FACULTY OF INFORMATICS
CHAIR FOR COMPUTER VISION AND ARTIFICIAL INTELLIGENCE

LEARNING GEOMETRY AND SEMANTICS FOR DEEP IMAGE RESTORATION

CANER HAZIRBAŞ

Fakultät für Informatik
Technische Universität München

Learning Geometry and Semantics for Deep Image Restoration

Caner Hazırbaş

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr. Matthias Nießner

Prüfende der Dissertation:
1. Prof. Dr. Daniel Cremers
2. Prof. Dr. Ian Reid,
   The University of Adelaide

Die Dissertation wurde am 31.10.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 29.04.2019 angenommen.

Abstract

This thesis investigates possible deep learning-based solutions for three challenging computer vision problems. We first tackle reconstructing depth from focus cue and devise a suitable network architecture for the problem of *depth from focus*. To this end, we generate a large dataset composed of light-field images and co-registered depth maps to train a deep network on real-world scenarios. Our approach outperforms classical methods by a large margin and is generalizable for other datasets and cameras with a minimal effort.

Following, we present a fusion-based CNN architecture to incorporate depth into *semantic segmentation*. Our network simultaneously extract features on color and depth images independently which are then fused into color features as the network goes deeper. Our comprehensive experiments show that the proposed method is the most effective way of fusing different input modalities into one network.

Furthermore, we propose a multimodal CNN architecture that exploits pixelwise semantic labels in addition to color information and thus improves *image restoration* tasks. Our method processes semantically homogeneous image regions independently and learns to combine them for a refined output. This novel architecture requires semantic annotations only during the training phase and therefore, can perform image restoration on images without the necessity of semantic labeling.

Consequently, we discuss the limitations of the proposed approaches and provide potential directions for future research.

Zusammenfassung

Diese Dissertation untersucht mögliche Deep Learning basierte Lösungen für drei anspruchsvolle Bilverarbeitungsprobleme. Wir beschäftigen uns zunächst mit der Rekonstruktion von Tiefe aus dem Fokus und entwickeln eine geeignete Netzwerkarchitektur für das Problem der Tiefenschärfe. Zu diesem Zweck erzeugen wir einen großen Datensatz aus Lichtfeldbildern und co-registrierten Tiefenkarten, um ein tiefes Netzwerk in realen Szenarien zu trainieren. Unser Ansatz übertrifft die klassischen Methoden um ein Vielfaches und ist für andere Datensätze und Kameras mit minimalem Aufwand verallgemeinerbar.

Im Folgenden präsentieren wir eine fusions-basierte CNN-Architektur, um Tiefe in die semantische Segmentierung einzubauen. Unser Netzwerk extrahiert gleichzeitig und unabhängig voneinander Funktionen für Farb- und Tiefenbilder, die dann zu Farbmerkmalen verschmolzen werden, wenn das Netzwerk tiefer geht. Unsere umfassenden Experimente zeigen, dass die vorgeschlagene Methode der effektivste Weg ist verschiedene Eingangsmodalitäten in einem Netzwerk zu verschmelzen.

Darüber hinaus schlagen wir eine multimodale CNN-Architektur vor, die neben farblichen Informationen auch pixelweise semantische Label ausnutzt und somit Bildwiederherstellung verbessert. Unsere Methode verarbeitet semantisch homogene Bildregionen unabhängig voneinander und lernt diese zu einer verfeinerten Ausgabe zu kombinieren. Diese neuartige Architektur erfordert semantische Annotationen nur während der Trainingsphase und kann daher eine Bildwiederherstellung auf Bildern ohne die Notwendigkeit einer semantischen Markierung durchführen.

Folglich diskutieren wir die Grenzen der vorgeschlagenen Ansätze und bieten mögliche Richtungen für zukünftige Forschung.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Dr. Daniel Cremers for giving me the opportunity to join his research group and for his guidance throughout my doctoral study.

My dear colleagues, success cannot be achieved alone. Thank you all for your supports and encouragement. I was in luck working with you and I appreciated every single moment of it.

I had them and everything got just easier. I had my beloved ones. I had my dear friends who were my source of inspiration and motivation. I first give my most gratitude to Pozor Team, a gang of whom have been there and will always be there. Each and every one of them knows that I am always with them and I know that they are always with me.

You always need one that is closer, that knows more than others, that can actually be there for longer. My dear friend Bilal Porgalı, I thank you for being with me any time I needed and appreciate for all you have done for me.

This PhD would not finish without an extreme help and support I have received from her, my mentor, my master, my dear friend and my sister Laura Leal-Taixé. I never doubted and I was right. I wish nothing but the very best for you.

I would like to take the chance to thank Cristian Canton Ferrer for his great friendship and support. I really appreciate your extreme help to build my future.

More than anybody, I must thank my family for every minute of my life, for their trust on me, their love for me and most of all for their infinite support on every decision I make.

I dedicate this thesis to my family, my friends and all others who were part of my success!

Contents

List of Figures

List of Tables

List of Abbreviations

1D, 2D, 3D	n-dimensional, $n = 1, 2, \ldots$
BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DFF	Depth From Focus
DFLF	Depth From Light-field
e.g.	for example (from Latin “exempli gratia”)
ELU	Exponential Linear Unit
etc.	and more (from Latin “et cetera”)
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
i.e.	that is (from Latin “id est”)
LReLU	Leaky Rectified Linear Unit
LSTM	Long Short-term Memory
MSE	Mean Squared Error
PReLU	Parametric Rectified Linear Unit
PSNR	Peak Signal-to-noise Ratio
ReLU	Rectified Linear Unit
RMS	Root Mean Square Error
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
Tanh	Hyperbolic Tangent Function

Part I

Introduction

Chapter 1
Overview

A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine.

— Alan Turing, *Intelligent Machinery: A Report by A. M. Turing (Summer 1948)*

Visual understanding aims to imitate the human vision and therefore it has been the core element of artificial intelligence for autonomous devices. Despite the fact that it is a very challenging and complex task, the recent developments in computer vision and deep learning push the limit of ongoing research rapidly towards reaching the capabilities of a human brain.

For decades we have investigated how to extract relevant information from visual content on the images. While the major challenge is to determine the relevant encoding, so-called features, a very large variety of objects and conditions on the images such as brightness or noise raise very serious difficulties on how to define or learn these features that are relevant for the intended task.

Feature extraction methods are usually focused on computing low-level features such as edges and corners. Although these basic structural elements are distinctive for simpler tasks, *e.g.*, background-foreground image/object classification, with the increasing amount of complexity on the task, for example pixel-wise image segmentation, it is not always possible to rely only on these low-level information. Beside the local responses, more generic globally important attributes of an image have become more important. As a result feature descriptors based on basic object features, *e.g.*, SIFT [8], SURF [9], ORB [10] have been more attractive for researchers to solve more complex imaging tasks. Nonetheless, these hand-crafted features and descriptors are lacking of generality for real-world problems and thus left their place to learning based approaches.

Biologically inspired deep convolutional networks [11] have started a new era in large-scale object detection and scene understanding in the last decade. Their powerful mechanism and robust performance on real-world challenges have drastically advanced the research in computer vision and robotics.

Deep networks are composed of series of non-linear functions. Each of these functions has a large amount of parameters and therefore they heavily require a very large data to be trained on. Moreover most of the time these data should be category-labeled.

ImageNet [12] was the very first large dataset that enabled training such deep networks for the task of image classification [13]. Following years brought the necessity of preparing huge datasets for many other tasks, for example, COCO dataset [14] for large-scale object detection, segmentation, and image captioning.

Interestingly, networks trained for a specific task on a synthetic dataset had also capability to generalize to real-world scenarios. Very first deep network for optical flow, *i.e.* FlowNet [15] was able to tackle real-world challenges on images although it was essentially trained on a synthetic dataset, *i.e.* Flying Chairs, which was only generated based on 2D motion in the image plane. Later, FlyingThings3D [16] synthetic dataset enabled networks to generalize the motion estimation for scene flow.

On the other hand, this greediness allows deep networks to extract low and higher-order representations of an input image, which are more generic and robust for large scale problems. In addition, these learned features are proven to be re-usable for multiple challenges which allowed researchers to apply transfer learning from object classification to many other imaging tasks, *e.g.*, segmentation [17, 18], image-based camera localization [19, 20], depth from focus [21]. Following this paradigm I exploit deep networks for a set of problems I tackled in this thesis.

I first investigate how to recover/learn geometry from a focal stack in which the focus gradually changes from close to far objects. By having the so-called depth map of the scene, I present a novel approach to incorporate scene depth into segmentation networks that boosts the image labeling performance. Finally, I reach the goal of highly accurate image restoration technique that learns the semantics implicitly to improve the results.

1.1 Contributions of this Thesis

Main contributions of this thesis are summarized in the following:

- **Depth reconstruction from focus cue.** We tackle the classical ill-posed *depth from focus* (DFF) using convolutional neural networks. We propose a deep network architecture to process an input focal stack where the focus gradually changes from close to far objects and produce a depth map of the scene. To this end, we introduce a new large dataset, generated for the problem of DFF, to the best of our knowledge. This approach is described in Part II, Chapter 3 and presented in [21].

- **An efficient method to fuse depth into semantic segmentation.** We present a novel way of incorporating depth and RGB input modalities in neural networks. We employ our novel *fusion* architecture on the RGB-Depth input data for *semantic image segmentation*. This novel approach is described in Part II, Chapter 4 and published in [18].

- **Semantic image restoration.** We have formalized deblurring *image restoration* task in a noval multimodal convolutional neural network architecture which processes semantically homogeneous regions of an image independently and learns to

combine their output into the final result. To the best of our knowledge, our approach is the very first one that employs semantic labels during the training phase and processes the test images without the need of any semantic annotation. This method is described in Part II, Chapter 5.

1.2 Thesis Outline

This thesis is structured in 7 chapters which are grouped into three parts as following:

- **Part I:** Introduction (Chapters 1–2)
- **Part II:** Depth Prediction and Learning Semantics for Image Restoration (Chapters 3–5)
- **Part III:** Conclusions & Outlook (Chapters 6–7)

Part I: Introduction provides the motivation and overview of this thesis in Chapter 1. Chapter 2 provides a brief introduction to the theoretical fundamentals of the feed-forward and recursive networks, optimization schemes and different network components.

Part II: Depth Prediction and Learning Semantics for Image Restoration outlines the three major contributions of this thesis. In Chapter 3, we propose a deep network to reconstruct the depth information from focal stacks and to this end we introduce the largest dataset generated for the problem of *depth from focus*, to the best of our knowledge. Chapter 4 addresses how to incorporate different input modalities in the same network and introduces a novel deep network architecture for semantic image segmentation for RGB-Depth input data. This leads us to *semantic* image restoration, a novel class-aware image restoration method proposed in Chapter 5.

Part III: Conclusions & Outlook is the last part of this thesis which provides a summary and discussion on the results. Furthermore we point out the open questions for future research.

Chapter 2
Theoretical Fundamentals

Science is a differential equation. Religion is a boundary condition.

— Alan Turing, *Epigram to Robin Gandy (1954); reprinted in Andrew Hodges, Alan Turing: the Enigma (Vintage edition 1992), p. 513*

Amount of data is increasing every other day and it is now very important to have a mechanism to process these data to extract some useful information. Many machine learning algorithms are intended to process big data to retrieve the relevant information in order to develop a machine intelligence. The every-day growth on the vast amount of data bring the requirement of learning methods to be faster, more robust, scalable, reliable and more generalized for the real-world problems. Amongst many proposed learning approaches, deep neural networks are one suited algorithm to tackle challenging real-world problems.

Neural networks are constructed as a composition of non-linear functions and therefore are capable of learning complex functions. However they are highly non-convex and do not guarantee a globally optimum solution. On the other hand, although the complexity of training increases proportional to the network and data size, available large datasets make it possible to train such deep networks for many tasks to achieve state-of-the-art performance. In computer vision, neural networks showed an extraordinary performance in many tasks, *e.g.*, image classification, object recognition, scene understanding, *etc.* This achievement led researchers to dig into these non-convex, complex models to generalize them to many other tasks as well as disciplines. On the other hand, neural networks are data-dependent algorithms and their performance rely on the size and distribution of data.

A network might represent the data that it was trained on very well and this may result in diverging from the expected *true* model. In neural networks, this is known as over-fitting, refers to that every single datum in training data is perfectly represented by the model whereas it does not generalize to new *unseen* test data. Since this violates the principle of generalization, recent developments and approaches focus on resolving *over-fitting* in training.

Over-fitting can occur also due to the network architecture and it is not always straight-forward to construct a network for a specific task. Networks have many small

components that can drive them into a very bad local minimum or even cause over-fitting. While a naive solution is to manipulate the training data (data augmentation), more sophisticated approaches such as better algorithmic training schemes were also presented.

I dedicate this chapter to introduce the fundamentals of neural networks and give a review on the basic mathematical background and major concepts of deep networks, including different loss functions, training schemes, variant architectures and network components. Moreover, I will also discuss variant weight initialization techniques and regularization methods designed to avoid over-fitting.

2.1 Neural Networks

Let $\mathbf{x} \in \mathbb{R}^m$ an input vector, $\mathbf{y} \in \mathbb{R}$ output and $\theta \in \mathbb{R}^n$ the parameters, a neural network is a mapping $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$. Last layer in a network is an *output non-linearity* which is chosen such a way that the output of network aligns with the loss. For example, in multi-class object classification, it is defined as the *softmax function*:

$$\hat{p}(\mathbf{y} \mid \mathbf{x}, \theta) = \frac{\exp(f(\mathbf{x}, \theta))}{\left\| \exp(f(\mathbf{x}, \theta) \right\|_1} , \tag{2.1}$$

which scales the output vector of network to a probability distribution.

Parameters (*weights*) of the network are determined by optimizing a *loss function* $\mathcal{L} : \mathbb{R} \to \mathbb{R}$, given a dataset $\mathcal{D}$ composed of input data and their expected output. However, most of the time loss itself is not enough to generalize the model to test samples and therefore to avoid over-fitting, a regularization term $\mathcal{R}(\theta)$ that is a constraint on the weights is added to the final cost:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \theta) = \mathcal{L}(f(\mathbf{x}, \theta), \mathbf{y}) + \lambda \mathcal{R}(\theta) , \tag{2.2}$$

where λ is a weighting factor, that determines the amount of penalization on the loss.

Network f is constructed as a composition of functions. To achieve this each block of network is designed with affine transformation

$$\mathbf{W}\mathbf{x} + \mathbf{b} , \tag{2.3}$$

where $\mathbf{x} \in \mathbb{R}^n$ is input, $\mathbf{W} \in \mathbb{R}^{m \times n}$ is the *weight matrix* and $\mathbf{b} \in \mathbb{R}^m$ is the *bias* term, thereby $\theta = \{\mathbf{W}, \mathbf{b}\}$. In order to construct a highly non-linear network, after each affine transformation, a non-linear function $g : \mathbb{R}^m \to \mathbb{R}^m$ (see Section 2.1.1) is applied to the output. Network f is then constructed as a concatenation of arbitrarily many non-linear functions as follow:

$$f \equiv g_k \circ g_{k-1} \cdots \circ g_1 , \tag{2.4}$$

$$f(\mathbf{x}, \theta) = g_k(\mathbf{W}_k \cdot (g_{k-1}(\mathbf{W}_{k-1} \cdot (\cdots (g_1(\mathbf{W}_1 \cdot x + \mathbf{b}_1)) \cdots) + \mathbf{b}_{k-1})) + \mathbf{b}_k), \tag{2.5}$$

where each $h_k = f_k(\mathbf{W}_k \cdot \mathbf{x}_k + \mathbf{b}_k)$ is then considered as one layer in the neural network, and called as a *hidden layer*.

This simple architecture of a neural network is known as *feed-forward* network as the output of one layer is the input of following layer. A neural network is called "*deep*" when $k > 2$.

2.1.1 Non-linear activation functions

What makes neural networks so robust and powerful tools to approximate complex functions is how the network architecture is chosen, *i.e.* number of layers and weights in each layer and *non-linearity*. Non-linear functions are small decision mechanisms in the networks which decide when the neuron fires (the neuron is active). An activation function should be free of *vanishing gradient* problem which is an important factor to be able to train very deep networks (see Section 2.3). Amongst many non-linear activation functions most commonly used ones are discussed below.

Logistic Sigmoid function

$$\sigma(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\exp(\mathbf{x}) + 1} \ , \tag{2.6}$$

$$\sigma'(\mathbf{x}) = \sigma(\mathbf{x})(1 - \sigma(\mathbf{x})) \ , \tag{2.7}$$

transforms the input to interval of $[0, 1]$. This activation function is not zero-centered and it tends to vanish the gradient for very small and big inputs. Moreover, it is computationally expensive due to the exponential function.

Hyperbolic Tangent function

$$\mathrm{Tanh} = \frac{\exp(\mathbf{x}) - \exp(-\mathbf{x})}{\exp(\mathbf{x}) + \exp(-\mathbf{x})} \ , \tag{2.8}$$

$$\mathrm{Tanh}' = 1 - \mathrm{Tanh}^2(\mathbf{x}) \ , \tag{2.9}$$

also suffers from vanishing gradients.

Rectified Linear Unit function [22]

$$\mathrm{ReLU}(\mathbf{x}) = \max(0, \mathbf{x}) \ , \tag{2.10}$$

has been recently used in many deep networks. This simple *piecewise* liner function is fast to compute and unbounded. Although it is not differentiable at $\mathbf{x} = 0$, one can arbitrarily choose 0 or 1 to fill the point. ReLUs are also non-zero centered and they have the problem of *vanishing gradient* for $\mathbf{x} < 0$. In order to fix this *dying* ReLU problem, *leaky* ReLUs [23] and exponential linear units [24] are introduced.

Leaky-ReLU [23]

$$\mathrm{LReLU}(\mathbf{x}) = \max(0, \mathbf{x}) + \alpha \min(0, \mathbf{x}) \ , \tag{2.11}$$

where $\alpha \in [0, 1)$ is a constant factor, ensures the gradient flow for the negative values of $\mathbf{x}$, *i.e.* when the neuron is inactive. Preferably, α is set to a small value. One alternative to LReLU is the *parametric* ReLU (PReLU) [25] that allows α to be a learnable parameter for each neuron or for each layer in the network.

Exponential linear unit [24] is defined as

$$\text{ELU} = \begin{cases} \alpha(\exp(\mathbf{x}) - 1) & \text{for } \mathbf{x} \leq 0 \\ \mathbf{x} & \text{for } \mathbf{x} > 0 \end{cases} \tag{2.12}$$

$$\text{ELU}' = \begin{cases} \text{ELU}(\mathbf{x}) + \epsilon & \text{for } \mathbf{x} \leq 0 \\ 1 & \text{for } \mathbf{x} > 0 \end{cases} \tag{2.13}$$

ϵ is a positive constant controls the saturation value for the negative inputs. ELUs lead to faster convergence and produce more accurate results. The function has natural gradient close to 0 and therefore enables faster learning compared to ReLUs or LReLUs.

Maxout unit [26] computes the function

$$h_i(\mathbf{x}) = \max_{j=[1,d]} z_{ij}, \tag{2.14}$$

where $\mathbf{x} \in \mathbb{R}^m$, $z_{ij} = \mathbf{W}\mathbf{x} + \mathbf{b}$ and $\mathbf{W} \in \mathbb{R}^{m \times n \times d}$ and $\mathbf{b} \in \mathbb{R}^{n \times d}$. For $d = 2$, a *maxout* unit is then a generalized version of ReLU and LReLU:

$$\max(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1, \mathbf{W}_2\mathbf{x} + \mathbf{b}_2). \tag{2.15}$$

Note that number of learnable parameters are increased by a factor of 2. As stated in [26], a maxout unit can be considered as a piecewise linear approximation of an arbitrary convex function and maxout networks are able to learn the activation functions of each hidden units.

2.1.2 Weight initialization

Saturation, when the neuron becomes inactive, can still be an issue for many of the functions discussed in Section 2.1.1 depend on the initialization of parameters, *i.e.* *weights* $\mathbf{W}$ and *biases* $\mathbf{b}$.

To ensure the gradient flow during back-propagation (see Section 2.3.3), different weight initialization techniques have been proposed to prevent networks from the *vanishing* or *exploding* gradient problem. In this section, I give a brief summary of different initializations proposed for variant activation functions.

A simple and intuitive way is to initialize the weights with zero. This will cause the network to be a linear model as the neurons will compute the same output, hence the same gradient during the training. This is not a desired behavior. In order to guarantee of learning complex functions, we must ensure that the weights might have an arbitrarily

different value so that preferably half of the neurons becomes inactive while the other half stays active.

Therefore, weights are usually randomly sampled from a defined distribution. Very first choice is the standard normal distribution $\mathcal{N}(\mu, \sigma^2)$ where mean $\mu = 0$ and variance $\sigma^2 = 1$. In order to break the symmetry, one can sample the weights from this distribution with a small σ. Nevertheless, a layer that has small weights will compute very small gradients during training and therefore gradients will vanish. With big values of σ the network will compute big gradients, hence the *exploding* gradient problem will occur.

In order to avoid gradient issues, Xavier and Bengio [27] proposed to choose the variance of the distribution depend on the number of inputs and outputs of each neuron,

$$\text{Var} = \frac{2}{n_{in} + n_{out}} \ , \tag{2.16}$$

so that forward and backward propagated information stay in a reasonable range. *Xaiver* initialization initially designed for Tanh and sigmoid activation functions to prevent gradients become 0, *i.e.* where the function gets linear.

On the other hand, for most of the recent activation functions, He *et al.* [25] proposed a minor modification as the half of the ReLU activation is zero and the size of weight variance should be doubled to keep the input/output signal's variance constant,

$$\text{Var} = \frac{2}{n_{in}} \ . \tag{2.17}$$

Neural networks are non-convex functions and therefore weight initialization of a neural network has a major impact on its performance. However, no matter how the weights are initialized, due to backpropogation, there is no guarantee to keep the input/output distribution same through the layers. Input/output distribution of each layer change as the weights are updated. In order to reduce this *internal covariance shift*, Goodfellow and Szegedy [28] proposed a network layer, called *batch normalization*.

Batch normalization [28] normalizes the input to have zero mean and variance of 1. For a layer with an input $\mathbf{x} \in \mathbb{R}^{b \times d}$, each d dimension of this input is normalized as

$$\mathbb{E}[\mathbf{x}^{\{d\}}] = \frac{1}{b} \sum_{i}^{b} \mathbf{x}_i^{\{d\}} \tag{2.18}$$

$$\text{Var}[\mathbf{x}^{\{d\}}] = \frac{1}{b} \sum_{b}^{b} (\mathbf{x}_i^{\{d\}} - \mathbb{E}[\mathbf{x}^{\{d\}}])^2 \tag{2.19}$$

$$\hat{\mathbf{x}}^{\{d\}} = \frac{\mathbf{x}^{\{d\}} - \mathbb{E}[\mathbf{x}^{\{d\}}]}{\sqrt{\text{Var}[\mathbf{x}^{\{d\}}]}} \ . \tag{2.20}$$

Nevertheless, normalizing input may change what network can represent and therefore in order to allow networks learn identity transformation, normalized output is scaled $\gamma^{\{d\}}$ and shifted $\beta^{\{d\}}$ with learnable parameters

$$\mathbf{y}^{\{d\}} = \gamma^{\{d\}}\hat{\mathbf{x}}^{\{d\}} + \beta^{\{d\}} \ . \tag{2.21}$$

At each iteration of training input $\mathbf{x}$ is normalized with its mean and variance. γ and β are parameters of the network that are learned during training. At inference, running mean and variance computed over all training samples are used to normalize the test inputs.

Instance normalization [29] is a variant of batch normalization and samples in a batch are normalized independently as

$$\hat{\mathbf{x}}^{\{id\}} = \frac{\mathbf{x}^{\{id\}} - \mathbb{E}[\mathbf{x}^{\{id\}}]}{\sqrt{\mathrm{Var}[\mathbf{x}^{\{id\}}]}} \ . \tag{2.22}$$

As stated in [29] instance normalization prevents instance-specific mean and covariance shift and thus simplifies the learning process. At inference time instance normalization is also applied for each test sample.

Bias initialization. In most cases, initialization of biases is depend on the weights, however, with the aforementioned advanced methods, that are symmetry breaking techniques, biases are initialized with zero. In some cases, to allow neurons to fire in the beginning of training, biases are set to a small value wherever ReLU activation is used.

2.2 Regularization

A network is trained with a defined loss function on the expected and predicted output, by minimizing the cost function in the form of Equation (2.2). Regularization, $\mathcal{R}(\theta)$, is necessary to enforce a prior on the parameters as well to avoid over-fitting on the training data and thereby almost all network trainings involve a regularization term in the loss function. This form of regularization is the most common in the literature and in neural networks ℓ_1 and ℓ_2 regularizations, a.k.a *weight decay*, are the most practiced ones. They are both convex functions, although, they lead to different results.

ℓ_1-norm on the weights,

$$\mathcal{R}(\theta) = \sum_i \|\theta_i\|_1 \ , \tag{2.23}$$

ensures the sparsity in the weight vector by enforcing most of the weights to be zero or very small values. This has a certain impact on avoiding over-fitting. Moreover, due to sparsity, ℓ_1-norm has built-in feature selection property, which favors only some of the weights to have a large value. On the other hand, this function is not differentiable at zero and one of the sub-gradients (mostly zero) is chosen during the weight update.

ℓ_2-norm, in contrast,

$$\mathcal{R}(\theta) = \sum_i \theta_i^2 \ , \tag{2.24}$$

is differentiable at every point. But, it does not produce sparse outputs, thus, has no implicit feature selection. Nevertheless in most cases weights are penalized with ℓ_2 regularization so that the output will not only depend on a few nodes.

Data augmentation. In practice, careful weight initialization and a well-chosen regularizer are not enough to train a very deep network due to over-fitting and vanishing gradients. One simple solution to this is to alter the input data -augmented input- so that the seen examples by the network represents a better distribution. Increasing variety in data decreases over-fitting, hence improves the overall performance.

Data augmentation may involve random transformations on the input signal such as rotation, scale, translation, or projective transformations, random noise, contrast or brightness change in case of images.

Dropout [30] is an interesting choice of regularization that is implemented as a layer in neural networks and it is based on *dropping* randomly chosen weights. With dropout, feed-forward operation at layer k with input x_k, weights W_k and bias b_k is performed as

$$r_k \sim \text{Bernouille}(p) \tag{2.25}$$
$$\hat{\mathbf{x}}_k = r_k * \mathbf{x}_k \tag{2.26}$$
$$\mathbf{y}_k = g(\mathbf{W}_k \hat{\mathbf{x}}_k + \mathbf{b}_k) \ , \tag{2.27}$$

where r_k is sampled from a Bernoulli distribution with a probability of p. As the input is scaled with a factor of p during training, at test time input should be scaled with $1/p$.

Dropout is essentially useful to learn an ensemble of many different functions since at each iteration of training, a randomly chosen set of weights are updated. It is a strong regularization on weights and many recent approaches still benefit from it. Dropout can be placed after any layers of the network f. Another application of dropout is to use it also in the *output non-linearity*. DropMax [31] applied dropout to the *softmax* classifier in Equation (2.1) for multi-class training and drops the activations of non-target classes,

$$\hat{p}(\mathbf{y} \mid \mathbf{x}, \theta) = \frac{\exp(r f(\mathbf{x}, \theta)) + \epsilon}{\left\| \exp(r f(\mathbf{x}, \theta) \right\|_1 + c\,\epsilon} \ , \tag{2.28}$$

where c is the total number of target classes.

In practice, dropout is only applied during training to preserve the deterministic results at inference. However, recent approaches [32–34] exploit dropout during test time to estimate *uncertainty* on the result. Input is feed-forwarded through the network a couple of times. As the output slightly changes due to dropout, mean of the predicted outputs is the final result and variance over these predictions gives the *uncertainty* on the result.

2.3 Training networks: minimizing a non-convex cost function

As the distribution of the data $P(\mathbf{x}, \mathbf{y})$ is unknown, optimizing cost functions as in Equation (2.2) is solved using the principle of *empirical risk minimization* and it is performed by averaging the loss over training samples. Minimization guarantees a set of optimal parameters where the loss $\mathcal{L}$ in Equation (2.2) is minimum:

$$f = \arg\min_{f} \sum_{i} \mathcal{L}(f(\mathbf{x}_i, \theta), \mathbf{y}_i, \theta). \tag{2.29}$$

Neural networks are highly non-convex functions and therefore the optimal solution is not unique. This is also the main reason why the weight initialization is very important to reach a better local minimum.

In neural networks, training has been performed using the variants of *gradient descent* optimization scheme.

2.3.1 Gradient descent

Gradient descent is a first-order iterative optimization algorithm to find the global minimum of a convex function. But, it can also be used in neural networks with no guarantee of a global minimum. Gradient descent iteratively updates the parameters in the negative direction of the loss function with respect to the parameters. An update step in gradient descent is performed as

$$\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}(\theta) \ . \tag{2.30}$$

γ is called step size or learning rate and it determines the speed of convergence. Considering a convex function, very small values of *gamma* leads to very slow convergence while bigger values of *gamma* might cause oscillation and moreover, in case of non-convex functions, current local minimum might be left to a plateau or even to a worse local minimum. How to set the learning rate is still an open question, however, current methods usually use a learning rate scheduler as explained in Section 2.3.5.

2.3.2 Stochastic Gradient Descent

The major challenge of using *gradient descent* is to calculate the derivatives on the entire dataset. As this is not practical, neither possible due to hardware limitations, instead, *stochastic gradient descent* (SGD) [35] is used in which the loss computed on a small proportion of the data (batch) and at each iteration this batch is randomly chosen from the dataset. Then parameters are updated over this batch as

$$\theta \leftarrow \theta - \frac{\gamma}{B} \sum_{i}^{B} \nabla_\theta \mathcal{L}(\theta) \ . \tag{2.31}$$

2.3.3 Backpropagation

Another concept using gradient descent to minimize a function composition is *backpropagation*, applying chain-rule in the gradient calculation. In neural networks, parameters at each layer must be updated with a gradient, propagated back from their following layer. For any weight in layer $k-1$ of f, derivative of the loss function is then computed as follow:

$$\frac{\partial \mathcal{L}}{\partial \theta_{k-1}} = \frac{\partial \mathcal{L}}{\partial g_k} \frac{\partial g_k}{\partial g_{k-1}} \frac{\partial g_{k-1}}{\partial \theta_{g-1}} \ . \tag{2.32}$$

This update rule follows as same for all other parameters in all layers, too. Considering this formula, one can easily conclude that in case of small gradients, due to the series of multiplication, final gradient for a specific weight might vanish and therefore, gradient descent would stop updating that parameter. This is known as **vanishing gradient**. Conversely, gradients might have very big magnitude so does the final gradient, and this is referred as **exploding gradient**. Hence, weight initialization and regularization have important role to prevent networks from having vanishing or exploding gradients.

