Convolutional Neural Networks for Image and Video Demosaicing

by

Montgomery Lloyd Fischer

(Under the Direction of Alexander Petukhov)

Abstract

Color image demosaicing is an ill-posed inverse problem that arises in the formation of digital color images. By designing a demosaicing algorithm that operates on sequences of mosaiced video frames instead of isolated mosaiced images, one might hope to achieve a higher quality reconstruction. We propose two different deep convolutional networks for demosaicing that demonstrate the ability to effectively exploit temporal context frames in producing superior reconstructions as compared to single-frame networks. The first network explicitly registers frames using a computed optical flow; the second network adapts a recurrent back-projection architecture originally proposed for video super-resolution. Additionally, we show that single-frame demosaicing networks benefit from dense residual connections. Our contributions are supplemented with a review of the theory of proximal operators, image processing, neural networks, and demosaicing.

Index words: Demosaicing, convolutional neural networks, proximal algorithms, inverse problems, optical flow, super-resolution, video demosaicing.

Convolutional Neural Networks for Image and Video Demosaicing

by

Montgomery Lloyd Fischer

B.S. Mathematics, University of Georgia, 2020

B.S. Computer Science, University of Georgia, 2020

A Thesis Submitted to the Graduate Faculty of the

University of Georgia in Partial Fulfillment of the Requirements for the Degree.

Master of Arts

Athens, Georgia

2020

Convolutional Neural Networks for Image and Video Demosaicing

by

Montgomery Lloyd Fischer

Major Professor: Alexander Petukhov

Committee: Neil Lyall

Qing Zhang

Electronic Version Approved:

Ron Walcott

Interim Dean of the Graduate School

The University of Georgia

August 2020

Dedication

I dedicate this work to my parents, John and Teresa Fischer.

Acknowledgments

I would like to express my great thanks to Dr. Alexander Petukhov for being my thesis advisor. His guidance over the course of this project has been invaluable. At the time of our first meeting, I knew almost nothing about the active research area of image reconstruction with deep convolutional neural networks. Now, at the conclusion of this project and having learned a great deal, I am even more aware of the limitations of my knowledge. Dr. Petukhov has been a patient and gracious guide throughout this intellectual endeavor, and I am deeply grateful for his mentorship and helpful suggestions.

I thank Dr. Neil Lyall for serving on my thesis committee, reviewing this manuscript, and mentoring me during my four years of studying mathematics in college, above all. I have taken more classes with Neil than with any other professor during my time at the University of Georgia, and enjoyed his conversation on too many occasions to count. Under his skilled instruction, I developed a great appreciation of mathematical analysis for which I am profoundly grateful.

I thank Dr. Qing Zhang for serving on my thesis committee, reviewing this manuscript, and deepening my appreciation for applied mathematics. The first 8000-level graduate mathematics class I enrolled in during my degree was stochastic analysis with Dr. Zhang, and it was in this class that he introduced me to deep learning. Dr. Zhang's deep knowledge of probability and mathematical finance has been an inspiration to me, and I will remember fondly the many conversations about these subjects that we shared.

Aside from the members of my thesis committee, there are many professors in the mathematics department of the University of Georgia who deserve my hearty thanks. From the very first day of my

college orientation, Professor Mo Hendon has encouraged my efforts in mathematics through his wise academic advising, sharing departmental knowledge, and much enjoyable discussion (mathematical and otherwise). Dr. Giorgis Petridis, who along with Neil oversaw the first research project I undertook in my sophomore year, has been the model of a helpful, thoughtful mentor. Giorgis has always been willing to lend me his ear, and I am grateful for the advice and counsel he has given me. I would also like to thank Dr. Michael Usher, who, in teaching the first college mathematics course I took, helped me to realize that this field was exactly right for me. Additionally, I am grateful for the mentorship of Dr. Marcel Blais and Dr. Stephan Sturm during a summer research project in mathematical finance at Worcester Polytechnic Institute.

I would not have had the opportunity to attend the University of Georgia without the Foundation Fellowship. I would especially like to thank Jessica Hunt for her wise and encouraging help over the past four years. I am not the first, nor will I be the last, to celebrate her dedication to helping students pursue their goals.

My friends and loved ones have been a ceaseless source of encouragement and support during my studies and, in particular, during my work on this thesis. I am so grateful to my parents, John and Teresa Fischer, for their unending love and support for the 21 years of my life so far. I am also grateful for my brother, Vic, whose engineering and musical talents never fail to inspire and encourage his big brother. Johanna Hoover deserves a mountain of thanks for her constant and loving support. I do not want to think of how difficult it would have been to persevere on this project without her. My good friends Jacob Sparks, Nico Leis, Joseph Campbell, Matt Hamil, Sebastian Puerta, and Rev. Matt Siple have made these past four years much better than I could have possibly imagined beforehand. There are undoubtedly many, many people not mentioned here who should have been – my thanks to them all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

So the natural light makes it clear to me that my ideas are like pictures or images that can easily fall short of the perfection of the things from which they are taken, but which can't exceed it.

— Rene Descartes, *Meditations on First Philosophy* [13].

Color image demosaicing is an ill-posed inverse problem that arises in the formation of digital color images. A patterned filter on top of the sensor grid enables a camera to differentiate color, but the measurements thereby obtained are incomplete. The process of generating a complete, viewable image from the incomplete, mosaiced measurements is called *demosaicing*. Recent efforts in the development of demosaicing algorithms have employed convolutional neural networks to great success. In this work, we consider an extension of the single-image demosaicing problem to video or film scanning applications. By designing a demosaicing algorithm that operates on sequences of mosaiced video frames instead of isolated mosaiced images, one might hope to achieve a higher quality reconstruction. We propose two different deep convolutional networks for demosaicing that demonstrate the ability to effectively exploit temporal context frames in producing superior reconstructions compared to single-frame networks of similar size.

In Chapters 2 – 5, we discuss relevant material from the theory of proximal operators, image processing, neural networks, and demosaicing. Chapter 2 covers inverse problems, an important framework for many problems in computational imaging, including demosaicing. Chapter 3 reviews material from signal processing and digital imaging relevant to our present efforts. Chapter 4 discusses deep learning, the core component of our proposed methods. The connection between

convolutional neural networks and convolutional sparse coding, as established in [48], is also presented. Chapter 5 reviews the demosaicing problem, solutions from the literature, and networks for super-resolution and optical flow estimation that are pertinent to our work.

In Chapter 6, we present the results of our numerical experiments with convolutional networks for the image and video demosaicing problems. We define several architectures for spatial and temporal demosaicing along with training details and accuracy as evaluated on several demosaicing test datasets. We evaluate identity-bypass networks, dense residual connection networks, and deep back-projection networks for the image demosaicing problem. For the video demosaicing problem, we propose and evaluate MFD-Net, a network that registers several frames using an optical flow and exploits the temporal information therein. We also adapt a recurrent back-projection network architecture originally proposed in the super-resolution literature for the video demosaicing problem. We include both quantitative and qualitative comparison of demosaicing networks. Our findings are threefold: First, dense residual connections are advantageous to single image demosaicing networks. Second, MFD-Net suffers from a learning problem but demonstrates promising performance over single image methods. Third, deep recurrent back-projection networks effectively and efficiently exploit temporal context for demosaicing video sequences. Finally, future research directions are proposed.

Chapter 2

Inverse problems

We introduce the theory of inverse problems. Many important applications of mathematics rely on the inverse problem framework, including our present topic of digital image acquisition. Inverse problems are often ill-posed, in which case regularization methods are considered. Inverse problems rarely admit closed-form solutions and there are many proposed methods to obtain approximate solutions. In particular, we review the Bayesian maximum a posteriori method. We consider the class of proximal algorithms for minimization, and show how the majorization-minimization framework for optimization can be seen as an instance of the proximal gradient method.

Section 2.1

What is an inverse problem?

A physical theory enables us to predict the outcome of measurements of a physical system using a model of that system. The model encapsulates our understanding of *causes*, from which observable *effects* may be predicted. In order for the predictions to be reliable, the physical theory underlying the model must be accurate, and the parameters used in constructing the model must be well-tuned. Once we have a tuned model, the process of predicting outcomes is known as the *forward problem*.

The *inverse problem*, by contrast, is to infer the set of model parameters from the observed data. Inverse problems are widespread in applied mathematics and the sciences. Many problems in medical and commercial imaging and image recovery can be analyzed as inverse problems: X-ray

computed tomography, MRI, digital camera image acquisition, image inpainting, and more. As an example, consider the problem of recovering an image that has been blurred with a convolutional filter[1]. In this case, the measurement operator M is a convolution with the blurring filter, and we seek a way to invert the convolution to acquire the original, unblurred image x from the blurry variant y.

In most situations observations are noisy, making the forward problem nondeterministic. In many situations, multiple sets of model parameters could feasibly have generated the same observed data, making the solution of the inverse problem non-unique. Techniques for the solution of inverse problems must be robust to these difficulties.

Section 2.2

Mathematical characterization

Let x denote the parameters of a model, y denote the obtained measurements, and M denote the *measurement operator* of the forward problem that maps the internal parameters of the model to the observed measurements. We assume that $x \in X, y \in Y, M : X \to Y$ where X, Y are (e.g.) Hilbert spaces and M is a mapping between them. A no-frills inverse problem is therefore reconstructing x from y, where

$$y = Mx$$

models the system. The choice of M and X encapsulate all that is known about the system under observation; the choice of Y is made in accordance with the measured data.

In practice, the measurements from a physical system are noisy. To incorporate noise into the inverse problem, define the noise term $n := y - Mx$. The noise term captures whatever error arises

[1] This is the *deconvolution* problem

from either the measurement process itself, or a misspecification of the forward operator M. The inverse problem with noise is then

$$y = Mx + n. \tag{2.1}$$

It is common to model n as a random variable with some specified distribution, referred to as the *noise model* of the inverse problem. If n could be modeled deterministically, then it could be incorporated into M.

If M is injective, that is,

$$Mx_1 = Mx_2 \implies x_1 = x_2 \quad \forall x_1, x_2 \in X$$

then an inverse operator

$$M^{-1} : Im(M) \subset Y \to X$$

is in principle well-defined. However, if the inversion of the forward operator amplifies noise in the measurements to an unacceptable degree, or if the forward operator is not injective, then the inverse problem may be considered *ill-posed*. There is not a universal definition of ill-posedness for inverse problems as the degree to which noise amplification is tolerated in the inverse operator is domain-specific.

For a more complete discussion of inverse problems, including solution methods and regularization techniques, refer to [63, 28, 61, 3].

Section 2.3

Methods for solution

Rarely do inverse problems of interest admit closed-form solutions. Numerical algorithms are thus a widely popular class of solution methods. It is often useful to incorporate into these methods prior information about the nature of the parameter space X. The incorporation of prior information into a numerical method for solution is called *regularization*.

Tikhonov regularization (or L_2 regularization), [63, 28], seeks a regularized solution to the inverse problem by

$$x^* = \arg\min_x \|Mx - y\|_2^2 + \alpha\|x\|_2^2 \tag{2.2}$$

where $\alpha > 0$ tunes the regularization.[2] Other terms, such as total variation or a sparsity-inducing transformation of the reconstructed solution, can be used to promote certain types of reconstructions.

Another common way of accomplishing regularization is with a Bayesian framework. Here, two probability distributions are needed: the first is the likelihood function $p(y|x)$ expressing the conditional probability of observing y when the "true" parameters are x. As the measurement operator M is deterministic, knowledge of the probability density of the noise term n is sufficient to characterize the likelihood function since for every $x \in X$,

$$p(y|x) = p(Mx + n|x) = p_n(n)$$

where p_n is the probability density of the noise term. The second distribution we need is a density function $p(x)$ encoding the prior information known about the parameter space X. Given these

[2] Tikhonov himself [63] proposed a class of regularizing operator for ill-posed inverse problems on function spaces with uniqueness and stability guarantees that is similar to, but not identical with, the parametrized L_2 regularization which now bears his name.

two distributions, we apply Bayes' theorem to compute the posterior distribution

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}.$$

A solution to the inverse problem may then be obtained by means of *maximum likelihood estimation* on the posterior distribution. Our solution x^* to the inverse problem $y = Mx + n$ is then computed as follows:

$$\begin{aligned} x^* &= \arg\max_{x \in X} p(x|y) \\ &= \arg\max_{x \in X} [\log p(x|y)] \\ &= \arg\min_{x \in X} [-\log p(y|x) - \log p(x)] \end{aligned}$$

Such a Bayesian approach is not computationally trivial. The optimization problem requires repeated sampling of the posterior function, which is potentially very complicated. Introducing logarithms, as above, separates the posterior its constituent likelihood and prior distributions and is a first step towards tractable minimization.

In domains like image processing, the prior $p(x)$ is a probability distribution on the set of possible images. Representing an image requires many parameters, and defining a realistic prior distribution on the set of all possible images is a complex task. Indeed, a totally satisfactory prior would incorporate a large amount of statistical information about the structure of images, and it may be very difficult to compute with.

Section 2.4

Proximal algorithms for minimization

The theory of proximal operators has been brought to bear on the solution of inverse problems, with some substantial successes [12]. With this in mind, we examine the essential elements of proximal operators. For a more detailed reference survey, we refer the reader to [49].

Proximal algorithms are tools for the minimization of (potentially non-smooth) functions. The basic element of a proximal algorithm is the *proximal operator* of a function $f : \mathbb{R}^n \to \mathbb{R} \cup \infty$. We require f to be convex, i.e.

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) \quad \forall x_1, x_2 \in \mathbb{R}^n, \lambda \in [0,1]$$

and everywhere subdifferentiable, that is,

$$\partial f(x) \neq \emptyset \quad \forall x \in \mathbb{R}^n$$

where $\partial f(x)$ denotes the *subgradient set* of f at x,

$$\partial f(x) := \{ g \mid \forall y \ f(y) \geq f(x) + g \cdot (y - x) \quad \}.$$

The proximal operator of f with parameter $\lambda > 0$ is $\text{prox}_{\lambda f} : \mathbb{R}^n \to \mathbb{R}^n$ given by

$$\underset{\lambda f}{\text{prox}}(v) = \underset{x \in \mathbb{R}^n}{\arg\min} \left[f(x) + \frac{1}{2\lambda} \|x - v\|_2^2 \right].$$

The parameter λ controls how close $\text{prox}_{\lambda f}(v)$ is to v compared to the minimizer x^* of f.

It is worth mentioning that regardless whether f is strictly convex or not, $\text{prox}_{\lambda f}$ is well-defined since $f(x)+\frac{1}{2\lambda}\|x-v\|$ is a strictly convex function of x. We also observe that the proximal operator generalizes the notion of projection. For a closed convex set C, consider the indicator function

$$f(x) = \begin{cases} 0 & x \notin C \\ \\ \infty & x \in C \end{cases}$$

and note that $\text{prox}_f(v)$ gives the standard orthogonal projection of v onto C,

$$\arg\min_{x \in C} \|x - v\|_2^2.$$

In general, the fixed points of a proximal operator are precisely the minimizers of f.

Theorem 1. *x^* minimizes f if and only if $x^* = \text{prox}_f(x^*)$.*

Proof. ($\Rightarrow$) Suppose that x^* minimizes f. Then for any $z \in \mathbb{R}^n$, we have that

$$\begin{aligned} f(z) + \frac{1}{2}\|z - x^*\|_2^2 &\geq f(x^*) + \frac{1}{2}\|z - x^*\|_2^2 \\ &\geq f(x^*) + \frac{1}{2}\|x^* - x^*\|_2^2 \\ &= f(x^*) \end{aligned}$$

so $x^* = \text{prox}_f$.

($\Leftarrow$) Suppose that $x^* = \text{prox}_f(x^*)$. Let $Q(y) = f(y) + \frac{1}{2}\|x^* - y\|_2^2$, so that x^* minimizes Q by definition of the proximal operator. We use the following facts about the subgradient operator: first, that if h_1, h_2 are subdifferentiable functions, then $\partial(h_1 + h_2)(x) = \partial h_1(x) + \partial h_2(x)$ where $+$ denotes set addition; second, that x minimizes a subdifferentiable function h if and only if $0 \in \partial h(x)$; third

that if h is differentiable at x with gradient $\nabla h(x)$, then $\partial h(x) = \{\nabla h(x)\}$. All three properties follow easily from the definition of the subgradient operator.

Thus

$$\begin{aligned} 0 &\in \partial Q(x^*) \\ &= \partial f(x^*) + \{\|x^* - x^*\|_1\} \\ &= \partial f(x^*) + \{0\} = \partial f(x^*). \end{aligned}$$

which concludes the proof, by the second property of the subgradient. □

The proximal operator may be interpreted as a gradient step. We introduce the Moreau envelope for convex subdifferentiable f as follows:

$$M_{\lambda f}(v) = \inf_{x \in \mathbb{R}^n} \left[f(x) + \frac{1}{2\lambda} \|v - x\|_2^2 \right]$$

so that

$$M_{\lambda f}(v) = f(\underset{\lambda f}{\text{prox}}(v)) + \frac{1}{2\lambda} \|v - \underset{\lambda f}{\text{prox}}(v)\|_2^2$$

for all v in the domain of f. Then by the envelope theorem,

$$\nabla M_{\lambda f}(x) = \frac{1}{\lambda}(x - \underset{\lambda f}{\text{prox}}(x))$$

or equivalently,

$$\underset{\lambda f}{\text{prox}}(x) = x - \lambda \nabla M_{\lambda f}(x).$$

Thus one application of the proximal operator is equivalent to a gradient descent step of size λ towards minimizing $M_{\lambda f}$. This is also a gradient step towards minimizing f.

Theorem 2. *For all $\lambda > 0$, f and $M_{\lambda f}$ have identical minimizers.*

Proof. Observe that

$$\begin{aligned}
\inf_v M_{\lambda f}(v) &= \inf_v \inf_x \left[f(x) + \frac{1}{2\lambda}\|v - x\|_2^2 \right] \\
&= \inf_x \inf_v \left[f(x) + \frac{1}{2\lambda}\|v - x\|_2^2 \right] \\
&= \inf_x f(x).
\end{aligned}$$

□

This suggests the following algorithm for minimization of f, known as *proximal minimization*. We iterate towards a minimizer of f by the rule

$$x^{(k+1)} = \operatorname*{prox}_{\lambda_k f}(x^{(k)})$$

where the (positive) step sizes $\{\lambda_k\}$ are subject to $\sum_{i=0}^{\infty} \lambda_k = \infty$. This algorithm is somewhat less than practical. In order to be preferred to simple gradient descent on f, it would need to be easier to compute the proximal operator by minimizing $f(x) + \frac{1}{2\lambda}\|x - v\|_2^2$ than taking the gradient of f.

The *proximal gradient method*, on the other hand, offers a more practical approach. If our objective function is the sum of differentiable f and not necessarily differentiable g, then the proximal gradient method algorithm iterates as follows:

$$x^{(k+1)} = \operatorname*{prox}_{\lambda_k g}(x^{(k)} - \lambda_k \nabla f(x^{(k)})) \tag{2.3}$$

We will see how this scheme can be seen as a special case of majorization-minimization optimization.

Section 2.5

Majorization-minimization for linear inverse problems

We consider the connection between majorization-minimization and proximal gradient descent noted in [5]. Suppose we have an inverse problem of type 2.1 where M is a linear operator and X is a subset of $\mathbb{R}^N$. Further, we require that $M^T M$ is positive-semidefinite with largest eigenvalue at most 1. One paradigm for solving such a problem is *regularized least squares (RLS)*. We introduce a regularization penalty function $\phi : X \to \mathbb{R}$ and seek a solution by

$$x^* = \arg\min_{x \in X} \|y - Mx\|_2^2 + \lambda\phi(x)$$

with λ controlling the relative strength of the regularization. Separate data-fidelity and regularization by $f(x) = \|y - Mx\|_2^2$ and $g(x) = \lambda\phi(x)$, with $F(x) = f(x) + g(x)$.

