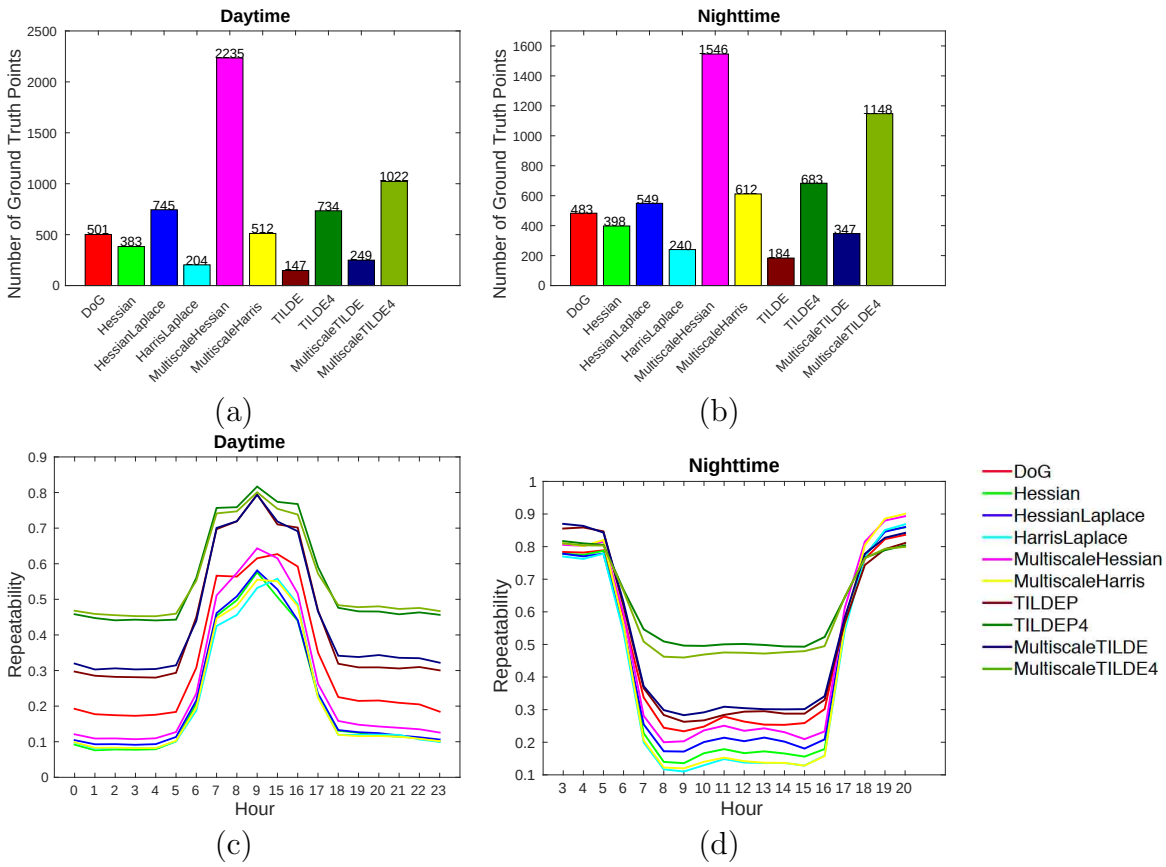


Figure A.3: (a) and (b) show average number of daytime and nighttime ground truth feature points for each detector respectively. We show repeatability of different detectors at different times of the day w.r.t. (c) daytime and (d) nightime ground truth feature points. Please note time periods that are used to compute the ground truth feature points are excluded for fair comparison.

the number of detected feature points for TILDE4 at daytime is the second smallest among all the detectors, the number of daytime ground truth feature points of TILDE4 is larger than 6 detectors. This implies that the feature points detected by TILDE4 for daytime images are quite stable across different images.

We use these ground truth feature points to compute the repeatability of the chosen detectors over different times of the day. Thus, repeatability is determined by measuring how often the ground truth points are re-detected. Fig. A.3 (c) and (d) show that the repeatability of features for nighttime images w.r.t. nighttime