2.3.4 SGD-based Optimization Schemes

Stochastic gradient descent is a common optimization method in neural networks, however, due to strong dependency on the learning rate and initial state, it can be intractable to optimize a highly non-convex cost function. A set of optimization schemes based on SGD have been proposed to overcome the issues SGD faced. SGD with momentum, Adadelta [36], Adagrad [37], ASGD [38], L-BFGS [39], Rprop [40], ADAM [41], ADAMAX [41] and RMSProp [42] are commonly used optimization schemes.

In the following, I briefly explain the momentum, ADAM and RMSProp optimization schemes as they are the most practiced ones.

Momentum accumulates the gradients over time in order to update the parameters with a smoother estimate of their accumulated gradients. This also reduces the chance of a sudden decrease or increase on the parameters. A momentum step is defined as

$$\nu_t \leftarrow \mathbf{m} \cdot \nu_{t-1} + \gamma \cdot \nabla_\theta \mathcal{L}(\theta) \tag{2.33}$$

$$\theta_r \leftarrow \theta_{t-1} - \nu_t \ , \tag{2.34}$$

where ν is the accumulated momentum and $\mathbf{m} \in (0, 1]$ is a decay factor to determine how much of the previously accumulated gradients are incorporated into the current update. Momentum can also be used within SGD, ADAM as well as RMSProp as it is a simple way of preventing networks from weight explosion.

Adam [41] is a first-order gradient-based optimization similar to SGD. However, it is based on the adaptive estimates of lower-order moments which is an internal mechanism

to adapt the learning rate at each iteration of training and therefore it is less dependent on the initial learning rate.

Let m and v first and second order moment estimates and $\beta_1, \beta_2 \in [0, 1)$ are the exponential decay rates for the moment estimates, respectively, one step of ADAM optimizer at a step t is computed as

$$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla_\theta \mathcal{L}_t \tag{2.35}$$
$$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot v_t^2 \tag{2.36}$$
$$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t) \tag{2.37}$$
$$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t) \tag{2.38}$$
$$\theta_t \leftarrow \theta_{t-1} - \gamma\, \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) \ , \tag{2.39}$$

where power of β_1 and β_2 to t is denoted as β_1^t, β_2^t. ADAM is efficient for the problems having large amount of data or parameters, and invariant to diagonal rescale of the gradients. It is an appropriate scheme for problems with noisy or sparse gradients. At the same time, hyper-parameters of the ADAM optimizer is more intuitive and hence requires little tuning.

RMSProp [42] is another optimizer which has also internal mechanism to adapt the learning rate over time. At each iteration t, gradients are divided by a running average of their recent magnitude. Let $\nabla_\theta \mathcal{L}$ be the gradient of parameter θ, RMSProp step is formulated as

$$\mathbb{E}[\nabla_\theta^2 \mathcal{L}]_t \leftarrow 0.9\, \mathbb{E}[\nabla_\theta^2 \mathcal{L}]_{t-1} + 0.1\, \nabla_\theta^2 \mathcal{L}_t \tag{2.40}$$
$$\theta_t \leftarrow \theta_{t-1} \frac{\gamma}{\sqrt{\mathbb{E}[\nabla_\theta^2 \mathcal{L}]_t + \epsilon}} \nabla_\theta \mathcal{L}_t \ . \tag{2.41}$$

Although ADAM and RMSProp are both adapting the learning rate, it is expected that the network converges to a minimum over time and therefore a learning rate scheduler is also used to decrease the learning rate in order to make sure that network stays in the same local minimum towards the end of training.

2.3.5 Learning rate scheduler

In neural networks, learning rate, step size in *Gradient Descent*, is a crucial hyper-parameter which must be carefully chosen. However, there is no existing method determining the learning rate and one must hand-tune this parameter for each training. On the other hand, considering the stochastic gradient descent, a large learning rate is necessary to start the training so that network is driven into a local minimum. But, in the very last steps of the training, network parameters should not be updated as much they are in the beginning. For this reason, usually a learning rate scheduler is used to reduce the rate. There are a couple of standard algorithms and I present here the most

significant ones.

Constant learning rate is rarely used as it brings the difficulty to find the most appropriate learning rate which is well-suited for the entire period of training.

Time-based decay drops the initial learning rate γ_0 with a decay factor ρ as

$$\gamma_t \leftarrow \frac{\gamma_0}{1 + \rho \cdot t} \ , \tag{2.42}$$

at each iteration t during training.

Step decay decreases the learning rate after each epoch $\mathbf{e}$ with a step size of $\mathbf{s}$

$$\gamma_t \leftarrow \gamma_0 \cdot \rho \left\lfloor \frac{\mathbf{e}}{\mathbf{s}} \right\rfloor \ . \tag{2.43}$$

Exponential decay drops the initial learning rate exponentially to zero towards the end of training

$$\gamma_t \leftarrow \gamma_0 \cdot \exp(-\rho \cdot t) \ . \tag{2.44}$$

2.4 Regression and Classification with Neural Networks

Neural networks are capable of solving both regression and classification problems. As networks' output can have arbitrary shape, regression and classification problems are formulated with a defined loss function for each task.

Regression loss can be defined as ℓ_1-norm (ℓ_1 distance in Equation (5.1)) between the network's prediction and the groundtruth. Another choices are ℓ_2-norm (Equation (3.7)) and Huber loss [43], which is less sensitive to the outliers than squared error loss

$$\mathcal{L}(f(\mathbf{x}, \theta), \mathbf{y}) = \begin{cases} \frac{1}{2}(f(\mathbf{x}, \theta) - \mathbf{y})^2 & \text{for } \|f(\mathbf{x}, \theta) - \mathbf{y}\|_1 \leq \zeta \\ \zeta \|f(\mathbf{x}, \theta) - \mathbf{y}\|_1 - \frac{1}{2}\zeta^2 & \text{otherwise} \ . \end{cases} \tag{2.45}$$

Classification loss, on the other hand, needs to be defined on a category-labeled output. Background-foreground labeling (two class classification) loss can be defined as *sigmoid function* (Equation (2.6)) where the output $f(\mathbf{x}, \theta)$ (thresholded with 0.5) and the groundtruth label $\mathbf{y}$ are in the set of $\{0, 1\}$ or as hinge loss [44] where the groundtruth label $\mathbf{y} \in \{-1, +1\}$

$$\mathcal{L}(f(\mathbf{x}, \theta), \mathbf{y}) = \max(0, 1 - \mathbf{y} \cdot f(\mathbf{x}, \theta)) \ . \tag{2.46}$$

In case of multi-label segmentation, common practice is to use *multinomial logistic regression*, a.k.a cross-entropy loss, which minimizes the Kullback–Leibler divergence [45]

$$\mathcal{L}(f(\mathbf{x},\theta),\mathbf{y}) = -\sum_{c} \log(\hat{p}(\mathbf{y}=c \mid \mathbf{x},\theta)) \ , \tag{2.47}$$

where c is the class label and $\hat{p}$ is the probability estimated with *softmax function* (Equation (2.1)).

Class-balancing. In most of the cases, training set does not have uniform distribution over the classes and therefore aforementioned loss functions cannot cope with the *unbalanced* data. While augmenting more samples for less frequent classes will improve the performance, another solution is to adjust the loss function depend on the class-frequency by weighting the loss for each class.

$$\mathcal{L}(f(\mathbf{x},\theta),\mathbf{y}) = -\sum_{l}^{c} w_l \cdot \log(\hat{p}(\mathbf{y}=l \mid \mathbf{x},\theta)) \ , \tag{2.48}$$

where $w_l \in \mathbb{R}$ defined for each class. Class weights are usually determined from training data by computing the frequency of each class.

2.5 Convolutional Neural Networks

Neural networks are powerful learning methods that learn the feature representations directly from data, hence require no handcrafted features. They build the hierarchical construction of features which yield better distinction among different categories and thereby improve the performance. However, they are not invariant to different sized inputs and with the increased input size and many hidden layers, networks require a longer time and larger data to converge as the number of parameters also increases drastically. Considering a very large input $\mathbf{x}$ in Equation (2.3), weight vector $\mathbf{W}$ also grows and therefore network's efficiency decays fast. Moreover, computational expense becomes intractable because of limited resources (data and memory).

Neural networks became more popular in computer vision because of two important factors, *i.e.* 1) recent developments in the graphical computational units (GPUs) to train deep networks with billions of parameters on very large datasets, and 2) introduction of convolutional networks [46].

Biologically inspired convolutional networks have been proposed for visual pattern recognition. They are capable of recognizing patterns based on their shapes regardless of their position on the input. Very first application of a convolutional networks was proposed for hand-written digit recognition [47]. However, AlexNet [13] was the breakthrough in the field. This very deep convolutional network, trained on ImageNet [12] opened a new era in computer vision.

Convolutional networks are based on simple convolution operations in which the kernels are composed of learnable weights. Main idea of a convolution operation is to learn a set of feature kernels at each layer of the network instead of a large transformation matrix, W. Herewith, a fixed kernel size drastically reduce the number of weights in

a network and bring the flexibility of learning different convolution kernels for various visual patterns.

Convolution operation for an input $\mathbf{x} \in \mathbb{R}^{m \times n \times d}$ at any layer, is performed as convolving the input for each convolution kernel $\mathbf{W}_k$ and then adding the corresponding bias $\mathbf{b}_k$

$$\mathbf{y}_k = g_k(\mathbf{W}_k * x + \mathbf{b}_k) \ , \tag{2.49}$$

where $\mathbf{W}_k \in \mathbb{R}^{j \times l \times d}$ is a convolution kernel has size of $j \times l$ and $\mathbf{b}_k$ is the corresponding bias. Output of a convolutional layer $\mathbf{y}_k$ is called "feature map". It is still necessary to use a "fully connected" layer which transforms the feature map into one dimensional hidden representation or to final output vector by applying a matrix multiplication as in Equation (2.3).

Convolution layers have a couple of parameters must be predefined. For example, convolution kernel size or stride, which determines how sparse a convolution should be applied on the input. Although, a convolution operation is performed at each pixel on an input image, this is computationally very expensive and moreover it is intuitively expected that information which the pixels in a close vicinity provide is redundant. This also holds for the intermediate layers in a network. Hence, stride parameter is used to reduce noise on the computed feature maps.

Padding is a necessary operation applied before convolution to keep the size of an input same after convolution. *Padding* is defined as extending the input by half size of a kernel on both sides on an image. It can be *zero padding*, adding zeros on the image boundaries or more sophisticated approach, *e.g.*, mirroring -replicating the borders on the image-. Mirroring the borders avoid artifacts on the feature maps.

2.5.1 Pooling

Pooling is a commonly used layer in convolutional networks to reduce the variance among features and computational cost. Pooling is a key operation to make the network *translation invariant*. It downsamples the feature map, either by taking the average or maximum of features in a neighborhood. While *average pooling* smooths out features, *max pooling* keeps the most dominant ones, such as strong edges. Feature map is downsampled by the size of pooling kernel since there is no padding and stride of a pooling set to kernel size. In order not to loose information, generally 2×2 max-pooling is performed after a set of convolution layer.

2.5.2 Transposed convolution

Convolutional network architectures composed of only convolution and pooling layers, *e.g.*, AlexNet [13] downsample the input to a lower spatial dimension so that the final embedding of an input has 1×1 resolution. This is reasonable for image or object classification tasks. However, for pixel-wise outputs, *e.g.*, dense depth estimation or

semantic image labeling, network's output is expected to have the same size with the input. Upsampling methods such as nearest neighbor, bilinear or bicubic interpolation are possible choices to resolve this. But these methods tend to over-smooth the activation (feature) maps and hence the object boundaries in the final result.

Therefore an upsampling layer is implemented to reconstruct the full image resolution. These layers keep strong activations as the feature maps are upsampled. To achieve this, in convolutional networks, a *deconvolution layer* -inverse of convolution- is implemented. This layer is also called *transposed convolution* and it has learnable kernels same as convolution. Different upsampling layers are introduced in the following section.

2.6 Convolutional Auto-Encoders

Auto-encoders are unsupervised neural networks that aim to reconstruct the input from an embedding. They are composed of *encoder* which is a set of layers downsample the input and *decoder* that is a mirrored encoder. Decoder layers share the weights of encoder and therefore allow to reconstruct the input from an *embedding* feature vector. These networks do not require a hand-annotated dataset and hence can be trained on any sort of inputs.

A convolution auto-encoder network (see Figure 3.6) can also be implemented to reconstruct the input as well with convolution layers in encoder and deconvoloution layers in decoder. Advantage of this is that one can use these networks for any sort of supervised tasks which require pixel-wise labeling such as image restoration, normal or depth estimation, semantic labeling, *etc.* On the other hand, compared to an auto-encoder, in convolutional auto-encoders decoder does not necessarily share the weights with encoder.

Decoders usually mirror the encoder part of the network and there are different upsampling strategies. For example, *unpooling* [48] inverts the max-pooling operation. For a 2×2 unpooling, each activation from the input map is replaced to top-left of the output 2×2 patch in the output feature map where the rest of the map is filled with zero. More sophisticated solution is to keep the maximum indices while applying *pooling* in the encoder and then using these indices in the corresponding *unpooling* in the decoder to replace the activation in the output feature map [17, 49]. Advantage of *unpooling* is that there are not learnable parameters, but it may require extra memory to store the indices.

2.7 Recurrent Neural Networks

Both fully connected and convolutional networks are feed-forward networks and therefore cannot exploit the information in sequential data. On the other hand, recurrent neural networks (RNNs) are a type of neural networks that process time-series data to find the similar patterns in time via so-called **memory** blocks. This is necessary to process sequential data, for example text, audio or video. Each unit (cell) in an RNN stores the memory in its *hidden state* and this intermediate feature vector is passed to unit itself

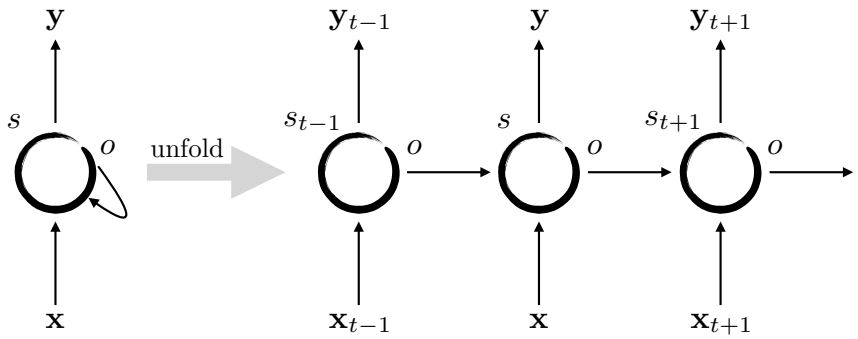


Figure 2.1: Basic RNN-cell. History, s , and internal output, o, are collected over a time series.

again with the next input. This recursion ensures that the memory is propagated to the next inputs.

A recurrent unit (RNN-cell) and its unfolded version are demonstrated in Figure 2.1. A basic RNN-cell computes an output o_t and a hidden state s_t at a step t as

$$s_t = \text{Tanh}(\mathbf{U} \cdot s_{t-1} + \mathbf{W} \cdot \mathbf{x}_{t-1} + \mathbf{b}) \tag{2.50}$$

$$o_t = \mathbf{V} \cdot s_t \tag{2.51}$$

$$\mathbf{y}_t = \text{softmax}(o_t) \ , \tag{2.52}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{m \times n}$, $\mathbf{U} \in \mathbb{R}^{m \times m}$, $s \in \mathbb{R}^m$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{V} \in \mathbb{R}^{k \times m}$ and $\mathbf{y} \in \mathbb{R}^k$. Hidden state s is a history or a summary collected over a series of inputs. These parameters are learned during training from the data with a special type of back-propagation method.

Backpropagation through time (BPTT) [50–52] is derived for recurrent neural networks. Given a sequence of inputs, BPTT is performed by inputting the sequence and computing the input-output pairs at each time step and then unrolling the network and updating the parameters based on the time step. This process is repeated until the network converges (loss is minimum) same as in gradient descent. It is a quite fast approach and allow mixture of different network architectures to be trained end-to-end. However BPTT may cause network to stuck in poor local minima and hence drastically drop the network performance.

Besides difficulty of training RNNs with BPTT, they also suffer from the *vanishing gradient* problem due to the *Tanh* function in the *hidden state* update in Equation (2.50). Although, an RNN-cell can easily capture the patterns repeated for a small period of time on the input sequence because of the *vanishing gradient* RNNs are not easily trainable for longer sequences. In order to improve the memory capacity of RNN-cells to longer sequences and to tackle *vanishing gradient* during training, new type of memory mechanisms, *i.e.* Long Short-term Memory (LSTM) unit and Gated Recurrent Unit

(GRU) are proposed. In the following sections, I discuss the differences between these units and give an overview based on http://colah.github.io/posts/2015-08-Understanding-LSTMs/, https://isaacchanghau.github.io/post/lstm-gru-formula/.

2.7.1 Long Short-term Memory (LSTM)

LSTM units [53] are variants of RNN-cells and compose of *input*, *output* and *forget* gates. Let i, o, f denote the input, output and forget gate, respectively, and $\odot$ denote the Hadamard product (element-wise multiplication), forward-pass in an LSTM cell is computed as

$$i_t = \sigma(\mathbf{U}^i \cdot \mathbf{x}_t + \mathbf{W}^i \cdot s_{t-1} + \mathbf{b}^i) \tag{2.53}$$
$$f_t = \sigma(\mathbf{U}^f \cdot \mathbf{x}_t + \mathbf{W}^f \cdot s_{t_1} + \mathbf{b}^f) \tag{2.54}$$
$$o_t = \sigma(\mathbf{U}^o \cdot \mathbf{x}_t + \mathbf{W}^o \cdot s_{t_1} + \mathbf{b}^o) \tag{2.55}$$
$$\tilde{C}_t = \text{Tanh}(\mathbf{U}^g \cdot \mathbf{x}_t + \mathbf{W}^g \cdot s_{t_1} + \mathbf{b}^g) \tag{2.56}$$
$$C_t = \sigma(f_t \odot C_{t-1} + i_t \odot \tilde{C}_t) \tag{2.57}$$
$$s_t = \text{Tanh}(C_t) \odot o_t \ . \tag{2.58}$$

Input gate controls how much of the input should pass into LSTM memory and output gate controls how much of the current output should be incorporated in the hidden state. LSTMs differ from RNNs in terms of *memory* implementation. Hidden state of an LSTM cell is computed based on the previous and current internal *memories*, $\tilde{C}_t, C_t$ as well as the output o_t. LSTM units learn to adapt the *memory* to the input sequence, which allows network to forget the history completely or remember partially. They are more robust to the length of the sequence compared to RNNs and easier to train with BPTT.

2.7.2 Gated Recurrent Unit (GRU)

GRUs [54] are also a variant of RNNs and in comparison to LSTMs they are simplified and have only *reset* and *update* gates, denoted respectively as r and z in Equations (2.59, 2.60). GRUs merge the input and forget gates to a reset gate and merge also cell states with hidden states to make changes. An update on the hidden state is formalized as

$$z_t = \sigma(\mathbf{U}^z \cdot \mathbf{x}_t + \mathbf{W}^z \cdot s_{t-1} + \mathbf{b}^z) \tag{2.59}$$
$$r_t = \sigma(\mathbf{U}^r \cdot \mathbf{x}_t + \mathbf{W}^r \cdot s_{t-1} + \mathbf{b}^r) \tag{2.60}$$
$$\tilde{s}_t = \text{Tanh}(\mathbf{U}^h \cdot \mathbf{x}_t + (r_t \odot s_{t-1}) \cdot \mathbf{W}^h + \mathbf{b}^h) \tag{2.61}$$
$$h_t = (1 - z_t) \odot s_{t-1} + z_t \odot \tilde{h}_t \ . \tag{2.62}$$

Part II

Depth Prediction and Learning Semantics for Image Restoration

Chapter 3
Depth Reconstruction from Focus Cue

Estimating depth from a single image is an ill-posed problem and therefore, it is usually hard and intractable to recover the geometry. Although modeling 3D objects to minimize a complex cost function or training a deep network on stereo pairs as in [55] might improve the performance, more observations, *e.g.*, multiple views of the scene from different view points, are required to reconstruct the absolute depth of objects from the camera. One other interesting choice is to recover depth from a focal stack. *Depth from focus* (DFF) is a classical ill-posed inverse problem in computer vision. DFF methods intends to recover depth at each pixel based on the focal setting which exhibits maximal sharpness. However it is not obvious how to reliably estimate the sharpness level, particularly in low-textured areas. In this chapter, we propose an end-to-end learning method, which we name 'Deep Depth From Focus' and a large DFF dataset to train a deep network. In order to obtain a significant amount of focal stacks with groundtruth depth, we propose to leverage a light-field camera with a co-calibrated RGB-D sensor. This allows us to digitally create focal stacks of arbitrary sizes. We generate a dataset 25 times larger than other existing benchmarks and it enables the use of machine learning for the DFF problem. In our extensive experiments we exhibit the efficiency and high performance of our proposed method 'DDFFNet' compared to state-of-the-art approaches and provide an insight how a network can be used as a sharpness level in DFF. Moreover, we present the results of our best model on a focal stack, captured by a smartphone, showing the promising usage of our approach on mobile devices. This work has been published in [21].

3.1 Introduction

The goal of *depth from focus* (DFF) is to reconstruct a pixel-accurate depth map given a stack of images with gradually changing optical focus. The key observation is that a pixel's sharpness is maximal when the object it belongs to is in focus. Hence, most methods determine the depth at each pixel by finding the focal distance at which the contrast measure is maximal. Nonetheless, DFF is an ill-posed problem, since this assumption does not hold for all cases, especially for textureless surfaces where sharpness cannot be determined. This is why most methods rely on strong regularization to obtain meaningful depth maps which in turn leads to an often oversmoothed output.

While spatial smoothness is a rather primitive prior for depth reconstruction, with the advent of convolutional neural networks (CNNs) we now have an alternative technique

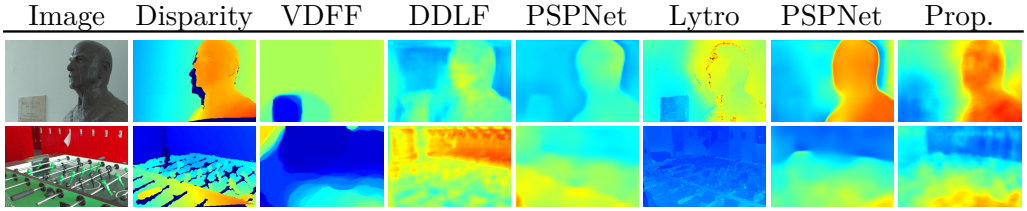


Figure 3.1: Qualitative results of the DDFFNet versus state-of-the-art methods. Results are normalized by the maximum disparity, which the focal stacks are generated on (0.28 pixel). Warmer colors represent closer distances. Best viewed in color.

to resolve classical ill-posed problems such as semantic segmentation [18, 49, 56–58] or optical flow estimation [15, 16]. The underlying expectation is that the rather naive and generic spatial smoothness assumption used in variational inference techniques is replaced with a more object-specific prior knowledge absorbed through huge amounts of training data.

A big strength of CNNs is their ability to extract meaningful image features, and correlate pixel information via convolutions. Our intuition is that a network will be able to find the image in the focal stack at which a pixel is maximally sharp, thereby correlating focus and depth. We therefore, propose to tackle the task of depth from focus using end-to-end-learning. To that end, we create the first DFF dataset with real-world scenes and groundtruth measured depth. In order to obtain focal stacks in a reliable and fast way, we propose to use a light-field camera. Also called plenoptic camera, it allows us to obtain multi-view images of a scene with a single photographic exposure. All-in-focus images as well as focal stacks can be recovered digitally from a light-field image. Using this new dataset, we perform end-to-end learning of the disparity given a focal stack.

3.1.1 Contributions

We present *Deep Depth From Focus Network* (DDFFNet), an auto-encoder-style convolutional neural network that outputs a disparity map from a focal stack. To train such a net, we create a dataset with 720 light-field images captured using a plenoptic camera, *i.e.* Lytro ILLUM, covering 12 indoor scenes. Given a light-field image, we can digitally generate a focal stack. Groundtruth depth is obtained from an RGB-D sensor which is calibrated to the light-field camera.

To the best of our knowledge, this is the largest dataset with groundtruth for the problem of DFF. We experimentally show that this amount of data is enough to successfully fine-tune a network. We compare our results with state-of-the-art DFF methods and provide a comprehensive study on the impact of different variations of the encoder-decoder type of network.

Our main contributions in this chapter are:

- *DDFFNet*, the first end-to-end learning method to compute depth maps from focal stacks.
- *DDFF 12-Scene* dataset, composed of 720 light-field images and co-registered groundtruth depth maps recorded with an RGB-D sensor.
- extensive evaluation of *DDFFNet* and its comparison to several state-of-the-art methods for DFF, as well as several variations of the encoder-decoder architecture.
- fast depth reconstruction: we can compute depth maps in 0.6 seconds on an NVidia Pascal Titan X GPU.

3.1.2 Related work

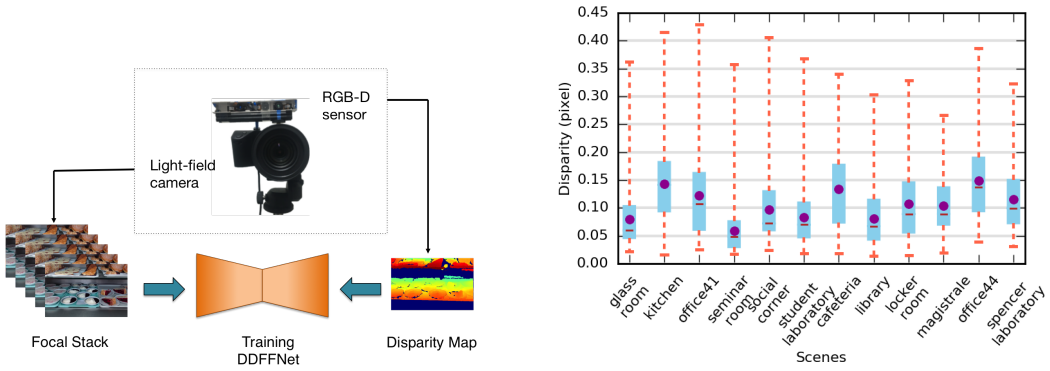
Depth from focus or Shape from focus. Conventional methods aim at determining the depth of a pixel by measuring its sharpness or focus at different images of the focal stack [59]. Developing a discriminative measure for sharpness is non trivial, we refer the reader to [59] for an overview. Other works aim at filtering the contrast coefficients before determining depth values by windowed averaging [60] or non-linear filtering [61]. Another popular approach to obtain consistent results is to use total variation regularization. [62] proposed the first variational approach to tackle DFF while [63] defines an objective function composed of a smooth but non-convex data term with a non-smooth but convex regularizer to obtain a robust (noise-free) depth map. Suwajanakorn *et al.* [64] computes DFF on mobile devices, focusing on compensating the motion between images of the focal stack. This results in a very involved model, that depends on optical flow results, and takes 20 minutes to obtain a depth map. Aforementioned methods heavily rely on priors/regularizers to increase the robustness of the algorithm, meaning their models may not generalize to all scenes. Interestingly, shape from focus was already tackled using neural networks in 1999 [65, 66], showing their potential on synthetic experiments. The increasing power of deep architectures makes it now possible to move towards estimating depth of real-world scenarios.

Plenoptic or light-field cameras. A light-field or plenoptic camera capture angular and spatial information on the distribution of light rays in space. In a single photographic exposure, these cameras are able to obtain multi-view images of a scene. The concept was first proposed in [67], and has recently gained interest from the computer vision community. These cameras have evolved from bulky devices [68] to hand-held cameras based on micro-lens arrays [69]. Several works focus on the calibration of these devices, either by using raw images and line features [3] or by decoding 2D lenslet images into 4D light-fields [70]. An analysis of the calibration pipeline is detailed in [71]. Light-field cameras are particularly interesting since depth and all-in-focus images can be computed directly from the 4D light-field [72–74]. Furthermore, focal stacks, *i.e.* images taken at different optical focuses, can be obtained from plenoptic cameras with a single photographic exposure. For this reason, we choose to capture our training dataset using

these cameras, though any normal camera that captures images at different optical focuses can be used at test time.

To the best of our knowledge, there are only two light-field datasets with groundtruth depth maps [1, 75]. While [75] provides 7 synthetic and only 6 real-scene light-fields, [1] generates a hand-crafted synthetic light-field benchmark composed of only 24 samples with groundtruth disparity maps. Our dataset is 25 times larger, composed of 12 indoor scenes, in total of 720 light-field samples with co-registered groundtruth depth obtained from an RGB-D sensor, ranging from 0.5 to $7m$. In this work, we show that our data is enough to fine-tune a network for the specific task of predicting depth from focus. Note that the Standford Light-field dataset (`http://lightfields.stanford.edu/`) has more samples than our dataset, but does not provide groundtruth depth maps, instead lambda-scaled depth maps generated by the standard Lytro toolbox. Therefore, the maps are not in real distance metrics and there is no camera calibration provided. Furthermore, Lytro depth maps can be inaccurate as we show in Section 3.4, and cannot be considered as groundtruth.