A majorization-minimization (MM) approach to minimizing a function $F(x)$ operates by iteratively minimizing a substitute *majorizer* for F. A majorizer $\hat{F} : X \times X \to \mathbb{R}$ must satisfy

$$\hat{F}(x, t) \geq F(x) \text{ for all } x, t \in X, \tag{2.4}$$

$$\hat{F}(x, x) = F(x) \text{ for all } x \in X. \tag{2.5}$$

The choice of a majorizer is otherwise free. The algorithm updates by minimizing the majorizer $F(x, x^{(k)})$ with respect to x:

$$x^{(k+1)} = \arg\min_{x \in X} \hat{F}(x, x^{(k)}).$$

Let us consider an approach to solving the RLS linear inverse problem described above using MM. We note that for all $x, t \in X$,

$$\begin{aligned}
f(x) &= f(t) + f(x) - f(t) \\
&= f(t) + \|y - Mx\|_2^2 - \|y - Mt\|_2^2 \\
&= f(t) + (\|y\|_2^2 - 2\langle Mx, y\rangle + \|Mx\|_2^2) - (\|y\|_2^2 - 2\langle Mt, y\rangle + \|Mt\|_2^2) \\
&= f(t) + 2\langle M(x-t), -y\rangle + \|Mx\|_2^2 - \|Mt\|_2^2 \\
&= f(t) + 2\langle M(x-t), -y\rangle + \|M(x-t)\|_2^2 - 2\langle Mt, Mt\rangle + 2\langle Mx, Mt\rangle \\
&= f(t) + 2\langle M(x-t), Mt - y\rangle + \|M(x-t)\|_2^2 \\
&= f(t) + 2\langle M(x-t), Mt - y\rangle + (x-t)^T M^T M(x-t) \\
&\leq f(t) + 2\langle M(x-t), Mt - y\rangle + \|x-t\|_2^2
\end{aligned}$$

where the inequality is obtained by replacing $M^T M$ with I since $I \geq M^T M$ by assumption. We then construct a majorizer $\hat{F}$ by selectively majorizing f:

$$\hat{F}(x,t) = g(x) + f(t) + 2\langle M(x-t), Mt - y\rangle + \|x-t\|_2^2.$$

This calculation demonstrates that condition (2.4) is satisfied. It is trivial to verify that (2.5) is satisfied by evaluating $\hat{F}(x,x)$.

Observe that

$$\begin{aligned}
\|x - (t - M^T(Mt - y))\|_2^2 &= \|x\|_2^2 - 2\langle x, t\rangle + 2\langle Mx, Mt\rangle \\
&\quad - 2\langle Mx, y\rangle + R_1(t,y) \\
&= \|x-t\|_2^2 + 2\langle M(x-t), Mt - y\rangle + R_2(t,y)
\end{aligned}$$

$$= \hat{F}(x,t) - g(x) + R_3(t,y)$$

and

$$\nabla f(x) = \nabla(\|y - Mx\|_2^2) = 2M^T(Mx - y).$$

We conclude that the MM algorithm is equivalent to the proximal gradient method (2.3) with $\lambda_k = \frac{1}{2}$, as

$$\begin{aligned} x^{(k+1)} &= \arg\min_{x \in X} \hat{F}(x, x^{(k)}) \\ &= \arg\min_{x \in X} \left[g(x) + \|x - (x^{(k)} - M^T(x^{(k)} - y))\|_2^2 \right] \\ &= \arg\min_{x \in X} \left[g(x) + \|x - (x^{(k)} - \tfrac{1}{2}\nabla f(x^{(k)}))\|_2^2 \right] \\ &= \operatorname{prox}_{\frac{1}{2}g} \left(x^{(k)} - \tfrac{1}{2}\nabla f(x^{(k)}) \right) \end{aligned}$$

demonstrates.

Chapter 3

Image processing

We introduce concepts from signal processing used in the representation and manipulation of digital images. The sampling theorem, giving the conditions under which a bandlimited continuous signal may be perfectly recovered from discrete samples, frames the study. Image resampling is discussed, as well as interpolation kernels for situations when the sampling theorem does not hold. In particular, we use polynomial splines as a class of especially popular interpolation kernels for image resampling. We also review optical flow, a scheme for representing motion between frames in digital videos.

Section 3.1

Sampling and perfect recovery

Some of the essential theory of signal processing will be used throughout. We give the basics here; a more thorough presentation can be found in [42].

Consider the case of a one-dimensional continuous-time signal, represented by the function $f : \mathbb{R} \to \mathbb{R}$. In order to store and manipulate arbitrary signals using a computer, the signal must be sampled at discrete points. Suppose that we are able to sample f with a uniform sampling distance T; that is, we can observe the data $\{f(nT)\}_{n\in\mathbb{Z}}$ (we assume an infinite sampling grid $D = T\mathbb{Z}$). The problem of reconstructing f from its samples is general – demosaicing is merely one example of a type of this problem.

An important classical result in signal processing gives conditions under which f may be perfectly recovered from its samples on D by convolution with the sinc kernel h_T. We represent the samples as a function $f_D : \mathbb{R} \to \mathbb{R}$ with

$$f_D(x) = \sum_{n \in D} f(nT)\delta(x - nT).$$

Lemma 1. *With f and f_D as above, we have*

$$\widehat{f_D}(\xi) = \frac{1}{T} \sum_{n \in \mathbb{Z}} \hat{f}\left(\xi - \frac{2\pi n}{T}\right)$$

Proof. We write f_D as the product of f with a Dirac comb,

$$f_D(x) = f(x) \sum_{n \in \mathbb{Z}} \delta(x - nT).$$

Taking a Fourier transform and applying the convolution formula,

$$\begin{aligned}
\hat{f_D}(\xi) &= \frac{1}{2\pi} \hat{f} * \left(\int_{\mathbb{R}} \sum_{n \in \mathbb{Z}} \delta(x - nT) e^{-ix\cdot} dx \right)(\xi) \\
&= \frac{1}{2\pi} \hat{f} * \left(\sum_{n \in \mathbb{Z}} e^{-inT\cdot} \right)(\xi) \\
&= \frac{1}{2\pi} \hat{f} * \left(\frac{2\pi}{T} \sum_{n \in \mathbb{Z}} \delta\left(\cdot - \frac{2\pi n}{T}\right) \right)(\xi) \\
&= \frac{1}{T} \sum_{n \in \mathbb{Z}} \hat{f}\left(\xi - \frac{2\pi n}{T}\right)
\end{aligned}$$

where the second-to-last equality follows from the Poisson summation formula. □

Theorem 3 (Shannon-Whittaker)**.** *Suppose $f : \mathbb{R} \to \mathbb{R}$ and supp $\hat{f} \subset [-\pi/T, \pi/T]$. Then*

$$f(x) = (f_D * h_T)(x) = \sum_{n\in\mathbb{Z}} f(nT)h_T(x - nT)$$

where

$$h_T(x) = \frac{\sin(\pi x/T)}{\pi x/T}.$$

Proof. We would like to take the inverse Fourier transform of the bandlimit function $\chi_{[-\pi/T,\pi/T]}$ as follows:

$$\begin{aligned}
\frac{1}{2\pi}\int_{\mathbb{R}} \chi_{[-\pi/T,\pi/T]}(\xi)e^{i\xi x}d\xi &= \frac{1}{2\pi}\int_{-\pi/T}^{\pi/T} e^{i\xi x}d\xi \\
&= \frac{1}{2\pi ix}(e^{ix\pi/T} - e^{-ix\pi/T}) \\
&= \sin(\pi x/T)/(\pi x/T) \\
&=: h_T(x).
\end{aligned}$$

Although $h_T(x)$ is not an L_1 function, the inversion can be made rigorous by extending the Fourier transform to L_2 functions. Thus we have that $\widehat{h_T}(\xi) = T\chi_{[-\pi/T,\pi/T]}(\xi)$. With Lemma 1 and the convolution theorem, we have

$$\widehat{h_T * f_D}(\xi) = \chi_{[-\pi/T,\pi/T]}(\xi)\sum_{k\in\mathbb{Z}} \hat{f}(\xi - 2\pi k/T) = \hat{f}(\xi).$$

Applying the inverse Fourier transform, we are finished. □

Hence band-limited signals (those with bounded support in the frequency domain) may be, in principle, perfectly recovered from countably many samples provided those samples are close enough together, i.e. T is chosen small enough. In practice, we are not so fortunate as to have T as small

as necessary. This produces the phenomenon of *aliasing*: If the signal f has support outside of $[-\pi/T, \pi/T]$, then naively applying the Shannon-Whittaker recovery convolution produces

$$\begin{aligned}\widehat{f_D * h_T}(\xi) &= T\chi_{[-\pi/T,\pi/T]}(\xi)\widehat{f_D}(\xi) \\ &= \chi_{[-\pi/T,\pi/T]}(\xi)\sum_{k\in\mathbb{Z}}\hat{f}(\xi - 2\pi k/T).\end{aligned}$$

For any ξ, all frequencies of $\hat{f}$ of the form $\hat{f}(\xi - 2\pi n/T)$ will overlap in the summation, ruining the reconstruction $\hat{f_D} * h_T$ at ξ. In other words, the entire frequency bandwidth of $\hat{f}$ is packed into the $[-\pi/T, \pi/T]$ frequency band of the reconstructed $f_D * h_T$, producing a function which *a priori* may produce completely incorrect frequencies.

Section 3.2

Resampling

Suppose we have a sampled image on the integer lattice

$$f_D(x) = \sum_{n\in\mathbb{Z}^2} f(n)\delta(x - n)$$

and we wish to apply a spatial transformation to the image. This is very common situation in computer graphics, as images often need to be transformed by resizing, rotating, flipping, translating, or something else entirely. A fuller treatment can be found in [67]. Here we will do no more than specify the transformation by an operator $W : \mathbb{R}^2 \to \mathbb{R}^2$ relating coordinates in the original image x to coordinates in the transformed, or *warped*, image, x'; thus $Wx = x'$. We wish to obtain a

warped image

$$g_{D'}(x') = \sum_{n \in \mathbb{Z}^2} g(n)\delta(x' - n).$$

We do so by the following procedure:

1. Interpolate a continuous $f(x)$ from $f_D(x)$

2. Define a *resampling grid* D' in terms of the original coordinate space

3. Sample $f(x)$ on D' to obtain $g_{D'}(x')$

For an interpolation kernel h and antialiasing kernel k, we have

$$\begin{aligned} f(x) &= (f_D * h)(x) \\ \tilde{g}(x') &= f(W^{-1}(x')) = f(x) \\ g(x') &= (g * k)(x') \\ g_{D'}(x') &= \sum_{n \in \mathbb{Z}^2} g(n)\delta(x' - n). \end{aligned}$$

Our choice of h is made to provide a suitable recovery of the continuous signal f from the discrete samples f_D. The choice of k is made to prevent aliasing when sampling $g_{D'}$ by bandlimiting $g_{D'}$ to the range determined by the sampling rate in accordance with Theorem 3.

We can combine the steps to obtain a direct relationship between $g_{D'}$ and f_D. For $x' \in D'$, we have

$$\begin{aligned} g_{D'}(x') &= g(x') \\ &= (\tilde{g} * k)(x') \\ &= \int_{\mathbb{R}^2} \tilde{g}(t)k(x' - t)dt \end{aligned}$$

$$
\begin{aligned}
&= \int_{\mathbb{R}^2} f(W^{-1}(t))k(x'-t)dt \\
&= \int_{\mathbb{R}^2} (f_D * h)(W^{-1}(t))k(x'-t)dt \\
&= \int_{\mathbb{R}^2} \left[\sum_{n\in\mathbb{Z}^2} f_D(n)h(W^{-1}(t)-n) \right] k(x'-t)dt \\
&= \sum_{n\in\mathbb{Z}^2} f_D(n) \underbrace{\int_{\mathbb{R}^2} h(W^{-1}(t)-n)k(x'-t)dt}_{:=\rho(x',n)} \qquad (3.1) \\
&= \sum_{n\in\mathbb{Z}^2} f_D(n)\rho(x',n), \qquad (3.2)
\end{aligned}
$$

the term $\rho(x', n)$ being the *resampling kernel* that provides, for each pixel of the resampled image $g_{D'}$, the coefficients of the linear combination of pixel values in f_D that calculate the resampled pixel. Note that in the above formulation, the inverse warp W^{-1} is used to compute the resampling kernel with an integral in the warped space. In the case that W is differentiable, we may make the change of variables $t = W(u)$ so $dt = \det D(W)(u)du$ and

$$
\rho(x', n) = \int_{\mathbb{R}^2} h(u-n)k(x'-W(u)) \det D(W)(u)du
$$

so that the resampling kernel may be evaluated in the pre-warp space [67]. Presently, we are interested in image resampling as it pertains to image registration with optical flow, wherein the inverse warp W^{-1} is known and W itself is not necessary to compute; hence we compute the rasampling kernel with (3.2). See Section 3.5 for full details.

Section 3.3

Interpolation

We now discuss the selection of an interpolating kernel h to recover the continuous signal $f(x)$ from the samples $f_D(x)$ by

$$\begin{aligned} f(x) &= (f_D * h)(x) \\ &= \left(\sum_{n \in \mathbb{Z}} f(n)\delta(x-n) \right) * h(x) \\ &= \sum_{n \in \mathbb{Z}} f(n)(\delta_n * h)(x) \\ &= \sum_{n \in \mathbb{Z}} f(n)h(x-n) \end{aligned}$$

from which we see that an interpolation kernel h may be represented as a set of coefficients on the grid D for combining the discrete samples of f_D to obtain an interpolated value $f(x)$. For simplicity, we will consider the one-dimensional case. There are several ways of treating multidimensional kernels, but the simplest is to consider separable kernels: $H(x,y) = H_1(x)H_2(y)$, for which the one-dimensional theory is sufficient.

By Theorem 3, a bandlimited signal f may be perfectly recovered from its samples by convolution with the sinc function in the spatial domain. In practice, most signals sampled at a rate T are have support in the frequency domain exceeding the bandlimit $[-\pi/T, \pi/T]$ required for perfect interpolation with sinc. Sinc interpolation on a sample from this large class of signals would produce a bandlimited interpolant which is guaranteed *not* to belong to the original class. There are yet further difficulties: sinc has infinite support in the spatial domain, and has very slow decay: $h_T(t) = O(1/t)$. To resolve these issues, a variety of *apodization* techniques have been proposed

whereby sinc is truncated by multiplication with a finite support window [62]. Other means of interpolating continuous signals from discrete samples not based on a truncation of the sinc filter have also been proposed.

Section 3.4

Polynomial splines for interpolation

An extensive literature exists on polynomial splines for interpolation and approximation tasks (see the surveys [55, 35, 9, 21]). For our present purposes, we will use a very limited portion of this theory to explain common alternatives to apodized filters for interpolation.

In brief, polynomial splines are piecewise polynomial functions connected at a discrete set of *knots*, with continuity requirements at these knots. The one-dimensional polynomial splines of order l on the knot set X have

$$s \in C_{l-1},$$

$$s \in P_l([x_i, x_{i+1}), \quad x \in [x_i, x_{i+1}) \quad \forall i \in I.$$

The knot set is used to denote those points at which we have available data samples. For digital images, our samples form a regularly-spaced grid. This restriction on the knot set simplifies our discussion.

A special class of polynomial splines have especially nice properties for computation and interpolation. These are the *cardinal B-splines*. A suitable description of the cardinal B-splines $\{B_k\}_{k=0}^{\infty}$

for our present purposes is given recursively by

$$B_0(x) = \chi_{[-\frac{1}{2},-\frac{1}{2})}(x)$$

$$B_k(x) = (B_0 * B_{k-1})(x), \quad k > 0.$$

It is not difficult to show that the cardinal B-splines are in fact polynomial splines.

We note that the corresponding Fourier transforms of the cardinal B-splines are powers of the sinc function.

$$\begin{aligned}
\hat{B}_0(\xi) &= \int_{-\infty}^{\infty} \chi_{[-\frac{1}{2},-\frac{1}{2})}(x) e^{-2\pi i \xi x} dx \\
&= \int_{-\frac{1}{2}}^{\frac{1}{2}} e^{-2\pi i \xi x} dx \\
&= \frac{e^{\pi i \xi} - e^{\pi i \xi}}{2\pi i \xi} \\
&= \frac{\sin \pi \xi}{\pi \xi} \\
&= \operatorname{sinc}(\xi)
\end{aligned}$$

so by the convolution theorem,

$$\hat{B}_k(\xi) = (\operatorname{sinc}(\xi))^{k+1}$$

thus a continuous function obtained from cardinal B-spline interpolation will not, in general, be bandlimited.

Interpolating by convolution with B_0 is known as *nearest-neighbor interpolation*. Although extremely computationally simple, a function reconstructed with nearest-neighbor interpolation is discontinuous and likely to create severe "pixelated" artifacts in the resampled image. Interpolation with kernels B_1, B_2, and B_3 is called *linear, quadratic*, and *cubic* (spline) interpolation respectively.

The two-dimensional separable analogues are *bilinear, biquadratic,* and *bicubic* interpolation respectively[1]. Of these, bilinear and bicubic interpolation are widely used in applications as a workable compromise between efficiency and accuracy of interpolations. It is particularly useful that fast GPU implementations of bilinear interpolation exist and are easily used through standard deep learning libraries.

Section 3.5

Optical flow

Estimating the motion ocurring in a digital sequence of frames is important in computer vision, with a diverse range of applications from video compression to autonomous driving. Of particular relevance here is the use of optical flow for the task of image registration.

What constitutes motion? Humans can easily identify types of motion ocurring in videos natural scenes – a person walking, the movement of the camera, a spinning top, and innumerable others. Humans observe and understand this motion without deliberately scrutinizing every pixel of every frame in order to determine which individual pixels moved where. Computers do not come equipped with a robust visual system and must use the latter approach, known as *optical flow*, to keep track of motion in video sequences.

We define optical flow as follows. Given two subsequent frames of a video sequence $F^1, F^2 : \mathbb{R}^2 \to [0, 1]$, the forward optical flow from F^1 to F^2 is the operator $\nu : \mathbb{R}^2 \to \mathbb{R}^2$ such that $\nu(x)$ gives the displacement vector from F^1 to F^2, i.e.

$$F^2(x) = F^1(x + \nu(x)). \tag{3.3}$$

[1]In the literature, bicubic interpolation may refer to one of several cubic spline interpolation methods. We refer the interested reader to [67] for a more thorough discussion.

Equation (3.3) can be easily applied to images sampled on the discrete grid using the framework of section 3.2 with $W^{-1} = (I + \nu)$, and F^1, F^2 the interpolated versions of discrete sampled frames F^1_D, F^2_D. Optical flow can then be used to define a resampling grid D' such that sampling the interpolated F^1 on D' gives another perspective on the scene represented by F^2_D computed from the pixels of F^1_D by the resampling kernel. In general, neither the inverse optical flow ν^{-1} nor the Jacobian $D(\nu)$ are available without additional computational cost, so the resampling kernel is computed using (3.2). This application of optical flow is known as *image registration*, and is of use when we have captured multiple noisy or corrupt images of some scene with some kind of motion between frames as in burst photography or video sequences.

Stated in this way, optical flow is ill-posed or undefined for essentially all realistic video sequences. Indeed, the definition operates on the unrealistic assumptions that the video sequence experiences no change of lighting between frames, that there is no occlusion of objects in the sequence, and that the intensity value of a pixel corresponds exactly to the motion of an object across frames. Some approaches in the literature [57] use real-valued flow in order to account for sub-pixel motion, resampling with a bilinear interpolation. Other approaches include with each flow an estimated pixel-wise confidence, or probability that the computed flow is correct, at an additonal computational cost [70, 65]. We discuss PWC-Net, a deep neural network for computing optical flow, in Chapter 5.

Chapter 4

Deep convolutional neural networks

We introduce the class of popular computational models for data-driven function approximation tasks called neural networks. Neural networks are parameterized models consisting of successive applications of affine transformations and nonlinearities. The parameters specific transformations used are learned from data by minimizing an objective function consisting of a loss term and a regularization term over a training dataset. The objective functions of neural networks are not, in general, convex, yet techniques from convex optimization are widely used to train neural networks. We discuss the convolutional network architecture, which has been employed to great effect in signal processing and image classification tasks. We review some of the mathematical literature on neural networks, including the convolutional sparse coding framework for convolutional networks.

Section 4.1

Neural networks: overview

Neural networks are computational models that have lately achieved great success in a diversity of technical areas, especially computer vision and natural language processing [38]. This success is owed to a combination of advances in network design, the availability of large suitable datasets, specialized hardware, and widespread scientific attention. Neural networks consist of multiple layers, each of which computes features of the input data from the features of the previous layer using an affine transformation followed by a nonlinearity. Networks with many layers (called *deep*) have

the capacity to represent quite complicated relationships and features of data. In particular, by restricting the linear portion of the affine transformations of each layer to be a convolution, the resulting model is a *convolutional neural network* or CNN, introduced in [37, 36]. CNNs have been incredibly successful empirically in achieving state-of-the-art results in image classification [34] and many other computer imaging tasks (see [38] for a survey).

