Wei Yu

Machine Learning Applications in Computer Vision

DOCTORAL THESIS

to achieve the university degree of
Doktorin der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr. Thomas Pock
Univ.-Prof. Robert Sablatnig

Graz, Austria, April. 2017

To Xia, Kang, Shun and ShuoPeng.

The scientist is not a person who gives the right answers, he's one who asks the right questions.

Claude Lvi-Strauss (1908 - 2009)

Abstract

Machine Learning (ML) is a branch of Artificial Intelligence (AI), that describes methods to automate analytical model building. Related methods provide computers with the ability to learn without the need of explicit programming, i.e. these methods learn from and make predictions on data. Because of the ability to tackle problems involving large-scale data, ML methods can be applied across many industrial applications, like for instance, fraud detection, driving assistance, image classification and image captioning. The first ML algorithm, the so-called perceptron algorithm, was introduced in 1957 for the task of binary classification. Later it was generalized to the multi-layer perceptron, which is the prototype of the neural network. ML algorithms, like e.g. Deep Neural Network (DNN) or Support Vector Machines (SVMs), are based on large-scale optimization problems, where the choice of the optimization method has an influence on the training and consequently on the provided solution for the corresponding ML tasks, which should not be underestimated.

The two main topics of this thesis are ML methods and large-scale optimization techniques. Besides giving an overview on theoretical foundations, including topics like optimization algorithms and inverse problems, this thesis reviews several classical and popular ML problems including dimensionality reduction, multi-task learning, feature selection, SVMs, matrix completion and matrix factorization. We compare the different ML problems based on a large variety of first-order optimization algorithms, including the Forward-backward Splitting Method (FBS), the Fast Iterative Shrinkage-thresholding Algorithm (FISTA), the Optimal Subgradient Algorithm (OSGA), and the Primal-Dual Algorithm (PDCP). We also present the Online PDCP which provides a faster empirical convergence than PDCP. In the remaining part of the thesis, we apply ML methods to three fundamental Computer Vision (CV) problems: Single-Image Inpainting, Shape from Light Field (SfLF) and Light Field Super Resolution (LFSR).

Firstly, we present a diffusion model for image inpainting. The model is based on the so-called trained Reaction-Diffusion Model (RDM), which can also be seen as a Recurrent

Neural Network (RNN). We train two diffusion models based on the Structural Similarity Image Measure (SSIM) for two types of inpainting tasks. The first one is a generic model, which aims to recover large regions of scattered pixels or to restore small connected regions on images. The second model is referred to as a specific model, which aims to inpaint the texture of a given image. Adequate inpainting results show that both inpainting models can transform corrupted images to visually pleasing images.

Secondly, we use u-shaped Convolutional Neural Networks (CNNs) for the task of depth estimation in the Light Field (LF) setting, which is referred to as Shape from Light Field. We train our network on so-called Epipolar Plane Images (EPIs), rather than the entire LF. As a result, our network is able to provide efficient and accurate reconstructions. We demonstrate the superior performance of our network on synthetic and real-world data. In addition, we present a comprehensive comparison to the state of the art.

Finally, we continue to the problem of LFSR. Here we extend the trained RDM to perform 3D filtering in the LF setting, i.e. the trained models explore more than just one single view. To be efficient, we only recover the high-resolution center view of a so-called EPI volume. By comparing the results of our 3D RDM to the result of a correspond 2D RDM, we are able to show the benefit of our proposed approach.

Keywords. Machine Learning, Computer Vision, Neural Network, Reaction-diffusion Model, First-order Method, Image Inpainting, Shape from Light Field, Light Field Superresolution

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date	Signature

Acknowledgments

Pursuing my PhD at TU Graz enriched my life and provided me the great opportunity to get acquainted with and learn from many outstanding and intelligent researchers. First and foremost, I would like to express my sincere gratitude to my advisor Prof. Thomas Pock for supervising my thesis and giving me the opportunity to attend a top Ph.D program in the field of Machine Learning and Computer Vision. Without the continuous support, the sagacious suggestions and the immense knowledge from Prof. Thomas Pock, the completion of this study would not have been possible. Moreover, I would like to thank Prof. Robert Sablatnig for acting as my second supervisor. Further, my work also benefited from the encouraging teamwork environment and the positive atmosphere in the institute of Computer Graphics and Vision. Our group meeting provided an opportunity for cooperation and blending of each other's strengths. Thus, my sincere thanks also goes to our group members. Especially, I would like to thank my co-worker, Stefan Heber, for the help and the insight. Also I thank my friends in the institute, Jens Grubert, Jan Egger, Horst Possegger, Pedro Boechat, Stefanie Zollmann, Tobias Langlotz, Markus Steinberger, Yunjin Chen and so forth. Aside from work, I want to thank my schoolmate, Linke Li, for all the information shared with me. Last but not the least, I want to thank my parents, my brothers and my sisters for their selfless love.

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 Motivation

Computer vision is a fast growing field concerned with the development of mathematical techniques and algorithms for understanding and interpreting digital images. Computer vision applications span a large spectrum of abstraction levels, ranging from extracting pointwise information (low-level vision) all the way to obtaining a semantic understanding of the observed scene (high-level vision). The basic starting point for all computer vision applications is the digital image, that can be acquired by different input devices and techniques like digital cameras, image scanners, or line scanners, to name but a few. The digital image is usually represented as a 2D matrix also called raster image, where each element represents a measured intensity value.

Computer vision is an inverse discipline. When people observe the surrounding 3D world, the images projected on their retinas are only two-dimensional. Due to the reduction of one dimension, a lot of information about the observed scene is lost, e.g., information about the distance to certain objects. Thus humans and animals only observe incomplete measurements of the environment, but they are nevertheless able to effortlessly navigate and interact with this environment, i.e. that they are able to invert the image formation process. In most cases this inversion leads to a reasonable interpretation of the scene. However, due to the incomplete measurements this inverse problem might also lead to ambiguous results, consider for instance various optical illusions. An example of such an optical illusion is shown in Figure 1.1, where the illustrated 3D object can have both opposite meanings, “YES” or “NO”, by observing

Figure 1.1: An example of an optical illusion invented by the Swiss artist Markus Raetz. The word "YES" slowly turns into "NO" by changing the viewpoints.

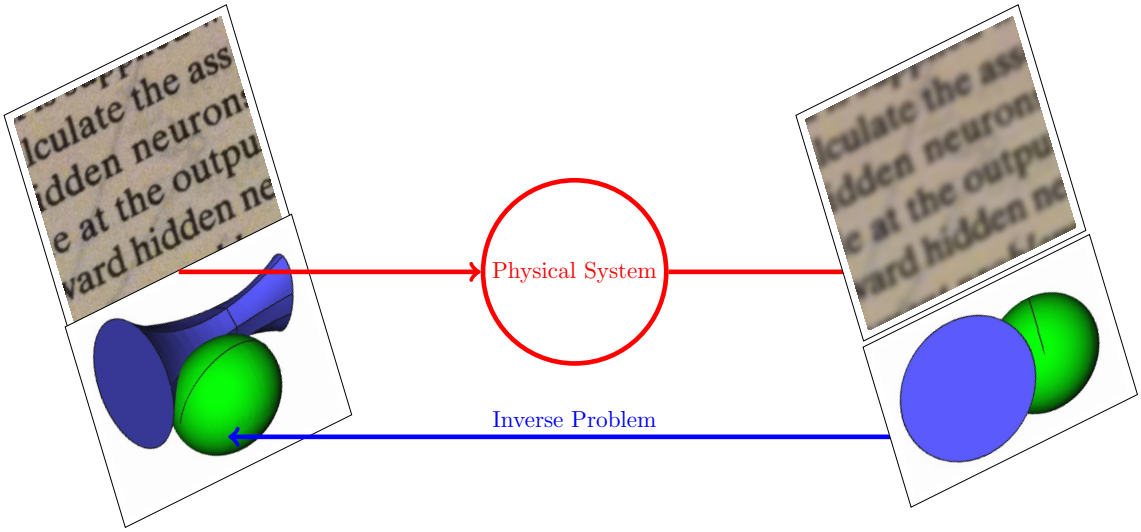


Figure 1.2: Illustration of inverse problems. Given the output signal, an inverse problem is to recovery the input signal.

it from different viewpoints. This is an example where the human visual system is not able to correctly resolve the 3D structure of the scene. The general inverse problem in the human visual system is to retrieve information about the 3D environment using only the more limited information contained in the 2D image projected on the retina of the eye. Likewise, this is the intrinsic difficulty arising in computer vision and digital image processing, where researcher are trying to understand the fundamental nature of the inverse problem for several decades. Two common inverse problems in computer vision are illustrated in Figure 1.2 (image superresolution and depth estimation). New understanding interleaved with massive engineering efforts and an explosion of computational resources led to inspiring developments in many corresponding applications, including image registration, image denoising, and image inpainting.

Although computer vision problems are tough, digital image processing becomes prevalent more than ever and has quietly infiltrated into people's daily lifes, which is illustrated by a few examples. Smartphones nowadays naturally come with built-in digital cameras, where the photo quality is close to professional stand alone digital

cameras. After snapping a picture with their phones, users can edit and modify the pictures right on the device. In digital cameras, face detection has become a mature technology for tasks like auto focus, or removing the unwanted red-eye effect and so forth [126, 157]. For security, fingerprints and the iris of the eye are used as a biometric identification by door lock systems or governments [166, 173]. Indoor security cameras with face recognition can send notifications when your children or elderly parents are home and alert when it sees a stranger [180]. In industrial applications, digital image processing is frequently used for quality control, i.e. finding defective manufactured items [89, 112]. In astronomy, digital image processing is used to remove geometric and photometric distortions in images [86, 87]. A last example is medical image processing, which is a sub-field of digital image processing [5, 48, 96]. The InnerEye research project of Microsoft for instance focuses on the automatic analysis of patients' medical scans. One achievement of the InnerEye research project is the automatic detection and localization of the anatomy of the patient.

One reason for the success of digital image processing is the application of Machine Learning (ML). ML develops as a scientific discipline in the late 1990s and the application of ML has fast spread throughout computer science and beyond in the last decades. ML benefits from increasingly cheaper and faster computational processing, large varieties of available data on the Internet and more and more affordable data storage devices. For a vision problem, rather than designing a specific model, ML allows to learn how to find hidden patterns in the data and use them to automatically build models for the given task. The learned models can generate predictions for the unseen data. This is because large training sets contain adequate numbers of patterns that frequently occur in unseen data. In contrast to statistical models, ML does not depend on the understanding of the structure of the data and the profound theory of the distributions of the data. Ethem Alpaydin summarized this in the following quote [6]

> Machine learning is programming computers to optimize a performance criterion using example data or past experience.

As one of the most promising and exciting technologies, ML attracts extensive attention of giant companies. Nowadays many companies provide ML platforms or frameworks that allow users to easily create ML models. Examples include IBM's 'Watson Developer Cloud' [Watson Developer Cloud], 'Amazon Machine Learning' [Amazon Machine Learning], Microsoft's 'Azure Machine Learning' [Azure Machine Learning], and Google's 'TensorFlow' [TensorFlow].

The main reason that led to the prevalence of ML is that related algorithms allow to learn from given experience when it is not clear for a human how to model or program the corresponding behaviour. For example, Google's self-driving cars depend on sensors and software, without explicitly programming how to drive, those cars learn to navigate through unseen and complicated scenarios on city streets by utilizing ML algorithms. Facebook's DeepFace is a facial recognition system of a nine-layer neural network involving

more than 120 million parameters. DeepFace achieved an accuracy of 97.35%, which is closely approaching the human test result of 97.53%. The game of Go is an ancient and complex game whose possible legal positions are way more than the atoms in the universe. Google's AlphaGo program beat Lee Sedol, a South Korean professional Go player of 9-dan rank. Furthermore, the recommendations from Online Shopping website like Amazon or eBay, or the movie recommendations from Netflix are further well known examples of ML applications. Instead of investigating how to drive, how to conquer the game of Go, how to predict the taste of every distinct human being, ML allows to learn from the formal experiment and breaks records with stunning results. Thus for a complex problem, learning to find a solution using ML is in many cases easier and more effective than seeking for a possible analytical solution. But, unfortunately, ML does need expertise to guide the process of learning. The process of creating a ML application consists of the following steps: (i) preparing a training data set and a test data set, (ii) extracting features from the raw data, (iii) selecting a suitable, possible an off-the-shelf, ML algorithm, (iv) formulating a performance criterion, and (v) selecting a suitable optimization algorithm. All of those steps are important to create a successful machine leaning application. First of all, the availability of a suitable training data is absolutely essential. Moreover, the more training data fed in a ML algorithm, the more accurate the solution could be. The tricky part of ML is that the solution is optimised based on a training data set, but the goal is to achieve a good result on a test data set, where it is assumed that the training data set and the test data set share the same distribution and have similar patterns that can be explored. The amount of training data for an application can be tremendous or even infinite. Unfortunately, there exist no rule how to select the optimal amount of data needed to train a specific model.

Feature extraction is another important step of a ML application. Unfortunately it is also a creative "black art" [50]. There are basically two ways to approach the feature extraction step: On the one hand, feature engineering [50, 71, 90, 106, 107] describes the process of manually extracting features from the raw data by using domain knowledge of the data. On the other hand, feature learning or representation learning [15, 45, 46] integrates the learning of features into the ML model.

It stands to reason, that selecting the right ML algorithm is one key determining success or failure of an application. Without any claim to completeness, ML algorithms can be divided into categories like anomaly detection, multi-class classification, two-class classification, and regression. Different methods can be compared based on sampling methods (e.g. cross validation, random re-sampling), score functions (e.g. deviations, recall), or overall comparisons (e.g. t-test). Moreover, diverse variations and ensemble methods have been proposed in the literature, like for example bagging, boosting, or stacking [24, 61, 62, 168].

In ML, almost all problems are formulated as optimization problems, where the goal is to optimize an objective function. One part of the input includes the raw data samples or the extracted features in the case of feature learning or feature engineering, respectively.

The other part of the input are parameters used in the optimization algorithm, e.g. the learning rate in the case of training a Neural Network (NN), or the number of nearest neighbours in k-means. Tuning the various optimization parameters is crucial, especially for non-convex ML models.

The optimization method is the final ingredient for an successful ML application. For a convex problem, the convergence rate of an optimization algorithm determines the speed of finding the solution. This is increasingly important since large scale ML is demanded in the year of big data. However, for a non-convex problem, optimization algorithms only assure local solutions, i.e. different initializations or optimization parameters will lead to different solutions. However, research shows that if we drop the requirement of a convex model, a descriptive model including non-convex terms can lead to improved performance in different image processing problems (e.g. image denoising or image compression). IPiano [125], for instance, is an optimization algorithm for solving such non-convex problems which has a faster empirical convergence rate than the theoretical bound.

In this thesis, we first frame mathematical foundations of ML and optimization. Based on that, we then further illustrate how to build up state-of-the-art ML applications for digital image processing. More specifically, we will tackle tasks like inpainting, stereo reconstruction, and super-resolution.

1.2 Dissertation Overview

Organizational Themes. This thesis focuses on applications for digital image processing using ML and covers the following contributions:

First, we present a detailed analysis and evaluation of optimization methods for varying ML applications (cf. Section 3.2). Secondly, we propose state of the art models for different image processing applications, including image inpainting (cf. Section 4.1.1), Light Field depth estimation (cf. Section 4.2.1) and Light Field superresolution (cf. Section 4.2.2).

Chapter Descriptions. Chapter 2 presents notations and basic mathematical definitions utilized throughout the remaining parts of the thesis. The overview includes a short introduction to the tensor calculus, inverse problems in digital image processing, ML and optimization.

Chapter 3 investigates the relation between ML and optimization. Specifically, Chapter 3 compares several first-order optimization algorithms based on classic ML problems, like dimensionality reduction, linear Support Vector Machine (SVM), kernel SVM, feature selection, multi-task learning, matrix completion and matrix factorization.

After the primitive applications shown in Chapter 3, in Chapter 4, we first turn to tackle the problem of image inpainting. We use a so-called diffusion model for single view image processing. Here we illustrate how to learn parameterized linear filters and influence functions. Secondly, we tackle the problems of Light Field (LF) depth estimation and LF superresolution. We use deep learning techniques to estimate depth information in the

LF setting by leveraging LF dataset. The resulting method provides a good combination between accuracy and efficiency. Finally, we use a 3D Reaction-Diffusion Model (RDM) for Light Field Super Resolution (LFSR). Note, that this chapter comprises parts of the publications [151, 176].

Chapter 5 summarizes the main points of the thesis and discusses future directions.

2

Mathematical Foundation

Contents

2.1 Notation and Conventions

In this section we introduce the mathematical notations which are consistently used throughout the thesis. Scalar values are represented by italic fonts, e.g. x. For representing vectors, we use boldface lower case letters, e.g., $\mathbf{v} = [v_1\ v_2\ \cdots\ v_n]^\top$. $\mathbf{v}$ is called a column vector with n components, where v_j denotes the j^{th} component. In this thesis, if n components are arranged in a horizontal array, as in

$$\mathbf{v}^\top = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix},$$

then $\mathbf{v}^\top$ is called a row vector. $\mathbf{v} = \max\{\mathbf{a}, \mathbf{b}\}$ denotes the pointwise maximum of vectors $\mathbf{a}$ and $\mathbf{b}$, i.e.

$$v_i = \max\{a_i, b_i\}. \tag{2.1}$$

Likewise, the element-wise minimum of two vectors $\mathbf{a}$ and $\mathbf{b}$ is $\mathbf{v} = \min\{\mathbf{a}, \mathbf{b}\}$. The space of vectors with n components is depicted as, $\mathbb{R}^n, n \geq 1$. If we write $\mathbf{x} - c$ with $\mathbf{x} \in \mathbb{R}^n$,

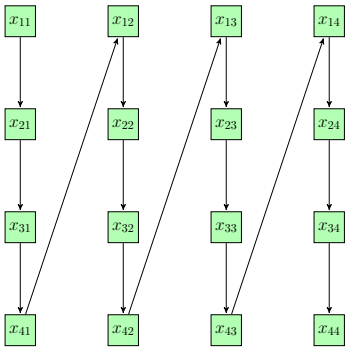


Figure 2.1: Column major representation of a matrix.

and $c \in \mathbb{R}$, then the scalar c is broadcasted to all elements of $\mathbf{x}$, i.e.

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} c \\ \vdots \\ c \end{bmatrix}.$$

The boldface upper case letters denote matrices,

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix}.$$

The matrix $\mathbf{X}$ has m rows and n columns. The space of matrices with m rows and n columns is denoted as $\mathbb{R}^{m \times n}, m, n \geq 1$. In this thesis, we also use the column major vector representation to represent a vector which traverses the matrix by columns. Therefore for a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, the column-major vector of it is $\mathbf{X}(:) \in \mathbb{R}^{mn}$, cf. Figure 2.1.

We denote by $\mathbf{I}$ the identity matrix, unless otherwise stated. Given a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, $\mathbf{X}^\top$ denotes the transpose matrix of $\mathbf{X}$. Two subscript indices $X_{i,j}$ represent the entry at the i^{th} row and the j^{th} column. $\mathbf{X}_{*j}$ indicates the j^{th} column of $\mathbf{X}$ and similarly $\mathbf{X}_{i*}$ indicates the i^{th} row of $\mathbf{X}$. We denote by $\mathbf{X} \cdot \mathbf{Y}$ the Hadamard product between matrices $\mathbf{X}$ and $\mathbf{Y}$, unless otherwise stated. The Hadamard product also applies to vectors as a special type of matrices, where there is only one row or one column.

A functional mapping between different spaces is given in upper case calligraphic letters, e.g., $\mathcal{F}$. $\nabla\mathcal{F}$ denotes the gradient or subgradient of the function $\mathcal{F}$. And we denote by $\mathrm{dom}(\mathcal{F})$ the domain of a function $\mathcal{F}$.

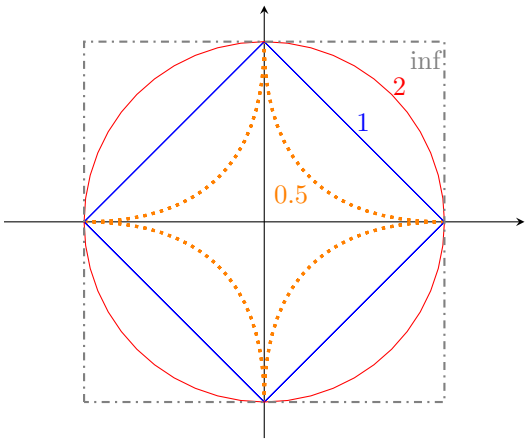


Figure 2.2: The figure depicts the contours of equal distance for the $\frac{1}{2}$norm, 1-norm, 2-norm, and infinity-norm.

The indicator function δ of a subset A of a set $\mathbb{R}^n$ is defined as,

$$\delta_A(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in A, \\ +\infty & \text{if } \mathbf{x} \notin A. \end{cases} \tag{2.2}$$

Vector Norm. A vector norm on $\mathbb{R}^n$ is a function $\|\cdot\|$, from $\mathbb{R}^n$ to $\mathbb{R}_+$ with the following properties,

(i) $\|\mathbf{v}\| \geq 0,\ \|\mathbf{v}\| = 0 \iff \mathbf{v} = \mathbf{0},\ \forall \mathbf{v} \in \mathbb{R}^n$ (positivity)

(ii) $\|\mathbf{av}\| = |a|\,\|\mathbf{v}\|\ \forall a \in \mathbb{R}, \mathbf{v} \in \mathbb{R}^n$ (scaling)

(iii) $\|\mathbf{v} + \mathbf{w}\| \leq \|\mathbf{v}\| + \|\mathbf{w}\|\ \forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ (triangle inequality)

Common vector norms are

- $\|\mathbf{v}\|_1 = |v_1| + \cdots + |v_n|$ (1-norm)
- $\|\mathbf{v}\|_2 = (|v_1|^2 + \cdots + |v_n|^2)^{\frac{1}{2}}$ (2-norm)
- $\|\mathbf{v}\|_{\text{inf}} = \max\{|v_i| \mid 1 \leq i \leq n\}$ (infinity-norm)

Generally, the $\ell_p - \mathbf{norm}, p \geq 1$ defined as

$$\|\mathbf{v}\|_p = (\sum |v_i|^p)^{\frac{1}{p}} \tag{2.3}$$

includes all the norms depicted above as special cases. The norm can induce a metric which is frequently used to measure the error or loss of a model.

Metric. Let S be a set. A metric is a function $\mathcal{D} : S \times S \to \mathbb{R}$ if it satisfies the following properties

(i) $\mathcal{D}(\mathbf{x}, \mathbf{y}) \geq 0$, $\mathcal{D}(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$, $\forall\ \mathbf{x},\ \mathbf{y} \in \mathbb{R}^n$ (non-negativity and identity)

(ii) $\mathcal{D}(\mathbf{x}, \mathbf{y}) = \mathcal{D}(\mathbf{y}, \mathbf{x})$ (symmetry)

(iii) $\mathcal{D}(\mathbf{x}, \mathbf{z}) \leq \mathcal{D}(\mathbf{x}, \mathbf{y}) + \mathcal{D}(\mathbf{y}, \mathbf{z})$ (triangle inequality)

A norm induces a metric by letting $\mathcal{D}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$.

Matrix Norm. For a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$, its norm is a function $\|\mathbf{M}\| : \mathbb{R}^{m \times n} \to \mathbb{R}_+$ that meets the following properties,

(i) $\|\mathbf{M}\| \geq 0,\ \|\mathbf{M}\| = 0 \iff \mathbf{M} = \mathbf{0},\ \forall \mathbf{M} \in \mathbb{R}^{m \times n}$ (positivity)

(ii) $\|a\mathbf{M}\| = |a|\, \|\mathbf{A}\|$, where $a \in \mathbb{R}$ (homogeneity)

(iii) $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|,\ \forall \mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$ (triangle inequality)

Some matrix norms also satisfy the following additional properties,

(iv) $\|\mathbf{M}\mathbf{x}\| \leq \|\mathbf{M}\| \|\mathbf{x}\|$ (sub-ordinance)

(v) $\|\mathbf{A}\mathbf{B}\| \leq \|\mathbf{A}\| \|\mathbf{B}\|$ (sub-multiplicativity)

It can be shown that the operator norm of a matrix $\mathbf{M}$, can also be defined as,

$$\|\mathbf{M}\|_p = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{M}\mathbf{x}\|_p}{\|\mathbf{x}\|_p} = \sup_{\|\mathbf{x}\|_p = 1} \|\mathbf{M}\mathbf{x}\|_p. \tag{2.4}$$

Another important norm is the trace norm also called the nuclear norm,

$$\|\mathbf{M}\| = \text{trace}(\sqrt{\mathbf{M}^{\mathrm{T}}\mathbf{M}}) = \sum_{i=1}^{\min\{m,n\}} \sigma_i, \tag{2.5}$$

where σ_i are the singular values of $\mathbf{M}$. The nuclear norm, also known as the Schatten 1-norm is the ℓ_1-norm of the vector of its singular values. Therefore, the sparsity to the vector of singular values are demanded. That is to reduce the rank of the original matrix $\mathbf{M}$. The nuclear norm is widely leveraged as a convex relaxation of the low-rank matrix [136, 138].

Inner Product. An inner product on the vector space $\mathbb{R}^n$ is a function that takes a pair of vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ and outputs a real number. That is, $\langle \mathbf{v}, \mathbf{w} \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. The inner product satisfies three axioms for all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ and scalars $c, d \in \mathbb{R}$,

(i) $\langle c\mathbf{u} + d\mathbf{v}, \mathbf{w} \rangle = c \langle \mathbf{u}, \mathbf{w} \rangle + d \langle \mathbf{v}, \mathbf{w} \rangle$ (bilinearity)

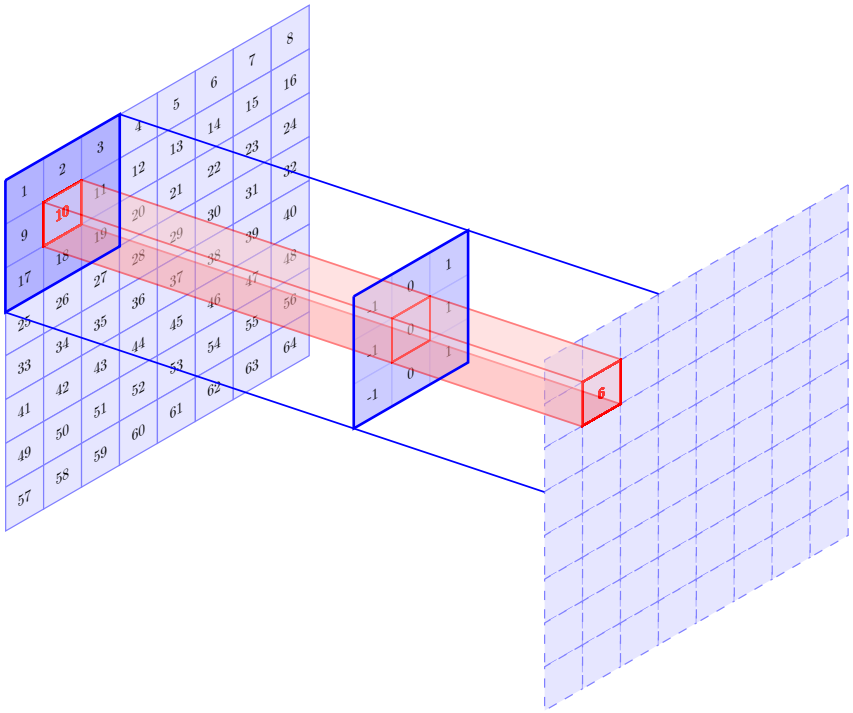


Figure 2.3: The left matrix represents the matrix $\mathbf{I}$ and the middle matrix represents the flipped impulse response of $\mathbf{K}$ in both horizontal and vertical direction. The convolution $\mathbf{M}$ is shown on the right.

(ii) $\langle \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{w}, \mathbf{v} \rangle$ (symmetry)

(iii) $\langle \mathbf{v}, \mathbf{v} \rangle \geq \mathbf{0}$, $\langle \mathbf{v}, \mathbf{v} \rangle = 0 \iff \mathbf{v} = \mathbf{0}$ (positivity)

The inner product produces an associated norm by taking the positive square root of the inner production of the vector with itself,

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}. \tag{2.6}$$

The most common inner product is the dot product,

$$\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v} \cdot \mathbf{w} = v_1 w_1 + v_2 w_2 + \cdots + v_n w_n = \sum_{i=1}^{n} v_i w_i. \tag{2.7}$$

The ℓ_2 norm is found by taking the square root of the inner product of $\mathbf{v}$ with itself,

$$\|\mathbf{v}\|_2 = \sqrt{\mathbf{v} \cdot \mathbf{v}}. \tag{2.8}$$

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

(a) Source image **(b)** LoG filter **(c)** Output image

Figure 2.4: Illustration of image edge detection using convolution with the Laplacian of Gaussian filter.

2D Discrete Convolution. One of the most common operations used in digital image processing and Convolutional Neural Networks (CNNs) is the 2D discrete convolution which extends 1D convolution by convolving both horizontal and vertical directions. Convolution reflects the information by getting the neighbour pixels involved in the calculation. Convolution has many different effects, for instance, denoising, sharpening, edge detection or feature extraction. Given an image $\mathbf{I} \in \mathbb{R}^{m \times n}$ and one finite impulse response $\mathbf{K} \in \mathbb{R}^{(2p+1)\times(2q+1)}$, the 2D discrete convolution of $\mathbf{I}$ and $\mathbf{K}$, $\mathbf{M} = \mathbf{I} * \mathbf{K}$ is defined as,

$$\mathbf{M}_{i,j} = \sum_{a=-p}^{+p} \sum_{b=-q}^{+q} \mathbf{I}_{i-a,j-b} \mathbf{K}_{a,b} \tag{2.9}$$

Equation (2.9) shows that each element of $\mathbf{M}$ is the sum of element-wise multiplication of the impulse response $\mathbf{K}$ with the matrix $\mathbf{I}$. The full convolution is given by sliding the impulse response over the matrix $\mathbf{I}$ (cf. Figure 2.3 for an illustration). In digital image processing convolutional filtering plays an important role in many algorithms. Consider for instance the task of detecting edges in images, that is illustrated in Figure 2.4.

Likewise, 3D convolution applies multi-channel 3D filters to multi-channel 3D images. Given one RGB-image $\mathbf{I} \in \mathbb{R}^{lx \times ly \times lz}$ and one finite impulse response $\mathbf{K} \in \mathbb{R}^{(2o+1)\times(2p+1)\times(2q+1)}$, the 3D discrete convolution of $\mathbf{I}$ and $\mathbf{K}$, $\mathbf{M} = \mathbf{I} * \mathbf{K}$ is defined as,

$$\mathbf{M}_{i,j,k} = \sum_{a=-o}^{+o} \sum_{b=-p}^{+p} \sum_{c=-q}^{+q} \mathbf{I}_{i-a,j-b,k-c} \mathbf{K}_{a,b,c}. \tag{2.10}$$

Convolution is the most important operation of CNNs which are the core of most Machine Learning (ML) systems today.

Smoothing non-smooth Functions. If a function has a uniquely defined first gradient at every point in its domain, then it is referred to as a smooth function. Otherwise it is a non-smooth function. Smoothing a non-smooth function [12, 35, 133] is an important

technique in optimization theory. In this section, we show the technique we used in this thesis to smooth the absolute value function, the hinge loss and the insensitive loss. We define the absolute value function $\mathcal{F}(x) = |x|$. The smooth function of the absolute value function is

$$\mathcal{F}(x) = \begin{cases} \frac{x^2}{2\theta} + \frac{\theta}{2} & |x| \leq \theta \\ |x| & \text{otherwise} \end{cases} \qquad \text{(Huber loss)}$$

where $\theta \in \mathbb{R}^+$ is a parameter deciding the degree of the smoothness.

For the hinge loss, we follow the scheme in [35]. Let the hinge loss be $\mathcal{F}(x) = \max\{x, 0\}$. We define the corresponding smooth function as

$$\mathcal{F}(x) = \begin{cases} 0 & x < -\theta \\ \frac{(x+\theta)^2}{4\theta} & |x| \leq \theta \\ x & x > \theta \end{cases}.$$

Suppose the insensitive loss is defined as $\mathcal{F}(x) = \max(|x| - h, 0), h \in \mathbb{R}^+$. Then the smoothed surrogate is:

$$\mathcal{F}(x) = \begin{cases} \frac{(x-h+\theta)^2}{4\theta} & h - \theta < x < h + \theta \\ 0 & -h + \theta \leq x \leq h - \theta \\ \frac{(-x-h+\theta)^2}{4\theta} & -h - \theta < x < -h + \theta \\ |x| - h & x \in [h + \theta, +\infty] \text{ or } [-\infty, -h - \theta] \end{cases}$$

where $0 < \theta < h$.

2.2 Matrix Calculus

In digital image processing, variables are usually defined as matrices or vector, i.e. that the variables of objective functions (energy functions) are also defined as matrices or vectors. There are rules to calculate various partial derivatives of a single function with respect to multi-variables, that help to find the optimal points of a multi-variate function. In this section, some useful formulas of matrix calculus are collected. They often appear and are used in later derivations.

The Derivatives of Vector Functions. Let $\mathcal{F}$ be a function, $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathbf{x} = [x_1 \ \cdots \ x_n]^\top$ and $\mathbf{y} = [y_1 \ \cdots \ y_m]^\top$ with $\mathbf{y} = \mathcal{F}(\mathbf{x})$. The derivative of $\mathcal{F}$ with respect

$\mathcal{F}$	$\frac{\partial \mathcal{F}}{\partial \mathbf{x}}$
$\mathbf{Ax}$	$\mathbf{A}^\top$
$\mathbf{x}^\top\mathbf{A}$	$\mathbf{A}$
$\mathbf{x}^\top\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^\top\mathbf{Ax}$	$\mathbf{Ax}+\mathbf{A}^\top\mathbf{x}$

Table 2.1: Useful vector derivative formulas.

to vector $\mathbf{x}$ is a $n \times m$ matrix, given as

$$\frac{\mathrm{d}\mathcal{F}}{\mathrm{d}\mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}. \tag{2.11}$$

Table 2.1 summarizes several frequently used vector derivative formulas.

The Chain Rule of Vector Functions. Let $\mathcal{F}$ and $\mathcal{G}$ be two functions, $\mathcal{G}(\mathbf{y}) = \mathcal{G}(\mathcal{F}(\mathbf{x}))$. The derivative of $\mathcal{G}$ with respect to $\mathbf{x}$ is,

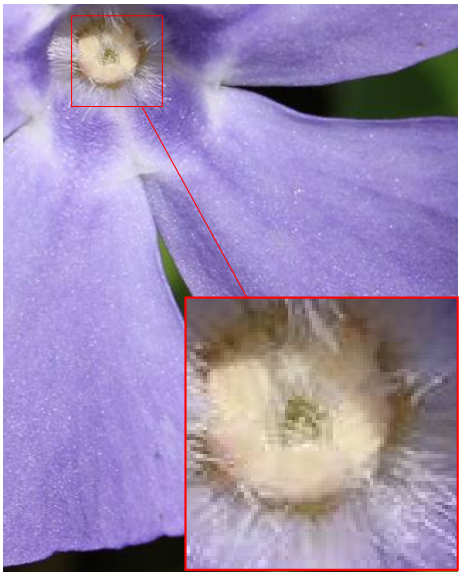
$$\frac{\partial \mathcal{G}}{\partial \mathbf{x}} = \frac{\partial \mathcal{F}}{\partial \mathbf{x}} \frac{\partial \mathcal{G}}{\partial \mathbf{y}}. \tag{2.12}$$

The Derivative of Scalar Functions of a Matrix. Given $\mathbf{X} \in \mathbb{R}^{m\times n}$ and a scalar function $\mathcal{F} : y = \mathcal{F}(\mathbf{X})$. The derivative of $\mathcal{F}$ with respect to the matrix $\mathbf{X}$ is defined as $\frac{\partial \mathcal{F}}{\partial \mathbf{X}} \in \mathcal{R}^{m\times n}$,

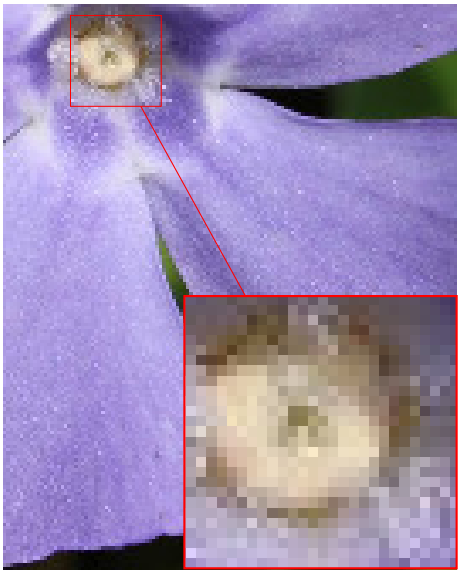
$$\frac{\partial \mathcal{F}}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \frac{\partial y}{\partial x_{12}} & \cdots & \frac{\partial y}{\partial x_{1n}} \\ \frac{\partial y}{\partial x_{21}} & \frac{\partial y}{\partial x_{22}} & \cdots & \frac{\partial y}{\partial x_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{m1}} & \frac{\partial y}{\partial x_{m2}} & \cdots & \frac{\partial y}{\partial x_{mn}} \end{bmatrix}. \tag{2.13}$$

2.3 Inverse Problems in Digital Image Problems

To illustrate inverse vision problems, we considers the problem of Super Resolution (SR) as an example. The central aim of SR is to generate a higher resolution image from one or more lower resolution images. Figure 2.5 illustrates this problem, where a low-resolution image is given and the object is to generate the corresponding high-resolution image. The physical system (black box) is composed of two steps. First the image is captured (with additive sensor noise) and secondly a down-sampling is applied. In mathematical terms

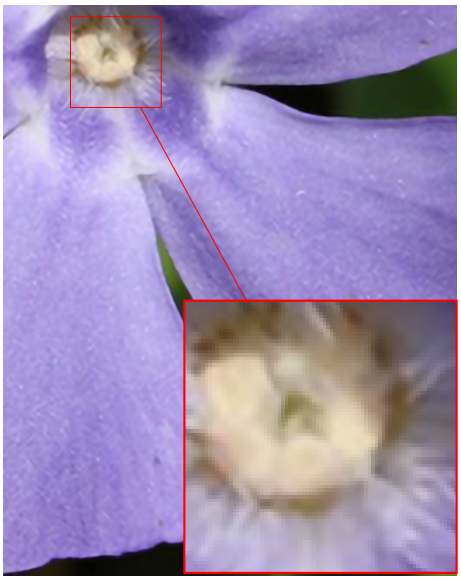


(a) High-resolution image



(b) Low-resolution image

(c) Bicubic upsampled



(d) Superresolved image [152]

Figure 2.5: Illustration of Image Super Resolution. (a) shows the high-resolution image, (b) shows the low-resolution image, (c) shows the Bicubic upsampled version and (d) shows the result of the super-resolution method presented in [152].

this can be written as

$$\mathbf{I} = \mathbf{D}(\mathbf{H} + \mathbf{N}), \tag{2.14}$$

where $\mathbf{I}$ and $\mathbf{H}$ denote the low-resolution image and the high-resolution image respectively. As the names suggest the resolution of $\mathbf{H}$ is higher than the resolution of $\mathbf{I}$. Moreover, we

assume that $\mathbf{H}$ is obtained by a perfect camera. $\mathbf{D}$ is a down-sampling matrix and $\mathbf{N}$ is the random noise arising e.g. during acquisition or transmission.

Suppose $\mathbf{I}, \mathbf{D}$ and $\mathbf{N}$ are given, the object of SR is to recover the high-resolution image $\mathbf{H}$. In this way, Equation (2.14) defines a system of linear equations. Due to the fact, that the resolution of $\mathbf{H}$ is higher than the resolution of $\mathbf{I}$, one obtains an underdetermined system. Hence there are multiple solutions for $\mathbf{H}$, which means that the SR problem is ill-defined. Moreover, we assumed that the noise $\mathbf{N}$ is known, which is usually not the case. There are different ways to model image noise e.g. Gaussian noise, shot noise, salt-and-pepper noise and so on. The problem of estimating or removing the noise $\mathbf{N}$ from an image is called image denoising, which is one of oldest topics in digital image processing. The down-sampling matrix $\mathbf{D}$ can be generated according to an interpolation method. Some typical interpolation methods are Nearest-neighbor interpolation, bilinear interpolation and Bicubic interpolation. In theory, $\mathbf{D}$ can be any reasonable matrix. It is an NP-Hard problem to find a down-sampling matrix which can give an ideal $\mathbf{H}$.