Deep learning. Deep learning has had a large impact in computer vision since showing its excellent performance in the task of image classification [76–78]. A big part of its success has been the creation of very large annotated datasets such as ImageNet [79]. Of course, this can also be seen as a disadvantage, since creating such datasets with millions of annotations for each task would be impractical. Numerous recent works have shown that networks pre-trained on large datasets for seemingly unrelated tasks like image classification, can easily be fine-tuned to a new task for which there exists only a fairly small training dataset. This paradigm has been successfully applied to object detection [80], pixel-wise semantic segmentation [18, 33, 49, 56, 57, 81], depth and normal estimation [55, 82, 83] or single image-based 3D localization [19, 20], to name a few. Another alternative is to generate synthetic data to train very large networks, *e.g.*, for optical flow estimation [15, 16]. Using synthetic data for training is not guaranteed to work, since the training data often does not capture the real challenge and noise distribution of real data. Several works use external sources of information to produce groundtruth. [84] uses sparse multi-view reconstruction results to train a CNN to predict surface normals, which are in turn used to improve the reconstruction. In [55, 85], the authors aim at predicting depth from a single image, but create groundtruth depth data from matching stereo images. We propose to use an RGB-D sensor that can be registered to our light-field camera to obtain the groundtruth depth map. Even though an RGB-D sensor is not noise-free, we show that the network can properly learn to predict depth from focus even from imperfect data. We use the paradigm of fine-tuning a pre-trained network and show that this works even if the tasks of image classification and DFF seem relatively unrelated.



(a) Experimental setup **(b) Whisker diagram**

Figure 3.2: (a) Experimental setup. We place an RGB-D sensor on top of a plenoptic camera in order to capture calibrated groundtruth depth maps and light-field images from which we then create focal stacks. These two inputs are used to train the *DDFFNet*. **(b) Whisker diagram** of the disparity distribution for each scene. Circle and red lines are the mean and median, respectively. Minimum disparity is 0.015 pixels (bottom orange lines), maximum disparity is 0.43 pixels (top orange lines).

3.2 DDFF 12-Scene benchmark

In this section we present our indoor DDFF 12-Scene dataset for *depth from focus*. This dataset is used for the training and evaluation of the proposed and several state-of-the-art methods. We first give the details on how we generate our data, namely the focal stack and groundtruth depth maps.

Why a 4D-lightfield dataset for depth from focus? To determine the depth of a scene from focus, we first need to generate a focal stack obtainable by using any camera and changing the focal step manually to retrieve the refocused images. Nonetheless, this is a time-consuming task that would not allow us to collect a significant amount of data as it is required to train deep models. Given the time that it takes to change the focus on a camera, the illumination of the scene could have easily changed or several objects could have moved. We therefore propose to leverage a lightfield camera as it has the following advantages: (i) only one image per scene needs to be taken, meaning all images will have the same photographic exposure and the capturing process will be efficient, (ii) refocusing can be performed digitally, which allows us to easily generate stacks with different focal steps, (iii) the dataset can be a benchmark not only for DFF, but also for other tasks such as depth from light-field or 3d reconstruction from lightfield. Even though we do not intend to tackle these tasks in this work, we do show some comparative results on depth from light-field in (DDLF) Section 3.4.

Light-field imaging. With light-field imaging technology, the original focus of the camera can be altered after the image is taken. Following this, we use a commercially

available light-field camera, *i.e.* Lytro ILLUM [86], to collect data and then generate focal stacks. Plenoptic cameras capture a 4D light-field $L(u, v, x, y)$ which stores the light rays that intersect the image plane Ω at (x, y) and the focus or camera plane Π at (u, v). The pixel intensity $I(x, y)$ is then:

$$I(x, y) = \int_u \int_v L(u, v, x, y) \; \partial u \, \partial v \, . \tag{3.1}$$

Refocusing on an image corresponds to shifting and summing all sub-apertures, $I_{(u,v)}(x, y)$. Given the amount of shift, pixel intensities of a refocused image are computed as follows [87]:

$$I'(x, y) = \int_u \int_v L(u, v, x + \Delta_x(u), y + \Delta_y(v)) \, \partial u \, \partial v \, . \tag{3.2}$$

The shift $(\Delta_u, \; \Delta_v)$ of each sub-aperture uv can be physically determined given an arbitrary depth Z in m, at which the camera is in-focus:

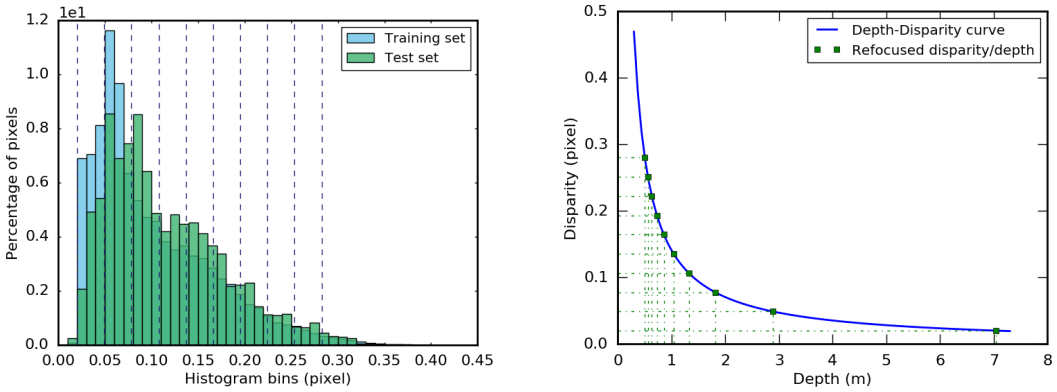
$$\begin{pmatrix} \Delta_x(u) \\ \Delta_y(v) \end{pmatrix} = \underbrace{\frac{\text{baseline} \cdot \mathbf{f}}{Z}}_{\text{disparity}} \cdot \begin{pmatrix} u_{center} - u \\ v_{center} - v \end{pmatrix} \, , \tag{3.3}$$

where the baseline is the distance between adjacent sub-apertures in meter/pixel, $\mathbf{f}$ is the focal length of the microlenses in pixels and $(u, \, v)^T$ indicates the spatial position of the sub-aperture in the Π plane in pixels. Although shifting can be performed using bilinear or bicubic interpolation, to be able to perform subpixel accurate focusing on the images, following [72] we use the *phase shift algorithm* to observe the impact of subpixel shifts on the images:

$$\mathcal{F}\{I'(x + \Delta_x(u))\} = \mathcal{F}\{I(x)\} \cdot \exp(2\pi i \Delta_x(u)) \, , \tag{3.4}$$

where $\mathcal{F}\{\cdot\}$ is the 2D discrete Fourier transform. We generate the focal stacks within a given disparity range, for which the focus shift on the images is clearly observable from close objects to far ones present in our dataset. Disparity values used in refocusing in Equation (3.3) are sampled linearly in the given interval for a stack size of S, meaning that the focus plane equally shifts in-between the refocused images. Example refocused images for disparity $\in \{0.28, 0.17, 0.02\}$ are shown in Figure 3.4. Note that we chose to use a light-field camera since it is easy to obtain a focal stack from it. Nonetheless, at test time, any imaging device could be used to take images at different optical focus.

Light-field camera calibration. For consistent capturing over all scenes, we fix the focal length of the main lens to $9.5mm$ and lock the zoom. To increase the re-focusable range of the camera, we use the *hyperfocal mode* (see [86] for details). Theoretically, we can then refocus from $27cm$ distance to infinity. We set the white-balancing, ISO and shutter speed settings to *auto* mode. In order to estimate the intrinsic parameters of the light-field camera, we use the calibration toolbox by Bok *et al.* [3] with a chessboard pattern composed of $26.25mm$ length squares. This toolbox generates sub-apertures based on a *radius* r_m of a microlens image, which is set to 7 pixels for the



(a) Disparity Distribution **(b) Depth to disparity**

Figure 3.3: (a) Disparity distribution of the training and test sets. Dashed blue lines represent the sampled disparity values used to generate the focal stacks.
(b) Depth to disparity conversion for DDFF 12-Scene dataset. Sampled disparities used for refocusing and their corresponding depths are denoted with green boxes.

Parameters of the Lytro ILLUM			
r_m	7	F_x	7299.7
K_1	-2.768	F_y	7317.0
K_2	1982.0	C_x	3991.6
k_1	0.388	C_y	2629.6
k_2	-0.0361	K_1/F'	27e−5

Table 3.1: Estimated intrinsic parameters of the Lytro ILLUM. F_* and C_* are respectively the focal length and optical center of the main lens in pixels. Baseline (K_1/F') is the distance between two adjacent sub-apertures in meter/pixel, where $F' = \max(F_x, F_y)$. Refer to [3] for details.

Lytro ILLUM camera. Although the toolbox generates 13×13 sub-apertures, we follow the authors' recommendation [3] to only use the ones within the displacement of $i^2+j^2 < (radius-1)^2$. This results in 9×9 undistorted sub-apertures, each of which has 383×552 image resolution. Estimated intrinsic parameters of the Lytro ILLUM and microlenses are given in Table 3.1 and in Table 3.2. Intrinsics of the microlenses are computed as

$$\text{Int} = \begin{bmatrix} F_x/(2r_m) & 0 & C_x/(2r_m) \\ 0 & F_x/(2r_m) & C_y/(2r_m) \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

where F_* and C_* are respectively the focal length and optical center of the main lens in pixels.

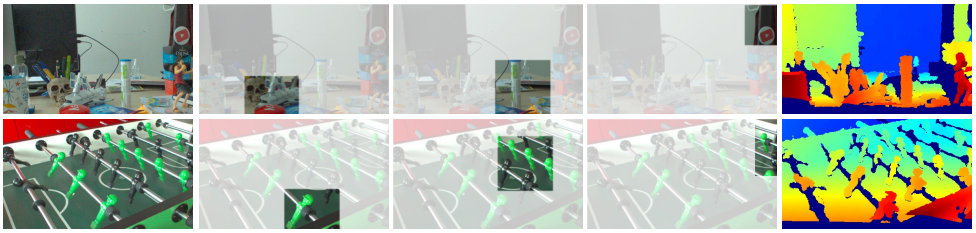


Figure 3.4: Example refocused images. *First column:* Center sub-aperture image. *Last column:* groundtruth disparity maps from the RGB-D sensor. *Middle columns:* Refocused images for varying disparity values in pixels, regions in focus are highlighted. Best viewed in color.

$\mathbf{f}$	$\mathbf{c}_x$	$\mathbf{c}_y$	baseline
521.4	285.11	187.83	27e−5

Table 3.2: Estimated intrinsic parameters of the microlenses. $\mathbf{f}_*$ and $\mathbf{c}_*$ are the focal length ($\mathbf{f} = \mathbf{f}_x = \mathbf{f}_y$) and optical center of the microlenses in pixels. Baseline is the distance between two adjacent sub-apertures in meter/pixel.

Groundtruth depth maps from an RGB-D sensor. Along with the light-field images, we also provide groundtruth depth maps. To this end, we use an RGB-D structure sensor, *i.e.* ASUS Xtion PRO LIVE, and mount it on the hot shoe of the light-field camera (see Figure 3.2a). Since we only need the infrared camera of the RGB-D sensor, we align the main lens of Lytro ILLUM to the infrared image sensor as close as possible for a larger overlap on the field of views of both cameras. We save the 480×640 resolution depth maps in millimeters. RGB-D sensors are not accurate on glossy surfaces and might even produce a large amount of invalid/missing measurements. In order to reduce the number of missing values, we take nine consecutive frames and save the median depth of each pixel during recording/capturing.

Stereo camera calibration. We perform mono and stereo camera calibration to estimate the relative pose of the depth sensor with respect to the light-field camera. To this end, we use *Camera Calibration Toolbox for Matlab* the publicly available on `www.vision.caltech.edu/bouguetj/calib_doc/`. We use the same calibration pattern as for the light-field calibration. Stereo calibration is performed between the center sub-aperture $(u, v)^T = (5, 5)^T$ and the infrared camera image. While we fix the intrinsics of the light-field camera as given in Table 3.1, depth sensor is calibrated only for intrinsic parameters (no distortion). After the calibration procedure, we register the depth maps onto the center sub-aperture images. As one can observe in the examples in Figure 3.4, due to the RGB-D sensor noise and the calibration procedure, some pixels around object boundaries do not contain depth measurements (represented in dark blue). Recorded depth maps can be improved further for a better domain adaptation [88–91]. We leave

the possible improvements for future work. We convert depth to disparity to generalize the method to different cameras as follows:

$$\text{disparity} = \frac{\text{baseline} \cdot \mathbf{f}}{Z} \,, \tag{3.6}$$

where $Z > 0$.

DDFF 12-Scene benchmark. We collect the dataset in twelve different indoor environments: *glassroom, kitchen, office41, seminar room, social corner, student laboratory, cafeteria, library, locker room, magistrale, office44* and *spencer laboratory*. First six scenes are composed of 100 light-field images and depth pairs and the latter six scenes are composed of 20 pairs. Our scenes have at most 0.5 pixel disparity while the amount of measured disparity gradually decreases towards far distances. Figure 3.2b plots the Whisker diagrams for each scene. In Figure 3.3a, we plot the normalized disparity histogram of the training and test sets. We generate the focal stacks for 10 sampled disparities in the interval of $[0.28, 0.02]$ pixels (equivalent to $[0.5, 7]$ meters), indicated with blue dashed lines in Figure 3.3a. We also plot the depth to disparity conversion for the given baseline and focal length of the microlenses in Figure 3.3b. Refocused disparity values and their corresponding depths are denoted with a green box. Note that disparity is inversely proportional to depth and therefore, linear sampling in disparity corresponds to non-linear sampling in depth. We choose to sample disparities to have a linear focus change between stack images. Example center sub-aperture images for *office41* and *locker room* scenes and their corresponding disparity maps are shown in Figure 3.4. In Figure 3.5, disparity distributions of the twelve scenes are shown.

Since the dataset consists mainly of indoor scenes, flat surfaces (wall, desk), textureless objects (monitor, door, cabinet) and glossy materials (screen, windows) are often present. Our dataset is therefore more challenging and 25 times larger than previous synthetic datasets [1, 75]. DDFF 12-Scene dataset consisting of the light-field images, generated focal stacks and registered depth maps are publicly available on https://vision.cs.tum.edu/data/datasets/ddff12scene.

3.3 Depth Reconstruction from Focal Stack with Deep Convolutional Networks

In this section, we detail our proposed approach to estimate a depth map given a focal stack. In order to achieve this, we formalize our problem as a minimization of a regression function, which is an end-to-end trained convolutional neural network.

Let $\mathcal{S}$ be a focal stack consisting of S refocused images $I \in \mathbb{R}^{H \times W \times C}$ and the corresponding target disparity map $D \in \mathbb{R}^{H \times W}$, we minimize the least square error between the estimated disparity $f(\mathcal{S})$ and the target D:

$$\mathcal{L} = \sum_{p}^{HW} \mathcal{M}(p) \cdot \left\| f(\mathcal{S}(p), \mathbf{W}) - D(p) \right\|_2^2 + \lambda \left\| \mathbf{W} \right\|_2^2 . \tag{3.7}$$

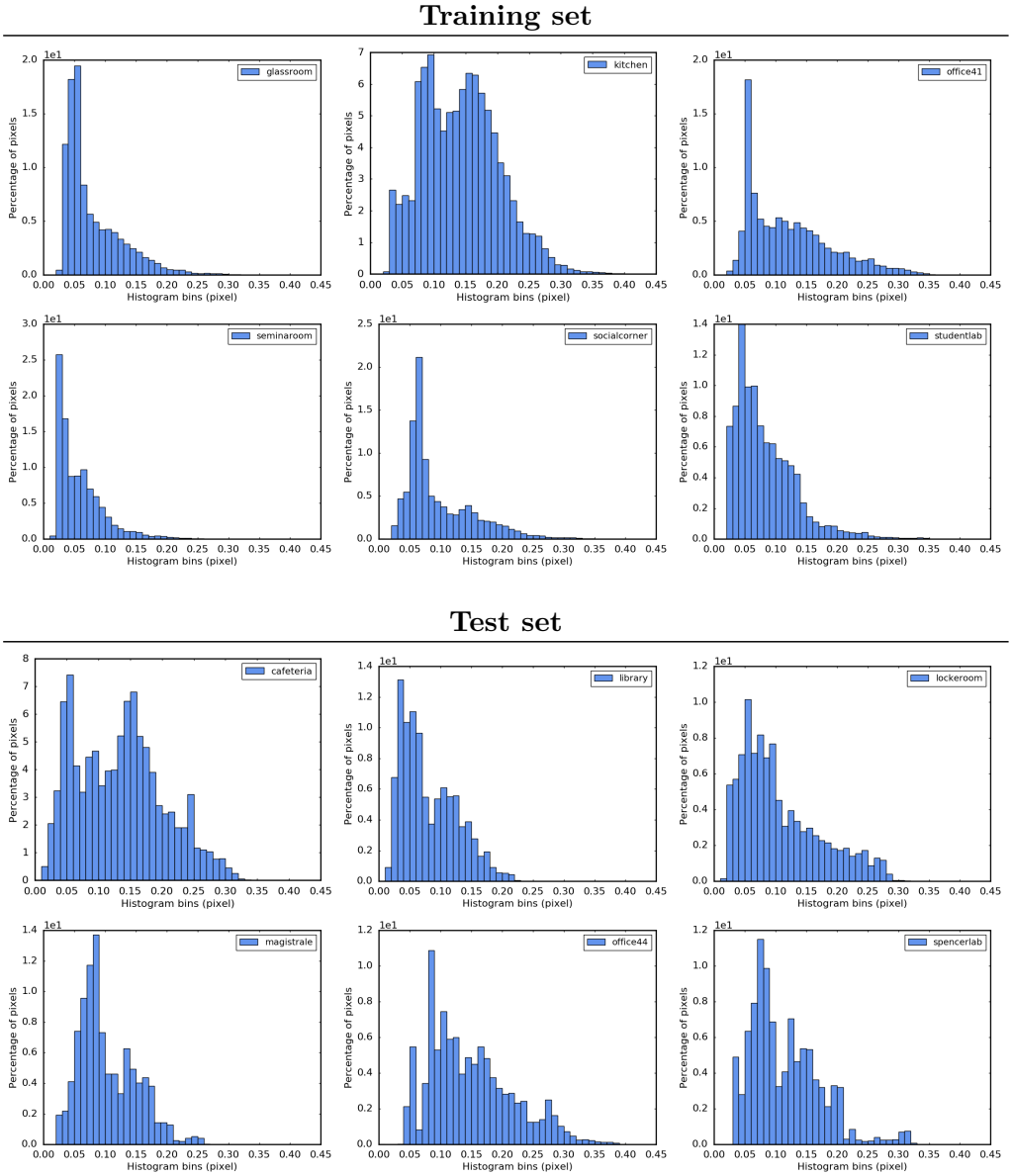


Figure 3.5: Disparity distribution of the DDFF 12-Scene dataset. Each of the first six scenes is composed of 100 light-field samples and used for training. Each of the latter six scenes is composed of 20 light-field samples and used for testing.

Loss function $\mathcal{L}$ is summed over all valid pixels p where $D(p) > 0$, indicated by the mask $\mathcal{M}$ and $N : \mathbb{R}^{S\times H\times W\times C} \rightarrow \mathbb{R}^{H\times W}$ is a convolutional neural network. Weights ($\mathbf{W}$) are penalized with ℓ_2-norm (Equation (2.24)). Depth/disparity maps captured by RGB-D sensors often have missing values, indicated with a value of 0. Therefore, we ignore the missing values during training in order to prevent networks from outputting artifacts.

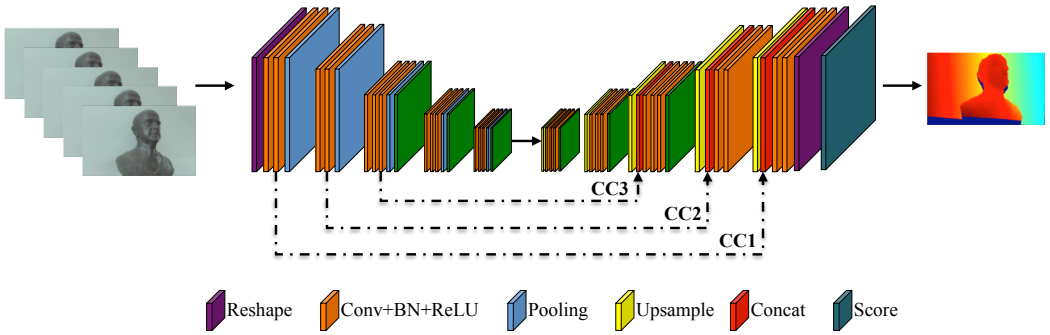


Figure 3.6: DDFFNet. Proposed auto-encoder-style architecture that takes in a focal stack and produces a disparity map. We present several architectural modifications, namely CC connections, Upsample, *i.e.* Unpool, BL and UpConv (check the text for details).

Network architecture. We propose an end-to-end trainable auto-encoder style convolutional neural network. CNNs designed for image classification are mostly encoder type networks which reduce the dimension of the input to a 1D vector [76–78]. This type of networks are very powerful at constructing descriptive hierarchical features later used for image classification. This is why for tasks which require a pixel-wise output, the encoder part is usually taken from these pre-trained networks [76–78] and a mirrored decoder part is created to upsample the output to image size. We follow this same paradigm of hierarchical feature learning for pixel-wise regression tasks [15, 16, 33, 49, 56] and design a convolutional auto-encoder network to generate a dense disparity map as shown in Figure 3.6.

As a baseline for the encoder network, we use the VGG-16 net [77]. It consists of 13 convolutional layers, 5 poolings and 3 fully-connected layers. In order to reconstruct the input size, we remove the fully-connected layers and reconstruct the decoder part of the network by mirroring the encoder layers. We invert the 2×2 pooling operation with 4×4 upconvolution (deconvolution) [56] with a stride of 2 and initialize the weights of the upconvolution layers with bilinear interpolation, depicted as upsample in Figure 3.6.

Similar to the encoder part, we use convolutions after upconvolution layers to further sharpen the activations. To accelerate the convergence, we add batch normalization [28] after each convolution and learn the scale and shift parameters during training. Batch normalization layers are followed by rectified linear unit (ReLU) activations. Moreover, after the 3rd, 4th and 5th poolings and before the corresponding upconvolutions, we apply dropout with a probability of 0.5 during training similar to [33]. In order to preserve the sharp object boundaries, we concatenate the feature maps of early convolutions `conv1_2, conv2_2, conv3_3` with the decoder feature maps: outputs of the convolutions are concatenated with the outputs of corresponding upconvolutions. Figure 3.6 demonstrates a sketch of our network.

We refer to this architecture as *DDFFNet*. There are several architectural choices that one can make that can significantly increase or decrease the performance of auto-

encoder networks. Some of these changes are the way upsampling is done in the decoder part or the skip connections. For the problem of DFF, we study the performance of the followings variants:

- *DDFFNet-Upconv*: In the decoder part, we keep the upconvolutions.
- *DDFFNet-Unpool:* Upconvolutions are replaced by 2×2 unpooling operation [48].
- *DDFFNet-BL:* Upconvolutions are replaced by 2×2 bilinear interpolation (upsampling).
- *DDFFNet-CCx:* Here we study the effect of several concatenation connections, designed to obtain sharper edges in the depth maps.

Network input. VGG-16 net takes the input size of $H \times W \times C$, precisely $224 \times 224 \times 3$. In contrast, we need to input the whole focal stack $\mathcal{S}$ into the network. Computing features per stack image I is a general way of incorporating sharpness into DFF approaches [63] and we make use of this intuition within our end-to-end trained CNN. Since the depth of a pixel is correlated with the sharpness level of that pixel and the convolutions are applied through input channels C, we consider the network as a feature extractor and therefore, we reshape our input to $(B \cdot S \times C \times H \times W)$ with a batch size of B. Hence, the network generates one feature map per image in the stack with a size of $(B \cdot S \times 1 \times H \times W)$. In order to train the network end-to-end, we reshape the output feature maps to $(B \times S \times H \times W)$ and apply 1×1 convolution as a regression layer through the stack, depicted as *Score* layer in Figure 3.6.

3.4 Experimental Evaluation

We evaluate our method on the DDFF 12-Scene dataset proposed in Section 3.2. We also present results on the 4D light-field dataset [1] in Section 3.7. We first split the twelve scenes into training and test sets. We use the six scenes, *i.e. cafeteria, library, locker room, magistrale, office44, spencer laboratory* for testing as these scenes have in total 120 focal stacks and are also a good representation of the whole benchmark, as shown in Figure 3.2a. The other six scenes are then used for training with a total of 600 focal stacks.

Evaluation metrics. Following [1, 55, 83, 92] we evaluate the resulting depth maps with eight different error metrics:

- MSE : $\frac{1}{|\mathcal{M}|} \sum_{p \in \mathcal{M}} \left\| f(\mathcal{S}(p)) - D(p) \right\|_2^2$
- RMS : $\sqrt{\frac{1}{|\mathcal{M}|} \sum_{p \in \mathcal{M}} \left\| f(\mathcal{S}(p)) - D(p) \right\|_2^2}$
- log RMS : $\sqrt{\frac{1}{|\mathcal{M}|} \sum_{p \in \mathcal{M}} \left\| \log f(\mathcal{S}(p)) - \log D(p) \right\|_2^2}$

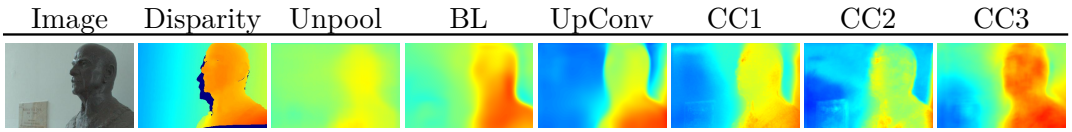


Figure 3.7: Qualitative results of the variants of DDFFNet. While BL oversmooths the edges, CC1 and CC2 introduce artificial edges on the disparity map. Unpooling is not suitable well to recover fine edges. Best viewed in color.

- Absolute relative : $\frac{1}{|\mathcal{M}|}\sum_{p\in\mathcal{M}}\frac{|f(\mathcal{S}(p))-D(p)|}{D(p)}$
- Squared relative : $\frac{1}{|\mathcal{M}|}\sum_{p\in\mathcal{M}}\frac{\left\|f(\mathcal{S}(p))-D(p)\right\|_2^2}{D(p)}$
- Accuracy: % of $D(p)$ s.t $\max\left(\frac{f(\mathcal{S}(p))}{D(p)}, \frac{D(p)}{f(\mathcal{S}(p))}\right) = \delta < thr$
- BadPix(τ): $\frac{|\{p\in\mathcal{M}:|f(\mathcal{S}(p))-D(p)|>\tau\}|}{|\mathcal{M}|}\cdot 100$
- Bumpiness: $\frac{1}{|\mathcal{M}|}\sum_{p\in\mathcal{M}}\min(0.05, \|\mathrm{H}_\Delta(p)\|_F)\cdot 100$

where $\Delta = f(\mathcal{S}(p)) - D(p)$ and H is the Hessian matrix. The first five measures are standard error measures, therefore lower is better while for the Accuracy measure higher is better. BadPix(τ) quantifies the number of wrong pixels with a given threshold τ while Bumpiness metric focuses on the smoothness of the predicted depth maps [1].

Experimental setup. For our experiments, we generate the focal stacks for $S = 10$ with disparities linearly sampled in $[0.28, 0.02]$ pixel (equivalent to $[0.5, 7]$ meters). We found this to be a good compromise between obtaining pixel sharpness at all depths and memory consumption and runtime, which heavily increases for larger focal stacks without bringing improved depth accuracy (see Figure 3.9).

DDFF 12-Scene consists of 383×552 images, thus training on full resolution stacks is inefficient. One solution would be to downsample the images, however, interpolation could change the blur kernels, eventually affecting network performance. The solution we adapt is to train the network on $10 \times 224 \times 224 \times 3$ stack patches. To do so, we crop the training stacks and corresponding disparity maps with a patch size of 224 and a stride of 56, ensuring that cropped patches cover the whole image. Patches with more than 20% missing disparity values are removed from the training set. 20% of the training data is used as validation for model selection. At test time, results are computed on the full resolution 383×552 images.

We run all experiments on an NVidia Pascal Titan X GPU. Encoder part of the network is initialized from the pre-trained VGG-16 net, decoder part is initialized with variance scaling [25]. We use the SGD optimizer with momentum decay of 0.9 (see Section 2.3.4). Batch size B is set to 2 and learning rate to 0.001. Every fourth epoch we exponentially reduce the learning rate by a factor of 0.9. Training set is shuffled in the beginning of each epoch and we set the weight decay λ for convolution layers to $5\,e^{-4}$.

	Method	MSE ↓	RMS ↓	log RMS ↓	Abs. rel. ↓	Sqr. rel.↓	Accuracy ($\delta = 1.25$) δ ↑	δ^2 ↑	δ^3 ↑	Bump. ↓
DDFFNet	Unpool	$2.9\,e^{-3}$	0.050	0.50	0.64	0.05	39.95	63.46	78.26	0.62
	BL	$2.1\,e^{-3}$	0.041	0.43	0.46	0.03	51.29	74.81	85.28	0.54
	UpConv	$1.4\,e^{-3}$	0.034	0.33	0.30	0.02	52.41	83.09	93.78	0.54
	CC1	$1.4\,e^{-3}$	0.033	0.33	0.37	0.02	60.38	82.11	90.63	0.75
	CC2	$1.8\,e^{-3}$	0.039	0.39	0.39	0.02	44.80	76.27	89.15	0.75
	CC3	$9.7\,e^{-4}$	0.029	0.32	0.29	0.01	61.95	85.14	92.99	0.59
	PSPNet	$9.4\,e^{-4}$	0.030	0.29	0.27	0.01	62.66	85.90	94.42	0.55
	Lytro	$2.1\,e^{-3}$	0.040	0.31	0.26	0.01	55.65	82.00	93.09	1.02
	PSP-LF	$2.7\,e^{-3}$	0.046	0.45	0.46	0.03	39.70	65.56	82.46	0.54
	DFLF	$4.8\,e^{-3}$	0.063	0.59	0.72	0.07	28.64	53.55	71.61	0.65
	VDFF	$7.3\,e^{-3}$	0.080	1.39	0.62	0.05	8.42	19.95	32.68	0.79

Table 3.3: Quantitative results of the proposed method. ***DDFFNet-CC3*** is the best depth from focus method and provides also better results compared to Lytro, *i.e.* depth from light-field. Metrics are computed on the predicted and the groundtruth disparity maps.