The use of deep neural networks for image classification gives an example: in order to accurately compute the probability that an image is (say) a dog rather than a cat, the layers of a deep neural network must compute very complicated features of a high-dimensional vector representing an image. The first layer might compute features that detect edges in the image; the second layer might detect simple shapes from the edges detected by the first layer; subsequent layers compute increasingly intricate features. Ultimately, a classifier uses the highest-level features to compute the probabilities that the image belongs to either class.

Deep learning refers to a general set of methods to learn *from data* the best features to compute. Instead of hand-selecting which features are useful, a deep learning approach will approximate or *learn* an optimal set of features with respect to some predetermined prediction or classification task, given a sufficiently large representative dataset. This is known as *training*. For the above example, a deep learning approach takes a dataset of properly labeled dog and cat images and trains a neural network to compute those features which are capable of distinguishing between a dog image vector and a cat image vector.

Deep learning with neural networks is limited by the availability of datasets (which usually must be constructed by humans), the intense computational demands of the training process, and the present domination of heuristic methods for network design. As of yet, there is no accepted general-purpose theory for designing a neural network to solve some problem. A large mathematical literature exists providing some theoretical results on the capabilities of neural networks, as well as

frameworks for understanding the mathematically the functioning of neural networks (in particular, CNN [43, 44, 48, 69, 66]).

Section 4.2

Neural networks: notation

A neural network is a parameterized function $f_\theta : X \to Y$ with a particular structure, capable of performing regression or classification tasks. The structure of a neural network is a heavily simplified mathematical approximation of animal neural activity, and consists of a sequence of *layers* of *neurons* followed by a nonlinear *activation function*, connected by weight matrices and added bias vectors. A one-layer neural network $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$ is defined by $f_\theta(x) = a(Wx+b)$, where $W \in M^{m\times n},\ b \in \mathbb{R}^m,\ \theta = \{W, b\},\ a : \mathbb{R} \to \mathbb{R}$ and the nonlinear function a is applied element-wise to the vector $Wx+b$. More complicated networks are formed by adding *hidden layers* whose output are not the final product of the network. Given weight matrices $W = \{W_i \in M^{n_i\times n_{i-1}}\}_{i=1}^k$ and bias vectors $b = \{b_i \in \mathbb{R}^{n_i}\}_{i=1}^k$ summarized by $\theta = (W, b)$, and activation functions $\{a_i : \mathbb{R} \to \mathbb{R}\}_{i=1}^k$, we define a k-layer neural network by:

$$f(x;\theta) = (f_k \circ f_{k-1} \circ \cdots \circ f_1)(x;\theta); \quad f_i(x;\theta) = a_i(W_i x + b_i) \tag{4.1}$$

where $n_0 = n, n_k = m$.

Activation functions play the crucial role of introducing nonlinearities into what would otherwise be a purely affine function, thus enabling neural networks to internally represent complicated features of data. We require that a neural network f be differentiable almost everywhere with respect to its parameters, whence the activation functions used by f must be differentiable almost everywhere.

Common choices of activation function include the Rectified Linear Unit (ReLU), $a(x) = \max\{0, x\}$ [46], hyperbolic tangent $a(x) = \tanh(x)$, or sigmoid $\sigma(x) = (1 + e^{-x})^{-1}$ functions.

The term *network architecture* is used to describe f together with any constraints on the parameter set. The network defined by (4.1), has a *fully connected* architecture with $(k-1)$ hidden layers. If we constrain the weights to be banded Toeplitz matrices, the resulting architecture would be *convolutional* with k convolutional layers[1]. Another common network architecture involves the use of *skip* or *residual connections*. As an example, consider

$$f_{res}(x;\theta) = (f_{res,\ k} \circ f_{res,\ k-1} \circ \cdots \circ f_{res,1})(x;\theta);$$

with

$$f_{res,i}(x;\theta) = x + a_i(W_i x + b_i),$$

for a network architecture that employs skip connections, and

$$f_{res,i}(x;\theta) = x - a_i(W_i x + b_i),$$

for a residual network architecture. Both cases must obey dimensionality constraints so that $n = m = n_i$ for every i. The selection of a particular network architecture is a matter of experimentation, experience, and the network's intended purpose.

[1]Multiplication by a banded Toeplitz matrices represents the convolution of a vector with a filter; see Section 4.6 for a fuller discussion of convolutional networks.

Section 4.3

Loss functions and regularization

Provided a network architecture f, it remains to learn a robust parameter set θ from a training set. The training set for a *supervised learning* problem consists of sets $T_X = \{x_i\}_i \subset X$, $T_Y = \{y_i)\}_i \subset Y$ of input vectors x_i and ground truth vectors $y_i := u(x_i)$ for an unknown function $u : X \to Y$. The goal of supervised learning, given f, is to select θ such that $f_\theta(x) \approx u(x)$ for $x \in X$, especially for $x \notin T_X$. Consider $y_* = u(x_*)$ for $x_* \notin T_X$, and suppose that we have trained f on the training data to obtain f_θ. If we consider the trained neural network as a function of the training set $f_\theta(x) = f(x; T_x)$, we see that the trained network acts as a point estimator $\widehat{y_*}$ for the unknown, fixed quantity x^*. This viewpoint requires that we treat T_X as a random variable sampled from a data-generating distribution ψ. In this way, we may apply the bias–variance tradeoff, namely,

$$\mathbb{E}_\psi \left[(y_* - f(x_*; X_t))^2 \right] = (y_* - f(x_*; X_t))^2 + \mathbb{V}_\psi(f(x_*; X_t)),$$

demonstrating that the generalization error consists of a squared bias term and the variance of the network's prediction with respect to the training data set. It is thus important when training a network that we do not *overfit*, reducing prediction error on the training set T_X, T_Y at the cost of a high variance in the network's predictions for $x \notin T_X$. Overfitting may result from the naive minimization of a distance function ℓ in prediction space over the training set

$$\theta^* = \arg\min_\theta \sum_i \ell(y_i, f_\theta(x_i)).$$

To this end, various regularization techniques have been proposed and adopted for the training of deep neural networks. By choosing a regularization function on the parameter space $\Theta : \Omega \to \mathbb{R}^+$

and a regularization strength $\lambda \in \mathbb{R}^+$, we obtain an objective function $L : Y \times X \times \Omega \to \mathbb{R}^+$ given by

$$L(y, x, \theta) = \lambda\Theta(\theta) + \sum_i \ell(y_i, f_\theta(x_i)). \tag{4.2}$$

From (4.2) we seek the optimal parameters θ^* by

$$\theta^* = \arg\min_\theta \sum_i L(y_i, f_\theta(x_i, \theta)). \tag{4.3}$$

Section 4.4

Optimization algorithms

Deep learning depends upon the robust selection of the parameters of a network by (4.3). This optimization process is called *training* the network. The following principles guide the training of neural networks:

1. A neural network $f(x; \theta)$ is differentiable almost everywhere with respect to its parameters θ; the gradient of the objective function $\nabla_\theta L(x_i; \theta)$ can be computed by *automatic differentiation* using numerical libraries.

2. The parameter space Ω is very high-dimensional, commonly consisting of hundreds of thousands or millions of parameters.

3. The size N of the training set T_x, T_y may be large or small in comparison to the dimensionality of θ depending on the specific application domain and available data.

4. Overfitting the training set produces a useless network.

We are interested in the minimization of the high-dimensional objective function L with respect to θ. By principle two, second-order methods that require the computation of a Hessian matrix is infeasible because of the prohibitively high dimension of Ω. Instead, first-order optimization methods are in widespread use. Of these, the simplest is *gradient descent*. The parameters are iteratively updated by the rule

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\eta}{N} \sum_{i=1}^{N} \nabla_\theta L(y_i, x_i; \theta^{(t)})$$

where η is the *learning rate*, a *hyperparameter* not included with the learned parameters θ. For gradient descent, each step of training requires computing the gradient of the objective function evaluated at each of the N data points, which is expensive to compute when N is large. Two related alternatives are stochastic gradient descent and mini-batch gradient descent.

Stochastic gradient descent updates the model parameters based of the gradient evaluated at a single data point $x_{i^{(t)}} \in T_X$, selected uniformly at random at each step.

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta L(y_{i^{(t)}}, x_{i^{(t)}}; \theta^{(t)})$$

Computing the gradient of the objective function at a single data is N times less expensive than computing it for the entire dataset, leading to an N-fold increase in the rate of parameter updates. This comes at a penalty to the accuracy of the updated parameters over the whole dataset. As a widely popular compromise between iteration frequency and update accuracy, *mini-batch gradient descent* randomly partitions the training data set into subsets of size B (possibly repeating some data), and successively updates parameters over these "mini-batches".

More sophisticated optimization techniques than the stochastic gradient descent have been empirically observed to produce faster convergence to an optimal parameter set. Momentum-based

methods bias parameter updates in favor of a term accumulating past gradients called *momentum*. Classical momentum [50] updates by

$$m^{(t+1)} = \beta m^{(t)} - \nabla_\theta f(x; \theta)$$

$$\theta^{(t+1)} = \theta^{(t)} + m^{(t+1)}$$

If we assume that ∇_θ is constant, it is easy to see that $m^{(t)} = -\nabla_\theta \sum_{i=1}^{t} \beta^{i-1}$, so

$$\lim_{t \to \infty} m^{(t)} = -\frac{\nabla_\theta}{1 - \beta},$$

which gives a $\frac{1}{1-\beta}$ speedup over stochastic gradient descent[2]. *Nesterov accelation*, proposed in [58] for neural network optimization as a reformulation of work by Nesterov [47], updates thus:

$$m^{(t+1)} = \beta m^{(t)} - \nabla_\theta f(x; \theta + \eta m^{(t)})$$

$$\theta^{(t+1)} = \theta^{(t)} + \eta m^{(t+1)}$$

where $m^{(0)} = 0$ and $\beta \in [0, 1)$ controls the rate of momentum[3].

Another family of optimization techniques adapts the learning rate for each individual component of the parameter vector θ. Adam [30] is a popular example of this, with update computed by estimating the first and second moments of the update vector. With initialization $m^{(0)} = v^{(0)} = 0$, fixed $\beta_1, \beta_2 \in (0, 1)$, and initial learning rate α, Adam updates the gradient according to Algorithm 1[4].

[2] It is of course implausible that ∇_θ is constant over *all* of Ω, but if the loss surface is smooth, then for step sizes small enough the gradient of the loss will be approximately constant over some local region of Ω.

[3] Although formulas here are given for updating parameters from a single sample for simplicity, in practice the mini-batch variation of these algorithms are in wide use.

[4] $\odot, \oslash$, and $\sqrt{\ }$ operate element-wise on vectors in the sense of Hadamard.

Algorithm 1: Adam algorithm for adaptive learning rate optimization [30]

Data: $\beta_1, \beta_2 \in (0, 1)$: moment estimation parameters, α: base learning rate, θ: network parameter initialization, $\epsilon > 0$: stability parameter

$m^{(0)} = 0$

$v^{(0)} = 0$

$\theta^{(0)} = \theta$

$t = 1$

while *training* **do**

$m^{(t+1)} = \beta_1 m^{(t)} + (1 - \beta_1)\nabla_\theta f(x; \theta)$

$v^{(t+1)} = \beta_2 v^{(t)} + (1 - \beta_2)\nabla_\theta f(x; \theta) \odot \nabla_\theta f(x; \theta)$

$\hat{m}^{(t+1)} = m^{(t)}/(1 - \beta_1^t)$

$\hat{v}^{(t+1)} = v^{(t)}/(1 - \beta_2^t)$

$\theta^{(t+1)} = \theta^{(t)} - \alpha\hat{m}^{(t+1)} \oslash (\sqrt{\hat{v}^{(t+1)}} + \epsilon)$

end

Exponential moving averages compute biased estimates of the first and second moment, $m^{(t)}, v^{(t)}$, which are then corrected by scaling factors which correct for the initialization of these terms to 0. [30] provide a convergence analysis for Adam given a convex cost function L. In practice, Adam is widely used to train convolutional neural networks which are non-convex in general.

Section 4.5

A note on convexity

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if

$$f(\gamma\alpha + (1 - \gamma)\beta) \leq \gamma f(\alpha) + (1 - \gamma)f(\beta) \quad \forall \alpha, \beta \in \mathbb{R}^n, \gamma \in [0, 1].$$

There exists an extensive literature on the optimization of convex functions which one may wish to apply to the optimization of neural networks. Certainly neural networks, which are valuable precisely for their ability to approximate arbitrary unknown functions [11], are not convex with respect to

their data x. However, we show that the objective functions of neural networks are not convex with respect to the network paramaters θ either. Many optimization algorithms and theoretical results that apply to convex functions are not, therefore, rigorously justified in their application to optimizing the parameters of a neural network. Empirical work demonstrates nevertheless that convex optimization techniques can yield impressive results when used to optimize neural network parameters.

Why are the objective functions of neural networks non-convex? The composition of convex functions is not, in general, convex [7], but we provide a very simple example in which convexity is violated.

Theorem 4. *The loss function of a neural network is not, in general, a convex function of its parameters.*

Proof. Consider a neural network network given by

$$f(x; w_{11}, w_{12}, w_{21}, w_{22}) = w_{21}\rho(w_{11}x) + w_{22}\rho(w_{12}x)$$

$$x, w_{11}, w_{12}, w_{21}, w_{22} \in \mathbb{R}$$

where $\rho(z)$ denotes $\max\{0, z\}$, with squared error objective function

$$L(y, x, w_{11}, w_{12}, w_{21}, w_{22}) = (y - f(x; w_{11}, w_{12}, w_{21}, w_{22}))^2$$

and training set consisting of a single sample $x_0 = a, y_0 = a$ for $a > 0$. Then,

$$L(a, a; 1, -1, 1, -1) = (a - (\rho(a) - \rho(-a)))^2 = 0$$

$$L(a, a; -1, 1, -1, 1) = (a - (-\rho(-a) + \rho(a)))^2 = 0$$

$$L(a, a; 0, 0, 0, 0) = (a - 0)^2 = a^2 > 0$$

Thus for $\alpha = (a, a, 1, -1, 1, -1)^T, \beta = (a, a, -1, 1, -1, 1)^T$, and $\gamma = \frac{1}{2}$,

$$L(\gamma\alpha + (1 - \gamma)\beta) = L(\tfrac{1}{2}\alpha + \tfrac{1}{2}\beta) = L(a, a, 0, 0, 0, 0) = a^2$$

while

$$\gamma L(\alpha) + (1 - \gamma)L(\beta) = \tfrac{1}{2}L(\alpha) + \tfrac{1}{2}L(\beta) = 0.$$

Hence L is not a convex function of its weights $\{w_{i,j}\}$. □

By Theorem 4, we have no *a priori* guarantee that convex optimization techniques like gradient descent will converge to a global minimum of $f(x; \theta)$ instead of a saddle point or local minimum.The choice of initial weight parameters (*initialization*) may influence the point to which these algorithms converge. For our present purposes, we will always employ the initalization given by [24] unless stated otherwise.

Section 4.6

Convolutional neural networks

Convolutional neural networks (CNNs), introduced in [37, 36], have been responsible for some of the most remarkable achievements of deep learning [38]. A CNN is a neural network with the constraint that the weight matrices W must perform discrete cross-correlations across their input. This constraint allows CNNs to exploit the inherent self-similarity of images, video, audio, and

text, while requiring far fewer parameters than an equivalently deep fully-connected neural network. Nonlinearities between layers of the network are still used.

A convolutional layer is specified by the number of channels n of its input and a set of m $K \times K$ *filter banks*. We can represent the ith filter bank by a set of $K \times K$ matrices (called *filters*, or *kernels*) $\{\phi_{i,j}\}_{j=1}^{n}$. The output of a convolutional layer is a set of matrices $\{Y_i\}_{i=1}^{m}$ related to the input set of matrices $\{X_j\}_{j=1}^{n}$ by

$$Y_i = \sum_{j=1}^{m} X_j * \phi_{i,j}$$

where $*$ denotes discrete cross-correlation in two dimensions, i.e.

$$(X_j * \phi_{i,j})[a,b] = \sum_{h=1}^{K_1} \sum_{w=1}^{K_2} X_j[h+a, w+b]\phi_{i,j}[h,w]$$

for those a, b indexing Y_i. Hence each output channel Y_i of a convolutional layer is a sum of discrete 2d cross-correlations between each of the n input channels and its corresponding filter in the ith filter bank. It is helpful to think of a filter bank as a 3d "block" of stacked filters, which convolves across the 3d "block" of input channels computing inner products at each increment of the convolution. Each block inner product computes a single entry of Y_i.

The discrete cross-correlation of a matrix $X \in \mathbb{R}^{N_1,N_2}$ with a kernel $\phi \in \mathbb{R}^{K_1,K_2}$ produces a matrix in $\mathbb{R}^{N_1-K_1+1,N_2-K_2+1}$. This kind of dimensionality reduction is usually undesirable, and hence the input channels to a convolutional layer are usually *padded* with the appropriate number of zeros around their border. This operation makes little sense when thinking about matrices as linear transformations, but considerably more sense when a matrix simply represents a grid of pixel values. To avoid asymmetric padding, it is common to only use odd-sized kernels in CNN layers.

Note that although in this notation, a CNN f operates on a set of matrices X to produce another set of matrices Y, it is for convinence that we pursued this formalization. A two-dimensional

convolutional layer can be formulated in terms of matrix multiplication in at the cost of easy comprehensibility; we omit the construction here. In one dimension, it is easily seen how the discrete cross-correlation of vectors is equivalent to matrix multiplication using Toeplitz matrices. The Toeplitz matrix of a kernel $\phi \in \mathbb{R}^k$ for inputs in $\mathbb{R}^n$ is given by

$$T(\phi|n) = \begin{pmatrix} \phi[1] & \dots & \dots & \phi[k] & 0 & \dots & \dots \\ 0 & \phi[1] & \ddots & \ddots & \phi[k] & 0 & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \dots & \dots & \phi[1] & \dots & \dots & \phi[k] \end{pmatrix}$$

so that $T(\phi|n) \in \mathbb{R}^{(n-k+1)\times k}$ and

$$x * \phi = T(\phi|n)x$$

where $*$ denotes discrete cross-correlation. Alternatively, we may consider the non-circular structured Hankel matrix of x with respect to the kernel ϕ,

$$H(x|k) = \begin{pmatrix} x[1] & \dots & x[k] \\ x[2] & \dots & x[k+1] \\ \vdots & \ddots & \vdots \\ x[n-k+1] & \dots & x[n] \end{pmatrix} \in \mathbb{R}^{(n-k+1)\times n}$$

and thus

$$x * \phi = H(x|k)\phi.$$

We note that [69] have developed a theory of *deep convolutional framelets* to investigate deep convolutional neural networks within the context of solving inverse problems, finding conditions

under which perfect recovery from an input signal under a CNN coding is possible and investigating the role of ReLU nonlinearities and skip connections.

Section 4.7

Theoretical work on neural networks

As yet, there is no comprehensive, standard mathematical theory of deep neural networks, despite a proliferation of work towards this end. Neural networks have been studied from a variety of mathematical viewpoints, including approximation theory, wavelet theory, and the theory of sparse representation. Convolutional neual networks have, in particular, received much attention for their astounding recent applications in image processing and classification. We will review two select examples from this mathematical literature here and in Section 4.8; a comprehensive review is out of the scope of this work.

Empirical demonstrations that neural networks can successfully learn approximations of many kinds of unknown functions from data samples has been accompanied by a large literature of approximation theoretic results. We limit ourselves to a single, important example. In [11], Cybenko showed that for any continuous function on the unit cube $g : [0, 1]^n \to \mathbb{R}$ and $\epsilon > 0$, there exists a two-layer neural network f

$$f(x) = W_2(\sigma(W_1 x + b)), \quad W_1 \in \mathbb{R}^{m \times n}, W_2 \in \mathbb{R}^{1 \times m}$$

such that

$$\sup_{[0,1]^n} |f - g| < \epsilon$$

where σ is any fixed continuous sigmoidal function, i.e. $\lim_{t\to-\infty}\sigma(t)=0$ and $\lim_{t\to\infty}\sigma(t)=1$. This is the *universal approximation property* of neural networks. Unfortunately, this result gives no bound on the width m of the hidden layer or indeed any relationship between an optimal m and the properties of g. Further, it is the case in practice that successful neural network are much deeper than two layers. Recent work investigating multi-layer networks and a review of earlier such work can be found in [6].