Now the question is how to recovery a reasonable high-resolution image $\mathbf{H}$ or how to choose one which is appealing to the human visual system. In the 1960s, Tikhonov proposed a regularization method to solve underdetermined systems. The regularization method is based on the method of least squares. In order to constrain the ideal solution with desirable properties, a regularization term is added to the sum of squared residuals. In addition, the penalty on the noise matrix $\mathbf{N}$ is added by the constrains. Hence, the Tikhonov regularization method is given as,

$$\mathbf{H}^* = \arg\min_{\mathbf{H}} \|\mathbf{DH} - \mathbf{I}\|_2^2 + \mu\|\nabla\mathbf{H}\|_2^2 \tag{2.15}$$

where ∇ denotes the finite derivative operator and $\mu > 0$ is a weighting parameter. Equation (2.15) can also be interpreted with Bayesian statistics. Let us consider the following version of the Bayes' theorem,

$$\Pr(\mathbf{H}|\mathbf{I}) = \frac{\Pr(\mathbf{I}|\mathbf{H})\Pr(\mathbf{H})}{\Pr(\mathbf{I})} \tag{2.16}$$

where $\Pr(\mathbf{H})$ is called the prior probability, which corresponds to the prior known information. The likelihood that the observation $\mathbf{I}$ can be explained by the high-resolution image $\mathbf{H}$ is denoted as $\Pr(\mathbf{I}|\mathbf{H})$, and is called conditional probability. Note that $\Pr(\mathbf{H})$ and $\Pr(\mathbf{I}|\mathbf{H})$ correspond to the regularization term and data term in Equation (2.15), respectively. The conditional probability $\Pr(\mathbf{H}|\mathbf{I})$ is equal to the marginal probability $\Pr(\mathbf{H})$ multiplied by the conditional probability $\Pr(\mathbf{I}|\mathbf{H})$ and divided by the marginal probability $\Pr(\mathbf{I})$. The evidence is the available information given from an experiment or survey. In Equation (2.16), the term $\Pr(\mathbf{I})$ is a normalizing constant when optimizing $\Pr(\mathbf{H}|\mathbf{I})$. Thus, we can write

$$\Pr(\mathbf{H}|\mathbf{I}) \propto \Pr(\mathbf{I}|\mathbf{H})\Pr(\mathbf{H}). \tag{2.17}$$

Next we take the logarithm of both sides and obtain

$$\log \Pr(\mathbf{H}|\mathbf{I}) \propto \log \Pr(\mathbf{I}|\mathbf{H}) + \log \Pr(\mathbf{H}). \tag{2.18}$$

Equation (2.18) shows that $\log \Pr(\mathbf{I}|\mathbf{H})$ corresponds to $\|\mathbf{DH} - \mathbf{I}\|_2^2$ (data term) and $\log \Pr(\mathbf{H})$ corresponds to $\mu\|\nabla\mathbf{H}\|_2^2$ (regularizer).

2.4 Overview of Machine Learning

ML allows to analyse high dimensional data. The advantage of ML is to automatically detect patterns in the given data, and use those patterns to predict new or unseen data. ML enables computers to make decisions for unseen situations without the dependence to explicitly program.

2.4.1 Evolution of Machine Learning

ML originated from Artificial Intelligence in the 1950s. However, for several years, ML as a statistical method, failed because of limited data. Until the mid-1980s, the endeavour of the researchers generated a surge of interest in ML by the success of backpropagation. In the 1990s ML began to flourish. It benefited from the methods and models of statistics and probability theory and the rising availability of distribution of data on the Internet. Nowadays, ML becomes one of the most important tools of our lifes which enables new waves of intelligent applications and services. The availability of Graphics Processing Units (GPUs) aids the achievement of ML. ML demands a massive amount of computing power for training and GPUs allow to deal with the computation in parallel.

2.4.2 Classification of Machine Learning

A ML method falls into one of four categories. The first leading categories are supervised learning and unsupervised learning. The other two categories are semisupervised learning and reinforcement learning. For any ML algorithm, the purpose is to predict a desired output given an input. The output can be a label in the classification or be a scalar in regression.

Supervised Learning. Supervised ML is the most practical ML algorithm. Supervised learning leverages not only the inputs but also the desired outputs assigned to the inputs. The training set of supervised learning includes both the input and the desired results. The training set corresponds to the knowledge which supervises the ML process.

Supervised learning can be divided into regression and classification. When the output is a categorical variable from a finite set, the problem is regard as classification, for instance, handwriting recognition or image segmentation. On the other side, when the

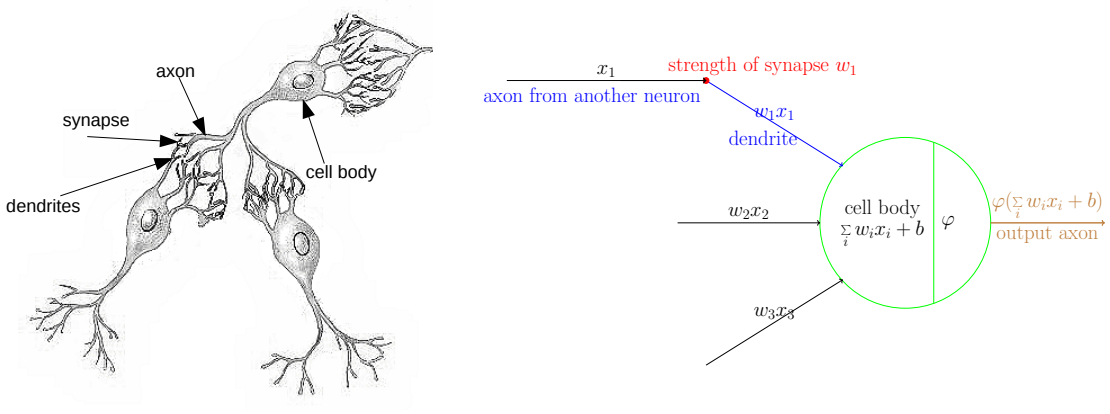
output is a real variable, the problem is known as regression, for instance, Super Resolution or disparity prediction.

Unsupervised Learning. Unsupervised learning is also called knowledge discovery. In the training dataset, there are no outputs available. Unsupervised learning aims to find the hidden patterns in the data. Clustering, dimensionality reduction and association are classical problems of unsupervised learning. A clustering problem is to group the data with the same inherent pattern and structure. Taking e-commerce for instance, the requirement and hobbies of customers are used to cluster products into interesting and non-interesting sets. Projecting high dimension data to a lower dimensional subspace is a useful technique. The learnt lower dimensional subspace can better reflect the pattern of the data. This is because some dimensions do not capture the intrinsic nature of the data, consider for instance, the dimensions arising because of noise. Association is to investigate the relation between different data. Unlike supervised learning, there is no correct answer. Unsupervised learning is to model the distribution in the data and unveil the structure of the data.

Semisupervised Learning. The problems of semisupervised learning [36, 114] are in-between both supervised and unsupervised learning. That is, the training data is partially labelled. In this case, in addition to unlabelled data, the training dataset has some inputs with labels. In semisupervised learning, the labelled data can be treated as constraints and semisupervised learning conducts unsupervised learning with those constrains. In contrast, the distribution of data can be drawn from the unlabelled data and semisupervised learning based on the distribution, conducts the supervised learning. The object of semisupervised learning is to improve the performance of solely using either supervised learning or unsupervised learning. The algorithems of semisupervised learning include self training [110, 139, 174], generative models [63, 122], graph-based algorithms [13, 132] and so forth.

Reinforcement Learning. Reinforcement learning is an inter-discipline involving for instance Optimal Control, Reward System, Bounded Rationality, Machine Learning and Operations Research. In reinforcement learning, there is no external supervisor, i.e. training dataset, but the reward signals or punishment signals. The principle of reinforcement learning is to maximises the accumulative reward in the long term. Reinforcement learning aims to handle interactive problems which are impractical to collect data set. Time plays an important role in reinforcement learning. Feedback signals are delayed and affect the subsequent input data. This is referred to as a trial and error process. Examples of reinforcement learning are playing chess, robot walking and managing an investment portfolio.

2.4.3 Artificial Neural Networks



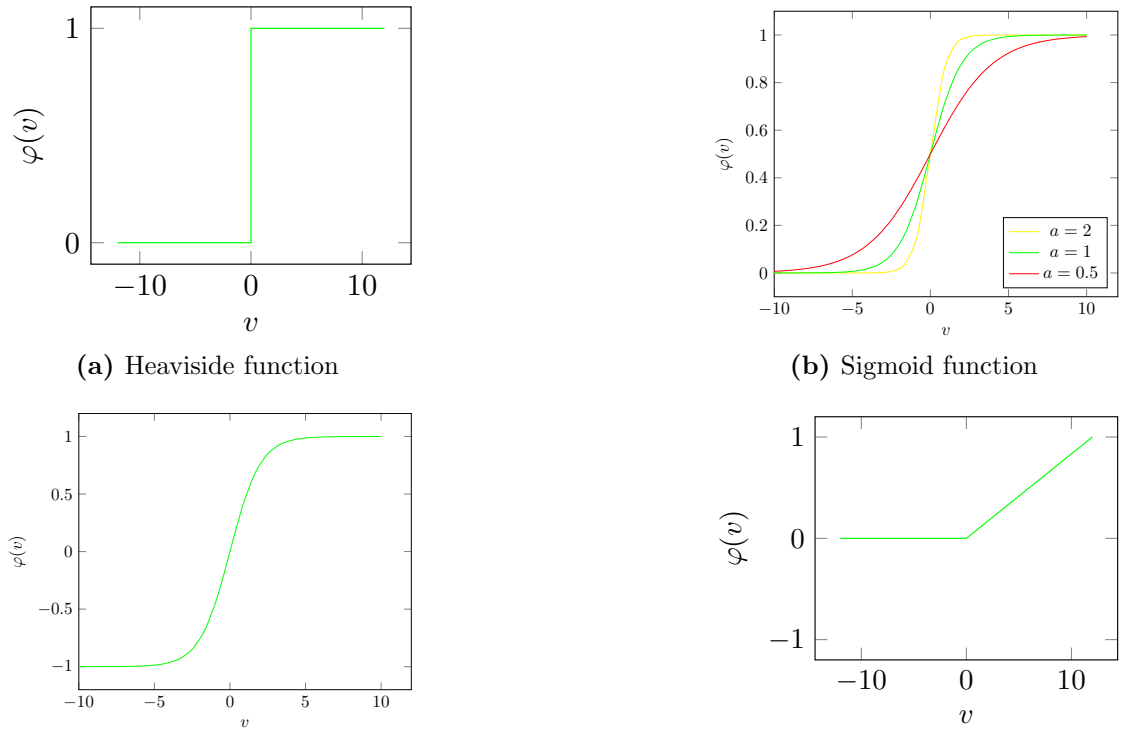
(a) Sketch of a biological neuron **(b)** Artificial neuron

Figure 2.6: Comparison between a biological neuron and an artificial neuron.

As one of the most important ML algorithms, artificial Neural Network (NN)s, also referred to as "neural networks", are biologically-inspired by the human brain. The human brain has roughly 100 billion neurons which are the basic building components of the nervous system [80]. And each neuron connects to between 5,000 and 10,000 other neurons, passing signals to each other via up to 1000 trillion synapses. The human brain normally is considered as a vast parallel and complex system. Dendrites are the receivers of input signals. And axons are in charge of sending the output signals. Axons branch out and connect by synapses to dendrites of other neurons. While scientists are still investigating how the brain develops and can be trained, artificial NNs are already designed in hardware or simulated in software to deal with tasks of pattern recognition or perception. One definition of a NN is given by Haykin

> A NN is a massive parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:
>
> 1. Knowledge is acquired by the network from its environment through a learning process.
> 2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.
>
> (Haykin [73])

Activation Functions. A synapse strength or synapse weight w_i defines the interaction between its axon and the next neurons. The signals x_i travel from axons to the dendrites of the other neurons. In artificial NNs, synaptic strengths are learned



(a) Heaviside function

(b) Sigmoid function

(c) Hyperbolic tangent

(d) Rectified Linear Unit

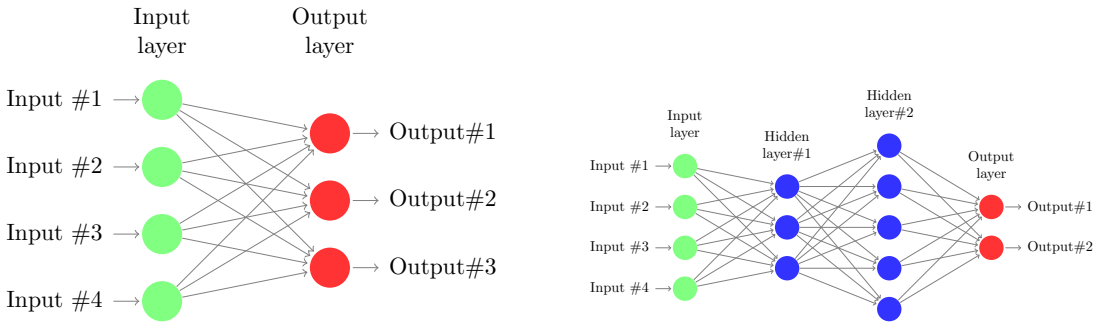
Figure 2.7: Illustration of common activation functions. (a) shows the threshold or Heaviside function, (b) shows the sigmoid function for different settings of the parameter a (cf. Equation (2.19)), (c) shows the hyperbolic tangent and (d) illustrates the Rectified Linear Unit.

in a way such that the outputs of artificial NNs are as correct as possible. A neuron sum up all input signals weighted by the respective synaptic weights in addition with a bias b. An activation function φ applies to the summation before outputting the signal, $\varphi\left(\sum_i w_i x_i + b\right)$. Figure 2.6b shows an example of a nonlinear model of a neuron with three input signals. One common activation function φ is the threshold function which is defined as

$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}.$$

The threshold function, as shown in Figure 2.7a, is also referred to as the Heaviside function. The Perceptron as the simplest NN just uses a threshold function as the activation function. If the output is 1, it means the neuron fires, otherwise, it has a negative output 0. A differentiable approximation of a threshold function is a sigmoid function. The sigmoid function also ranges from 0 to 1, and is defined as

$$\varphi(v) = \frac{1}{1 + e^{-av}}, \tag{2.19}$$



(a) A single-layer feedforward networks **(b)** A multilayer feedforward networks

Figure 2.8: Illustration of single-layer and multilayer feedforward networks. (a) depicts a single-layer feedforward network which only has an input layer and an output layer. (b) shows a multilayer feedforward network with two hidden layers.

where $a > 0$ is a scalar to control the slope. Examples are shown in Figure 2.7b. The sigmoid function was frequently used and is currently out of favour. One reason is that the output of a sigmoid function is not zero-centered which slows down the training. The second reason is the sigmoid function is not robust to the initialization because of its saturation effect. When the output is either 0 or 1, the gradient is zero and the NN is hard to learn. To overcome the nonzero-centered problem, the sigmoid function is replaced by the hyperbolic tangent function

$$\varphi(v) = \frac{2}{1 + e^{-2v}} - 1.$$

Figure 2.7c shows that the range of the hyperbolic tangent function is $[-1, 1]$. The other increasing popular activation function is the Rectified Linear Unit (ReLU) as shown in Figure 2.7d. To some extent, the ReLU overcomes the problem of saturation. In addition, the ReLU is less time consuming, i.e. the calculation is more efficient.

Single-layer and Multilayer Feedforward Networks. According to the number of hidden layers, NNs fall into two categories, single-layer and multilayer feedforward networks as shown in Figure 2.8. Every NN has one input layer and one output layer. The hidden layers are composed of hidden neurons or hidden units, which are not involved in the input or output layers. The hidden layers increase the complexity of NNs and enable NNs to accomplish more difficult tasks. The structure of a NN mainly depends on the problem to be solved. Take handwriting recognition for instance [MNIST]. This is a classification problem where there are ten digital numbers. The size of each training sample in the MNIST dataset is 28×28. And it can be formulated as a vector with length 784. Thus each component of the vector is one input signal. The output signal can be arranged to be a vector of length 10, indicating the probability of each class (i.e. digit). Example images for this NN are illustrated in Figure 2.9. The network in Figure 2.10 is

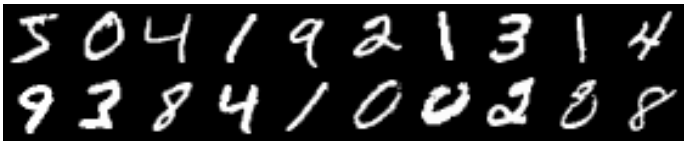


Figure 2.9: Illustration of the MNIST dataset. The figure shows 20 images of size 28×28.

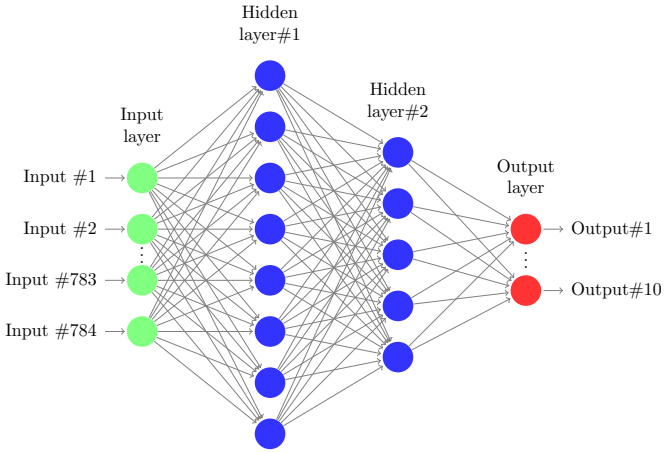


Figure 2.10: A Neural Network for handwriting recognition.

referred to as a 784-8-5-10 network because it has 784 input signals, 2 hidden layers with 8 neurons in the first hidden layer and 5 neurons in the second hidden layer and 10 output neurons.

The examples of NNs shown above are fully connected networks such that every neuron in each layer is connected to every other neuron in the next layer. A network is called partially connected if any synaptic connections are absent in the network.

Recurrent Neural Network. In feedforward neural networks, data streams flow unidirectionally from an input layer through hidden layers to an output layer. No data stream forms cycles in the feedforward NN topology. But the human brain is a network of neurons with feedback connections. In the light of the biological structure, Recurrent Neural Networks (RNNs) attract a lot of attention especially for solving ML problems. RNNs are similar to feedforward neural networks in the sense that they map inputs to outputs, with or without supervision. But to make RNNs biologically more plausible and adaptive, RNNs allow any neuron connect to any other, even to itself. This introduces the concept of time in RNNs. There are discrete RNNs over discrete time steps and continuous RNNs over a continuous time t.

Convolutional Neural Networks. In the late 1980s, Yann LeCun et al.[98, 99] propose a special type of multi-layer NNs, where weights are shared across layers. To share the weights, they conduct convolutions, which is the key operation in the CNN architecture. CNNs consist of one or more convolutional layers, where the different layers are connected such that layer l is the input for layer $l+1$. The convolutional layer l can be seen as the convolution result from layer $l-1$ and is the input of layer $l+1$. CNNs are inspired by the visual cortex of monkeys [84] and have a sparse solution by sharing weights. The cells in the visual cortex play a role as convolution filters, to detect lines for instance. CNNs are partially connected which are efficient for processing images. We take a simple example of a 1D CNN with only one convolutional layer. As shown in Figure 2.11, this is

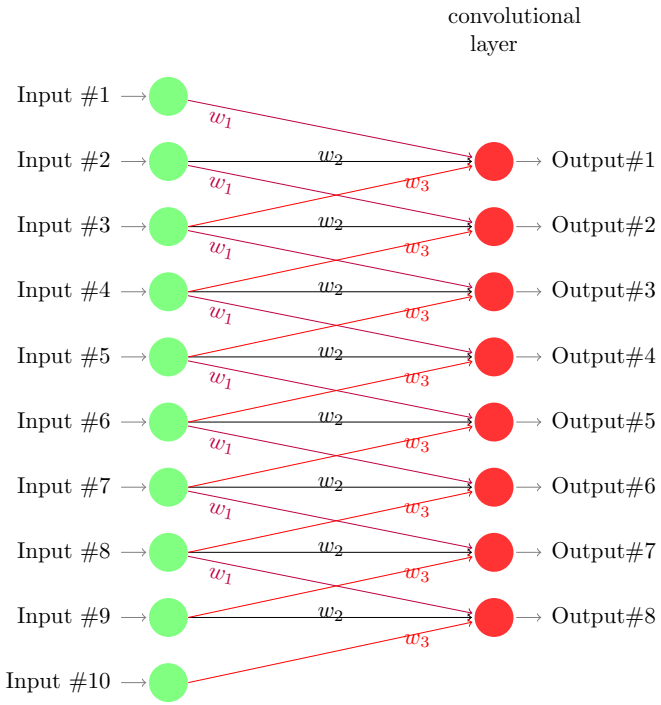


Figure 2.11: Illustration of a 1D Convolutional Neural Network. The input signal is 1D $\in \mathbb{R}^{10}$. The convolutional layer is the convolution result of the input signal with a 1D filter $\in \mathbb{R}^3$.

a partially connected network and shares a filter $\mathbf{f} = [w_1, w_2, w_3]$. The filter can be used to extract the structure of the input signals. For a fully connected network with the same size 10-8, there are 80 weights. Thus, CNNs can provide a practical solution especially for high-resolution images in which complex fully connected networks suffer from the big data.

2.5 Isotropic & Anisotropic Diffusion

Diffusion describes the process of density changes. In isotropic diffusion, the propagation of diffusion in each direction is the same. Let $\mathcal{U} : \mathbb{R}^n \times t \to \mathbb{R}$ be a function where $\mathbb{R}^n$ is the spacial domain and t is the time domain. The isotropic diffusion is defined as

$$\frac{\partial \mathcal{U}}{\partial t} = \operatorname{div}\left(d \nabla \mathcal{U}\right) = d \nabla^2 \mathcal{U}, \tag{2.20}$$

where d is called a diffusion coefficient and div is the divergence operator. The most simple example of isotropic diffusion is the heat equation,

$$\frac{\partial \mathcal{U}}{\partial t} = k \nabla^2 \mathcal{U}, \tag{2.21}$$

where k is a positive constant called the thermal diffusivity and ∇^2 denotes the Laplace operator. Suppose the space domain is one dimensional ($x \in \mathbb{R}$), then Equation (2.21) becomes

$$\frac{\partial \mathcal{U}}{\partial t} = k \frac{\partial^2 \mathcal{U}}{\partial x^2}, \tag{2.22}$$

which measures the thermal conductivity given a specific material. The larger the thermal diffusivity k is the better a material will conduct heat. Let the initial condition be $\mathcal{U}\left(x, 0\right)$, then the solution of Equation (2.22) is given as

$$\mathcal{U}\left(x, t\right) = \int_{-\infty}^{\infty} \mathcal{U}\left(s, 0\right) \mathcal{H}_t\left(x - s\right) ds, \tag{2.23}$$

where $\mathcal{H}_t$ is a Gaussian kernel

$$\mathcal{H}_t = \frac{1}{4\pi k t} e^{-x^2/4kt}, \ t > 0.$$

Equation (2.23) is the spatial convolution of function $\mathcal{U}\left(s, 0\right)$ and $\mathcal{H}_t(x)$. Likewise the 2D heat equation undergoes a 2D convolution of the initial 2D signal with a Gaussian filter. In image processing this is the famous Gaussian denoising. It is well-known that the Gaussian denoising blurs the image and cannot preserve edges well. To avoid blurring the image edges one could use anisotropic diffusion where diffusion coefficients along different directions are adjusted based on e.g. the image content, leading to

$$\frac{\partial \mathcal{U}}{\partial t} = \operatorname{div}\left(\mathcal{D} \nabla \mathcal{U}\right), \tag{2.24}$$

where $\mathcal{D} : (\mathbb{R}^n \times t) \to \mathbb{R}^{(n\times n)}$ is called the diffusion tensor. Take a 3D example, $\mathcal{D}$ can be

$$\mathcal{D}(x,y,z,t) = \begin{bmatrix} \mathcal{U}_{xx}(x,y,z,t) & \mathcal{U}_{xy}(x,y,z,t) & \mathcal{U}_{xz}(x,y,z,t) \\ \mathcal{U}_{yx}(x,y,z,t) & \mathcal{U}_{yy}(x,y,z,t) & \mathcal{U}_{yz}(x,y,z,t) \\ \mathcal{U}_{zx}(x,y,z,t) & \mathcal{U}_{zy}(x,y,z,t) & \mathcal{U}_{zz}(x,y,z,t) \end{bmatrix}.$$

Perona-Malik equations often used in image denoising, defines $\mathcal{D}(x,y,z,t)$ as

$$\mathcal{D}(x,y,z,t) = e^{-(\|\nabla \mathcal{U}\|/k)^2} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

or

$$\mathcal{D}(x,y,z,t) = \frac{1}{1 + \left(\frac{\|\nabla \mathcal{U}\|}{k}\right)^2} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

where k is a constant controlling the sensitivity to edges. A large $\nabla \mathcal{U}$, that will occur for instance at edges or corners, will lead to a small $\mathcal{D}(x,y,z,t)$.

2.6 Fields of Experts

The Fields of Experts (FoE) [140, 142] is a high-order Markov random filed and is successfully used in many applications of image processing, for instance, image denoising and image inpainting. The FoE is an adaptation of the Product of Experts (PoE) [82, 83] model. A PoE normalizes the product of a number of single component models, also called experts. In PoE, experts usually are defined on a low-dimensional subspace. Consider for example the 2D image processing, where a 1D subspace is frequently used. Let an image patch $\mathbf{x}$ be a vector of length n, $\mathbf{x} \in \mathbb{R}^n$. Suppose that there are m experts whose distributions are defined on $\mathbf{w}_j^\top \mathbf{x},\ j = 1, \cdots, m$. The basis vector, $\mathbf{w}_j \in \mathbb{R}^n$ defines the projection onto the subspace. We denote by the function Φ the j^{th} expert and $\boldsymbol{\alpha}_j$ are the associated parameters, then the PoE is given as

$$\mathcal{P}(\mathbf{x}) = \frac{1}{\mathcal{Z}(\Theta)} \prod_{j=1}^{m} \Phi\left(\mathbf{w}_j^\top \mathbf{x}; \boldsymbol{\alpha}_j\right), \tag{2.25}$$

where $\Theta = \{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_m\}$ with $\boldsymbol{\theta}_j = \left\{\boldsymbol{\alpha}_j, \mathbf{w}_j^\top\right\}$ and $\mathcal{Z}(\Theta) = \int \prod_{j=1}^{m} \Phi\left(\mathbf{w}_j^\top \mathbf{x}; \boldsymbol{\alpha}_j\right) \mathrm{d}\mathbf{x}$ is the normalization. Since $\mathbf{w}_j$ and $\mathbf{x}$ are the same size, it is reasonable to apply PoE to the small patches rather than a whole image. FoE is proposed to solve this problem. FoE is able to deal with any arbitrary image size. In FoE the basic components are pixels in an image and $\mathbf{w}_j$ is considered as a linear filter. Without loss of generality let $\mathbf{x}_{(k)}$ be a vectorized square image patch centered at x_k among all pixels $k = 1, \cdots, n$, then the FoE

is given as

$$\mathcal{P}(\mathbf{x}) = \frac{1}{\mathcal{Z}(\Theta)} \exp\left(-\mathrm{E}_{\mathrm{FoE}}\left(\mathbf{x}, \Theta\right)\right) \text{ with } \mathrm{E}_{\mathrm{FoE}}\left(\mathbf{x}, \Theta\right) = -\sum_{i=1}^{n}\sum_{j=1}^{m} \log \Phi_j\left(\mathbf{w}_j^\top \mathbf{x}_{(i)}; \boldsymbol{\alpha}_j\right). \tag{2.26}$$

$\mathbf{w}_j^\top \mathbf{x}_{(i)}$ can be considered as the i^{th} pixel of the convolution of $\mathbf{w}_j$ and the image $\mathbf{x}$. Equation (2.26) can be simplified to

$$\mathcal{P}(\mathbf{x}) = \frac{1}{\mathcal{Z}(\Theta)} \prod_{i=1}^{n}\prod_{j=1}^{m} \Phi_j\left(\mathbf{w}_j^\top \mathbf{x}_{(i)}; \boldsymbol{\alpha}_j\right). \tag{2.27}$$

Motivated by the property of natural images which have heavy-tailed marginal distributions, Roth and Black suggest two experts: (i) a Student t-distribution, and (ii) a less heavy-tailed expert based on the ℓ_1 norm. A Student t-distribution has the following formulation,

$$\Phi_{stu}\left(\mathbf{w}_{stu}^\top \mathbf{x}_{(i)}; \boldsymbol{\alpha}_{stu}\right) = \left(1 + \frac{1}{2}\left(\mathbf{w}_{stu}^\top \mathbf{x}_{(i)}\right)^2\right)^{-\boldsymbol{\alpha}_{stu}}. \tag{2.28}$$

The Student t-distribution has been successfully used in the PoE for patch-based problems. For the less heavy-tailed expert, Roth and Black leveraged the "smooth" penalty function [37] which results in

$$\Phi_{ht}\left(\mathbf{w}_{ht}^\top \mathbf{x}_{(i)}; \alpha_1, \alpha_2\right) = e^{-\alpha_1 \sqrt{\alpha_2 + \left(\mathbf{w}_{ht}^\top j \mathbf{x}_{(i)}\right)^2}}. \tag{2.29}$$

Image Denoising using FoE Roth and Black tried to recovery a denoised image $\mathbf{x}$ which maximizes the following posterior probability given the observed image $\mathbf{y}$, which is corrupted by Gaussian noise with zero mean and known standard deviation σ.

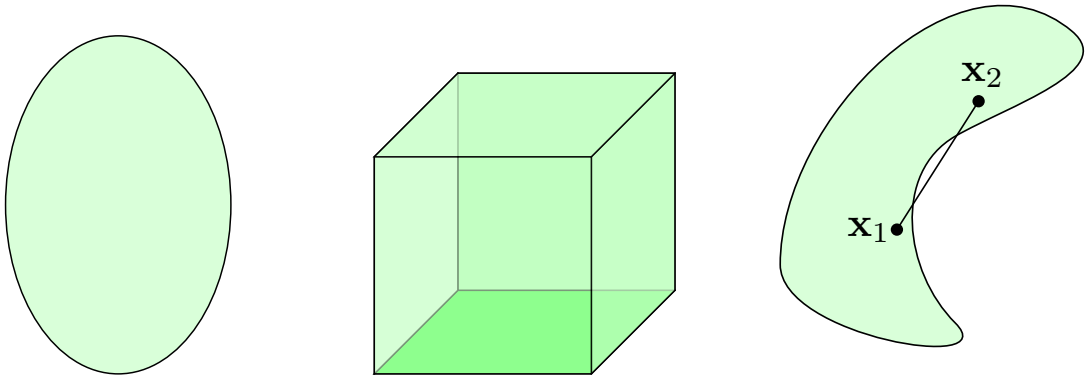
$$\mathcal{P}(\mathbf{x}|\mathbf{y}) \propto \mathcal{P}(\mathbf{y}|\mathbf{x}) \cdot \mathcal{P}(\mathbf{x}). \tag{2.30}$$

Roth and Black formulate the likelihood as

$$\mathcal{P}(\mathbf{y}|\mathbf{x}) \propto \prod_i e^{-\frac{1}{2\sigma^2}(y_i - x_i)^2}, \tag{2.31}$$

among all pixels $i = 1, \cdots, n$. Calculating the gradient ascent on the logarithm of the posterior probability allows to simplify the calculation. Hence, one gets

$$\frac{\partial \log \mathcal{P}(\mathbf{y}|\mathbf{x})}{\partial \mathbf{x}} = \frac{1}{\sigma^2}(\mathbf{y} - \mathbf{x}). \tag{2.32}$$



(a) 2D convex set **(b)** 3D convex set **(c)** 2D non-convex set

Figure 2.12: Illustration of convex and non-convex sets. (a) and (b) show convex sets in 2D and 3D respectively. (c) is an example of non-convex set. Note that the line segment that joins $\mathbf{x}_1$ and $\mathbf{x}_2$ is not entirely included in the set.

In addition, the gradient of the logarithm prior probability leads to.

$$\frac{\partial \log \mathcal{P}(\mathbf{x})}{\partial \mathbf{x}} = \sum_{j=1}^{m} \overline{\mathbf{J}}_j * \boldsymbol{\psi}_j \left(\mathbf{J}_j * \mathbf{x}\right) \tag{2.33}$$

where $\mathbf{J}_j * \mathbf{x}$ denotes the convolution of $\mathbf{x}$ with filter $\mathbf{J}_i$, $\overline{\mathbf{J}}_j$ is obtained by rotating $\mathbf{J}_j$ 180 degrees and $\boldsymbol{\psi}_j\left(\mathbf{y}\right) = \frac{\partial \Phi_j(\mathbf{y};\boldsymbol{\alpha}_j)}{\partial \mathbf{y}}$. Based on the aforementioned gradients, the final gradient ascent denoising algorithm of FoE leads to

$$\frac{\partial \mathbf{x}}{\partial t} = \sum_{j=1}^{m} \overline{\mathbf{J}}_j * \boldsymbol{\psi}_j \left(\mathbf{J}_j * \mathbf{x}\right) + \frac{\lambda}{\sigma^2}(\mathbf{y} - \mathbf{x}), \tag{2.34}$$

where λ is a weight parameter.

2.7 Optimization Algorithms

Convex Sets. Most ML problems can be formulated as large-scale optimization problems. Hence optimization is of crucial importance in ML. In practice, current problems of ML often involve a massive and complex data set. Thus, efficiency of the optimization algorithm is highly demanded. In addition, ML problems are usually cast as non-convex optimization problems which increases the challenges. This section briefly describes the differences between convex and non-convex optimization. Further, we review important definitions and theories, which are the foundation of many state-of-the-art optimization algorithms.

Convex and non-convex sets are both defined by the categories of curvature. Convex means that the curvature expands outwards, and non-convex means the a curvature is bending inwards. A convex set C is a region such that any line segment between any two points in C also lies in C. In mathematical terms this means that, for any $\mathbf{x}_1, \mathbf{x}_2 \in C$ and any $\theta \in \mathbb{R}$ with $0 \leq \theta \leq 1$, the follow holds,

$$\theta \mathbf{x}_1 + (1 - \theta)\mathbf{x}_2 \in C. \tag{2.35}$$

A set is non-convex if for any pair of points lying in the set, there exists at least one corresponding line segment, which is not entirely in the set. A non-convex set is also sometimes referred to as a concave set. Some typical examples of convex and non-convex sets are shown in Figure 2.12.

Convex Optimization. Convex optimization is a subfield of optimization that considers the problems of minimizing convex functions over convex sets. A convex optimization problem has the following form

$$\begin{aligned} \text{minimize} \quad & \mathcal{F}_0(\mathbf{x}) \\ \text{subject to} \quad & \mathcal{F}_i(\mathbf{x}) \leq b_i, \ i = 1, \cdots, m, \\ \text{and} \quad & \mathcal{H}_i(\mathbf{x}) = 0, \ i = 1, \cdots, p, \end{aligned} \tag{2.36}$$

where the functions $\mathcal{F}_0, \cdots, \mathcal{F}_m : \mathbb{R}^n \to \mathbb{R}$ are convex functions, the functions $\mathcal{H}_0, \cdots, \mathcal{H}_p : \mathbb{R}^n \to \mathbb{R}$ are affine functions. Affine functions are expressed in the standard form

$$\mathcal{H}_i(\mathbf{x}) = \mathbf{a_i}^\top \mathbf{x} + b_i, \tag{2.37}$$

where $\mathbf{a}_i$ is a vector and b_i is a scalar. In the literature, $\mathcal{F}_0$ is referred to as the objective function, the energy function or the cost function.

Convex Functions. Convex functions satisfy the following constrains,

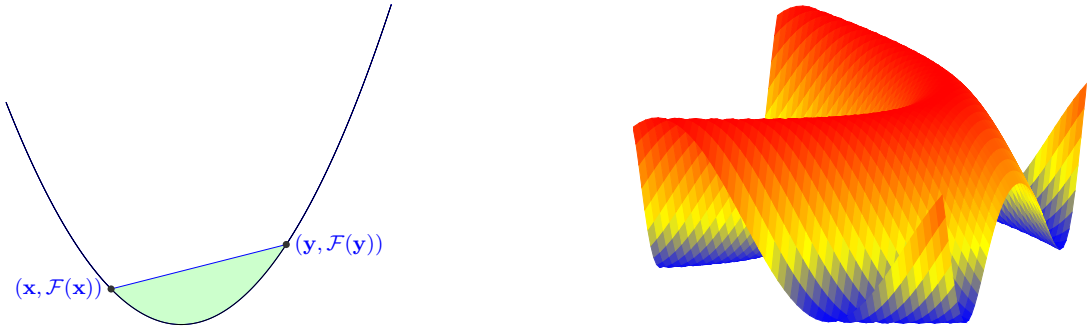
$$\mathcal{F}_i(\alpha \mathbf{x} + \beta \mathbf{y}) \leq \alpha \mathcal{F}_i(\mathbf{x}) + \beta \mathcal{F}_i(\mathbf{y})$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$ and the domain of $\mathcal{F}$ is a convex set. Plug the constraint $\alpha + \beta = 1$ in the equation and get,

$$\mathcal{F}_i(\alpha \mathbf{x} + (1 - \alpha)\mathbf{y}) \leq \alpha \mathcal{F}_i(\mathbf{x}) + (1 - \alpha)\mathcal{F}_i(\mathbf{y}).$$

This means the line segment between $(\mathbf{x}, \mathcal{F}(\mathbf{x}))$ and $(\mathbf{y}, \mathcal{F}(\mathbf{y}))$ is above the graph of $\mathcal{F}$, cf. Figure 2.13a for an illustration.

Optimization problems that violate one or more conditions of the problem in Equation (2.36) are called non-convex problem (cf. Figure 2.13b for an illustration).



(a) Convex function **(b)** Non-convex function

Figure 2.13: Illustration of convex and non-convex functions. (a) shows a convex function in 1D , and (b) shows a non-convex function of two variables.

The feasible set X of the convex optimization problem in Eqn. (2.36) is given as

$$X := \{\mathbf{x} \in \mathbb{R}^n : \mathcal{F}_i(\mathbf{x}) \leq b_i,\ i = 1, \cdots, m, \quad \mathcal{H}_i(\mathbf{x}) = 0,\ i = 1, \cdots, p\}\,. \tag{2.38}$$

Usually the object of an optimization problem is to seek minimizers which define the so-called optimal set X^{opt}. The optimal set is a subset of the feasible set and it meets the following additional condition,

$$X^{\text{opt}} = \arg\min \mathcal{F}_0(\mathbf{x}). \tag{2.39}$$

A vector $\mathbf{x}$ is referred to as an optimal solution if $\mathbf{x} \in X^{\text{opt}}$. If the optimization problem is infeasible, then the optimal set is empty. This can be due to the fact that the feasible set is empty or the optimal solution is only attained in the limit. Take $\min_x e^{-x}$ for instance, the optimal value 0 is only achieved in the limit $x \to +\infty$.

A vector $\mathbf{z}$ is called local optimal if there is a scaler $r \geq 0$ such that $\mathbf{z}$ is optimal for the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \mathcal{F}_0(\mathbf{x}) \\ \text{subject to} \quad & \mathcal{F}_i(\mathbf{x}) \leq b_i,\ i = 1,\ \cdots, m, \\ \text{and} \quad & \mathcal{H}_i(\mathbf{x}) = 0,\ i = 1, \cdots, p \\ \text{and} \quad & \|\mathbf{z} - \mathbf{x}\|_2 \leq r. \end{aligned} \tag{2.40}$$

A local optimal solution is only optimal for nearby elements in the feasible set which may not be the optimal value of the original optimization problem. An optimal solutions of the original optimization problem is denoted as a global optimum. In convex optimization, a local optimal solution is also the global optimal solution. But in non-convex optimization,

a local optimal solution is not necessarily global optimal.