3.4.1 Ablation studies

We first evaluate our architecture variations such as the three upsampling layers: unpooling, upconvolution and bilinear interpolation. We can see from Figure 3.7, *Unpool* does not preserve the fine object edges while *BL* oversmooths them due to naive linear interpolation. These observations are also supported by the quantitative experiments in Table 3.3. Hence, we choose to use *UpConv* for the rest of the experiments.

Within the tested concatenation schemes, *DDFFNet-CC1* and *DDFFNet-CC2* preserve too many edges as they benefit from larger feature maps. However, this produces incorrect depth and therefore achieving overall worse MSE compared to that of *DDFFNet-CC3*, see Table 3.3. On the other hand, *DDFFNet-CC3* preserves only the most important edges corresponding to object boundaries. Going deeper in the concat connections would not provide sufficiently fine structures, hence, we do not test connections after *CC3*.

We further plot the BadPix measure when changing the threshold τ in Figure 3.9. In this plot, we compare our best architecture *DDFFNet-CC3* with focal stacks of varying sizes, $S \in \{5, 8, 10, 15\}$. Having light-field images allows us to digitally generate focal stacks of varying sizes, enabling us to find an optimal size. Even though increasing the stack size S should theoretically decrease the depth error, S-15 quickly overfits due to the fact that it was trained with a batch size of 1 to fit into the memory of a single GPU. We find that a focal stack of 10 images is the best memory-performance compromise, which is why all further experiments are done with $S = 10$.

3.5 Comparison to state-of-the-art

We have implemented several baselines to compare our method with:

Variational DFF. We compare our results with the state-of-the-art variational method, VDFF [63], using their GPU code (`https://github.com/adrelino/variational-depth-from-focus`) and the same focal stacks as in our method. We run a grid search on several VDFF parameters and the results reported are for the best set of parameters. VDFF outputs a real valued index map. Each pixel is assigned to one of the stack images, where the pixel is in focus. Therefore, we directly interpolate these indices to their corresponding disparity values and compute our metrics on the mapped disparity output.

PSPNet for DFF. *Pyramid Scene Parsing Network* [81] is based on a deeper encoder (ResNet) and also capable of capturing global context information by aggregating different-region-based context through the pyramid pooling module. It is originally designed for semantic segmentation, however, we modified the network for depth from focus problem (input and output) and trained it end-to-end on our dataset. We also compare to PSPNet in order to observe the effects of significant architectural changes in terms of a deeper encoder (ResNet) with a recent decoder module for the problem of depth from focus.

Lytro depth. For completion, we also compare with the depth computed from the light-field directly by the Lytro [86]. Although this method technically does not compute DFF, we still think it is a valuable baseline to show the accuracy that depth from light-field methods can achieve. Lytro toolbox predicts depth in lambda unit, thus the output is not directly comparable to our results. For this reason, we formulate the rescaling from Lytro depth to our groundtruth as an optimization problem that finds a unique scaling factor k^*. To do so, we minimize the least squares error between the resulting depth $\tilde{Z}(p)$ and the groundtruth depth $Z(p)$ to find the best scaling factor k^*:

$$k^* = \arg\min_k \sum_p \|k \cdot \tilde{Z}(p) - Z(p)\|_2^2 \,, \tag{3.8}$$

where $k \in \mathbb{R}$. Note that this is the best possible mapping in terms of MSE to our groundtruth depth maps provided that the focal stack has uniform focal change, therefore, we are not penalizing [86] during the conversion process. Evaluation metrics are then computed on $k^* \cdot \tilde{D}(p)$ and $D(p)$.

Depth from light-field. Even though we focus on the task of DFF, we want to provide a comparison to depth from light-field. For this purpose, we follow [93, 94] and train our network (*DDFFNet-CC3*) as well as PSPNet with 11 sub-apertures from the light-field camera as input. Sampling pattern is shown in Figure 3.8. We denote these models as DFLF and PSP-LF, respectively. To the best of our knowledge, there is no code for [93, 94] to test their full pipeline on our dataset.

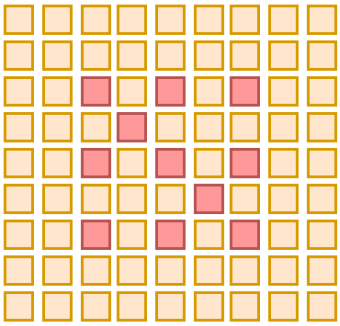


Figure 3.8: Sampled sub-apertures used as an input for the *depth from light field* networks. We use 11 sub-apertures (denoted by red) sampled from 9×9 light-fields.

	Method	Runtime (s.)	Depth (m.)
DDFFNet	Unpool	0.55	1.40
	BL	0.43	1.10
	UpConv	0.50	0.58
	CC1	0.60	0.79
	CC2	0.60	0.86
	CC3	0.58	0.86
	DFLF	0.59	1.50
	VDFF	2.83	8.90
	Lytro	25.26 (CPU)	0.99

Table 3.4: Runtime and Depth error. DDFFNet is faster and more accurate than other state-of-the-art methods. For completeness, we also report the runtime of Lytro toolbox on CPU. VDFF performs worse as it requires many iterations of optimization during test.

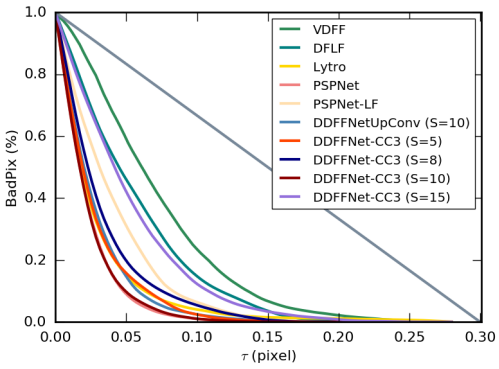


Figure 3.9: Badpix(%) for DDFFNet-CC3 for $S \in \{5, 8, 10, 15\}$, DDFFNet-UpConv for S=10, for Lytro, DFLF and VDFF. While τ increases BadPix error decreases. DDFFNet with stack size of 10 is better than VDFF and Lytro by a large margin.

As we can see from Table 3.3, *DDFFNet-CC3* outperforms the other depth from focus method, *i.e.* VDFF [63], in all evaluation metrics, reducing depth error by more than 75%. The major reason for this is that VDFF proposes an optimization scheme that relies on precomputed hand-crafted features, which can handle the synthetic or clean high resolution images but fail in the realistic, challenging scenes of our dataset.

PSPNet performs on-par to *DDFFNet-CC3*, nevertheless, as shown in Figure 3.1, pyramid pooling leads to oversmooth depth maps due to its upsampling strategy. Although this network is very efficient for semantic segmentation, we found that our decoder choice with skip connection CC3 yields more accurate, non-smooth depth maps.

Lytro, on the other hand, computes very inaccurate depth maps on flat surfaces (Figure 3.1) while our *DDFFNet* estimates a smoother and also more accurate disparity maps. Note that Lytro depth is what is provided as groundtruth depth maps in the Stanford Light-field dataset, but this depth computation is relying on an algorithm that is not always accurate as we have shown. This is another advantage of the proposed dataset which provides groundtruth maps from an external RGB-D sensor.

As we can see in Table 3.3, our approach *DDFFNet-CC3* still performs better than depth from light-field (DFLF) with a similar number of images. Note that it would not be possible to fit all sub-aperture images provided by the light-field camera into GPU memory. In contrast to DFLF, our method is usable with any camera.

Moreover, we present the mean squared errors of computed depth maps and runtime in Table 3.4. *DDFFNet-CC3* achieves a much lower error when compared to VDFF or DFLF. *DDFFNet-UpConv* has a better depth error than *DDFFNet-CC3*, but, its badpix error is significantly larger than *DDFFNet-CC3*, demonstrated in Figure 3.9.

We demonstrate further qualitative results in Figure 3.10. Note the poor performance of classic methods like VDFF, and even the wrong disparity maps produced by Lytro in the first row. In Figure 3.11 we present two failure cases where the network output is not sharp or not consistent.

Overall, experiments show that our method is more accurate by a large margin when compared to the classical variatonal DFF method [63] while also being orders of magnitude faster on a GPU. It is also more accurate than the Lytro predicted depth or a network trained for depth from light-field. Several network architectures were explored, and finally *CC3* was deemed the best with overall lowest disparity error while keeping object boundaries in the disparity map.

3.6 What is the network learning?

Following the convention in depth from focus, one can see DDFFNet as a sharpness measure. Our network takes an input of $B \times S$ and reshapes it before the first convolution. This design allows the network to learn in which image of the stack the pixel is sharpest. From this sharpness level, our regression layer, denoted as *Score*, regresses the depth from sharpness (focus). Without the *Score* layer, DDFFNet can be considered as a function which measures the sharpness of each pixel through the stack. In order to visually prove the concept, we demonstrate the activation maps in Figure 3.12, where one can clearly see how the activations move towards the closest object (the chair) as we advance in the focal stack, hinting at the fact that network is indeed exploiting the focus cue to estimate the depth.

3.7 Results on the 4D light-field dataset

To show the generality of our method, we also present the results of *DDFFNet-CC3* on the 4D light-field dataset [1]. Note that this dataset is not designed for depth from focus but rather depth from light-field and therefore, we generate a focal stack of refocused

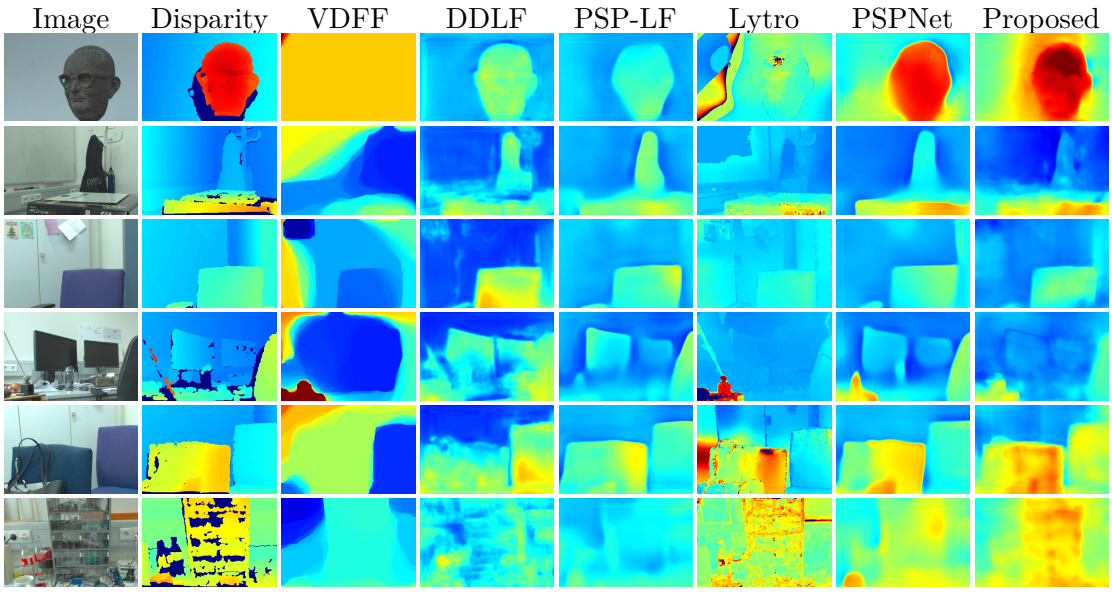


Figure 3.10: Qualitative results of the DDFFNet versus state-of-the-art methods. Results are normalized by the maximum disparity. Warmer colors represent closer distances. Best viewed in color.

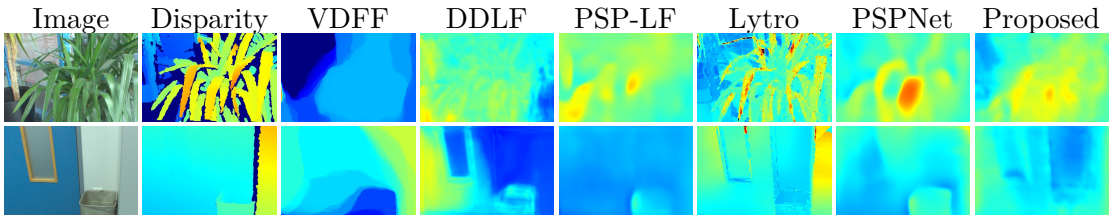


Figure 3.11: Failure cases. Results are normalized by the maximum disparity. Warmer colors represent closer distances. Best viewed in color.

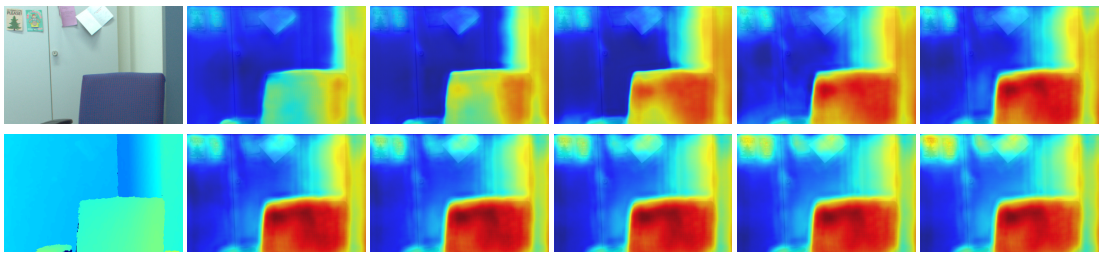


Figure 3.12: Activation heat maps for the refocused images in a focal stack. First column: *top:* center subaperture image, *bottom:* groundtruth disparity map. The rest of the columns show from left to right, top to bottom, how the activations on the focal stack images evolve. Note how the activations slowly shift to the closest object (the chair) as we advance in the focal stack.

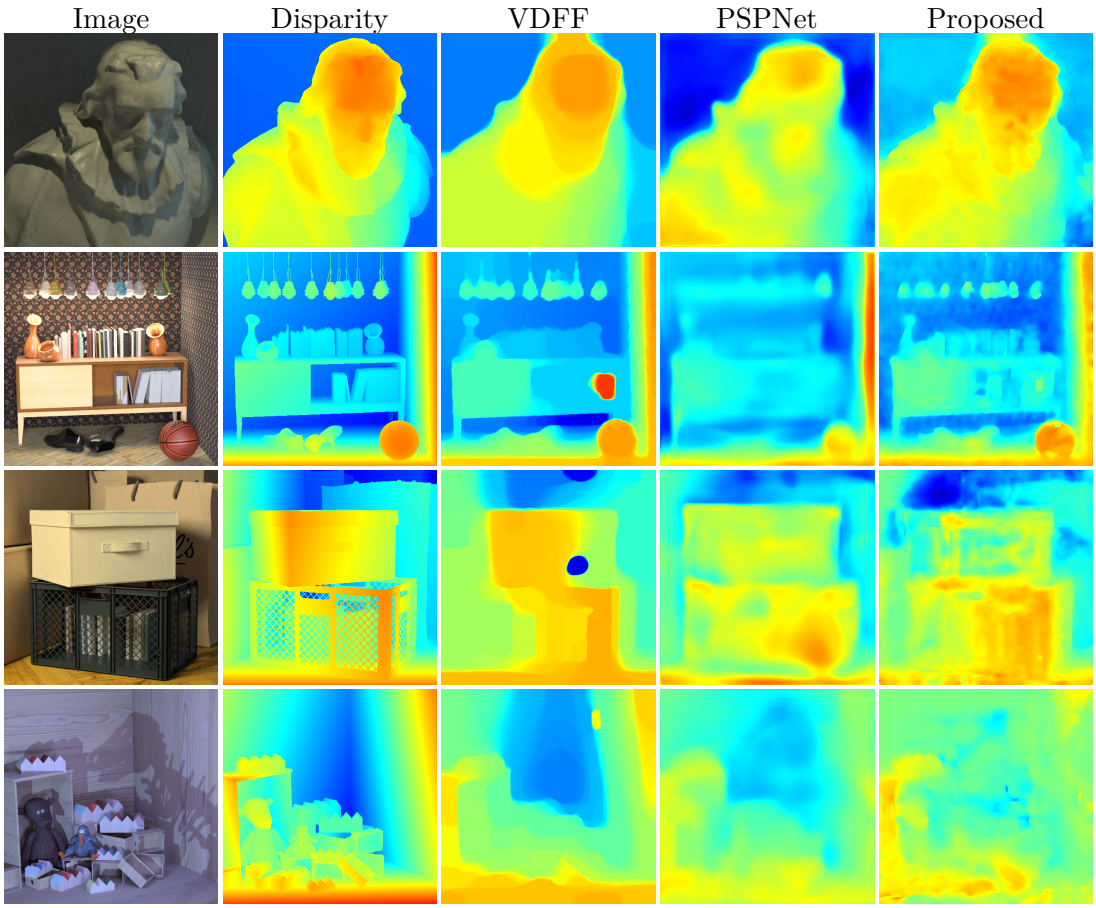


Figure 3.13: Results on 4D light-field dataset [1]. DDFFNet can keep the finer structures in the disparity maps. On the other hand, VDFF and PSPNet produces incorrect and over-smoothed results.

images per scene in the disparity interval of $[-2.5, 2.5]$. from the light-field images. The number of images in this benchmark is limited and training a network from scratch with the provided data does not work. We propose to do transfer learning from our DDFF 12-Scene dataset by fine-tuning *DDFFNet-CC3* and PSPNet. As training set for fine-tuning we use the 16 light-field images marked as "additional" and 4 marked as "stratified". As test set we use the 4 scenes marked as "train" set.

In Figure 3.13 we show the qualitative results and in Table 3.5 we present the quantitative results. *DDFFNet-CC3* outperforms other methods in terms of MSE and RMS error, showing that by using only 20 images from a completely different setting, we can fine-tune our network to achieve accurate results.

Method	MSE ↓	RMS ↓	Bump. ↓
VDFF	1.30	1.15	1.58
PSPNet	0.37	0.53	1.21
DDFFNet-CC3	0.19	0.42	1.92

Table 3.5: Quantitative results of the proposed method. *DDFFNet-CC3* outperforms VDFF also on the 4D light-field benchmark. Metrics are computed on the predicted and groundtruth disparity maps.

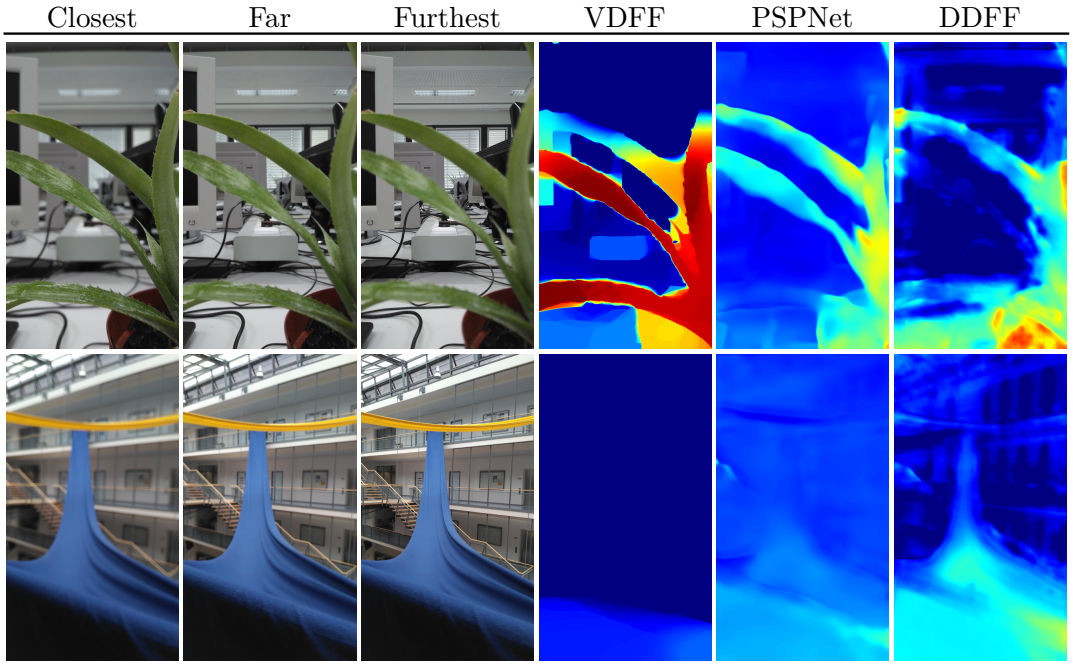


Figure 3.14: Qualitative results of *DDFFNet-CC3*, VDFF and PSPNet on the focal stacks captured with an Android smartphone.

3.8 Is DDFFNet generalizable to other cameras?

In order to present the generality of our method, we captured two focal stacks with an Android smartphone where we change the focus, gradually and linearly from 0.1 to 4 meters and compute the disparities with *DDFFNet-CC3*, VDFF and PSPNet. Qualitative results are illustrated in Figure 3.14. VDFF can perform well in the first image, however, cannot compute a reasonable depth on the second image. PSPNet suffers estimating a finer depth for far distances. On the other hand, our method preserves finer edges on both images for close and far objects.

3.9 Conclusions

Depth from focus (DFF) is a highly ill-posed inverse problem because the optimal focal distance is inferred from sharpness measures which fail in untextured areas. Existing variational solutions revert to spatial regularization to fill in the missing depth, which are not generalized to more complex geometric environments. In this chapter, we introduced a novel approach to recover depth from a focal stack, where the focus is gradually changing from close to far objects. To this end, we first generated a large dataset composed of focal stacks and their co-registered depth pairs and trained a deep network to solve the DFF problem. Moreover, we devised a suitable network architecture and demonstrated that our network, *i.e.* *DDFFNet* outperforms state-of-the-art methods, reducing the depth error by more than 75% and predicting a disparity map in only 0.6 seconds. Furthermore, we also show that our method can be easily generalized for other datasets and cameras with minimal effort.

Chapter 4

Incorporating Depth into Semantic Segmentation

In Chapter 3, we discussed *depth from focus*, reconstructing depth from focus cue. Depth is an essential complementary information to color and texture and therefore is necessary in many computer vision tasks, *e.g.*, image-based localization, scene reconstruction, 3D object/scene modeling, object/scene recognition/classification and more interestingly semantic scene understanding, to name few.

This chapter addresses the problem of semantic labeling of indoor scenes on RGB-D data. With the availability of RGB-D cameras, it is expected that additional depth measurement will improve the accuracy. Here we investigate a solution how to incorporate complementary depth information into a semantic segmentation framework by making use of convolutional neural networks (CNNs). We propose an auto-encoder-style network, where the encoder part is composed of two branches of networks that simultaneously extract features from RGB and depth images and fuse depth features into the RGB branch as the network goes deeper. We provide a comprehensive experimental evaluation to demonstrate that the proposed fusion-based architecture achieves competitive results with the state-of-the-art methods on the challenging SUN RGB-D [4] and NYUv2 [6] benchmarks. This work has been published in [18].

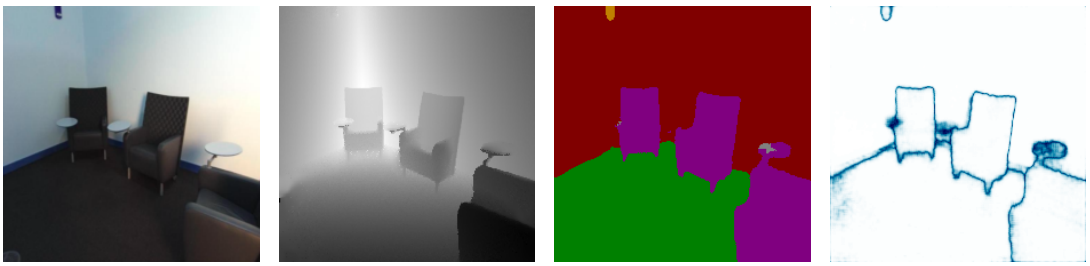


Figure 4.1: An exemplar input and output of FuseNet. From left to right: input RGB and depth images, the predicted semantic labeling and the probability of the corresponding labels, where white and blue denote high and low probabilities, respectively.

4.1 Introduction

Visual scene understanding in a glance is one of the most amazing capability of the human brain. In order to model this ability, semantic segmentation aims at giving a class label for each pixel on the image according to its semantic meaning. This problem is one of the most challenging tasks in computer vision, and has received a lot of attention from the computer vision community [5, 49, 56, 95–98].

Convolutional neural networks (CNNs) have recently attained a breakthrough in various classification tasks such as semantic segmentation. CNNs have been shown to be powerful visual models that yields hierarchies of features. The key success of this model mainly lies in its general modeling ability for complex visual scenes. Currently CNN-based approaches [56, 97, 99] provide the state-of-the-art performance in several semantic segmentation benchmarks. In contrast to CNN models, by applying hand-crafted features one can generally achieve rather limited accuracy.

Utilizing depth additional to the appearance information (*i.e.* RGB) could potentially improve the performance of semantic segmentation, since the depth channel has complementary information to RGB channels, and encodes structural information of the scene. The depth channel can be easily captured with low cost RGB-D sensors. In general object classes can be recognized based on their color and texture attributes. However, the auxiliary depth may reduce the uncertainty of the segmentation of objects having similar appearance information. Couprie *et al.* [100] observed that the segmentation of classes having similar depth, appearance and location is improved by making use of the depth information too, but it is better to use only RGB information to recognize object classes containing high variability of their depth values. Therefore, the optimal way to fuse RGB and depth information has been left an open question.

In this chapter we address the problem of indoor scene understanding assuming that both RGB and depth information simultaneously available (see Figure 4.1). This problem is rather crucial in many perceptual applications including robotics. We remark that although indoor scenes have rich semantic information, they are generally more challenging than outdoor scenes due to more severe occlusions of objects and cluttered background. For example, indoor object classes, such as *chair*, *dining table* and *curtain* are much harder to recognize than outdoor classes, such as *car*, *road*, *building* and *sky*.

Contributions can be summarized as follows:

- We investigate a solution how to incorporate complementary depth information into a semantic segmentation framework. To this end we propose an auto-encoder-style network, referred to as FuseNet, where the encoder part is composed of two branches of networks that simultaneously extract features from RGB and depth images and fuse depth features into the RGB branches as the network goes deeper (see Figure 4.2).

- We propose and examine two different ways of fusing RGB and depth features. We also analyze the proposed network architectures, referred to as dense and sparse fusion (see Figure 4.3), in terms of the level of fusion.

- We experimentally show that our proposed method is successfully able to fuse RGB and depth information for semantic segmentation also on cluttered indoor scenes. Moreover, our method achieves competitive results with state-of-the-art methods in terms of segmentation accuracy evaluated on the challenging SUN RGB-D [4] and NYUv2 [6] datasets.

4.2 Related Work

A fully convolutional network (FCN) architecture has been introduced in [56] that combines semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed segmentations by applying end-to-end training. Noh *et al.* [49] have proposed a novel network architecture for semantic segmentation, referred to as DeconvNet, which alleviates the limitations of fully convolutional models (*e.g.*, very limited resolution of labeling). DeconvNet is composed of deconvolution and unpooling layers on top of the VGG 16-net [77]. To retrieve semantic labeling on the full image size, Zeiler *et al.* [101] have introduced a network composed of deconvolution and unpooling layers. Concurrently, a very similar network architecture has been presented [17] based on the VGG 16-net [77], referred to as *SegNet*. In contrast to DeconvNet, SegNet consists of smoothed unpooled feature maps with convolution instead of deconvolution. Kendall *et al.* [33] further improved the segmentation accuracy of SegNet by applying dropout [30] during test time [102].

Some recent semantic segmentation algorithms combine the strengths of CNN and conditional random field (CRF) models. It has been shown that the poor pixel classification accuracy, due to the invariance properties that make CNNs good for high level tasks, can be overcome by combining the responses of the CNN at the final layer with a fully connected CRF model [99]. CNN and CRF models have also been combined in [97]. More precisely, the method proposed in [97] applies mean field approximation as the inference for a CRF model with Gaussian pairwise potentials, where the mean field approximation is modeled as a recurrent neural network, and the defined network is trained end-to-end refining the weights of the CNN model. Recently, Lin *et al.* [5] have also combined CNN and CRF models for learning patch-patch context between image regions, and have achieved the current state-of-the-art performance in semantic segmentation. One of the main ideas in [5] is to define CNN-based pairwise potential functions to capture semantic correlations between neighboring patches. Moreover, efficient piecewise training is applied for the CRF model in order to avoid repeated expensive CRF inference during the course of back-propagation.

In [96] a feed-forward neural network has been proposed for scene labeling. The long range (pixel) label dependencies can be taken into account by capturing sufficiently large input context patch, around each pixel to be labeled. The method [96] relies on a recurrent convolutional neural networks (RCNN), *i.e.* a sequential series of networks sharing the same set of parameters. Each instance takes as input both an RGB image and the predictions of the previous instance of the network. RCNN-based approaches are

known to be difficult to train, in particular, with large data, since long-term dependencies are vanished while the information is accumulated by the recurrence [98].

Byeon *et al.* [98] have presented long short term memory (LSTM) recurrent neural networks for natural scene images taking into account the complex spatial dependencies of labels. LSTM networks have been commonly used for sequence classification. These networks include recurrently connected layers to learn the dependencies between two frames, and then transfer the probabilistic inference to the next frame. This allows to easily memorize the context information for long periods of time in sequence data. It has been shown [98] that LSTM networks can be generalized well to any vision-based task and efficiently capture local and global contextual information with a low computational complexity.

State-of-the-art CNNs have the ability to perform segmentation on different kinds of input sources such as RGB or even RGB-D. Therefore a trivial way to incorporate depth information would be to stack it to the RGB channels and train the network on RGB-D data assuming a four-channel input. However, it would not fully exploit the structure of the scene encoded by the depth channel. This will be also shown experimentally in Section 4.4. By making use of deeper and wider network architecture one can expect the increase of the robustness and the accuracy. Hence, one may define a network architecture with more layers. Nevertheless, this approach would require huge dataset in order to learn all the parameter making the training infeasible even in the case when the parameters are initialized with a pre-trained network.