Section 4.8

Convolutional sparse coding and CNNs

Papyan et al. observe [48] significant connections between CNN and the proposed *convolutional sparse coding (CSC)* scheme in the sparse representation literature. Suppose we have a signal $X\in\mathbb{R}^N$ and a *dictionary* $D\in\mathbb{R}^{N\times M}$ with $M\geq N$ and normalized columns. The sparse coding problem is to find an optimally *sparse* signal Γ to represent X over D:

$$\Gamma^*=\arg\min_{\Gamma}\|\Gamma\|_0 \quad \text{s.t. } D\Gamma=X$$

with $\|\cdot\|_0$ counting the number of non-zero entries of its vector argument. A computationally tractable relaxation of the problem is *basis pursuit*, which employs the ℓ_1 norm:

$$\Gamma^*=\arg\min_{\Gamma}\|\Gamma\|_1 \quad \text{s.t. } D\Gamma=X$$

If a sparse representation has $\|\Gamma\|_0<\frac{1}{2}(1+\frac{1}{\mu(D)})$, then the representation is unique and basis pursuit can recover it[5][48]. Approximate solutions to the sparse coding and basis pursuit problems

[5]Define the *mutual coherence* by $\mu(D)=\max_{i\neq j}|d_i^Td_j|$

are given by the hard and soft thresholding schemes

$$\Gamma^* = \arg\min_{\Gamma} \|\Gamma - D^T X\|_2^2 + \beta\|\Gamma\|_0 \quad \text{(hard threshold)} \tag{4.4}$$

$$\Gamma^* = \arg\min_{\Gamma} \|\Gamma - D^T X\|_2^2 + \beta\|\Gamma\|_1 \quad \text{(soft threshold)} \tag{4.5}$$

which admit closed form solutions $H_\beta(D^T X)$ and $S_\beta(D^T X)$ respectively, using the hard and soft *thresholding operators*

$$H_\beta(x) = \begin{cases} x & |x| > \beta \\ 0 & \text{otherwise} \end{cases}$$

$$S_\beta(x) = \begin{cases} x+\beta & x < -\beta \\ x-\beta & x > \beta \\ 0 & \text{otherwise.} \end{cases}$$

When we require (without loss of generality) that the approximate sparse encoding have positive entries, the soft thresholding operator becomes the ReLU function $\rho(x - \beta)$.

Convolutional sparse coding imposes the requirement that D must be *convolutional*, such that each local *patch* $x_i \in \mathbb{R}^n$ of X is recovered by $\Omega\gamma_i$, where $\gamma_i \in \mathbb{R}^{(2n-1)m}$ is a local *stride* in Γ and $\Omega \in \mathbb{R}^{n\times(2n-1)m}$ consists of rows i through $i + n - 1$ of D, with zero columns removed. If D is convolutional, Ω is identical across i. This structure imposes that each x_i is the discrete cross-correlation of γ_i with the m filters encoded in Ω. Enforcing a convolutional dictionary lessens the complexity of finding a sparse representation for large signals without reducing the signal to a set of unconnected patches. The *multi-layer convolutional sparse coding (ML-CSC)* applies the CSC scheme recursively on the generated representations. The *deep coding problem* DCP_λ, given a

sequence of dictionaries $\{D_k\}_{k=1}^K$ is to find a set of sparse representations $\{\Gamma_i\}_{i=1}^K$ such that

$$\Gamma_{k-1} = D_k\Gamma_k, \qquad 1 \leq k \leq K, \qquad \|\Gamma_k\|_{0,\infty}^s \leq \lambda_k \tag{4.6}$$

where $\Gamma_0 := X$ and $\|\cdot\|_{0,\infty}^s = \max_i \|\gamma_i\|_0$ is the local sparsity norm $\ell_{0,\infty}$, taken across strides γ_i in the argument. If the input signal is corrupted by noise $Y = X + E$, then deep coding problem becomes DCP_λ^ϵ, with (4.6) replaced by

$$\|\Gamma_{k-1} - D_k\Gamma_k\|_2 \leq \epsilon_k, \qquad 1 \leq k \leq K, \qquad \|\Gamma_k\|_{0,\infty}^s \leq \lambda_k. \tag{4.7}$$

The hard/soft thresholding approximate solutions (4.5, 4.4) can be extended to ML-CSC to obtain the *layered thresholding*. In particular, for the soft thresholding operator with thresholds $\{\beta_k\}_{k=1}^K$the approximate solutions are

$$\hat{\Gamma}_k = S_{\beta_k}(D_k^T\Gamma_{k-1}), \qquad 1 \leq k \leq K \tag{4.8}$$

which bears a striking resemblence to the definition of a neural network (4.1). In particular, for a one-dimensional CNN with K layers and convolutional matrices given by $\{W_k^T\}$,

$$f_k = ReLU(W_k^T f_{k-1} + b_k), \qquad 1 \leq k \leq K.$$

Papyan et al. [48] conclude that the ML-CSC basis pursuit by layered thresholding is equivalent to the CNN forward pass (up to a discrepancy in the thresholds caused by the CNN bias vectors $\{b_k\}$), and prove a number of theoretical guarantees for the above approaches which we briefly summarize.

We may wish to have guarantees about the uniqueness of optimal DCP_λ representations, and the stability of optimal representations in the presence of noise. Uniqueness of representations $\{\Gamma_k\}$

satisfying DCP_λ for a signal X is guaranteed under the sparsity condition

$$\|\Gamma_i\|_{0,\infty}^s \leq \lambda_k \leq \frac{1}{2}\left(1 + \frac{1}{\mu(D_k)}\right), \qquad 1 \leq k \leq K.$$

A bound for the error between the true representations $\{\Gamma_k\}$ of a signal X and the representations $\{\hat{\Gamma}_k\}$ of the noisy $Y = X + E$ is provided. This bound is unfortunately exponential in the number of layers:

$$\|\Gamma_k - \hat{\Gamma_k}\|_2^2 \leq \epsilon^2 \prod_{j=1}^{k} \frac{4}{1 - (2\|\Gamma_j\|_{0,\infty}^s - 1)\mu(D_j)},$$

with $\|E\|_2 \leq \epsilon$.

Although layered thresholding gives only an approximation of the optimal representations, [48] prove that with local sparsity assumptions on the optimal representations that depend on $\frac{\min_i |\Gamma[i]|}{\max_i |\Gamma[i]|}$, the layered thresholding approximate solutions have correct support and bounded $\ell_{2\infty}$ error. The layered soft thresholding requires stricter local sparsity assumptions and has a less bounded $\ell_{2,\infty}$ error. Using ReLU for CNN nonlinearities, which corresponds to the soft thresholding operator under this analysis, is less well-suited for the approximation of optimal sparse representations across layers compared to hard thresholding.

Instead of seeking approximately optimal representations with thresholding-based solutions, Papyan et al. propose the *layered basis pursuit* to recover the optimal representations themselves. This scheme has

$$\hat{\Gamma}_k = \arg\min_{\Gamma_k} \|\Gamma_k\|_1 + \frac{1}{2\xi_k}\|D_k\Gamma_k - \hat{\Gamma}_{k-1}\|_2 \tag{4.9}$$

with constants $\{\xi_k\}$ proportional to the level of corrupting noise. Note that (4.9) is a version of the proximal operator of Section 2.4 for layered pursuit. Papyan et al. propose the iterative soft

thresholding algorithm for the solution (4.9) by

$$\hat{\Gamma}_k^{(t)} = S_{\xi_k/c_k}\left(\hat{\Gamma}_k^{(t-1)} + \frac{1}{c_k} D_k^T(\hat{\Gamma}_{k-1} - D_k\hat{\Gamma}_k^{(t-1)})\right)$$

and give conditions for when layered basis pursuit is guaranteed to recover optimal representations, as well as an (exponential in depth) bound on errors for noisy signals [48].

This framework provides some illuminating connections between CNNs and the body of work on sparse representations, but suffers from several limitations. A trained CNN can be expected to have learned convolutional dictionaries for sparse coding only when a suitable regularization is imposed such as the ℓ_1 norm on filter coefficients – but successful networks exist that do not impose sparsity. The role of skip connections in existing successful CNN architectures is not addressed, and the relationship between the number of filter channels per layer and CNN effectiveness is not explored.

Chapter 5

Demosaicing

Color image demosaicing, one of the first steps in the digital image formation pipeline, consists of the interpolation of unobserved color measurements on the pixel grid of a digital camera. Many commercially available cameras rely on some demosaicing algorithm to recover usable pictures from light measurements. There is a substantial academic literature proposing demosaicing techniques, including some recent work on deep neural networks for demosaicing.

We introduce the problem of color image demosaicing. A model of digital color image formation is presented along with the basics of colorimetry necessary to understand the demosaicing literature. At its core, demosaicing is the interpolation of missing information about the color channels of an acquired digital image. The missing information is inherent in the process used to measure colored light using a color filter array in digital cameras. Traditional heuristic interpolation methods for demosaicing are presented, as well as recent methods using deep convolutional networks. Demosaicing can be modeled as an inverse problem; we review a deep learning architecture designed around this principled observation that has been proposed. The problem of temporal or multi-frame demosaicing is presented. Finally, we review a recent successful application of convolutional networks to the problem of computing optical flow.

Section 5.1

Digital color image formation

Digital cameras are designed to create a discrete, digital representation of a natural scene from a brief exposure to the photons that enter the camera during the interval of exposure. Entering photons are refracted by a lens before passing through the open shutter, where they are absorbed by charge-coupled device (CCD) image sensor[1]. The CCD consists of a grid of capacitors, each of which has an electron potential well. Each capacitor on the CCD grid accumulates electrons in its potential well in proportion to the photons illuminating the capacitor. Once exposure is complete, the pattern of electrons in the potential wells of the grid is converted into a digital signal that can be used by subsequent digital processing components of the camera. For a more thorough reference on the material presented in Sections 5.1, 5.2, and 5.3, see [39].

The digital image acquisition pipeline is multi-stage and varies from one camera manufacturer to the next. Each camera must be calibrated so as to counteract the complicated noise inherent in CCD measurement, any defects or nonuniformities in the sensor, black-level correction and white balancing, color correction, and demosaicing, our present subject. Before addressing demosaicing, it is necessary to address the basics of color.

Section 5.2

Light, color, and colorimetry

Light is an highly complex natural phenomenon. We will simplify our discussion of light to suit our present needs, with the understanding that our treatment is highly incomplete. Visible light

[1]CMOS sensors are an alternative, but we will assume a CCD camera in the following.

refers to an electromagnetic waves with temporal frequencies in the range 4×10^{14} Hz to 7.8×10^{14} Hz. Light carries energy which is always absorbed in discrete quantities $h\nu$, where h is Planck's constant and ν is the frequency of the light.

The human visual system continually translates visible light that illuminates the human eyes into the qualitative perception in the mind called sight. Human perception of color is highly context-dependent – it is not the case that light of a given frequency will appear as the same color to a human observer.

The human eye contains four kinds of photoreceptive units: rods and three types of cones. Rods are responsible for light perception in low brightness situations, while cones are responsible for light perception in bright situations. The different kinds of photoreceptors in the eye are responsive to different wavelengths of light. Rods are more sensitive to high-frequency light than are cones. The three types of cones also vary in their responses to the spectrum of light. S, M, and L cones are most sensitive to high, medium, and low light frequencies respectively (S, M, and L stand for short, medium, and long wavelengths). The visual perception of color is created from the different responses of the three types of cones to light.

The field of *colorimetry* seeks to specify the physical aspects of color stimuli using the framework of vector spaces. Colorimetry operator on the receptor-level theory: a color stimulus is completely specified by its effects on human photoreceptors, and in particular, the three types of cones. Colorimetry was established before it was technically possible to directly measure human cone responses, and so instead relied upon psychophysical color matching experiments. In such an experiment, three *primaries* are selected, **(R), (G), (B)** such that the combination of one unit of radiant power from each primaries results in "equal-energy white", which has equal radiant power at every visible wavelength.[2] A color stimulus **(C)** is then projected onto one half of a viewing field, and a linear

[2]The term *radiant power* describes the energy flow per unit time through a point in a given direction caused by light.

Figure 5.1: A equally-spaced gradient in sRGB space (top) and an equally-spaced gradient in linRGB, converted to sRGB for display (bottom)

combination of the primaries is projected onto the opposite half. The coefficients R, G, B of the combination of primaries are adjusted until the relationship

$$\mathbf{(C)} \equiv R\mathbf{(R)} + G\mathbf{(G)} + B\mathbf{(B)}$$

holds, with $\equiv$ denoting the perceptual equivalence of both halves of the viewing field. When the stimulus $\mathbf{(C)} = \mathbf{(C)}(\lambda)$ consists of monochromatic light of wavelength λ, the *color matching functions* $a_r(\lambda), a_g(\lambda), a_b(\lambda)$ of the primaries are obtained by varying λ:

$$\mathbf{(C)}(\lambda) \equiv a_r(\lambda)\mathbf{(R)} + a_g(\lambda)\mathbf{(G)} + a_b(\lambda)\mathbf{(B)}$$

The standard color space for use on computer monitors and is known as sRGB space [2]. The representation of color in sRGB depends on the ITU-R BT.709 reference primaries, with an additional correction for the nonlinear response of computer monitor displays. Suppose a signal $\mathbf{(C)}$ has chromaticity coefficients R, G, B with respect to the sRGB primaries. The vector $[R\ G\ B]^T$ gives the *linear RGB (linRGB)* representation of $\mathbf{(C)}$. The sRGB standard instead represents $\mathbf{(C)}$ as $[R'\ G'\ B']^T$ where

$$X' = \begin{cases} X/12.92 & X \leq 0.003040 \\ 1.055X^{1/2.4} - 0.055 & X > 0.003040 \end{cases}$$

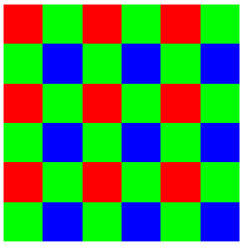


Figure 5.2: A 6x6 patch of a Bayer color filter array

for $X \in \{R, G, B\}$. This fact is important for working with digital image formats – if an image is encoded as sRGB, the color coefficients of each pixel are actually a nonlinear function of the chromaticity coefficients with respect to colorimetric primaries. Figure 5.1 illustrates the danger of ignoring the distinction between sRGB and linRGB space. The linear gradient created by equally-spaced values in sRGB space appears choppy and broken, while the linear gradient in linRGB space that is then converted to a (nonlinear) sRGB gradient appears smooth. When performing interpolation tasks, therefore, it is important to consider the choice of color space to work in.

Section 5.3

The problem of demosaicing

Much like the human eye uses three kinds of cones to perceive color, each sensitive to different spectra of light, a digital camera employs a color filter array (CFA) overlayed on top of its grid of capacitors to capture color information. The CFA is a grid of differently photosensitive materials that are sensitive to specific wavelengths of light. The Bayer CFA, a common choice of color filter array, consists of two rectangular grids of pixels sensitive to red and blue light, and a checkerboard grid of pixels sensitive to green light (see Figure 5.2). The Bayer CFA samples the green channel at twice the rate as the red and blue channels. The human visual system is most sensitive to

mid-spectrum light, and so it is reasonable to reserve the highest sampling rate of the CFA for green (mid-spectrum) light. Because of this, the green channel sampled by the Bayer CFA is commonly referred to as the *luminance component*, while the red and green channels are referred to as the *chrominance components*. Although there exist color systems that explicitly formulate color channels as luminance and chrominance (for example the NTSC YIQ color system used in color television), this is not what is meant by luminance and chrominance components in reference to a Bayer CFA. Instead, the terms are used to denote that the green channel contains the better-sampled brightness (luminance) information, while the red and green channels contain the relatively worse-sampled color (chrominance) information. Indeed, Bayer used the luminance / chrominance vocabulary in his original patent for the CFA that bears his name [4].

When taking a picture, a digital camera using a Bayer CFA is exposed to the light of some real-world scene but only captures the red, green, and blue linRGB coefficients according the Bayer CFA grid. To represent the image digitally, all three linRGB coefficients are needed for each pixel. In this way, 2/3 of the total required information is not recorded. Furthermore, the third of information that is recorded is corrupted by noise from the CCD. *Demosaicing* is the task of generating the 2/3 of missing information from the measured 1/3 of (noisy) information. Often, demosaicing is discussed along with *denoising*, the task of correcting the noisy measurements. Together, demosaicing and denoising constitute a crucial segment of the image capture pipeline.

A model for the statistics of the noise corrupting the Bayer measurements is given by Foi et al. [17]. The distribution of the noise n is decomposed into a signal-independent Gaussian component and a signal-dependent Poisson component. The Poisson term accounts for photon noise (which is proportional to the strength of the signal at a given pixel), whereas the Gaussian term accounts for background noise that applies across the image independent of signal level. This combined noise term can be approximated by a signal-dependent Gaussian distribution. The joint demosaicing-denoising

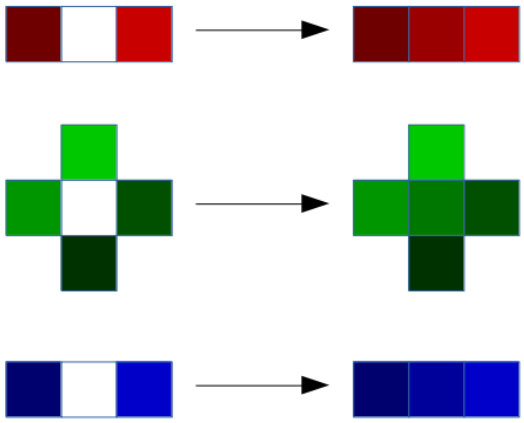


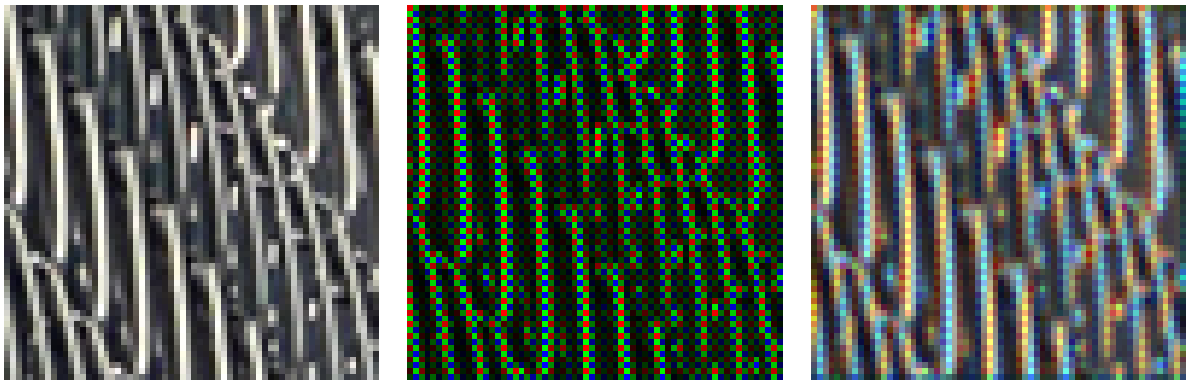
Figure 5.3: A demonstration of demosaicing by bilinear interpolation

literature often uses a Gaussian approximation of the noise term. We will be treating the denoised or non-noisy case in Chapter 6, but include this discussion to acknowledge the incompleteness of demosaicing without taking noise into consideration.

Section 5.4

Demosaicing by interpolation

Let us consider a very simple approach to the demosaicing problem. Given a mosaiced image Mx, we separately interpolate each of the three color channels of Mx using bilinear interpolation (see Figure 5.3). The result for a very high frequency image patch is shown in Figure 5.4. Distracting false color artifacts are noticeably present when using bilinear interpolation to demosaic high frequency images, although in low-frequency patches bilinear interpolation performs adequately. Indeed, separate bilinear interpolation across each of the three color channels completely ignores the information that the three color channels contain jointly.



(a) Ground truth patch (b) Bayerized patch (c) Bilinear interpolation

Figure 5.4: Demosaicing by bilinear interpolation

However, there exist more sophisticated interpolation-based demosaicing methods in the literature. We will briefly review some of them here. The review papers [41] [45] [20] contain a more exhaustive overview of traditional interpolation-based demosaicing methods.