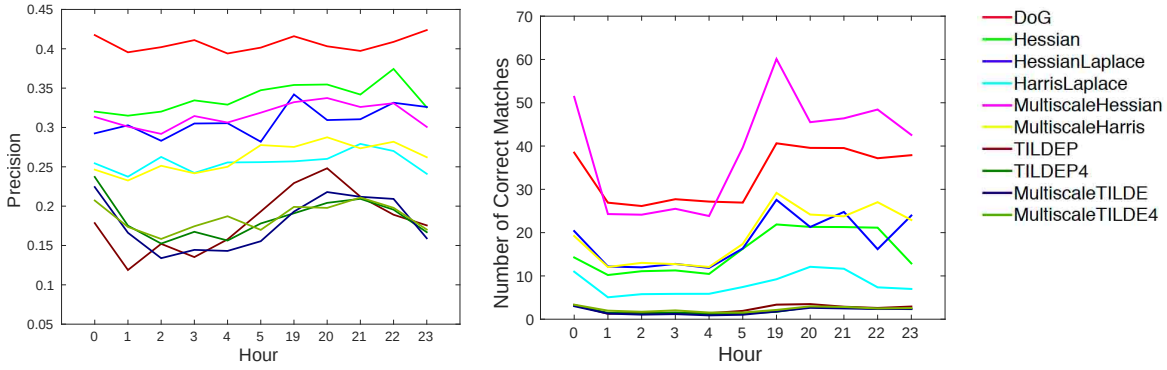
ground truth is very high for all the detectors; this is because the illumination of nighttime images are usually quite stable without the effect of sunlight (*cf.* Fig. A.1). For comparison, the repeatability of daytime images w.r.t. daytime ground truth is smaller and the performance of different detectors varies a lot. Moreover, both Fig. A.3 (c) and (d) show that the repeatability of day-night image pairs is very low for most detectors, which implies that detectors are heavily affected by day-night illumination changes. The drop-off between 05:00-07:00 and 17:00-18:00 is caused by illumination changes between dusk and dawn. The peaks of the repeatability, as 09:00 in Fig. A.3 (c) and 03:00 and 20:00 in Fig. A.3 (d), appear because they are close to the time from which the ground truth feature points are computed. Among all the detectors, both single scale and multiple scale TILDE have high repeatabilities of around 50% for day-night image pairs. This is not surprising since the TILDE detector was constructed to be robust to illumination changes by learning the impact of these changes from data. Based on the fact that nearly every second TILDE keypoint is repeatable, we would expect that TILDE is well-suited for feature matching between day and night.

A.3.3 Matching Day-Night Image Pairs

In theory, every repeatable keypoint should be matchable with a descriptor since its corresponding regions in the two images have a high overlap. In practice, the number of repeatable keypoints is only an upper bound since the descriptors extracted from the regions might not match. For example, local illumination changes

might lead to very different descriptors. In this section, we thus study the performance of detector+descriptor on matching day-night image pairs. We try to answer the question **whether finding repeatable feature detections is the main challenge of day-night feature matching**, *i.e.*, whether finding repeatable keypoints is the main bottleneck or whether additional problems are created by the descriptor matching stage. We use both precision and recall of feature descriptor matching to answer this question. Suppose for a day-night image pair, N true matches are provided by detectors. N_f matched feature points are found by matching descriptors, among which N_c matches are true matches. Then the precision and recall of detector+descriptor are defined as N_c/N_f and N_c/N, respectively. Precision is a usual way to evaluate the accuracy of matching by detector+descriptor. Recall, on the other hand, tells us what is the main challenge to increase the number of matches. A low recall means improving feature descriptors is the key to getting more matches. On the contrary, feature detection is the bottleneck for getting more matches if a high recall is observed, but still an insufficient number of matching features are found.

For each image sequence, images taken at 00:00 - 05:00 and 19:00 - 23:00 are used as nighttime images and those taken at 09:00 - 16:00 are daytime images. One image is randomly selected from each hour in these time periods and every nighttime image is paired with every daytime image to create the day-night image pairs. As the SIFT descriptor is still the first choice in many computer vision problems, and its extension, RootSIFT [185] performs better than SIFT, we use RootSIFT as the feature descriptor. To match descriptors, we use nearest neighbor



(a) (b)

Figure A.4: (a) shows the precision of RootSIFT matching of day-night image pairs for different detectors at different nighttime. (b) shows their corresponding number of correct matched feature points.

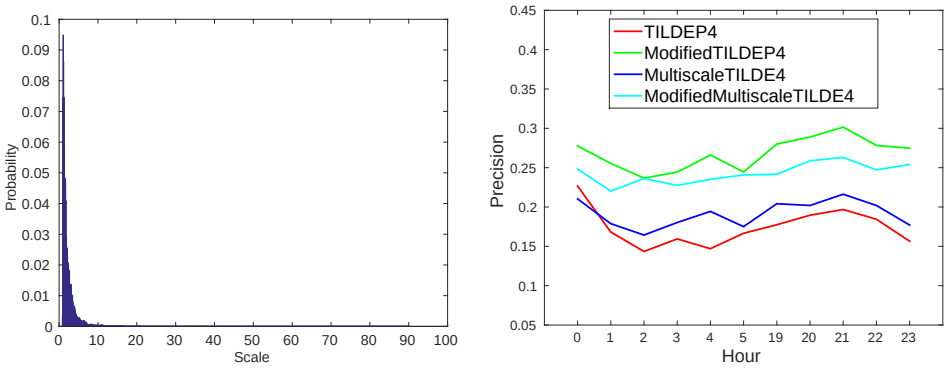
search and apply Lowe's ratio test [176] to remove unstable matches. The default ratio provided by vlfeat [184] is used in our evaluation. In practice, the ratio test is used to reject wrong correspondences but also rejects correct matches. The run-time of subsequent geometric estimation stages typically depends on the percentage of wrong matches. Sacrificing recall for precision is thus often preferred since there is enough redundancy in the matches.