Convex optimization is a well studied field since the 1940s and has a rigorous mathematical foundation. Convex optimization problems can be solved efficiently up to a large size, say, millions of variables or constraints, because of their property that any local optimal solution is also global optimal. However, convex optimization only makes up a small section of all optimization problems. Left are non-convex optimization problems from economic to scientific field. Non-convex optimization problems are usually way harder to solve than convex ones if at all. There exist only a few non-convex optimization problems that can be solved exactly and attain a global solution [72].

The Lagrange Dual Function. Let $\mathbf{x} \in \mathbb{R}^n$ and suppose that the feasible set of the following minimization problem is nonempty,

$$\begin{aligned} \text{minimize} \quad & \mathcal{F}_0(\mathbf{x}) \\ \text{subject to} \quad & \mathcal{F}_i(\mathbf{x}) \leq 0, i = 1, \cdots, m, \\ \text{and} \quad & \mathcal{H}_i(\mathbf{x}) = 0, i = 1, \cdots, p, \end{aligned} \tag{2.41}$$

by integrating the constraints in Equation (2.41) into the objective function, the Lagrangian $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{v}) = \mathcal{F}_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \mathcal{F}_i(\mathbf{x}) + \sum_{i=1}^{p} v_i \mathcal{H}_i(\mathbf{x}). \tag{2.42}$$

where λ_i is the i^{th} component of $\boldsymbol{\lambda}$ and is refer to as the Lagrange multiplier for the i^{th} inequality constraint $\mathcal{F}_i(\mathbf{x}) \leq 0$. Likewise, v_i is the Lagrange multiplier of the i^{th} equality constraint $\mathcal{H}_i(\mathbf{x}) = 0$. In Equation (2.42), the vectors $\boldsymbol{\lambda}$ and $\mathbf{v}$ are referred to as dual variables or Lagrange multiplier vectors.

Given a Lagrangian $\mathcal{L}$, the Lagrange dual function $\mathcal{D} : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is a function of the dual variables and is given as,

$$\mathcal{D}(\boldsymbol{\lambda}, \mathbf{v}) = \inf_{\mathbf{x} \in \mathbb{R}^n} \left\{ \mathcal{F}_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \mathcal{F}_i(\mathbf{x}) + \sum_{i=1}^{p} v_i \mathcal{H}_i(\mathbf{x}) \right\}, \tag{2.43}$$

which is a concave function because it is the pointwise infimum of affine functions. The optimal value $\widehat{\mathcal{F}_0}$ is equal or larger than $\mathcal{D}(\boldsymbol{\lambda}, \mathbf{v})$, $\forall \boldsymbol{\lambda} \succcurlyeq 0$, $\forall \mathbf{v}$, i.e. $\mathcal{D}(\boldsymbol{\lambda}, \mathbf{v}) \leq \widehat{\mathcal{F}_0}$. See [23] for a proof. Hence, a lower bound on $\widehat{\mathcal{F}_0}$ is given by $\mathcal{D}(\boldsymbol{\lambda}, \mathbf{v})$ when $\forall \boldsymbol{\lambda} \succcurlyeq 0$, $\forall \mathbf{v}$. Among different combinations of $(\boldsymbol{\lambda}, \mathbf{v})$, the following Lagrange dual problem results in the best lower bound of $\mathcal{F}_0(\mathbf{x})$,

$$\begin{aligned} \text{maximize} \quad & \mathcal{D}(\boldsymbol{\lambda}, \mathbf{v}) \\ \text{subject to} \quad & \lambda \succcurlyeq 0. \end{aligned} \tag{2.44}$$

We denote by $(\widehat{\boldsymbol{\lambda}}, \widehat{\mathbf{v}})$ an optimal solution of the Equation (2.44). The best lower bound on

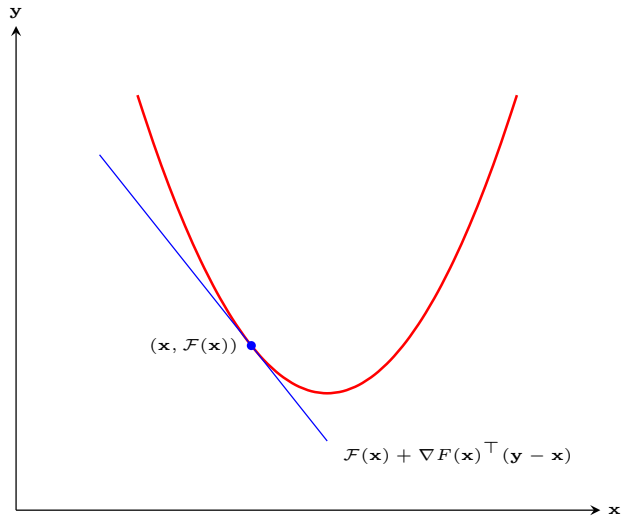


Figure 2.14: Illustration of the first-order condition of a convex function. If $\mathcal{F}$ is convex and differentiable, then $\mathcal{F}(\mathbf{x}) + \nabla\mathcal{F}(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) \leq \mathcal{F}(\mathbf{y})$ for all $\mathbf{x}$, $\mathbf{y}$ in the domain of $\mathcal{F}$.

$\widehat{\mathcal{F}_0}$ is hence given as the optimal value of the Lagrange dual problem, $\hat{d} = \mathcal{D}(\widehat{\boldsymbol{\lambda}}, \widehat{\mathbf{v}})$. Also note that weak duality states that $\hat{d} \leq \widehat{\mathcal{F}_0}$ and strong duality states that $\hat{d} = \widehat{\mathcal{F}_0}$, i.e. the optimal duality gap $\widehat{\mathcal{F}_0} - \hat{d}$ is zero. Strong duality does not always hold. One important sufficient condition in convex optimization for strong duality to hold is Slater's condition. Slater's condition implies that strong duality holds when there exists an $\mathbf{x}$ lying in the relative interior of the feasible set and $\mathcal{F}_i(\mathbf{x}) < 0,\ i = 1, \cdots, m$.

Verifying Convexity of a Function. In addition to verifying the definition of a convex function, there are other more convenient methods for this task. In the following, we briefly outline some of those methods.

First-order condition: Let a function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$ be differentiable. $\mathcal{F}$ is referred to as a convex function if and only if the domain of $\mathcal{F}$ is convex and for all $\mathbf{x}, \mathbf{y}$ in the domain of $\mathcal{F}$, the following equation holds

$$\mathcal{F}(\mathbf{x}) + \nabla\mathcal{F}(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) \leq \mathcal{F}(\mathbf{y}),$$

where

$$\nabla\mathcal{F}(\mathbf{x}) = \begin{bmatrix} \frac{\partial\mathcal{F}(\mathbf{x})}{\partial x_1} & \frac{\partial\mathcal{F}(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial\mathcal{F}(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

is the gradient of $\mathcal{F}(\mathbf{x})$. Figure 2.14 illustrates this important condition, which also shows that for a convex function $\mathcal{F}$ one can extract global information, in terms of a global underestimator from the local neighbourhood of $\mathbf{x}$.

Second-order condition: Let a function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable with convex

domain. $\mathcal{F}$ is convex if and only if the Hessian $\nabla^2\mathcal{F}(\mathbf{x})$ is positive semidefinite for all $\mathbf{x}$ in the domain of $\mathcal{F}$, where $\nabla^2\mathcal{F}(\mathbf{x})$ is a symmetric $n \times n$ matrix whose components are the second-order partial derivatives of $\mathcal{F}$,

$$\left[\nabla^2\mathcal{F}(\mathbf{x})\right]_{i,j} = \frac{\partial^2\mathcal{F}(\mathbf{x})}{\partial x_i \partial x_j}, \quad \text{with} \;\; i,\; j = 1, \cdots, n.$$

Composition with scalar functions: Let $\mathcal{G} : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{H} : \mathbb{R} \to \mathbb{R}$ be two functions, and $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathcal{G}(\mathbf{x}))$. Then $\mathcal{F}$ is convex if

(i) $\mathcal{G}$ is convex and $\mathcal{H}$ is nondecreasing and convex,

(ii) $\mathcal{G}$ is concave and $\mathcal{H}$ is nonincreasing and convex.

Operations that preserve convexity: $\mathcal{F}$ is convex if it is obtained by operations that can preserve convexity, like for example:

(i) Non-negative weighted sum: If all $\mathcal{F}_i$ are convex, then the function $\mathcal{F} = \sum_i \alpha_i \mathcal{F}_i,\; \alpha_i > 0$ is convex.

(ii) Composition with affine functions: If $\mathcal{F}$ is convex, then the function $\mathcal{F}(A\mathbf{x} + \mathbf{b})$ is convex.

(iii) Pointwise maximum: The function $\mathcal{F}(\mathbf{x}) = \max\{\mathcal{F}_1(\mathbf{x}), \cdots, \mathcal{F}_m(\mathbf{x})\}$ is convex if $\mathcal{F}_1(\mathbf{x}), \cdots, \mathcal{F}_m(\mathbf{x})$ are convex functions.

Although there are principles for verifying convexity, it may not be an easy problem even for polynomials of degree four. In the conference of complexity theory for numerical optimization 2012, N. Z. Shor asked the question [127],

> "Given a degree-4 polynomial in n variables, what is the complexity of determining whether this polynomial describes a convex function?"

A degree-4 polynomial, like for instance, $6x^4 - xy - yz + 8y^2x$, does not seem to be very complex. However, Ahmadi et al. [3] showed the somehow surprising result, that it is an NP-hard problem which cannot be solved in polynomial time.

Conjugate Function. The conjugate function is defined as the pointwise supremum over the difference of a linear function with a given function. Let $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$ be a given function. Then the conjugate function of $\mathcal{F}$ denoted as $\mathcal{F}^* : \mathbb{R}^n \to \mathbb{R}$, is defined as,

$$\mathcal{F}^*(\mathbf{y}) = \sup_{\mathbf{x} \in \mathrm{dom}(\mathcal{F})} \left\{\langle \mathbf{x}, \mathbf{y}\rangle - \mathcal{F}(\mathbf{x})\right\}. \tag{2.45}$$

Given an arbitrary function $\mathcal{F}$, the conjugate function $\mathcal{F}^*$ is closed and convex because it is the pointwise supremum over affine functions. Figure 2.15 illustrates a 1D conjugate function.

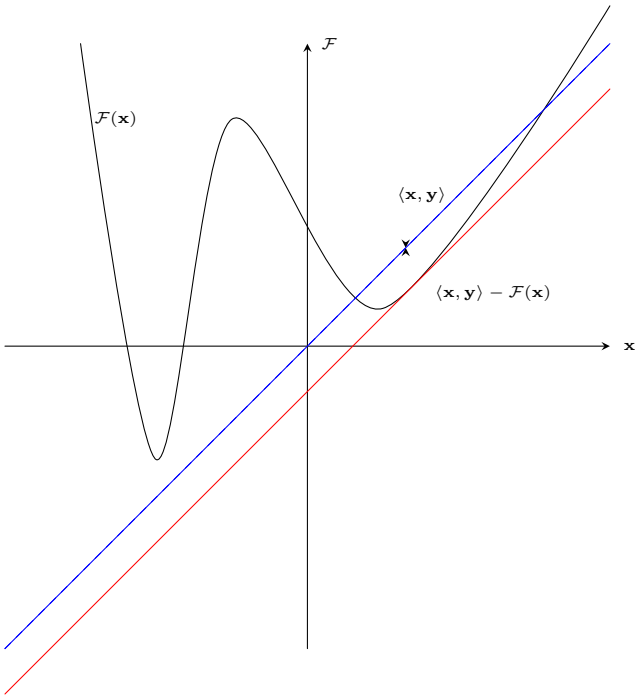


Figure 2.15: An illustration of a conjugate function.

The Conjugate of the Hinge Loss. Let us consider the hinge loss $\mathcal{F}(\mathbf{z}) = \|\max\{0, 1 - \mathbf{z}\}\|_1$ as an example. The hinge loss is a loss function used for maximum-margin classifiers in Support Vector Machines (SVMs). We simplify the definition of the hinge loss to

$$\mathcal{F}(\mathbf{z}) = \|\max\{\mathbf{0}, \mathbf{z}\}\|_1, \quad \text{with} \quad \mathbf{z} \in \mathbb{R}^n.$$

The corresponding convex conjugate is,

$$\mathcal{F}^*(\mathbf{y}) = \begin{cases} 0 & \mathbf{y} \in A \\ +\infty & \mathbf{y} \notin A \end{cases}$$

where $A = \left\{\mathbf{y} \in \mathbb{R}^N : \forall y_i \in [0, 1]\right\}$ and y_i is the i^{th} component of $\mathbf{y}$. Further we let $\mathbf{z} = 1 - K\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^d$ and $K \in \mathbb{R}^{n \times d}$. Because of $\mathcal{F}(\mathbf{z}) = \max_{\mathbf{y}} \{\langle \mathbf{z}, \mathbf{y} \rangle - \mathcal{F}^*(\mathbf{y})\}$, by substituting $\mathbf{z}$ with $1 - K\mathbf{x}$, it attains,

$$\mathcal{F}(1 - K\mathbf{x}) = \max_{\mathbf{y}} \{\langle 1 - K\mathbf{x}, \mathbf{y} \rangle - \mathcal{F}^*(\mathbf{y})\}.$$

The Conjugate of the ℓ_1-norm Consider the absolute loss function $\mathcal{L}(\mathbf{z}) = \|\mathbf{z}\|_1, \mathbf{z} \in \mathbb{R}^n$, as another example. Its convex conjugate is,

$$\mathcal{L}^*(\mathbf{y}) = \begin{cases} 0 & \mathbf{y} \in A \\ +\infty & \mathbf{y} \notin A \end{cases}$$

where $A = \{\mathbf{y} \in \mathbb{R}^n : \forall y_i \in [-1, 1]\}$, y_i is the i^{th} component of $\mathbf{y}$. Thus, one obtains

$$\mathcal{L}(\mathbf{z}) = \max_{\mathbf{y}} \{\langle \mathbf{z}, \mathbf{y} \rangle - \mathcal{L}^*(\mathbf{y})\}.$$

If we let $\mathbf{z} = \mathbf{K}\mathbf{x} - \mathbf{l}, \mathbf{K} \in \mathbb{R}^{n \times d}, \mathbf{x} \in \mathbb{R}^d, \mathbf{l} \in \mathbb{R}^n$, then we have

$$\mathcal{L}(\mathbf{K}\mathbf{x} - \mathbf{l}) = \max_{\mathbf{y}} \{\langle \mathbf{K}\mathbf{x} - \mathbf{l}, \mathbf{y} \rangle - \mathcal{L}^*(\mathbf{y})\}.$$

Therefore, $\mathcal{F}(\mathbf{x}) = \|\mathbf{K}\mathbf{x} - \mathbf{l}\|_1$ can be rewritten as,

$$\mathcal{F}(\mathbf{x}) = \max_{\mathbf{y}} \{\langle \mathbf{K}\mathbf{x} - \mathbf{l}, \mathbf{y} \rangle - \mathcal{L}^*(\mathbf{y})\} = \max_{\mathbf{y}} \{\langle \mathbf{K}\mathbf{x}, \mathbf{y} \rangle - \langle \mathbf{l}, \mathbf{y} \rangle - \mathcal{L}^*(\mathbf{y})\}.$$

Hence in the saddle-point model, we can let $\mathcal{F}^*(\mathbf{x}) = \langle \mathbf{l}, \mathbf{y} \rangle + \mathcal{L}^*(\mathbf{y})$, which is frequently used in Chapter 3.

Proximal Operator. Given a proper convex function $\mathcal{G} : \mathbb{R}^n \to \mathbb{R}$, the proximal operator of $\mathcal{G}$ with parameter $\tau > 0$ is defined by

$$\mathrm{prox}_{\tau\mathcal{G}}(\mathbf{v}) = (I + \tau\nabla\mathcal{G})^{-1}(\mathbf{v}) = \arg\min_{\mathbf{x}} \left\{ \tau\mathcal{G}(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|_2^2 \right\}. \tag{2.46}$$

Since the Euclidean norm is strongly convex, $\mathrm{prox}_{\tau\mathcal{G}}(\mathbf{v})$ is unique. The proximal operator is frequently used in optimization algorithms.

The Proximal Mapping of the Infinity Norm. Let $\mathbf{x} \in \mathbb{R}^n$ be a vector, the infinity norm of $\mathbf{x}$ is,

$$\mathcal{G}(\mathbf{x}) = \max_i |x_i|, \quad i = 1, \cdots, n.$$

The Proximal Mapping of the infinity norm $\mathcal{G}$ at $\tilde{\mathbf{x}}$ is given as,

$$\mathbf{x} = \mathrm{Prox}_{\lambda\mathcal{G}}(\tilde{\mathbf{x}}) = \arg\min_{\mathbf{z}} \left\{ \frac{1}{2}\|\mathbf{z} - \widetilde{\mathbf{x}}\|_2^2 + \lambda\mathcal{G}(\mathbf{z}) \right\},$$

where λ is a scalar. This means that $\mathbf{x}$ should be close to $\tilde{\mathbf{x}}$ but maintain an infinity norm $\mathcal{G}(\mathbf{x})$ as small as possible. To calculate $\mathrm{Prox}_{\lambda\mathcal{G}}(\tilde{\mathbf{x}})$, we propose the following propositions.

Proposition i Suppose that $\mathcal{G}(\mathbf{x}) = \omega$, then there is

$$x_i = (\mathrm{Prox}_{\lambda\mathcal{G}}(\tilde{\mathbf{x}}))_i = \begin{cases} \tilde{x}_i & \tilde{x}_i \leq \omega \\ \omega & \tilde{x}_i > \omega. \end{cases}$$

Proposition ii If we randomly change the signs of $\tilde{x}_i$, $\mathcal{G}(\mathbf{x})$ does not change however the signs of $\mathbf{x}$ should be changed accordingly. Also if the order of components of $\tilde{\mathbf{x}}$ changes (e.g., exchanging $\tilde{x}_i$ and $\tilde{x}_j$), the resultant $\mathbf{x}$ should change in the same order.

So we need to know $\mathcal{G}(\mathbf{x})$ to calculate the proximal mapping of the infinity norm $\mathcal{G}$ at $\tilde{\mathbf{x}}$. To this end, first we consider the constrained optimization problem,

$$\begin{aligned} &\mathrm{Prox}_{\lambda\mathcal{G}}(\tilde{\mathbf{x}}) = \arg\min_{\mathbf{z}} \left\{ \frac{1}{2}\|\mathbf{z} - \widetilde{\mathbf{x}}\|_2^2 + \lambda\mathcal{G}(\mathbf{z}) \right\} \\ &\text{subject to } \mathcal{G}(\mathrm{Prox}_{\lambda\mathcal{G}}(\tilde{\mathbf{x}})) = m, \text{ where } m \in \{|\tilde{x}_i|, i \in \{1, \cdots, n\}\}. \end{aligned} \tag{2.47}$$

Next, we sort $|\widetilde{\mathbf{x}}|$ in descending order. The new vector is denoted by $\tilde{\mathbf{c}}$. According to the Proposition ii, Equation (2.47) is equal to solve the following problem,

$$\begin{aligned} &\mathbf{c} = \mathrm{Prox}_{\lambda\mathcal{G}}(\tilde{\mathbf{x}}) = \arg\min_{\mathbf{z}} \left\{ \frac{1}{2}\|\mathbf{z} - \widetilde{\mathbf{c}}\|_2^2 + \lambda\mathcal{G}(\mathbf{z}) \right\} \\ &\text{subject to } \mathcal{G}(\mathbf{c}) = m, \text{ where } m \in \{\tilde{c}_i, i \in \{1, \cdots, n\}\}. \end{aligned} \tag{2.48}$$

Now we compare two circumstances $\mathcal{G}(\mathbf{c}) = \tilde{c}_k,\ k \in \{1, \cdots, n-1\}$ and $\mathcal{G}(\mathbf{c}) = \tilde{c}_{k+1}$. According to the Proposition i, the following equations hold.

	$\frac{1}{2}\|\mathbf{z} - \widetilde{\mathbf{c}}\|_2^2 + \lambda\mathcal{G}(\mathbf{z})$
$\mathcal{G}(\mathbf{c}) = \tilde{c}_k$	$\mathcal{E}(k) = \frac{1}{2}\sum_{i=1}^{k} (\tilde{c}_i - \tilde{c}_k)^2 + \lambda\tilde{c}_k$
$\mathcal{G}(\mathbf{c}) = \tilde{c}_{k+1}$	$\mathcal{E}(k+1) = \frac{1}{2}\sum_{i=1}^{k+1} (\tilde{c}_i - \tilde{c}_{k+1})^2 + \lambda\tilde{c}_{k+1}$

By algebra, it is easy to show that if $\mathcal{E}(k) \leq \mathcal{E}(k+1)$, then there is $\mathcal{E}(k+1) \leq \mathcal{E}(k+2)$. This leads to the following lemma.

Lemma The solution of m in Equation (2.48) is

$$m = \max_k \left\{ \widetilde{c}_k | 2\lambda < 2\sum_{i=1}^{i<k} \widetilde{c}_i + (k-2, 0)\left(-\widetilde{c}_k - \widetilde{c}_{k+1}\right) - 2\widetilde{c}_{k+1},\ k \geq 1 \right\}. \tag{2.49}$$

This shows that m is the largest $\widetilde{c}_k$ which satisfies the inequality in Equation (2.49). In the second step, we calculate the exact $\mathcal{G}(\mathbf{x})$. There are two cases. One is $\mathcal{G}(\mathbf{x})$ should

be larger and the other is $\mathcal{G}(\mathbf{x})$ should be smaller. Suppose that $m = \widetilde{c}_j$. We calculate $u = \lambda - \sum_{i=1}^{i<j} \widetilde{c}_i + (j-1)\widetilde{c}_j$ and define d,

$$d = \begin{cases} -u & j = 1 \\ -\frac{u}{k} & u > 0, j > 1 \\ -\frac{u}{l} & u \leq 0, j > 1 \end{cases}$$

where l is the cardinality of $I = \{i \,|\, |\tilde{x}_i| > m\}$. We define $\widehat{m} = m + d$, then

$$x_i = \begin{cases} \text{sign}(\tilde{x}_i)\widehat{m} & |\tilde{x}_i| > \widehat{m} \\ \tilde{x}_i & \text{otherwise} \end{cases}.$$

2.7.1 First-order Methods

Convex optimization falls into two categories, smooth optimization and non-smooth optimization. Optimizing a non-smooth convex function is much harder than the smooth one [91] because a random subgradient may not be a descent direction. Some researchers [11, 54, 147] choose to divide an objective function into two sub-functions. Among those algorithms, a substantial amount of literature [11, 146, 147] assumes that at least one of the sub-functions must be smooth. If an optimization algorithm only uses first-order derivatives of the objective function to determine the search direction of each iteration, then it is referred to as a first-order optimization algorithm. In the following, we introduce several first-order optimization algorithms that are commonly used in ML.

Duchi and Singer [54] proposed an algorithm for online and batch learning based on the forward-backward splitting method, which is called Forward-backward Splitting Method (FBS). This method aims to solve a convex optimization problem that involves a sum of two convex bounded functions:

$$\min_{\mathbf{x}} \mathcal{E}(\mathbf{x}), \quad \text{with} \quad \mathcal{E}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathcal{G}(\mathbf{x}),$$

where $\mathcal{E}$ denotes the objective function. By convention, $\mathcal{F}(\mathbf{x})$ usually denotes a loss function and $\mathcal{G}(\mathbf{x})$ denotes a regularization term. FBS consists of two steps: an explicit step for $\nabla\mathcal{F}$ and an implicit step by applying a resolvent mapping of $\nabla\mathcal{G}$. Duchi and Singer [54] proved that the convergence rate of FBS in a batch setting is $O(1/\sqrt{n})$ in terms of the error of energy under a fairly general assumption that the subgradients of $\mathcal{F}$ and $\mathcal{G}$ are bounded. And the sizes of time steps are decided by this assumption.

Beck and Teboulle [11] proposed a method which is an accelerated gradient method with convergence rate $O(1/n^2)$ in terms of the error of the energy. The so-called Fast Iterative Shrinkage-thresholding Algorithm (FISTA) still aims to solve a convex problem with a sum of two convex functions. However, it requires that $\mathcal{F}$ is a $C^{1,1}$ smooth

convex function, that is, continuously differentiable with Lipschitz continuous gradient. For this reason, before using FISTA to solve a problem with a non-smooth loss function $\mathcal{F}$, smoothing techniques need to be applied. Hence FISTA may not achieve an optimal solution of the original problem. Moreover, smoothing techniques always results in a piecewise smooth function. This recasts the loss function as a sum of piecewise smooth functions. Thus, FISTA becomes more time consuming to calculate the gradient of a smoothed loss function.

Neumaier [120] proposed a fast subgradient algorithm with optimal complexity named Optimal Subgradient Algorithm (OSGA). OSGA is proposed to solve a convex function without knowing global information, e.g., the Lipschitz constant. When the objective function is not strongly convex, the convergence rate for non-smooth convex functions is $O(1/\sqrt{n})$ w.r.t. the error of the energy. If the strong convexity is fulfilled, then the convergence rate can be $O(1/n)$. For smooth convex functions, OSGA converges with rate $O(1/n^2)$. The convergence rate is $O(1/e^n)$ if $\mathcal{E}$ is a smooth, strongly convex function. Ahookhosh [4] applied OSGA to large-scale inverse problems including image restoration, ℓ_2^2-ℓ_2^2 problem and sparse optimization ℓ_2^2-ℓ_1. Ahookhosh [4] reported experimental results for the aforementioned linear inverse problems and showed that OSGA is more competitive than other state-of-the-art optimization algorithms, like e.g. Nesterov's algorithms.

Another important category of first-order optimization algorithms are primal-dual algorithms [31, 58, 172, 179]. Chambolle and Pock [31] proposed a Primal-Dual Algorithm (PDCP) with a convergence rate of $O\left(1/n\right)$ for the primal-dual gap. And if convex models have more smoothness structures, the variations of PDCP have faster convergence rates $\left(O\left(1/n^2\right),\ O\left(1/e^n\right)\right)$. This optimization algorithm is applied to several imaging problems, for instance, image denoising or image deconvolution. Yang et al. [172] proposed a primal-dual algorithm with the rate $O\left(1/n\right)$ for the primal-dual gap and applied the proposed algorithm to ML tasks, such as grouped feature selection and multi-task learning. For a primal-dual algorithm, it is still the same in essence if the ways to iterate the primal and the dual variable are exchanged. Therefore, the algorithm of Chambolle and Pock [31] and the algorithm of Yang et al. [172] are identical. However, PDCP is more general in the following aspects. The time step size of PDCP is $\sqrt{2}$ times larger and enables time steps for both the primal and the dual variables. Thus, we can use different ratios between primal and dual variables which leads to a different performance.

Forward-backward Splitting Method. The FBS was independently introduced in [102] and [129] in 1979. The principle of this method is to divide an overall convex function $\mathcal{E}$ into two suitable subfunctions, for instance, $\mathcal{F}$ and $\mathcal{G}$ such that $\mathcal{E} = \mathcal{F} + \mathcal{G}$. Its iteration consists of two steps, a forward gradient step on $\mathcal{F}$, that is a convex function having a Lipschitz continuous gradient, and a backward step on $\mathcal{G}$, that is a lower semicontinuous convex function [10, 39]. The forward-backward splitting method is widely used in various areas of research.

Duchi and Singer [54] proposed an algorithm named FBS using forward-backward splitting. They proved the convergence rates for an arbitrary convex function under a fairly general assumption that the gradients of $\mathcal{F}$ and $\mathcal{G}$ are bounded. FBS is proposed to solve a convex minimization problem such as the sum of two convex functions.

$$\begin{aligned} \text{minimize} \quad & F(\mathbf{x}) + \mathcal{G}(\mathbf{x}), \\ \text{subject to} \quad & \mathbf{x} \in C \end{aligned} \tag{2.50}$$

where both $\mathcal{F}(\mathbf{x})$ and $\mathcal{G}(\mathbf{x})$ are convex functions that are bounded from below, and C is a convex set. This fits FBS for the following prevalent ML problems where $\mathcal{F}(\mathbf{x})$ is an empirical loss and $\mathcal{G}(\mathbf{x})$ is a regularization term, e.g., ℓ_p norm. FBS in a batch setting can be summarized as below,

- Initialization: $\mathbf{x}^0$
- Iterations $n \geq 1$: Update $\widehat{\mathbf{x}}^n$ and $\mathbf{x}^n$ as follows

$$\begin{cases} \widehat{\mathbf{x}}^n &= \mathbf{x}^{n-1} - \tau^n \nabla \mathcal{F}(\mathbf{x}^{n-1}) \\ \mathbf{x}^n &= (\mathbf{I} + \tau^{n+1} \nabla \mathcal{G})^{-1} \widehat{\mathbf{x}}^n \end{cases} \tag{2.51}$$

where $\tau^n = \frac{c}{\sqrt{n}}$ and c is an arbitrary constant.

In Equation (2.51), $\mathbf{x}^n$ is the proximal mapping of the convex function, i.e. $\tau^{n+1}\mathcal{G}$ at position $\widehat{\mathbf{x}}^n$,

$$\mathbf{x}^n = \arg\min_{\mathbf{x} \in C} \left\{ \tau^{n+1} \mathcal{G}(\mathbf{x}) + \frac{1}{2} \|\mathbf{x} - \widehat{\mathbf{x}}^n\|_2^2 \right\}. \tag{2.52}$$

Therefore, in the first step of FBS, it conducts a subgradient step with respect to the function $\mathcal{F}$. And in the second step, based on the result of the first step, it calculates the proximal mapping of the convex function $\tau^{n+1}\mathcal{G}$ at $\widehat{\mathbf{x}}^n$.

The convergence rate of FBS was proven to be $O(1/\sqrt{n})$ [54]. For a problem with a non-smooth loss function $\mathcal{F}$, it is tricky to set the value of the constant c. In some cases, c not only depends on the upper bound of the distance from the initial point to the optimal set but also depends on the Lipschitz constants of the gradients of $\mathcal{F}$ and $\mathcal{G}$. However, for a non-smooth function, estimating the Lipschitz constant usually underestimates the time step size. A large step size leads to a fast convergence in practice, but it is also prone to diverge, due to occurring fluctuation.

Fast Iterative Shrinkage-thresholding Algorithm. The Iterative Shrinkage-thresholding Algorithm (ISTA) is considered as an adaption of the gradient descent algorithm and is frequently used in linear inverse problems of digital image processing, for instance, total variation based image restoration. Beck and Teboulle [11] proposed the FISTA which owns a fast global rate of convergence of $O(1/n^2)$ and keeps

the simplicity of ISTA in computation. FISTA aims to deal with problems formulated as

$$\text{minimize} \quad \{\mathcal{F}(\mathbf{x}) + \mathcal{G}(\mathbf{x})\}$$

where $\mathcal{F}, \mathcal{G}$ are convex functions and $\mathcal{F}$ must have a Lipschitz continuous gradient. Let L be the Lipschitz constant of $\nabla\mathcal{F}$. Then the algorithm of FISTA is summarized as,

- Initialization: L, $\mathbf{x}^0$, $\overline{\mathbf{x}}^0 = \mathbf{x}^0$, $\tau^0 = 1$
- Iterations $n \geq 1$: Update $\mathbf{x}^n, \overline{\mathbf{x}}^n$ as follows,

$$\begin{cases} \mathbf{x}^n &= (\mathbf{I} + \frac{1}{L}\nabla\mathcal{G})^{-1}(\overline{\mathbf{x}}^{n-1} - \frac{1}{L}\nabla\mathcal{F}(\overline{\mathbf{x}}^{n-1})) \\ \tau^n &= \frac{1+\sqrt{1+4(\tau^{n-1})^2}}{2} \\ \overline{\mathbf{x}}^n &= \mathbf{x}^n + (\frac{\tau^{n-1}-1}{\tau^n})(\mathbf{x}^n - \mathbf{x}^{n-1}) \end{cases} \tag{2.53}$$

In the case of a non-computable L, FISTA uses a backtracking stepsize rule which introduces a new energy,

$$\mathcal{Q}_L(\mathbf{x}, \mathbf{y}) = \mathcal{F}(\mathbf{y}) + \langle \mathbf{x} - \mathbf{y}, \nabla\mathcal{F}(\mathbf{y})\rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2 + \mathcal{G}(\mathbf{x}), \tag{2.54}$$

that admits an unique minimizer

$$\mathbf{p}_L(\mathbf{y}) = \arg\min_{\mathbf{x}} \{\mathcal{Q}_L(\mathbf{x}, \mathbf{y})\}. \tag{2.55}$$

By letting the gradient w.r.t. $\mathbf{x}$ of the right hand side of Equation (2.55) be zero, we obtain

$$\nabla\mathcal{F}(\mathbf{y}) + L(\mathbf{x} - \mathbf{y}) + \nabla\mathcal{G}(\mathbf{x}) = 0,$$

which further leads to

$$\mathbf{x} = (\mathbf{I} + \frac{1}{L}\nabla\mathcal{G})^{-1}(\mathbf{y} - \frac{1}{L}\nabla\mathcal{F}(\mathbf{y})).$$

If we set $\mathbf{y} = \overline{\mathbf{x}}^{n-1}$, we obtain

$$\mathbf{x}^n = \mathbf{p}_L(\overline{\mathbf{x}}^{n-1}), \tag{2.56}$$

which is the basic step of FISTA of Equation (2.53). FISTA with backtracking is summarized in the following algorithm,

• Initialization: $L^0 > 0, \eta > 1$, x^0, $\overline{\mathbf{x}}^0 = \mathbf{x}^0$, $\tau^0 = 1$
• Iterations $n \geq 1$: Update $\mathbf{x}^n, \overline{\mathbf{x}}^n$ as follows,
Find the smallest nonnegative integers(power) i^n such that with $\overline{L} = \eta^{i^n} L^{n-1}$

$$\mathcal{E}(\mathbf{p}_{\overline{L}}(\overline{\mathbf{x}}^{n-1})) \leq \mathcal{Q}_{\overline{L}}(\mathbf{p}_{\overline{L}}(\overline{\mathbf{x}}^{n-1}), \overline{\mathbf{x}}^{n-1})$$

and define $L^n = \eta^{i^n} L^{n-1}$,

$$\begin{cases} \mathbf{x}^n &= (\mathbf{I} + \frac{1}{L^n}\nabla\mathcal{G})^{-1}(\overline{\mathbf{x}}^{n-1} - \frac{1}{L^n}\nabla\mathcal{F}(\overline{\mathbf{x}}^{n-1})) \\ \tau^n &= \frac{1+\sqrt{1+4(\tau^{n-1})^2}}{2} \\ \overline{\mathbf{x}}^n &= \mathbf{x}^n + (\frac{\tau^{n-1}-1}{\tau^n})(\mathbf{x}^n - \mathbf{x}^{n-1}) \end{cases} \tag{2.57}$$

Note that, in Equation (2.57), $\mathbf{x}^n$ can be simplified to $\mathbf{x}^n = \mathbf{p}_{L^n}(\overline{\mathbf{x}}^{n-1})$, which shows that $\mathbf{x}^n$ is a minimizer of $\left\{\mathcal{Q}_{L^n}(\mathbf{x}, \overline{\mathbf{x}})^{n-1}\right\}$ (cf. Equation (2.55)) and thus an upper bound of $\mathcal{E}$.

Optimal Subgradient Algorithm. Neumaier [120] proposed an Optimal Subgradient Algorithm, i.e. a first-order algorithm with optimal complexity [119], $O(1/\sqrt{n})$ for Lipschitz continuous nonsmooth optimization problems and $O(1/n^2)$ for smooth optimization problems with Lipschitz continuous gradient. OSGA is a simple algorithm in the sense that only the values of the target functions and corresponding subgradients are required in the case of non-strongly convex functions. In the case of strong convex optimization, a strong convexity parameter is needed. The algorithm is to solve the following problem,

$$\begin{aligned} \text{minimize} \quad & \mathcal{F}(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in C \end{aligned}$$

where C is a convex set and $\mathcal{F}(\mathbf{x}) : C \to \mathbb{R}^n$ is a proper and convex function.

The principle of OSGA [120] is to monotonically decrease the upper bound of the error $\mathcal{F}^n - \widehat{\mathcal{F}}$ where $\mathcal{F}^n$ is the function value at the n$^{\text{th}}$ iteration and $\widehat{\mathcal{F}}$ is the optimal value $\widehat{\mathcal{F}} = \min_{\mathbf{x}\in C} \mathcal{F}(\mathbf{x})$. OSGA defines proper linear relaxations

$$\mathcal{F}(\mathbf{z}) \geq \gamma + \langle \mathbf{y}, \mathbf{z} \rangle, \ \forall \mathbf{z} \in C, \tag{2.58}$$

where $\mathbf{y}$ is in the dual space of C. One example which makes Equation (2.58) hold is,

$$\gamma = \mathcal{F}(\mathbf{x}) - \langle \nabla\mathcal{F}(\mathbf{x}), \mathbf{x} \rangle \text{ and } \mathbf{y} = \nabla\mathcal{F}(\mathbf{x}).$$

In addition, OSGA defines a continuously differentiable strong convex function $\mathcal{Q} : C \to \mathbb{R}$

with a unit strong convexity parameter,

$$\mathcal{Q}(\mathbf{z}) \geq \mathcal{Q}(\mathbf{x}) + \langle \nabla \mathcal{Q}(\mathbf{x}), \mathbf{z} - \mathbf{x} \rangle + \frac{1}{2}\|\mathbf{z} - \mathbf{x}\|_2^2, \ \forall \ \mathbf{x}, \mathbf{z} \in C. \tag{2.59}$$

For instance, $\mathcal{Q}$ can be given as,

$$\mathcal{Q}(\mathbf{z}) = q + \frac{1}{2}\|\mathbf{z} - \mathbf{x}\|_2^2, \tag{2.60}$$

where $q = \left\{ q > 0 \middle| \inf_{\mathbf{x} \in C} \mathcal{Q}(\mathbf{x}) \right\}$ and $\mathbf{x} \in C$. Based on the above definitions, OSGA defines

$$\mathcal{E}_{\gamma,\mathbf{y}}(\mathbf{x}) = -\frac{\gamma + \langle \mathbf{x}, \mathbf{z} \rangle}{\mathcal{Q}(\mathbf{x})}, \ \text{with } \mathbf{x} \in C. \tag{2.61}$$

OSGA needs to simply calculate the following variables,

$$\mathcal{E}(\gamma, \mathbf{y}) = \sup_{\mathbf{x}} \mathcal{E}_{\gamma,\mathbf{y}}(\mathbf{x}) \tag{2.62}$$

and

$$\mathcal{U}(\gamma, \mathbf{y}) = \arg\sup \mathcal{E}_{\gamma,\mathbf{y}}(\mathbf{x}). \tag{2.63}$$

Neumaier [120] showed that for arbitrary $\gamma \in \mathbb{R}$ and $\mathbf{y}$, the following holds

$$\gamma + \langle \mathbf{y}, \mathbf{z} \rangle \geq -\mathcal{E}(\gamma, \mathbf{y})\mathcal{Q}(\mathbf{z}), \ \forall \ \mathbf{z} \in C. \tag{2.64}$$

If we let $\gamma^n = \gamma - \mathcal{F}(\mathbf{x}^n), \mathcal{E}(\gamma^n, \mathbf{y}) \leq \eta$, then there is

$$0 \leq \mathcal{F}(\mathbf{x}^n) - \widehat{\mathcal{F}} \leq \eta \mathcal{Q}(\widehat{\mathbf{x}}). \tag{2.65}$$

Based on the above equations, we can summarize OSGA for solving a non-strongly convex function as follows:

• Initialization: $\alpha_{\max} \in (0,1),\ 0 < k' \leq k,\ \mathbf{x}^0 \in C$
$\mathbf{y} = \nabla\mathcal{F}(\mathbf{x}^0);\ \gamma = \mathcal{F}(\mathbf{x}^0) - \langle \mathbf{y}, \mathbf{x}^0 \rangle$
$\gamma^0 = \gamma - \mathcal{F}(\mathbf{x}^0);\ \mathbf{u} = \mathcal{U}(\gamma^0, \mathbf{y});\ \eta = \mathcal{E}(\gamma^0, \mathbf{y})$
$\alpha = \alpha_{\max}$
• Iterations $n \geq 1$: Update x^n as follows,

$$
\begin{cases}
\mathbf{x} = \mathbf{x}^{n-1} + \alpha(\mathbf{u} - \mathbf{x}^{n-1}) \\
\overline{\mathbf{y}} = \mathbf{y} + \alpha(\nabla\mathcal{F}(\mathbf{x}) - \mathbf{y}) \\
\overline{\gamma} = \gamma + \alpha(\mathcal{F}(\mathbf{x}) - \langle \nabla F(\mathbf{x}), \mathbf{x} \rangle - \gamma) \\
\mathbf{x}'^{n-1} = \underset{\mathbf{z} \in \{\mathbf{x}, \mathbf{x}^{n-1}\}}{\arg\min}\ \mathcal{F}(\mathbf{z}) \\
\gamma'^{n-1} = \overline{\gamma} - \mathcal{F}(\mathbf{x}'^{n-1}) \\
\mathbf{u}' = \mathcal{U}(\gamma'^{n-1}, \overline{\mathbf{y}}) \\
\mathbf{x}' = \mathbf{x}^{n-1} + \alpha(\mathbf{u}' - \mathbf{x}^{n-1}) \\
\text{choose } \overline{\mathbf{x}}^{n-1} \text{ with } \mathcal{F}(\overline{\mathbf{x}}^{n-1}) \leq \min(\mathbf{F}(\mathbf{x}'^{n-1}), \mathbf{F}(\mathbf{x}')) \\
\overline{\gamma}^n = \overline{\gamma} - \mathcal{F}(\overline{\mathbf{x}}^{n-1}) \\
\overline{\mathbf{u}} = \mathcal{U}(\overline{\gamma}^n, \overline{\mathbf{y}}) \\
\overline{\eta} = \mathcal{E}(\overline{\gamma}^n, \overline{\mathbf{y}}) \\
\mathbf{x}^n = \overline{\mathbf{x}}^{n-1} \\
\text{update the parameters by Equation (2.67).}
\end{cases}
\tag{2.66}
$$

The update scheme of the global tuning parameters is:

• Initialization: $q \in (0, e^{-k}], \alpha_{\max} \in (0,1), 0 < k' \leq k$
• input: $\alpha,\ \eta,\ \overline{\gamma},\ \overline{\mathbf{y}},\ \overline{\eta},\ \overline{\mathbf{u}}$
• output: $\alpha,\ \eta,\ \gamma,\ \mathbf{y},\ \mathbf{u}$

$$
\begin{cases}
R = \frac{\eta - \overline{\eta}}{q\alpha\eta} \\
\text{if } R < 1 \\
\quad \overline{\alpha} = \alpha e^{-k} \\
\text{else} \\
\quad \overline{\alpha} = \min(\alpha e^{k'(R-1)}, \alpha_{\max}) \\
\text{end} \\
\alpha = \overline{\alpha} \\
\text{if } \overline{\eta} < \eta, \\
\quad \gamma = \overline{\gamma};\ \mathbf{y} = \overline{\mathbf{y}};\ \eta = \overline{\eta};\ \mathbf{u} = \overline{\mathbf{u}} \\
\text{end.}
\end{cases}
\tag{2.67}
$$

Primal-dual Algorithm. Chambolle and Pock [31] proposed a primal-dual algorithm (hereinafter referred as PDCP) and applied it to several imaging problems.