In [10], hue-based interpolation heuristics are considered. The *hue* at a pixel x relates the value of a chrominance component at R_x to the luminance component G_x by either R_x/G_x or $\log R_x - \log G_x$. Hue-based methods operate on the assumption that hue channels are smoother than color channels. First, the missing pixels of the green channel are interpolated using bilinear interpolation. From this full-resolution luminance component, the hue values for the observed chrominance components may be calculated. Then, the missing red and blue hues are bilinearly interpolated from the calculated hues, and converted back to color channel components.

Adams [1] extends beyond this to address the reality that interpolating colors across edges in an image creates zipper artifacts, as we observed in Figure 5.4. A heuristic adaptive interpolation methods is proposed that first approximates the horizontal and vertical derivatives of the luminance component, and interpolates either horizontally, vertically, or bilinearly depending on whether the horizontal and vertical derivatives exceed a set threshold. This method has the advantage of

handling edges slightly better than plain bilinear interpolation or bilinear hue interpolation while remaining computationally simple.

An interesting recent development in the demosaicing literature has been the introduction of deep convolutional neural networks trained to demosaic images. Gharbi et al. [19] propose a network that uses a 1-channel, 2-dimensional Bayer mosaic M as input. Prior to convolutions, M is rearranged into a four channel image at half the spatial resolution by

$$F_c^0(x, y) = M(2x + (c \bmod 2), 2y + \lfloor c/2 \rfloor)$$

where $c \in \{1, 2, 3, 4\}$ and $(c \bmod 2)$ refers to the integer remainder of dividing c by 2. This operation, sometimes referred to as *compression*, effectively increases the range of subsequent convolutions performed across the feature channels. A $K \times K$ convolutional kernel applied to the compressed image has an effective range of $4K \times 4K$ pixels in the input space. A fifth channel is included to explicitly model the variance σ of the zero-mean Gaussian noise corrupting the input sample M,

$$F_5^0(x, y) = \sigma$$

Gharbi's network processes the compressed image through D convolutional layers with ReLU nonlinearities,

$$F_c^d = \rho\left(b_c^d + \sum_{c'=1}^{W_{d-1}} w_{c,c'}^d * F_c^{d-1}\right) \qquad c \in \{1, \ldots, W_d\}, d \in \{1, \ldots, D\}$$

where F_c^d denotes the cth channel of the dth layer output, b_c^d is the 2-dimensional bias for the cth channel of the dth layer, and $\{w_{c,c'}\}_{c=1}^{W_{d-1}}$ are $K \times K$ convolutional kernels that convolve with each of the W_{d-1} channels of F^{d-1}, the output of the previous layer. The input to each convolution

layer is padded by $\frac{K-1}{2}$ pixels to prevent shrinkage in the spatial direction. The ReLU nonlinearity $\rho(x) = \max(0, x)$ is applied element-wise to its input. The $(D+1)$th layer upsamples F^D to obtain W_D feature channels at full spatial resolution by

$$F_c^{D+1}(x, y) = F_{c^*}^D(\lfloor x/2 \rfloor, \lfloor y/2 \rfloor),$$

where $c^* = 4c - 3 + (x \bmod 2) + 2(y \bmod 2)$, giving the inverse operation to compression called *pixel shuffle* in [56]. In addition, identity-forwarding channels obtained by masking M to extract separate color channels are concatenated to F^{D+1} by

$$F_c^{D+1}(x, y) = M(x, y) m_{c-W_{D+1}}(x, y); \quad c \in \{W_{D+1} + 1, \ldots W_{D+1} + 3\}$$

with m_1, m_2, m_3 binary matrices with 1s at the Bayer grid patterns for the red, green, and blue measurements of M and 0s elsewhere, respectively. A final convolutional layer interpolates the learned residual channels with the channel-wise identity mapping from M,

$$F_c^{D+2} = \rho\left(b_c^{D+2} + \sum_{c'=1}^{W_{D+2}} w_{c,c'}^{D+2} * F_{c'}^{D-1}\right), \quad c \in \{1, \ldots, W_{D+2}\}$$

and the output F is given by a learned affine combination of the last W_{D+2} feature channels:

$$F_c = b_c^F + \sum_{c'=1}^{W_{D+2}} w_{c,c'}^{D+2} * F_{c'}^{D-1}, \quad c \in \{1, \ldots, W_{D+2}\}$$

The hyperparameters are set as $D = 15, K = 3$, and

$$W_i = \begin{cases} 5 & i = 0 \\ 12 & i = D \\ 64 & \text{otherwise.} \end{cases}$$

The convolutional kernels $\{w^d_{c,c'}\}_{d,c,c'}$, affine weights $\{a_{c,c'}\}$, and biases $\{b^d_c\}_{c,d}$ are learned by using Adam to optimize a normalized L_2 loss between ground truth 128×128 pixel sRGB patches X_i and the demosaiced output $F(M_i)$ of the network on Bayerized patches M_i. Patches are artificially mosaiced in sRGB space and recovered in sRGB space by the network. A custom training dataset was created wherein images that earlier iterations of the demosaicing network struggled to interpolated correctly were gathered to create a dataset of difficult patches. The patches were extracted from images belonging to the ImageNet and MirFlickr datasets. At the time of its publication in 2016, [19] achieved state of the art performance for single-image demosaicing and denoising in sRGB space, and was competitive at the joint problem in linear space without any fine-tuning of network parameters despite having been trained on sRGB images only.

Gharbi's network demonstrated the power of applying deep convolutional networks to the low-level image processing problem of demosaicing. Significantly, the network architecture of [19] is not designed with any explicit model for the recovery of mosaiced data, but rather relies on experimental observations and tinkering to fine-tune the hyperparameters of an essentially straightforwardly convolutional model. This may be described as a *deep learning heuristic* approach, since the domain-specific knowledge applied to the problem is fundamentally knowledge about the design of convolutional neural network rather than knowledge about the demosaicing problem.

There are several examples of other such deep learning heuristic approaches in the literature. Syu et al. [59] proposose DMCNN-VD, a 20 layer deep convolutional network which uses as input a 3-channel, uncompressed, mosaiced image obtained by multiplying the input channels of an sRGB image by the mask matrices m_1, m_2, m_3 described above. Unlike [19], [59] convolves entirely in the full resolution reconstruction space. The output of the last convolutional layer is added to a bilinear interpolation of the mosaiced input fast-forwarded past the interior convolutions. The SELU nonlinearity [31] is used between convolutional layers,

$$selu(x) = \begin{cases} \lambda x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$$

with $\lambda \approx 1.05, \alpha \approx 1.67$ as in [31]. All convolutional layers in DMCNN-VD consist of 64 3×3 kernels except the last, which computes the additive residual with 3 filter banks of 3×3 kernels. The authors also propose the Flickr500 dataset consisting of high-quality, high-frequency sRGB images without the visual artifacts of some earlier datasets. Mosaicing and interpolation are performed exclusively in sRGB space, and experiments indicate generalization to linRGB demosaicing using the MSR dataset [29].

Tan et al. [60] use the higher sampling frequency of the green channel to design their convolutional network architecture. The network uses a two-stage process: first, an intermediate green channel is residually interpolated from the input using a convolutional network; then all channels are residually interpolated from the input using a second convolutional network that takes the intermediate channels as input. Instead of using mosaiced images as input, the network operates directly on the full-resolution bilinear interpolation of the mosaic. To promote the accurate reconstruction of the intermediate green channel, the network is trained using the sum of an L_2 loss comparing the

intermediate green channel to the ground truth as well as an L_2 loss comparing the final output to the ground truth patch. This architecture can be seen as a deep convolutional version of the hue-based methods of [1, 10], although the hue is formulated here as an additive difference in the reconstruction space instead of a ratio or logarithmic difference.

Qian et al. [51] present a joint solution for demosaicing, denoising, and super-resolution using a convolutional network. This network is distinctive for being built out of so-called *residual-in-residual dense blocks* (RRDBs), first introduced in [64] for single-image super-resolution. We will defer the discussion of RRDBs to Chapter 6.

These deep learning heuristic framework for demosaicing, albeit effective as in [19], ultimately does not exploit the specific structure of the demosaicing problem. Indeed, the fundamental connection to the demosaicing problem in [19] inheres in the training dataset instead of the design of the convolutional network. The proposed architecture does not depend on an analytical model of the demosaicing problem.

Section 5.5

Demosaicing as an inverse problem

We may interpret demosaicing a noisy image as an inverse problem. Given the mosaiced image $y \in Y_{bayer}$, we wish to recover the true image $x \in X_{img}$ where

$$y = Mx + n, \tag{5.1}$$

$M : X_{img} \to Y_{bayer}$ is the measurement operator corresponding to a Bayer degradation of the "true" image x and $n \in Y_{bayer}$ is noise. It is worth noting that M is hopelessly singular – there is no

way to distinguish images x_1, x_2 that agree on the Bayer grid but do not agree off of it once they have been corrupted by M. This strongly suggests that a prior on the reconstruction space will be necessary for recovery. The additional corruption by noise n further complicates recovery. Finally, it is worth noting that for camera pipeline that requires demosaicing, x represents the hypothetical quantity of the color measurement that would have been obtained were the number of photons in each of the red, green, and blue sensitivity bandwidths accurately counted at *every* pixel of the CCD grid, instead of only those pixels corresponding to the Bayer CFA pattern overlayed thereon. Such light *did* did actually enter the camera, but was not observed on account of the CFA grid. As such, realistic "ground truth" images $x \in M_{img}$ are difficult to obtain and often artificial data such as that created by applying a 0-1 mask to an already-demosaiced image are used for training.

Kokkinos and Lefkimmiatis [32] design a network for joint denoising and demosaicing inspired by classical regularization techniques for solving inverse problems. Their core approach is to analytically reduce demosaicing to a denoising problem which can be iteratively solved by a trained CNN. We will now examine this approach in detail.

The representation of images as unravelled one-dimensional vectors is valid and more convenient for the present approach. For [32], the ground truth space X_{img} is $\mathbb{R}^N$. The measurement operator M is then a binary diagonal $N \times N$ matrix corresponding to the Bayer CFA pattern. We then have $Y_{bayer} = Im(M) \subset \mathbb{R}^N$, and thus $n = Mn_0$ where n_0 is N-dimensional noise. We have also that M is symmetric and orthogonal, and that M is the identity in Bayer space: $M|_{Y_{bayer}} = I|_{Y_{bayer}}$.

[32] assumes that the observed image is corrupted by additive, zero-mean, Gaussian noise $n \sim \mathcal{N}(0, \sigma^2)$ in Y_{bayer}. Using the Bayesian maximum a posteriori (MAP) approach, [32] seeks x^* such that

$$x^* = \arg\max_x \log(p(x|y))$$

$$
\begin{aligned}
&= \arg\max_x \left[\log(p(y|x)) + \log(p(x))\right] \\
&= \arg\max_x \left[-\frac{1}{2\sigma^2}\|y - Mx\|_2^2 + \log(p(x))\right] \\
&= \arg\min_x \underbrace{\left[\frac{1}{2\sigma^2}\|y - Mx\|_2^2 + \phi(x)\right]}_{:=Q(x)}
\end{aligned} \tag{5.2}
$$

where $\log(p(y|x)) = -\frac{1}{2\sigma^2}\|y - Mx\|_2^2$ follows from the assumption on the distribution of the noise n, and $\phi(x)$ is a regularization term representing the negative log prior of observing x. It is important to observe that the regularization term is not given explicitly. Instead, an optimal regularization will be learned from data.

From (5.2), a majorization-minimization approach (see Section 2.5) is employed such that the choice of majorizer permits a decoupling of the degradation operator M from x. The data-fidelity term $\frac{1}{2\sigma^2}\|y - Mx\|_2^2$ is selectively majorized by

$$
f(x, x_0) = \frac{1}{2\sigma^2}\left[\|y - Mx\|_2^2 + (x - x_0)^T[I - M^T M](x - x_0)\right]
$$

since $M^T M = M$ is a binary diagonal matrix, hence $I - M^T M$ is positive semidefinite, whence $f(x, x_0) \geq \frac{1}{2\sigma^2}\|y - Mx\|_2^2$ for all x, x_0 with equality at $x = x_0$. Following [32], we observe that

$$
\begin{aligned}
&\|y - Mx\|_2^2 + (x - x_0)^T[I - M^T M](x - x_0) \\
&= (y - Mx)^T(y - Mx) + (x - x_0)^T(x - x_0) - (x - x_0)^T M^T M(x - x_0) \\
&= y^T y - 2y^T Mx + (Mx)^T Mx + x^T x - 2x_0^T x + x_0^T x_0 - x^T M^T Mx + 2x^T Mx_0 - x_0^T Mx_0 \\
&= y^T y - 2y^T Mx + x^T Mx + x^T x - 2x_0^T x + x_0^T x_0 - x^T Mx + 2x^T Mx_0 - x_0^T Mx_0 \\
&= y^T y - 2y^T Mx + x^T x - 2x_0^T x + x_0^T x_0 + 2x^T Mx_0 - x_0^T Mx_0,
\end{aligned} \tag{5.3}
$$

showing that using f to majorize the data-fidelity term effectively decouples M from x. By introducing $z = y + (I - M)x_0$ we may collect terms from (5.3) and rewrite

$$f(x, x_0) = \frac{1}{2\sigma^2}\|x - z\|_2^2 + C(y, x_0)$$

giving the majorizer F of Q,

$$F(x, x^{(t)}) = \frac{1}{2\sigma^2}\|x - z\|_2^2 + \phi(x) + C(y, x^{(t)}). \tag{5.4}$$

with associated iterative algorithm

$$x^{(t+1)} = \arg\min_x F(x, x^{(t)}) = \arg\min_x \frac{1}{2\sigma^2}\|x - z\|_2^2 + \phi(x) \tag{5.5}$$

Note that $z = y + (I - M)x^{(t)}$ combines the observed, mosaiced information from y with the reconstructed, unobserved measurements from $x^{(t)}$. In the case of perfect reconstruction $x^{(t)} = x$, we have $z = Mx + n + (I - M)x = x + n$, the ground truth image corrupted with noise supported on the CFA grid. Indeed, as noted in [32], (5.5) is the objective function of a denoising problem with noisy measurement z and ground truth x.

Thus each iteration step for the proposed majorization-minimization demosaicing algorithm reduces to the application of a denoising network to the term $z^{(t)} = y + (I - M)x^{(t)}$. The authors of [32] propose ResDNet, a residual convolutional denoising network first proposed in [40]. ResDNet takes the additional parameters σ, an estimate of the variance of the original noise corrupting the data, and γ_i a trainable parameter used to modify the computed residual noise to have the correct variance for the current iteration.

Algorithm 2: Forward Pass of MMNet for Iterative Joint Demosaicing and Denoising, [32]

Data: M: Bayer operator, y: input, K: number of iterations, $w \in \mathbb{R}^K$: extrapolation weights, σ: variance of original noise, $\gamma \in \mathbb{R}^K$: projection parameters

$x^{(0)} = 0$

$x^{(1)} = y - (I - M)x^{(0)}$

for $i \leftarrow 1$ **to** K **do**

  $u = x^{(i)} + w_i(x^{(i)} - x^{(i-1)})$

  $x^{(i+1)} = \text{ResDNet}((I - M)u + y, \sigma, \gamma_i)$

end

Algorithm 2 defines MMNet, a *recurrent neural network* that iteratively applies the ResDNet architecture to increasingly fine-tuned demosaicing estimates. There are not K different copies of ResDNet, but instead a single set of parameters that are shared across all iterations. This significantly increases the memory demands of computing the gradient of the loss with respect to the parameters in proportion to K. This situation can be remedied by use of Truncated Back-propagation Through Time (TBPTT) [53], which segments the K iterations into smaller stages of k iterations. Parameters are then updated at the end of each kth iteration during training, which frees up the memory used in backpropagation to allow for more total iterations K. ResDNet is pre-trained using an L_2 loss on a denoising dataset, and then fine-tuned with TBPTT for an average L_1 loss on a noisy mosaiced dataset. Using $K = 2$, MMNet outperforms [19] on the MSR demosaicing test set for noisy images in both linRGB and sRGB color spaces, with even better performance with $K = 10$ and $K = 20$ [32].

Section 5.6

Super-resolution

Related to demosaicing is the task of digital image super-resolution (SR). The single-image super-resolution task, given a low-resolution image X_{LR}, is to produce a realistic higher-resolution image X_{HR} at some fixed higher multiple of the original resolution, such as two, four, or eight[3]. As with demosaicing, spline interpolation methods like bilinear or bicubic interpolation are capable of providing a rough start, but produce undesirable artifacts in the high-resolution images in proportion to the resolution factor.

Super-resolution is a similar task to demosaicing in that both problems require the accurate upsampling of light measurements in a digital image. If we consider the compressed version of a Bayer image as in Section 5.4, the comparison becomes even plainer: the 4 color channels of a compressed mosaiced image must be used to produce the 3 color channels of a demosaiced image with twice the spatial resolution. The corresponding 2xSR problem must produce the 3 color channels of the super-resolved image from 3 original low-resolution color channels. In this way, the 2xSR problem uses fewer input measurements than the demosaicing problem. It is not the case, however, that demosaicing is trivially reducible to 2xSR because the geometry of the compressed Bayer input is fundamentally different than that of a low-resolution input to SR. The color channels in a compressed Bayer image are not aligned with each other in a strict sense – they record values of light corresponding to physically different spots on the digital camera CCD array. The similarity of the two problems remains high enough, though, that techniques from the SR literature can be brought to bear on the demosaicing problem.

[3]This multiple applies in both height and width dimensions; thus a 2x super-resolved image will contain 4 times as many pixels as the original

Some early work on super-resolution treats the multiple image super-resolution (MISR) problem instead of the single-image super-resolution (SISR) described above. In this problem, a set of low-resolution images $X_1, \ldots, X_N$ are observed and a single super-resolved image $X_{N,HR}$ is desired. Irani and Peleg [27] give an iterative algorithm for MISR inspired by back-projection in computer tomography. It is assumed that the observed images differ from each other by translations and rotations, and thus the motion warps can be easily parametrized and estimated. After an initial approximation $X_N^{(0)}$ of the high-resolution image is made, each step k of the algorithm simulates the forward operation by using an image formation model along with the estimated warps to obtain a set of approximated low-resolution images $X_1^{(k)}, \ldots, X_N^{(k)}$. These approximate images are compared to the true low-resolution input set, and the differences are back-projected to the super-resolution domain. The next iteration $X_N^{(k+1)}$ is obtained by correcting $X_N^{(k)}$ with these back-projected residuals. In total, this scheme performs one initial upsampling back-projection followed by K iterations of a downsampling forward operation followed by an upsampling back-projection.

This iterative back-projection scheme for MISR has been replicated in deep neural network architectures for SR. Haris et al. [22] propose the Deep Back-Projection Network (DBPN), a CNN in the style of [27]. The $2K$ forward operator and back-projections of the K iterations of [27] are replaced by $2K$ convolutional modules with learned weights. Each back-projection module computes residual high-resolution features from a low-resolution residual input; these features give increasingly fine details as iteration depth increases. The downsampling modules reduce the high-resolution residual of the previous step to a low-resolution quantity used as input to the next back-projection module. The network outputs a learned affine combination of the K high-resolution residual maps. A dense variant is also proposed where all previous high- or low-resolution residuals are combined as an input to the following module. We study the application of this scheme to spatial demosaicing in Chapter 6.

Haris et al. [23] also propose the Recurrent Back-Projection Network (RBPN) for video super-resolution (VSR). Video SR is a specific case of MISR where the observed frames are assumed to belong to the same video sequence with consistent motion between the frames. Hence the ordering of the frames is usable information, unlike in MISR when only the set of observations itself matters. For a sequence $\{X_0, \ldots, X_N\}$ of low-resolution video frames, RBPN will compute N high-resolution residual feature maps that are combined in a final convolution to produce the super-resolved output. The low resolution features L_0 are initialized by convolving the low-resolution input with a convolutional filter bank ϕ_0. Each residual map is computed by the following:

$$\begin{aligned} M_k &= \text{FeatureExtraction}(X_{k-1}, X_N) \\ H_k &= \text{Encoder}(L_{k-1}, M_k) \\ L_k &= \text{Decoder}(H_k) \end{aligned}$$

where FeatureExtraction is a convolutional layer to extract features from the low-resolution frames and Encoder is defined by

$$\begin{aligned} H_{k-1}^l &= \text{SISR}(L_{k-1}) \\ E_k &= \text{ResidualFeatures}(H_{k-1}^l - \text{MISR}(M_k)) \\ \text{Encoder}(L_{k-1}, M_k) &= \text{SISR}(L_{k-1}) + \text{ResidualFeatures}(E_k). \end{aligned}$$

Here, SISR is an implementation of DBPN, while MISR, ResidualFeatures, and Decoder are based off of the ResNet architecture [25]. Since RBPN is a recurrent neural network, the parameters of the encoder and decoder modules are shared between all N passes. We will adapt this architecture to the video demosaicing problem in Chapter 6.