The precisions of matching day-night images for all the detectors at different daytimes are shown in Fig. A.4 (a). We found that though different versions of TILDE have the highest repeatability among all the detectors, in general, they have the lowest precision. There are more than 20% drop in precision w.r.t. DoG in most cases. This shows that a higher repeatability of a detector does not necessarily mean better performance for finding correspondences, and detectors and descriptors are highly correlated with each other for matching feature points. As shown in Fig. A.4 (b), the number of correct matches detected by RootSIFT for all the detectors are quite small. Even for detectors like DoG, which has the highest precision, only

20 - 40 correct matches can be found. As a result, for applications that need a large number of matches between day-night image pairs, the performance of these detector+descriptor may not be satisfactory.

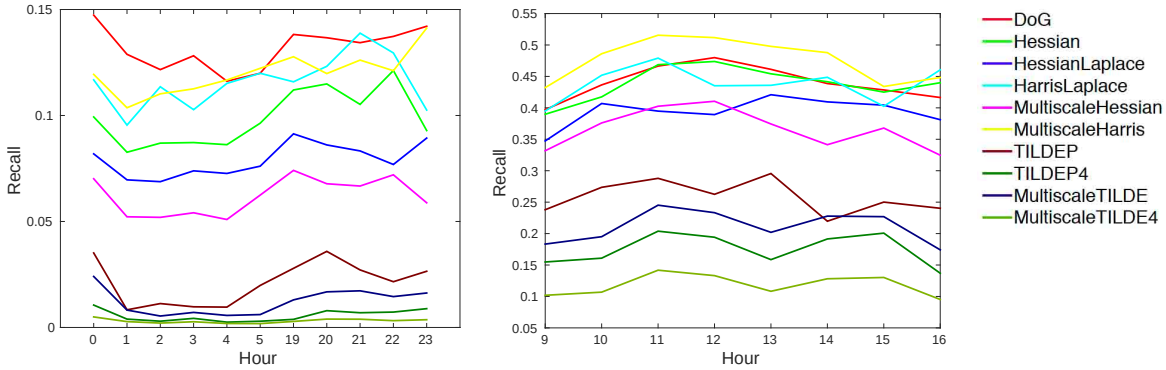
Another interesting finding from Fig. A.4 is that the multiple scale version of TILDE, MultiscaleTILDE4, does not have a higher precision compared with TILDE4. One possible reason may be that the scales of features detected by MultiscaleTILDE4 are set to 10, 20 and 40, which are too large. Fig. A.5 (a) shows that most of the correctly matched features of DoG+RootSIFT have scales within 10. This is because within a smaller region, the illumination changes between day and night images are more likely to be uniform, to which the RootSIFT feature is designed to be robust. To better understand the effect of scales, we set the scale of TILDE4 to be 1 and scales of MultiscaleTILDE4 to be 1, 2 and 4, and denote these two modified versions as ModifiedTILDE4 and ModifiedMultiscaleTILDE4. Fig. A.5 (b) compares the precision of them with TILDE4 and MultiscaleTILDE4. We find that by setting the scale to be small, there is around $5\% - 10\%$ increase in precision. Intuitively, with larger scales, features may contain more information, which should be beneficial for matching. However, descriptors will need to be trained to make good use of those information, especially when robustness under viewpoint changes is required.

To examine the main challenge of finding more correspondences, the recall of these detectors for matching day-night image pairs is shown in Fig. A.6 (a). We find that the recall of each detector is very low. As a consequence, **one way to improve the performance of day-night image matching is to improve**



(a) (b)

Figure A.5: (a) shows the histogram of scales for correctly matched RootSIFT features using DoG as detector. (b) compares the precision of TILDE4 and MultiscaleTILDE4 with small and large scales.

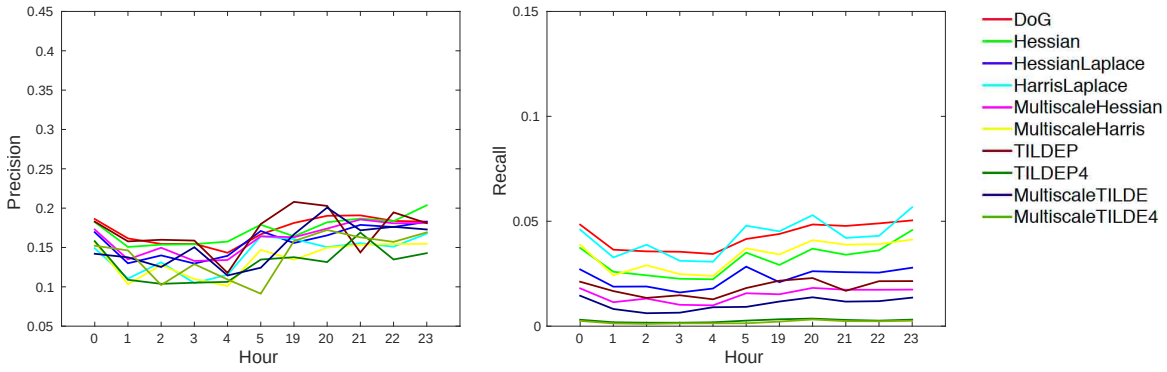


(a) (b)

Figure A.6: (a) shows the recall of RootSIFT matching of day-night image pairs for different detectors at different nighttimes. (b) shows the recall of RootSIFT matching of day-day image pairs for different detectors at different daytimes.

the robustness of descriptors to severe illumination changes. As shown in Fig. A.6, a much higher recall can be noticed for day-day image pairs, meaning that RootSIFT is robust to small illumination changes at daytime. However, it is not so robust to severe illumination changes between day and night. The low recall of day-night image pairs implies that there are a lot of "hard" patches from which RootSIFT cannot extract good descriptors.