Given a continuous linear operator $K : X \to Y$ with an induced norm

$$\|K\| = \max \{\|\mathbf{K}\mathbf{x}\|_2 : \mathbf{x} \in X,\ \|\mathbf{x}\|_2 \leq 1\},$$

where X, Y are two finite dimensional real vector spaces. Further given $\mathcal{G}$ and $\mathcal{F}^*$, both are proper convex, lower-semicontinuous functions PDCP aims to solve the generic saddle-point problem

$$\min_{\mathbf{x}\in X} \max_{\mathbf{y}\in Y} \{\langle \mathbf{K}\mathbf{x}, \mathbf{y}\rangle + \mathcal{G}(\mathbf{x}) - \mathcal{F}^*(\mathbf{y})\}. \tag{2.68}$$

By convention $\mathbf{x}$ is called a primal variable and $\mathbf{y}$ is called a dual variable. The corresponding primal form of (2.68) is

$$\underset{\mathbf{x}\in X}{\text{minimize}} \quad F(\mathbf{K}\mathbf{x}) + G(\mathbf{x}). \tag{2.69}$$

To introduce the dual variables $\mathbf{y}$, one could use the Lagrange formalism, e.g., applicable when the function represents hard constraints. Or one could calculate the convex conjugate, $\mathcal{F}(\mathbf{K}\mathbf{x}) = \max_{\mathbf{y}} \langle \mathbf{K}\mathbf{x}, \mathbf{y}\rangle - F^*(\mathbf{y})$.

PDCP iteratively maximizes w.r.t. the dual variables and minimizes w.r.t. the primal variables. The condition for convergence of PDCP is $\tau\sigma\|\mathbf{K}\|^2 \leq 1$. We summarize PDCP as follows.

- Initialization: $\tau\sigma\|\mathbf{K}\|^2 \leq 1$, $\theta \in [0, 1]$, $(\mathbf{x}^0,\ \mathbf{y}^0) \in X \times Y$, $\overline{\mathbf{x}}^0 = \mathbf{x}^0,\ \lambda \in \mathbb{R}$
- Iterations $n \geq 1$: Update $\mathbf{x}^n,\ \mathbf{y}^n,\ \overline{\mathbf{x}}^n$ as follows,

$$\begin{cases} \mathbf{y}^n &= (\mathbf{I} + \sigma\nabla\mathcal{F}^*)^{-1}(\mathbf{y}^{n-1} + \sigma\mathbf{K}\overline{\mathbf{x}}^{n-1}) \\ \mathbf{x}^n &= (\mathbf{I} + \tau\nabla\mathcal{G})^{-1}(\mathbf{x}^{n-1} - \tau\mathbf{K}^\top\mathbf{y}^n) \\ \overline{\mathbf{x}}^n &= \mathbf{x}^n + \theta(\mathbf{x}^n - \mathbf{x}^{n-1}) \end{cases} \tag{2.70}$$

The PDCP calculates the proximal operator of $\sigma\mathcal{F}^*$ followed by the proximal operator of $\tau\mathcal{G}$. Then PDCP makes its scheme semi-implicit by letting $\overline{\mathbf{x}}^n = \mathbf{x}^n + \theta(\mathbf{x}^n - \mathbf{x}^{n-1})$. This operation makes one more step in the direction of $\mathbf{x}^n - \mathbf{x}^{n-1}$ scaled by θ.

Heuristics for the Primal Dual Algorithm. Although the convergence criteria of the PDCP is $\tau\sigma\|\mathbf{K}\|^2 \leq 1$, it is still an unsolved problem how to choose τ and σ to achieve the best performance. Let $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ be the saddle point. Chambolle and Pock [31] showed

that the following holds

$$\begin{aligned}[\langle \mathbf{K}\mathbf{x}_N, \widehat{\mathbf{y}}\rangle - \mathcal{F}^*(\widehat{\mathbf{y}}) + \mathcal{G}(\mathbf{x}_N)] - [\langle \mathbf{K}\widehat{\mathbf{x}}, \mathbf{y}_N\rangle - \mathcal{F}^*(\mathbf{y}_N) + \mathcal{G}(\widehat{\mathbf{x}})] \\ \leq \frac{1}{N}\left(\frac{\|\widehat{\mathbf{y}} - \mathbf{y}^0\|^2}{2\sigma} + \frac{\|\widehat{\mathbf{x}} - \mathbf{x}^0\|^2}{2\tau}\right),\end{aligned} \tag{2.71}$$

where $\mathbf{x}_N = \frac{1}{N}(\sum_{n=1}^{N} \mathbf{x}^n)$, $\mathbf{y}_N = \frac{1}{N}(\sum_{n=1}^{N} \mathbf{y}^n)$. The LHS of Equation (2.71) is called the primal-dual gap and the RHS of Equation (2.71) is the obtained upper bound. Chambolle and Pock [31] observed that the primal dual gap is non-negative because

$$\begin{aligned}[\langle \mathbf{K}\mathbf{x}_N, \widehat{\mathbf{y}}\rangle - \mathcal{F}^*(\widehat{\mathbf{y}}) + \mathcal{G}(\mathbf{x}_N)] \geq [\langle \mathbf{K}\widehat{\mathbf{x}}, \widehat{\mathbf{y}}\rangle - \mathcal{F}^*(\widehat{\mathbf{y}}) + \mathcal{G}(\widehat{\mathbf{x}})] \geq \\ [\langle \mathbf{K}\widehat{\mathbf{x}}, \mathbf{y}_N\rangle - \mathcal{F}^*(\mathbf{y}_N) + \mathcal{G}(\widehat{\mathbf{x}})]\,.\end{aligned} \tag{2.72}$$

Thus if $(\mathbf{x}_N, \mathbf{y}_N)$ is a saddle point, the RHS of Equation (2.71) equals zero which attains the minimal value. To minimize the upper bound of the primal-dual gap, we substitute $\tau = \frac{1}{\|\mathbf{K}\|^2\sigma}$ into the RHS of Equation (2.71) and get

$$\frac{1}{N}\left(\frac{\|\widehat{\mathbf{y}} - \mathbf{y}^0\|^2}{2\sigma} + \frac{\|\widehat{\mathbf{x}} - \mathbf{x}^0\|^2\|\mathbf{K}\|^2\sigma}{2}\right), \tag{2.73}$$

which is a convex function of the variable σ. We take the derivative w.r.t. σ, and set it to zero, which leads to

$$\sigma = \frac{\|\widehat{\mathbf{y}} - \mathbf{y}^0\|}{\|\widehat{\mathbf{x}} - \mathbf{x}^0\|\|\mathbf{K}\|}.$$

Thus we can conclude that $\frac{1}{N}\left(\frac{\|\widehat{\mathbf{y}}-\mathbf{y}^0\|^2}{2\sigma} + \frac{\|\widehat{\mathbf{x}}-\mathbf{x}^0\|^2}{2\tau}\right)$ reaches its minimum when $\sqrt{\frac{\tau}{\sigma}} = \frac{\|\widehat{\mathbf{x}}-\mathbf{x}^0\|}{\|\widehat{\mathbf{y}}-\mathbf{y}^0\|}$. Inspired by this, we observe that the convergence condition [31] $\tau\sigma \leq \frac{1}{\|\mathbf{K}\|^2}$ can be relaxed in order to accelerate the algorithm. We refer to the resulting scheme as Online PDCP. Although we do not prove its convergence theoretically, we can show that Online PDCP converges empirically. We will show in Chapter 3 that Online PDCP can converge faster than PDCP. The difference between PDCP and online PDCP is that online PDCP starts with a larger step size ($\tau\sigma > \frac{1}{\|\mathbf{K}\|^2}$) and decreases it according to a certain rule. One important use of online PDCP is the case where it is hard to compute $\|\mathbf{K}\|$. Online PDCP leverages the following scheme:

$$\begin{cases} \widetilde{L}^{n+1} &= \frac{\left\langle \mathbf{K}(\mathbf{x}^n - \mathbf{x}^{n-1}), \mathbf{y}^{n+1} - \mathbf{y}^n \right\rangle}{\|\mathbf{x}^n - \mathbf{x}^{n-1}\|\|\mathbf{y}^{n+1} - \mathbf{y}^n\|} \\ L^{n+1} &= \max\left\{L^n, \widetilde{L}^{n+1}\right\} \\ \tau^{n+1} &= \frac{a}{L^{n+1}}, \sigma^{n+2} = \frac{1}{aL^{n+1}} \end{cases}. \tag{2.74}$$

Choosing a proper L is the main concern of online PDCP. Chambolle and Pock [31] proved the convergence for the case $L = \|\mathbf{K}\|$, which is an upper bound of L^n in Equation (2.74). As shown in Equation (2.74), we let $L^{n+1} = \max\left\{L^n, \widetilde{L}^{n+1}\right\}$. If $L^n < \widetilde{L}^{n+1}$, we increase L^{n+1} to $\widetilde{L}^{n+1}$. A small L^n leads to large step sizes τ and σ. Online PDCP may take a risky initial step because of a large step sizes. An inappropriate large step sizes may lead to divergence. As a consequence, the new L^{n+1} calculated in the next step is tend to be large such that for instance, a large $\widetilde{L}$ may be close to $\|\mathbf{K}\|$. When $\widetilde{L}$ equals $\|\mathbf{K}\|$, online PDCP becomes PDCP. To balance the two different situations, we choose to smoothly increase L until we attain a suitable one. Therefore, we increase L to an appropriate degree rather than choose the maximum between L^n and $\widetilde{L}^{n+1}$. There exits different ways for increasing L. In this thesis, we let $L^{n+1} = \frac{\left(L^n + \kappa \max\left\{L^n, \widetilde{L}^{n+1}\right\}\right)}{1+\kappa}, \kappa > 0$. We set $\kappa = 0.618$ for all experiments in this thesis.

Summary of the First-order Optimization Algorithms. In this section, we introduced four first-order optimization algorithms. FBS [54] and FISTA [11] aim to solve a convex problem which is a sum of two convex functions. Neumaier [120] proposes a fast subgradient algorithm with optimal complexity for minimizing a convex function. Chambolle and Pock [31] propose a primal-dual algorithm and applied it to several imaging problems. Suppose the objective function is $\mathcal{E}$ which can be divided into two sub-functions $\mathcal{E} = \mathcal{F} + \mathcal{G}$, we can summarize the four solvers as shown in Table 2.2. Each solver can achieve the convergence rate under the property of $\mathcal{F}, \mathcal{G}, \mathcal{E}$ given by each row in Table 2.2. Note that when extra conditions of the functions hold, this might lead to a faster convergence rate.

Solver	Convergence rate	$\mathcal{F}$	$\mathcal{G}$	$\mathcal{E}$
FBS	$O(1/\sqrt{n})$	convex	convex	-
FISTA	$O(1/n^2)$	$C^{1,1}$	convex	-
OSGA	$O(1/\sqrt{n})$	-	-	convex
PDCP	$O(1/n)$	convex	convex	-

Table 2.2: Comparison of different first-order optimization algorithms.

2.7.2 Second-order Methods

Limited-memory BFGS. Limited-memory BFGS (L-BFGS) is a quasi-Newton method that is proposed based on the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS). Before reviewing quasi-Newton methods, we briefly introduce Newton's method.

Define the function to be minimized as $\mathcal{E}(\mathbf{x}),\ \mathbf{x} \in \mathbb{R}^d$. The quadratic Taylor expansion of $\mathcal{E}(\mathbf{x})$ at $\mathbf{x} + \nabla\mathbf{x}$ is

$$\mathcal{E}(\mathbf{x} + \nabla\mathbf{x}) \approx \mathcal{E}(\mathbf{x}) + \nabla\mathcal{E}(\mathbf{x})^\top (\nabla\mathbf{x}) + \frac{1}{2} (\nabla\mathbf{x})^\top \mathbf{H}(\mathbf{x}) (\nabla\mathbf{x}) \tag{2.75}$$

where $\nabla\mathcal{E}(\mathbf{x})$ and $\mathbf{H}(\mathbf{x})$ is the gradient and Hessian of $\mathcal{E}$ at the point $\mathbf{x}$. Equation (2.75) holds when $\nabla\mathbf{x} \rightarrow \mathbf{0}$. Newton's method chooses $\nabla\mathbf{x}$ to minimize the quadratic approximation of $\mathcal{E}(\mathbf{x} + \nabla\mathbf{x})$ whose gradient with respect to $\nabla\mathbf{x}$ is

$$\frac{\partial\mathcal{E}(\mathbf{x} + \nabla\mathbf{x})}{\partial\nabla\mathbf{x}} = \nabla\mathcal{E}(\mathbf{x}) + \mathbf{H}(\mathbf{x}) (\nabla\mathbf{x}) .$$

By letting $\frac{\partial\mathcal{E}(\mathbf{x}+\nabla\mathbf{x})}{\partial\nabla\mathbf{x}} = 0$, it attains the direction

$$\nabla\mathbf{x} = -\mathbf{H}^{-1}\nabla\mathcal{E}(\mathbf{x}). \tag{2.76}$$

The rate of convergence of Newton's method is quadratic under certain conditions, but solving Equation (2.76) needs $\mathcal{O}\left(d^3\right)$ operations. It is increasingly common to have hundreds of millions of parameters in machine-learning based computer vision problems and this limits the applications of the Newton's method.

Quasi-Newton Methods were proposed to approximate the inverse of the Hessian matrix implicitly. Let the direction at the n^{th} iteration be,

$$\mathbf{d}^n = -\mathbf{P}^n\nabla\mathcal{E}(\mathbf{x}^n). \tag{2.77}$$

If $\mathbf{P}^n$ is a symmetric positive definite matrix, i.e. $(\mathbf{d^n})^\top\nabla\mathcal{E}(\mathbf{x}^n) < 0$, the first-order Taylor expansion shows that $\mathbf{d}^n$ is a descent direction. If $\mathbf{P}^n$ is the identity matrix, this leads to the steepest descent method. In BFGS, besides demanding $\mathbf{P}^n$ to be a symmetric positive definite matrix, it also demands the secant condition which requires $\mathbf{P}^n$ to share the properties of a Hessian matrix. The secant equation is as follows,

$$\mathbf{P}^n \left(\nabla\mathcal{E}(\mathbf{x}^n) - \nabla\mathcal{E}(\mathbf{x}^{n-1})\right) = \mathbf{x}^n - \mathbf{x}^{n-1}. \tag{2.78}$$

To uniquely calculate $\mathbf{P}^n$, BFGS demands the update close to the one from the last iteration,

$$\min_{\mathbf{P}^n} \|\mathbf{P}^n - \mathbf{P}^{n-1}\|^2. \tag{2.79}$$

Let

$$\mathbf{y}^n = \nabla\mathcal{E}(\mathbf{x}^{n+1}) - \nabla\mathcal{E}(\mathbf{x}^n),$$

and

$$\mathbf{s}^n = \mathbf{x}^{n+1} - \mathbf{x}^n.$$

The solution of the above minimization problem of BFGS is

$$\mathbf{P}^n = \left(\mathbf{I} - \boldsymbol{\rho}^{n-1}\mathbf{s}^{n-1}\left(\mathbf{y}^{n-1}\right)^\top\right)\mathbf{P}^{n-1}\left(\mathbf{I} - \boldsymbol{\rho}^{n-1}\mathbf{y}^{n-1}\left(\mathbf{s}^{n-1}\right)^\top\right) + \boldsymbol{\rho}^{n-1}\mathbf{s}^{n-1}\left(\mathbf{s}^{j-1}\right)^\top, \tag{2.80}$$

where $\boldsymbol{\rho}^{n-1} = \left(\left(\mathbf{y}^{n-1}\right)^\top \mathbf{s}^{n-1}\right)^{-1}$. Equation (2.80) decrease the operations from $\mathcal{O}\left(d^3\right)$ in Newton's method to $\mathcal{O}\left(d^2\right)$. BFGS is usually summarized as follows:

- Initialization: $\mathbf{x}^0$, $\mathbf{P}^0$
- Iterations $n \geq 1$: Update $\mathbf{x}^n$ as follows

$$\begin{cases} \mathbf{d}^{n-1} &= -\mathbf{P}^{n-1}\nabla\mathcal{E}(\mathbf{x}^{n-1}) \\ t^{n-1} &= \text{LineSearch}\left\{\mathbf{x}^{n-1}, \mathcal{E}\right\} \\ \mathbf{x}^n &= \mathbf{x}^{n-1} + t^{n-1}\mathbf{d}^{n-1} \\ \text{Compute} & \mathbf{P}^n \quad \text{from Equation (2.80).} \end{cases} \tag{2.81}$$

Nevertheless when d is very large, $\mathcal{O}\left(d^2\right)$ is still time and space consuming. L-BFGS is proposed to use only the past m iteration to approximate the inverse Hessian without storing the entire approximated inverse Hessian $\mathbf{P}^n$. This leads to $\mathcal{O}\left(md\right)$ which is efficient when $m \ll d$ holds. The calculation of the descent direction $\mathbf{d}^n$ is

- Initialization: $\mathbf{q} = \gamma^n\nabla\mathcal{E}(\mathbf{x}^n)$ with $\gamma^n = \left(\left(\mathbf{s}^{n-1}\right)^\top \mathbf{y}^{n-1}\right)\left(\left(\mathbf{y}^{n-1}\right)^\top \mathbf{y}^{n-1}\right)^{-1}$
- Iterations $i = n-1, n-2, \cdots, n-m$, Update $\mathbf{q}$ as follows

$$\begin{cases} \boldsymbol{\alpha}^i &= \boldsymbol{\rho}^i\left(\mathbf{s}^i\right)^\top\mathbf{q} \\ \mathbf{q} &= \mathbf{q} - \boldsymbol{\alpha}^i\mathbf{y}^i \end{cases} \tag{2.82}$$

- Iterations $i = n-m, n-(m-1), \cdots, n-1$, Update $\mathbf{r}$ as follows

$$\begin{cases} \boldsymbol{\beta} &= \boldsymbol{\rho}^i\left(\mathbf{y}^i\right)^\top\mathbf{r} \\ \mathbf{r} &= \mathbf{r} - \mathbf{s}^i\left(\boldsymbol{\alpha}^n - \boldsymbol{\beta}\right) \end{cases} \tag{2.83}$$

- Output: $\mathbf{d}^n = -\mathbf{r}$.

Although some problems in ML are non-convex, L-BFGS is one of the most successful and popular solvers used in practice because of its efficiency. L-BFGS guarantees to converge to a local minimum when solving a non-convex problem.

3

Some Classical Machine Learning Algorithms

Contents

3.1 First-order Algorithms for Machine Learning

Optimization plays an important role in Machine Learning (ML) because most ML problems can be cast as optimization problems. The most common structure of optimization problems in ML is a sum of a loss function and a regularization term. The loss function measures the difference between the ground truth and the prediction. For example, the well-known square loss is commonly used in regression problems and the hinge loss is an example for the purpose of maximum margin classification. The regularization term or short regularizer usually involves a specific norm. For instance, group lasso is an variation of the lasso for feature selection. It can lead to a sparse solution within a group. In addition, practical applications of ML frequently involve a massive and complex data set. This poses a challenge for solving ML problems. The performance of an optimization algorithm can be evaluated based on efficiency, accuracy and generalization [16]. A large number of papers present dedicated optimization algorithms for specific ML problems. Nevertheless, little attention has been devoted to compare the performance of a solver on different ML problems. Therefore, in this chapter we present a comprehensive comparison of some state-of-the-art first-order optimization algorithms for convex optimization problems in ML. The first-order optimization algorithms include the Forward-backward Splitting Method (FBS) [54], the Fast Iterative Shrinkage-thresholding Algorithm (FISTA) [11], the Optimal Subgradient Algorithm (OSGA) [120] and the Primal-Dual Algorithm (PDCP) [31]. This chapter

tackles several smooth and non-smooth ML problems that include a loss function plus a regularizer. We present tasks within dimensionality reduction via compressive sensing, Support Vector Machines (SVMs), group lasso regularizer for grouped feature selection, $\mathcal{L}_{1,\infty}$ regularization for multi-task learning, trace norm regularization for max-margin matrix completion and matrix factorization.

ML problems in this chapter can be formulated as a convex minimization problem consisting of a loss function $\mathcal{F}(\mathbf{x})$ and a regularizer $\mathcal{G}(\mathbf{x})$. We define the energy

$$\mathcal{E}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \lambda\mathcal{G}(\mathbf{x}), \tag{3.1}$$

where λ is a parameter controlling the tradeoff between a good generalization performance and over-fitting. The loss function calculates the disparity between the prediction of a solution and the ground truth. For example, the well known square loss is for the purpose of regression problems. It is defined as,

$$\mathcal{F}_{\ell_2^2}(\mathbf{x}) = \|\mathbf{K}\mathbf{x} - \mathbf{y}\|_2^2, \tag{3.2}$$

where $\mathbf{y}$ represents the vector of true values and $\mathbf{K}\mathbf{x}$ is a vector of predictions. Take the hinge loss as another example. Hinge loss is for the purpose of maximum margin classification. Suppose there is a binary classification problem with labels $l \in \{\pm 1\}$. Then, the hinge loss is given as,

$$\mathcal{F}_{\text{hinge}}(\mathbf{x}) = \max\left\{0, 1 - l(\mathbf{s}^\top\mathbf{x})\right\}, \tag{3.3}$$

where $\mathbf{s}$ is the feature of a sample belonging to the label l and $\mathbf{x}$ is the weight vector. The last example is ϵ-insensitive loss as defined below,

$$\mathcal{F}_\epsilon(\mathbf{x}) = \max\left\{0, \left|\mathbf{s}^\top\mathbf{x} - l\right| - \epsilon\right\}, \tag{3.4}$$

where ϵ controls the range of the acceptable error.

The regularizer usually uses a vector norm. For example, in the case of lasso [154], it is defined as,

$$\mathcal{G}_1(\mathbf{x}) = \|\mathbf{x}\|_1. \tag{3.5}$$

The ℓ_1 regularizer plays an important role in compressed sensing [29, 51]. Group lasso [177] is an extension of the lasso for feature selection. It can lead to a sparse solution within a group. The definition of a group lasso is,

$$\mathcal{G}_{1,2}(\mathbf{x}) = \sum_i \sqrt{l_i}\|\mathbf{x}_i\|_2, \tag{3.6}$$

where $\mathbf{x}_i$ is the i^{th} group within $\mathbf{x}$ and l_i is the length of $\mathbf{x}_i$. Take the trace norm as the last example. The trace norm is the Schatten p-norm with $p = 1$. Recently the trace norm

has been in wide use because the trace norm is a convex relaxation of the rank function [136]. Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ be a matrix. The definition of the trace norm is,

$$\mathcal{G}_{\text{trace}}(\mathbf{X}) = \text{trace}(\sqrt{\mathbf{X}^*\mathbf{X}}) = \sum_{i=1}^{\min\{m,n\}} \sigma_i \tag{3.7}$$

where σ_i denote the singular values of $\mathbf{X}$.

Most of the above mentioned examples lead to non-smooth problems. In general it is more challenging to optimize non-smooth functions than smooth functions. FBS can solve a wide range of convex functions. Nevertheless, it is hard to estimate upper bounds for subgradients of specific convex functions. FISTA [11] enjoys a faster convergence rate of $O\left(1/n^2\right)$ when solving smooth problems involving a $C^{1,1}$ smooth convex function $\mathcal{F}(\mathbf{x})$ with Lipschitz continuous gradient L, i.e.

$$\|\nabla\mathcal{F}(\mathbf{x}) - \nabla\mathcal{F}(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall\ \mathbf{x},\ \mathbf{y}. \tag{3.8}$$

Clearly, a non-smooth loss function does not fulfill this requirement. Thus, when using FISTA to solve a ML problem with a non-smooth loss function, it is necessary to apply smoothing techniques to the loss function. Therefore, one cannot guarantee that the solution provided by FISTA coincides with the optimum of the original non-smooth problem. As a matter of fact, the energy of FISTA's solution may be not equal to a potentially minimal energy. OSGA [120] and the PDCP [31, 172] can handle non-smooth functions. Nevertheless, calculating closed form solutions of proximal mappings for constrained problems in OSGA may be expensive. In order to use PDCP, we have to connect the given problem into a saddle-point formulation.

In what follows, we will evaluate different state-of-the-art first-order algorithms (FBS [54], FISTA [11], OSGA [120] and PDCP [31]) on various ML problems. The first two problems we will consider are dimensionality reduction via compressive sensing and kernel SVM. We will further consider non-smooth convex optimization problems chosen from [172] which include lasso regularization for grouped features, $\mathcal{L}_{1,\infty}$ regularization for multi-task learning and trace norm regularization for matrix factorization and matrix completion. Table 3.2 provides an overview of all considered ML problems in this chapter.

3.2 Experiments

In the following sections, we first give the definitions of the loss function $\mathcal{F}$ and the regularizer $\mathcal{G}$ for each ML problem. The key equations for first-order optimization algorithms include the gradient (subgradient for a non-smooth function) of the loss function $\mathcal{F}$, the regularizer $\mathcal{G}$ and the proximal operator for $\mathcal{F}^*$ and $\mathcal{G}$. For clarification, we do not repeat key equations. In FBS, we will illustrate how to define the subgradient of a non-smooth loss function. For FISTA, first if the loss function is non-smooth, we will

ML problem	Ref.	$\mathcal{F}$	$\mathcal{G}$
Dimensionality Reduction[1]	[65]	square	$\ell_{2,1}$
Linear SVM[2]	[130]	hinge	$\mathbf{x}^\top \mathbf{Q}\mathbf{x}$
Kernel SVM[2]	[35]	hinge	$\mathbf{x}^\top H\mathbf{x}$
Kernel SVM[2]	-	$\mathbf{x}^\top \hat{\mathbf{H}}\mathbf{x}$	$-\sum x_i + \delta(\mathbf{x})$
Feature Selection[3]	[171]	absolute loss	group lasso
Multi-Task Learning[1]	[172]	ϵ-insensitive	$\mathcal{L}_{1,\infty}$
Matrix Factorization[4]	[172]	hinge	trace norm
Matrix Completion[4]	[172]	absolute loss	trace norm

Table 3.1: Overview of Machine Learning (ML) problems considered in this chapter.

describe how to smooth a non-smooth loss function. Next we will show how to derive the gradient of the smoothed loss function. In OSGA, we will use the same scheme as in FBS and FISTA to calculate the subgradient or gradient of $\mathcal{F}$. Thus we do not repeat those equations again. Subsequently, we will present how to formulate a primal problem to a primal dual problem. In light of those key equations, it is easy to derive all necessary equations in every first-order solver for solving ML problems in this chapter.

Furthermore, we will demonstrate results for the different ML problem listed in Table 3.1. In order to provide a convincing comparison, we draw the comparison among solvers with the best-performance parameters which we could achieve. Within a solver, we compare the performance of its parameters with different values. For example, the constant c of Equation (2.51) in FBS is the initial time step size which is hard to choose for an arbitrary non-smooth problem. The other tricky problem is how to select the optimal smoothness parameter when smoothing a non-smooth loss function. We use a smoothness parameter θ to decide the degree of smoothness. We infer that the most suitable smoothness parameter should be related to the objective function. Since the loss function depends on the training data set, we conclude that the training data set also has an impact on the optimal smoothness parameter. In PDCP, we define the ratio between the step sizes of the primal variable and the dual variable, $a = \sqrt{\tau/\sigma}$. In conclusion, we test the influences of c for FBS, θ for FISTA and a for PDCP. We show figures comparing the performance with different values of a and θ. We use the same proximal mappings and parameter settings according to Ahookhosh [4] where OSGA achieves good performance compared with other solvers. Given a ML problem, we evaluate the

[1] MNIST is available at http://yann.lecun.com/exdb/mnist/.
[2] 'svmguide1' is available at http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/.
[3] MEMset Donar is available at http://genes.mit.edu/burgelab/maxent/ssdata/.
[4] '100K MovieLens' is available at http://www.grouplens.org/node/12.

performance of a solver by the running time to converge to an optimal solution. The solver having the best performance means it can reach the optimal solution in the least running time. By choosing the overall best-performance parameter for each solver, we draw a figure to compare FBS, FISTA, OSGA and PDCP. We denote by $\widehat{e}$ the minimal energy. We plot $\log(e^n - \widehat{e})$ versus $\log n$ where n is the number of iterations. Note that in FISTA, no matter the loss function is smoothed or not, we use the non-smoothed loss function to calculate e.

We initialize the primal variable and the dual variable to a null vector. In the last three experiments, we divide the Loss function $\mathcal{F}$ by the number of samples and examine $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$ separately. All algorithms were implemented in Matlab and executed on a 2.66 GHz CPU, running a 64 Bit Windows system.

3.2.1 Dimensionality Reduction

Dimensionality reduction is an important topic in ML. It aims to extract a low-dimensional structure from a high-dimensional data without incurring significant information loss. For more detailed information we refer to Fodor [60], Van der Maaten et al. [156] and Burges [27]. Gao et al. [65] present a semi-supervised dimensionality reduction algorithm named CS-PCA inspired by compressive sensing. In this section, we choose CS-PCA as the first experiment and show how to solve this problem using the first-order optimization algorithms.

We cast CS-PCA as an optimization problem and solve it by FISTA, OSGA and PDCP. We employ the MNIST database[5] consisting of 60k images of handwritten digits from number 0 to 9. Every grayscaled image has a resolution of 28×28. Because MNIST has 10 digits, we have 10 classes. We randomly take 100 images per digit as the training data set and take 50 images per digit as the test data set. We turn each digital image into the corresponding column-major vector representation, which corresponds to an element in the high-dimensional space $\mathbb{R}^k$. Since the size of each image is 28×28, we have $k = 28^2$. Suppose there are $\mathrm{n_{trn}}$ training samples $\mathbf{s}^i_{\mathrm{trn}} \in \mathbb{R}^k, i = 1, \cdots, \mathrm{n_{trn}}$ and $\mathrm{n_{tst}}$ test samples $\mathbf{s}^i_{\mathrm{tst}} \in \mathbb{R}^k, i = 1, \cdots, \mathrm{n_{tst}}$. First, CS-PCA builds a representation matrix based on the training dataset,

$$\mathbf{\Psi} = \left[\mathbf{s}^1_{\mathrm{trn}}, \cdots, \mathbf{s}^{\mathrm{n_{trn}}}_{\mathrm{trn}}\right] \in \mathbb{R}^{k \times \mathrm{n_{trn}}}. \tag{3.9}$$

According to compressive sensing, random projection from a high dimension space to a low dimension space preserves the distances between the points. CS-PCA applies a random matrix $\mathbf{\Phi} \in \mathbb{R}^{d \times k}$ to $\mathbf{\Psi}$ and to each test sample, $d \ll k$. The new representation matrix is denoted by $\mathbf{K} = \mathbf{\Phi}\mathbf{\Psi}$ and the test samples become $\mathbf{s}^i = \mathbf{\Phi}\mathbf{s}^i_{\mathrm{tst}}$. Let $\mathbf{x}$ be the primal variable associated with the $\mathrm{i^{th}}$ test sample $\mathbf{s}^i$. CS-PCA aims to find a sparse

[5]The data set is available at http://yann.lecun.com/exdb/mnist/.

solution of Equation (3.10) to represent each new sample $\mathbf{s}^i$,

$$\underset{\mathbf{x}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{K}\mathbf{x}-\mathbf{s}^i\|_2^2+\lambda\|\mathbf{x}\|_{\ell_2/\ell_1}. \tag{3.10}$$

We define $\mathcal{F}(\mathbf{x})=\frac{1}{2}\|\mathbf{K}\mathbf{x}-\mathbf{s}^i\|_2^2$ and $\mathcal{G}(\mathbf{x})=\|\mathbf{x}\|_{\ell_2/\ell_1}=\sum_{\text{ind}_g}\|\mathbf{x}_{\text{ind}_g}\|_2$ where $\mathbf{x}_{\text{ind}_g}$ is the sub-vector $\mathbf{x}$ corresponding to the 10 classes. Equation (3.10) is an unconstrained convex minimization problem. We solve the above equation for each test sample $\mathbf{s}^i, i=1,\cdots,\mathrm{n}_{\text{tst}}$ and denote by $\widehat{\mathbf{x}}^i$ the optimal solution of Equation (3.10). Thus $\widehat{\mathbf{x}}^i$ is the sparse solution of $\mathbf{s}^i$. The last step is to apply PCA on the obtained sparse solutions $\widehat{\mathbf{x}}^i$ to reduce the dimensionality. In what follows, we will illustrate how to solve Equation (3.10). The final result is shown in Figure 3.3.

FISTA for Dimensionality Reduction. In this case, $\mathcal{F}(\mathbf{x})=\frac{1}{2}\|\mathbf{K}\mathbf{x}-\mathbf{s}^i\|_2^2$ is continuously differentiable. Its gradient is,

$$\nabla\mathcal{F}=\mathbf{K}^\top(\mathbf{K}\mathbf{x}-\mathbf{s}^i), \tag{3.11}$$

and the Lipschitz constant of $\nabla\mathcal{F}$ is $\|\mathbf{K}^\top\mathbf{K}\|$.

PDCP for Dimensionality Reduction. Let the dual variable be $\mathbf{y}\in\mathbb{R}^n$. According to the convex conjugate, $\mathcal{F}(\mathbf{x})$ can be defined as,

$$\mathcal{F}(\mathbf{x})=\max_{\mathbf{y}}\left\langle\mathbf{K}\mathbf{x}-\mathbf{s}^i,\mathbf{y}\right\rangle-\frac{1}{2}\|\mathbf{y}\|_2^2. \tag{3.12}$$

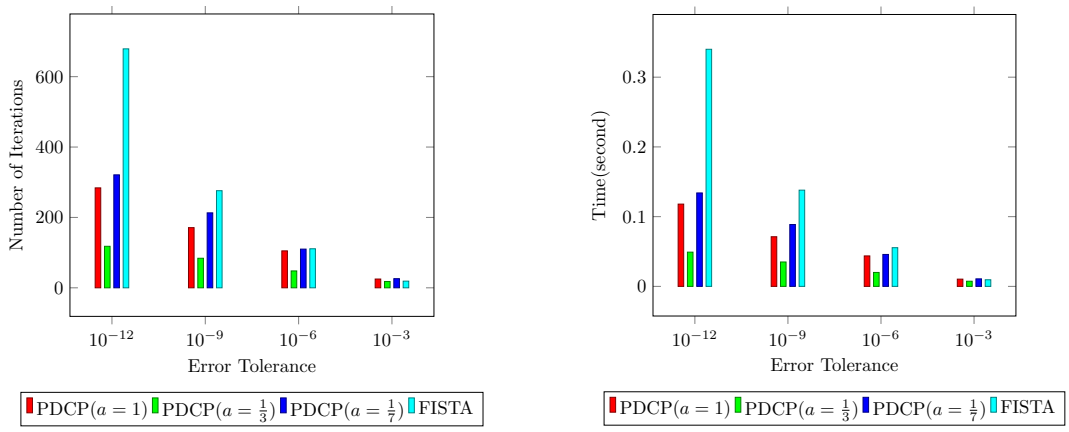
Thus,the conjugate function $\mathcal{F}^*$ of $\mathcal{F}$ is, $\mathcal{F}^*(\mathbf{x})=\left\langle\mathbf{s}^i,\mathbf{y}\right\rangle+\frac{1}{2}\|\mathbf{y}\|_2^2$. Hence we update $\mathbf{x}^n,\mathbf{y}^n$ as follows,

$$\begin{cases} \mathbf{y}^n &= \left(\mathbf{y}^{n-1}+\sigma(\mathbf{K}\overline{\mathbf{x}}^{n-1}-\mathbf{s}^i)\right)\frac{1}{1+\sigma} \\ \widetilde{\mathbf{x}}^n &= \mathbf{x}^{n-1}-\tau\mathbf{K}^*\mathbf{y}^n \\ \mathbf{x}^n_{\text{ind}_g} &= \max\left\{\|\widetilde{\mathbf{x}}^n_{\text{ind}_g}\|_2-\tau\lambda,0\right\}\frac{\widetilde{\mathbf{x}}^n_{\text{ind}_g}}{\|\widetilde{\mathbf{x}}^n_{\text{ind}_g}\|_2} \end{cases} \tag{3.13}$$

Comparison of Dimensionality Reduction. Let $\widehat{e}$ be the minimal energy of

$$\mathcal{E}(\mathbf{x})=\mathcal{F}(\mathbf{x})+\lambda\mathcal{G}(\mathbf{x}). \tag{3.14}$$

In this experiment $\widehat{e}$ is attained by running PDCP (with $a=11$) 1e4 times. The running times per iteration of PDCP, OSGA and FISTA are 4.16×10^{-4}, 6.25×10^{-4} and 5.00×10^{-4} seconds, respectively. In addition, the loss function is smooth. Thus we only demonstrate the influence of different values of $a=\sqrt{\tau/\sigma}$, which is the square root of the ratio of the step size of the primal variables to the step size of the dual variables in PDCP. From Figure 3.1, we can observe that FISTA is as fast as PDCP when approaching to the error



(a) **(b)**

Figure 3.1: Performance evaluation of the PDCP with different values of a as well as the FISTA in the application of dimensionality reduction. (a) refers to the number of iterations and (b) refers to CPU times (sec) to drop the error $(e^n - \widehat{e})$ below the error tolerance 10^{-12}, 10^{-9}, 10^{-6} and 10^{-3}.