Section 5.7

Temporal demosaicing

The *temporal* or *multi-frame demosaicing* problem generalizes the above single-frame demosaicing problem. The single-frame demosaicing problem concerns the recovery of information from one Bayerized image. In other contexts, though, more information can be brought to bear on the problem. Consider the demosaicing problem for a sequence of images $\{x_t\}$, as in video or burst photography applications. We observe the sequence of mosaiced frames $\{y_t\}$, and wish to produce a close approximation $\{\hat{x}_t\}$ of the original sequence. For a fixed t_0, the neighboring observed frames y_{t_0-1} and y_{t_0+1} contain color information which may, in principle, be used to improve the quality of our estimate $\hat{x}_t$.

The literature on temporal demosaicing is considerably sparser than the single-frame problem. Farsiu et al. [15] offers an inverse problem approach to multiframe demosaicing with explicit regularization terms. Kokkinos and Lefkimmiatis [33] generalizes their majorization-minimization approach to single-frame demosaicing to burst photography sequences of mosaiced data, where frame registration is accomplished with orthogonal transformations. Ehret et al. [14] use a network inspired heavily by [19] and assume that image registration can be accomplished by affine transformations of coordinates. Interestingly, [14] demonstrates how a demosaicing network can be trained in an unsupervised fashion using an aligned burst of images. However, no existing approach in the literature has considered demosaicing frame sequences that differ by non-affine warps. We examine such a network for demosaicing arbitrarily warped frame sequences in Chapter 6. This methodology depends on the estimation of optical flow (see Section 3.5), which has recently received attention from CNN researchers. We also will consider adapting the RBPN of [23] for the video demosaicing

problem. As shown in [23], this scheme also benefits from the explicit estimation of a dense optical flow.

Subsection 5.7.1

ProxNet

To model the forward problem for the capture of a sequence of B frames with the goal of demosaicing the Bth frame, [33] uses the model

$$Y_i = MS_i(X) + n; \qquad i = 1, \ldots, B \tag{5.6}$$

with the new operator S_i corresponding to the spatial transformation that registers frame K to frame i. Kokkinos and Lefkimmiatis [33] assume that this spatial transformation can be represented by a composition of translation and rotation. Supposing that we have calculated all the warping transformations S_i, the objective function for maximum a posteriori reconstruction can then be generalized from the single frame problem to compare the reconstruction $\hat{X}$ to all B mosaiced observed frames.

$$X^* = \arg\min_X \left[\frac{1}{2B\sigma^2} \sum_{i=1}^{B} \|MS_i(X) - Y_i\|_2^2 + \phi(X) \right] \tag{5.7}$$

We are interested in a proximal method for minimizing this objective function, so we wish to obtain derivatives of the data-fidelity summation in (5.7). As noted in [33], the derivative of the warping transformation can be linearized in the discrete implementation space of digital images. We recall from Section 3.5 that optical flow ν can be applied to images sampled on the discrete grid by defining the warping transformation $W^{-1} = (I + \nu)$. The operation S_i in the forward model (5.6) corresponds to resampling the ground truth pixels X using the warp W_i^{-1} obtained from the optical

flow from $X = X_K$ to the earlier scene X_i. This optical flow can be approximated by using a flow prediction algorithm on a cheap interpolation of the mosaiced inputs Y_K, Y_i. Having obtained such a flow, the operator $S_i(X)$ reduces to that matrix multiplication on X giving the resampling coefficients of the interpolated warp, $S_i(X) = \rho_i X$. Thus $\nabla(S_i(X)) = \rho_i^T$, and hence

$$\nabla_X \left(\frac{1}{2B\sigma^2} \sum_{i=1}^{B} \|MS_i(X) - Y_i\|_2^2 \right) = \frac{1}{B\sigma^2} \sum_{i=1}^{B} \rho_i^T M^T (M\rho_i X - Y_i)$$

We observe that this equation is a half-truth. The computed warps S_i themselves should depend upon our estimate of X, but by linearizing the warping operation we conveniently ignore this dependence. In reality, recomputation of the B warps for every iteration on $X^{(t)}$ is a costly operation that may be impractical. However, future work may investigate joint iterative flow refinement and demosaicing in the vein of [68].

With the gradient of the data-fidelity term in hand, the proximal gradient method for the multi-frame demosaicing problem is

$$x^{(t+1)} = \underset{\sigma^2\phi}{\text{prox}} \left(x^{(t)} + \frac{1}{B} \sum_{i=1}^{B} \rho_i^T M^T (Y_i - M\rho_i X^{(t)}) \right).$$

Supposing that we have a network ProxNet trained for the proximal minimization task, the architecture of [33] for burst demosaicing is given by Algorithm 3. The design of ProxNet itself, a residual architecture for data denoising, is discussed in [32].

Algorithm 3: BurstDemosaicNet [33]

Data: $\{Y_i\}_{i=1}^{B}$ Bayer input frames, K: number of iterations, $\sqsupset$: bilinear interpolation kernel, ProxNet: network to compute proximal operator, $w \in \mathbb{R}^K$: extrapolation weights

$X_0 \leftarrow 0$
$X_1 \leftarrow Y_B$
for $i \leftarrow 1$ **to** $B-1$ **do**
  $\nu_i \leftarrow \text{ComputeAffineWarp}(\sqsupset * Y_B, \sqsupset * Y_i)$
  $\rho_i \leftarrow \text{ComputeResamplingMatrix}(I + \nu_i)$
end
for $t \leftarrow 1$ **to** K **do**
  $u \leftarrow x^{(t)} + w_t(x^{(t)} - x^{(t-1)})$
  $z \leftarrow \frac{1}{B}\sum_{i=1}^{B} \rho_i^T M^T (M \rho_i u)$
  $x^{(t+1)} = \text{ProxNet}(x^{(t)} - z)$
end
return $x^{(K+1)}$

Section 5.8

Deep optical flow estimation

As discussed in Chapter 3, optical flow is a methodology for describing motion in digital video. In particular, an optical flow estimate can be used to align a video sequence by resampling frames according to the computed flow. Although optical flow is subject to many limitations at the theoretical level (lighting changes, scene occlusions, and large displacements are difficult to handle), for simplified scenes it is possible to obtain good estimates of optical flow.

Horn and Schunck [26] introduced a variational approach for computing optical flow using a system of PDEs the solution of which would give the scene optical flow. There is a large literature of subsequent optical flow approaches which we will not review here, instead skipping to recent developments in learning optical flow using deep neural networks.

The seminal deep learning optical flow network was Flownet [16], a deep convolutional neural network for estimating optical flow. A proliferation of work soon followed, including PWC-Net [57].

PWC-Net has proven to be a popular architectural approach to the problem of learning optical flow. The basic approach of PWC-Net is to estimate flow using a multi-scale pyramid of learned features from a pair of input frames. This approach has precedent in the classical literature. A pair of input frames are downsampled and convolved with learned filters to create a multi-scale pyramid of the image features, each at half the resolution of the last. Starting from the lowest-resolution level, an optical flow is successively computed, upsampled, and used as input to the next lowest resolution level. Using two input frames F_1^l, F_2^l, computation of flow at resolution level l first warps $F_{2,l}$ towards $F_{1,l}$ using the (bilinearly upsampled) flow from the previous level.

$$F_w^l(x) = F_2^l(x + up_2(\nu^{l+1})(x))$$

Using the warped features, a *cost volume* is computed. A cost volume is used to represent the "cost" of associating feature-pixels in the warped second image with those in the first.

$$cv^l(x_1, x_2) = \frac{1}{N_l}(F_1^l(x_1))^T F_w^l(x_2)$$

where N_l is the number of features in level l of the feature pyramid. It is computationally expensive to compute cv^l for every pair of feature pixels x_1, x_2, requiring $2^{-2l}H^2W^2N_l$ multiplications for each level of the feature pyramid. Therefore, PWC-Net computes a local cost volume such that $\|x_1 - x_2\|_\infty \leq d$. The effective reach of local cost volume cv^l with reach d is $2^l d$ pixels on the original frames, owing the upsampling of flow between levels of the pyramid. The resulting cost volume of the lth feature level has size $2^{-2l}HW(2d+1)^2$ for each pair of input frames in the training batch, eliminating the quadratic dependence on image size required by a full cost volume.

The lth level optical flow is learned by a sequence of convolutional layers with indpedendent weights from prior and subsequent feature levels. The inputs to the network are the features of

the first image, the computed local cost volume, and the upsampled flow from the previous level. From these the network outputs the estimated optical flow for the lth level. Finally, PWC-Net refines this flow using a context network inspired by earlier methods of estimating flow. The context network uses dilated convolutional layers to transform its inputs, the estimated flow along with a hidden layer of the flow estimation layer output, into its ouput, the refined optical flow for the lth level of features. The use of dilated convolutions in the context network enlarges the range of information available in computing each optical flow vector at a reduced computational cost compared to ordinary convolutions of the similar range.

The objective function of PWC-Net combines a multiscale L_2 loss on the accuracy of flow reconstruction on the different levels of the network with L_2 regularization on the network weights. This multiscale loss encourages the internal warping layers to perform the intended purpose of estimating an image alignment warp instead of learning some other, less intuitive mapping between input and optical flow output. The objective function is

$$L(\theta) = \left(\sum_{l=l_0}^{L} \alpha_l \sum_{x} \|\nu_\theta^l(x) - \downarrow_{2^l} \nu_{GT}^l(x)\|_2 \right) + \gamma \|\theta\|_2$$

where ν_θ^l denotes the learned flow at level l, ν_{GT}^l is the ground truth flow used for supervised learning, $\downarrow_{2^l}$ denotes the 2^l-times downsampling operator, and l_0 is the desired resolution level of the estimated flow. In [57], $l_0 = 2$ is used in experiments. This condition allows the network to be more lean than it would otherwise be if a full-resolution flow were to be learned. The resulting downsampled flow can then be scaled to full-resolution with a simple bilinear upsampling or some more sophisticated method.

Chapter 6

Numerical experiments with CNNs for spatial and temporal demosaicing

This chapter details the results of numerical experiments with different demosaicing CNN architectures. We define several architectures for spatial and temporal demosaicing along with training details and accuracy as evaluated on several demosaicing test datasets. We used popular deep learning frameworks TensorFlow and PyTorch for these experiments. Code for the experiments is available at https://github.com/montefischer/deep-demosaic.

We present the results of our numerical experiments with convolutional networks for the image and video demosaicing problems. We define several architectures for spatial and temporal demosaicing along with training details and accuracy as evaluated on several demosaicing test datasets. We evaluate identity-bypass networks, dense residual connection networks, and deep back-projection networks for the image demosaicing problem. For the video problem, we propose and evaluate MFD-Net, a network that registers several frames using an optical flow and exploits the temporal information therein. We also adapt a recurrent back-projection network architecture originally proposed in the super-resolution literature for the video demosaicing problem. We include both quantitative and qualitative comparison of demosaicing networks. We find that dense residual connections are advantageous to single image demosaicing networks, that MFD-Net suffers from a learning problem but demonstrates promising performance over single image methods, and that deep recurrent back-projection networks effectively and efficiently exploit temporal context for demosaicing video sequences.

Section 6.1

Datasets for demosaicing

We review the existing datasets used to train and evaluate demosaicing models. As discussed in 4.3, any CNN model for a supervised learning problem requires a *training set* of pairs of ground truth and inputs. In the case of demosaicing, it is difficult to obtain accurate ground truth information corresponding to a mosaiced input. This is precisely because two thirds of the ground truth measurements are not simply not recorded by a digital camera using the Bayer CFA (see Section 5.3). To circumvent this issue several proposed datasets for demosaicing remove pixels from already-demosaiced sRGB images according to the Bayer CFA pattern. Gharbi et al. [19] introduced a large dataset of images selected for the difficulty they posed to demosaicing methods. The original images were all in sRGB space and some were corrupted by JPEG artifacts. To correct for this, the images included in the dataset are downsampled by a factor of 4 using a bicubic kernel. The large size of this dataset greatly increases the training time for CNN demosaicing architectures, and other proposed datasets are much smaller. The Flickr500 dataset of Syu et al. [59] consists of 500 high-frequency sRGB images. The popular demosaicing test sets, Kodak PhotoCD [18] and McMaster [71] also use the artificially-mosaiced sRGB image scheme. We will use these two datasets for evaluating our own experiments in spatial demosaicing.

Khashabi et al. [29] propose a more realistic training dataset that better models the actual conditions under which demosaicing is performed. As discussed in Sections 5.2 and 5.3, the demosaicing operation is performed on raw images of linRGB measurements instead of gamma-adjusted, color-corrected images in sRGB space. As such, the approaches in the academic literature that propose to train and evaluate demosaicing algorithms using artificially mosaiced sRGB images are solving an artificial problem. To remedy this, the authors propose a novel downsampling technique

that is performed on images converted to linRGB space [29]. The resulting dataset is the Microsoft Research Demosaicing Dataset, or MSR for short. The MSR dataset consists of both a training set and a test set of images. However, it is not an obvious choice to use the MSR training dataset for deep demosaicing architectures. Gharbi et al. [19] find that a network trained only on sRGB examples actually outperforms existing methods on the MSR test set without further training, and that training instead on the MSR training set does not improve demosaicing quality. Syu et al. [59] give a similar result, exhibiting a network trained on artificially-mosaiced sRGB examples that nonetheless is capable of outperforming linRGB-trained methods on the MSR linRGB testing set. In light of these results, we use the Flickr500 sRGB dataset for training the single-image demosaicing models in Section 6.2 because of its rich variety of high-frequency, difficult-to-demosaic patches.

We note that Qian et al. [51] recently proposed the demosaicing datasets PixelShift200 and PixelShiftTest which use a Sony camera equipped with pixel shift hardware to create "perfect" demosaicing ground truth. The pixel shift camera physically shifts its sensor grid by single-pixel offsets to take four mosaiced images of a static scene. By combining these images, the camera eliminates the need for demosaicing altogether. Any motion present in such a scene would create artifacts in the merged image, so the dataset includes only high-resolution static scenes. We do not propose to evaluate this dataset in the present work, but present it as an interesting new advance in demosaicing dataset creation.

To the best of our knowledge, there are no currently published datasets specifically intended for demosaicing video sequences of frames. Previous works addressing multi-frame demosaicing [33, 14] assume that the multiple frames differ from each other by at most rotations and translations. Although this may be a good approximation for bursts of images obtained on modern digital cameras, it is a very bad assumption for arbitrary video sequences that may include complex motion. Short of creating a novel training and test dataset, we are left with the option of using artificially

mosaiced frame sequences from datasets for video processing (denoising, super-resolution, etc.). In particular, the recent Vimeo-90k dataset proposed in [68] contains many seven-frame video sequences obtained from the Internet video sharing website Vimeo. This is far from an ideal dataset for video demosaicing as many sequences have been processed by video editing software and some contain added graphics or other video effects. Worse, visual inspection of the dataset reveals the presence of significant JPEG block artifacts which make any network trained on the dataset impractical to use on real-world data. Additionally, many of the sequences prove to be easier to demosaic than is desirable for training a network to perform well on edge cases. Nonetheless, the dataset has been used to some success in video super-resolution and other tasks [68, 23], and we will use it to evaluate our own proposed architectures. The full dataset is very large and would greatly increase the length of each epoch; as a compromise between training time and training set variety we select a 500-sequence subset of the full dataset for training, a 100-sequence subset of different sequences for validation, and a final different 200-sequence subset for testing. Details of which sequences we selected are available at https://github.com/montefischer/deep-demosaic.

To measure demosaicing quality, we use the peak-signal to noise ratio (PSNR) as is well-represented in the demosaicing literature. The PSNR (in decibels) of a one-channel $N \times M$ demosaiced image y compared to the ground truth image x is computed by

$$MSE(y, x) = \frac{1}{NM} \|y - x\|_2^2$$
$$PSNR(y, x) = 10 \log_{10} \left(\frac{1}{MSE(y, x)} \right)$$

assuming that pixel intensities are represented in the range $[0, 1]$ [8]. For images with three color channels, we compute the PSNR by averaging the separate color channel PSNRs. All our reported quantitative performance results are measured in dB PSNR, averaged across color channels.

Section 6.2

Single-frame demosaicing models

A wide range of single-frame (spatial) demosaicing models have been proposed can be described as *deep learning heuristic demosaicing*, as discussed in Section 5.4. We propose to independently implement and evaluate a number of these architecture types using a standard dedication of training time, training data, and optimization algorithms. We evaluate the residual-in-residual dense block CNN module, as originally proposed for use in single-image super-resolution, for use in demosaicing networks.

As [32] demonstrates, recurrent architectures for single-image demosaicing inspired by proximal methods in the inverse problem literature can significantly improve on heuristic-based methods in both accuracy and computational complexity. Following this, we evaluate the performance of the network [32] for single-frame demosaicing when trained under a similar computational budget as the other proposed heuristic methods.

Subsection 6.2.1

Identity bypass convolutional demosaicing

Following [19], we implement and train a CNN for learning the residual between input and an identity skip connection from the mosaiced input image, performing our convolutions in the (compressed) measurement space with a final decompression to obtain the residual. We intend this network to serve as a basis for comparison with the other experiments herein, since the exact results of [19] employs a different training dataset, optimization method, and length of training time. The network is defined in Algorithm 4. Note that we use the $\circ$ operator to represent channel-wise concatenation,

the $\downdownarrows$ operator to represent the Bayer compression operation, and the $\upuparrows$ operator to represent the Bayer decompression operation (see Section 5.4). Here and henceforth we omit the added bias terms in algorithms for simplicity.

Algorithm 4: Identity Bypass Demosaicing CNN

Data: X: Bayer input, $\{\Phi_k\}_{k=1}^{D}$: convolutional filter banks, ρ: nonlinearity
$X_0 \leftarrow \downdownarrows (X)$
for $i \leftarrow 1$ **to** $D-2$ **do**
  $X_i \leftarrow \rho(\Phi_i * X_{i-1})$;
end
$R = \upuparrows X_{D-2}$
$X_{D-1} \leftarrow \rho(\Phi_{D-1} * (R \circ X))$
return $\Phi_D * X_{D-1}$

Instead of a identity mapping from the channels of the input image to the final reconstruction, we also examine the use of a bilinear interpolation against which residual channel information is learned. This technique has been employed in previous demosaicing CNN architectures [59]. Our bilinear residual network is defined in Algorithm 5. We use the convolutional operator $\sqsupset$ to denote bilinear interpolation of an uncompressed Bayer mosaic.

Algorithm 5: Bilinear Bypass Demosaicing CNN

Data: X: Bayer input, $\{\Phi_k\}_{k=1}^{D}$: convolutional filter banks, ρ: nonlinearity
$X_0 \leftarrow \downdownarrows (X)$
for $i \leftarrow 1$ **to** $D-2$ **do**
  $X_i \leftarrow \rho(\Phi_i * X_{i-1})$
end
$R = \upuparrows X_{D-2}$
$X_{bilin} \leftarrow \sqsupset * X$
$X_{D-1} \leftarrow \rho(\Phi_{D-1} * (R \circ X_{bilin}))$
return $\Phi_D * X_{D-1}$

We compare the results of training these networks on the Flickr500 training dataset using data augmentation regularization by extracting, at each epoch, a random square 64×64 pixel patche from each image and applying a random symmetry of D_8 to each patch. We initialize all network

Table 6.1: CNN Demosaicing with a single residual connection. Networks trained with a learning rate scheduler that automatically decreased learning rate upon a stall in loss function improvement for 20 epochs are labeled with "sch." in training field.

Network	Epochs	Training	Kodak	McMaster
IdentityBypass	1000	L_2, AMSgrad@1e-3; sch.	38.139	36.096
IdentityBypass	2000	L_2, Adam@1e-4	35.332	33.756
BilinearBypass	1000	L_2, AMSgrad@1e-3; sch.	37.497	36.125
BilinearBypass	2000	L_2, Adam@1e-4	39.845	37.780
BilinearBypass	3000	L_2, Adam@1e-4[1]	40.320	38.054

parameters as in [24]. The networks we define contain 493,071 trainable parameters each. We optimize with Adam [30] or AMSGrad [52]. For some training procedures, we use a learning rate scheduler that decreases the learning rate by a factor of 2 when there have been no improvements in the loss in 20 epochs. These results are presented in Table 6.1.