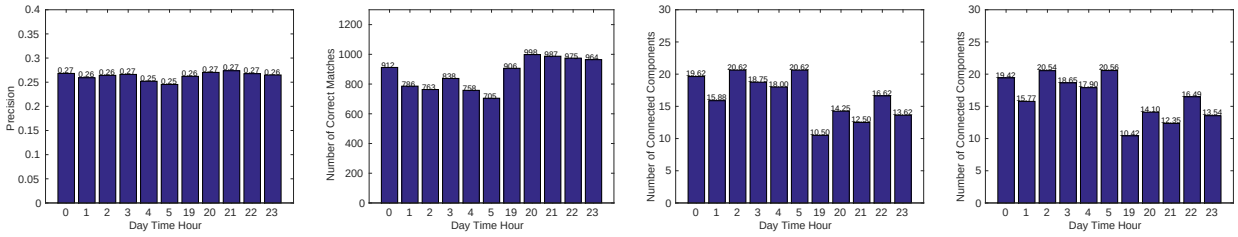
With the development of deep learning, novel feature descriptors based on



(a) (b)

Figure A.7: (a) shows the precision of matching of day-night image pairs for different detectors at different nighttime using cnn feature. (b) shows the recall of matching of day-night image pairs for different detectors at different nighttime using cnn feature.

convolutional neural networks have been proposed. Many of them [8, 186, 187] outperform SIFT. We choose the feature descriptor proposed in [186] as an example to evaluate the performance of the learned descriptor+detector. [186] is chosen since their evaluation method is Euclidean distance, which can be used easily in our evaluation framework. Fig. A.7 shows that this CNN feature performs even worse than RootSIFT+detectors. One reason is that [186] is learned from the data provided by [177], which mainly focus on viewpoint changes, and illumination changes that are small. Though [186] shows its robustness to small illumination changes as in DaLI dataset [188], it is not very robust to illumination changes between day and night in our dataset.[2]



(a) (b) (c) (d)

Figure A.8: (a) shows the precision of matching dense RootSIFT for day-night image pairs at different nighttimes. (b) shows the number of correct matches of dense RootSIFT for day-night image pairs. (c) shows the number of connected components of matched points at different nighttime. (d) shows the number of connected components that contain no matched points of DoG+RootSIFT.

A.4 Potential of Improving Detectors

In this section, we try to examine **the potential of improving feature detectors** by fixing the descriptor to be RootSIFT.

Inspired by [166], we extract dense RootSIFT features from day-night image pairs for matching. When doing the ratio test, we select the neighbor that lies outside the region from which nearest neighbor's RootSIFT feature is extracted to avoid comparing similar features. Figure A.8 (a) and (b) show the precision of dense RootSIFT matching and the number of matched feature points. Though the precision is not improved compared with the best performing detector+RootSIFT, the number of matched feature points improves a lot. This means that there are a lot of "easy" RootSIFT features that could be matched for day-night image pairs. However, we find that the matched RootSIFT features tend to cluster. Since detectors would usually perform non-maximum suppression to get stable detections, in

[2]We also tried to tune the descriptor using day-night patch pairs, but were not able to increase the descriptor's performance.

(a) (b)

Figure A.9: (a) correct matches of DoG+RootSIFT. (b) correct matches of dense RootSIFT.

the worst case, only one feature could be detected from each cluster. As a result, the number of these matched features is an upper bound that cannot be reached. Instead, we try to get a lower bound on the number of additional potential matches that could be found. To achieve that, we count the number of connected components for those matched RootSIFT features and show the result in Fig. A.8 (c). Taking DoG as an example, we show the number of connected components that contain no correct matches found by detector+RootSIFT in Fig. A.8 (d). We found that matches found by detector+RootSIFT have almost no overlap with the connected components of the matched dense RootSIFT, meaning that there is great potential to improve feature detectors. Moreover, we notice that there are generally 10 - 20 connected components found by dense RootSIFT. This is in the order of correct matches we could get for day-night image matching shown in Fig. A.4.