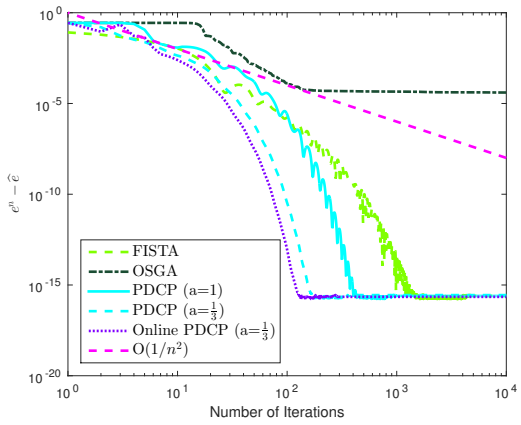


Figure 3.2: Log-log plot of the error $e^n - \widehat{e}$ to the number of iterations n in the experiment of dimensionality reduction.

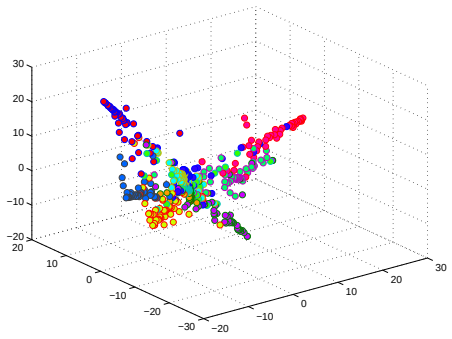


Figure 3.3: Classification in 3D: Applying PCA on the obtained sparse solutions to reduce the dimensionality of three.

tolerance 10^{-3}. But PDCP, especially with $a = \frac{1}{3}$, is faster to reach the smaller error tolerance, for instance, 10^{-12}.

Figure 3.2 illustrates the performance of FISTA, OSGA, PDCP and Online PDCP. It shows that FISTA is the fastest algorithm in the first 5 iterations. However it falls behind in the later iterations and the PDCP method is fastest to converge towards 10^{-15}. In practice convergence rates are much better than the theoretical convergence rates. In Figure 3.3, we show the 3D visualization result of CS-PCA solved by PDCP.

3.2.2 Linear SVM

Support Vector Machines (SVMs) are very popular ML models for classification and regression analysis. In this section we consider the problem of soft margin SVM. As one type of linear SVM, soft margin SVM is a supervised ML algorithm which provides simplicity and efficiency. The data set we use is a non-linear separable data set called 'svmguide1'[6] with 2 classes. This dataset has 3089 training samples and 4e3 test samples and each sample has 4 features. Pele et al. [130] propose an efficient and highly expressive non-linear feature map method. This method embeds single features and pairs of features into a vector space. This involves discretization and interpolation of individual feature values and feature pairs. Pele et al. [130] map the training set into a new feature space and show the efficiency and accuracy of a linear SVM trained on the rebuilt training sets. In this experiment, we use the non-linear feature map method [130] and assign the value of each feature into 8 bins. Then we train a soft margin SVM on the new feature space. For simplicity, the offset is absorbed into the primal variable, so we do not introduce an additional parameter.

Suppose there are $\mathrm{n_{trn}}$ training samples $\mathbf{s}^i \in \mathbb{R}^d$ belonging to the label l^i, $i = 1, \cdots, \mathrm{n_{trn}}$. The soft margin SVM aims to output weights $\mathbf{x} \in \mathbb{R}^d$ whose linear combination predicts the label of $\mathbf{s}^i$,

$$\begin{aligned} \underset{\mathbf{x},\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + p\sum_{i=1}^{\mathrm{n_{trn}}} \xi_i \\ \text{subject to} \quad & 1 - \xi_i - l^i({\mathbf{s}^i}^\top \mathbf{x}) \leq 0, \quad \xi_i \geq 0, \forall i \in \{1, \cdots, \mathrm{n_{trn}}\} \end{aligned} \tag{3.15}$$

where $\mathbf{Q} \in \mathbb{R}^{d\times d}$ is a diagonal matrix. The first $d-1$ elements down the main diagonal of $\mathbf{Q}$ are one and the last element is zero which corresponds to the offset. The parameter $p \in \mathbb{R}^+$ is a tradeoff parameter usually chosen by grid search [34] or cross-validation. Ben-Hur and Weston [14] show that similar decision boundaries of linear SVMs can be obtained using different combinations of SVM parameters. De Bie et al. [49] propose a convex way to optimize p for the soft margin SVM and in [44] the authors present a methodology to select parameters for SVM regression. A small p makes constraints easily to be ignored and leads to a large margin. A soft margin SVM becomes a hard margin SVM when p approaches $+\infty$. Since our object is to verify the performance of solvers, we do not employ a systematic method to calculate the parameter p. Therefore we simply set $p = 0.1$. To formulate an unconstrained problem, we move the linear constraints into the energy function,

$$\underset{\mathbf{x}}{\text{minimize}} \quad p\sum_{i=1}^{\mathrm{n_{trn}}} \max\left\{0, 1 - l^i \mathbf{x}^\top \mathbf{s}^i\right\} + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x}. \tag{3.16}$$

[6]The data set is available at http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/.

Let $\mathcal{F}$ and $\mathcal{G}$ be,

$$\mathcal{F}(\mathbf{x}) = p\sum_{i=1}^{\mathrm{n_{trn}}} \max\left\{0, 1 - l^i \mathbf{x}^\top \mathbf{s}^i\right\} \quad \text{and} \quad \mathcal{G}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x}. \tag{3.17}$$

Note that we can also write $\mathcal{F}\{\mathbf{x}\} = \sum_{i=1}^{\mathrm{n_{trn}}} \max\left\{0, 1 - l^i \mathbf{x}^\top \mathbf{s}^i\right\}$ and $\mathcal{G}(\mathbf{x}) = \frac{1}{2\lambda}\mathbf{x}^\top \mathbf{Q}\mathbf{x}$ with $\lambda = \frac{1}{p}$. Next we show the iteration equations of PDCP. Note we omit the equations of FISTA since they are the same as in Section 3.2.4. The reason that we do not use the dual form is that the dual form of the soft margin SVM is an optimization problem with a box and an affine constrain. The affine constrain is introduced to take the offset into account. There is no 'simple' solution of the proximal mapping to a domain satisfying the box and affine constrain at the same time.

PDCP for Linear SVM By introducing Lagrange multipliers $\mathbf{y}$ and $\mathbf{z}$, Equation (3.15) can be written as,

$$\min_{\mathbf{x},\boldsymbol{\xi}} \max_{\mathbf{y},\mathbf{z}} \left\{ \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + p\sum_{i=1}^{\mathrm{n_{trn}}} \xi_i + \sum_{i=1}^{\mathrm{n_{trn}}} y_i \left[1 - \xi_i - l^i(\mathbf{s}^{i\top}\mathbf{x})\right] - \sum_{i=1}^{\mathrm{n_{trn}}} z_i \xi_i \right\}, \tag{3.18}$$

with dual variables $y_i \geq 0$ and $z_i \geq 0$. We take the gradient of (3.18) with respect to $\boldsymbol{\xi}$. Then we get $p - y_i - z_i = 0, i = 1, \cdots, \mathrm{n_{trn}}$. The equation can be simplified by substituting $p = y_i + z_i$ into Equation (3.18), leading to

$$\min_{\mathbf{x}} \max_{\mathbf{y}} \left\{ \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \sum_{i=1}^{\mathrm{n_{trn}}} y_i \left[1 - l^i(\mathbf{s}^{i\top}\mathbf{x})\right] \right\}, \tag{3.19}$$

where $\{y_i \mid 0 \leq y_i \leq p, i = 1, \cdots, \mathrm{n_{trn}}\}$. Let the set be $P = \{\mathbf{y} \in \mathbb{R}^{\mathrm{n_{trn}}} : \forall y_i \in [0, p]\}$ and $\mathbf{K} = \begin{bmatrix} -l_1\mathbf{s}^1 & \cdots & -l_{\mathrm{n_{trn}}}\mathbf{s}^{\mathrm{n_{trn}}} \end{bmatrix}^\top$. The loss function can be formulated as,

$$F(\mathbf{x}) = \max_{\mathbf{y}} \left\{ \langle \mathbf{K}\mathbf{x} + 1, \mathbf{y} \rangle - \delta_P(\mathbf{y}) \right\}. \tag{3.20}$$

Thus we have

$$F^*(\mathbf{y}) = -\sum_{i=1}^{\mathrm{n_{trn}}} y_i + \delta_P(\mathbf{y}). \tag{3.21}$$

The iteration equations of $\mathbf{x}^n, \mathbf{y}^n$ can be concluded as:

$$\begin{cases} \mathbf{y}^n &= \left[\mathbf{y}^{n-1} + \sigma(\mathbf{K}\overline{\mathbf{x}}^{n-1} + 1)\right]_P \\ \widetilde{\mathbf{x}}^n &= \mathbf{x}^{n-1} - \tau\mathbf{K}^\top \mathbf{y}^n \\ \mathbf{x}^n &= \mathbf{A}\widetilde{\mathbf{x}}^n \end{cases} \tag{3.22}$$

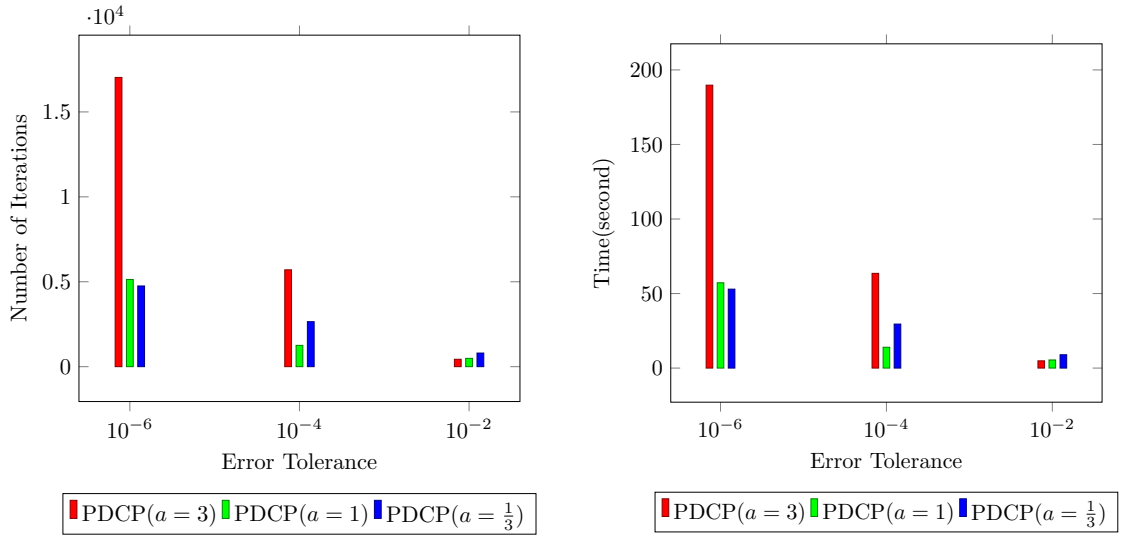


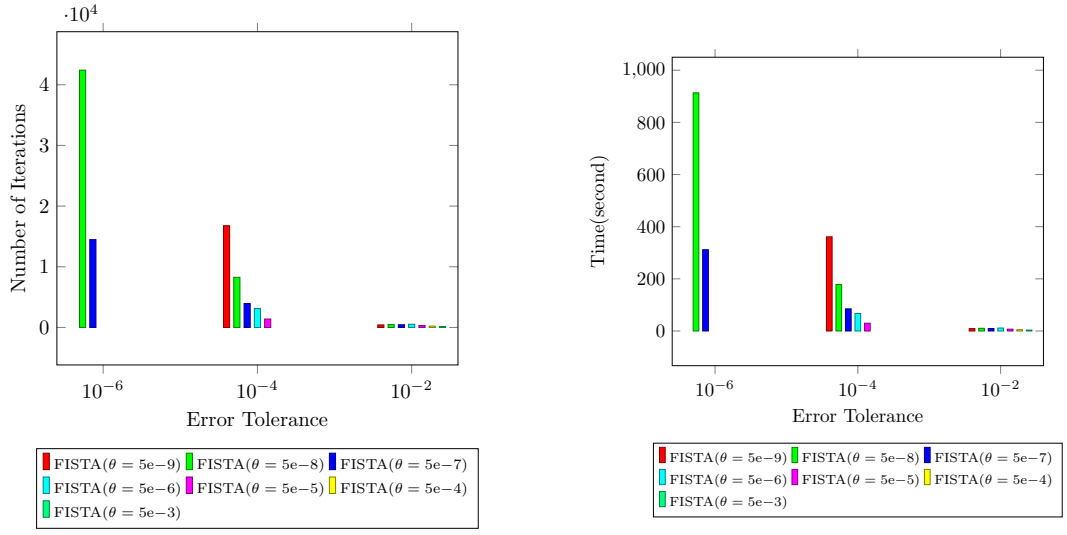
Figure 3.4: Performance evaluation of the PDCP with different values of a in the application of linear SVM. (a) refers to the number of iterations and (b) refers to CPU times (sec) to drop the error $(e^n - \widehat{e})$ below the error tolerance 10^{-2}, 10^{-4} and 10^{-6} within a maximum number of 5e5 iterations.

where $\mathbf{A} = \text{diag}\left(\frac{p}{p+\tau}, \cdots, \frac{p}{p+\tau}, 1\right)$.

Comparison of Linear SVM. In the experiment of linear SVM, $\widehat{e}$ is selected by running PDCP ($a = 11$) 5e5 times. The running times per iteration of PDCP , OSGA and FISTA are 11.14×10^{-3}, 12.14×10^{-3} and 21.54×10^{-3} seconds, respectively. Figure 3.4 shows that PDCP with $a = \frac{1}{3}$ takes more time to drop the error below 10^{-2}. Nevertheless PDCP with $a = \frac{1}{3}$ takes less time to drop the error below 10^{-6}.

Figure 3.5 shows the performance of FISTA with different values of the smoothness parameter. We can observe that FISTA has the best performance when the smoothness parameter equals 5×10^{-7}. We can see FISTA is faster to converge within the error tolerance 10^{-2} with $\epsilon = 0.005$. Whereas FISTA with $\epsilon = 0.005$ fails to drop the error below 10^{-4}. By comparing Figure 3.4 and Figure 3.5, FISTA becomes very slow to converge to the optimal solution.

Figure 3.6 shows FISTA has the best performance until around the 10^{th} iteration. From about the 100^{th} iteration to the 800^{th} iteration, OSGA reaches the minimal values. In the later iterations, PDCP outperforms the others and converges to the optimal solution $\widehat{e}$. We also observe that FISTA and OSGA cannot reach the optimal solution within 5e5 iterations. Figure 3.6 shows that the empirical convergence rate of PDCP is better than $O(1/n^2)$ although the theoretical convergence rate of PDCP is $O(1/n)$. We test the SVM solved by PDCP ($a = 11$) on the test data set with 4000 samples and the resulting accuracy is 0.8627.



(a) **(b)**

Figure 3.5: Performance evaluation of the FISTA with different values of the smoothness parameter θ in the application of linear SVM. (a) refers to the number of iterations and (b) refers to the CPU times (sec) to drop the error $(e^n - \widehat{e})$ below the error tolerance. The empty bars indicate that the algorithm failed to drop the error below the error tolerance within a maximum number of 5e5 iterations.

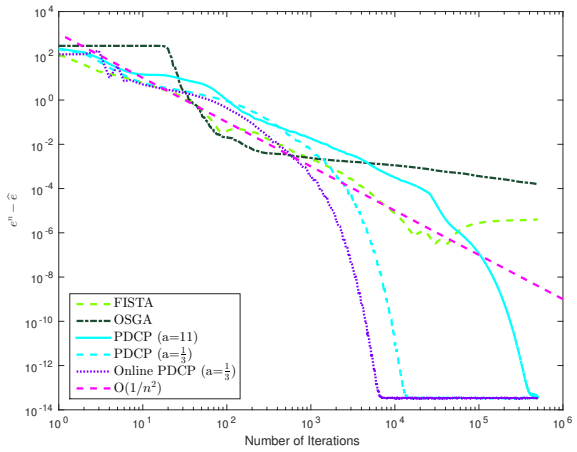


Figure 3.6: Log-log plot of the error $e^n - \widehat{e}$ to the number of iterations n in the experiment of linear SVM.

3.2.3 Kernel SVM

Kernel SVM is a popular tool for the classification of a non-linear separable data set. In this section, we train a classifier with the kernel trick. We show how to solve the dual form

of this primal problem [35] by FISTA and PDCP. Since this dual form is a constrained optimization problem, we use OSGA to solve the original primal problem. We choose Gaussian radial basis functions $\mathcal{K}(\mathbf{x},\mathbf{v}) = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{v}\|^2}{2\rho^2}\right\}$ as the kernel function and ρ is the bandwidth parameter to determine the effective dimensionality of the high-dimensional space. We propose a method to set ρ according to the training set. Suppose there are $\mathrm{n_{trn}}$ training samples. We define a variable generated by the training data set

$$\nu_{(i,j)}(\rho) = \exp\left\{-\frac{\|\mathbf{s}_i - \mathbf{s}_j\|^2}{2\rho^2}\right\},\ i \in \{1,\cdots,\mathrm{n_{trn}}\},\ j \in \{1,\cdots,\mathrm{n_{trn}}\},\ i \neq j. \tag{3.23}$$

$\mathbf{s}_i - \mathbf{s}_j$ is a pair of samples in a training data set and there are $\mathrm{n_{pair}} = \frac{1}{2}\mathrm{n_{trn}}(\mathrm{n_{trn}}-1)$ pairs. Similar to PCA, we set the value of ρ such that the variance of ν can reach its maximum. The principle of the proposed method is to make full use of the training samples. We show that this method can select a reasonable value for the width parameter ρ of the Gaussian kernel in synthetic and real-world experiments. If we randomly choose the value of ρ, SVMs leads to a good classification if and only if the value of ρ is proper, for instance, a small ρ leads to overfitting. The variance of $\nu_{(i,j)}(\rho)$ is,

$$\mathrm{Var}(\nu) = \frac{1}{\mathrm{n_{pair}}}\sum_{i\neq j}\left(\exp\left(-\frac{\|\mathbf{s}_i-\mathbf{s}_j\|^2}{2\rho^2}\right) - \bar{\nu}\right)^2 \text{ and } \bar{\nu} = \frac{1}{\mathrm{n_{pair}}}\sum_{i\neq j}\exp\left(-\frac{\|\mathbf{s}_i-\mathbf{s}_j\|^2}{2\rho^2}\right) \tag{3.24}$$

is the arithmetic mean of ν. The variance Var can be seen as a function of ρ. The graph of the function Var of argument ρ roughly is a "bell-curve" shape. On the left side of the optimal ρ, the gradients of Var are positive. By contrast, on the right side of the optimal ρ, the gradient of Var are negative. Thus we use a bisection method to calculate the optimal ρ. The gradient of ρ is,

$$\begin{aligned}\frac{8}{\mathrm{n_{pair}}\rho^3}\Bigg(&\sum_{i\neq j}\|\mathbf{s}_i-\mathbf{s}_j\|^2\exp\left(-\frac{\|\mathbf{s}_i-\mathbf{s}_j\|^2}{\rho^2}\right) - \\ &\frac{1}{\mathrm{n_{pair}}}\sum_{i\neq j}\|\mathbf{s}_i-\mathbf{s}_j\|^2\exp\left(-\frac{\|\mathbf{s}_i-\mathbf{s}_j\|^2}{2\rho^2}\right)\sum_{i\neq j}\exp\left(-\frac{\|\mathbf{s}_i-\mathbf{s}_j\|^2}{2\rho^2}\right)\Bigg).\end{aligned} \tag{3.25}$$

In the following, we show the classification result of the synthetic experiments using a bandwidth calculated by this method.

Instead of using $\mathbf{x}$ as the primal variable in Equation (3.15), we introduce the primal variable according to Chapelle [35]. Suppose there is a reproducing kernel Hilbert spaces $\mathcal{H}$. In this case, we rewrite Equation (3.15) as,

$$\underset{\mathcal{Q}\in\mathcal{H}}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{Q}\|_{\mathcal{H}}^2 + q\sum_{i=1}^{\mathrm{n_{trn}}}\max\{0, 1 - l_i\mathcal{Q}(\mathbf{s}_i)\}. \tag{3.26}$$

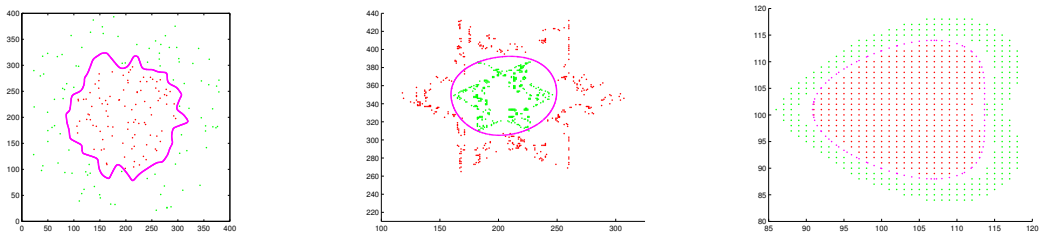


Figure 3.7: Illustration of synthetic experiments for classification with kernel SVM.

Because the subgradient evaluated at the optimal solution vanishes, the form of a solution we seek can be defined as,

$$\widehat{\mathcal{Q}^*}(\mathbf{x}) = \sum_{i=1}^{\mathrm{n_{trn}}} \beta_i \mathcal{K}(\mathbf{s}_i, \mathbf{x}), \tag{3.27}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{\mathrm{n_{trn}}}$ is referred to as the new primal variable. Therefore we can write Equation (3.15) with respect to the new primal variable $\boldsymbol{\beta}$,

$$\frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}\boldsymbol{\beta} + q\sum_{i=1}^{\mathrm{n_{trn}}} \max\left\{0, 1 - l_i \mathbf{H}_{.i}^\top \boldsymbol{\beta}\right\}, \tag{3.28}$$

where $\mathbf{H}_{.i}$ is the i^{th} column of $\mathbf{H}$ and $\mathbf{H}_{i,j} = \mathcal{K}(\mathbf{s}_i, \mathbf{s}_j)$. Generally, when ρ approaches to plus infinity, $\mathbf{H}$ approaches to be a matrix whose elements are all one. Whereas when ρ approaches to zero, $\mathbf{H}$ becomes to be an identity matrix. Now we define

$$F(\boldsymbol{\beta}) = q\sum_{i=1}^{\mathrm{n_{trn}}} \max\left\{0, 1 - l_i \mathbf{H}_{.i}^\top \boldsymbol{\beta}\right\} \quad \text{and} \quad G(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}\boldsymbol{\beta}. \tag{3.29}$$

We let $q = 1$ in our experiment.

Although the primal variable $\boldsymbol{\beta}$ has a different meaning compared with the primal variable $\mathbf{x}$ in the linear SVM, both use the hinge loss. However, in this primal form there is no offset. Thus we can use the dual form to avoid using the non-smooth hinge loss in the primal form. We proceed by showing how to solve the kernel SVM in dual form using FISTA.

FISTA for Kernel SVM. Since $\mathcal{F}(\boldsymbol{\beta})$ is a hinge loss, we introduce the dual variable $\mathbf{y}$ by its conjugate,

$$\mathcal{F}(\boldsymbol{\beta}) = q\max_{\mathbf{y}} \sum_{i=1}^{\mathrm{n_{trn}}} \left\{\left\langle y_i, 1 - l_i \mathbf{H}_{.i}{}^\top \boldsymbol{\beta}\right\rangle - \mathcal{F}^*(\mathbf{y})\right\} \tag{3.30}$$

where

$$\mathcal{F}^*(\mathbf{y}) = \begin{cases} 0 & \mathbf{y} \in P \\ +\infty & \mathbf{y} \notin P \end{cases} \tag{3.31}$$

is a indicator function and $P = \{\mathbf{y} \in \mathbb{R}^{\mathrm{n_{trn}}} \mid \forall y_i \in [0,1]\}$. By KKT condition, we have $\boldsymbol{\beta} = \boldsymbol{\ell} \cdot \mathbf{y}$, $\boldsymbol{\ell} = \left[l_1, \cdots, l_{\mathrm{n_{trn}}},\right]$. Thus, the dual form is:

$$\max_{\mathbf{y}}(\frac{1}{2} - q)\left(\boldsymbol{\ell} \cdot \mathbf{y}\right)^\top \mathbf{H}\left(\boldsymbol{\ell} \cdot \mathbf{y}\right) + q\sum_{i=1}^{\mathrm{n_{trn}}} y_i - \mathcal{F}^*(\mathbf{y}). \tag{3.32}$$

By reversing the sign, we attain

$$\min_{\mathbf{y}}(q - \frac{1}{2})\left(\boldsymbol{\ell} \cdot \mathbf{y}\right)^\top \mathbf{H}\left(\boldsymbol{\ell} \cdot \mathbf{y}\right) - q\sum_{i=1}^{\mathrm{n_{trn}}} y_i + \mathcal{F}^*(\mathbf{y}). \tag{3.33}$$

By changing the variable $\mathbf{y}$ of Equation (3.33) to $\mathbf{x}$ for uniformity, we further obtain

$$\min_{\mathbf{x}}\ (q - \frac{1}{2})\left(\boldsymbol{\ell} \cdot \mathbf{x}\right)^\top \mathbf{H}\left(\boldsymbol{\ell} \cdot \mathbf{x}\right) - q\sum_{i=1}^{\mathrm{n_{trn}}} x_i + \mathcal{F}^*(\mathbf{x}). \tag{3.34}$$

For the simplicity of calculating the resolvent of $\partial\mathcal{G}$, in the dual form we let

$$\mathcal{F}_{\mathrm{dual}}(\mathbf{x}) = (q - \frac{1}{2})\left(\boldsymbol{\ell} \cdot \mathbf{x}\right)^\top \mathbf{H}\left(\boldsymbol{\ell} \cdot \mathbf{x}\right) \tag{3.35}$$

and

$$\mathcal{G}_{\mathrm{dual}}(\mathbf{x}) = -q\sum_{i=1}^{\mathrm{n_{trn}}} x_i + \mathcal{F}^*(\mathbf{x}). \tag{3.36}$$

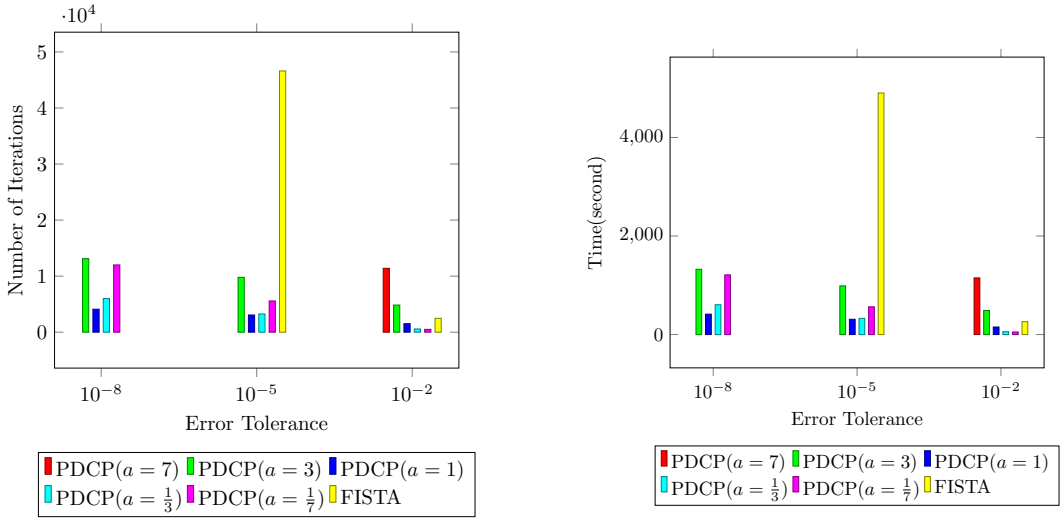
Because $\mathcal{F}^*$ is an indicator function, we can further let $\delta(\mathbf{x}) = \mathcal{F}^*(\mathbf{x})$. Thus Equation (3.34) becomes

$$\min_{\mathbf{x}}\ (q - \frac{1}{2})\left(\boldsymbol{\ell} \cdot \mathbf{x}\right)^\top \mathbf{H}\left(\boldsymbol{\ell} \cdot \mathbf{x}\right) - q\sum_{i=1}^{\mathrm{n_{trn}}} x_i + \delta(\mathbf{x}). \tag{3.37}$$

We write $\left(\boldsymbol{\ell} \cdot \mathbf{x}\right)^\top \mathbf{H}\left(\boldsymbol{\ell} \cdot \mathbf{x}\right)$ as $\mathbf{x}^\top\widehat{\mathbf{H}}\mathbf{x}$ where $\widehat{\mathbf{H}} = \mathbf{H} \cdot (\boldsymbol{\ell}\boldsymbol{\ell}^\top)$. Thus we have the conjugate of

$$\mathcal{F}_{\mathrm{dual}}(\mathbf{x}) = (q - \frac{1}{2})\mathbf{x}^\top\widehat{\mathbf{H}}\mathbf{x}. \tag{3.38}$$

The gradient of the loss function is $\nabla\mathcal{F}_{\mathrm{dual}}(\mathbf{x}) = 2(q - \frac{1}{2})\widehat{\mathbf{H}}\mathbf{x}$. And the Lipschitz constant of $\nabla\mathcal{F}(\mathbf{x})$ is $2(q - \frac{1}{2})\|\widehat{\mathbf{H}}\|$. Note that no matter what value of ρ is, $\mathbf{H}$ always is an approximate positive semidefinite matrix which means $\mathbf{H}$ has very small negative eigenvalues. Therefore, $q > \frac{1}{2}$ can make this model be an approximate convex model.



(a) **(b)**

Figure 3.8: Performance evaluation of the PDCP with different values of a and the FISTA in the application of kernel SVM. The (a) refers to the number of iterations and (b) refers to CPU times (sec) to drop the error $(e^n - \widehat{e})$ below the error tolerance 10^{-2}, 10^{-5} and 10^{-8} within a maximum number of 5e3 iterations.

PDCP for Kernel SVM. Now we continue to demonstrate the involved equations for PDCP. The conjugate of Equation (3.38) and its gradient are

$$\mathcal{F}^*_{\text{dual}}(\mathbf{x}) = \frac{\mathbf{x}^\top \widehat{\mathbf{H}}^{-1}\mathbf{x}}{4p-2} \quad \text{and} \quad \nabla\mathcal{F}^*_{\text{dual}}(\mathbf{x}) = \frac{\widehat{\mathbf{H}}^{-1}\mathbf{x}}{2p-1}. \tag{3.39}$$

Thus, the primal-dual model is,

$$\min_{\mathbf{x}}\max_{\mathbf{y}}\left\{\langle \mathbf{x},\mathbf{y}\rangle - \frac{\mathbf{y}^\top \widehat{\mathbf{H}}^{-1}\mathbf{y}}{4p-2} - q\sum_{i=1}^{\mathrm{n_{trn}}} x_i + \delta(\mathbf{x})\right\}. \tag{3.40}$$

The PDCP algorithm for the given task can then be summarized as

$$\left\{\begin{array}{rcl} \mathbf{y}^n & = & (\mathbf{I} + \frac{\sigma\widehat{\mathbf{H}}^{-1}}{2p-1})^{-1}(\mathbf{y}^{n-1} + \sigma\overline{\mathbf{x}}^{n-1}) \\ \mathbf{x}^n & = & \left[\mathbf{x}^{n-1} - \tau\mathbf{y}^n + \tau q\right]_P. \end{array}\right. \tag{3.41}$$

To calculate $(\mathbf{I} + \frac{\sigma\widehat{\mathbf{H}}^{-1}}{2q-1})^{-1}$, we use the Woodbury matrix identity [169], i.e.

$$(\mathbf{I} + \tfrac{\sigma\widehat{\mathbf{H}}^{-1}}{2q-1})^{-1} = \tfrac{(2q-1)}{\sigma}\widehat{\mathbf{H}} - \tfrac{(2q-1)}{\sigma}\widehat{\mathbf{H}}\left(\mathbf{I} + \tfrac{(2q-1)}{\sigma}\widehat{\mathbf{H}}\right)^{-1}\tfrac{(2q-1)}{\sigma}\widehat{\mathbf{H}} \tag{3.42}$$

to avoid calculating the inverse matrix of $\widehat{\mathbf{H}}$.

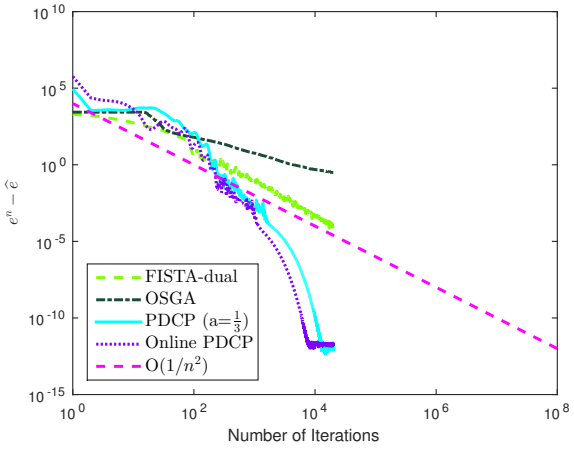


Figure 3.9: Log-log plot of the error $e^n - \widehat{e}$ to the number of iterations n in the experiment of kernel SVM.

Comparison of Kernel SVM. We select $\widehat{e}$ by running PDCP (a=$\frac{1}{3}$) 2e4 times. The running times per iteration of PDCP, OSGA and FISTA are 0.1009, 0.3000 and 0.1052 seconds, respectively. We compare the performances of PDCP and FISTA in Figure 3.8. In Figure 3.9, we can see that the convergence rate of FISTA coincides with its theoretical rate after the 100^{th} iteration. PDCP and Online PDCP converging with a much better rate outperform FISTA and OSGA in the later iterations. We test the kernel SVM solved by PC CP (a=$\frac{1}{3}$) running 2e4 times on the test data set with 4000 samples and the accuracy is 0.906309.

3.2.4 Grouped Feature Selection

High-dimensional regression and classification are challenges in the field of science. The group lasso [177], used as a regularizer, yields a solution with grouped sparsity when solving a problem of high-dimensional regression and classification. Following the experimental setting of Yang et al. [171], we conduct experiments of high-dimensional classification. In this experiment, we use the so-called MEMset Donar dataset[7]. Each sample in this dataset consists of a feature described by a sequence of $\{A, C, G, T\}$ of length 7. The training set consists of replicated 8415 true and 179438 false donor sites, and the testing set has replicated 4208 true and 89717 false donor sites. We use dummy variables to encode each original feature of the samples. We generate group features with up to three-way interactions between the 7 positions. When each element itself is treated as a group, from the original feature sequence we can get 7 groups; when two-way interactions is considered, there are $\frac{7\times6}{2} = 21$ combinations; similarly for three-way interactions, we can draw $\frac{7\times6\times5}{3\times2} = 35$ combinations. This results in g groups where $g = 63$. Because each

[7]The data set is available at http://genes.mit.edu/burgelab/maxent/ssdata/.

original feature is expressed as one of $\{A, C, G, T\}$, for the first 7 groups, each group is decoded as a 4-digit number, 1000, 0100, 0010, 0001. Likewise for one of 21 groups of two-way interactions, it is decoded by a 16-digit number, for example 0000 0000 0000 0001. A 64-digit number expresses each group of three-way interaction. Therefore the primal variable is formulated as $\mathbf{x} = \left[\mathbf{x}_1^\top, \mathbf{x}_2^\top, \cdots, \mathbf{x}_g^\top\right]$ where the number of attributes is $d = 2604 = 7 \times 4 + 21 \times 16 + 35 \times 64$. We denotes by $d_{\text{ind}_g} \in \{4, 16, 64\}$ the length of each group.

Two non-smooth loss functions are examined in this experiment, namely absolute loss and hinge loss. Suppose there are n_{trn} training samples. The primal variable is $\mathbf{x} \in \mathbb{R}^d$, where d is the dimension of the attribute space. The absolute loss is defined as:

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{\text{n}_{\text{trn}}} \left| \mathbf{s}^{i\top}\mathbf{x} - l^i \right| \tag{3.43}$$

where $\mathbf{s}^i$ is the feature of the i^{th} sample and l^i is the label of the i^{th} sample. The hinge loss is:

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{\text{n}_{\text{trn}}} \max(0, 1 - l^i \mathbf{x}^\top \mathbf{s}^i). \tag{3.44}$$

The group lasso is defined as follows

$$G(\mathbf{x}) = \sum_{\text{ind}_g=1}^{g} \sqrt{d_{\text{ind}_g}} \|\mathbf{x}_{\text{ind}_g}\|_2, \tag{3.45}$$

where ind_g is the index of groups.

FBS for Grouped Feature Selection. In this section, we demonstrate the equation of the subgradient of the loss function used in FBS. Calculus leads to the subgradient of the absolute loss as follows,

$$\nabla\mathcal{F} = \sum_{i \in P_1} \mathbf{s}^i - \sum_{i \in P_2} \mathbf{s}^i, \tag{3.46}$$

where sets P_1, P_2 are defined as $P_1 = \left\{ i \mid \mathbf{s}^{i\top}\mathbf{x} - l_i > 0 \right\}$ and $P_2 = \left\{ i \mid \mathbf{s}^{i\top}\mathbf{x} - l_i < 0 \right\}$. The subgradient of the hinge loss with respect to $\mathbf{x}$ is,

$$\nabla\mathcal{F} = \sum_{i \in P_3} -l_i \mathbf{s}^i, \tag{3.47}$$

where the set P_3 is defined as $P_3 = \left\{ i \mid 1 - l_i \mathbf{x}^\top \mathbf{s}^i > 0 \right\}$.

FISTA for Grouped Feature Selection. Since the absolute loss and the hinge loss are both non-smooth. We apply smoothing techniques to the absolute loss and the hinge loss before using FISTA. We denote by θ the smoothness parameter. The absolute loss

and the hinge loss compute the sum of all samples. This leads to the absolute loss,

$$\mathcal{F}_{\text{abs}}(\mathbf{x}) = \sum_{i=1}^{\text{n}_{\text{trn}}} \mathcal{F}_i(\mathbf{x}) \quad \text{with} \quad \mathcal{F}_i(\mathbf{x}) = \begin{cases} \frac{(\mathbf{s}_i{}^\top \mathbf{x} - l_i)^2}{2\theta} + \frac{\theta}{2} & \left|\mathbf{s}_i{}^\top \mathbf{x} - l_i\right| \leq \theta \\ \left|\mathbf{s}_i{}^\top \mathbf{x} - l_i\right| & \text{otherwise} \end{cases}. \tag{3.48}$$

Thus the corresponding gradient is as follows,

$$\nabla\mathcal{F}_{\text{abs}}(\mathbf{x}) = \sum_{i=1}^{\text{n}_{\text{trn}}} \nabla\mathcal{F}_i(\mathbf{x}) \quad \text{where} \quad \nabla\mathcal{F}_i(\mathbf{x}) = \begin{cases} \frac{\mathbf{s}_i{}^\top \mathbf{x} - l_i}{\theta}\mathbf{s}_i & \left|\mathbf{s}_i{}^\top \mathbf{x} - l_i\right| \leq \theta \\ \mathbf{s}_i & \mathbf{s}_i{}^\top \mathbf{x} - l_i > \theta \\ -\mathbf{s}_i & \mathbf{s}_i{}^\top \mathbf{x} - l_i < -\theta \end{cases}. \tag{3.49}$$

Likewise, the hinge loss is,

$$\mathcal{F}_{\text{hinge}}(\mathbf{x}) = \sum_{i=1}^{\text{n}_{\text{trn}}} \mathcal{F}_i(\mathbf{x}) \quad \text{with} \quad \mathcal{F}_i(\mathbf{x}) = \begin{cases} 0 & 1 - l_i\mathbf{x}^\top\mathbf{s}_i < -\theta \\ \frac{(1 - l_i\mathbf{x}^\top\mathbf{s}_i + \theta)^2}{4\theta} & \left|1 - l_i\mathbf{x}^\top\mathbf{s}_i\right| \leq \theta \\ 1 - l_i\mathbf{x}^\top\mathbf{s}_i & 1 - l_i\mathbf{x}^\top\mathbf{s}_i > \theta \end{cases}. \tag{3.50}$$

The gradient $\nabla\mathcal{F}_{\text{hinge}}(\mathbf{x})$ results as

$$\nabla\mathcal{F}_{\text{hinge}}(\mathbf{x}) = \sum_{i=1}^{\text{n}_{\text{trn}}} \nabla\mathcal{F}_i(\mathbf{x}) \quad \text{where} \quad \nabla\mathcal{F}_i(\mathbf{x}) = \begin{cases} 0 & 1 - l_i\mathbf{x}^\top\mathbf{s}_i < -\theta \\ \frac{-l_i(1 - l_i\mathbf{x}^\top\mathbf{s}_i + \theta)}{2\theta}\mathbf{s}_i & \left|1 - l_i\mathbf{x}^\top\mathbf{s}_i\right| \leq \theta \\ -l_i\mathbf{s}_i & 1 - l_i\mathbf{x}^\top\mathbf{s}_i > \theta \end{cases}. \tag{3.51}$$

The smoothing techniques lead to piecewise smooth functions for both of the absolute loss and the hinge loss. It is easy to derive the subgradient functions $\nabla\mathcal{F}$. However, the Lipschitz constants of $\nabla\mathcal{F}$ for absolute loss and hinge loss are not computable. For this reason, we use FISTA with backtracking.