Subsection 6.2.2

Residual-in-residual dense blocks for demosaicing

Wang et al. [64] introduced the *residual-in-residual dense block* (RRDB) for applications in super-resolution. The parallels between demosaicing and super-resolution (most obviously that, for the green channel, demosaicing is exactly a 2x super-resolution task) inspired [51] to apply RRDB for the joint problem. However, in [51] a total of six RRDBs were used in succession leading. The authors do not elaborate on the total number of parameters used, but our own calculations suggest that this produces a network with approximately 6.5 million parameters, when earlier state-of-the-art networks do not exceed 600,000 parameters [19] and have used as few as 380,567 [32]. Here, we apply RRDB directly to the spatial demosaicing problem in a network we call RRDBNet, taking care to adjust the network architecture to be comparable with the size of earlier proposed networks.

A RRDB module consists of N *residual dense blocks* of depth K, defined in Algorithm 6, in consecutive order as specified in Algorithm 7. The dimensionality of the convolutional filter banks must agree with the structure implicit in the definition of the network architectures. Thus if the input X has s_0 channels, and convolution $\Phi_1 * X$ creates an s_1-channel output, then Φ_2 must define $s_0 + s_1$ filters for each of its s_2 output channels. Wang et al. [64] set the *exterior channel* count $s_0, s_K = 64$ and the *internal channel* count $s_i = 32$ otherwise.

Algorithm 6: Residual Dense Block (RDB)

Data: X: input signal, K: depth of RDB, $\{\Phi_k\}_{k=1}^{K}$: convolutional filter banks, ρ: nonlinearity, α: residual weight
$X_1 \leftarrow \rho(\Phi_1 * X)$
for $i \leftarrow 2$ **to** K **do**
  $Z_i \leftarrow X \circ X_1 \circ \cdots \circ X_{i-1}$
  $X_i \leftarrow \rho(\Phi_i * Z_i + \beta_i)$
end
return $\alpha \cdot X_K + X$

Algorithm 7: Residual-in-Residual Dense Block (RRDB)

Data: X: input signal, $\{RDB_j\}_{j=1}^{N}$: N RDB subcomponents, α: residual weight
$X_0 \leftarrow X$
for $j \leftarrow 1$ **to** N **do**
  $X_i = RDB_j(X_{i-1})$;
end
return $\alpha \cdot X_N + X$

We define RRDBNet in Algorithm 8 as the sequential application of M RRDBs to learn a residual input. As always, filter banks ϕ_1, ϕ_2 must match the input / output filter dimensionality requirements of the compressed 4-channel input X and the 12-channel input to the decompression operator $\uparrow\uparrow$.

We give the results of training RRDBNet variants under different optimization schemes in Table 6.2. For some networks we train using AMSGrad [54] optimization with an initial learning rate of 10^{-3} that was reduced by a factor of 2 when the training loss had ceased to improve for over 20

Algorithm 8: RRDBNet – Residual-in-Residual Dense Block Network for Spatial Demosaicing

Data: X Bayer input signal, $\{RRDB_h\}_{h=1}^{M}$: M RRDB modules, ϕ_1: initial convolutional filter bank, ϕ_2: final convolutional filter bank, $\sqsupset$: bilinear interpolation kernel

$X_0 \leftarrow \phi_1 * (\Downarrow X)$

for $h \leftarrow 1$ **to** M **do**

| $X_i = RRDB_h(X_{i-1})$;

end

return $\sqsupset * X + \Uparrow (\phi_2 * X_M)$

epochs of training. Numerical experiments suggest that this choice of learning rate scheduler is poor for use with the stochastic Flickr500 training dataset, as the scheduler had drastically decreased the learning rate by the 1000th epoch to 1e-8. Such a low learning rate is unlikely to generate any meaningful updates to the parameters of these particular networks, but by keeping a higher learning rate for more epochs superior performance can be obtained.

In the interest of comparison with the above bypass networks with almost four times fewer parameters as the RRDBNet defined in [64] ($M = 3, N = 3, K = 5$), we also report the result of training lower-capacity versions of RRDBNet. We label the RRDBNet defined by M, N, K, e exterior channels, and i internal channels as $\text{RRDBNet}_{M,N,K,e,i}$. We use $\alpha = 0.2$ for all networks.

Our results indicate that the RRDB architecture significantly outperforms bypass-style architectures at comparable network size and training time. Unsurprisingly, the full RRDBNet with over 2 million parameters demonstrates a higher capacity, but at a significant memory cost. Even when scaling RRDBNet down ($M = 1, N = 3, K = 5, e = 32, i = 32$) to have *fewer* parameters than the earlier bypass networks, improved performance can be empirically observed within the first 1000 epochs of training. Interestingly, although PSNR is computed from a squared-error loss, we find here that training using the L_1 loss substantially improves quantitative output quality compared to training with an L_2 loss under similar conditions. Comparing networks trained with an initial

Table 6.2: RRDBNet demosaicing with $M = 3, N = 3, K = 5$, 64 exterior channels, 32 internal channels. At the 1000th and the 2000th epochs, the learning rate is reduced by a factor of 10.

Network	Parameters	Epochs	Training	Kodak	McM
$\text{RRDBNet}_{3,3,5,64,32}$	2 167 564	1000	L_1, AMSgrad@1e-3; sch.	41.162	39.205
$\text{RRDBNet}_{3,3,5,64,32}$	2 167 564	1000	L_2, AMSgrad@1e-3; sch.	40.953	38.777
$\text{RRDBNet}_{3,3,5,64,32}$	2 167 564	1000	L_2, Adam@1e-4;	40.923	38.593
$\text{RRDBNet}_{3,3,5,64,32}$	2 167 564	2000	L_2, Adam@1e-4;	41.250	39.029
$\text{RRDBNet}_{1,3,5,32,32}$	419 852	1000	L_2, AMSgrad@1e-4	34.016	33.305
$\text{RRDBNet}_{1,3,5,32,32}$	419 852	1000	L_1, Adam@1e-3	40.736	38.444
$\text{RRDBNet}_{1,3,5,32,32}$	419 852	2000	L_1, Adam@1e-3	41.028	38.818
$\text{RRDBNet}^*_{1,3,5,32,32}$	419 852	20 700	L_1, Adam@1e-3	41.569	38.991
$\text{RRDBNet}_{1,3,5,32,32}$	419 852	1000	L_2, Adam@1e-3	40.000	38.089
$\text{RRDBNet}_{1,3,5,32,32}$	419 852	2000	L_2, Adam@1e-3	40.972	38.864
$\text{RRDBNet}_{1,3,4,64,32}$	535 084	1000	L_2, Adam@1e-4	39.514	36.699
$\text{RRDBNet}_{1,3,4,64,32}$	535 084	2000	L_2, Adam@1e-4	39.835	37.592

10^{-3} learning rate and networks training with a 10^{-4} initial learning rate, we see the harmful effect of choosing an initial learning rate that is too small. We also train RRDBNet* for a large number of epochs[2] with an L_1 loss to test the full capacity of the RRDBNet architecture. This network produces the highest-quality prediction results, even outperforming larger networks trained for fewer epochs.

Subsection 6.2.3

Deep back-projection demosaicing

Inspired by the super-resolution literature [22] discussed in Section 5.6, we investigate the suitablility of deep back-projection networks (DBPN) for single-frame demosaicing. The architecture of DBPN iteratively applies up- and down-projections from low-resolution feature space to high-resolution feature space. Unlike [32], the back-projections of DBPN do not produce actual 3-channel full-

[2]We reduce the learning rate for RRDBNet* by a factor of 10 after the 1000th epoch and the 2000th epoch.

Table 6.3: DBPN demosaicing. Every 1000 epochs after the first 1000, the learning rate is decreased by a factor of 10.

Network	Parameters	Epochs	Training	Kodak	McM
DBPN_2	494 919	2000	L_2, Adam@1e-4	39.818	38.167
DBPN_2	494 919	3000	L_2, Adam@1e-4	40.583	38.613
DBPN_2	494 919	4000	L_2, Adam@1e-4	40.539	38.655

resolution color images, but instead produce dozens of full-resolution feature maps. Only at the end of the forward pass does the DBPN architecture predict a demosaiced 3-channel image.

Algorithm 9: DBPN_T – Deep Back-Projection Network for Spatial Demosaicing

Data: Y: Bayer input signal, T: number of back-projection steps, $\{\text{UpModule}_i\}_{i=1}^{T}$: back-projection modules, $\{\text{DownModule}_i\}_{i=1}^{T-1}$: forward operator modules, ϕ_1, ϕ_2: initial convolutions, $\{\psi_j\}_{j=1}^{S}$: interpolating convolutions, ρ: nonlinearity

$Y_0 \leftarrow (\downdownarrows Y)$
$Y_1 \leftarrow \rho(\phi_1 * \rho(\phi_0 * Y_1))$
$Z = \text{UpModule}_1(Y_1)$
$H \leftarrow (\square * Y_0) \circ Z$
for $i \leftarrow 2$ **to** T **do**
  $Z \leftarrow \text{DownModule}_{i-1}(Z)$
  $Z \leftarrow \text{UpModule}_i(Z)$
  $H \leftarrow H \circ Z$
end
$X \leftarrow H$
for $j \leftarrow 1$ **to** S **do**
  $X \leftarrow \rho(\psi_j * X)$
end
return X

Subsection 6.2.4

Qualitative comparison of spatial demosaicing networks

For qualitative comparison of several of the proposed demosaicing methods, we include Figure 6.1. Here, we compare a ground truth (GT) 32×32 pixel patch extracted from a high-frequency

Algorithm 10: UpModule – Back-Projection Module for Spatial Demosaicing

Data: L: low-resolution input, $\phi_\uparrow, \psi_\uparrow$: upsampling convolutions, $\phi_\downarrow$: downsampling convolution

$H_0 \leftarrow \phi_\uparrow * L$
$L_0 \leftarrow \phi_\downarrow * H_0$
$E \leftarrow L_0 - L$
$H_1 \leftarrow \psi_\uparrow * E$
return $H_0 + H_1$

Algorithm 11: DownModule – Forward Operator Module for Spatial Demosaicing

Data: H: high-resolution input, $\phi_\downarrow, \psi_\downarrow$: downsampling convolutions, $\phi_\uparrow$: upsampling convolution

$L_0 \leftarrow \phi_\downarrow * H$
$H_0 \leftarrow \phi_\uparrow * L_0$
$E \leftarrow H_0 - H$
$L_1 \leftarrow \psi_\downarrow * E$
return $L_0 + L_1$

region of an image in either the Kodak or McMaster datasets to the performance of several of our proposed demosaicing networks on an artificially mosaiced version of the patch. The false color and zippering artifacts of bilinear demosaicing are clearly noticeable here. We also observe that bypass-style networks have difficulty demosaicing the high-frequency vertical stripes of the third row patch, compared to deep back-projection or residual-in-residual dense block architectures.

The specific networks shown in Figure 6.1 are as follows. "Bilinear" means simple bilinear demosaicing. "Id. Bypass" refers to the best performing identity bypass network of Table 6.1. "RRDB" denotes the best performing RRDBNet architecture network in Table 6.2, itself with an L_1 loss. "DBPN$_2$" denotes the third network in Table 6.3.

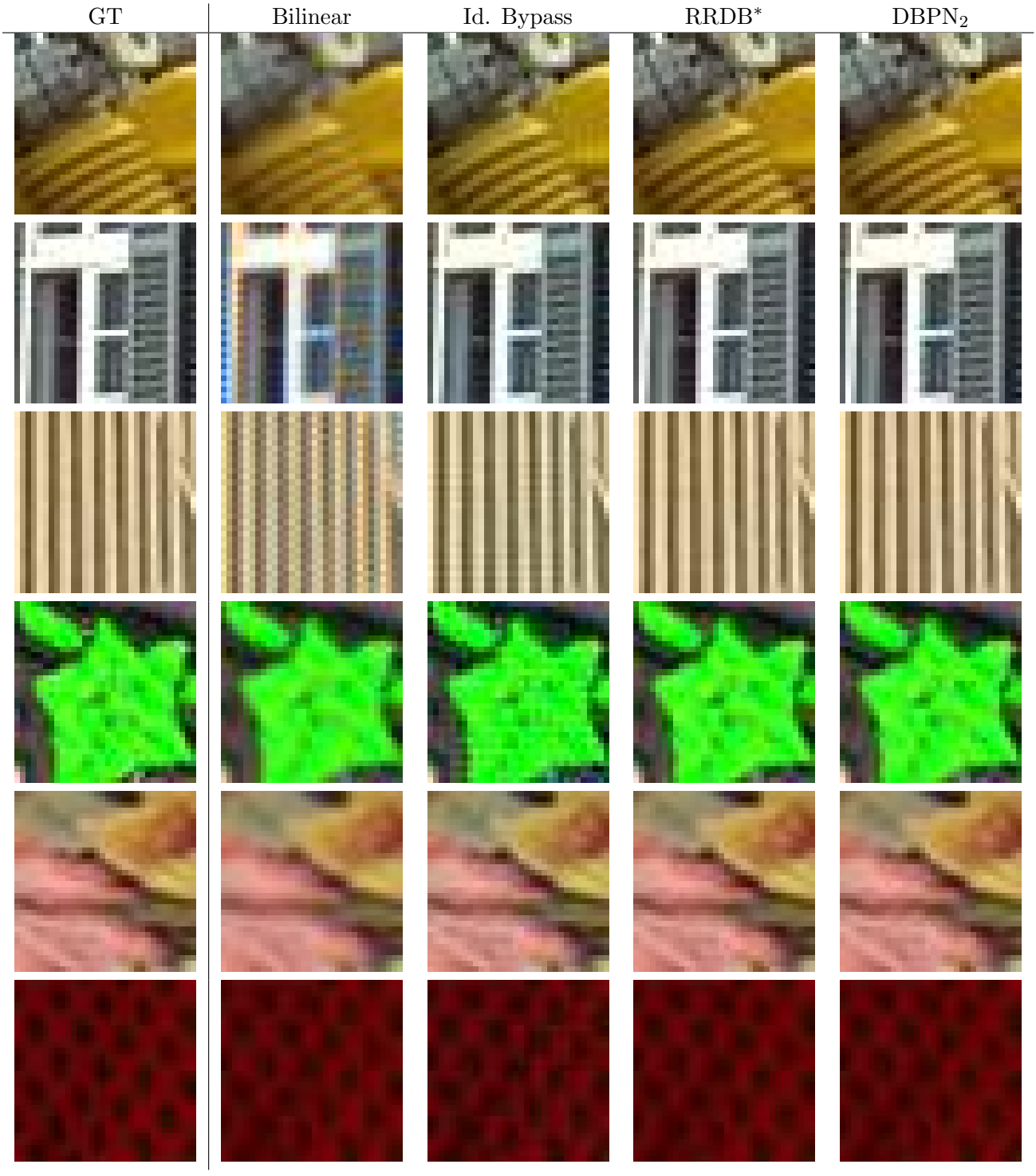


Figure 6.1: Qualitative evaluation of several proposed demosaicing methods. Ground truth (GT) patches are on the left.

Section 6.3

Temporal demosaicing models

As discussed above, algorithms for demosaicing a related sequence of frames have are less well-represented in the literature. The existing literature makes the assumption that frames can be registered by means of an affine warp, and does not take temporal order into account when demosaicing. We propose and examine MFD-Net, a network that uses an implementation of PWC-Net to compute optical flow for frame alignment, and combines the aligned frames using a pixel-wise softmax layer. Unlike previous networks for multi-frame demosaicing, MFD-Net is capable of aligning images in situations with complex motion. We also examine the extension of the Recurrent Back-Projection Network (RBPN) [23], a successful recurrent CNN for video super-resolution, to the video demosaicing problem.

Subsection 6.3.1

MFD-Net

For a more sophisticated temporal demosaicing algorithm, we propose the Multi-Frame Demosaicing Network (MFD-Net). As input, MFD-Net accepts a contextual sequence of mosaiced input frames $\{Y_i\}_{i=1}^{B-1}$ and a target frame Y. We use B to denote the total number of frames used as input to a temporal demosaicing network. MFD-Net consists of 3 submodules. The first module is a spatial demosaicing network that produces a first-pass demosaiced sequence $\{\tilde{X}_i\}$ and target $\tilde{X}$. The next module aligns each frame X_i to the target $\tilde{X}$ using a PWC-Net [57] style architecture to estimate optical flow. A final module is used to extract interpolation features from the sequence of aligned frames and the computed flows. The interpolation features learned from the final module are fed

into a softmax component, which generates a frame-wise probability distribution for each pixel. Softmax for a vector input x is calculated by

$$\text{softmax}(x)[i] = \frac{e^{x[i]}}{\sum_j e^{x[j]}}.$$

Thus a vector passed through softmax is normalized, and can be interpreted as a per-pixel probability distribution on the correctness of information in aligned frames. In our case, softmax operates over each feature vectors (one feature vector per pixel) in the final interpolation features. The output is obtained by calculated the expected value of each pixel subject to the frame-wise probability distribution.

We observe that, abeyant ground-truth optical flow for the training dataset, there is no guarantee that MFD-Net is computing an accurate optical flow internally. If MFD-Net is trained end-to-end, it may learn a way to warp the original input data in a way that "cooperates" with the spatial demosaicing subnetwork and softmax pixel prediction component to produce a final demosaiced frame, but would not be acceptable as a standalone optical flow. To circumvent this issue, we simply transplant the architecture and learned weights of PWC-Net into MFD-Net, wholesale. This approach is called *transfer learning* in the literature. By transferring PWC-Net into MFD-Net, we avoid the difficulty of having to train a flow estimation network from scratch on real-world video sequences without ground truth flow[3]. However, as in [68], we may fine-tune the internal flow network for the particular demosaicing task after having imported the main parameters.

Training MFD-Net is a difficult task because of the problem of *softmax saturation*. We empirically observe this problem in almost every one of our experiments. Softmax saturation occurs when, instead of learning to use a combination of pixels across the different aligned frames, the interpolation feature submodule instead "cheats" by discarding all frames except the first-pass demosaiced target

[3]Our numerical experiments indicate that this is a very difficult task.

frame (saturating the target frame softmax coefficients). This behavior is observed with gradient-descent based training algorithms if the spatial demosaicing network employed in MFD-Net is high-performing and the aligned frames are slightly misaligned. On the basis of our current experiments, every MFD-Net with a pre-trained spatial demosaicing subnetwork learns to cheat with softmax saturation. In light of this, to show a table of results for such networks amounts to repeating the earlier spatial demosaicing section.

Algorithm 12: MFD-Net

Data: $\{Y_i\}_{i=1}^{B}$: Bayer input frames, $\{\phi_i\}_{i=1}^{5}$: convolutional filter banks for learning input features, SpatialDemosaic: A pre-trained spatial demosaicing network (such as RRDBNet) that operates on each frame separately, CostVolume: algorithm to compute the cost volume of section 5.8, ResFlowCorrection: network (such as RRDBNet) to compute the residual flow correction with convolutional layers, Softmax: pixel-wise probability estimator

$Z_0 \leftarrow Y_1 \circ \cdots \circ Y_B$ **for** $i \leftarrow 1$ **to** 5 **do**
  $Z_i \leftarrow \downarrow_2 Z_{i-1}$
  $F_i \leftarrow \phi_i * Z_i$
end
$F_0 \leftarrow \text{SpatialDemosaic}(Z_0)$
$\nu_5 \leftarrow 0$
for $i \leftarrow 5$ **to** 1 **do**
  $C_i \leftarrow \text{CostVolume}(F_i, \nu_{i-1})$
  $\nu_{i-1} \leftarrow \uparrow_2 (\nu_i + \text{ResFlowCorrection}(C_i \circ \nu_{i-1})$
  $\tilde{F}_{i-1} = \text{Warp}(F_{i-1}, \nu_{i-1})$
end
$S_0 \leftarrow \tilde{F}_0 \circ \nu_0$
for $i \leftarrow 1$ **to** K **do**
  $S_i \leftarrow \psi_i * S_{i-1}$
end
$p = \text{Softmax}(S_K)$
return $\mathbb{E}_p[\tilde{F}_0]$

To circumvent the softmax saturation problem, we consider training MFD-Net architectures with low-quality spatial demosaicing. If a simple bilinear interpolation is used to generate the first-pass demosaiced sequence, we observe empirically that the resulting network does learn to use

Table 6.4: MFD-Net demosaicing a 2-frame sequence using bilinear spatial subcomponent vs. purely spatial bilinear demosaicing

Network	Parameters	Epochs	Training	Vimeo90K Test Subset
MFD-Net$_{bilin}$	9 868 232	50	L_2, Adam@1e-4	33.85
Bilinear	-	-	-	32.415

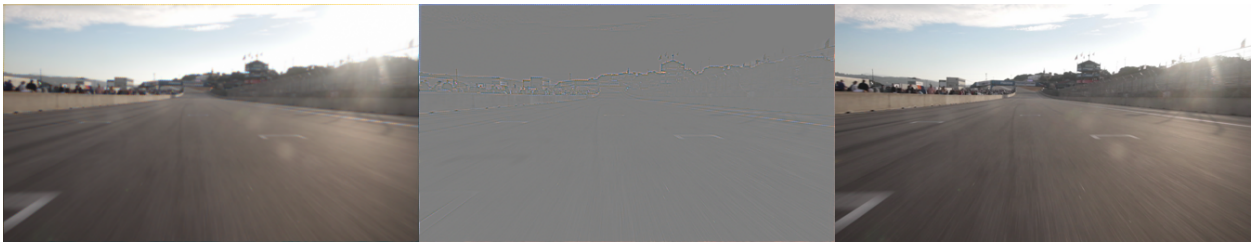


Figure 6.2: Left: MFD-Net demosaiced target frame of a sample 2-frame video sequence. Center: Difference map between interpolation and ground truth. Right: Ground truth target frame. Best viewed zoomed in on digital PDF.

softmax interpolation across the different frames. Quantitative results are given in Table 6.4, and qualitative results are shown in Figures 6.2 and 6.3.