Fig. A.9 shows an example of correct matches found by DoG+RootSIFT and dense RootSIFT. For this day-night image pair, DoG+RootSIFT can only find 4 correct matches whereas dense RootSIFT can find 188 correct matches. Fig. A.10 shows the detected feature points using DoG for the day and night images and their corresponding heat map of cosine distance of dense RootSIFT. The colored

(a) (b) (c)

Figure A.10: (a) and (b) shows an example of nighttime and daytime image with detected feature points using DoG. (c) shows the heat map of the cosine distance of dense RootSIFT for (a) and (b).

rectangles in Fig. A.9 (b) and those in Fig. A.10 (a), (b) (c) are the same area. It is clearly shown that the cosine distances of points in that area between day and night images are very large, and Fig. A.9 (b) shows that they can be matched using dense RootSIFT. However, though many feature points can be detected in the daytime image, no feature points are detected by DoG for the nighttime image[3]. As a result, matches that could be found by RootSIFT are missed due to the detector. In conclusion, a detector which is more robust to severe illumination changes can help improve the performance of matching day-night image pairs.

A.5 Summary

In this work, we evaluated the performance of local features for day-night image matching. Extensive experiments show that repeatability alone is not enough for evaluating feature detectors. Instead, descriptors should also be considered. Through the discussion about precision and recall of matching day-night images and examining the performance of dense feature matching, we concluded that there

[3]The area in the rectangle of the night image actually has a lot of structure, it appears to be totally dark due to low resolution

is great potential for improving both feature detectors and descriptors. Thus, further evaluation with parameter tuning and advanced descriptors [189] as well as principal research on the day-night matching problem is needed.

Bibliography

[1] YiChang Shih, Sylvain Paris, Connelly Barnes, William T. Freeman, and Frédo Durand. Style transfer for headshot portraits. *ACM Trans. Graph.*, 33(4), 2014.

[2] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. In *CVPR*, 2013.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

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

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[8] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015.

[9] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *ECCV*, 2016.

[10] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[11] Abhishek Kar, Christian Hne, and Jitendra Malik. Learning a multi-view stereo machine. In *NIPS*, 2017.

[12] Matan Sela, Elad Richardson, and Ron Kimmel. Unrestricted facial geometry reconstruction using image-to-image translation. In *ICCV*, 2017.

[13] Hao Zhou, Jin Sun, Yaser Yacoob, and David W. Jacobs. Label denoising adversarial network (ldan) for inverse lighting of faces. In *CVPR*, June 2018.

[14] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'aurelio Ranzato, and Nando D. Freitasa. Predicting parameters in deep learning. In *NIPS*, 2013.

[15] Maxwell D. Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. *CoRR*, abs/1412.1442, 2014.

[16] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[17] Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for deep neural networks. In *BMVC*, 2015.

[18] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10, 2009.

[19] Tom Goldstein, Christoph Studer, and Richard G. Baraniuk. A field guide to forward-backward splitting with a FASTA implementation. *arXiv eprint*, abs/1411.3406, 2014.

[20] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[21] Ji Liu, P. Musialski, P. Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on PAMI*, 35(1), 2013.

[22] Wei Deng, Wotao Yin, and Yin Zhang. Group sparse optimization by alternating direction method. In *SPIE*, volume 8858, 2013.

[23] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, 2012.

[24] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, 2017.

[25] Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Synthesizing training images for boosting human 3d pose estimation. In *3DV*, 2017.

[26] Chi Li, Zeeshan Zia, Quoc huy Tran, Xiang Yu, Gregory D. Hager, and Manmohan Chandraker. Deep supervision with shape concepts for occlusion-aware 3d object parsing. In *CVPR*, 2017.

[27] Jiajun Wu, Tianfan Xue, Joseph J Lim, Yuandong Tian, Joshua B Tenenbaum, Antonio Torralba, and William T Freeman. Single image 3d interpreter network. In *ECCV*, 2016.

[28] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, July 2017.

[29] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[30] Tinghui Zhou, Philipp Krähenbühl, Mathieu Aubry, Qixing Huang, and Alexei A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *CVPR*, 2016.

[31] J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on PAMI*, 37(8), 2015.

[32] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[33] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[34] R. Basri and D. W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on PAMI*, 25(2), 2003.

[35] R. Ramamoorthi and P. Hanrahan. On the relationship between radiance and irradiance: Determining the illumination from images of a convex lambertian object. *JOSA*, 2001.

[36] Hao Zhou, Jose M. Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *ECCV*, 2016.

[37] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

[38] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[39] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[40] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[41] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *ACM MM*, 2014.

[42] Michaël Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. *CoRR*, abs/1312.5851, 2013.

[43] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2014.

[44] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *ACM SIGKDD*, 2006.

[45] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NIPS*, 2014.

[46] Geoffrey E. Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*, 2014.

[47] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[48] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Penksy. Sparse convolutional neural networks. In *CVPR*, 2015.

[49] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V. Oseledets, and Victor S. Lempitsky. Speeding-up convolutional neural networks using finetuned cp-decomposition. In *ICLR*, 2015.

[50] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014.

[51] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *NIPS*, 2015.

[52] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *NIPS*, 1990.

[53] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, 1993.

[54] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *NIPS*, 2015.

[55] Dong Yu, F. Seide, Gang Li, and Li Deng. Exploiting sparseness in deep neural networks for large vocabulary speech recognition. In *ICASSP*, 2012.