The State of the Arts on RGB-D Data. A new representation of the depth information has been presented by Gupta *et al.* [95]. This representation, referred to as HHA, consists of three channels: disparity, height of the pixels and the angle between of normals and the gravity vector based on the estimated ground floor, respectively. By making use of the HHA representation, a superficial improvement was achieved in terms of segmentation accuracy [95]. On the other hand, the information retrieved only from the RGB channels still dominates the HHA representation. As we shall see in Section 4.4, the HHA representation does not hold more information than the depth itself. Furthermore, computing HHA representation requires high computational cost. In this paper we investigate a better way of exploiting depth information with less computational burden.

Li *et al.* [103] have introduced a novel LSTM Fusion (LSTM-F) model that captures and fuses contextual information from photometric and depth channels by stacking several convolutional layers and an LSTM layer. The memory layer encodes both short- and long-range spatial dependencies in an image along vertical direction. Moreover, another LSTM-F layer integrates the contexts from different channels and performs bi-directional propagation of the fused vertical contexts. In general, these kinds of architectures are rather complicated and hence more difficult to train. In contrast to recurrent networks, we propose a simpler network architecture.

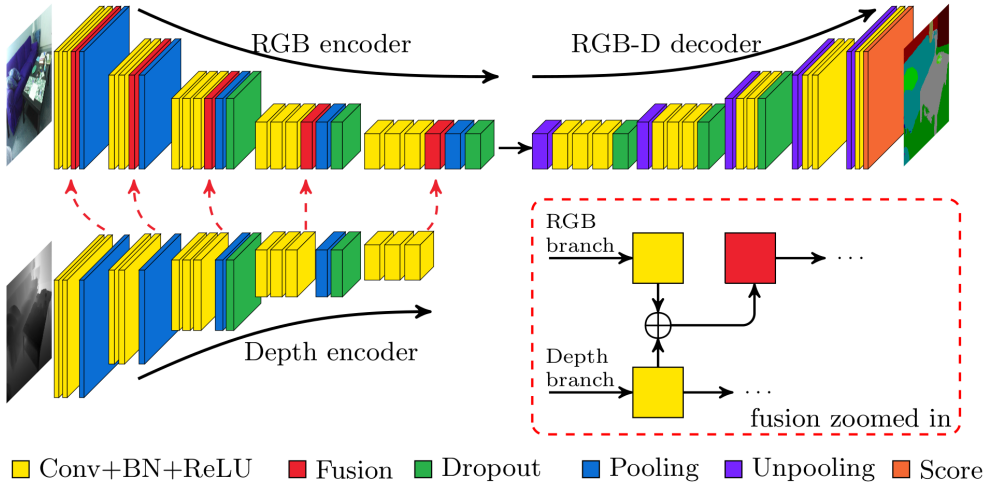


Figure 4.2: FuseNet architecture. Colors indicate the layer type. The network contains two branches to extract features from RGB and depth images, and the feature maps from depth is constantly fused into the RGB branch, denoted with the red arrows. In our architecture, the fusion layer is implemented as an element-wise summation, demonstrated in the dashed box.

4.3 A CNN-based Framework for Incorporating Depth into Semantic Networks

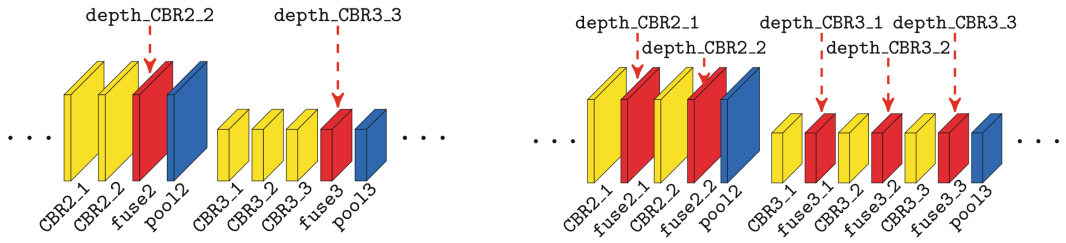
In this section, we detail the problem we tackled, provide details of FuseNet architecture in Section 4.3.1 and reason the fusion of two different input modalities, *i.e.* RGB and Depth in Section 4.3.2.

We aim to solve the semantic segmentation problem on RGB-D images. Let us define the label set as $\mathbb{L} = \{1, 2, \ldots, c\}$. We assume that we are given a training set $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ where $\mathbf{x}_i \in \mathbb{R}^{H \times W \times 4}, \mathbf{y}_i \in \mathbb{L}^{H \times W}$ consisting of N four-channel RGB-D images $(\mathbf{x}_i)$, having the same size $H \times W$, along with the groundtruth labeling $(\mathbf{y}_i)$. Moreover, we assume that the pixels are drawn as *i.i.d.* samples following a categorical distribution. Based on this assumption, we may define a CNN model to perform multinomial logistic regression.

The network extracts features from the input layer and through filtering provides classification score for each label as an output at each pixel. We model the network as a composition of functions corresponding to k layers with parameters denoted by $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k]$., as in Equation (2.4)

$$f(\mathbf{x}, \mathbf{W}) = g_k(g_{k-1}(\cdots g_2(g_1(\mathbf{x}, \mathbf{w}_1), \mathbf{w}_2) \cdots, \mathbf{w}_{k-1}), w_k) \,. \tag{4.1}$$

The classification score of a pixel $\mathbf{x}$ for a given class c is obtained from the function $f_c(\mathbf{x}, \mathbf{W})$, which is the cth component of $f(\mathbf{x}, \mathbf{W})$. Using the *softmax* function in Equation (2.1), we can map this score to a probability distribution



(a) Sparse fusion **(b) Dense fusion**

Figure 4.3: Illustration of different fusion strategies at the second (`CBR2`) and third (`CBR3`) convolution blocks of VGG 16-net. (a) Fusion layer is only inserted before each pooling layer. (b) Fusion layer is inserted after each `CBR` block.

$$\hat{p}(c \mid \mathbf{x}, \mathbf{W}) = \frac{\exp\left(f_c(\mathbf{x}, \mathbf{W})\right)}{\left\| \exp(f(\mathbf{x}, \mathbf{W}) \right\|_1} \,. \tag{4.2}$$

For the training of the network, *i.e.* learning the optimal parameters $\mathbf{W}^*$, the cross-entropy loss is used, which minimizes the KL-divergence between the predicted and the true class distribution:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \; -\sum_{i=1}^{N} \sum_{j=1}^{HW} \log \hat{p}(\mathbf{y}_{ij} \mid \mathbf{x}_{ij}, \mathbf{W}) + \lambda \left\| \mathbf{W} \right\|_2^2 \,, \tag{4.3}$$

where $\mathbf{x}_{ij} \in \mathbb{R}^4$ stands for the jth pixel of the ith training image and $\mathbf{y}_{ij} \in \mathbb{L}$ is its groundtruth label. The hyper-parameter $\lambda > 0$ is chosen to apply weighting for the regularization of the parameters (*i.e.* ℓ_2-norm (Equation (2.24)) of $\mathbf{W}$).

At inference, a probability distribution is predicted for each pixel via softmax normalization, defined in (Equation (4.2)), and the labeling is calculated based on the highest class probability.

4.3.1 FuseNet Architecture

We propose an auto-encoder-style network architecture as shown in Figure 4.2. The proposed network has two major parts: 1) the *encoder* part extracts features and 2) the *decoder* part upsamples the feature maps back to the original input resolution. This auto-encoder-style network has been already introduced in several previous works such as DeconvNet [49] and SegNet [57] and has achieved good segmentation performance. Although our proposed network is based on this type of architecture, we further consider to have two encoder branches. These two branches extract features from RGB and depth images. We note that the depth image is normalized to have the same value range as color images, *i.e.* into the interval of $[0, 255]$. In order to combine information from both input modules, we fuse the feature maps from the depth branch into the feature maps of the RGB branch. We refer to this architecture as *FuseNet* (see Figure 4.2).

The encoder part of FuseNet resembles the 16-layer VGG net [77], except of the fully connected layers `fc6`, `fc7` and `fc8`, since the fully connected layers reduce the resolution with a factor of 49, which increases the difficulty of the upsampling part. In our network, we always use batch normalization (`BN`) after convolution (`Conv`) and before rectified linear unit (`ReLU`) to reduce the internal covariate shift [28]. We refer to the combination of convolution, batch normalization and `ReLU` as `CBR` block, respectively. The `BN` layer first normalizes the feature maps to have zero-mean and unit-variance, and then scales and shifts them afterwards. Scale and shift parameters are learned during training, mean and variance are computed on the training set before testing. As a result, network learns how to combine color and depth features, which may have a different scale, in an optimal way.

The decoder part of the network is a counterpart of the encoder, where memorized unpooling is applied to upsample the feature maps. In the decoder part, we again use the `CBR` blocks. We also did experiments with deconvolution instead of convolution, and observed very similar performance. As proposed in [33], we also apply dropout both in the encoder and decoder to further boost the performance. However, dropout is deactivated during test time.

The key ingredient of the FuseNet architecture is the fusion block, which combines the feature maps of the depth and RGB branches. The fusion layer is implemented as element-wise summation. In FuseNet, we always insert the fusion layer after the `CBR` block. By making use of fusion the discontinuities of feature maps computed on the depth image are added into the RGB branch in order to enhance the RGB features. As it can be observed in many cases, the features in the color domain and in the geometric domain complement each other. Based on this observation, we propose two fusion strategies: a) dense fusion (`DF`), where the fusion layer is added after each `CBR` block of the RGB branch. b) sparse fusion (`SF`), where the fusion layer is only inserted before each pooling. These two strategies are illustrated in Figure 4.3.

4.3.2 Fusion of Feature Maps

In this section, we reason the fusion of the feature maps between the RGB and depth branches. To utilize depth information a simple way would be just stacking the RGB and depth images into a four-channel input. However, we argue that by fusing RGB and depth information the features are usually more discriminant than the ones obtained from the stacked input.

As we introduced before in (Equation (4.1)), each layer is modeled as a function g that maps a set of input x to a set of output $\mathbf{a}$ with parameter $\mathbf{w}$. We denote the lth feature map in the kth layer by $g_k^{(l)}$. Suppose that the given layer operation consists of convolution and `ReLU`, therefore

$$\mathbf{x}_k^{(l+1)} = g_k^{(l)}(\mathbf{x}^{(l)}, \mathbf{w}_l^{(l)}) = \texttt{ReLU}(\langle \mathbf{w}_k^{(l)}, \mathbf{x}^{(l)} \rangle + \mathbf{b}_k^{(l)}) \ . \tag{4.4}$$

If the input is a four-channel RGB-D image, then the feature maps can be decomposed as $x = [\boldsymbol{a}^\mathtt{T} \ \boldsymbol{b}^\mathtt{T}]^\mathtt{T}$, where $\boldsymbol{a} \in \mathbb{R}^{d_1}$, $\boldsymbol{b} \in \mathbb{R}^{d_2}$ with $d_1 + d_2 = d := \dim(x)$ are features learned

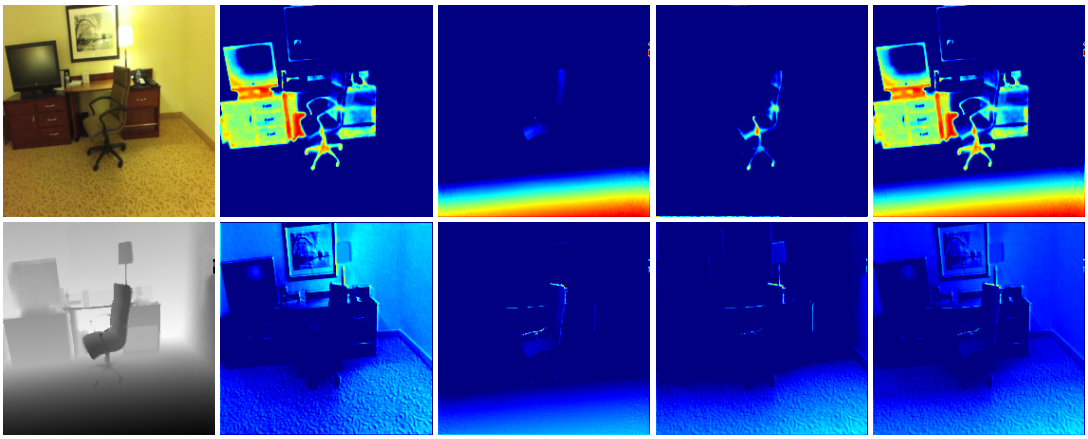


Figure 4.4: Comparison of two out of 64 feature maps produced at the `CBR1_1` layer. The features from RGB and depth mostly compensate each other, where the textureless region usually have rich structure features and structureless regions usually present texture features. This visually illustrates that the proposed fusion strategy better preserves the informative features from color and depth than applying element-wise summation followed by `ReLU`.

from the color channels and from the depth channel, respectively. According to this observation, we may write that

$$\begin{aligned}
\mathbf{x}_k^{(l+1)} &= \texttt{ReLU}(\langle w_k^{(l)}, x^{(l)} \rangle + b_k^{(l)}) = \texttt{ReLU}(\langle \mathbf{u}_k^{(l)}, \mathbf{a}^{(l)} \rangle + c_k^{(l)} + \langle \mathbf{v}_k^{(l)}, \mathbf{b}^{(l)} \rangle + d_k^{(l)}) \\
&= \max\left(\mathbf{0}, \langle \mathbf{u}_k^{(l)}, \mathbf{a}^{(l)} \rangle + c_k^{(l)} + \langle \mathbf{v}_k^{(l)}, \mathbf{b}^{(l)} \rangle + d_k^{(l)})\right) \\
&\leq \max(\mathbf{0}, \langle \mathbf{u}_k^{(l)}, \mathbf{a}^{(l)} \rangle + c_k^{(l)}) + \max(\mathbf{0}, \langle \mathbf{v}_k^{(l)}, \mathbf{b}^{(l)} \rangle + d_k^{(l)}) \\
&= \sigma(\langle \mathbf{u}_k^{(l)}, \mathbf{a}^{(l)} \rangle + c_k^{(l)}) + \sigma(\langle \mathbf{v}_k^{(l)}, \mathbf{b}^{(l)} \rangle + d_k^{(l)}) \,,
\end{aligned} \tag{4.5}$$

where we applied the decomposition of $\mathbf{w}_k^{(l)} = [\mathbf{u}_k^{(l)^\intercal} \; \mathbf{v}_k^{(l)^\intercal}]^\intercal$ and $\mathbf{b}_k^{(l)} = c_k^{(l)} + d_k^{(l)}$.

Based on the inequality in Equation (4.5), we show that the fusion of activations of the color and the depth branches (*i.e.* their element-wise summation) produces a stronger signal than the activation on the fused features. Nevertheless, the stronger activation does not necessarily lead to a better accuracy. However, with fusion, we do not only increase the neuron-wise activation values, but also preserve activations at different neuron locations. The intuition behind this can be seen by considering low-level features (*e.g.*, edges). Namely, due to the fact that the edges extracted in RGB and depth images are usually complementary to each other. One may combine the edges from both inputs to obtain more information. Consequently, these low-level features help the network to extract better high-level features, and thus enhance the ultimate accuracy.

To demonstrate the advantage of the proposed fusion, we visualize the feature maps produced by `CBR1_1` in Figure 4.4, which corresponds to low-level feature extraction (*e.g.*, edges). As it can be seen the low-level features in RGB and depth are usually

complementary to each other. For example, the textureless region can be distinguished by its structure, such as the lap against the wall, whereas the structureless region can be distinguished by the color, such as the painting on the wall. While combining the feature maps before the `ReLU` layer fail to preserve activations, the proposed fusion strategy applied after the `ReLU` layer preserves well all the useful information from both branches. Since low-level features help the network to extract better high-level ones, the proposed fusion thus enhances the ultimate accuracy.

4.4 Experimental Evaluation

In this section, we evaluate the proposed network through extensive experiments. For this purpose, we use the publicly available SUN RGB-D scene understanding benchmark [4]. This dataset contains 10335 synchronized RGB-D pairs, where pixel-wise annotation is available. The standard trainval-test split consists of 5050 images for testing and 5285 images for training/validation. This benchmark is a collection of images captured with different types of RGB-D cameras. The dataset also contains in-painted depth images, obtained by making use of multi-view fusion technique. In the experiments we used the standard training and test split with in-painted depth images. However, we excluded 587 training images that are originally obtained with RealSense RGB-D camera. This is due to the fact that raw depth images from the aforementioned camera consist of many invalid measurements, therefore in-painted depth images have many false values. We remark that the SUN RGB-D dataset is highly unbalanced in terms of class instances, where 16 out of 37 classes rarely present. To prevent the network from over-fitting towards unbalanced class distribution, we weighted the loss for each class with the median frequency class balancing according to [104]. In particular, the class *floormat* and *shower-curtain* have the least frequencies and they are the most challenging ones in the segmentation task. Moreover, approximately 0.25% pixels are not annotated and do not belong to any of the 37 target classes. Source code of our method is available on `https://github.com/tum-vision/fusenet`.

Training. We trained the all networks end-to-end. Therefore images were resized to the resolution of 224×224. To this end we applied bilinear interpolation on the RGB images and nearest-neighbor interpolation on the depth images and the groundtruth labeling. The networks were implemented with the CAFFE framework [105] and were trained with SGD optimizer (see Section 2.3.4) using a batch size of 4. The input data was randomly shuffled after each epoch. The learning rate was initialized to 0.001 and was multiplied by 0.9 in every $50,000$ iterations. We used a momentum of 0.9 and set weight decay to 0.0005. We trained the networks until convergence, when no further decrease in the loss was observed. The parameters in the encoder part of the network were fine-tuned from the VGG 16-layer model [77] pre-trained on the ImageNet dataset [79]. The original VGG net requires a three-channel color image. Therefore, for different input dimensions we processed the weights of first layer (*i.e.* `conv1_1`) as follows:

1) averaged the weights along the channel for a single-channel depth input;

Method	Global	Mean	IoU
FCN-16s [56]	67.51	38.65	27.15
FCN-32s [56]	68.35	41.13	29.00
Bayesian SegNet [33] (RGB)	71.20	45.90	30.70
LSTM [103]	-	48.10	-
Context-CRF [5] (RGB)	78.40	53.40	42.30
FuseNet-`SF5`	76.27	48.30	37.29
FuseNet-`DF1`	73.37	50.07	34.02

Table 4.1: Segmentation results on the SUN RGB-D benchmark [4] in comparison to the state of the art. Our methods `DF1` and `SF5` outperforms most of the approaches, except of the Context-CRF [5].

2) stacked the weights with their average for a four-channel RGB-D input;

3) duplicated the weights for a six-channel RGB-HHA input.

Testing. We evaluated the results on the original 5050 test images. For quantitative evaluation, we used three criteria. Let TP, FP, FN denote the total number of true positive, false positive, false negative, respectively, and N denotes the total number of annotated pixels. We define the following three criteria:

1) Global accuracy, referred to as *global*, is the percentage of the correctly classified pixels, defined as

$$\text{Global} = \frac{1}{N}\sum_c \text{TP}_c \ , \quad c \in \{1...K\} \ .$$

2) Mean accuracy, referred to as *mean*, is the average of classwise accuracy, defined as

$$\text{Mean} = \frac{1}{K}\sum_c \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c} \ .$$

3) Intersection-over-union (IoU) is average value of the intersection of the prediction and groundtruth regions over the union of them, defined as

$$\text{IoU} = \frac{1}{K}\sum_c \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c + \text{FN}_c} \ .$$

Among these three measures, the global accuracy is relatively less informative due to the unbalanced class distribution. In general, the frequent classes receive a high score and hence dominate the less frequent ones. Therefore we also measured the average class accuracy and IoU score to provide a better evaluation of our method.

Method	Global	Mean	IoU
FCN-16s [56]	65.40	46.10	34.0
FCN-32s [56]	64.30	44.90	32.80
Bayesian SegNet [33] (RGB)	68.00	45.80	32.40
LSTM [103]	-	49.40	-
Context-CRF [5] (RGB)	70.00	53.60	40.60
FuseNet-SF5	66.00	43.40	32.70

Table 4.2: Segmentation results on the NYUv2 dataset [6] in comparison to the state of the art. Our method SF5 outperforms most of the methods in terms of global accuracy (except of the Context-CRF [5]) and have on-par IoU scores.

4.4.1 Quantitative Results

In the first experiment, we compared our FuseNet to the state-of-the-art methods. The results are presented in Table 4.1. We denote the SparseFusion and DenseFusion by SF, DF, respectively, following by the number of fusion layers used in the network (*e.g.*, SF5). The results shows that FuseNet outperforms most of the methods with a significant margin. On the other hand, FuseNet is not as competitive in comparison to the Context-CRF [5]. However, it is also worth noting that the Context-CRF trains the network with a different loss function that corresponds to piecewise CRF training. It also requires mean-field approximation at the inference stage, followed by a dense fully connected CRF refinement to produce the final prediction. Applying the similar loss function and post-processing, FuseNet is likely to produce on-par or better results.

In the second experiment, we compare the FuseNet to network trained with different representation of depth, in order to further evaluate the effectiveness of depth and variant fusion variations. The results are presented in Table 4.3. It can be seen that stacking depth and HHA into color gives slight improvements over network trained with only color, depth or HHA. In contrast, with the depth fusion of FuseNet, we improve over a significant margin, in particular with respect to the IoU scores. We remark that the depth fusion is in particular useful as a replacement for HHA. Instead of preprocessing a single channel depth images to obtain hand crafted three-channel HHA representation, FuseNet learns high dimensional features from depth end-to-end, which is more informative as shown by experiments.

In Table 4.3, we also analyzed the performance of different variations of FuseNet. Since the original VGG 16-net has 5 levels of pooling, we increase the number of fusion layers as the network gets deeper. The experiments show that segmentation accuracy improves from SF1 to SF5, however the increase appears saturated up to the fusion after the 4th pooling, *i.e.* SF4. The possible reason behind the accuracy saturation is that depth already provides very distinguished features at low-level to compensate textureless regions in RGB, and we consistently fuse features extracted from depth into the RGB-branch. Same trend can be observed with DF.

Input	Global	Mean	IoU
Depth	69.06	42.80	28.49
HHA	69.21	43.23	28.88
RGB	72.14	47.14	32.47
RGB-D	71.39	49.00	31.95
RGB-HHA	73.90	45.57	33.64
FusetNet-SF1	75.48	46.15	35.99
FusetNet-SF2	75.82	46.44	36.11
FusetNet-SF3	76.18	47.10	36.63
FusetNet-SF4	76.56	48.46	37.76
FusetNet-SF5	**76.27**	48.30	**37.29**
FusetNet-DF1	73.37	**50.07**	34.02
FusetNet-DF2	73.31	49.39	33.97
FusetNet-DF3	73.37	49.46	33.52
FusetNet-DF4	72.83	49.53	33.46
FusetNet-DF5	72.56	49.86	33.04
FusetNet-DF13	70.54	45.83	30.03

Table 4.3: Segmentation results of FuseNet in comparison to the networks trained with RGB, depth, HHA and their combinations. The second part of the table provides the results of variations of FuseNet. We show that FuseNet obtained significant improvements by extracting more informative features from depth.

In the third experiment, we further compare FuseNet-SF5, FuseNet-DF1 to the network trained with RGB-D input. In Table 4.4 and 4.5, we report the classwise accuracy and IoU scores of 37 classes, respectively. For class accuracy, all the three network architectures give very comparable results. However, for IoU scores, SF5 outperforms in 30 out of 37 classes in comparison to other two networks. Since the classwise IoU is a better measurement over global and mean accuracy, FuseNet obtains significant improvements over the network trained with stacked RGB-D, showing that depth fusion is a better approach to extract informative features from depth and to combine them with color features.

Note that, due to the very low label frequency of the *floor mat* (denoted as *mat*) class, all models fail to segment the pixels of this class. Main reason is that *floor mat* neither has any flat depth information on the input depth map nor any textural cue on the RGB image.

In Figure 4.5, we demonstrate some visual comparison of FuseNet-SF5 to different input and fusion strategies. Our method has shaper boundaries and more accurate segmentation results. In Figure 4.6, we provide visual results of DFs. SF5 produces on-par segmentations compared to DF1. Nevertheless, deeper dense fusion models produce inconsistent noisy segmentation labels.

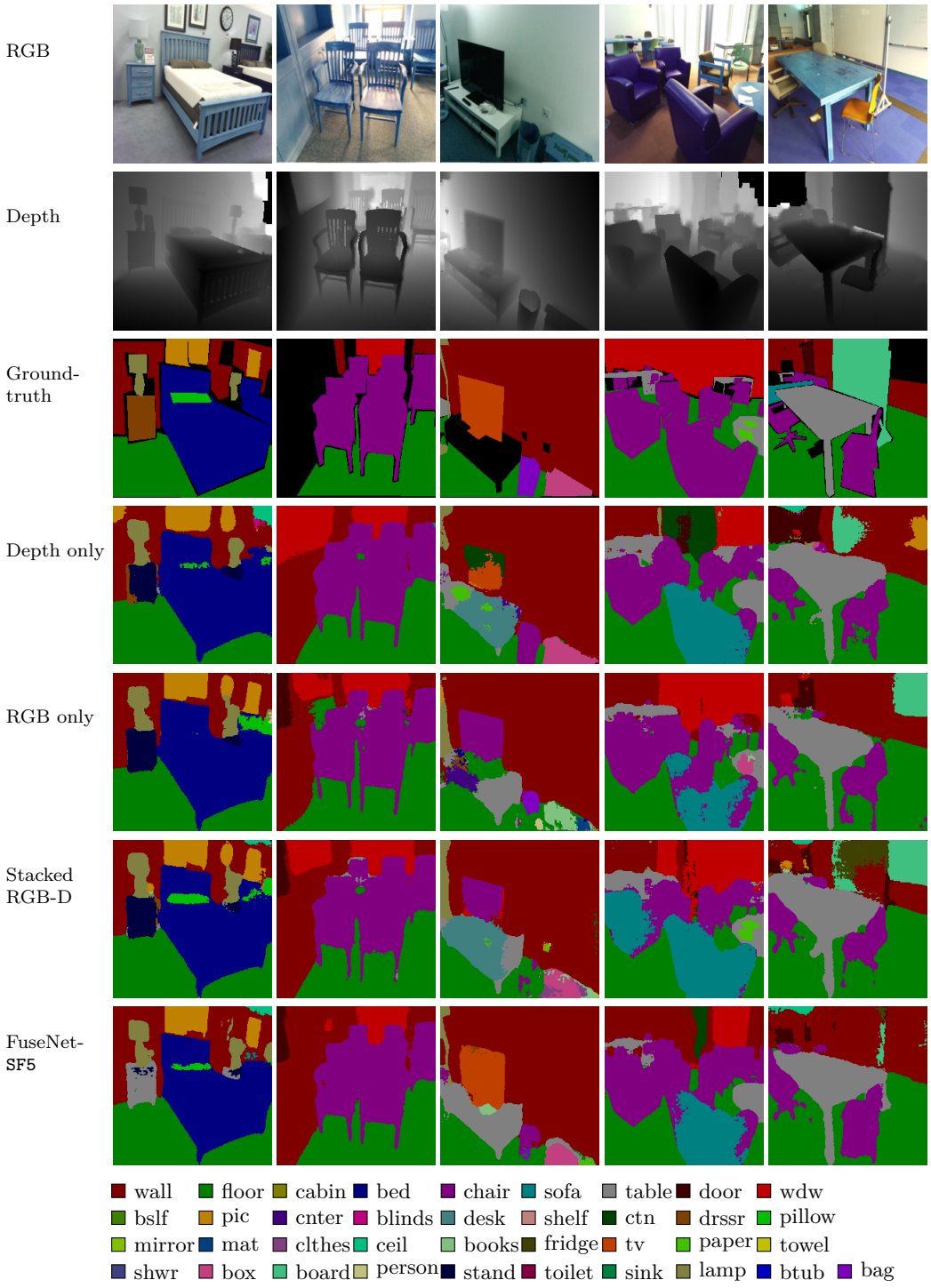


Figure 4.5: Qualitative segmentation results for different architectures. The first three rows contain RGB and depth images along with the groundtruth, respectively, followed by the segmentation results. Last row shows the results obtained by our SF5 model.