This proof-of-concept result suggests that it is possible to leverage contextual temporal information from arbitrary frame sequences for the multi-frame demosaicing problem. Further work may investigate alternative training schemes for networks like MFD-Net to avoid the softmax saturation phenomenon.

Subsection 6.3.2

Recurrent back-projection for video demosaicing

We consider an application of the Recurrent Back-Projection Network (RBPN) of [23] discussed in Section 5.6 to the video demosaicing problem. The architecture of this model consists of a central projection module which is recurrently applied to concatenated frame pairs $\{Y_i, Y\}$ where Y_i is the

Figure 6.3: Visualization of softmax coefficients for MFD-Net$_{bilin}$ for a sample 2-frame sequence. Black indicates pixels used from context frame, white indicates pixels used from target

t^{th} contextual mosaiced frame and Y is the target frame to be demosaiced. At each step of the projection module, a low-resolution bundle of features is iteratively refined and used as input to the next pass of the module, starting from the frame closest (temporally) to the target. Each step of the projection module also produces a high-resolution bundle of features. After all frame pairs have been processed, a final convolutional layer interpolates the demosaiced target frame from the intermediate high-resolution feature bundles. We define the architecture RBPN$_B$ more precisely in Algorithm 13.

We define the RBPN$_2$ network for video demosaicing with a single temporal context frame. Following [23], we use a modified DBPN$_2$ network for SpatialNet$_\uparrow$ that accepts a 128-channel low-resolution input and produces a 64-channel high-resolution output. For MultiFrameNet$_\uparrow$, ResidualNet, and DecoderNet$_\downarrow$ we use a ResNet architecture of depth 14 with skip connections every other layer. Each low-resolution feature bundle produced contains 128 channels, and each high-resolution feature bundle produced contains 64 channels. The resulting architecture contains 1,195,531 parameters for 1 context frame ($B = 2$). Because the architecture is recurrent, parameters are reused between input frames and thus the number of parameters increases only slightly with B. This network is heavier than most previously proposed for demosacing single frames or burst sequences,

but is an order of magnitude lighter than flow estimation networks like PWC-Net. Importantly, the proposed network does *not* perform optical flow estimation. Haris et al. find that including flow information for the temporal context frames effects only a marginal improvement in network predictions.

We also investigate how DBPN performs under more restrictive conditions on network size. We train a reduced architecture with only 64 low-resolution intermediate feature channels and 32 high-resolution intermediate feature channels, with the internal ResNets with depths of only 6 and internal feature channels of 32 instead of the earlier 64, and the internal DBPN_2 using 32 internal feature channels instead of 64. We also increase the number of temporal context frames from 1 to 3. The resulting network RBPN_4 has 413,563 trainable parameters.

We find that RBPN_2 is capable of significantly outperforming spatial-only demosaicing methods on the Vimeo90K 200 sequence testing set. RBPN_4 is also able to outperform spatial-only demosaicing at a reduced network size. We report our results of training this network in Table 6.5 compared to the best spatial-only demosaicing RRDBNet method we trained (labeled RRDBNet*) and simple bilinear interpolation applied to the target frame of the sequence. These results show that the incorporation of temporal context information into the demosaicing problem for video sequences can lead to improved demosaicing quality.

Subsection 6.3.3

Qualitative comparison for temporal demosaicing

For qualitative comparison of several of the proposed temporal demosaicing methods, we include Figure 6.4. We include 64×64 patches where temporal context helps RBPN_4 outperform the spatial-only RRDB^* network, and one patch where the networks are equally matched. The JPEG

Algorithm 13: RBPN_B for video demosaicing

Data: Y: target Bayer frame, $\{Y_i\}_{i=1}^{B-1}$: Bayer context frames (temporally closest at index 1, temporally furthest at index $B-1$), ϕ, ψ, Φ: convolutional operators (with biases), ρ: nonlinearity, $\text{SpatialNet}_{\uparrow}$: CNN that back-projects from low-resolution feature space to high-resolution feature space, $\text{MultiFrameNet}_{\uparrow}$: CNN that obtains a back-projected residual from temporal context frames, ResidualNet: CNN for computing high-resolution residual to improve accuracy, $\text{DecoderNet}_{\downarrow}$: CNN corresponding to the forward operator, reduces high-resolution features to low-resolution features

Initialize low-res features L_0 and hi-res features H
$L_0 \leftarrow \rho(\phi * Y)$
$H \leftarrow 0$
Loop through context frames and apply projection module
for $i \leftarrow 1$ **to** $B-1$ **do**
  Learn features M from concatenated frames
  $Z_i \leftarrow Y_i \circ Y$
  $M_i \leftarrow \psi \circ Z_i$
  Back-projection module obtains high-res features
  $H_{i,s} \leftarrow \text{SpatialNet}_{\uparrow}(L_{i-1})$
  $H_{i,m} \leftarrow \text{MultiFrameNet}_{\uparrow}(M_i)$
  $E_i \leftarrow \text{ResidualNet}(H_{i,s} - H_{i,m})$
  Save hi-res feature bundle for later concatenation
  $\hat{H}_i \leftarrow H_{i,s} + E_i$
  $H \leftarrow H \circ \hat{H}_i$
  Forward module obtains next iteration of low-res features
  $L_i = \text{DecoderNet}_{\downarrow}(\hat{H}_i)$
end
return $\rho(\Phi * H)$

Table 6.5: RBPN networks for video demosaicing compared to spatial-only methods. The learning rate was reduced by a factor of 10 at epoch 189 for RBPN_2 and at epoch 125 for RBPN_4.

Network	Parameters	Epochs	Training	Vimeo90K Test Subset
RBPN_2	1 195 531	208	L_1, Adam@1e-3	45.271
RBPN_4	413 563	175	L_1, Adam@1e-3	44.795
RRDBNet*	419 852	20 700	L_1, Adam@1e-3	44.748
Bilinear	-	-	-	35.390

artifacts that spoil the Vimeo-90K dataset are noticeable in the difference maps, a reminder that a high-quality dataset is necessary before a sufficiently generalizable network can be trained.

Interestingly, RBPN_4 often outperforms RRDB^* on the edges of objects in the frames. This result is slightly counterintuitive, since the edge of an image in a sequence of frames constitutes a high-frequency moving target that, a priori, one might expect to be difficult to demosaic using temporal methods.

Section 6.4

Conclusion

In this manuscript we have examined the use of convolutional neural networks for the image and video demosaicing problems. We briefly summarize our conclusions from the numerical experiments in this chapter. First, dense residual architectures like the residual-in-residual dense block for single-image demosaicing offers a significant improvement over existing identity or bilinear bypass networks, even when controlling for the network parameter count. Second, the MFD-Net architecture for use in burst and video demosaicing has demonstrated the potential to improve demosaicing quality over purely spatial methods, but training such network architectures poses significant difficulties. Finally, by adapting the recurrent back-projection architecture (inspired by early work in super-resolution) for the demosaicing problem, we showed that a video demosaicing network without the estimation of optical flow can produce significantly higher-quality demosaiced images compared to spatial-only demosaicing networks.

Future work might consider developing a theoretical basis for the use of dense residual connections in iterative reconstruction schemes, overcoming the training problem for MFD-Net, and investigating the conditions under which regions of a reconstructed image become spoiled. Additionally, the

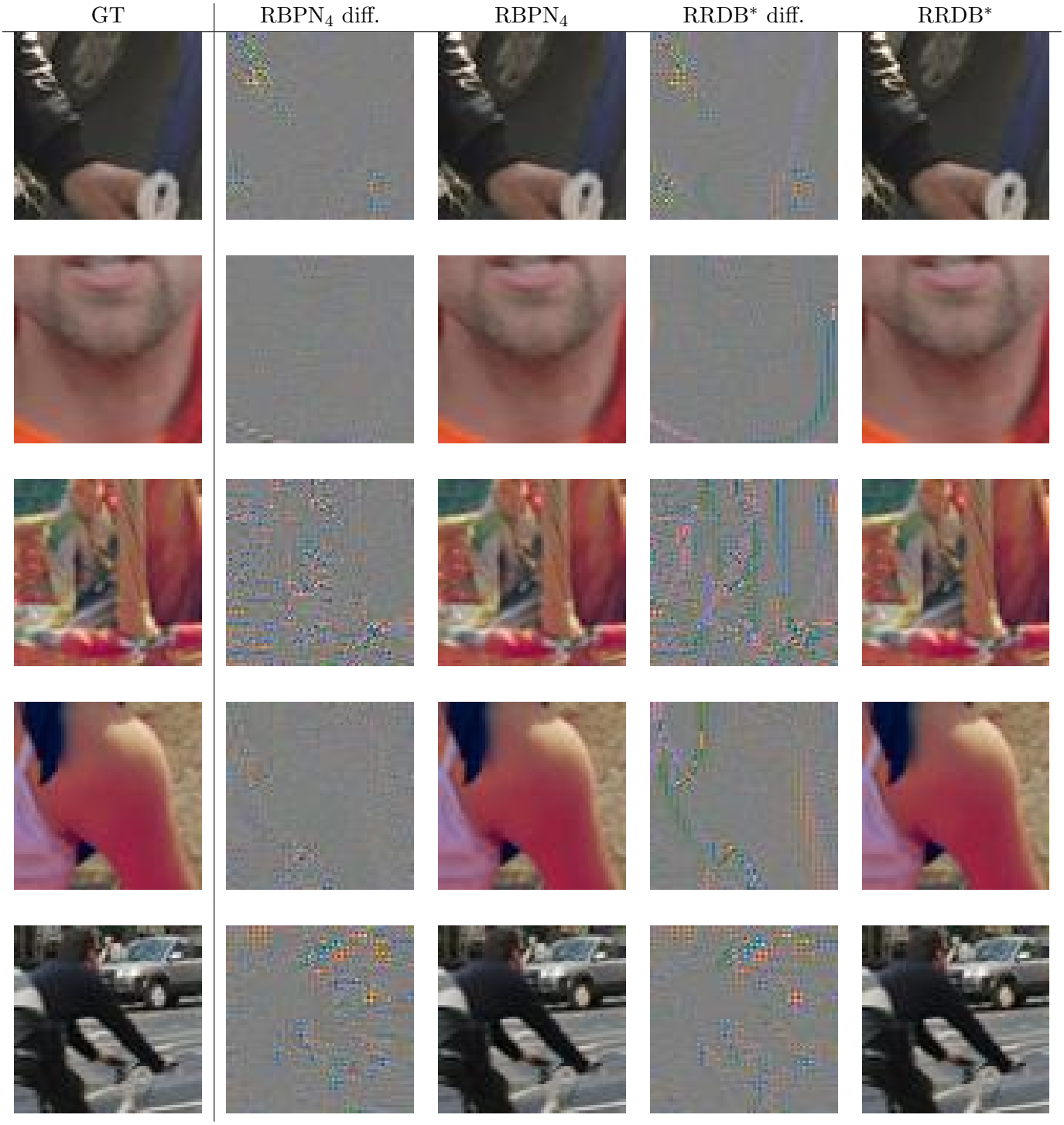


Figure 6.4: Qualitative evaluation of temporal demosaicing methods. The difference maps are amplified by a factor of 5 for easier comparison. Ground truth (GT) is displayed on the left.

construction of a realistic video demosaicing dataset would greatly aid efforts to develop practical temporal demosaicing algorithms for video and film scanning applications.

Bibliography

[1] J. E. Adams, "Interactions between color plane interpolation and other image processing functions in electronic photography," in *Cameras and Systems for Electronic Photography and Scientific Imaging*, C. N. Anagnostopoulos and M. P. Lesser, Eds., SPIE, Mar. 1995.

[2] M. Anderson, R. Motta, S. Chandrasekar, and M. Stokes, "Proposal for a standard default color space for the internet—sRGB," in *Color and imaging conference*, Society for Imaging Science and Technology, vol. 1996, 1996, pp. 238–245.

[3] G. Bal, "Introduction to inverse problems," *Lecture Notes-Department of Applied Physics and Applied Mathematics, Columbia University, New York*, 2012.

[4] B. E. Bayer, *Color imaging array*, US Patent 3,971,065, Jul. 1976.

[5] A. Beck and M. Teboulle, "Gradient-based algorithms with applications to signal recovery," *Convex optimization in signal processing and communications*, pp. 42–88, 2009.

[6] H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen, "Optimal approximation with sparsely connected deep neural networks," *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 1, pp. 8–45, 2019.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[8] K. Bredies and D. Lorenz, *Mathematical Image Processing*. Springer International Publishing, 2018.

[9] W. Cheney and W. Light, *A Course in Approximation Theory*. Cole Publishing Company, Pacific Grove, California, 2000.

[10] D. R. Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal," 4,642,678, Feb. 1987.

[11] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[12] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.

[13] R. Descartes, "Meditations on first philosophy in which are demonstrated the existence of god and the distinction between the human soul and body," trans. by J. Bennett, *Descartes: Selected Philosophical Writings*, pp. 73–122, 2010.

[14] T. Ehret, A. Davy, P. Arias, and G. Facciolo, "Joint demosaicking and denoising by fine-tuning of bursts of raw images," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019.

[15] S. Farsiu, M. Elad, and P. Milanfar, "Multiframe demosaicing and super-resolution of color images," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 141–159, Jan. 2006.

[16] P. Fischer, A. Dosovitskiy, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Dec. 2015.

[17] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, "Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data," *IEEE Transactions on Image Processing*, vol. 17, no. 10, pp. 1737–1754, 2008.

[18] R. W. Franzen, Eastman Kodak Company: PhotoCD PCD0992.

[19] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, "Deep joint demosaicking and denoising," *ACM Trans. Graph.*, vol. 35, no. 6, 191:1–191:12, Nov. 2016, ISSN: 0730-0301.

[20] B. Gunturk, J. Glotzbach, Y. Altunbasak, R. Schafer, and R. Mersereau, "Demosaicking: Color filter array interpolation," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 44–54, Jan. 2005.

[21] G. Hämmerlin and K. Hoffman, *Numerical Mathematics*. Springer New York, 1991.

[22] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks for super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1664–1673.

[23] M. Haris, G. Shakhnarovich, and N. Ukita, "Recurrent back-projection network for video super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3897–3906.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15, USA: IEEE Computer Society, 2015, pp. 1026–1034, ISBN: 9781467383912.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185–203, Aug. 1981.

[27] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical models and image processing*, vol. 53, no. 3, pp. 231–239, 1991.

[28] J. Kaipio and E. Somersalo, *Statistical and Computational Inverse Problems*. Springer Science & Business Media, 2006, vol. 160.

[29] D. Khashabi, S. Nowozin, J. Jancsary, and A. W. Fitzgibbon, "Joint demosaicing and denoising via learned nonparametric random fields," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 4968–4981, Dec. 2014.

[30] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, Dec. 2014.

[31] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, *Self-normalizing neural networks*, 2017. arXiv: `1706.02515 [cs.LG]`.

[32] F. Kokkinos and S. Lefkimmiatis, "Iterative joint image demosaicking and denoising using a residual denoising network," *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 4177–4188, Aug. 2019.

[33] F. Kokkinos and S. Lefkimmiatis, "Iterative residual cnns for burst photography applications," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5929–5938.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[35] M.-J. Lai and L. L. Schumaker, *Spline Functions on Triangulations*, 110. Cambridge University Press, 2007.

[36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[37] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[38] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[39] H.-C. Lee, *Introduction to Color Imaging Science*. Cambridge University Press, Feb. 2005.

[40] S. Lefkimmiatis, "Universal denoising networks : A novel CNN architecture for image denoising," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018.

[41] X. Li, B. Gunturk, and L. Zhang, "Image demosaicing: A systematic survey," in *Visual Communications and Image Processing 2008*, W. A. Pearlman, J. W. Woods, and L. Lu, Eds., SPIE, Jan. 2008.

[42] S. Mallat, *A Wavelet Tour of Signal Processing*. Elsevier, 1999.

[43] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, Jul. 2012.

[44] S. Mallat, "Understanding deep convolutional networks," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20 150 203, 2016.

[45] D. Menon and G. Calvagno, "Color image demosaicking: An overview," *Signal Processing: Image Communication*, vol. 26, no. 8-9, pp. 518–533, Oct. 2011.

[46] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[47] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $o(1/k^2)$," *Dokl. akad. nauk Sssr*, vol. 269, pp. 543–547, 1983.

[48] V. Papyan, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2887–2938, 2017.

[49] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014, ISSN: 2167-3888.

[50] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[51] G. Qian, J. Gu, J. S. Ren, C. Dong, F. Zhao, and J. Lin, "Trinity of pixel enhancement: A joint solution for demosaicking, denoising and super-resolution," *arXiv preprint arXiv:1905.02538*, 2019.

[52] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018.

[53] A. J. Robinson and F. Fallside, *The utility driven dynamic error propagation network.* University of Cambridge Department of Engineering, 1987.

[54] J. R. Sashank, K. Satyen, and K. Sanjiv, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018.

[55] L. Schumaker, *Spline Functions: Basic Theory.* Cambridge University Press, 2007.

[56] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016.

[57] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018.

[58] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.

[59] N.-S. Syu, Y.-S. Chen, and Y.-Y. Chuang, *Learning deep convolutional networks for demosaicing*, 2018. arXiv: `1802.03769 [cs.CV]`.

[60] D. S. Tan, W.-Y. Chen, and K.-L. Hua, "DeepDemosaicking: Adaptive image demosaicking via multiple deep fully convolutional networks," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2408–2419, May 2018.

[61] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, 2005.

[62] P. Thévenaz, T. Blu, and M. Unser, "Image interpolation and resampling," *Handbook of medical imaging, processing and analysis*, vol. 1, no. 1, pp. 393–420, 2000.

[63] A. N. Tikhonov and V. Y. Arsenin, "Solutions of ill-posed problems," 1977.

[64] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 63–79.

[65] A. S. Wannenwetsch and S. Roth, "Probabilistic pixel-adaptive refinement networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 642–11 651.

[66] T. Wiatowski and H. Bolcskei, "A mathematical theory of deep convolutional neural networks for feature extraction," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1845–1866, Mar. 2018.

[67] G. Wolberg, *Digital Image Warping*. IEEE computer society press Los Alamitos, CA, 1990, vol. 10662.

[68] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.

[69] J. C. Ye, Y. Han, and E. Cha, "Deep convolutional framelets: A general deep learning framework for inverse problems," *SIAM Journal on Imaging Sciences*, vol. 11, no. 2, pp. 991–1048, 2018.

[70] Z. Yin, T. Darrell, and F. Yu, "Hierarchical discrete distribution decomposition for match density estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6044–6053.

[71] L. Zhang, X. Wu, A. Buades, and X. Li, "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding," *Journal of Electronic imaging*, vol. 20, no. 2, p. 023 016, 2011.