[56] Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *ACL-IJCNLP*, 2009.

[57] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *AISTATS*, 2011.

[58] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q.V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G.E. Hinton. On rectified linear units for speech processing. In *ICASSP*, 2013.

[59] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Technical report, Department of Computer Science, University of Toronto, 2009.

[60] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[61] Soumith Chintala. soumith/imagenet-multigpu.torch. url=https://github.com/soumith/imagenet-multiGPU.torch, 2015.

[62] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[63] D. Shahlaei and V. Blanz. Realistic inverse lighting from a single 2d image of a face, taken under unknown and complex lighting. In *FG*, 2015.

[64] Miguel Heredia Conde, Davoud Shahlaei, Volker Blanz, and Otmar Loffeld. Efficient and robust inverse lighting of a single face image using compressive sensing. In *ICCV Workshops*, 2015.

[65] B. Peng, W. Wang, J. Dong, and T. Tan. Optimized 3d lighting environment estimation for image forgery detection. *IEEE Transactions on IFS*, 12(2), 2017.

[66] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

[67] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, 1999.

[68] O. Aldrian and W. A. P. Smith. Inverse rendering of faces with a 3d morphable model. *IEEE Transactions on PAMI*, 35(5), 2013.

[69] Y. Wang, L. Zhang, Z. Liu, G. Hua, Z. Wen, Z. Zhang, and D. Samaras. Face relighting from a single image under arbitrary unknown lighting conditions. *IEEE Transactions on PAMI*, 31(11), 2009.

[70] James Booth, Epameinondas Antonakos, Stylianos Ploumpis, George Trigeorgis, Yannis Panagakis, and Stefanos Zafeiriou. 3d face morphable models "in-the-wild". In *CVPR*, 2017.

[71] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

[72] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras. Neural face editing with intrinsic image disentangling. In *CVPR*, 2017.

[73] Anh Tuan Tran, Tal Hassner, Iacopo Masi, and Gerard Medioni. Regressing robust and discriminative 3d morphable models with a very deep neural network. In *CVPR*, 2017.

[74] Hyeongwoo Kim, Michael Zollhöfer, Ayush Tewari, Justus Thies, Christian Richardt, and Christian Theobalt. Inversefacenet: Deep single-shot inverse face rendering from A single image. *ArXiv e-prints*, abs/1703.10956, 2017.

[75] Benot Frnay and Ata Kaban. A comprehensive introduction to label noise. In *ESANN*, 2014.

[76] Volodymyr Mnih and Geoffrey E. Hinton. Learning to label aerial images from noisy data. In *ICML*, 2012.

[77] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Learning from noisy labels with deep neural networks. In *ICLR*, 2015.

[78] Samaneh Azadi, Jiashi Feng, Stefanie Jegelka, and Trevor Darrell. Auxiliary image regularization for deep cnns with noisy labels. In *ICLR*, 2016.

[79] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015.

[80] Giorgio Patrini, Alessandro Rozza, Aditya Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: a loss correction approach. In *CVPR*, 2017.

[81] I. Jindal, M. Nokleby, and X. Chen. Learning deep networks from noisy labels with dropout regularization. In *ICDM*, 2016.

[82] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[83] Swami Sankaranarayanan, Yogesh Balaji, Carlos D. Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. *ArXiv e-prints*, abs/1704.01705, 2017.

[84] Kuniaki Saito, Yusuke Mukuta, Yoshitaka Ushiku, and Tatsuya Harada. Deep modality invariant adversarial network for shared representation learning. In *ICCV*, 2017.

[85] Ronen Basri, David Jacobs, and Ira Kemelmacher. Photometric stereo with general, unknown lighting. *IJCV*, 72(3), 2007.

[86] Lei Zhang and Dimitris Samaras. Face recognition from a single training image under arbitrary unknown lighting using spherical harmonics. *IEEE Transaction on PAMI*, 28(3), 2006.

[87] M. K. Johnson and H. Farid. Exposing digital forgeries in complex lighting environments. *IEEE Transactions on IFS*, 2(3), 2007.

[88] R. Ramamoorthi. Analytic pca construction for theoretical analysis of lighting variability in images of a lambertian object. *IEEE Transactions on PAMI*, 24(10), 2002.

[89] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *ArXiv e-prints*, abs/1701.07875, 2017.

[90] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6a, overview of mini-batch gradient descent. `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`, 2012.

[91] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *ArXiv e-prints*, abs/1212.5701, 2012.

[92] Akshay Asthana, Stefanos Zafeiriou, Shiyang Cheng, and Maja Pantic. Robust discriminative response map fitting with constrained local models. In *CVPR*, 2013.

[93] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3d face model for pose and illumination invariant face recognition. In *AVSS*, 2009.

[94] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. *Image Vision Computing*, 28(5), 2010.

[95] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[96] Martín Abadi, Ashish Agarwal, Paul Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[97] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[98] Yi Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011.

[99] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David W. Jacobs. Deep single-image portrait relighting. In *ICCV*, 2019.