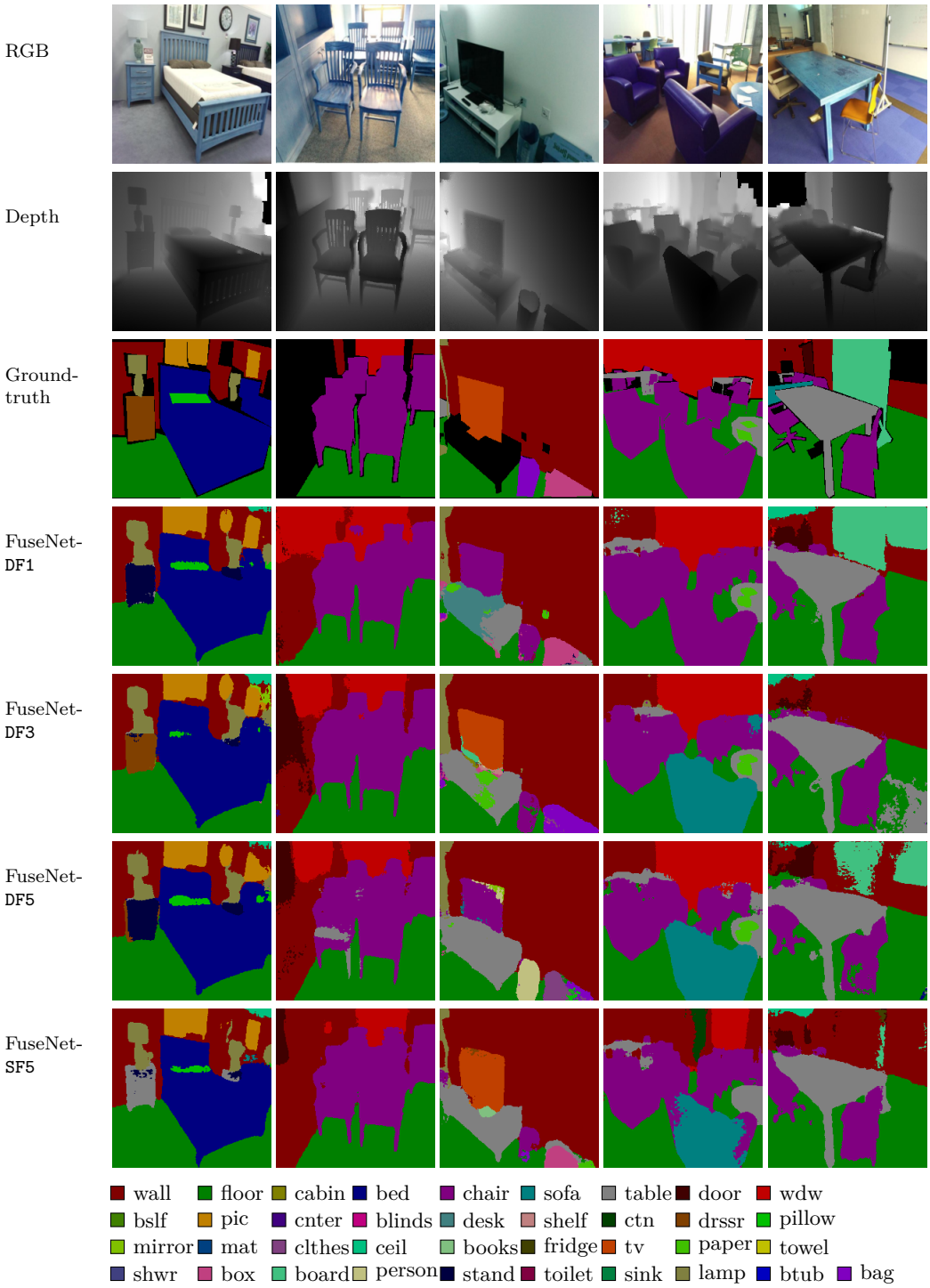


Figure 4.6: Qualitative segmentation results of FuseNet models. SF5 is more accurate than DFs in terms of correct semantic labels for the segmented regions.

	wall	floor	cabin	bed	chair	sofa	table	door	wdw	bslf
freq.	27.09	23.40	2.89	4.19	11.66	3.32	7.46	2.68	3.17	0.62
RGB-D	77.19	93.90	**62.51**	74.62	71.22	59.09	**66.76**	42.27	62.73	29.51
SF5	**90.20**	**94.91**	61.81	**77.10**	**78.62**	**66.49**	65.44	**46.51**	62.44	**34.94**
DF1	82.39	93.88	56.97	73.76	78.02	62.85	60.60	45.43	**67.22**	28.79

	pic	cnter	blinds	desk	shelf	ctn	drssr	pillow	mirror	mat
freq.	0.68	0.92	0.40	2.37	0.37	1.03	0.56	0.68	0.66	0.04
RGB-D	64.66	**48.19**	**48.80**	12.12	9.27	63.26	40.44	52.02	52.99	0.00
SF5	67.39	40.37	43.48	**25.63**	**20.28**	**65.94**	44.03	54.28	52.47	0.00
DF1	**67.50**	39.89	44.73	20.98	14.46	61.43	**48.63**	**58.59**	**55.96**	0.00

	clthes	ceil	books	fridge	tv	paper	towel	shwr	box	board
freq.	0.32	1.07	0.38	0.32	0.21	0.34	0.18	0.03	0.61	0.65
RGB-D	**38.38**	84.06	**57.05**	**34.90**	45.77	**41.54**	**27.92**	4.99	**31.24**	69.08
SF5	25.89	84.77	45.23	34.52	34.83	24.08	21.05	**8.82**	21.94	57.45
DF1	30.52	**86.23**	53.86	32.31	**53.13**	36.67	27.14	1.96	26.61	**66.36**

	person	stand	toilet	sink	lamp	btub	bag	mean
freq.	0.12	0.12	0.31	0.45	0.29	0.21	0.22	
RGB-D	16.97	42.70	76.80	**69.41**	50.28	65.41	24.90	49.00
SF5	19.06	37.15	76.77	68.11	49.31	73.23	12.62	48.30
DF1	**30.91**	**43.89**	**81.38**	66.47	**52.64**	**74.73**	**25.80**	**50.07**

Table 4.4: Classwise segmentation accuracy of 37 classes. We compare FuseNet-SF5, FuseNet-DF1 to the network trained with stacked RGB-D input. DF1 has the best classwise performance and outperforms other methods on 14 out of 37 classes. On the other hand, SF5 is compatative to DF1 and outperforms on 11 out of 37 classes. We also report the *frequency* of each class in per ten thousand (‰).

Segmentation results on the NYUv2 dataset. NYU Depth Dataset v2 [6] is composed of 1449 labeled benchmark and divided into 795 training and 654 test images. Although it is already involved in 37-class SUN RGB-D benchmark, we also test our best model, *i.e.* FuseNet-SF5, with the 40-class annotation provided in [106]. We train our models on 320×240 resolution. To obtain the original resolution of 640×480, one may use bilinear upsampling or a CRF-based refinement on the segmentation results similar to Context-CRF [5]. However, this may alter the final segmentation results and accuracy scores may differ. We compare the FuseNet-SF5 model with state-of-the-art methods in Table 4.2. FuseNet-SF5 outperforms most of them in terms of global accuracy (except Context-CRF [5]) and have on-par IoU scores.

	wall	floor	cabin	bed	chair	sofa	table	door	wdw	bslf
RGB-D	69.46	86.10	35.56	58.29	60.02	43.09	46.37	27.76	43.30	19.70
SF5	**74.94**	**87.41**	**41.70**	**66.53**	**64.45**	**50.36**	**49.01**	**33.35**	**44.77**	**28.12**
DF1	69.48	86.09	35.57	58.27	60.03	43.09	46.38	27.78	43.31	19.75

	pic	cnter	blinds	desk	shelf	ctn	drssr	pillow	mirror	mat
RGB-D	36.24	25.48	29.11	10.19	5.34	43.02	23.93	30.70	31.00	0.00
SF5	**46.84**	**27.73**	**31.47**	**18.31**	**9.20**	**52.68**	**34.61**	**37.77**	**38.87**	0.00
DF1	36.30	25.44	29.12	15.61	7.44	42.24	28.74	31.99	34.73	0.00

	clths	ceil	books	fridge	tv	paper	towel	shwr	box	board
RGB-D	**17.67**	63.10	21.79	22.69	31.31	12.05	13.21	4.13	14.21	40.43
SF5	16.67	**67.34**	**27.29**	**31.31**	31.64	16.01	**16.55**	**6.06**	**15.77**	49.23
DF1	15.82	60.09	24.28	23.63	**37.67**	**16.45**	13.60	1.54	15.47	**45.21**

	person	stand	toilet	sink	lamp	btub	bag	mean
RGB-D	10.00	11.79	59.17	45.85	26.06	51.75	12.38	31.95
SF5	14.59	**19.55**	**67.06**	**54.99**	**35.07**	**63.06**	9.52	**37.29**
DF1	**15.49**	17.46	63.38	48.09	27.06	56.85	**12.92**	34.02

Table 4.5: Classwise IoU scores of 37 classes. We compare FuseNet-SF5, FuseNet-DF1 to the network trained with stacked RGB-D input. SF5 has the best IoU performance and outperforms other methods on 30 out of 37 classes, showing that deeper fusion in the network yields better segmentation accuracy in terms of intersection-over-union.

4.5 Conclusions

This chapter presents a new fusion-based CNN network for semantic labeling on RGB-D data. More precisely, we have proposed a solution to fuse depth information with RGB data by making use of a convolutional network. The proposed network has an auto-encoder-style architecture, where the encoder part is composed of two branches that simultaneously extract features from RGB and depth channels, that are then fused into the RGB feature maps as the network goes deeper. We have investigated two possible fusion approaches, *i.e.* dense fusion and sparse fusion. By applying the latter one with a single fusion operation we have obtained a slightly better performance. Nevertheless we may conclude that both fusion approaches provide similar results.

By conducting a comprehensive evaluation, we conclude that the our approach is a competitive solution for semantic segmentation on RGB-D data. The proposed FuseNet outperforms the current CNN-based networks on the challenging SUN RGB-D [4] and NYUv2 [6] benchmarks. Interestingly, we can also claim that HHA representation itself provides a superficial improvement to the depth information.

We also remark that a straight-forward extension of the proposed approach can be applied for other classification tasks such as image or scene classification. Furthermore, our fusion network architecture is quite general and suitable for many other applications to fuse different input modalities efficiently.

Chapter 5
Deep Semantic Image Restoration

Figure 5.1: Given a corrupted image, proposed technique employs semantic knowledge learned during the training phase to restore the degraded input.

In the previous chapters, we first introduced a CNN-based depth reconstruction method from focal stacks in Chapter 3 and then a fusion-based CNN architecture for semantic image labeling on RGB-Depth data in Chapter 4. Semantics are important content cue on images that can be very informative in many vision algorithms, *e.g.*, SLAM, 3D scene reconstruction and also in image restoration.

In this chapter, we explore the complementarity of object categories to improve image restoration tasks, specifically single image deblurring (Figure 4.1). To this end, we propose a novel multimodal CNN architecture and training methodology that exploits pixelwise semantic labels in addition to color (RGB) information. Our technique processes semantically homogeneous regions of an image independently and learns to combine their output into the final result. Unlike other multimodal fusion strategies, a crucial aspect of our technique is that it confines the usage of annotations to the training phase. Semantic maps are notoriously tedious to obtain and therefore, their requirement beyond training would limit the usability of the technique in practical applications. Comprehensive evaluation demonstrates the effectiveness of the proposed architecture on three different semantic segmentation datasets, achieving substantially

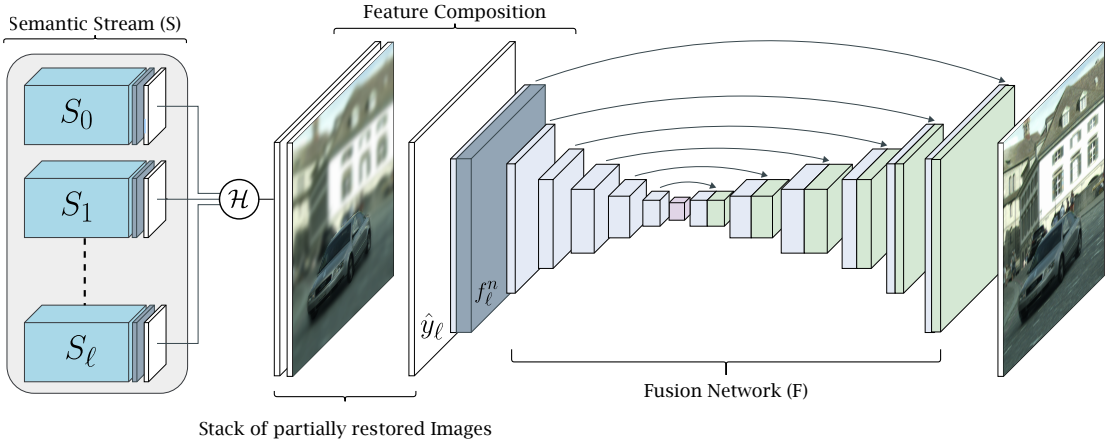


Figure 5.2: Multi-stream semantic network (MSN) architecture. A degraded input image is processed through each semantic stream S_ℓ, yielding a set of partially restored images $\hat{\mathbf{y}}_\ell$ and features $\mathbf{f}_\ell^n$. The output images $\hat{\mathbf{y}}_\ell$ are stacked together and given as input to the fusion network F, while feature maps $\mathbf{f}_\ell^n$ are summed-up element-wise and concatenated with the output of first convolutional block of F.

better performance compared to class-agnostic models and well established multimodal fusion techniques. This work has been done in a collaboration with Federico Perazzi[1], Christopher Schroers[2] and Daniel Cremers[3]. A research paper involving this chapter will be submitted to *IEEE International Conference on Computer Vision (ICCV) 2019*[4] in March, 22 2019.

5.1 Introduction

Image restoration refers to the recovery of an unknown *true* image from its degraded measurement. In the form of deblurring, denoising or super-resolution, image restoration is experiencing a renaissance as a result of convolutional neural networks (CNNs) establishing themselves as powerful generative models. These one-to-many inverse problems are inherently underdetermined and therefore it is crucial to leverage additional properties of the data to successfully recover the lost details. Exemplar based approaches, such as sparse dictionary learning, have defined the state-of-the art of image restoration for the past decade but were recently superseded by convolutional neural networks.

Recent techniques [107–109] directed their attention towards deeper convolutional architectures and improved loss functions. Despite promising results in similar fields, most of them neglected the availability of complementary information from other domains. Multimodal cues enabled significant improvements on several computer vision

[1]Adobe Research, email: fperazzi@adobe.com
[2]Disney Research, Zurich, email: christopher.schroers@disneyresearch.com
[3]Technical University of Munich, email: cremers@tum.de
[4]ICCV 2019, http://iccv2019.thecvf.com/

tasks. For example, segmentation strongly benefits from depth information [4], while motion measurements are fundamental to the task of recognizing actions [110].

Motivated by this observation, we conjecture that semantic information potentially plays an equally important role for image restoration tasks. But incorporating multimodal data in neural networks is not trivial. Although this problem is an active research topic, there is not yet a common consensus on the optimal solution. Instead, the current state of research indicates that the optimal multimodal fusion strategy might depend on the application. Furthermore, semantic maps are expensive to obtain and common fusion techniques require them to be available at test-time, which limits their deployment in real-life scenarios.

To this end, we propose a novel architecture which is specifically designed to guide image restoration with pixelwise semantic annotations. Our model combines advantages of recently proposed multimodal feature integration strategies [18, 110, 111]. Most importantly, it enables us to confine the usage of semantic information within the training stage. This aspect renders our technique suitable to a wide range of practical applications. Our model consists of a multi-stream structure cascaded with a fusion network. Individual streams are responsible for processing a single semantic class while the fusion network combines the output of the streams into the resulting image (Figure 5.2).

Extensive evaluation on three different semantic segmentation datasets consistently demonstrates the effectiveness of our approach compared to a number of well established fusion strategies and to other competitive approaches for the tasks of deblurring images.

5.2 Related Works

Class-agnostic Image Restoration. Classical image restoration problems such as denoising, deblurring and super-resolution are inherently underdetermined since multiple solutions exist for a given corrupted image. This class of problems requires suitable priors to mitigate their rank deficiency and to recover a satisfactory solution. Common priors are based on edge statistics [112–114], sparse representation [115–117], gradients [118–120], self-similarities [117, 121, 122] or a combination thereof [123]. The search for stronger priors recently ceased in lieu of the remarkable success of neural networks at recovering degraded images [124–131]. Currently, the dominant trend is that of investigating deeper network architectures with larger receptive fields and higher learning capacity [132].Unlike the aforementioned approaches, we confront the problem from a different perspective. In this work we aim to incorporate semantic information as an additional form of guidance to the problem of restoring corrupted images.

Semantic Image Restoration. The usage of semantic information to guide image restoration techniques is relatively unexplored. While different object classes have been addressed [133–136], a substantial body of literature focused their attention on human faces [137–143]. Despite producing good results, most of these methods tailored their solution around a specific object class, *e.g.*, human face, or application [144]. Hence, in contrast to our approach they do not generalize to complex natural images contain-

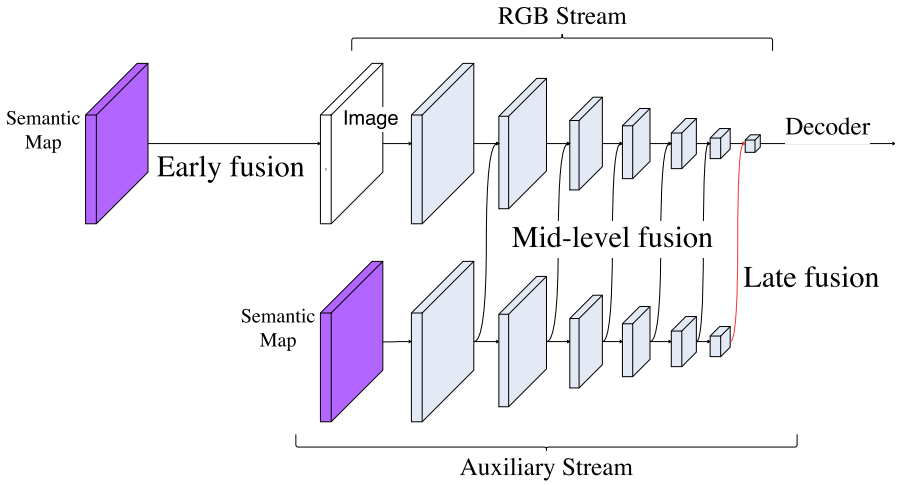


Figure 5.3: Multimodal fusion strategies for CNNs. Early fusion is implemented extending the input data layer of the RGB stream. Late fusion processes the modalities in separate streams and combines the features only in the latent space, whereas, mid-level fusion combines the features from the two streams progressively.

ing multiple object classes. Our idea of exploiting semantic information to help image restoration was inspired by Isola *et al.* [7]. They define the problem of *image-to-image* translation as the task of finding a mapping between two images from different types of representations. Given a semantic label map their model is able to synthesize a photo-realistic image that conforms to the input layout. The ability of convolutional networks to synthesize photographic images from semantic labels lead us to investigate them as an additional guidance for image restoration applications. Recently Chen and Koltun [145], proposed an alternative approach that synthesizes multiple diverse images from the same semantic layout. Similarly to our technique, they score the quality of results based on a virtual image that is assembled selecting the best result for each semantic class. Differently from us, however, the assembled image is only a proxy used during training to encourage diversity and not the end goal of their application. While their approach is different and therefore not directly comparable to our work, it is interesting to note that they generate multiple images from a single stream while we generate a single image combining the output of multiple streams.

5.3 Semantic Image Restoration

Let $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ denote a degraded image and $\mathbf{y} \in \mathbb{R}^{H \times W \times C}$ the corresponding true image. Given a training dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$, solving a class-agnostic image restoration problem entails learning a function $f : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{H \times W \times C}$ that generates an image $\hat{\mathbf{y}} = f(\mathbf{x})$, as similar as possible to the true image $\mathbf{y}$. Without loss of generality, let us define an arbitrary degradation process $\delta : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{H \times W \times C}$. We define the

objective as the function f that minimizes the reconstruction error between the true image $\mathbf{y}$ and the network output $\hat{\mathbf{y}}$:

$$f = \underset{f}{\operatorname{argmin}}\ \mathbb{E}_{\mathbf{y}}||f(\mathbf{x}) - \mathbf{y}||_1 \tag{5.1}$$

The function f is learned from the data as an approximation to the inverse of the degrading operator δ. This formulation enable the treatment of different image restoration problems such as deblurring and denoising within the same unified framework. In practice, a parametric model, such as a convolutional network, can be trained to approximate f for an arbitrary restoration task simply selecting an appropriate degrading operator δ.

There exists several adaptations of the class-agnostic image restoration process described above that enables us to incorporate multimodal information, such as pixelwise semantic annotations into a convolutional network (Figure 5.3). Different data modalities can be fused together early in the network by extending the depth dimension of its first layer. Alternatively the fusion can be performed at later stages combining the output of two independent networks and progressively fusing the intermediate features. A visualization of the most common strategies is provided in Figure 5.3. Although these techniques have been found to work well in related field such as segmentation [111, 146], they are subject to the same shortcoming: they tightly couple the restoration process to the availability of the semantic annotation limiting their practical usability in real-life scenarios. A seemingly plausible workaround is to replace the human-annotated pixelwise labels with the output classification of a semantic segmentation network such as DeepLab [147]. However, the prediction is likely to be highly unreliable in case of severe degradation of the input image.

In summary, our analysis of the aforementioned techniques (further details in Section 5.5.1) indicates that semantic annotations are helpful to the task of reconstructing degraded images. Nevertheless, their usage should be confined to the training phase. Therefore, fusion strategies such as early, mid and late fusion, that extend the network architecture to explicitly incorporate the additional data cannot be employed. Instead, the auxiliary information should be passed through the loss function. In the following, we propose an approach that uses semantic information only during training and is more accurate than other fusion strategies.

5.4 Multi-stream Semantic Network

In order to establish high-level dependencies among image regions corresponding to different semantic classes and modalities while restricting the usage of semantic information to the training phase, we propose a novel architecture, the *multi-stream semantic network* (MSN). The architecture of MSN is illustrated in Figure 5.2. Given c semantic classes, the model is composed of a multi-stream architecture $S = \{S_\ell\}_{\ell=0}^{c}$ cascaded with a fusion network F that receives the output of the semantic streams $\{S_0, ..., S_c\}$ as input and generates the restored $\hat{\mathbf{y}}$ as output. The detailed structure is illustrated in

Figure 5.2. Each stream S_ℓ is given the same copy of the degraded image $\mathbf{x}$ and produces as output a partially restored image,

$$\hat{\mathbf{y}}_\ell = S_\ell(\mathbf{x}_c). \tag{5.2}$$

The resulting image $\hat{\mathbf{y}}_\ell$ is only restored in regions of $\mathbf{x}$ that are annotated with the label ℓ as illustrated in Figure 5.4. We denote these regions with Ω_ℓ. Therefore, the output of the semantic streams in S_ℓ is a set of partially reconstructed images $\{\hat{\mathbf{y}}_\ell\}_{\ell=0}^{c}$, each restored in different semantic regions such that their union covers the entire image, *i.e.* $\bigcup_{\ell=0}^{c} \Omega_\ell \rightarrow \Omega$. A simple, but suboptimal solution for assembling the set of partially recovered images $\{\hat{\mathbf{y}}_\ell\}_{\ell=0}^{c}$ is to copy and paste the restored regions from different images into the final result. The shortcoming of this approach is that since streams S_ℓ process $\mathbf{x}$ independently, their outputs $\hat{\mathbf{y}}_\ell$ do not take into account the complex inter-class dependencies that are necessary to generate satisfactory results. Instead we adopt a more principled approach and train a network F to combine the outputs of S_ℓ.

Let $\mathbf{f}_\ell^n \in \mathbb{R}^{\frac{m}{2} \times \frac{n}{2} \times d}$ denote the last feature layer of S_ℓ. The fusion network F receives $\theta = \{\theta_\ell\}_{\ell=0}^{c}$ as input and produces the restored image $\hat{\mathbf{y}}$ as output; $\theta_\ell = (\hat{\mathbf{y}}_\ell, \mathbf{f}_\ell^n)$ contains the last feature layer and the partially restored RGB image obtained from S_ℓ.

Given θ as input, the fusion network F generates the restored image,

$$\hat{\mathbf{y}} = (F \circ \mathcal{H})(\theta_0, \theta_1, ..., \theta_{l-1}). \tag{5.3}$$

$\mathcal{H}(\cdot)$ is a composite function that concatenates the set of partially restored images $\{\hat{\mathbf{y}}_l\}_{\ell=0}^{c}$ along their depth channel, and combines the features $\mathbf{f}_\ell^n$ with the element-wise sum, such that

$$\mathcal{H}(\theta) = [\hat{\mathbf{y}}_{0 \rightarrow l}, \sum_{i=0}^{l} \bar{\mathbf{f}}_i^n]. \tag{5.4}$$

As illustrated in Figure 5.2 the stack of RGB images is given to F as input data layer while the features are concatenated to the output of its first convolution.

5.4.1 Training

Let c denote the total number of classes in a semantically annotated dataset $\mathcal{D}_\ell = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{l}^{(i)})\}_{i=1}^{N}$, where $\mathbf{l} \in \{0, c\}^{m \times n \times 1}$ indicates the pixelwise annotated semantic map as shown in Figure 5.4. MSN is trained over $\mathcal{D}_\ell$ in two supervised stages. In the first stage, the semantic streams S_ℓ are independently trained to minimize the objective,

$$f_\ell = \underset{f_\ell}{\operatorname{argmin}}\ \mathbb{E}_{\mathbf{y}} ||f_\ell(\mathbf{x}) - \mathbf{y}_\ell||_1, \tag{5.5}$$

where $\mathbf{y}_\ell$ denotes the groundtruth image $\mathbf{y}$ degraded only in regions assigned to a label different from ℓ, *i.e.* $\bar{\Omega}_\ell = \Omega \backslash \Omega_\ell$. In the second phase we train the fusion network F to assemble the final result $\hat{\mathbf{y}}$. F is trained to minimize the reconstruction error defined in Equation (5.1). Initially we fix the parameters of S and only train F. Upon convergence, we jointly fine-tune the entire network MSN=$\{S, F\}$ to minimize the same objective. In

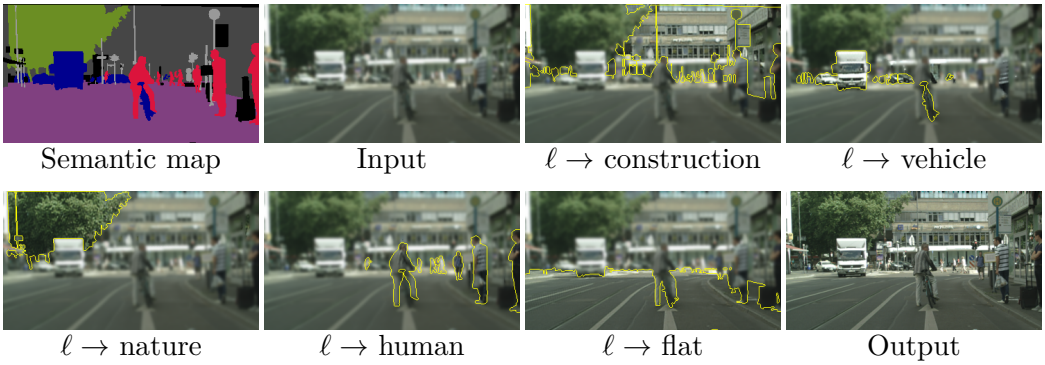


Figure 5.4: Semantic deblurring of individual classes from the Cityscapes dataset [2]. Each semantic network $S_{\ell=c}$ is trained to recognize and restore image regions corresponding to class c. Class boundaries are highlighted in yellow. Semantic annotations are shown for completeness but have not been used to produced the output.

our experiments we employ the ℓ_1-norm to minimize the reconstruction error since it was found to produce sharper results than the ℓ_2-norm [109]. Given a triplet $(\mathbf{x}, \mathbf{y}, \mathbf{l}) \in \mathcal{D}_\ell$, the overall training objective is defined as,

$$\begin{aligned}\mathcal{L}(F, S) = (1 - \mathbf{t}) \sum_{\ell=0}^{n} \left\| S_\ell(\mathbf{x}) - \mathbf{y}_\ell \right\|_1 \\ + \mathbf{t} \left\| F(\mathcal{H}(S_{0 \rightarrow \ell}(\mathbf{x}))) - \mathbf{y} \right\|_1 .\end{aligned} \tag{5.6}$$

The hyperparameter $\mathbf{t}$ balances the contribution of the two terms and is set to 0 during the first training phase and to 1 in the second.

5.4.2 Implementation Details

Architecture. Our architecture is inherently modular. The semantic streams S_ℓ and the fusion network F are structurally similar to the state-of-the-art auto-encoder of [7]. We only change the width of the network and adapt F to incorporate the features $\mathbf{f}_\ell^n$ from S as illustrated in Figure 5.2. We have chosen [7] because their publicly available code provided us with a solid framework and demonstrated good performance in generative tasks. Other approaches [131, 148] have chosen a similar encoder-decoder architecture, while the state-of-the-art deblurring network of [129] proposes a multi-scale network that processes the Gaussian pyramid of the degraded input.

The encoder branch is a contracting path. It transforms the input data into a low-dimensional latent representation, known as a context-vector, through the repeated application of 8 convolutional blocks (`Conv-InstanceNorm-ReLU`, 4×4 kernels and stride 2). The decoder is symmetric to the encoder and reconstructs the output, expanding the context-vector through a series of transposed convolutions. The network is augmented with skip-connections [149] that circumvent the bottleneck layer, enabling the propagation of localized context information into higher-resolution layers.

Conveniently, the width of the architecture is parameterized with a constant k that controls the number of filters at each layer. In our experiments, we ensure comparable learning capacity of the different networks by adjusting this parameter. Let P denote the target number of parameters—the learning capacity of S_ℓ—and F is set to $\mathrm{P}/(c+1)$ where c is the total number of semantic classes. In our experiments P is set to 3.97×10^8 for MSN models, which allows for training on a single GPU with 12 GB of memory.

Training. We use the training parameters provided in [7]. We employ Adam (see Section 2.3.4) as optimizer. The initial learning rate is set to $2{\cdot}10^{-4}$ and has a step decay. The batch size is 1. Each training phase consists of 200 epochs. Input images are downsampled to 286×286 resolution through bicubic interpolation and randomly cropped to size 256×256. The RGB space is normalized to zero mean and unit standard deviation. The semantic labels are resized accordingly but downsampled with nearest-neighbor interpolation.

5.5 Experiments

In the following, we describe the experiments conducted to determine the effectiveness of our technique. We provide an ablation study (Section 5.5.1) to analyze alternative fusion strategies compared to our multi-stream architecture MSN (Table 5.1). In Section 5.5.2 we compare the results of MSN with a number of state-of-the-art approaches that only employ the color information. We quantify the performance using the peak signal-to-noise ratio (PSNR). The experiments are conducted on images artificially blurred with a Gaussian kernel of size=$\{17, 33, 49\}$ and $\sigma = \{2, 4, 6\}$. Exemplar degraded inputs are shown in Figure 5.5.

Datasets. To ensure the coverage of sufficiently diverse semantic classes comprising indoor and outdoor scenes, we evaluate our model on three different semantic segmentation datasets.

Facades [2] is a small dataset depicting rectified images of buildings from the frontal view. It consists of 400 training, 100 validation and 106 test images. The 13 annotated classes define various elements of the building facades such as “window”,“door”, “balcony”. Our results are reported on the test set.

NYUv2 [6] includes a variety of indoor scenes recorded by an RGB-Depth sensor. In our experiments we do not exploit the depth data and instead only use the semantic maps provided for a subset of the dataset. We evaluate our technique on the 13-class partition with 795 images for training and 694 for testing provided in [104].