PDCP for Grouped Feature Selection using Absolute Loss. Suppose that the absolute loss is defined as:

$$\mathcal{F}(\mathbf{x}) = \|\mathbf{K}\mathbf{x} - \boldsymbol{\ell}\|_1, \tag{3.52}$$

where $\mathbf{K} = \left[\mathbf{s}^1 \ \cdots \ \mathbf{s}^{\text{n}_{\text{trn}}}\right]^\top$, and $\boldsymbol{\ell} = [l_1 \ \cdots \ l_{\text{n}_{\text{trn}}}]^\top$. We introduce the dual variable $\mathbf{y} \in \mathbb{R}^{\text{n}_{\text{trn}}}$ by the conjugate function,

$$\mathcal{F}^*(\mathbf{y}) = \langle \boldsymbol{\ell}, \mathbf{y} \rangle + \delta_P(\mathbf{y}) \quad \text{where} \quad \delta_P(\mathbf{y}) = \begin{cases} 0 & \mathbf{y} \in P \\ +\infty & \mathbf{y} \notin P \end{cases} \tag{3.53}$$

and the set $P = \{\mathbf{y} \in \mathbb{R}^{\text{n}_{\text{trn}}} : \forall y_i \in [-1, 1]\}$. This leads to the following loss function,

$$\mathcal{F}(\mathbf{x}) = \max_{\mathbf{y}} \{\langle \mathbf{K}\mathbf{x} - \boldsymbol{\ell}, \mathbf{y} \rangle - \delta_P(\mathbf{y})\}. \tag{3.54}$$

The scheme of PDCP reads,

$$\begin{cases} \mathbf{y}^n &= \left[\mathbf{y}^{n-1} + \sigma(\mathbf{K}\overline{\mathbf{x}}^{n-1} - \boldsymbol{\ell})\right]_P \\ \widetilde{\mathbf{x}}^n &= \mathbf{x}^{n-1} - \tau\mathbf{K}^\top\mathbf{y}^n \\ \mathbf{x}^n_{\mathrm{ind}_g} &= \max\left\{\|\widetilde{\mathbf{x}}^n_{\mathrm{ind}_g}\|_2 - \sqrt{d_{\mathrm{ind}_g}}\tau\lambda, 0\right\}\frac{\widetilde{\mathbf{x}}^n_g}{\|\widetilde{\mathbf{x}}^n_{\mathrm{ind}_g}\|_2} \end{cases} \tag{3.55}$$

where $\mathbf{x}_{\mathrm{ind}_g}$ and $\widetilde{\mathbf{x}}_{\mathrm{ind}_g}$ are the $\mathrm{ind}_g{}^{th}$ group of $\mathbf{x}$ and $\widetilde{\mathbf{x}}$, $\mathrm{ind}_g = 1, \cdots, g$.

PDCP for Grouped Feature Selection using Hinge Loss. We define the hinge loss with the matrix $\mathbf{K}_h$:

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{\mathrm{n_{trn}}} \max(0, 1 - l_i\mathbf{x}^\top\mathbf{s}^i) = \|\max(0, 1 + \mathbf{K}_h\mathbf{x})\|_1, \tag{3.56}$$

where $\mathbf{K}_h = \begin{bmatrix} -l_1\mathbf{s}^1 & \cdots & -l_{\mathrm{n_{trn}}}\mathbf{s}^{\mathrm{n_{trn}}} \end{bmatrix}^\top$. Let the dual variable be $\mathbf{y} \in \mathbb{R}^{\mathrm{n_{trn}}}$ and a set P be $P = \{\mathbf{y} \in \mathbb{R}^{\mathrm{n_{trn}}} : \forall y_i \in [0, 1]\}$. The convex conjugate is

$$\mathcal{F}^*(\mathbf{y}) = -\sum_{i=1}^{\mathrm{n_{trn}}} y_i + \delta_P(\mathbf{y}). \tag{3.57}$$

The loss function is summarized as,

$$\mathcal{F}(\mathbf{x}) = \max_{\mathbf{y}} \left\{\langle \mathbf{K}_h\mathbf{x} + 1, \mathbf{y}\rangle - \delta_P(\mathbf{y})\right\}. \tag{3.58}$$

Therefore, the iteration equations of $\mathbf{x}^n, \mathbf{y}^n$ are,

$$\begin{cases} \mathbf{y}^n &= \left[\mathbf{y}^{n-1} + \sigma(\mathbf{K}_h\overline{\mathbf{x}}^{n-1} + 1)\right]_P \\ \widetilde{\mathbf{x}}^n &= \mathbf{x}^{n-1} - \tau\mathbf{K}_h^*\mathbf{y}^n \\ \mathbf{x}^n_{\mathrm{ind}_g} &= \max\left\{\|\widetilde{\mathbf{x}}^n_{\mathrm{ind}_g}\|_2 - \sqrt{d_{\mathrm{ind}_g}}\tau\lambda, 0\right\}\frac{\widetilde{\mathbf{x}}^n_g}{\|\widetilde{\mathbf{x}}^n_{\mathrm{ind}_g}\|_2}. \end{cases} \tag{3.59}$$

Comparison of Grouped Feature Selection. Now we summarize our experimental findings. We test two different loss functions: absolute loss and hinge loss with group lasso as a regularizer for the case $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$. In all the cases, the optimal solution $\widehat{e}$ is chosen by running Online PDCP 10^5 times. For the absolute loss, the running time per iteration of PDCP is 1.76×10^{-2} seconds; the running time per iteration of FISTA is 6.40×10^{-2}; and the running time per iteration of OSGA is 4.40×10^{-2}. For the hinge loss, the running time per iteration of PDCP is 1.64×10^{-2} seconds; the running time per iteration of FISTA is 4.33×10^{-2}; and the running time per iteration of OSGA is 3.30×10^{-2}. Next, we summarize the results for absolute loss and hinge loss separately.

First we consider the result using absolute loss. Figure 3.10 shows the performances of PDCP with different values of a and FISTA with different values of smoothness parameter

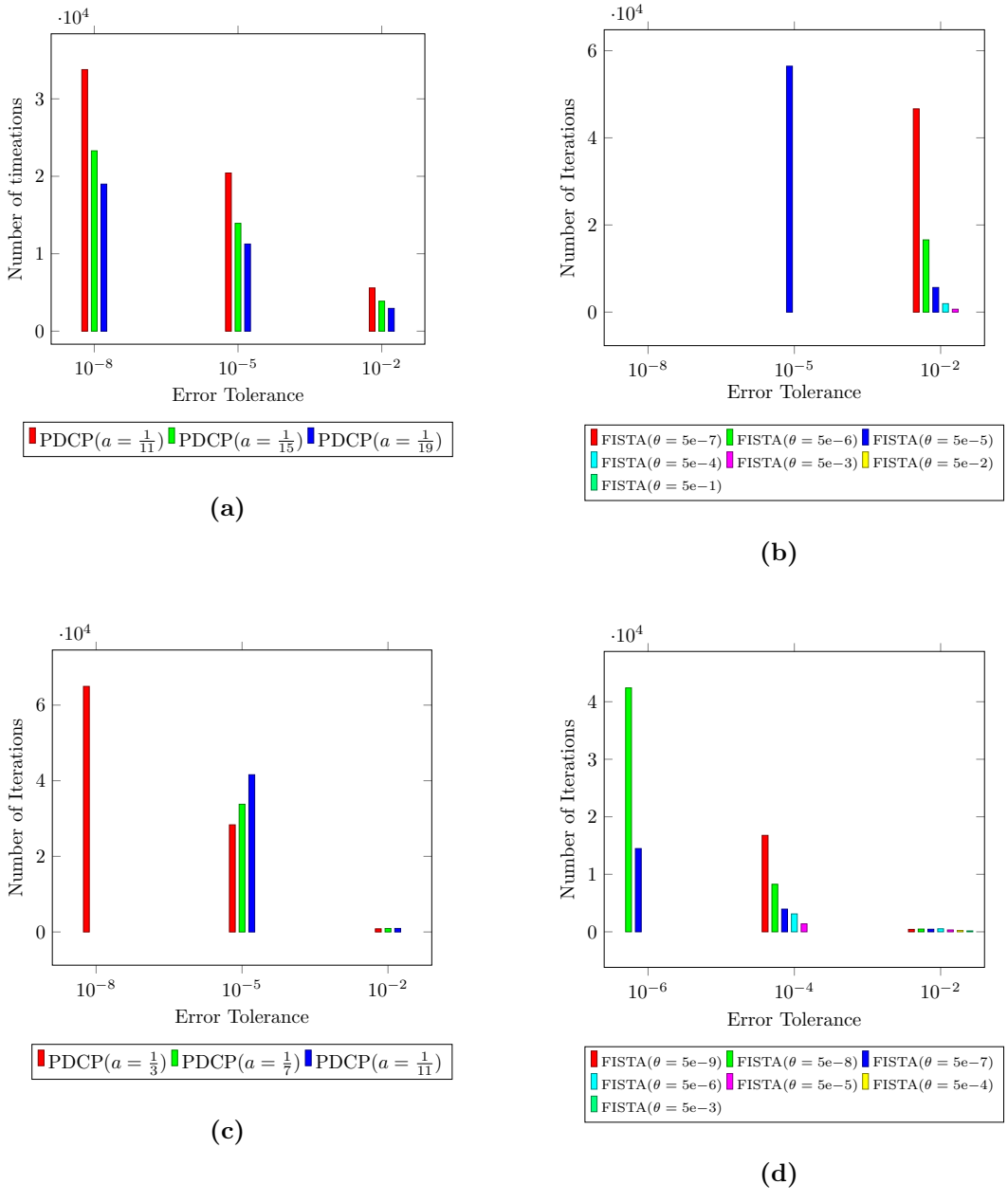
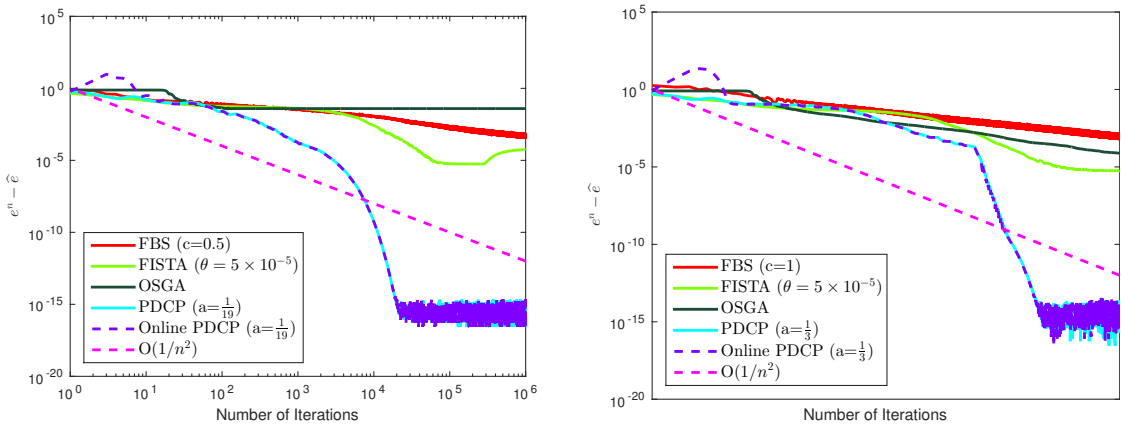


Figure 3.10: Performance evaluation of the PDCP with different values of a ((a) and (c)) and the FISTA with different values of θ ((b) and (d)) in the application of grouped feature selection with the absolute loss when $\lambda = 10^{-3}$ ((a) and (b)) and $\lambda = 10^{-5}$ ((c) and (d)). The figures show the needed number of iterations (y axis) to drop the error below a certain error tolerance (x axis) within a maximum number of 1e5 iterations.



(a) **(b)**

Figure 3.11: Log-log plot of error $e^n - \widehat{e}$ to the number of iterations n in the experiment of grouped feature selection with the absolute loss. (a) shows results for $\lambda = 10^{-3}$ and (b) shows results for $\lambda = 10^{-5}$.

θ. In Figure 3.10b and Figure 3.10d, the vacancies of bars indicate that the algorithms fail to drop the error below the error tolerance within a maximum number of 1e5 iterations.

From Figure 3.10 we can see that the number of iterations of PDCP is much less than FISTA. As shown in Figure 3.11, OSGA is better than FISTA and FBS from the 100^{th} to 1000^{th} iteration. Whereas in the later iterations, PDCP outperforms OSGA, FISTA and FBS. FBS achieves a convergence rate of $O(1/\sqrt{n})$. We observe fluctuations of FISTA since it is not a monotone algorithm. Within 10^6 iterations only PDCP begin to converge. The empirical convergence rate of PDCP is better than its theoretical convergence rate.

Finally we evaluate the grouped feature selection using hinge loss. Figure 3.12 shows the performances of PDCP with different values of a and FISTA with different values of smoothness parameter θ. In Figure 3.12b and Figure 3.12d, the vacancies of bars indicate that the algorithms fail to drop the error below the error tolerance within a maximum number of 1e5 iterations.

From Figure 3.13 we can see that the running time of PDCP is much less than FISTA. We can see OSGA is similar to and slightly better than FBS in Figure 3.13a and Figure 3.13b respectively. The convergence rate of PDCP again is more better than its theory convergence rate.

3.2.5 Multi-Task Learning

Multi-task learning is to learn several similar and related tasks at the same time, using the commonality among the tasks. In this experiment, we conduct supervised machine learn to classify digits. We use the MNIST dataset[8] and we choose 50 training samples for

[8]The data set is available at http://yann.lecun.com/exdb/mnist/.

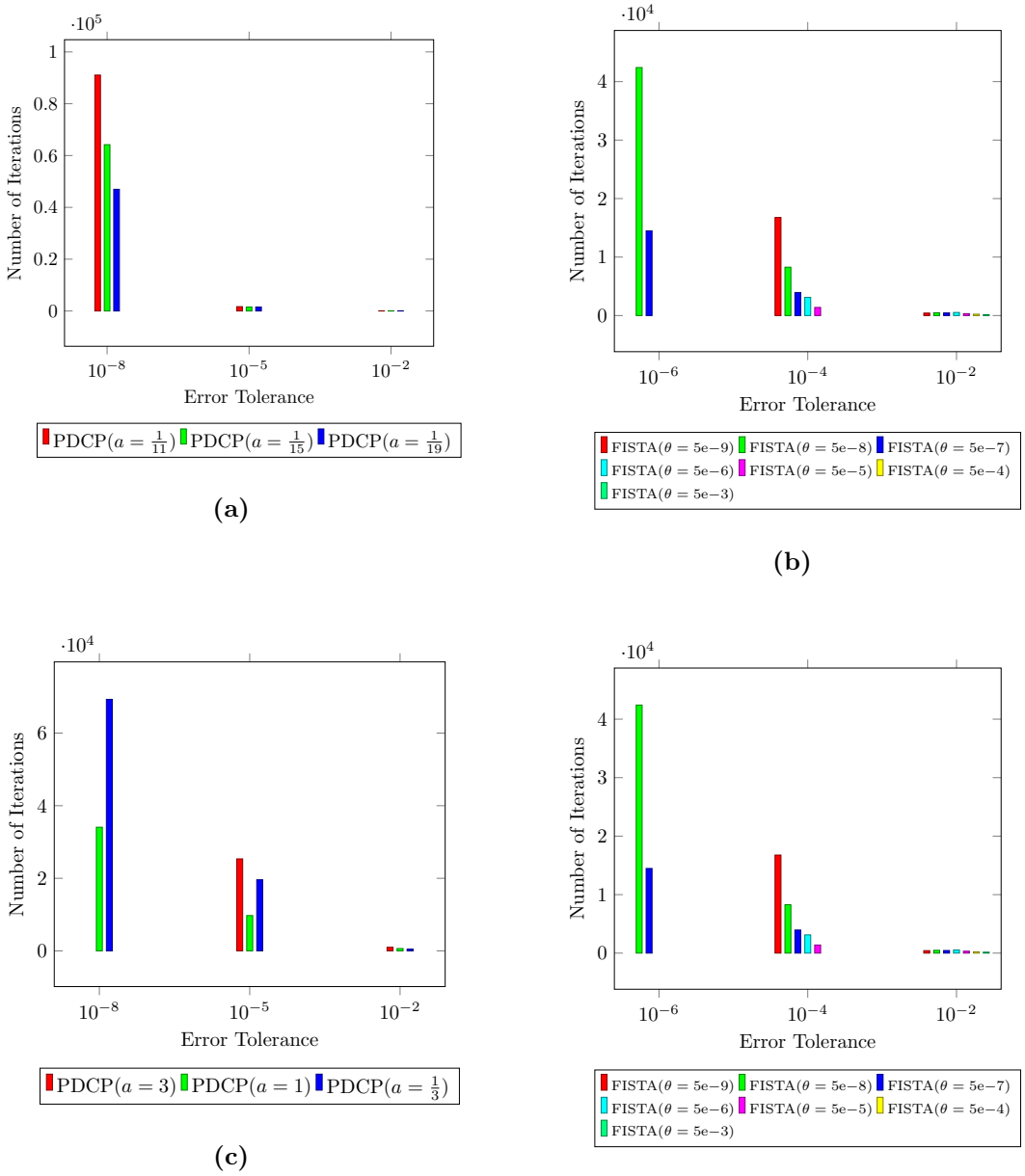


Figure 3.12: Performance evaluation of PDCP with different values of a ((a) and (c)) and FISTA with different values of θ ((b) and (d)) in the application of grouped feature selection with the hinge loss when $\lambda = 10^{-3}$ ((a) and (b)) and $\lambda = 10^{-5}$ ((c) and (d)). The figures show the needed number of iterations (y axis) to drop the error below a certain error tolerance (x axis) within a maximum number of 1e5 iterations.

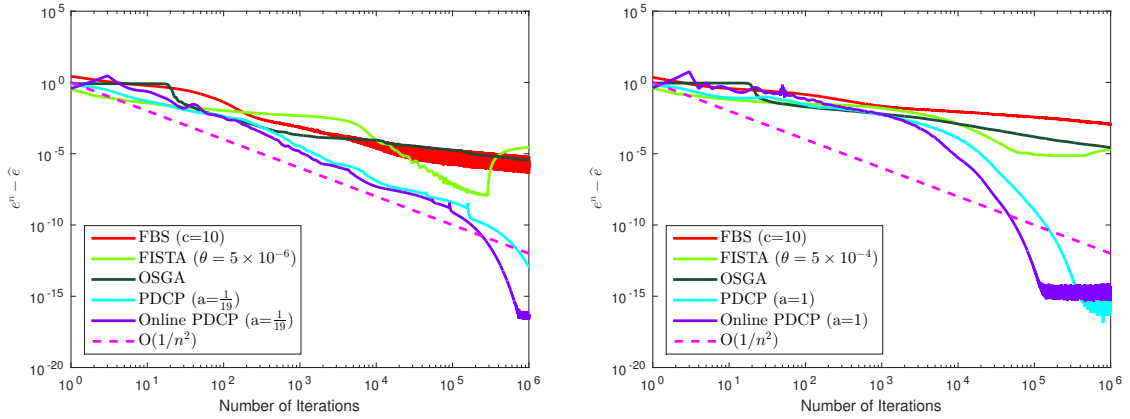


Figure 3.13: Log-log plot of the error $e^n - \widehat{e}$ to the number of iterations n in the experiment of grouped feature selection with the hinge loss. (a) shows the results for $\lambda = 10^{-3}$ and (b) shows the results for $\lambda = 10^{-5}$.

each task. The dimensionality of each sample is reduced to 12 using Principal Component Analysis (PCA). Each feature of samples is denoted by $\mathbf{s}^i \in \mathbb{R}^d,\ i = 1, \cdots, 100$. For each feature of samples, we use l_i to denote its category, either digit $l_i = 0$ or digit $l_i = 1$, and use k_i to denote the task where $k_i = 1$ denotes the task of assigning 0 to a vector of feature and $k_i = 2$ denotes the task of assigning 1 to a vector of feature. The loss functions of this experiment are the absolute loss and the ϵ-insensitive loss with $\epsilon = 0.01$. The regularizer is $\mathcal{L}_{1,\infty}$ norm. Suppose there are $\mathrm{n_{trn}}$ training samples and m tasks. The absolute loss is,

$$\mathcal{F}_{\mathrm{abs}}(\mathbf{X}) = \sum_{i=1}^{\mathrm{n_{trn}}} \left| \mathbf{s}^{i\top} \mathbf{X}_{.k_i} - l_i \right|, \tag{3.60}$$

and ϵ-insensitive loss is,

$$\mathcal{F}_{\epsilon}(\mathbf{X}) = \sum_{i=1}^{\mathrm{n_{trn}}} \max\left(\left| \mathbf{s}^{i\top} \mathbf{X}_{.k_i} - l_i \right| - \epsilon, 0 \right) \tag{3.61}$$

where $\mathbf{X} = [\mathbf{X}_{.1}, \mathbf{X}_{.2}, \cdots, \mathbf{X}_{.m}] \in \mathbb{R}^{d \times m}$ and $\mathbf{X}_{.i}$ is the $\mathrm{i^{th}}$ column of the matrix $\mathbf{X}$. The $\mathcal{L}_{1,\infty}$ norm is defined as a regularizer on the primal variable $\mathbf{X}$,

$$\mathcal{G}(\mathbf{X}) = \|\mathbf{X}\|_{1,\infty} = \sum_{i=1}^{d} \|\mathbf{X}_{i.}\|_{\infty} = \sum_{i=1}^{d} \max_{1 \le j \le m} |\mathbf{X}_{i,j}|. \tag{3.62}$$

For simplicity, we reformulate the above equations. We transform $\mathbf{X}$ to a vector $\mathbf{x}$ in

column-major vector representation. Then we create a sparse matrix $\mathbf{K} \in \mathbb{R}^{\mathrm{n_{trn}} \times (d \times m)}$,

$$\mathbf{K} = \underbrace{\left[\begin{array}{ccc} \mathbf{K}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{K}_m \end{array}\right]}_{d \times m \text{ columns}} \Bigg\} \mathrm{n_{trn}} \text{ rows}$$

where the rows of $\mathbf{K}_i, i = 1, \cdots, m$ are samples related to the task i. For example, in the task one of identifying digit 0, we select all of the samples of digit 0 and assign each sample to one row of $\mathbf{K}_1$. Since we have 50 samples of digit 0, the row of $\mathbf{K}_1$ is 50.

Therefore the absolute loss function can be written as:

$$F_{\mathrm{abs}}(\mathbf{x}) = \|\mathbf{K}\mathbf{x} - \boldsymbol{\ell}\|_1 \tag{3.63}$$

and $\boldsymbol{\ell}$ is the label of the corresponding examples $\boldsymbol{\ell} \in \mathbb{R}^{\mathrm{n_{trn}}}$. We already introduce how to solve absolution loss function in the experiment of feature selection. So in the next sections, we concentrate on the ϵ-insensitive loss function. The ϵ-insensitive loss can be written as:

$$F_\epsilon(\mathbf{x}) = \|\max\{|\mathbf{K}\mathbf{x} - \boldsymbol{\ell}| - \epsilon, 0\}\|_1. \tag{3.64}$$

FBS for Multi-Task Learning. The subgradient of the ϵ-insensitive loss $\mathcal{F}(\mathbf{x})$ with respect to $\mathbf{x}$ is

$$\nabla \mathcal{F}_\epsilon(\mathbf{x}) = \sum_{i \in P_1} \mathbf{K}_{i\cdot}^\top - \sum_{j \in P_2} \mathbf{K}_{j\cdot}^\top$$

where

$$P_1 = \{j \in \mathbb{R} : \mathbf{K}_{j\cdot}\mathbf{x} - l_j > \epsilon, j = 1, \cdots, \mathrm{n_{trn}}\} \tag{3.65}$$

and

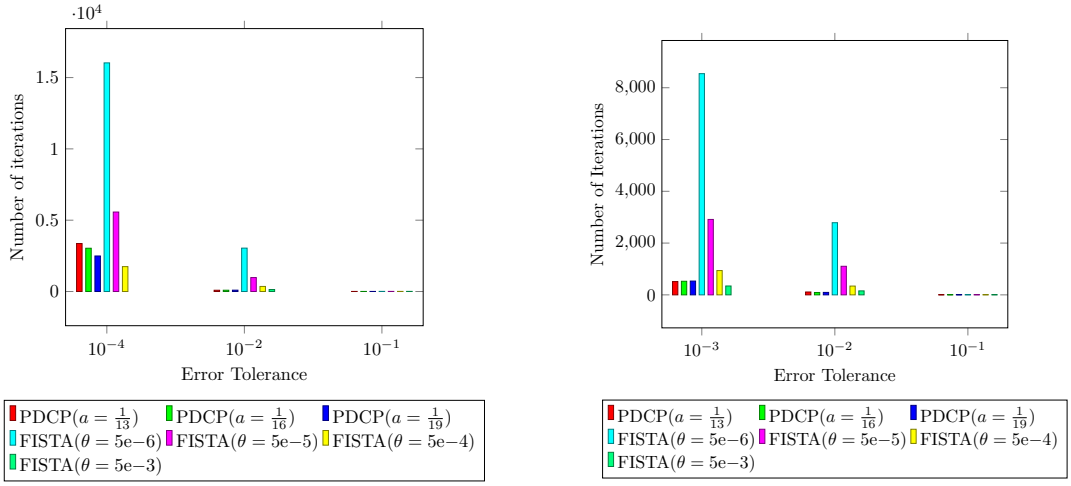
$$P_2 = \{j \in \mathbb{R} : \mathbf{K}_{j\cdot}\mathbf{x} - l_j < -\epsilon, j = 1, \cdots, \mathrm{n_{trn}}\}. \tag{3.66}$$

FISTA for Multi-Task Learning. We describe how to smooth the ϵ-insensitive loss. First we make a reasonable assumption that $\epsilon \geq \theta$. Then we rewrite

$$\begin{aligned} \mathcal{F}_\epsilon(\mathbf{x}) &= \|\max\{|\mathbf{K}\mathbf{x} - \boldsymbol{\ell}| - \epsilon, 0\}\|_1 && (3.67) \\ &= \sum_{i=1}^{i=\mathrm{n_{trn}}} |\max\{|\mathbf{K}_{i\cdot}\mathbf{x} - \boldsymbol{\ell}| - \epsilon, 0\}| && (3.68) \\ &= \sum_{i=1}^{i=\mathrm{n_{trn}}} \mathcal{F}_i(\mathbf{x}) && (3.69) \end{aligned}$$

where

$$\mathcal{F}_i(\mathbf{x}) = |\max\{|\mathbf{K}_{i\cdot}\mathbf{x} - \boldsymbol{\ell}| - \epsilon, 0\}|.$$



(a) **(b)**

Figure 3.14: Performance evaluation of the PDCP with different values of a and the FISTA with different values of θ when $\lambda = 10^{-3}$ (a) and $\lambda = 10^{-5}$ (b) in the application of multi-task learning with the absolute loss. The figures show the needed number of iterations (y axis) to drop the error below a certain error tolerance (x axis) within a maximum number of 1e6 iterations.

We smooth each subfunction $\mathcal{F}_i$ as

$$\mathcal{F}_i(\mathbf{x}) = \begin{cases} 0 & -\epsilon + \theta \leq r_i \leq \epsilon - \theta \\ \frac{1}{4\epsilon}(r_i - \epsilon + \theta)^2 & \epsilon - \theta < r_i < \epsilon + \theta \\ \frac{1}{4\epsilon}(-r_i - \epsilon + \theta)^2 & -\epsilon - \theta < r_i < -\epsilon + \theta \\ |r_i| - \epsilon & \text{otherwise} \end{cases} \tag{3.70}$$

where $r_i = \mathbf{K}_{i.}\mathbf{x} - \boldsymbol{\ell}$. This leads to the following gradient of $\mathcal{F}_i$,

$$\nabla\mathcal{F}_i(\mathbf{x}) = \begin{cases} \mathbf{0} & -\epsilon + \theta \leq r_i \leq \epsilon - \theta \\ \frac{1}{2\epsilon}(r_i - \epsilon + \theta)\mathbf{K}_{i.} & \epsilon - \theta < r_i < \epsilon + \theta \\ -\frac{1}{2\epsilon}(-r_i - \epsilon + \theta)\mathbf{K}_{i.} & -\epsilon - \theta < r_i < -\epsilon + \theta \\ \mathbf{K}_{i.} & r_i \geq \epsilon + \theta \\ -\mathbf{K}_{i.} & r_i \leq -\epsilon - \theta \end{cases}.$$

PDCP for Multi-Task Learning. Suppose the dual variable is $\mathbf{y} \in \mathbb{R}^{2\times \mathrm{n_{trn}}}$ which is composed of two vectors $\mathbf{y}^1 \in \mathbb{R}^{\mathrm{n_{trn}}}$ and $\mathbf{y}^2 \in \mathbb{R}^{\mathrm{n_{trn}}}$. We define a set

$$P = \left\{\mathbf{y} \in \mathbb{R}^{2\times \mathrm{n_{trn}}} : \forall y_i^1, y_i^2 \geq 0, y_i^1 + y_i^2 \leq 1\right\}. \tag{3.71}$$

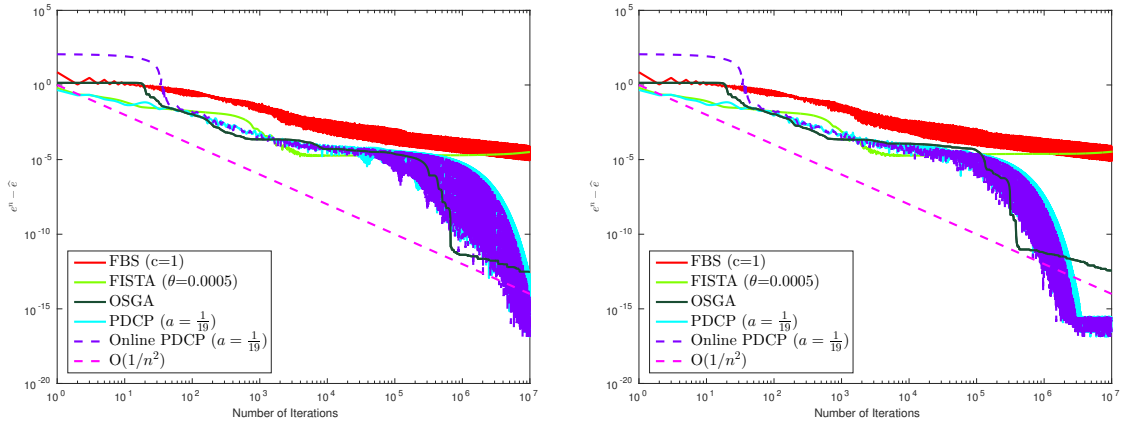


Figure 3.15: Log-log plot of the error $e^n - \widehat{e}$ to the number of iterations n in the experiment of multi-task learning with the absolute loss. (a) shows results for $\lambda = 10^{-3}$ and (b) shows results for $\lambda = 10^{-5}$.

Then the convex conjugate of the ϵ-Insensitive loss (cf. Equation (3.64)) is

$$\mathcal{F}^*(\mathbf{y}) = \langle \boldsymbol{\ell}_\epsilon + \epsilon, \mathbf{y} \rangle + \delta_P(\mathbf{y}) \tag{3.72}$$

where $\boldsymbol{\ell}_\epsilon = \left[\boldsymbol{\ell}; -\boldsymbol{\ell}\right]$ and $\mathbf{K}_\epsilon \in R^{2\times \mathrm{n_{trn}}\times(d\times m)}, \mathbf{K}_\epsilon = \begin{bmatrix} \mathbf{K} \\ -\mathbf{K} \end{bmatrix}$.

By introducing the dual variable $\mathbf{y}$, $\mathcal{F}$ can be defined as

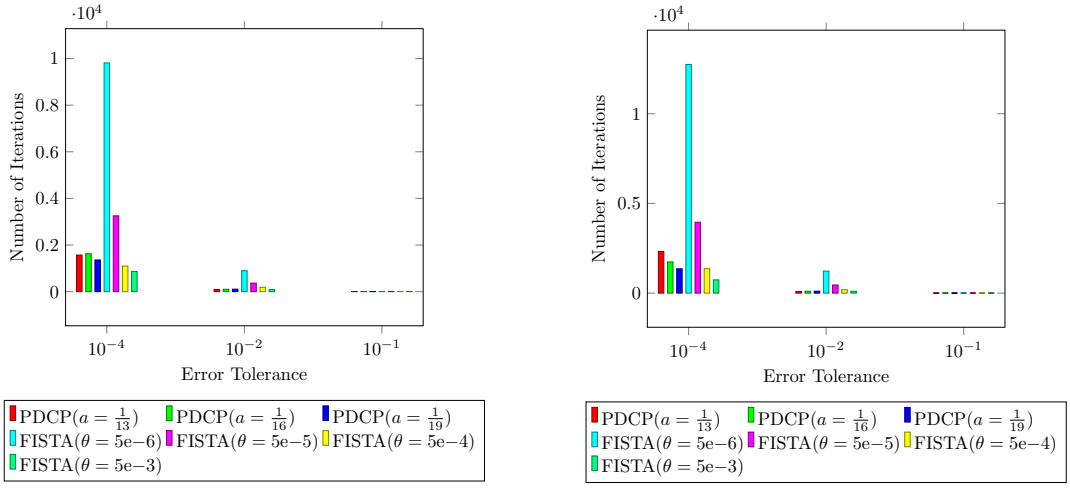
$$F_\epsilon(\mathbf{x}) = \max_{\mathbf{y}} \left\{ \langle \mathbf{K}_\epsilon \mathbf{x} - \boldsymbol{\ell}_\epsilon - \epsilon, \mathbf{y} \rangle - \delta_P(\mathbf{y}) \right\}. \tag{3.73}$$

Thus, we update $\mathbf{x}^n, \mathbf{y}^n, \overline{\mathbf{x}}^n$ as follows,

$$\begin{cases} \mathbf{y}^{n+1} &= [\mathbf{y}^n + \sigma(\mathbf{K}_\epsilon \overline{\mathbf{x}}^n - \boldsymbol{\ell}_\epsilon - \epsilon)]_P \\ \widetilde{\mathbf{x}} &= \mathbf{x}^n - \tau \mathbf{K}_\epsilon^* \mathbf{y}^{n+1} \\ \mathbf{x}^{n+1} &= \arg\min_{\mathbf{x}^{n+1}} \left\{ \frac{1}{2} \|\mathbf{x}^{n+1} - \widetilde{\mathbf{x}}\|_2^2 + \tau\lambda G(\mathbf{x}^{n+1}) \right\} \end{cases}. \tag{3.74}$$

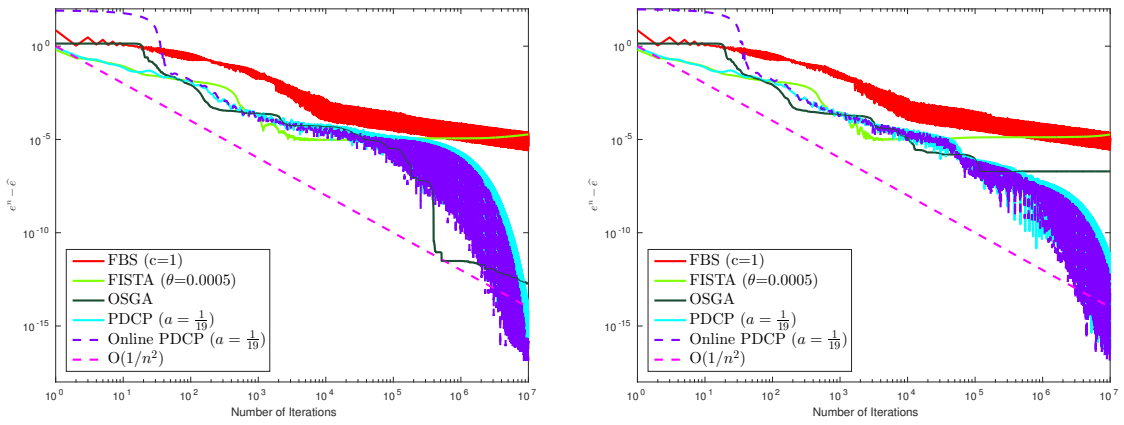
We can calculate the proximal mapping of the infinity norm by the proximal mapping of the convex conjugate of the infinity norm [53]. But here we suggest a direct method to calculate it. The proposed method is more efficient in this experiment. Refer to Section 2.7 for the calculation of $\mathbf{x}^{n+1}$.

Comparison of Multi-Task Learning. Now we compare the two loss functions: absolute loss and ϵ-Insensitive loss combined with $\mathcal{L}_{1,\infty}$ norm as a regularizer for the case $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$. In all the cases, the optimal solution $\widehat{e}$ is chosen by running Online PDCP. For the absolute loss, the running time per iteration of PDCP is



(a) **(b)**

Figure 3.16: Performance evaluation of the PDCP with different values of a and the FISTA with different values of θ when $\lambda = 10^{-3}$ (a) and $\lambda = 10^{-5}$ (b) in the application of multi-task learning with the ϵ-insensitive loss. The figures show the needed number of iterations (y axis) to drop the error below a certain error tolerance (x axis) within a maximum number of 1e6 iterations.



(a) **(b)**

Figure 3.17: Log-log plot of the error $e^n - \widehat{e}$ to the number of iterations n in the experiment of multi-task learning with the ϵ-insensitive loss. (a) shows the results for $\lambda = 10^{-3}$ and (b) shows the results for $\lambda = 10^{-5}$.

0.33×10^{-3} seconds; the running time per iteration of FISTA is 0.68×10^{-3}; and the running time per iteration of OSGA is 0.36×10^{-3}. For the insensitive loss, the running time per iteration of PDCP is 0.62×10^{-3} seconds; the running time per iteration of FISTA is 0.80×10^{-3}; and the running time per iteration of OSGA is 0.34×10^{-3}. Next,

we analyse the results for absolute loss and insensitive loss separately.

First, let us consider multi-task learning with absolute loss. From Figure 3.14 we can observe that it is less time consuming to decrease the error below 10^{-1}. However it takes more time to reach an error tolerance of 10^{-2} especially for FISTA using $\theta = 5 \times 10^{-6}$. The overall comparison are shown in Figure 3.15.

Next we consider multi-task learning with ϵ-insensitive loss. From Figure 3.16 we see that it needs more time to reach an error tolerance of 10^{-4} than 10^{-2} especially for FISTA using $\theta = 5 \times 10^{-6}$. However FISTA using $\theta = 5 \times 10^{-3}$ is the most efficient to reach the error below 10^{-4}. Figure 3.17 depicts the comparison for much smaller error tolerance.

3.2.6 Matrix Completion and Matrix Factorization

In this section, we conduct two unsupervised ML tasks: matrix completion and matrix factorization. The goal of matrix completion is to recover a full matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ from partially observed matrix $\mathbf{S}$. The energy is composed of a loss function measuring the difference between $\mathbf{X}$ and $\mathbf{S}$ on the observed entries and a trace norm regularizer on $\mathbf{X}$, assuming that $\mathbf{X}$ is low rank. Matrix factorization is also known as matrix decomposition. Seeking a low-dimensional factorization is equal to seeking a low-rank matrix [149]. So we use the trace norm regularizer for matrix completion and matrix factorization, and use the absolute loss for matrix completion and the hinge loss for Max-Margin matrix factorization. We evaluate on the 100K MovieLens data set[9] which contains one hundred thousand samples from 943 users on 1682 movies.