[100] O. Aldrian and W. A. P. Smith. Inverse rendering of faces with a 3d morphable model. *IEEE Transactions on PAMI*, 35(5), 2013.

[101] Yang Wang, Lei Zhang, Zicheng Liu, Gang Hua, Zhen Wen, Zhengyou Zhang, and Dimitris Samaras. Face relighting from a single image under arbitrary unknown lighting conditions. *IEEE Trans. PAMI*, 31(11), nov 2009.

[102] Bernhard Egger, Sandro Schönborn, Andreas Schneider, Adam Kortylewski, Andreas Morel-Forster, Clemens Blumer, and Thomas Vetter. Occlusion-aware 3d morphable models and an illumination prior for face image analysis. *IJCV*, 2018.

[103] Ayush Tewari, Michael Zollöfer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Theobalt Christian. MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction. In *ICCV*, 2017.

[104] Soumyadip Sengupta, Angjoo Kanazawa, Carlos D. Castillo, and David W. Jacobs. Sfsnet: Learning shape, refectance and illuminance of faces in the wild. In *CVPR*, 2018.

[105] Amnon Shashua and Tammy Riklin-raviv. The quotient image: Class-based re-rendering and recognition with varying illuminations. *IEEE Trans. on PAMI*, 23:129–139, 2001.

[106] Xiangyu Zhu Jianzhu Guo and Zhen Lei. 3ddfa. `https://github.com/cleardusk/3DDFA`, 2018.

[107] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of Eurographics Symposium on Geometry Processing*, 2007.

[108] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[109] Arne Stoschek. Image-based re-rendering of faces for continuous pose and illumination directions. In *CVPR*, 2000.

[110] Zhen Wen, Zicheng Liu, and T. S. Huang. Face relighting with radiance environment maps. In *CVPR*, 2003.

[111] Pieter Peers, Naoki Tamura, Wojciech Matusik, and Paul Debevec. Post-production facial performance relighting using reflectance transfer. In *SIGGRAPH*, 2007.

[112] Luan Tran and Xiaoming Liu. Nonlinear 3d face morphable model. In *CVPR*, 2018.

[113] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T. Freeman. Unsupervised training for 3d morphable model regression. *CVPR*, 2018.

[114] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. In *CVPR*, 2017.

[115] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *ECCV*, 2018.

[116] Zhixin Shu, Sunil Hadap, Eli Shechtman, Kalyan Sunkavalli, Sylvain Paris, and Dimitris Samaras. Portrait lighting transfer using a mass transport approach. *ACM Transactions on Graphics*, 37(2), November 2017.

[117] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 2014.

[118] Xiangyu Zhu, Xiaoming Liu, Zhen Lei, and Stan Z. Li. Face alignment in full pose range: A 3d total solution. *PAMI*, 2017.

[119] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[120] Image-to-image translation in pytorch. `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`. Accessed:2019.

[121] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.

[122] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[123] facetracker. `http://facetracker.net/`.

[124] Hao Zhou, Xiang Yu, and David W. Jacobs. Glosh: Global-local spherical harmonics for intrinsic image decomposition. In *ICCV*, 2019.

[125] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. *Computer Vision Systems*, 1978.

[126] Jens Ackermann and Michael Goesele. A survey of photometric stereo techniques. *Found. Trends. Comput. Graph. Vis.*, 9(3-4), 2015.

[127] S. Shirdhonkar and D. W. Jacobs. Non-negative lighting and specular object recognition. In *ICCV*, 2005.

[128] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. on Graphics (SIGGRAPH)*, 2014.

[129] Balazs Kovacs, Sean Bell, Noah Snavely, and Kavita Bala. Shading annotations in the wild. In *CVPR*, 2017.

[130] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017.

[131] Ruo Zhang, Ping-Sing Tsai, J. E. Cryer, and M. Shah. Shape-from-shading: a survey. *TPAMI*, 21(8), 1999.

[132] Jean-Denis Durou, Maurizio Falcone, and Manuela Sagona. Numerical methods for shape-from-shading: A new survey with benchmarks. *CVIU*, 109(1), 2008.

[133] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19, 1980.

[134] P. N. Belhumeur, D. J. Kriegman, and A. L. Yuille. The bas-relief ambiguity. *IJCV*, 35(1), 1999.

[135] Peter Vincent Gehler, Carsten Rother, Martin Kiefel, Lumin Zhang, and Bernhard Schölkopf. Recovering intrinsic images with a global sparsity prior on reflectance. In *NIPS*, 2011.

[136] J. Shi, Y. Dong, H. Su, and S. X. Yu. Learning non-lambertian object intrinsics across shapenet categories. In *CVPR*, 2017.

[137] Wei-Chiu Ma, Hang Chu, Bolei Zhou, Raquel Urtasun, and Antonio Torralba. Single image intrinsic decomposition without a single intrinsic image. In *ECCV*, 2018.

[138] Michael Janner, Jiajun Wu, Tejas D. Kulkarni, Ilker Yildirim, and Josh Tenenbaum. Self-supervised intrinsic image decomposition. In *NIPS*, 2017.

[139] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *TPAMI*, 31(5), 2009.