Cityscapes [150] is a recent large-scale dataset containing video sequences of street scenes recorded in 50 different cities. It is composed of 5000 accurately annotated images. Pixel-accurate semantic labels depict 8 different categories such as “human”, “nature” and “vehicle”. In our experiments, we train on 2975 images and evaluate the results on the 500 images of the validation set.

		Facades			NYUv2			Cityscapes		
Variants	Method	$\sigma = 2$	$\sigma = 4$	$\sigma = 6$	$\sigma = 2$	$\sigma = 4$	$\sigma = 6$	$\sigma = 2$	$\sigma = 4$	$\sigma = 6$
–	–	23.00	20.23	18.97	22.64	19.43	17.92	25.57	22.89	21.66
Baseline	SSN	31.35	26.18	23.58	33.48	28.08	25.59	36.14	29.98	27.61
Early fusion	SSN_ℓ	31.60	26.19	23.59	33.62	28.26	25.51	36.40	30.30	27.61
Mid-level fusion	TSN_{mid}	32.17	26.40	23.74	34.13	28.37	25.56	36.44	30.28	27.33
Late fusion	TSN_{late}	31.91	26.31	23.76	34.11	28.23	25.70	36.56	30.33	27.55
without ℓ	MSN	**32.15**	**26.55**	**23.92**	**34.31**	**28.65**	**26.06**	**36.59**	**31.46**	**28.29**

Table 5.1: Analysis of multimodal fusion techniques at different degradation levels. Independently from the strategy, additional semantic information improves the results. Our approach MSN is performing best on all three semantic segmentation datasets despite it confines the use of semantic maps to the training phase. The improvement is substantial at higher degradation levels ($\sigma = \{2, 4\}$). For such cases, TSN_{mid} and TSN_{late} are unable to learn meaningful features from the degraded input (visualized in Figure 5.5) and only perform comparably to SSN_ℓ. Mean PSNR score of the input images is given in the first row.

5.5.1 Multimodal fusion of semantic annotations

Although our final goal is to restore the original image without having semantic knowledge of its content, as an ablation study we first analyze different strategies of incorporating the semantic labels and compare the results with our multi-stream semantic network MSN. We categorize these approaches based on the depth at which the auxiliary data (semantic labels in our case) are combined together, *i.e.* early fusion (stacked image-label input), mid-level fusion [18] (also introduced in Chapter 4) and late fusion [56, 151]. As depicted in Figure 5.3, they all extend the class-agnostic baseline network described next.

RGB baseline. Our baseline (SSN) is a single-stream network similar to [7]. It is an encoder-decoder equipped with U-net [149] style skip-connections. It is structurally similar to the semantic streams S_ℓ of MSN. Given a degraded image $\mathbf{x}$, this model is trained to minimize Equation (5.1) using solely the color information.

Early fusion. A simple transformation of the baseline network SSN to incorporate semantic information is to augment the RGB input with the corresponding pixelwise semantic map $\mathbf{l}$. Channel augmentation is achieved by extending the input tensor $\mathbf{x}$ along its depth, *i.e.* $\mathbf{x}_c = [\mathbf{x}, \mathbf{l}] \in \mathbb{R}^{H \times W \times 4}$. Combining multiple sources of information at the level of the input layer is often referred to as *early* fusion [152]. Integrating different data modalities at such an early stage establishes precise pixel-level connectivity, which is beneficial for accurately localizing edges, but it is suboptimal for capturing higher contextual information.

Late fusion. We extend the baseline network SSN with an independent stream dedicated to semantic data. In practice, we augment SSN with a separate encoder branch

that processes the semantic information independently and combines its output in the context vector (Figure 5.3). Although late fusion techniques capture large receptive fields, they are limited to model interdependencies of different modalities at coarser spatial resolutions, therefore omitting local cues.

Mid-level fusion. This modality is an extension of early and late fusion along with connections that progressively combine features of intermediate layers. It has been shown that intermediate fusion is important for capturing the complementary information at each level from the auxiliary stream [18, 146].

To adequately compare the results of different architectures, we set their learning capacity to have a similar number of trainable parameters ($\sim 3.97 \times 10^8$). Therefore the size of the two-stream networks is reduced to accommodate the additional encoder. The size of the MSN modules S_ℓ and F is reduced accordingly based on the number of semantic classes (Section 5.4.2).

Results and Discussion. Table 5.1 shows that semantic information is beneficial to all fusion variants when compared to the class-agnostic baseline SSN, but with different degrees of improvement. While in our experiments we found that augmenting the input layer with semantic information (SSN_ℓ) consistently produces better results than SSN, the performance increase is marginal when compared with the effort required to obtain semantic labels. Substantial gains can be achieved by fusing heterogeneous data at later stages. The two-stream models implementing mid-level (TSN_{mid}) and late fusion (TSN_{late}) largely improve results compared to SSN but still require the availability of semantic segmentation at test time, which limits their usability in real-life scenarios. On the other hand, while having access to only color information, the proposed architecture MSN raises the PSNR substantially compared to SSN and in two datasets outperforms even the two-stream architectures that employ semantic annotations at test time.

5.5.2 Comparison to other approaches

In Table 5.2 we compare our technique with four recent approaches focusing on deblurring problems. Meinhardt *et al.* [153] (learned priors) propose to replace the proximal operator employed to regularize convex energy minimization algorithms with a denoising CNN. Their method generalizes well to different blurring kernels without the need of retraining the network. The performance however is inferior to pure CNN methods and considerably lower than MSN. We conducted the experiments of learned priors [153] with their publicly available implementation (`https://github.com/tum-vision/learn_prox_ops`). This method only requires the blurring kernel as an input and we did a small grid search on the hyperparameters on the small subset of training set of Facades dataset and used σ=1, α_{data}=41 and α_{data}=0 for all experiments.

The recent work of Pan *et al.* [115] (dark channel) estimates deblurring kernels by enforcing sparsity on the dark channel of blurred images. In our experiments, we set the kernel size to be identical to the size of the known blurring kernel and run all experiments

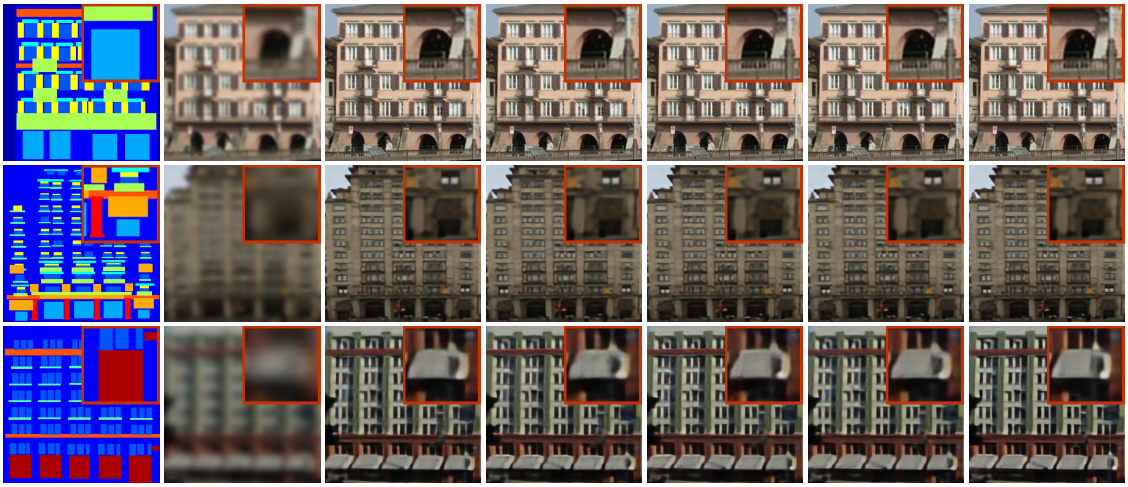


Figure 5.5: Visual comparison of different fusion strategies at different levels of degradation. Input images are degraded with a Gaussian filter of size=$\{17, 33, 49\}$ and $\sigma = \{2, 4, 6\}$ respectively. The numerical results in terms of PSNR are reported in Table 5.1.

	Facades			NYUv2			Cityscapes		
Method	$\sigma = 2$	$\sigma = 4$	$\sigma = 6$	$\sigma = 2$	$\sigma = 4$	$\sigma = 6$	$\sigma = 2$	$\sigma = 4$	$\sigma = 6$
Input	23.00	20.23	18.97	22.64	19.43	17.92	25.57	22.89	21.66
Dark channel [115]	24.53	21.50	19.62	24.16	17.84	15.58	27.40	24.29	22.51
Learned priors [153]	25.59	21.89	20.30	28.62	22.09	18.68	26.77	24.70	23.17
FFT [130]	26.39	20.74	20.52	29.39	20.50	18.98	29.32	23.41	23.89
VDSR [107]	27.35	23.32	21.41	30.91	25.42	22.95	31.62	26.67	24.60
DRRN [108]	27.68	-	-	31.39	-	-	31.53	-	-
DeepDeblur [129]	27.27	23.54	21.80	33.74	27.82	25.55	33.10	27.53	25.49
Pix2Pix [7] (SSN)	31.35	26.18	23.58	33.48	28.08	25.59	36.14	29.98	27.61
Proposed (MSN)	**32.15**	**26.55**	**23.92**	**34.31**	**28.65**	**26.06**	**36.59**	**31.46**	**28.29**

Table 5.2: Comparison to recent state-of-the-art approaches. The class-agnostic baseline adapted from Pix2Pix [7] outperforms all other methods. Nevertheless our technique MSN without the requirement of semantic annotations at test time further improves the baseline by an average of +0.66dB. Visual comparisons for each datasets are shown in Figures 5.8, 5.9, 5.10.

with their Matlab implementation (`http://vllab1.ucmerced.edu/~jinshan/projects/dark-channel-deblur/`). Analogously to learned priors [153] performance trail behind CNN techniques and are considerably below MSN.

Kruse *et al.* [130] (FFT) improves traditional fast Fourier transform techniques using a regularization based on CNNs. We tested the performance of FFT using the publicly available models. In FFT [130] implementation (`https://github.com/uschmidt83/`

fourier-deconvolution-network), there are two publicly available models trained for different range of σ, and this method requires the blurring kernel to process the images. For $\sigma \in \{2, 4\}$, we used the model trained for $\sigma \in [1, 3]$ and for σ=6, we used the model trained for a larger range of $\sigma \in [0.1, 12.75]$.

Finally, we evaluated the recent motion deblurring approach of Nah *et al.* [129] (DeepDeblur). DeepDeblur implements a coarse-to-fine approach using a multi-scale content loss computed on a three levels Gaussian pyramid of the input. We use their official and publicly available implementation (https://github.com/SeungjunNah/DeepDeblur_release). We use the same set of parameters and train until convergence (50 epochs for Facades, and $\sim$ 120 epochs for NYUv2 and Cityscapes). We report mixed results for DeepDeblur. On Facades the performance are substantially lower compared to the other datasets due to over-fitting after 40 epochs. While on Cityscapes and NYUv2 the performance of DeepDeblur are considerably better. In particular, DeepDeblur on NYUv2 have similar PSNR to our baseline Pix2Pix.

VDSR [107] and DRRN [108] were originally designed for super-resolution, but in practice, due to the nature of their upsampling routine, they operate as deblurrers. We trained a separate model of these methods on each dataset for 200 epochs. Both approaches were fine-tuned from their pre-trained models. For VDSR [107], we use the publicly available PyTorch (https://github.com/twtygqyy/pytorch-vdsr) implementation, which has identical performance for image resolution on Set5 with the original CAFFE implementation. We note that, for fine-tuning we used the available pretrained model for image super-resolution task, which we observed a relative improvement by means of approximately 1.5dB compared to training this method from scratch. We also note that this method is trained to restore the Y channel of YCbCr representation of an input RGB image. During evaluation we perform the method on the Y channel, however, compute the PSNR score on RGB images to have a fair comparison to our method. We test DRRN [108] using their publicly available code (https://github.com/tyshiwo/DRRN_CVPR17/tree/master/data). We retrain their network on our datasets using the given parameters and, similarly to what we did for VDSR, we generate the training data through the publicly available script. VDSR does not perform as good as our baseline SSN due to architectural differences of the networks. DRRN [108] proposes recursive learning of residual units to keep their very deep model compact and simple to train. It achieves performance similar to VDSR and substantially lower than our baseline SSN adapted from Pix2Pix [7]. From these experiments we conclude that we chose a competitive baseline and thus the higher PSNR of MSN over SSN is of significant importance.

Per-class evaluation. We evaluated the PSNR scores on each class and all the three segmentation datasets. We compared the performance with other fusion strategies. The results shown in Table 5.3, demonstrate that our multi-stream semantic network MSN delivers the best performance, compared to other fusion strategies, without the need of semantic maps at runtime. Analyzing per-class results on Facades and Cityscapes we note that our method achieves slightly worse performance in classes such as "background", "facade" and "sky". At first sight, it is plausible to think that more learning capacity should be assigned to streams dealing with most frequent classes, however we

$\ell \rightarrow$ *Facades*	$\ell_{\text{freq.}}$	SSN	SSN_ℓ	TSN_{mid}	TSN_{late}	MSN
background	19.47	32.79	33.07	**33.53**	33.26	33.34
balcony	4.01	29.06	29.27	29.78	29.59	**29.82**
blind	1.99	32.04	32.24	32.96	32.69	**32.98**
cornice	3.06	30.36	30.61	31.22	30.93	**31.34**
deco	1.88	30.52	30.75	31.29	31.14	**31.28**
door	1.26	33.04	33.22	33.87	33.54	**33.73**
facade	38.87	32.56	32.79	**33.39**	33.12	33.35
molding	9.23	31.41	31.62	32.13	31.91	**32.29**
pillar	2.11	32.64	32.85	33.53	33.29	**33.58**
shop	3.22	29.45	29.62	**30.34**	30.12	30.32
sill	2.37	30.40	30.69	31.22	31.00	**31.38**
window	12.54	29.76	30.05	30.69	30.43	**30.72**
mean	-	31.17	31.4	32.00	31.75	**32.01**
overall	-	31.35	31.60	32.17	31.91	**32.15**

$\ell \rightarrow NYUv2$	$\ell_{\text{freq.}}$	SSN	SSN_ℓ	TSN_{mid}	TSN_{late}	MSN
bed	3.51	34.72	34.81	35.49	35.30	**35.71**
books	0.54	29.21	29.47	30.20	29.98	**30.32**
ceiling	1.45	37.32	37.94	38.45	37.93	**38.19**
chair	3.38	34.44	34.52	35.19	35.05	**35.49**
floor	9.79	36.37	36.45	36.91	36.52	**37.19**
furniture	14.59	35.16	35.32	35.94	35.75	**36.18**
objects	12.87	32.04	32.25	32.98	32.76	**33.17**
picture	1.99	30.82	31.07	31.81	31.59	**31.95**
sofa	2.35	36.03	36.24	36.76	36.58	**36.97**
table	3.01	34.62	34.71	35.37	35.21	**35.64**
tv	0.55	36.79	36.96	37.43	37.35	**37.81**
wall	23.55	38.76	38.99	39.49	39.31	**39.73**
window	4.75	34.26	34.43	35.05	34.87	**35.30**
mean	-	34.66	34.86	35.47	35.24	**35.67**
overall	-	33.48	33.62	34.13	34.11	**34.31**

$\ell \rightarrow$ *Cityscapes*	$\ell_{\text{freq.}}$	SSN	SSN_ℓ	TSN_{mid}	TSN_{late}	MSN
construction	21.86	34.37	34.62	34.59	34.81	**34.76**
flat	38.77	42.49	42.74	42.73	42.86	**42.89**
human	1.19	32.34	32.65	32.84	32.82	**32.91**
nature	15.11	34.20	34.44	34.59	34.63	**34.74**
object	1.76	32.70	32.95	32.91	**33.14**	33.01
sky	3.57	38.63	38.93	38.69	**39.10**	38.95
vehicle	7.33	33.07	33.35	33.44	33.52	**33.59**
mean	-	35.40	35.67	35.68	**35.84**	**35.84**
overall	-	36.14	36.40	36.44	36.56	**36.59**

Table 5.3: PSNR scores per-class. Pixel frequencies of class ℓ are given in the second column. Facades: MSN outperforms the other fusion strategies on 9 out of 12 semantic classes. Interestingly, MSN performs slightly worse for frequent classes with low appearance variations such as "background" and "facades" and for rare classes with high internal variance such as "shop". NYUv2: our method performs best on all classes in this dataset. On Cityscapes: MSN outperforms the other fusion strategies in most cases. Similar to "shop" in Facades, MSN performs slightly worse on classes with limited amount of samples and high variance, in this case for "object". More details on these findings in Section 5.5.2.

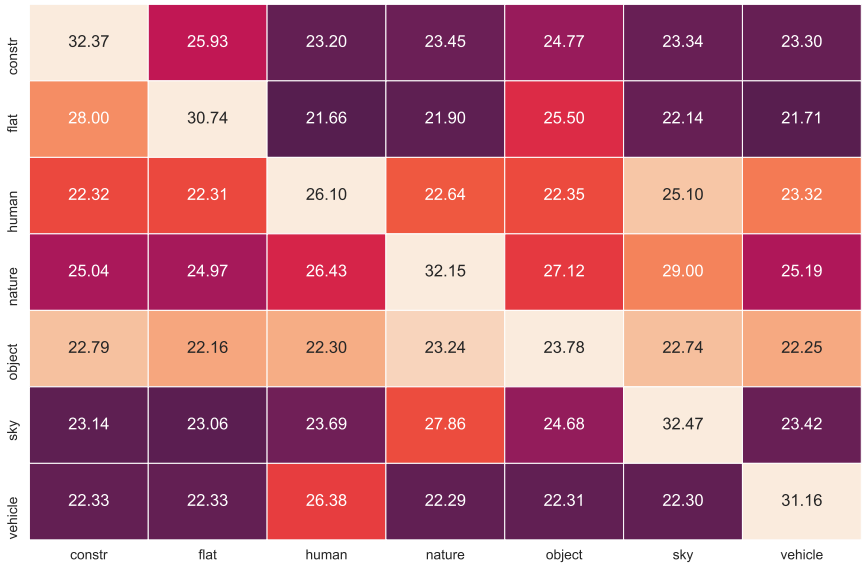


Figure 5.6: Performance of single semantic stream networks on the Cityscape dataset. Each row reports the PSNR of the respective network on each class. Cell colors are row-wise normalized. A brighter diagonal element indicates that the semantic stream obtain a higher PSNR for the class it was trained on.

note that these classes mostly cover homogeneous color regions with low variance. We conclude that the task assigned to these semantic streams is too easy and therefore they quickly converge to a suboptimal local minima. On the other hand, we note that MSN did not excel on infrequent classes with high internal variance such as "object" in Cityscapes and "shop" in Facades. While more samples from that classes will likely improve the results, we point out that our architecture enables the usage of additional datasets that only share a subset of the classes since semantic streams S_ℓ of MSN are first trained independently.

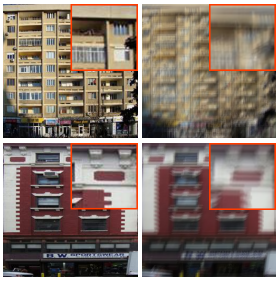
Per-class PSNR of intermediate results. We present the performance of single semantic stream networks on the Cityscape dataset in Figure 5.6. Each row reports the PSNR of the respective network on each class in the dataset. For example, cell R5/C4 correspond to the result of the semantic stream 'sky' on the class 'object'. It can be clearly seen that single semantic streams obtain a higher PSNR for the class they are trained on.

5.5.3 Comparison on motion deblurring

In Tab. 5.4, we provide additional experiments using the publicly available, motion-blur kernels of FFT. All approaches have been retrained for this two specific kernels. Figure 5.7 demonstrates groundtruth and degraded images. Our proposed method MSN outperforms all other methods on both motion kernels.

	Facades		NYUv2	
Method	$\kappa = 1$	$\kappa = 2$	$\kappa = 1$	$\kappa = 2$
Input	23.00	20.23	18.97	22.64
FFT	26.39	20.74	20.52	29.39
VDSR	27.35	23.32	21.41	30.91
DeepDeblur	27.27	23.54	21.80	33.74
Pix2Pix (SSN)	31.35	26.18	23.58	33.48
Proposed (MSN)	**32.15**	**26.55**	**23.92**	**34.31**

Table 5.4: PSNR evaluation on motion blurred images. We employ the kernels $\kappa = \{1, 2\}$ provided by FFT to degrade input images.



GT Input

Figure 5.7: Input degraded using FFT kernels $\kappa = \{1, 2\}$.

5.6 Conclusions

In this chapter, we investigated the complementarity of object categories to improve image restoration tasks. Our main contribution is a novel architecture, we termed as multi-stream semantic network (MSN), that employs previously learned semantic knowledge to recover both structure and fine details lost in the degraded image. An extensive set of experiments on three different semantic segmentation datasets demonstrate the effectiveness of our method compared to other state-of-the-art techniques and multimodal fusion strategies.

An interesting future work might address the scalability of the proposed approach to a higher number of classes. Scaling to higher number of classes linearly increases the computational resources necessary to train the semantic streams. Although trivially parallelizable, we believe that future works could improve the efficiency *e.g.*, investigating parameters sharing among different semantic streams.

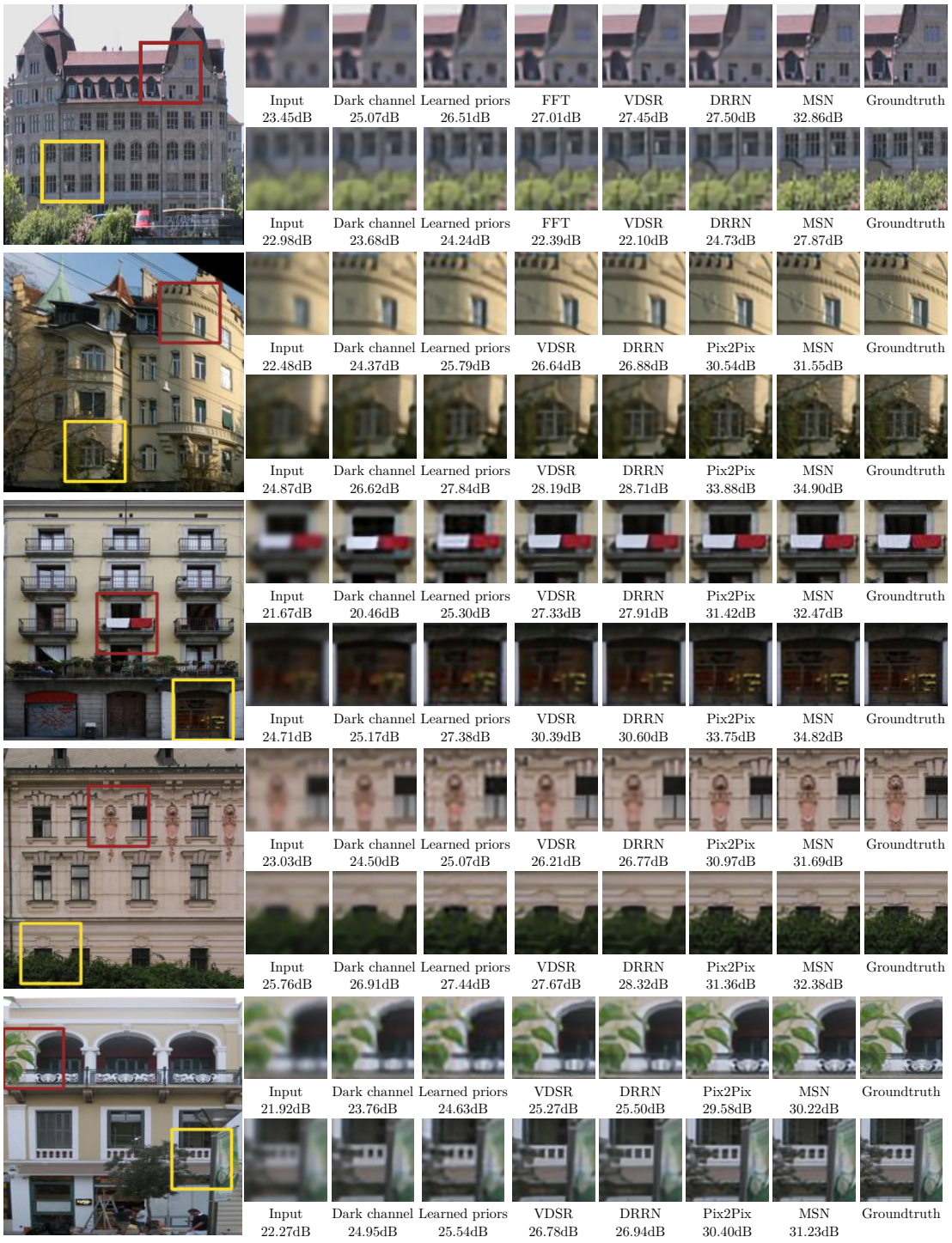


Figure 5.8: Qualitative results on the *Facades* dataset. We denote the results of SSN as Pix2Pix. MSN restores the details on the image while other methods do not recover the oversmooth regions.

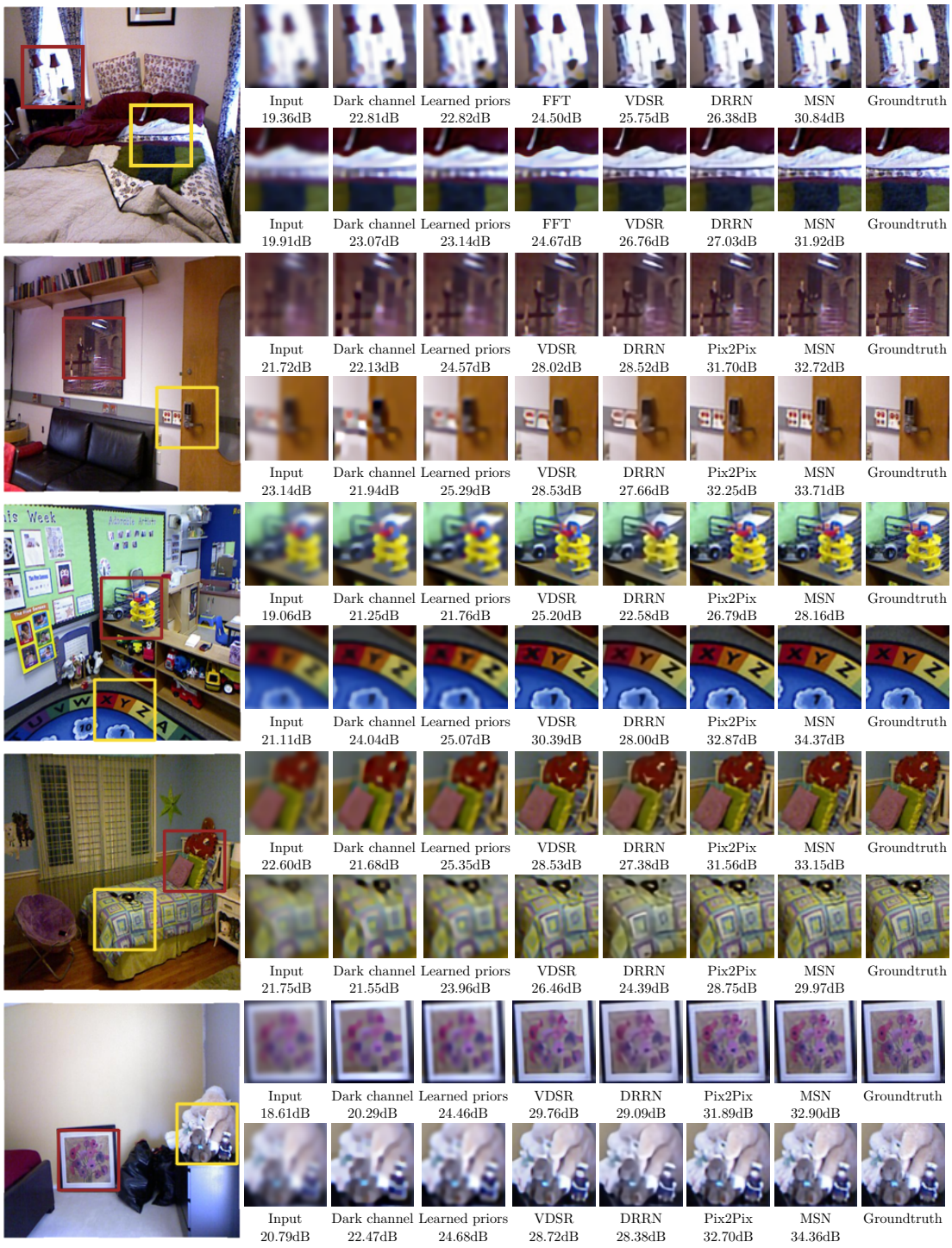


Figure 5.9: Qualitative results on the *NYUv2* dataset. We denote the results of SSN as Pix2Pix. MSN restores the details on the image while other methods do not recover the oversmooth regions.