The absolute loss is defined as

$$\mathcal{F}(\mathbf{X}) = \sum_{(i,j)} |P_\Omega\left(X_{i,j} - S_{i,j}\right)| \quad \text{where} \quad P_\Omega\left(X_{i,j}\right) = \begin{cases} X_{i,j} & (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{3.75}$$

and Ω is the set of indexes of observed index pairs.

For max-margin matrix factorization, suppose there are n_r distinct ratings. Then we introduce $\mathrm{n}_r - 1$ thresholds θ_r $(r = 1, \cdots, \mathrm{n}_r - 1)$ to measure the hinge loss between the predicted value $X_{i,j}$ and the ground truth $S_{i,j}$ [138]. This leads the hinge loss to

$$\mathcal{F}(\mathbf{X}) = \sum_{(i,j)} \sum_{r=1}^{\mathrm{n}_r - 1} \max\left(0, 1 - P_\Omega(T^r_{i,j}\left(\theta_r - X_{i,j}\right))\right) \text{ where } T^r_{i,j} = \begin{cases} 1 & r \geq S_{i,j} \\ -1 & r < S_{i,j} \end{cases}. \tag{3.76}$$

In addition, we define a function

$$\mathcal{F}_r(\mathbf{X}) = \sum_{(i,j)} \max\left\{0, 1 - P_\Omega(T^r_{i,j}\left(\theta_r - X_{i,j}\right))\right\} \tag{3.77}$$

[9]The data set is available at http://www.grouplens.org/node/12.

such that

$$\mathcal{F}(\mathbf{X}) = \sum_{r=1}^{\mathrm{n}_r - 1} \mathcal{F}_r(\mathbf{X}).$$

We define the trace norm regularizer as,

$$\mathcal{G}(\mathbf{X}) = \mathrm{trace}(\sqrt{\mathbf{X}^*\mathbf{X}}) = \sum_{i=1}^{\min(m,n)} \sigma_i \tag{3.78}$$

where σ_i denote the singular values of $\mathbf{X}$. To calculate the subgradient of $\mathcal{G}(\mathbf{X})$, we leverage the following method [8, 163].

$$\partial\mathcal{G}(\mathbf{X}) = \mathbf{U}_r {\mathbf{V}_r}^* \tag{3.79}$$

where singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$, $\mathbf{\Sigma}$ is a diagonal matrix whose diagonal elements are $\{\sigma_\mathrm{i}\}$, r is the rank of $\mathbf{X}$ and $\mathbf{A}_r$ means the restriction of $\mathbf{A}$ to the first r columns. We also need the following lemma presented in [28] for further the upcoming derivation

$$\mathcal{D}_\lambda(\mathbf{X}) = \arg\min_{\mathbf{W}} \left\{ \frac{1}{2}\|\mathbf{W} - \mathbf{S}\|_{\mathcal{F}}^2 + \lambda\mathcal{G}(\mathbf{W}) \right\} = \mathbf{U}\mathbf{D}_\lambda(\mathbf{\Sigma})\mathbf{V}^*, \tag{3.80}$$

where $\mathbf{D}_\lambda(\mathbf{\Sigma}) = \mathrm{diag}\left([\sigma_i - \lambda]_+\right)$, $[x]_+ = \max(x, 0)$ and $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$.

FBS for Matrix Completion and Matrix Factorization. We define the subgradient of the absolute loss evaluated at $X_{i,j}$ as below:

$$\nabla\mathcal{F}(X_{i,j}) = \begin{cases} 1 & (i,j) \in \Omega \text{ and } X_{i,j} - S_{i,j} > 0 \\ -1 & (i,j) \in \Omega \text{ and } X_{i,j} - S_{i,j} < 0 \\ 0 & \text{otherwise} \end{cases}. \tag{3.81}$$

The subgradient of the hinge loss at the point $X_{i,j}$ is defined as

$$(\nabla\mathcal{F}(\mathbf{X}))_{i,j} = \sum_{r=1}^{\mathrm{n}_r - 1} (\nabla\mathcal{F}_r(\mathbf{X}))_{i,j} \tag{3.82}$$

where

$$(\nabla\mathcal{F}_r(\mathbf{X}))_{i,j} = \begin{cases} T_{i,j}^r & (i,j) \in \Omega \text{ and } 1 - T_{i,j}^r(\theta_r - X_{i,j}) > 0 \\ 0 & \text{otherwise} \end{cases}.$$

FISTA for Matrix Completion and Matrix Factorization. In this section, we show the equations of the smoothed loss with the corresponding gradient evaluated at the

component $X_{i,j}$. The smoothed absolute loss and its gradient are

$$(\mathcal{F}(\mathbf{X}))_{i,j} = \begin{cases} \frac{(X_{i,j}-S_{i,j})^2}{2\epsilon} + \frac{\epsilon}{2} & |X_{i,j} - S_{i,j}| \leq \epsilon \\ |X_{i,j} - S_{i,j}| & \text{otherwise} \end{cases} \tag{3.83}$$

and

$$(\nabla\mathcal{F}(\mathbf{X}))_{i,j} = \begin{cases} P_\Omega(\frac{X_{i,j}-S_{i,j}}{\epsilon}) & |X_{i,j} - S_{i,j}| \leq \epsilon \\ P_\Omega(1) & X_{i,j} - S_{i,j} > \epsilon \\ P_\Omega(-1) & X_{i,j} - S_{i,j} < -\epsilon. \end{cases} \tag{3.84}$$

We provide the equations of the smoothed hinge loss,

$$(\mathcal{F}_r(\mathbf{X}))_{i,j} = \begin{cases} 0 & \chi < -\epsilon \\ \frac{(\chi+\epsilon)^2}{4\epsilon} & |\chi| \leq \epsilon \\ \chi & \chi > \epsilon \end{cases} \tag{3.85}$$

where $\chi = 1 - T^r_{i,j}(\theta_r - X_{i,j})$.

And its gradient at $X_{i,j}$ is as follows

$$(\nabla\mathcal{F}_r(\mathbf{X}))_{i,j} = \begin{cases} 0 & \chi < -\epsilon \\ P_\Omega(\frac{(\chi+\epsilon)T^r_{i,j}}{2\epsilon}) & |\chi| \leq \epsilon \;\; \chi = 1 - T^r_{i,j}(\theta_r - X_{i,j}). \\ P_\Omega(T^r_{i,j}) & \chi > \epsilon \end{cases} \tag{3.86}$$

PDCP for Matrix Completion. First we will illustrate the equations involved with matrix completion. Formulate $\mathbf{X}$ and $\mathbf{S}$ to the vectors $\mathbf{x}, \mathbf{s} \in \mathbb{R}^{m\times n}$ respectively in column-major vector representation. We define the dual variable $\mathbf{y} \in \mathbb{R}^{m\times n}$ and the set $P = \{\mathbf{y} \in \mathbb{R}^{m\times n} : \forall y_i \in [-1, 1]\}$ where y_i is the i^{th} component of $\mathbf{y}$. By introducing the dual variable, the loss function becomes

$$\mathcal{F}(\mathbf{x}) = \|\mathbf{K}(\mathbf{x} - \mathbf{s})\|_1 = \max_{\mathbf{y}} \{\langle \mathbf{Kx} - \mathbf{Ks}, \mathbf{y}\rangle - \delta_P(\mathbf{y})\} \tag{3.87}$$

where $\mathbf{K} \in \mathbb{R}^{(m\times n)\times(m\times n)}$ is a diagonal matrix

$$\mathbf{K}_{m(j-1)+i,m(j-1)+i} = \begin{cases} 1 & (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}, 1 \leq i \leq m, 1 \leq j \leq n.$$

The convex conjugate function is

$$\mathcal{F}^*(\mathbf{y}) = \langle \mathbf{Ks}, \mathbf{y}\rangle + \delta_P(\mathbf{y}). \tag{3.88}$$

We can conclude the iteration equations of PD CD for matrix completion as follows:

$$\begin{cases} \mathbf{y}^n &= \left[\mathbf{y}^{n-1} + \sigma(\mathbf{K}(\overline{\mathbf{x}}^{n-1} - \mathbf{s}))\right]_P \\ \widetilde{\mathbf{x}}^n &= \mathbf{x}^{n-1} - \tau \mathbf{K}^* \mathbf{y}^n \\ \widetilde{\mathbf{x}}^n &= \widetilde{\mathbf{X}}^n(:) \\ \widetilde{\mathbf{X}}^n &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \\ \mathbf{D}_{\tau\lambda}\left(\mathbf{\Sigma}\right) &= \operatorname{diag}\left(\left[\sigma_i - \tau\lambda\right]_+\right) \\ \mathbf{x}^n &= \mathbf{U}\mathbf{D}_{\tau\lambda}\left(\mathbf{\Sigma}\right)\mathbf{V}^* \end{cases} . \tag{3.89}$$

PDCP for Matrix Factorization. Next we will illustrate the equations involved with matrix factorization. Formulate $\mathbf{X}$ and $\mathbf{T}^r$ to $\mathbf{x}, \mathbf{t}^r \in \mathbb{R}^{m\times n}$ in column-major vector representation. We define $\mathbf{y}^r \in \mathbb{R}^{m\times n}, r = 1, \cdots, \mathrm{n}_r - 1$ and the set $H = \{\mathbf{y}^r \in \mathbb{R}^{m\times n} : \forall y_i^r \in [0,1]\}$ where y_i^r is the i^{th} component of $\mathbf{y}^r$. Let the dual variable be

$$\mathbf{y} = \left[\mathbf{y}^1, \quad \cdots, \quad \mathbf{y}^{\mathrm{n}_r - 1}\right]^\top .$$

We define the set $P = \left\{\mathbf{y} \in \mathbb{R}^{(\mathrm{n}_r-1)\times(m\times n)} : \forall \mathbf{y}^r \in H\right\}$. Then, the loss function becomes

$$\mathcal{F}(\mathbf{x}) = \sum_{r=1}^{\mathrm{n}_r - 1} \max_{\mathbf{y}^r} \left\{\langle \mathbf{K}^r\mathbf{x}, \mathbf{y}^r\rangle - \langle \mathbf{Q}(\theta_r t^r - 1), \mathbf{y}^r\rangle\right\} - \delta_H(\mathbf{y}) \tag{3.90}$$

where $\mathbf{K}^r$ and $\mathbf{Q} \in \mathbb{R}^{(m\times n)\times(m\times n)}$ are diagonal matrices

$$K^r_{m(j-1)+i,m(j-1)+i} = \begin{cases} T_{i,j} & (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

and

$$\mathbf{Q}_{m(j-1)+i,m(j-1)+i} = \begin{cases} 1 & (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases} .$$

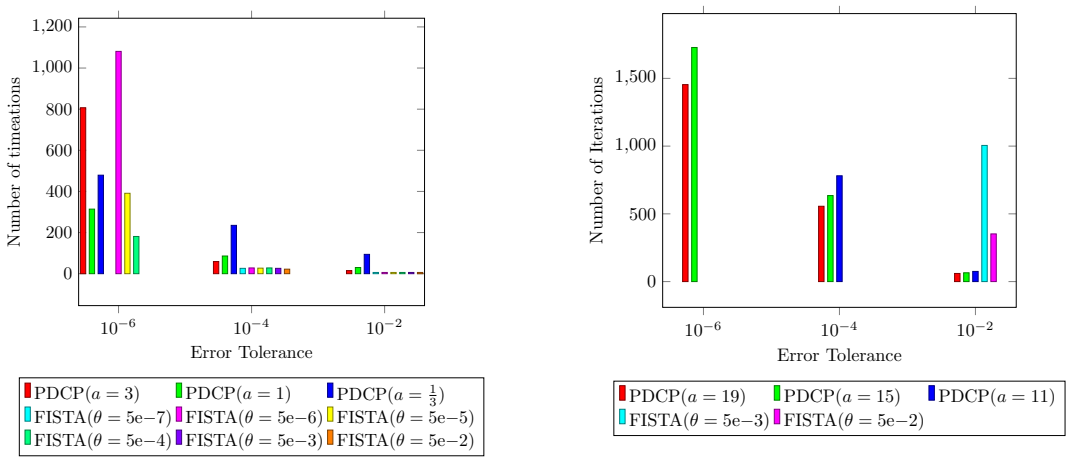
After simplification, we obtain, $\mathcal{F}(\mathbf{x}) = \max_{\mathbf{y}} \left\{\langle \mathbf{K}\mathbf{x}, \mathbf{y}\rangle - \langle \mathbf{q}, \mathbf{y}\rangle - \delta_P(\mathbf{y})\right\}$ where $\mathbf{K} = \left[\mathbf{K}^1 \ \cdots \ \mathbf{K}^{\mathrm{n}_r - 1}\right]^\top$ and

$$\mathbf{q} = \left[\mathbf{Q}(\theta_1\mathbf{t}^1 - 1), \quad \cdots, \quad \mathbf{Q}(\theta_{\mathrm{n}_r-1}\mathbf{t}^{\mathrm{n}_r-1} - 1)\right]^\top .$$

Therefore

$$\mathcal{F}^*(\mathbf{y}) = \langle \mathbf{q}, \mathbf{y}\rangle + \delta_P(\mathbf{y}).$$

The iteration equations of PDCP algorithm for Max-Margin matrix factorization can



(a) **(b)**

Figure 3.18: Performance evaluation of the PDCP with different values of a and the FISTA with different values of θ when $\lambda = 10^{-3}$ (a) and $\lambda = 10^{-5}$ (b) in the application of matrix completion. The figures show the needed number of iterations (y axis) to drop the error below a certain error tolerance (x axis) within a maximum number of 8000 iterations.

be written as,

$$\begin{cases} \mathbf{y}^n &= \left[\mathbf{y}^{n-1} + \sigma(\mathbf{K}\overline{\mathbf{x}}^{n-1} - \mathbf{q})\right]_P \\ \widetilde{\mathbf{x}}^n &= \mathbf{x}^{n-1} - \tau\mathbf{K}^\top\mathbf{y}^n \\ \widetilde{\mathbf{x}}^n &= \widetilde{\mathbf{X}}^n(:) \\ \widetilde{\mathbf{X}}^n &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \\ \mathbf{D}_{\tau\lambda}(\mathbf{\Sigma}) &= \operatorname{diag}\left(\left[\sigma_i - \tau\lambda\right]_+\right) \\ \mathbf{x}^{n+1} &= \mathbf{U}\mathbf{D}_{\tau\lambda}(\mathbf{\Sigma})V^* \end{cases}. \tag{3.91}$$

Comparison of Matrix Completion and Matrix Factorization. The optimal value $\widehat{e}$ is chosen by running Online PDCP 8000 times. For matrix completion, the running times per iteration of PDCP, FISTA and OSGA are 3.47, 10.42 and 5.80 seconds. For matrix factorization, the running times per iteration of PDCP, FISTA and OSGA are 4.69, 11.11 and 4.48 seconds. Next we will show the comparison of matrix completion.

We continue with the comparison on the task of matrix completion. From Figure 3.18 FISTA we can see that can failed to decrease to a lower tolerance. For example, on Figure 3.18a, the error of FISTA with $\theta = 5 \times 10^{-7}$ cannot decrease below 10^{-6} within 8000 iterations. On Figure 3.18b FISTA fails for the error tolerance 10^{-4} and 10^{-6}. As shown in Figure 3.19a, FISTA decreases fast at the first 100 iterations. But PDCP outperforms FISTA at the end with a convergence rate even better than $O(1/n^2)$.

Next we present the results for the task of matrix factorization. In Figure 3.20a, the error of FISTA with $\theta = 5 \times 10^{-2}$ cannot decrease below 10^{-5} within 8000 iterations. On

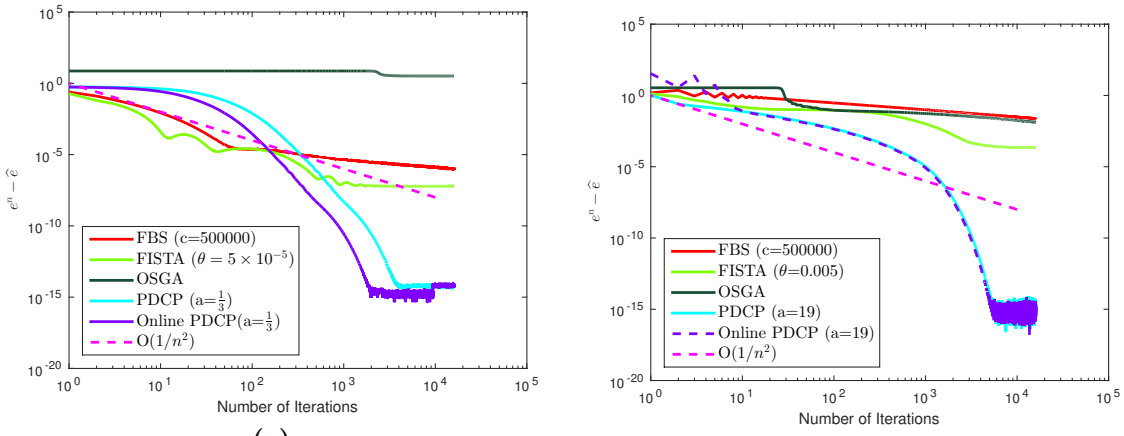


Figure 3.19: Log-log plot of the error $e^n - \widehat{e}$ to the number of iterations n in the experiment of matrix completion. (a) shows the results for $\lambda = 10^{-3}$ and (b) shows the results for $\lambda = 10^{-5}$.

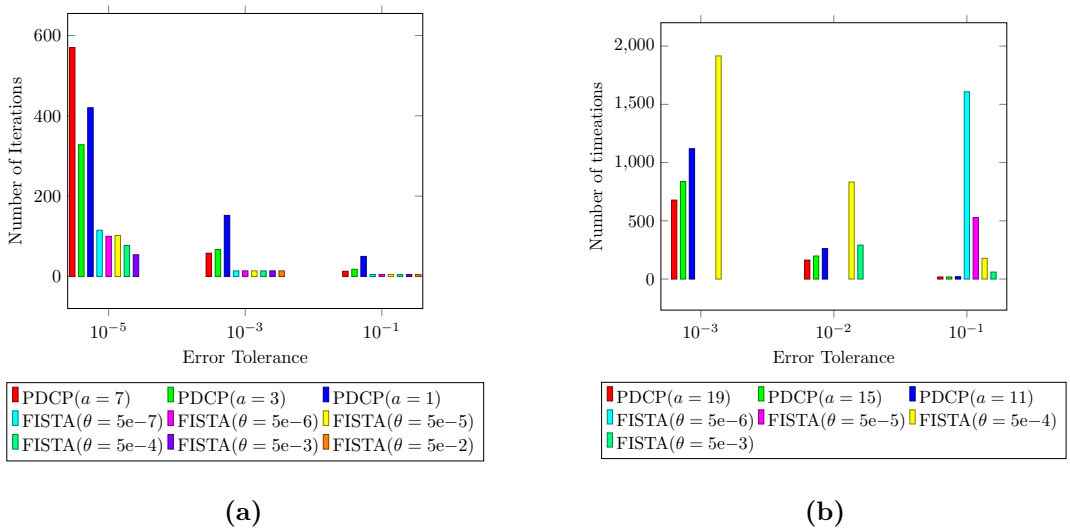
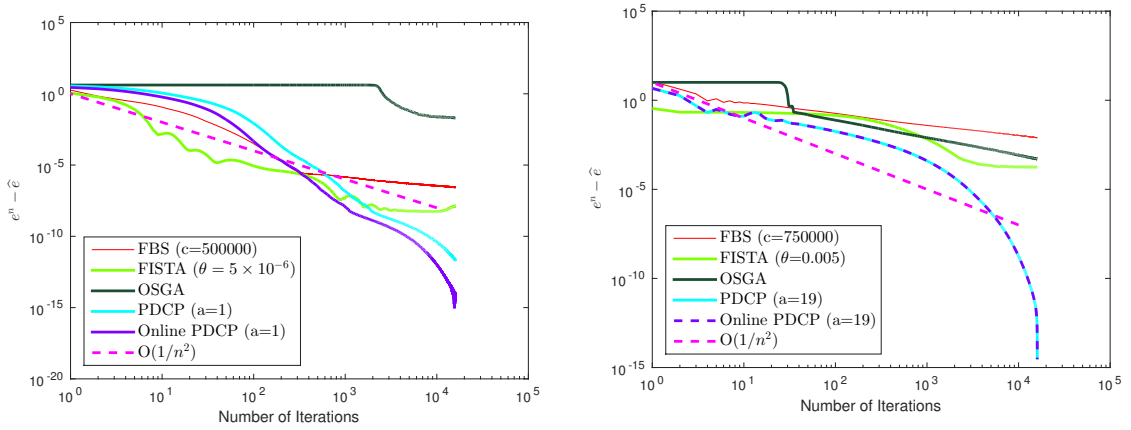


Figure 3.20: Performance evaluation of the PDCP with different values of a and the FISTA with different values of θ when $\lambda = 10^{-3}$ (a) and $\lambda = 10^{-5}$ (b) in the application of matrix factorization. The figures show the needed number of iterations (y axis) to drop the error below the error tolerance (x axis) within a maximum number of 8000 iterations.

Figure 3.20b, FISTA with $\theta = 5 \times 10^{-4}$ can decrease the error below 10^{-3}.



(a) **(b)**

Figure 3.21: Log-log plot of error the $e^n - \widehat{e}$ to the number of iterations n in the experiment of matrix factorization. (a) shows the results for $\lambda = 10^{-3}$ and (b) shows the results for $\lambda = 10^{-5}$.

As shown in Figure 3.21a, FISTA decreases fast at the first 1000 when $\lambda = 10^{-3}$ and at the first 100 iterations when $\lambda = 10^{-5}$, separately. But PDCP outperforms FISTA in the later iterations with a convergence rate even better than $O(1/n^2)$, which appears again in Figure 3.21b.

3.3 Conclusion

In this section, we described how to solve several benchmark problems of ML algorithms by using first-order optimization algorithms. To make the comparison as convincing as possible, if there is a tuning parameter involved, we chose the value of the tuning parameter giving best performance. We showed the influence of the ratio between the step sizes of the primal variable and the dual variable in PDCP. In addition, because FISTA cannot solve a non-smooth loss function, we used smoothing techniques before using FISTA. Thus, we studied the impact of the smoothness parameter θ on FISTA. Our experiments reveal that in a reasonable domain, FISTA with the bigger smoothness parameter converges fast at the beginning although it is hard to converge to the optimal solution. Thus we infer that a better strategy would be to descrease the value of the smoothness parameter with increasing number of iterations. However the selection of the parameter is always a tricky and difficult subject.

We draw log-log graphs to make an overall comparison for FBS, FISTA, OSGA, PDCP. During the different ranges of iterations, optimization algorithms have different performances. For example, sometimes FISTA is faster at the first few iterations but attains a lower empirical connvergency rate in the long run compared to e.g. PDCP. In the experiments for dimensionality reduction composed by a smooth loss function and a non-smooth regularizer, PDCP is faster to reach the optimal solution. Whereas in the

experiment of linear SVM composed by a non-smooth loss function and smooth regularizer, FISTA and OSGA do not converge within 5×10^5 iterations. However, PDCP converges with a convergence rate better than $O(1/n^2)$. When loss function and regularizer are both smooth as in kernel SVM, FISTA achieves its theoretical convergence rate of $O(1/n^2)$. Surprisingly, PDCP still reaches an empirical convergence rate better than $O(1/n^2)$. For the rest remaining experiments where loss functions and regularizers are both non-smooth, FBS shows a convergence rate $O(1/\sqrt{n})$. We may not get the optimal values by FISTA since we use the smoothing techniques. In the experiment of Multi-Task Learning, OSGA has a better performance than FBS and FISTA. Although theoretical convergence rate of PDCP is $O(1/n)$, PDCP has a much better practical convergence rate which is better than $O(1/n^2)$ and can converge in almost all ML problems. In all experiments, Online PDCP is better than or equal to PDCP. In summary, the primal dual algorithm [31] has the best performance with a fast empirical convergence rate in all problems concerned in this section and experiments show that PDCP is an efficient and robust solver for a ML problems.

4 Machine Learning Applications in Computer Vision

Contents

4.1 Single View Image Processing

2D single image processing is one historical and important topic in Computer Vision. Supervised machine learning techniques have been widely adapted to solve such problems and have made encouraging progress. In this section, we first present a brief overview of the trained Reaction-Diffusion Model (RDM) proposed in [43] which has been successfully applied several problems of digital image processing, for instance, Gaussian denoising, JPEG deblocking and image inpainting.

Trained Reaction-diffusion Model. Let $\Omega \subset \mathbb{R}^2$ be the image domain and consider an image as a mapping $\mathcal{U} : \Omega \times [\mathbf{0}, \infty) \to \mathbb{R}$, $\mathcal{U} = \mathcal{U}(\mathbf{x}, t)$, where $t \geqslant 0$ denotes the stage or the time. Then a general anisotropic diffusion model is given as

$$\partial_t \mathcal{U} = \operatorname{div}(\Gamma(\mathcal{U})\nabla\mathcal{U}) \,, \tag{4.1}$$

with the original image $\mathcal{F} : \Omega \to \mathbb{R}$ as the initial state, i.e. $\mathcal{U}(\mathbf{x}, 0) = \mathcal{F}(\mathbf{x})$, and with reflecting boundary conditions, i.e. $\partial_{\mathbf{n}}\mathcal{U} = \mathbf{0}$ on the image boundary $\partial\Omega$, where $\mathbf{n}$ denotes the normal to $\partial\Omega$. Note that in Equation (4.1) ∇ is taken w.r.t. the spatial variables $\mathbf{x}$ and $\Gamma(\mathcal{U})$ denotes the diffusion tensor, which is a positive definite symmetric matrix. In the case of isotropic diffusion $\Gamma(\mathcal{U})$ can be replaced by a positive scalar-valued diffusion coefficient also called diffusivity.

In [43], we modified Equation (4.1) in the following way: (i) we generalized the ∇

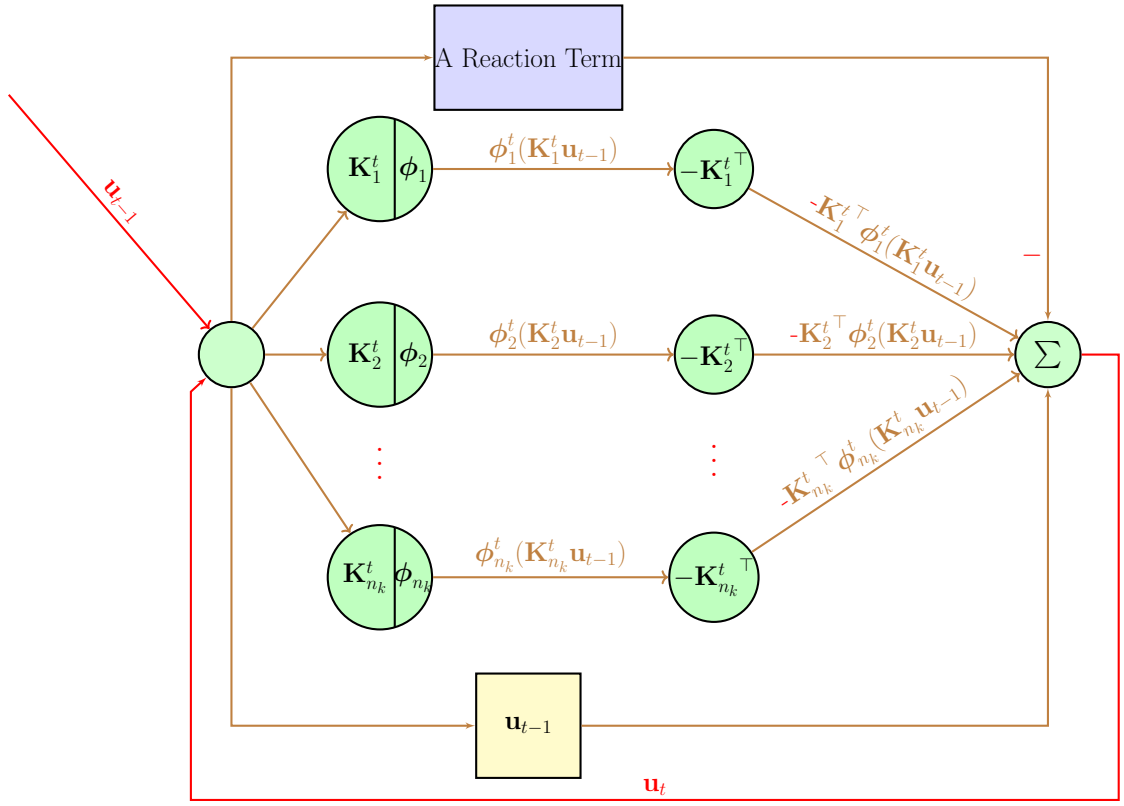


Figure 4.1: Illustration of the structure of the convolutional network presented by the proposed trained RDM.

operator to a set of filters represented by $\mathbf{K}_i^t$, (ii) we equipped each filter with its own diffusion coefficient represented by so-called influence functions $\boldsymbol{\phi}_i^t : \mathbb{R}^d \to \mathbb{R}^d$, and (iii) we added a reaction term $\psi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$, which basically yields a non homogeneous diffusion equation. Finally, we proposed the following diffusion model in the discrete setting

$$\mathbf{u}_t = \mathbf{u}_{t-1} - \sum_{i=1}^{n_k} \mathbf{K}_i^{t\top} \boldsymbol{\phi}_i^t(\mathbf{K}_i^t \mathbf{u}_{t-1}) - \psi(\mathbf{u}_{t-1}, \mathcal{F}) \,, \tag{4.2}$$

where $\mathbf{u}_t \in \mathbb{R}^d$ denotes the vectorized representation of an image at stage t, n_k denotes the number of filters, $\mathcal{F} \in \mathbb{R}^d$ is the vectorized corrupted input image, and $\mathbf{K}_i^t \in \mathbb{R}^{d \times d}$ is a sparse matrix representing a convolution operation with the kernel $\mathbf{k}_i^t$ (i.e. $\mathbf{K}_i^t \mathbf{u} = \mathbf{k}_i^t * \mathbf{u}$). The reaction term changes for different applications, like Gaussian denoising, or JPEG deblocking. Note that the influence functions $\boldsymbol{\phi}_i^t(\cdot)$ and filters $\mathbf{K}_i^t$ vary for different stages t. Equation (4.52) is one gradient descent step of Fields of Experts (FoE) [142] using trained parameters. In the training, the energy is formulated by the similarity of the ground truth and the output $\mathbf{u}_t$. Thus the output of each gradient descent step should always be close

to the unique ground truth. The principle of measuring the similarity between the ground truth and the output $\mathbf{u}_t$ includes Peak Signal to Noise Ratio (PSNR), Structural Similarity Image Measure (SSIM). The proposed trained RDM is one specific type of convolutional networks which are popular models in the field of computer vision and pattern recognition. Figure 4.1 shows in the convolutional network, there are two convolutional layers which are associated with constrains. One constrain is that the corresponding kernels in the first and the second convolutional layers are opposite. The second constrain is that the activation functions of the second convolutional layer are set to identity functions.

In the next section we present a trained diffusion model for image inpainting based on the the proposed trained RDM [43]. The principle of the measurement we adopted is the SSIM. The proposed diffusion model uses several parametrized linear filters and influence functions. Those parameters are learned in a loss based approach, where we first perform a greedy training before conducting a joint training to further improve the performance.

4.1.1 Diffusion Models for Image Inpainting

Image inpainting is a fundamental problem in Computer Vision (CV) with great practical importance. Given an image with lost, deteriorated or simply unknown regions, the task of image inpainting is to convincingly fill-up those unknown image regions. Hence, the main goal is to produce natural-looking and visually pleasant images. In this section we will modify the diffusion model proposed in [43] for the task of image inpainting. Contrary to [43], where a PSNR based training is used, we propose a training based on the SSIM [158], which seems to be more suitable for the task of image inpainting. In the experimental section we will present generic models, that are trained based on a complete dataset, and specific models, that are trained on the uncorrupted image regions of the image to be inpainted. We will provide a detailed evaluation on the TUM-image inpainting database [155, url], which will show that the proposed method is able to achieve superior performance compared to state-of-the-art inpainting methods, and is highly efficient at the same time. Further we will show that although our method falls into the category of Partial Differential Equation (PDE) based methods it is also capable to perform texture inpainting to a certain extend.

4.1.1.1 Related Work

Inpainting applications can be roughly divided into two classes. First, there are inpainting applications where one considers the task of reconstructing a set of small connected image regions. Example applications include the restoration of old photos, damaged videos [93], or artwork [123], and the removal of logos, superimposed text, or other unwanted objects in images, to name but a few. The second class of inpainting applications considers the task of restoring large regions of scattered pixels. Inpainting methods belonging to this class are closely related to image compression methods [41, 64, 104]. Galić et al. [64] for instance proposed a method for image compression based on edge-enhancing diffusion. Liu

et al. [104] proposed a framework for image compression, where the main idea is to extract edge-based assistant information in the encoding step and use this information to guide the inpainting in the decoding step.

Proposed inpainting methods in the literature tend to fall into one of the following categories. First, there are exemplar-based or patch-based methods [30, 47, 170], that try to fill the inpainting regions by propagating information from the remaining parts of the image at the patch-level. Thus those methods reconstruct the corrupted regions by sampling and copying uncorrupted patches taken from the same image (or from a certain dictionary). Those methods are usually based on the self-similarity principle, i.e. one assumes that images include a lot of repetitions of local information. It is not surprising, that the main idea of patch-based methods can be traced back to texture synthesis techniques [55]. Due to their non-local property, patch-based methods are well suited to inpaint large connected image regions with texture, but they are unable to handle densely scattered inpainting domains. The second category of inpainting methods involves either variational principles or is PDE based, noticeable examples include [17, 32, 33, 57, 113]. Those methods have been successfully used to smoothly inpaint small image regions. A common drawback of PDE based methods is their inability to properly reconstruct image texture, which is clearly visible when trying to reconstruct large inpainting domains. However, PDE based methods remain applicable in compression-like applications, i.e. inpainting task based on sparse data. Masnou and Morel [113] for example proposed an inpainting model based on variational principles, where they interpolated the inpainting regions by extending the isophotes, which are lines of constant intensities. Bertalmio et al. [17] proposed a PDE based inpainting method, that is inspired by the methodology of art conservators, where a transport process is interlinked with an anisotropic diffusion process. The overall idea is to fill the inpainting region such that the isophote lines are completed. In [32] authors extended the Rudin Osher Fatemi (ROF) image denoising model [143] to image inpainting, i.e. their variational model is based on the Total Variation (TV) and produces inpainting result with smallest possible isophotes. In a subsequent paper [33] a curvature driven diffusion equation was introduced to realize the connectivity principle. Esedoglu and Shen [57] proposed the so-called Mumford Shah Euler model. This model combines the celebrated Mumford Shah segmentation model [117] with Euler's elastica curve model. Besides the aforementioned two categories of inpainting methods, authors also proposed methods that attempt to combine the advantages of PDE based and patch-based methods [18, 25, 47, 70, 94]. Bugeau et al. [25] for instance proposed a framework that combines patch-based methods with PDE based methods by enforce coherence among neighboring pixels. In [100, 140, 141] authors investigated image statistics from natural images for the task of image inpainting. Roth and Black [140] for example proposed a framework to learn generic, expressive image priors that capture the statistics of natural scenes. The so-called FoE [142] uses continuous heavy-tailed potential functions and learns the parameters of experts by contrastive divergence learning in high-order Markov Random Field (MRF) models. In [145], Schmidt et al. modified the FoE model by using Gaussian

scale mixtures as potential functions. In [43], Chen et al. simplified the diffusion coefficients and then generalized the conventional nonlinear RDM by using learned filters and influence functions. They showed that the resulting energy functional of the proposed diffusion model is a generalization of the FoE model [142]. They trained models for Gaussian denoising and JPEG deblocking. Both models achieved a superior performance compared to the state-of-the-art and are highly efficient as well.

4.1.1.2 Methodology

Diffusion Model for Image Inpainting. For the inpainting task tackled in this thesis, the given gray-valued images $\mathbf{u}$ are assumed to be noisefree. Let $\mathbf{u}_t : \mathbb{R}^n \to \mathbb{R}^n$ be a family of vectorized gray-valued images $\mathbf{I}_t \in \mathbb{R}^{h\times w}$, where $n = hw$ and t denotes the stage or time. Hence the proposed method only updates the gray-values inside the unknown or missing image regions $\mathcal{A} \subset \Omega$. Unlike Gaussian denoising or JPEG deblocking discussed in [43], a challenge of image inpainting is that it has no available reaction term within the inpainting domain. Hence the proposed model relies on the information at the boundary of the inpainting domain $\partial\mathcal{A}$ and on the learned information from uncorrupted parts of the image (i.e. $\mathcal{A}^c$), or from a given set of images. Thus the main idea of the proposed model is to propagate the known information from outside the inpainting region in a meaningful way in order to reconstruct the values inside the unknown image regions. For this purpose we define the following diffusion model

$$\mathbf{u}_t = \mathbf{u}_{t-1} - \mathbf{m} \cdot \sum_{i=1}^{n_k} \mathbf{K}_i^{t\top} \boldsymbol{\phi}_i^t(\mathbf{K}_i^t \mathbf{u}_{t-1}) \,, \tag{4.3}$$

where $\cdot$ denotes the Hadamard product (i.e. pointwise multiplication), and $\mathbf{m} \in \mathbb{R}^n$ is a vectorized mask indicating the inpainting domain $\mathcal{A}$, i.e.

$$m_j = \begin{cases} 1 & j \in \mathcal{A} \,, \\ 0 & \text{otherwise} \,, \end{cases} \tag{4.4}$$

where m_j denotes the value at the j^{th} position of the vector $\mathbf{m}$. The influence functions $\boldsymbol{\phi}_i^t : \mathbb{R}^n \to \mathbb{R}^n$ are formulated as linear combinations of triangular-shaped basis functions,

$$\phi_i^t(\mathbf{v})_j = \xi_i^t(v_j), \quad j \in \{1, \ldots, n\} \tag{4.5}$$

where $\xi_i^t : \mathbb{R} \to \mathbb{R}$ is a piecewise linear function using the efficient technique of slice transform [79]. The other option is Gaussian basis functions which can provide a smooth solution. Although it is sensible to the initialization when triangular-shaped basis functions use slice transform, it can provide competitive results with fast calculation in the cases of image inpainting and image denoising. The output value of the influence function ϕ_i^t at the j^{th} position only related to the j^{th} component of the input variable.

The main reason for using those simple basis functions is to reduce the computational complexity.

The iterative process in Equation (4.3) can be summarized as follows: (i) filter the current image $\mathbf{u}_{t-1}$ with the kernel $\mathbf{k}_i^t$ for $1 \leqslant i \leqslant n_k$, (ii) calculate the output of the influence function $\boldsymbol{\phi}_i^t(\cdot)$ and filter the result with the kernel $\overline{\mathbf{k}}_i^t$ [1] for $1 \leqslant i \leqslant n_k$, where $\overline{\mathbf{k}}_i^t$ is obtained by rotating $\mathbf{k}_i^t$ by 180 degrees, (iii) sum up the n_k results and calculate the Hadamard product with the mask $\mathbf{m}$, and (iv) update $\mathbf{u}_{t-1}$ with the obtained result.

Equation (4.3) can also be obtained by considering the gradient of the following energy functional

$$\mathcal{E}(\mathbf{u}) = \sum_{i=1}^{n_k} \boldsymbol{\rho}_i^t(\mathbf{K}_i^t\mathbf{u}) + \lambda\,\mathcal{D}\left(\mathbf{m}^c \cdot \mathbf{u}, \mathbf{m}^c \cdot \mathbf{f}\right) , \tag{4.6}$$

where $\mathbf{m}^c = 1 - \mathbf{m}$, λ is a positive weighting factor, and $\boldsymbol{\rho}_i^t : \mathbb{R}^n \to \mathbb{R}$ are functions for which we assume that $\frac{\partial \boldsymbol{\rho}_i^t(\mathbf{v})}{\partial \mathbf{v}} = \boldsymbol{\phi}_i^t(\mathbf{v})$. The data term $\mathcal{D}(\cdot,\cdot)$ in Equation (4.6) enforces that $\mathbf{u}$ is close to the observed data $\mathbf{f}$ outside the inpainting domain. A common data term based on the ℓ_2-norm is $\mathcal{D}(\mathbf{v_1}, \mathbf{v_2}) = \|\mathbf{v_1} - \mathbf{v_2}\|_2^2$. If we let $\lambda \to +\infty$ in Equation (4.6), then the values outside the inpainting domain will not change (i.e. $u_j = f_j$ if $m_j = 0$). Thus in this case Equation (4.6) can be rewritten as

$$\mathcal{E}(\mathbf{u}) = \sum_{i=1}^{n_k} \boldsymbol{\rho}_i^t\left(\mathbf{K}_i^t\left(\mathbf{m} \cdot \mathbf{u} + \mathbf{m}^c \cdot \mathbf{f}\right)\right) . \tag{4.7}$$

By calculating the gradient of $\mathcal{E}(\mathbf{u})$ w.r.t. $\mathbf{u}$ one obtains the subgradient in Equation (4.3), which shows that (4.3) can be interpreted as a simple gradient descent step.