[140] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[141] Q. Chen and V. Koltun. A simple model for intrinsic image decomposition with depth cues. In *ICCV*, 2013.

[142] Junho Jeon, Sunghyun Cho, Xin Tong, and Seungyong Lee. Intrinsic image decomposition using structure-texture separation and surface normals. In *ECCV*, 2014.

[143] E. Shelhamer, J. T. Barron, and T. Darrell. Scene intrinsics and depth from a single image. In *ICCV (Workshop)*, 2015.

[144] Sai Bi, Xiaoguang Han, and Yizhou Yu. An l1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM ToG*, 34(4), 2015.

[145] Tinghui Zhou, Philipp Krähenbühl, and Alexei A Efros. Learning data-driven reflectance priors for intrinsic image decomposition. In *ICCV*, 2015.

[146] Thomas Nestmeyer and Peter V Gehler. Reflectance adaptive filtering improves intrinsic image estimation. In *CVPR*, 2017.

[147] Takuya Narihira, Michael Maire, and Stella X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *ICCV*, 2015.

[148] T. Narihira, M. Maire, and S. X. Yu. Learning lightness from human judgement on relative reflectance. In *CVPR*, 2015.

[149] Lechao Cheng, Chengyi Zhang, and Zicheng Liao. Intrinsic image transformation via scale space decomposition. In *CVPR*, 2018.

[150] Seungryong Kim, Kihong Park, Kwanghoon Sohn, and Stephen Lin. Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In *ECCV*, 2016.

[151] D. Zoran, P. Isola, D. Krishnan, and W. T. Freeman. Learning ordinal relationships for mid-level vision. In *ICCV*, 2015.

[152] Zhengqi Li and Noah Snavely. Learning intrinsic image decomposition from watching the world. In *CVPR*, 2018.

[153] Zhengqi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *ECCV*, 2018.

[154] Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David Wipf. Revisiting deep intrinsic image decompositions. In *CVPR*, 2018.

[155] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[156] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[157] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[158] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *CVPR*, 2017.

[159] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[160] Roger B. Grosse, Micah K. Johnson, Edward H. Adelson, and William T. Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *ICCV*, 2009.

[161] Elena Garces, Adolfo Munoz, Jorge Lopez-Moreno, and Diego Gutierrez. Intrinsic images by clustering. *Comput. Graph. Forum*, 31(4), 2012.

[162] Qi Zhao, Ping Tan, Qiang Dai, Li Shen, Enhua Wu, and Stephen Lin. A closed-form solution to retinex with nonlocal texture constraints. *TPAMI*, 34(7), 2012.

[163] Hao Zhou, Torsten Sattler, and David W. Jacobs. Evaluating local features for day-night matching. In *Computer Vision – ECCV 2016 Workshops*, 2016.

[164] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM transactions on graphics (TOG)*, 25(3):835–846, 2006.

[165] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on PAMI*, 32(8):1362–1376, 2010.

[166] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *CVPR*, 2015.

[167] B. Zeisl, T. Sattler, and M. Pollefeys. Camera Pose Voting for Large-Scale Image-Based Localization. In *ICCV*, 2015.

[168] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *RSS*, 2015.

[169] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *ICCV*, 2003.

[170] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65(1/2), 2005.

[171] Changchang Wu, B. Clipp, Xiaowei Li, J. M. Frahm, and M. Pollefeys. 3d model matching with viewpoint-invariant patches (vip). In *CVPR*, 2008.

[172] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM J. Img. Sci.*, 2(2), 2009.

[173] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, 2002.

[174] Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. TILDE: A Temporally Invariant Learned DEtector. In *CVPR*, 2015.

[175] Andrew Richardson and Edwin Olson. Learning convolutional filters for interest point detection. In *ICRA*, 2013.

[176] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[177] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *IEEE Transactions on PAMI*, 33(1), 2011.

[178] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on PAMI*, 36(8), 2014.

[179] Tomasz Trzcinski, Mario Christoudias, Vincent Lepetit, and Pascal Fua. Learning image descriptors with the boosting-trick. In *NIPS*, 2012.

[180] Niko Suenderhauf, Sareh Shirazi, Adam Jacobson, Feras Dayoub, Edward Pepperell, Ben Upcroft, and Michael Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. In *RSS*, 2015.

[181] Bill Triggs. Detecting keypoints with stable position, orientation, and scale under illumination changes. In *ECCV*, 2004.

[182] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, 2002.

[183] N. Jacobs, N. Roman, and R. Pless. Consistent temporal variations in many outdoor scenes. In *CVPR*, 2007.

[184] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[185] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[186] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative Learning of Deep Convolutional Feature Point Descriptors. In *ICCV*, 2015.

[187] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015.

[188] Edgar Simo-Serra, Carme Torras, and Francesc Moreno-Noguer. Dali: Deformation and light invariant descriptor. *IJCV*, 115(2), 2015.

[189] D. Mishkin, J. Matas, M. Perdoch, and K. Lenc. WxBS: Wide Baseline Stereo Generalizations. In *BMVC*, 2015.