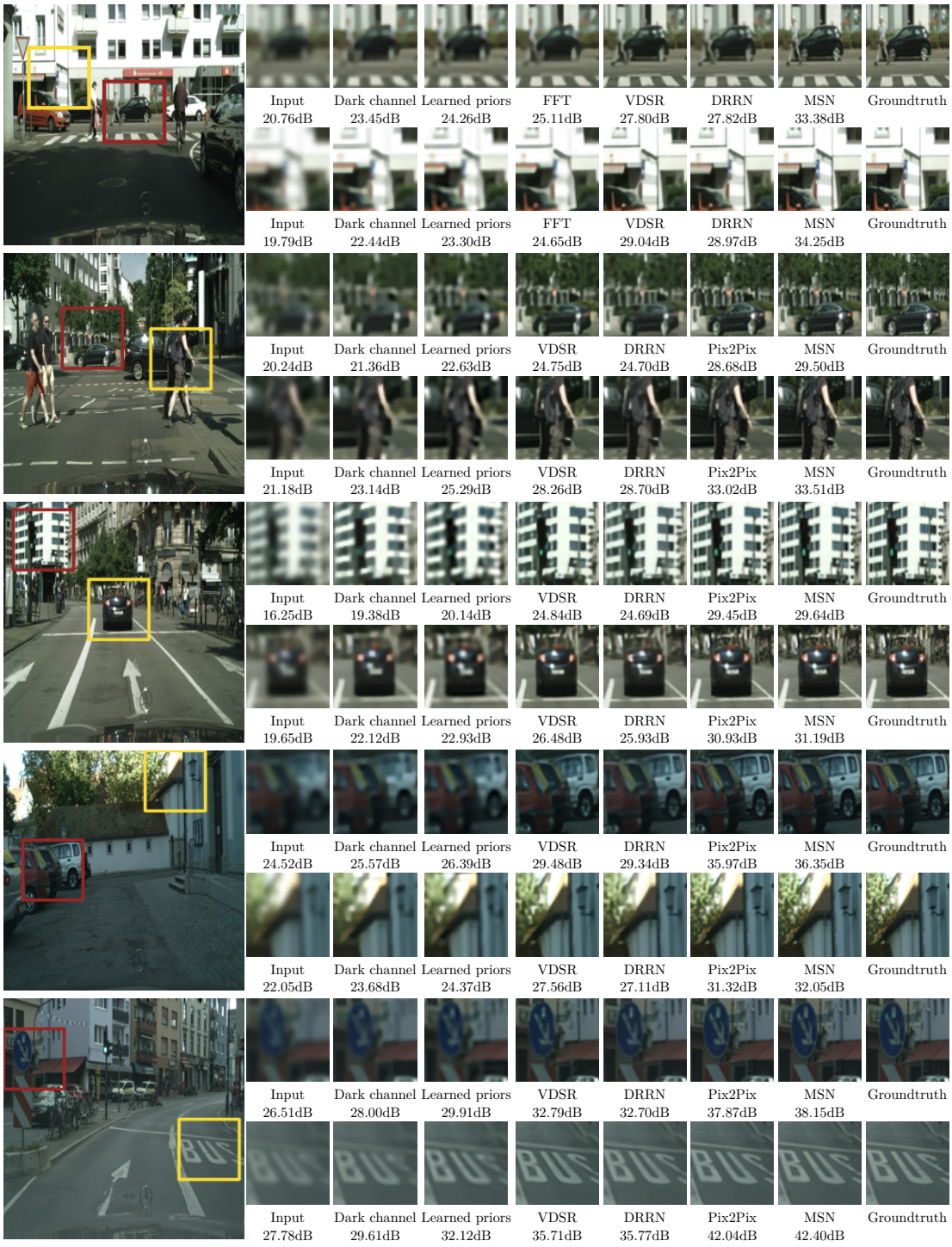


Figure 5.10: Qualitative results on the *Cityscapes* dataset. We denote the results of SSN as Pix2Pix. MSN restores the details on the image while other methods do not recover the oversmooth regions.

Part III

Conclusions & Outlook

Chapter 6
Concluding Remarks

Machines take me by surprise with great frequency.

— Alan Turing, *Computing Machinery and Intelligence (1950)*

This thesis investigated possible deep learning-based solutions for three challenging computer vision problems: reconstructing depth from focus cue (depth from focus), development of fusion strategies for RGB-Depth input modalities in semantic scene understanding and class-aware image restoration by exploiting hand-labeled annotations in a deep convolutional network. Each of these imaging problems were introduced in Part II and their concluding remarks are presented in the following.

Depth Reconstruction from Focus Cue. In Chapter 3 we first presented our large dataset which can be used not only for *depth from focus* problem but also many other computer vision tasks as the dataset consists of light-field images and co-registered groundtruth depth measurements. Further, we presented a novel convolutional network architecture for predicting the depth from focal stacks. We demonstrated that our *depth from focus* network is capable of processing focal stacks accurately and efficiently in terms of runtime compared to the state-of-the-art *depth from focus* and *depth from light-field* methods. Moreover, we show that our method is generalizable to other datasets and cameras.

Incorporating Depth into Semantic Segmentation. In Chapter 4 we discussed how to improve the performance of semantic segmentation by incorporating depth into the networks. We presented two novel fusion architectures to fuse the information retrieved from the complementary depth input with RGB features. We showed that our method can exploit the complementary information in an efficient way such that one would not need to preprocess inputs. Further, we experimentally showed that our approach is a competitive solution for semantic segmentation on RGB-D data.

Deep Semantic Image Restoration. In Chapter 5 we proposed a novel CNN architecture to exploit semantic annotations in the image restoration tasks. We demonstrated that complementarity of object categories significantly improves the results. To this end, we first discussed variant data fusion strategies, including *early, mid* and *late fusion* ap-

proaches and discussed the pros and cons of these strategies. Further, we presented a novel CNN architecture that requires the semantic annotations only during training and therefore it is generic to perform image restoration without the necessity of their semantic annotation at test time. We showed that our network can achieve outstanding performance on the image deblurring task compared to the state-of-the-art image restoration methods. Further, we conducted experiments in motion deblurring and hence showed that our method is generic to other image restoration tasks such as image denoising, super-resolution and demosaicking.

Chapter 7
Limitations and Future Work

We can only see a short distance ahead, but we can see plenty there that needs to be done.

— Alan Turing, *Computing Machinery and Intelligence (1950)*

7.1 Depth Reconstruction from Focus Cue

DDFF 12-Scene benchmark is not only generated for the *depth from focus* problem and it contains real-world light-field recordings along with co-registered groundtruth depth measurements and therefore this benchmark has new challenges can be tackled in many research areas. It is a convenient dataset for 3D reconstruction, depth from light-field, refocusing. On the other hand, as we generated this benchmark with a consumer light-field camera and therefore had to calibrate it with a public calibration toolbox. It has some drawbacks such as the baseline between sub-apertures is subpixel accurate. Another drawback was that we had to sacrifice the original recording image resolution due to the calibration procedure. However, despite these facts, our benchmark is still useful to evaluate different approaches.

DDFFNet is designed for a specific problem, however, its architecture is based on commonly used auto-encoder style convolutional networks. We visually demonstrated how network learns the focus cue, nevertheless the measure of sharpness learned by the network should be investigated deeper to improve the architecture for the problem of *depth from focus*.

DDFF is a deep learning based approach and can be easily integrated to mobile devices. However considering the fact that every device has different camera and therefore depth of field may vary, one should investigate to make the approach more sustainable and easily adaptable towards these factors. Our method has the necessity of a fine-tuning stage before it is deployed to a device and this could be changed by reformulating the problem: replacing the direct absolute depth estimation by index prediction (where each pixel is sharp in the focal stack).

DDFFNet can be also trained using stereo matching as in [55] or together with ℓ_2-loss in order to estimate more accurate depth maps. This way may reduce the artifacts in the results that have been caused by the wrong measurements in the recorded groundtruth depth maps.

7.2 Incorporating Depth into Semantic Segmentation

FuseNet is an efficient convolutional architecture proposed to fuse different input modalities in the same network. We evaluated two variant of fusion methods, *i.e.* *Dense-* and *SparseFusion*. Nonetheless these strategies were determined based on the VGG-16 net architecture and it could be generalized to different architectures as well by considering in which level of hierarchy fusion is necessary. To give an example, in the early layers of a network, features relating to fine structures of the objects present more frequent and therefore one could argue that *fusing* the features from these layers is essential to retrieve the complementary information. On the other hand, one may argue whether fusion is necessary in the decoder part of the network while recovering the finer structures from the coarser ones.

Moreover, although *FuseNet* is designed to tackle semantic image segmentation, the idea of fusion is beyond the application and can be exploited in different scenarios such as in image/scene classification, object detection, *etc.*

7.3 Deep Semantic Image Restoration

Our method is a unique way of exploiting semantics in image restoration tasks, however it has the limitation in terms of the annotated number of classes as the network grows linearly with the amount of predefined categories. Nonetheless, having a set of generic categories each of which consists variant object classes (such as 'vehicle' category would involve car, truck and *etc.*) allows our method cope with larger set of semantic annotations.

However, performance of the network may still decay if the objects present on the image are not observed during training. One way of solving this issue would be to train the semantic streams S_ℓ for a set of predefined classes and another semantic stream $S_{\ell \notin c}$ for the pixels whose categories are not defined or unknown. Although the fusion network F is capable of restoring the entire image, training a network for the unobserved objects/regions would yield improved results.

We presented results on two applications, *i.e.* Gaussian image deblurring and motion deblurring, although our method can be easily applied to other imaging tasks as well. Considering the similarity between deblurring and super-resolution, it is straight-forward to evaluate our network on super-resolution task as well.

Own Publications

1. C. Hazirbas, S. G. Soyer, M. C. Staab, L. Leal-Taixé, and D. Cremers. Deep depth from focus. In *Asian Conference on Computer Vision (ACCV)*, December 2018. URL: https://hazirbas.com/projects/ddff/, arXiv:1704.01085

2. C. Hazirbas, L. Ma, C. Domokos, and D. Cremers. FuseNet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian Conference on Computer Vision (ACCV)*, pages 213–228, Cham, 2016. Springer. URL: https://github.com/tum-vision/fusenet, doi:10.1007/978-3-319-54181-5_14

3. C. Hazirbas, J. Diebold, and D. Cremers. Optimizing the relevance-redundancy tradeoff for efficient semantic segmentation. In *Scale Space and Variational Methods in Computer Vision (SSVM)*, pages 243–255, Cham, 2015. Springer. URL: https://github.com/tum-vision/AFS, doi:10.1007/978-3-319-18461-6_20 (Note: Includes the results of Master's Thesis: C. HAZIRBAŞ. Feature Selection and Learning for Semantic Segmentation. MA thesis. Department of Informatics, Technische Universität München, June 2014.)

4. J. Diebold, N. Demmel, C. Hazirbas, M. Moeller, and D. Cremers. Interactive multi-label segmentation of RGB-D images. In *Scale Space and Variational Methods in Computer Vision (SSVM)*, pages 294–306, Cham, 2015. Springer. URL: https://github.com/NikolausDemmel/tvseg, doi:10.1007/978-3-319-18461-6_24

5. F. Stark, C. Hazirbas, R. Triebel, and D. Cremers. CAPTCHA recognition with active deep learning. In T. Villmann and F.-M. Schleif, editors, *German Conference on Pattern Recognition Workshop on New Challenges in Neural Computation (GCPRW)*, Machine Learning Reports 03/2015, October 2015. URL: https://github.com/tum-vision/captcha_recognition

6. A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, December 2015. URL: https://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15/, doi:10.1109/ICCV.2015.316

7. F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 627–637, October 2017. URL: http://hazirbas.com/datasets/tum-lsi, arXiv:1611.07890, doi:10.1109/ICCV.2017.75

8. T. Meinhardt, M. Moeller, C. Hazirbas, and D. Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1799–1808, October 2017. URL: https://github.com/tum-vision/learn_prox_ops, arXiv:1704.03488, doi:10.1109/ICCV.2017.198

9. N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy, and T. Brox. What makes good synthetic training data for learning disparity and optical flow estimation? *Internatioanl Journal of Computer Vision (IJCV)*, 126(9):942–960, September 2018. URL: http://lmb.informatik.uni-freiburg.de/Publications/2018/MIFDB18, arXiv:1801.06397, doi:10.1007/s11263-018-1082-6

References

[1] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *Asian Conference on Computer Vision (ACCV)*, 2016. xiv, 28, 33, 36, 37, 41, 43

[2] R. Š. Radim Tyleček. Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Computer Vision (GCPR)*, Saarbrucken, Germany, 2013. xv, 71, 72

[3] Y. Bok, H.-G. Jeon, and I. S. Kweon. Geometric calibration of micro-lens-based light field cameras using line features. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 39(2):287–300, Feb 2017. xvii, 27, 30, 31

[4] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A RGB-D scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, June 2015. xvii, 47, 49, 55, 56, 62, 67

[5] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Exploring context with deep structured models for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(6):1352–1366, June 2018. doi:10.1109/TPAMI.2017.2708714. xvii, 48, 49, 56, 57, 61

[6] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision (ECCV)*, 2012. xvii, 47, 49, 57, 61, 62, 72

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. xviii, 68, 71, 72, 73, 75, 76

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Internatioanl Journal of Computer Vision (IJCV)*, 60(2):91–110, November 2004. doi:10.1023/B:VISI.0000029664.99615.94. 3

[9] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008. doi:10.1016/j.cviu.2007.09.014. 3

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2564–2571, Nov 2011. 3

[11] Y. LeCun, Yannand Bengio and G. Hinton. Deep learning. *Nature*, 521:436, May 2015. doi:10.1038/nature14539. 3

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 4, 18

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. Curran Associates, Inc., 2012. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. 4, 18, 19

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, Cham, 2014. Springer. 4

[15] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, December 2015. URL: https://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15/, doi:10.1109/ICCV.2015.316. 4, 26, 28, 35

[16] N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, and T.Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. URL: http://lmb.informatik.uni-freiburg.de//Publications/2016/MIFDB16, arXiv:1512.02134. 4, 26, 28, 35

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017. 4, 20, 49

[18] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers. FuseNet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian Conference on Computer Vision (ACCV)*, pages 213–228, Cham, 2016. Springer. URL: https://github.com/tum-vision/fusenet, doi:10.1007/978-3-319-54181-5_14. 4, 26, 28, 47, 67, 73, 74, 105

[19] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946, 2015. 4, 28

[20] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 627–637, October 2017. URL: http://hazirbas.com/datasets/tum-lsi, arXiv:1611.07890, doi:10.1109/ICCV.2017.75. 4, 28

[21] C. Hazirbas, S. G. Soyer, M. C. Staab, L. Leal-Taixé, and D. Cremers. Deep depth from focus. In *Asian Conference on Computer Vision (ACCV)*, December 2018. URL: https://hazirbas.com/projects/ddff/, arXiv:1704.01085. 4, 25, 105

[22] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, USA, 2010. Omnipress. URL: http://dl.acm.org/citation.cfm?id=3104322.3104425. 9

[23] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013. 9

[24] D. C. andThomas Unterthiner andSepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. arXiv:1511.07289. 9, 10

[25] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 10, 11, 37

[26] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning (ICML)*, pages III–1319–III–1327. JMLR.org, 2013. URL: http://dl.acm.org/citation.cfm?id=3042817.3043084. 10

[27] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 249–256, 2010. URL: http://www.jmlr.org/proceedings/papers/v9/glorot10a.html. 11

[28] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, volume 37, pages 448–456. JMLR.org, 2015. 11, 35, 53

[29] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. URL: https://github.com/DmitryUlyanov/texture_nets, arXiv:1607.08022. 12

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, January 2014. 13, 49

[31] H. Lee, J. Lee, E. Yang, and S. J. Hwang. Dropmax: Adaptive stochastic softmax. *arXiv preprint arXiv:1712.07834*, 2017. arXiv:1712.07834. 13

[32] Y. Gal and Z. Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *International Conference on Learning Representations (ICLR) workshop track*, 2016. arXiv:1506.02158. 13

[33] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *The British Machine Vision Conference (BMVC*, 2017. arXiv:1511.02680. 28, 35, 49, 53, 56, 57

[34] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *International Conference on Robotics and Automation (ICRA)*, 2016. 13

[35] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012. 14

[36] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. arXiv:1212.5701. 15

[37] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. URL: http://dl.acm.org/citation.cfm?id=1953048.2021068. 15

[38] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, July 1992. doi:10.1137/0330046. 15

[39] Y. Xiao, Z. Wei, and Z. Wang. A limited memory bfgs-type method for large-scale unconstrained optimization. *Computers & Mathematics with Applications*, 56(4):1001–1009, 2008. doi:10.1016/j.camwa.2008.01.028. 15

[40] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *International Conference on Neural Networks (ICNN)*, pages 586–591 vol.1, 1993. doi:10.1109/ICNN.1993.298623. 15

[41] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. arXiv:1412.6980. 15

[42] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 2012. 15, 16

[43] P. J. Huber. *Robust Estimation of a Location Parameter*, pages 492–518. Springer, New York, NY, 1992. doi:10.1007/978-1-4612-4380-9_35. 17

[44] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same? *Neural Comput.*, 16(5):1063–1076, May 2004. doi:10.1162/089976604773135104. 17

[45] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 03 1951. doi:10.1214/aoms/1177729694. 17

[46] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr 1980. doi:10.1007/BF00344251. 18

[47] P. LeCun, Yannand Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–345, Berlin, Heidelberg, 1999. Springer. doi:10.1007/3-540-46805-6_19. 18

[48] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 20, 36

[49] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 20, 26, 28, 35, 48, 49, 52

[50] A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical report, Engineering Department, Cambridge University, Cambridge, UK, 1987. 21

[51] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339 – 356, 1988. doi:10.1016/0893-6080(88)90007-X.

[52] M. Mozer. A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, 3, 01 1995. 21

[53] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. doi:10.1162/neco.1997.9.8.1735. 22

[54] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL: http://www.aclweb.org/anthology/D14-1179. 22

[55] R. Garg, B. V. Kumar, G. Carneiro, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision (ECCV)*, pages 740–756. Springer, 2016. 25, 28, 36, 87

[56] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, Boston, MA, USA, June 2015. 26, 28, 35, 48, 49, 56, 57, 73

[57] V. Badrinarayanan, A. Handa, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv: 1505.07293*, 2015. arXiv:1505.07293. 28, 52

[58] V. Nekrasov, T. Dharmasiri, A. Spek, T. Drummond, C. Shen, and I. Reid. Real-time joint semantic segmentation and depth estimation using asymmetric annotations. *arXiv preprint arXiv:1809.04766*, 2018. arXiv:1809.04766. 26

[59] S. Pertuz, D. Puig, and M. A. Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition (PR)*, 46(5):1415–1432, May 2013. doi:10.1016/j.patcog.2012.11.011. 27

[60] A. Thelen, S. Frey, S. Hirsch, and P. Hering. Improvements in shape-from-focus for holographic reconstructions with regard to focus operators, neighborhood-size, and height value interpolation. *IEEE Transactions on Image Processing (TIP)*, 2009. 27

[61] M. Mahmood and T.-S. Choi. Nonlinear approach for enhancement of image focus volume in shape from focus. *IEEE Transactions on Image Processing (TIP)*, 2012. 27

[62] M. Mahmood. Shape from focus by total variation. In *(IEEE) Image, Video, and Multidimensional Signal Processing (IVMSP) Workshop*, 2013. 27

[63] M. Moeller, M. Benning, C. Schönlieb, and D. Cremers. Variational depth from focus reconstruction. *IEEE Transactions on Image Processing (TIP)*, 24(12):5369–5378, Dec 2015. 27, 36, 39, 40, 41

[64] S. Suwajanakorn, C. Hernandez, and S. M. Seitz. Depth from focus with your mobile phone. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3497–3506, June 2015. doi:10.1109/CVPR.2015.7298972. 27

[65] M. Asif and T. Choi. Learning shape from focus using multilayer neural networks. In *Proc. SPIE, Vision Geometry VIII*, 1999. 27

[66] M. Asif and T.-S. Choi. Shape from focus using multilayer feedforward neural networks. *IEEE Transactions on Image Processing (TIP)*, 10(11):1670–1675, Nov 2001. doi:10.1109/83.967395. 27

[67] E. Adelson and J. Wang. Single lens stereo with a plenoptic camera. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 1992. 27

[68] B. Wilburn, N. Joshi, V. Vaish, E. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. High performance imaging using large camera arrays. In *ACM Transactions on Graphics (TOG)*, 2005. 27

[69] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. Technical report, Stanford University Computer Science Tech Report CSTR 2005-02, 2005. 27

[70] D. Dansereau, O. Pizarro, and B. Williams. Decoding, calibration and rectification for lenselet-based plenoptic cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 27

[71] D. Cho, M. Lee, S. Kim, and Y.-W. Tai. Modeling the calibration pipeline of the lytro camera for high quality light-field image reconstruction. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 27

[72] H.-G. Jeon, J. Park, G. Choe, J. Park, Y. Bok, Y.-W. Tai, and I. S. Kweon. Accurate depth map estimation from a lenslet light field camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 27, 30

[73] F. Pérez-Nava and J. P. Lüke. Simultaneous estimation of super-resolved depth and all-in-focus images from a plenoptic camera. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, 2009.

[74] H. Lin, C. Chen, S. B. Kang, and J. Yu. Depth recovery from light field using focal stack symmetry. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 27

[75] S. Wanner, S. Meister, and B. Goldlücke. Datasets and benchmarks for densely sampled 4d light fields. In *Vision, Modeling and Visualization (VMV)*, pages 225–226, 2013. 28, 33

[76] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012. 28, 35

[77] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 35, 49, 53, 55

[78] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. `arXiv:1512.03385`. 28, 35

[79] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *Internatioanl Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 28, 55

[80] R. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 28

[81] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 28, 39

[82] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 28

[83] F. Liu, C. Shen, G. Lin, and I. D. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(10):2024–2039, 2016. doi:10.1109/TPAMI.2015.2505283. 28, 36

[84] S. Galliani and K. Schindler. Just look at the image: viewpoint-specific surface normal prediction for improved multi-view reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 28

[85] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 28

[86] I. Lytro. Lytro ILLUM lightfield camera. illum.lytro.com, 2006. Accessed: 2016-11-07. 30, 39

[87] M. Diebold and B. Goldluecke. Epipolar plane image refocusing for improved depth estimation and occlusion handling. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 30

[88] M. Y. Liu, O. Tuzel, and Y. Taguchi. Joint geodesic upsampling of depth images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 32

[89] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon. High quality depth map upsampling for 3d-tof cameras. In *IEEE International Conference on Computer Vision (ICCV)*, November 2011.

[90] J. Park, H. Kim, Y. Tai, M. S. Brown, and I. Kweon. High-quality depth map upsampling and completion for RGB-D cameras. *IEEE Transactions on Image Processing (TIP)*, 2014.

[91] J. Shen and S. C. S. Cheung. Layer depth denoising and completion for structured-light RGB-D cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 32

[92] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2366–2374, Cambridge, MA, USA, 2014. MIT Press. URL: http://dl.acm.org/citation.cfm?id=2969033.2969091. 36

[93] S. Heber and T. Pock. Convolutional networks for shape from light field. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 39

[94] S. Heber, W. Yu, and T. Pock. Neural epi-volume networks for shape from light field. In *IEEE International Conference on Computer Vision (ICCV)*, October 2017. 39

[95] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision (ECCV)*, volume 8695 of *Lecture Notes in Computer Science (LNCS)*, pages 345–360, Zurich, Switzerland, September 2014. Springer. 48, 50

[96] P. O. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *International Conference on Machine Learning (ICML)*, Beijing, China, June 2014. 49

[97] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1529–1537, Santiago, Chile, December 2015. 48, 49

[98] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with LSTM recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3547–3555, Boston, MA, USA, June 2015. 48, 50

[99] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015. 48, 49

[100] C. Couprie, C. Farabet, L. Najman, and Y. LeCun. Indoor semantic segmentation using depth information. In *International Conference on Learning Representations (ICLR)*, April 2013. 48

[101] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, volume 8689, pages 818–833, Zurich, Switzerland, September 2014. Springer. doi:10.1007/978-3-319-10590-1_53. 49

[102] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning (ICML)*, 2016. arXiv:1506.02142. 49

[103] L. Z. Li, G. Yukang, L. Xiaodan, Y. Yizhou, C. Hui, and L. Liang. RGB-D Scene labeling with long short-term memorized fusion model. *arXiv preprint arXiv:1604.05000v2*, 2016. arXiv:1604.05000v2. 50, 56, 57

[104] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2650–2658, 2015. doi:10.1109/iccv.2015.304. 55, 72

[105] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. arXiv:1408.5093. 55

[106] S. Gupta, P. Arbeláez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 564–571, June 2013. doi:10.1109/CVPR.2013.79. 61

[107] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 66, 75, 76

[108] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 75, 76

[109] Y. Tai, J. Yang, and X. Liu. Loss functions for neural networks for image processing. In *IEEE Transactions on Computational Imaging (TCI)*, 2017. 66, 71

[110] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 67

[111] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A.Sorkine-Hornung. Learning video object segmentation from static images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 67, 69

[112] A. Levin. Blind motion deblurring using image statistics. In *Advances in Neural Information Processing Systems (NIPS)*, 2006. 67

[113] R. Fattal. Image upsampling via imposed edge statistics. *ACM Transactions on Graphics (TOG)*, 2007.

[114] P. Qiu and P. S. Mukherjee. Edge structure preserving image denoising. *Signal Processing*, 2010. 67

[115] J. Pan, D. Sun, H. Pfister, and M. Yang. Blind image deblurring using dark channel prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 67, 74, 75

[116] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing (TIP)*, 2006.

[117] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 67

[118] J. Sun, Z. Xu, and H. Shum. Image super-resolution using gradient profile prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 67

[119] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 1992.

[120] D. Perrone and P. Favaro. Total variation blind deconvolution: The devil is in the details. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 67

[121] A. Buades, B. Coll, and J. Morel. A non-local algorithm for image denoising. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 67

[122] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics (TOG)*, 2006. 67

[123] J. Mairal, F. R. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *IEEE International Conference on Computer Vision (ICCV)*, 2009. 67

[124] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 2017. 67

[125] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision (ECCV)*, 2016.

[126] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[127] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[128] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2014.

[129] S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 71, 75, 76

[130] J. Kruse, C. Rother, and U. Schmidt. Learning to push the limits of efficient fft-based image deconvolution. In *IEEE International Conference on Computer Vision (ICCV)*, 2018. 75

[131] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. L. Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. *arXiv preprint arXiv:1711.07064*, 2017. arXiv:1711.07064. 67, 71

[132] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 67

[133] Y. Tsai, X. Shen, Z. Lin, K. Sunkavalli, and M. Yang. Sky is not the limit: semantic-aware sky replacement. *ACM Transactions on Graphics (TOG)*, 35(4), 2016. 67

[134] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[135] P. Svoboda, M. Hradis, L. Marsik, and P. Zemcík. CNN for license plate motion deblurring. In *IEEE International Conference on Image Processing (ICIP)*, 2016.

[136] J. Pan, Z. Hu, Z. Su, and M. Yang. Deblurring text images via l0 intensity and gradient prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 67

[137] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(9), 2002. 67

[138] G. Chrysos and S. Zafeiriou. Deep face deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.

[139] J. Pan, Z. Hu, Z. Su, and M. Yang. Deblurring face images with exemplars. In *European Conference on Computer Vision (ECCV)*, 2014.

[140] Y. Shih, V. Kwatra, T. T. Chinen, H. Fang, and S. Ioffe. Joint noise level estimation from personal photo collections. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[141] N. Joshi, W. Matusik, E. H. Adelson, and D. J. Kriegman. Personal photo enhancement using example images. *ACM Transactions on Graphics (TOG)*, 29(2), 2010.

[142] T. Remez, O. Litany, R. Giryes, and A. M. Bronstein. Deep class-aware image denoising. In *International Conference on Sampling Theory and Applications*, 2017.

[143] Z. Shen, W.-S. Lai, T. Xu, J. Kautz, and M.-H. Yang. Deep semantic face deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 67

[144] S. Bako, T. Vogels, B. McWilliams, M. Meyer, J. Novák, A. Harvill, P. Sen, T. DeRose, and F. Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Transactions on Graphics (TOG)*, 36(4), 2017. 67

[145] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 68

[146] L. Schneider, M. Jasch, B. Fröhlich, T. Weber, U. Franke, M. Pollefeys, and M. Rätsch. Multimodal neural networks: RGB-D for semantic segmentation and object detection. In P. Sharma and F. M. Bianchi, editors, *Image Analysis*, pages 98–109, Cham, 2017. Springer. doi:10.1007/978-3-319-59126-1_9. 69, 74

[147] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. arXiv:1606.00915. 69

[148] H. Chen, J. Gu, O. Gallo, M. Liu, A. Veeraraghavan, and J. Kautz. Reblur2deblur: Deblurring videos via self-supervised learning. In *IEEE International Conference on Image Processing (ICIP)*, pages 1–9, 2018. doi:10.1109/ICCPHOT.2018.8368468. 71

[149] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference On Medical Image Computing & Computer Assisted Intervention*, 2015. 71, 73

[150] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 72

[151] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017. 73

[152] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 73

[153] T. Meinhardt, M. Moeller, C. Hazirbas, and D. Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems.

In *IEEE International Conference on Computer Vision (ICCV)*, pages 1799–1808, October 2017. URL: https://github.com/tum-vision/learn_prox_ops, arXiv:1704.03488, doi:10.1109/ICCV.2017.198. 74, 75

[154] C. Hazirbas, J. Diebold, and D. Cremers. Optimizing the relevance-redundancy tradeoff for efficient semantic segmentation. In *Scale Space and Variational Methods in Computer Vision (SSVM)*, pages 243–255, Cham, 2015. Springer. URL: https://github.com/tum-vision/AFS, doi:10.1007/978-3-319-18461-6_20.

[155] J. Diebold, N. Demmel, C. Hazirbas, M. Moeller, and D. Cremers. Interactive multi-label segmentation of RGB-D images. In *Scale Space and Variational Methods in Computer Vision (SSVM)*, pages 294–306, Cham, 2015. Springer. URL: https://github.com/NikolausDemmel/tvseg, doi:10.1007/978-3-319-18461-6_24.

[156] F. Stark, C. Hazirbas, R. Triebel, and D. Cremers. CAPTCHA recognition with active deep learning. In T. Villmann and F.-M. Schleif, editors, *German Conference on Pattern Recognition Workshop on New Challenges in Neural Computation (GCPRW)*, Machine Learning Reports 03/2015, October 2015. URL: https://github.com/tum-vision/captcha_recognition.

[157] N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy, and T. Brox. What makes good synthetic training data for learning disparity and optical flow estimation? *Internatioanl Journal of Computer Vision (IJCV)*, 126(9):942–960, September 2018. URL: http://lmb.informatik.uni-freiburg.de/Publications/2018/MIFDB18, arXiv:1801.06397, doi:10.1007/s11263-018-1082-6.

Appendix A

Open-source Code & Datasets

Deep Depth From Focus

- https://hazirbas.com/projects/ddff/ Deep Depth From Focus source code and toolbox introduced in [21].

DDFF 12-Scene Dataset

- https://vision.cs.tum.edu/data/datasets/ddff12scene DDFF 12-Scene Benchmark introduced in [21].

FuseNet: Incorporating Depth into Semantic Segmentation

- https://github.com/tum-vision/fusenet CAFFE implementation of FuseNet introduced in [18].