Dataset. We will consider two types of trained diffusion models. First, we present generic models, that are learned on an entire dataset. The generic models are used to restore large regions of scattered pixels (e.g. 80% and 90% random missing pixels), and to inpaint small connected image regions. The second type of models are specifically learned for inpainting a certain image, i.e. those models are trained based on the uncorrupted parts of the given image. We will show that those specific models are able to learn the texture of a given image. Thus they are applicable for the task of texture inpainting.

For the generic models, the training dataset includes 400 images of size 180×180 as in [43]. The test dataset is the TUM-image inpainting database [url], which includes 17 images of size 640×480. For the specific models, we use 5 training images with different training masks. The experiments show that with small training datasets, the trained diffusion models can still provide good results. We assume the reason is that the training inpainting domain and the testing inpainting domain share many similarities which are learned in the training.

[1] ${\mathbf{K}_i^t}^\top \boldsymbol{\phi}_i^t(\mathbf{K}_i^t\mathbf{u}_{t-1})$ can be rewritten as $\overline{\mathbf{k}}_i^t * \phi_i^t(\mathbf{K}_i^t\mathbf{u}_{t-1})$.

Network Training. In this section we outline the learning approach. We propose a SSIM based learning approach to estimate the parameters $\Theta_t = \left\{\mathbf{k}_i^t, \phi_i^t\right\}$ on the right-hand side of Equation (4.3) for all stages $1 \leqslant t \leqslant T$. We start with a greedy training, where we optimize the parameters at the stage t and then use the optimal parameters to calculate the inpainted image $\mathbf{u}_t$. After that we optimize the parameters of the next stage, till we reach the maximum number of stages T. The result of the greedy training is used as an initialization for the following joint training, where we train all stages simultaneously. Let $\{\mathbf{f}^j, \mathbf{g}^j, \mathbf{m}^j\}_{j=1}^{\mathrm{n_{trn}}}$ denote the $\mathrm{n_{trn}}$ training samples, where $\mathbf{f}^j$ is the j^{th} corrupted image with unknown values at the inpainting regions indicated by the mask $\mathbf{m}^j$, and $\mathbf{g}^j$ is the j^{th} ground truth image. Then the greedy training minimizes

$$\mathcal{L}(\Theta_t) = \sum_{j=1}^{\mathrm{n_{trn}}} \ell(\mathbf{u}_t^j, \mathbf{g}^j)\,, \tag{4.8}$$

for each stage $1 \leqslant t \leqslant T$. The loss function $\ell(\mathbf{u}_t^j, \mathbf{g}^j)$ is defined based on the SSIM [158], which has already been used for image denoising [95, 137], where it has shown its superiority over PSNR [158]. SSIM assesses the luminance, the contrast, and the structure of two images $\mathbf{I}_1 : \Omega \to \mathbb{R}^2$ and $\mathbf{I}_2 : \Omega \to \mathbb{R}^2$ on the patch level. In this context the SSIM index of two image patches $\mathbf{p}_1$ and $\mathbf{p}_2$ is calculated as follows

$$\mathrm{SSIM}(\mathbf{p}_1, \mathbf{p}_2) = \frac{(2\mu_1\mu_2 + c_1)(2\sigma_{12} + c_2)}{(\mu_1^2 + \mu_2^2 + c_1)(\sigma_1^2 + \sigma_2^2 + c_2)}\,, \tag{4.9}$$

where c_j is a predefined constant, μ_j and σ_j^2 denote the average and the variance of the image patch $\mathbf{p}_j$, $j \in \{1, 2\}$, and σ_{12} is the covariance of the two patches. The final image measure, denoted as $\mathcal{S}(\mathbf{I}_1, \mathbf{I}_2)$, is obtained as the mean SSIM index of all image patches. Note that $\mathcal{S}(\mathbf{I}_1, \mathbf{I}_2)$ attains its maximum of 1 only if $\mathbf{I}_1 = \mathbf{I}_2$. We use the default parameter settings for calculating SSIM index [158]. Thus the SSIM index can be written as

$$\mathrm{SSIM}(\mathbf{u}_t^j, \mathbf{g}^j) = \frac{\mathcal{S}_1(\mathbf{u}_t^j, \mathbf{g}^j) \cdot \mathcal{S}_2(\mathbf{u}_t^j, \mathbf{g}^j)}{\mathcal{S}_3(\mathbf{u}_t^j, \mathbf{g}^j) \cdot \mathcal{S}_4(\mathbf{u}_t^j, \mathbf{g}^j)}, \tag{4.10}$$

where

$$\begin{aligned}
\mathcal{S}_1(\mathbf{u}_t^j, \mathbf{g}^j) &= 2(\mathbf{G}\mathbf{g}^j) \cdot (\mathbf{G}\mathbf{u}_t^j) + c_1,\\
\mathcal{S}_2(\mathbf{u}_t^j, \mathbf{g}^j) &= 2\mathbf{G}(\mathbf{g}^j \cdot \mathbf{u}_t^j) - (\mathbf{G}\mathbf{g}^j) \cdot (\mathbf{G}\mathbf{u}_t^j) + c_2,\\
\mathcal{S}_3(\mathbf{u}_t^j, \mathbf{g}^j) &= (\mathbf{G}\mathbf{g}^j) \cdot (\mathbf{G}\mathbf{g}^j) + (\mathbf{G}\mathbf{u}_t^j) \cdot (\mathbf{G}\mathbf{u}_t^j) + c_1,\\
\mathcal{S}_4(\mathbf{u}_t^j, \mathbf{g}^j) &= \mathbf{G}(\mathbf{u}_t^j \cdot \mathbf{u}_t^j) - (\mathbf{G}\mathbf{u}_t^j) \cdot (\mathbf{G}\mathbf{u}_t^j) + \mathbf{G}(\mathbf{g}^j \cdot \mathbf{g}^j) - (\mathbf{G}\mathbf{g}^j) \cdot (\mathbf{G}\mathbf{g}^j) + c_2,
\end{aligned} \tag{4.11}$$

are four functions, $\mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{G}$ is a matrix to implement the operation of the convolution with a Gaussian kernel and $c_1,\ c_2$ are two constants. Thus the SSIM between $\mathbf{u}_t^j$ and $\mathbf{g}^j$

is

$$\mathcal{S}(\mathbf{u}_t^j, \mathbf{g}^j) = \frac{1}{n} \sum_{i=1}^{n} \mathrm{SSIM}(\mathbf{u}_t^j, \mathbf{g}^j)_i \tag{4.12}$$

$$= \frac{1}{n} \mathbf{1}^\top \mathrm{SSIM}(\mathbf{u}_t^j, \mathbf{g}^j) \tag{4.13}$$

where $\mathbf{1}$ is a vector whose entries are all 1. For more details about Equation (4.10) we refer the reader to [158]. Because SSIM reaches the maximum value when two images are the same, we define the loss function in Equation (4.8) as

$$\ell(\mathbf{u}_t^j, \mathbf{g}^j) = 1 - \mathcal{S}(\mathbf{u}_t^j, \mathbf{g}^j)\,, \tag{4.14}$$

and minimize the resulting energy. For optimization we use the L-BFGS algorithm [105], which is a batch-based optimization algorithm. Using L-BFGS we need to specify the loss function (cf. (4.8)) and the according gradient. The gradient of Equation (4.8) is obtained by applying the chain rule, i.e.

$$\frac{\partial \mathcal{L}(\Theta_t)}{\partial \Theta_t} = \sum_{j=1}^{\mathrm{n_{trn}}} \frac{\partial \ell\left(\mathbf{u}_t^j, \mathbf{g}^j\right)}{\partial \Theta_t} = \sum_{j=1}^{\mathrm{n_{trn}}} \frac{\partial \mathbf{u}_t^j}{\partial \Theta_t} \frac{\partial \ell\left(\mathbf{u}_t^j, \mathbf{g}^j\right)}{\partial \mathbf{u}_t^j}\,, \tag{4.15}$$

where $\mathbf{u}_t^j$ is defined in Equation (4.3). After the greedy training we continue with the joint training, where we learn the parameters of all stages simultaneously. We seek to minimize the loss of the last stage. Thus the energy functional for the joint training can be formulated as

$$\mathcal{L}(\Theta_1, \ldots, \Theta_T) = \sum_{j=1}^{\mathrm{n_{trn}}} \ell(\mathbf{u}_T^j, \mathbf{g}^j)\,, \tag{4.16}$$

where, similar as in (4.18), we use the chain rule to obtain the gradient for the j^{th} training sample as

$$\frac{\partial \ell(\mathbf{u}_T^j, \mathbf{g}^j)}{\partial \Theta_t} = \frac{\partial \mathbf{u}_t^j}{\partial \Theta_t} \frac{\partial \mathbf{u}_{t+1}^j}{\partial \mathbf{u}_t^j} \cdots \frac{\partial \ell(\mathbf{u}_T^j, \mathbf{g}^j)}{\partial \mathbf{u}_T^j} \qquad \text{for} \qquad 1 \leqslant t \leqslant T\,. \tag{4.17}$$

Derivation of Greedy Training. The gradient of Equation (4.8) is obtained by applying the chain rule, i.e.

$$\frac{\partial \mathcal{L}(\Theta_t)}{\partial \Theta_t} = \sum_{j=1}^{\mathrm{n_{trn}}} \frac{\partial \ell(\mathbf{u}_t^j, \mathbf{g}^j)}{\partial \Theta_t} = \sum_{j=1}^{\mathrm{n_{trn}}} \frac{\partial \mathbf{u}_t^j}{\partial \Theta_t} \frac{\partial \ell(\mathbf{u}_t^j, \mathbf{g}^j)}{\partial \mathbf{u}_t^j}\,, \tag{4.18}$$

where $\mathbf{u}_t^j$ is defined in Equation (4.3). In the following section we will present the calculation of $\frac{\partial \ell(\mathbf{u}_t^j, \mathbf{g}^j)}{\partial \mathbf{u}_t^j}$, followed by the calculation of $\frac{\partial \mathbf{u}_t^j}{\partial \Theta_t}$. For simplicity, we set the

number of samples to $\mathrm{n_{trn}} = 1$. It is easy to generalize to the case $\mathrm{n_{trn}} > 1$ by adding up all samples.

Partial Derivative of the Loss w.r.t. $\mathbf{u}_t$. In this section, we will calculate the partial derivative of the loss $\ell(\mathbf{u}_t, \mathbf{g})$ w.r.t. $\mathbf{u}_t$. We omit the argument $\mathbf{g}^j$ in $\mathcal{S}_n(\mathbf{u}_t^j, \mathbf{g}^j), n \in \{1, 2, 3, 4\}$ for clarity. In the derivation, the following rules are used,

$$\frac{\partial \left[\mathcal{S}_i(\mathbf{u}_t) \cdot \mathcal{S}_j(\mathbf{u}_t)\right]}{\partial \mathbf{u}_t} = \frac{\partial \mathcal{S}_i(\mathbf{u}_t)}{\partial \mathbf{u}_t}\mathrm{diag}(\mathcal{S}_j(\mathbf{u}_t)) + \frac{\partial \mathcal{S}_j(\mathbf{u}_t)}{\partial \mathbf{u}_t}\mathrm{diag}(\mathcal{S}_i(\mathbf{u}_t)) \tag{4.19}$$

and

$$\frac{\partial \left[\mathcal{S}_i(\mathbf{u}_t) \cdot \mathcal{S}_j^{-1}(\mathbf{u}_t)\right]}{\partial \mathbf{u}_t} = \frac{\partial \mathcal{S}_i(\mathbf{u}_t)}{\partial \mathbf{u}_t}\mathrm{diag}(\mathcal{S}_j^{-1}(\mathbf{u}_t)) - \frac{\partial \mathcal{S}_j(\mathbf{u}_t)}{\partial \mathbf{u}_t}\mathrm{diag}(\mathcal{S}_i(\mathbf{u}_t) \cdot \mathcal{S}_j^{-2}(\mathbf{u}_t)), \tag{4.20}$$

where $\mathrm{diag}(\mathbf{v})\,,\ \mathbf{v} \in \mathbb{R}^n$ is an operation to generate a diagonal matrix, i.e.

$$\mathrm{diag}(\mathbf{v}) = \begin{bmatrix} v_1 & 0 & \cdots & 0 \\ 0 & v_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & v_n \end{bmatrix} \tag{4.21}$$

and $\mathcal{S}_j^{-1}(\mathbf{u}_t)$ and $\mathcal{S}_j^{-2}(\mathbf{u}_t)$ specify element-wise exponential. According to matrix calculus, the following equations hold,

$$\begin{aligned} \frac{\partial \mathcal{S}_1(\mathbf{u}_t)}{\partial \mathbf{u}_t} &= 2\mathbf{G}^\top \mathrm{diag}(\mathbf{G}\mathbf{g}) \\ \frac{\partial \mathcal{S}_2(\mathbf{u}_t)}{\partial \mathbf{u}_t} &= 2\mathrm{diag}(\mathbf{g})\mathbf{G}^\top - \mathbf{G}^T \mathrm{diag}(\mathbf{G}\mathbf{g}) \\ \frac{\partial \mathcal{S}_3(\mathbf{u}_t)}{\partial \mathbf{u}_t} &= 2\mathbf{G}^\top \mathrm{diag}(\mathbf{G}\mathbf{u}_t) \\ \frac{\partial \mathcal{S}_4(\mathbf{u}_t)}{\partial \mathbf{u}_t} &= 2\mathrm{diag}(\mathbf{u}_t)\mathbf{G}^\top - 2\mathbf{G}^\top \mathrm{diag}(\mathbf{G}\mathbf{u}_t). \end{aligned} \tag{4.22}$$

Because $\mathbf{G}$ represents a Gaussian kernel having the same deviation along each direction, the Gaussian kernel is rotationally symmetrical. Thus we have

$$\mathbf{G}^\top = \mathbf{G}. \tag{4.23}$$

Then by combining (4.22), (4.19) and (4.20), we attain

$$\begin{aligned}\frac{\partial \text{SSIM}(\mathbf{u}_t)}{\partial \mathbf{u}_t} = \mathbf{G}(&2\text{diag}(\mathbf{G}\mathbf{g}\cdot\mathcal{S}_2\cdot\mathcal{S}_3^{-1}\cdot\mathcal{S}_4^{-1})\\ &-\text{diag}(\mathbf{G}\mathbf{g}\cdot\mathcal{S}_1\cdot\mathcal{S}_3^{-1}\cdot\mathcal{S}_4^{-1})\\ &-2\text{diag}(\mathbf{G}\mathbf{u}_t\cdot\mathcal{S}_1\cdot\mathcal{S}_2\cdot\mathcal{S}_3^{-2}\cdot\mathcal{S}_4^{-1})\\ &+2\text{diag}(\mathbf{G}\mathbf{u}_t\cdot\mathcal{S}_1\cdot\mathcal{S}_2\cdot\mathcal{S}_3^{-1}\cdot\mathcal{S}_4^{-2}))\\ &+2\text{diag}(\mathbf{g})\mathbf{G}\text{diag}(\mathcal{S}_1\cdot\mathcal{S}_3^{-1}\cdot\mathcal{S}_4^{-1})\\ &-2\text{diag}(\mathbf{u}_t)\mathbf{G}\text{diag}(\cdot\mathcal{S}_1\cdot\mathcal{S}_2\cdot\mathcal{S}_3^{-1}\mathcal{S}_4^{-2})\end{aligned} \tag{4.24}$$

Note in Equation (4.24) we shorten $\mathcal{S}_j(\mathbf{u}_t)$ to $\mathcal{S}_j,\ j\in\{1,2,3,4\}$ for clearness. Therefore, the partial derivative of the loss w.r.t. $\mathbf{u}_t$ is

$$\frac{\partial \ell(\mathbf{u}_t,\mathbf{g}^j)}{\partial \mathbf{u}_t} = -\frac{\partial \mathcal{S}(\mathbf{u}_t,\mathbf{g})}{\partial \mathbf{u}_t} = -\frac{1}{n}\underbrace{\frac{\partial \text{SSIM}(\mathbf{u}_t,\mathbf{g})}{\partial \mathbf{u}_t}}_{\text{Equation (4.24)}}\mathbf{1}. \tag{4.25}$$

Partial Derivative of $\mathbf{u}_t$ w.r.t. the Parameters $\Theta_t = \left\{\mathbf{k}_i^t, \phi_i^t\right\}$. In this section, we will illustrate the calculation of the partial derivative of $\mathbf{u}_t$ w.r.t. the kernel $\mathbf{k}_i^t$ and w.r.t. the influence function ϕ_i^t.

The kernel $\mathbf{k}_i^t$ is the combination of DCT basises with zero-mean. Let the size of $\mathbf{k}_i^t$ be $\text{z}\times\text{z}$ and the number of DCT basises be $\lambda = \text{z}^2 - 1$. The kernel $\mathbf{k}_i^t$ is formulated as

$$\mathbf{k}_i^t = \sum_{j=1}^{\lambda}\alpha_j^{it}\mathbf{b}_j \text{ or } \mathbf{K}_i^t = \sum_{j=1}^{\lambda}\alpha_j^{it}\mathbf{B}_j \tag{4.26}$$

where $\mathbf{b}_j$ and $\mathbf{B}_j$ denote the DCT basises (e.g., $\mathbf{B}_j\mathbf{g} = \mathbf{b}_j * \mathbf{g}$) and $i = 1,\cdots,n_k$ is the number of kernels we used. Note at each stage t, we have n_k different kernels. Refer to Equation (4.3). Now it is clear the object is to learn the coefficients of $\alpha_j^{it}, j = 1,\cdots,\lambda$ for the i^{th} kernel at stage t. Let $\boldsymbol{\alpha}^{it} = \begin{bmatrix}\alpha_1^{it} & \cdots & \alpha_\lambda^{it}\end{bmatrix}^\top$ be a column vector.

By Equation (4.3), we obtain

$$\begin{aligned}\frac{\partial \mathbf{u}_t}{\partial \boldsymbol{\alpha}^{it}} &= -\frac{\partial\left\{\mathbf{m}\cdot {K_i^t}^\top \phi_i^t(\mathbf{K}_i^t\mathbf{u}_{t-1})\right\}}{\partial \boldsymbol{\alpha}^{it}}\\ &= -\frac{\partial\left\{{K_i^t}^\top \phi_i^t(\mathbf{K}_i^t\mathbf{u}_{t-1})\right\}}{\partial \boldsymbol{\alpha}^{it}}\text{diag}(\mathbf{m}).\end{aligned} \tag{4.27}$$

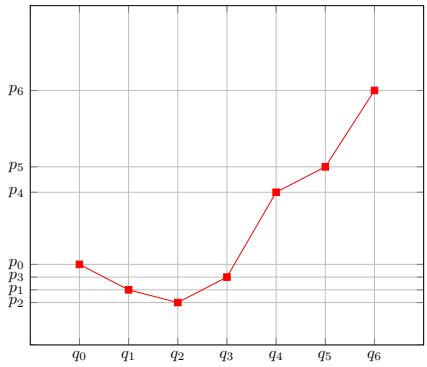


Figure 4.2: A demonstration of the piecewise linear function ξ_i^t.

By substituting Equation (4.26) into Equation (4.27), we have

$$\frac{\partial \left\{ {K_i^t}^\top \phi_i^t(\mathbf{K}_i^t \mathbf{u}_{t-1}) \right\}}{\partial \boldsymbol{\alpha}^{it}} = \underbrace{\begin{bmatrix} \phi_i^t(\mathbf{K}_i^t \mathbf{u}_{t-1})^\top \mathbf{B}_1 \\ \vdots \\ \phi_i^t(\mathbf{K}_i^t \mathbf{u}_{t-1})^\top \mathbf{B}_\lambda \end{bmatrix}}_{\mathbb{R}^{\lambda \times n}} + \underbrace{\begin{bmatrix} \mathbf{u}_{t-1}^\top \mathbf{B}_1^\top \\ \vdots \\ \mathbf{u}_{t-1}^\top \mathbf{B}_\lambda^\top \end{bmatrix}}_{\mathbb{R}^{\lambda \times n}} \underbrace{\frac{\partial \phi_i^t(\mathbf{K}_i^t \mathbf{u}_{t-1})}{\partial (\mathbf{K}_i^t \mathbf{u}_{t-1})}}_{\mathbb{R}^{n \times n}} \underbrace{\mathbf{K}_i^t}_{\mathbb{R}^{n \times n}}. \tag{4.28}$$

Note $\frac{\partial \phi_i^t(\mathbf{K}_i^t \mathbf{u}_{t-1})}{\partial (\mathbf{K}_i^t \mathbf{u}_{t-1})}$ is a diagonal matrix because the output value of the influence function ϕ_i^t at the j^{th} position only relates with the input value at the j^{th} position as shown in Equation (4.5). By substituting Equation (4.28) in to Equation (4.27), we can calculate $\frac{\partial \mathbf{u}_t}{\partial \boldsymbol{\alpha}^{it}}$.

Because we freely adjust the influence function ϕ_i^t and the kernel $\mathbf{K}_i^t$ simultaneously, we add a constrain to the coefficients of $\alpha_j^{it}, j = 1, \cdots, \lambda$,

$$\|\boldsymbol{\alpha}^{it}\|_2 = 1. \tag{4.29}$$

For this reason we introduce parameters

$$\boldsymbol{\alpha}^{it} = \frac{\boldsymbol{c}^{it}}{\|\boldsymbol{c}^{it}\|_2}, \tag{4.30}$$

where $\boldsymbol{c}^{it} \in \mathbb{R}^\lambda$. It is easy to get

$$\frac{\mathrm{d}\boldsymbol{\alpha}^{it}}{\mathrm{d}\boldsymbol{c}^{it}} = \frac{1}{\|\boldsymbol{c}^{it}\|_2} \left(\mathbf{I} - \frac{\boldsymbol{c}^{it} {\boldsymbol{c}^{it}}^\top}{\|\boldsymbol{c}^{it}\|_2^2} \right), \tag{4.31}$$

where $\mathbf{I} \in \mathbb{R}^{\lambda \times \lambda}$ is the identity matrix. Therefore, the partial derivative of $\mathbf{u}_t$ w.r.t. the

kernel is

$$\frac{\partial \mathbf{u}_t}{\partial \boldsymbol{c}^{it}} = -\left\{\frac{1}{\|\boldsymbol{c}^{it}\|_2}(\mathbf{I} - \frac{\boldsymbol{c}^{it}\boldsymbol{c}^{it\top}}{\|\boldsymbol{c}^{it}\|_2^2})\right\} \left\{\begin{bmatrix}\phi_i^t(\mathbf{K}_i^t\mathbf{u}_{t-1})^\top\mathbf{B}_1\\ \vdots \\ \phi_i^t(\mathbf{K}_i^t\mathbf{u}_{t-1})^\top\mathbf{B}_\lambda\end{bmatrix} + \begin{bmatrix}\mathbf{u}_{t-1}^\top\mathbf{B}_1^\top\\ \vdots \\ \mathbf{u}_{t-1}^\top\mathbf{B}_\lambda^\top\end{bmatrix}\frac{\partial\phi_i^t(\mathbf{K}_i^t\mathbf{u}_{t-1})}{\partial(\mathbf{K}_i^t\mathbf{u}_{t-1})}\mathbf{K}_i^t\right\}\text{diag}(\mathbf{m}) \tag{4.32}$$

In the following, we calculate the partial derivative of $\mathbf{u}_t$ w.r.t. the influence function ϕ_i^t. First we will illustrate how the influence function is formulated and next we will calculate the partial derivative. Suppose $\mathbf{v} = \mathbf{K}_i^t\mathbf{u}_{t-1} \in \mathbb{R}^n$, $\phi_i^t(\mathbf{v})$ is formulated as

$$\phi_i^t(\mathbf{v})_j = \xi_i^t(\mathbf{v}_j), \quad j \in \{1, \cdots, n\}. \tag{4.33}$$

To represent ξ_i^t, we use the technique of slice transform [79]. In this way $\xi_i^t : \mathbb{R} \to \mathbb{R}$ is a continuous piecewise linear function, e.g., as shown in Figure 4.2. Let the domain of ξ_i^t be $[a, b)$. The interval is divided into Υ bins, (e.g., $\Upsilon = 6$ in Figure 4.2), which generates a vector $\mathbf{q} = [q_0, \cdots, q_\Upsilon]^\top$ and $a = q_0,\ b = q_\Upsilon$. The function ξ_i^t is parameterised by the vector $\mathbf{p_i^t} = [p_0, \cdots, p_\Upsilon]^\top$. Thus the partial derivative of $\mathbf{u}_t$ w.r.t. the influence function ϕ_i^t is denoted as

$$\begin{aligned}\frac{\partial \mathbf{u}_t}{\partial \phi_i^t(\mathbf{K}_i^t\mathbf{u}_{t-1})} &= \frac{\partial \mathbf{u}_t}{\partial \mathbf{p}_i^t} \\ &= -\frac{\partial \phi_i^t(\mathbf{K}_i^t\mathbf{u}_{t-1})}{\partial \mathbf{p}_i^t}\mathbf{K}_i^t\text{diag}(\mathbf{m}) \\ &= -\frac{\partial \phi_i^t(\mathbf{v})}{\partial \mathbf{p}_i^t}\mathbf{K}_i^t\text{diag}(\mathbf{m}).\end{aligned} \tag{4.34}$$

Now we illustrate how to calculate $\frac{\partial \phi_i^t(\mathbf{v})}{\partial \mathbf{p}_i^t}$.

As in [79], we define the following functions,

$$\pi(x) = b \ \text{ if } \ x \in [q_{b-1}, q_b) \tag{4.35}$$

and

$$r(x) = \frac{x - q_{\pi(x)-1}}{q_{\pi(x)} - q_{\pi(x)-1}} \tag{4.36}$$

where $x \in \mathbb{R}$. Thus the scaler x can be written as a linear combination of $q_{\pi(x)}$ and $q_{\pi(x)-1}$

$$x = r(x)q_{\pi(x)} + (1 - r(x))q_{\pi(x)-1}. \tag{4.37}$$

Equation (4.37) can be formulated as

$$x = \mathbf{C}_{\mathbf{q}}(x)\mathbf{q} \tag{4.38}$$

where $\mathbf{C}_{\mathbf{q}}(x) \in \mathbb{R}^{\Upsilon+1}$ is a row vector

$$\mathbf{C}_{\mathbf{q}}(x) = [0, \cdots, 0, 1 - r(x), \underbrace{r(x)}_{\text{at the } \pi(x)^{\text{th}} \text{ entry}}, 0, \cdots, 0]. \tag{4.39}$$

Therefore, we have

$$\xi_i^t(x) = \mathbf{C}_{\mathbf{q}}(x)\mathbf{p} \tag{4.40}$$

and it is easy to get

$$\frac{\partial \xi_i^t(x)}{\partial \mathbf{p}} = \mathbf{C}_{\mathbf{q}}(x)^\top. \tag{4.41}$$

Now we return to our context and finally get

$$\begin{aligned} \frac{\partial \phi_i^t(\mathbf{v})}{\partial \mathbf{p}_i^t} &= \begin{bmatrix} \frac{\partial \phi_i^t(\mathbf{v})_1}{\partial \mathbf{p}_i^t} & \cdots & \frac{\partial \phi_i^t(\mathbf{v})_n}{\partial \mathbf{p}_i^t} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial \xi_i^t(\mathbf{v}_1)}{\partial \mathbf{p}_i^t} & \cdots & \frac{\partial \xi_i^t(\mathbf{v}_n)}{\partial \mathbf{p}_i^t} \end{bmatrix} \\ &= \begin{bmatrix} C_{\mathbf{q}}(\mathbf{v}_1)^\top & \cdots & C_{\mathbf{q}}(\mathbf{v}_n)^\top \end{bmatrix} \end{aligned} . \tag{4.42}$$

By plugging (4.42) into (4.34), we get the final result

$$\frac{\partial \mathbf{u}_t}{\partial \phi_i^t(\mathbf{K}_i^t \mathbf{u}_{t-1})} = -\begin{bmatrix} C_{\mathbf{q}}(\mathbf{v}_1)^\top & \cdots & C_{\mathbf{q}}(\mathbf{v}_n)^\top \end{bmatrix} \mathbf{K}_i^t \text{diag}(\mathbf{m}). \tag{4.43}$$

Note that to command the smoothness of $\mathbf{p}$ and to avoid overfitting, there are many techniques in the field of computer vision and machine learning, e.g., using regularizer, dropping out, enlarging the training database and so on.

Derivation of Joint Training. In the joint training, we use the parameters $\Theta_t = \left\{\mathbf{k}_i^t, \phi_i^t\right\}, t = 1, \cdots, T$ trained from the greedy training as initialization and we optimise the parameters of all stages simultaneously. The energy functional for the joint training can be formulated as

$$\mathcal{L}(\Theta_1, \ldots, \Theta_T) = \sum_{j=1}^{K} \ell(\mathbf{u}_T^j, \mathbf{g}^j). \tag{4.44}$$

By the chain rule, we get the gradient for the j^{th} training sample as

$$\frac{\partial \ell(\mathbf{u}_T^j, \mathbf{g}^j)}{\partial \Theta_t} = \frac{\partial \mathbf{u}_t^j}{\partial \Theta_t} \frac{\partial \mathbf{u}_{t+1}^j}{\partial \mathbf{u}_t^j} \cdots \frac{\partial \ell(\mathbf{u}_T^j, \mathbf{g}^j)}{\partial \mathbf{u}_T^j} \qquad \text{for} \qquad 1 \leqslant t \leqslant T\,. \tag{4.45}$$

In Equation (4.45), $\frac{\partial \ell(\mathbf{u}_T^j, \mathbf{g}^j)}{\partial \mathbf{u}_T^j}$ can be calculated from Equation (4.25) and $\frac{\partial \mathbf{u}_t^j}{\partial \Theta_t}$ can be calculated from Equation (4.32). In this section, we mainly illustrate the calculation of $\frac{\partial \mathbf{u}_{t+1}^j}{\partial \mathbf{u}_t^j}$. By applying the chain rule to Equation (4.3), we obtain

$$\frac{\partial \mathbf{u}_t^j}{\partial \mathbf{u}_{t-1}^j} = \mathbf{I} - (\sum_{i=1}^{N_k} \mathbf{K_i^t}^\top \frac{\partial \phi(\mathbf{K}_i^t \mathbf{u}_{t-1})}{\partial (\mathbf{K}_i^t \mathbf{u}_{t-1})} \mathbf{K_i^t}) \mathrm{diag}(\mathbf{m}), \tag{4.46}$$

where $\mathbf{I} \in \mathbb{R}^{N \times N}$ is an identity matrix. It is straightforward to obtain $\frac{\partial \mathbf{u}_{t+1}^j}{\partial \mathbf{u}_t^j}$ from $\frac{\partial \mathbf{u}_t^j}{\partial \mathbf{u}_{t-1}^j}$. To optimize the involved parameters in Equation (4.3) we use L-BFGS[2] [105], which minimizes the energy by iteratively approximating the inverse Hessian matrix.

4.1.1.3 Experiments

In this section, we demonstrate the results of generic models which are trained on an entire dataset and specific models which are learned for inpainting a certain image. When trying to evaluate inpainting algorithms one recognizes that there exists no well-defined ground truth. This is apparent when considering inpainting problems involving large image regions, where multiple natural-looking solutions might exist. However, in order to provide a quantitative evaluation for inpainting methods authors mainly use predefined ground truth images and compare their methods based on the PSNR or similar measures. Within this thesis we follow this type of evaluation, and we will present a comprehensive comparison to state-of-the-art inpainting methods based on the TUM-image inpainting database [155, url] using the following three measures: PSNR, SSIM [158], and Gradient Similarity Image Measure (GSIM) [103]. On the website of [url], the authors suggested four state-of-the-art inpainting algorithms for comparison [25, 67, 81, 170]. Algorithms [25, 81, 170] cannot be used for compression type inpainting tasks. Therefore, we will compare in this case to Laplacian, and TV inpainting [67], to the learned MRF prior proposed by Chen et al. [42], to the learned pairwise MRF model, and to the FoE model with Gaussian scale mixtures as experts, both proposed by Schmidt et al. [145]. For the remaining experiments we will compare to all aforementioned algorithms if a ground truth is available.

For all experiments we first conduct a greedy training, where L-BFGS runs for 200 iterations in each stage. Afterwards we jointly train the parameters of all stages. The filters $\mathbf{k}_i^t$ are initialized with the Discrete Cosine Transform (DCT) bases, and the influence functions are initialized as $\phi_i^t(x) = 0.01x$. The joint training is then initialized with the result of the greedy training. We mainly investigate two types of initialization of the inpainting regions. The first one is random initialization which randomly generates gray values in the unknown inpainting regions. The second one is initialization with linear

[2]Used solver: `http://www.cs.toronto.edu/~liam/software.shtml`

		Laplacian	TV [67]	Chen [42]	Schmidt MRF [145]	Schmidt FoE [145]	TD
80%	PSNR	21.5998	20.8886	22.5123	21.4210	21.8630	22.6301
	SSIM	0.7864	0.7376	0.8185	0.7781	0.8028	0.8267
	GSIM	0.7417	0.7041	0.7947	0.7451	0.7844	0.8048
90%	PSNR	20.0751	19.0507	20.4899	19.7449	20.1043	20.6663
	SSIM	0.6689	0.5710	0.7094	0.6519	0.6905	0.7287
	GSIM	0.6636	0.6029	0.7119	0.6562	0.7106	0.7409

Table 4.1: Quantitative inpainting results for 80% and 90% random missing pixels.

	Laplacian	TV [67]	Chen [42]	Schmidt MRF [145]	Schmidt FoE [145]	Bugeau [25]	Herling [81]	Xu [170]	TD
PSNR	35.1529	34.2324	35.5031	34.7119	35.1434	33.5746	33.1427	35.3308	35.9084
SSIM	0.9902	0.9884	0.9907	0.9897	0.9905	0.9880	0.9864	0.9900	0.9915
GSIM	0.9910	0.9898	0.9910	0.9909	0.9920	0.9917	0.9915	0.9919	0.9925

Table 4.2: Quantitative results for inpainting a small connected image region.

diffusion which conducts linear diffusion on the inpainting regions. We formulate the problem of linear diffusion as a system of linear equations and solve it by calculating the inverse of a matrix. We use the random initialization for inpainting of 80% and 90% random missing pixels because of its efficiency and good performance. For inpainting small connected regions, we use the initialization with linear diffusion because it provides better inpainted results. Thus for specific inpainting demonstrated in the supplementary material, we first conduct linear diffusion to generate the input images of the trained diffusion models. In the following we will use the short notation TD to denote the obtained trained diffusion model.

Inpainting of 80% and 90% random missing pixels. For this experiment we train two generic diffusion models, where we use 30 stages with 48 filters. In the case of 80% missing pixels we use a kernel size of 7×7. Inpainting of 90% random missing pixels is a more challenging problem, thus we allow the model to explore more available information by increasing the kernel size to 11×11. The parameters of the trained diffusion models are optimized in the joint training using 100 iterations of L-BFGS. We evaluate the results based on the TUM-image inpainting database [url]. Table 4.1 provides quantitative results in terms of the mean PSNR, SSIM, and GSIM. We observe that the proposed diffusion model achieves excellent results and is able to outperform the competing methods. Figure 4.3 provides corresponding qualitative results, where we also observe clearly visible qualitative improvements.

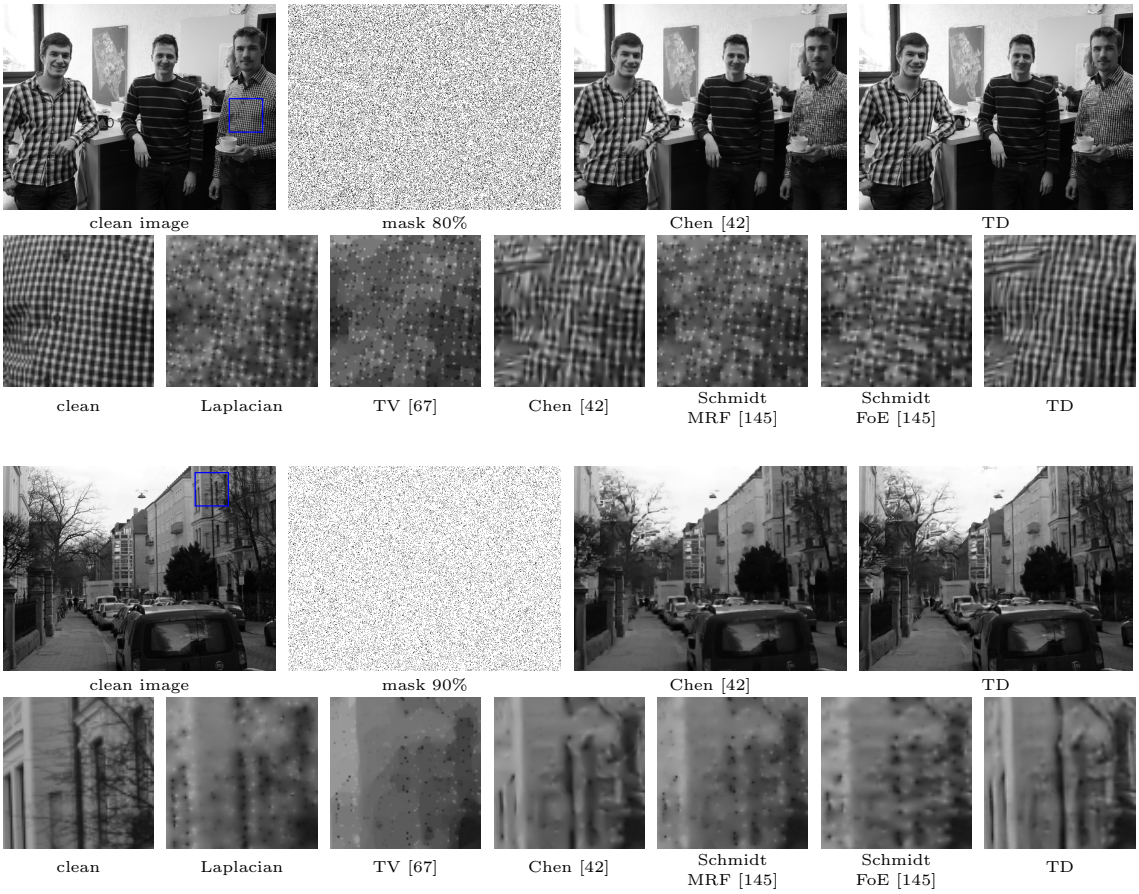


Figure 4.3: Qualitative inpainting result for 80% and 90% random missing pixels. The closeup views clearly show the advantage of the proposed trained diffusion (TD) model.

Inpainting of small regions. In this experiment, we train a 30-stage generic diffusion model with 24 filters of size 5×5. For the training, we randomly generate several small regions, where each region occupies about 300 connected pixels. Quantitative result for this experiment are provided in Table 4.2, where the proposed model achieves superior results compared to state-of-the-art methods. The table shows the mean PSNR, SSIM, and GSIM values for inpainting the images of the TUM-image inpainting database [url], where the inpainting region is defined by the mask shown in Figure 4.4. Closeup views are also presented in Figure 4.4, where we observe that the proposed method better preserves the image edges than the other PDE based image inpainting methods. The methods by Bugeau et al. [25] and Herling and Broll [81] introduce visual artifacts, e.g. at the tire of the bike (bottom left in the closeup views). The result of the method by Xu and Sun [170] has less visual artifacts, but it also introduces some noise. The proposed TD model on the other hand provides a convincing result with less artifacts.

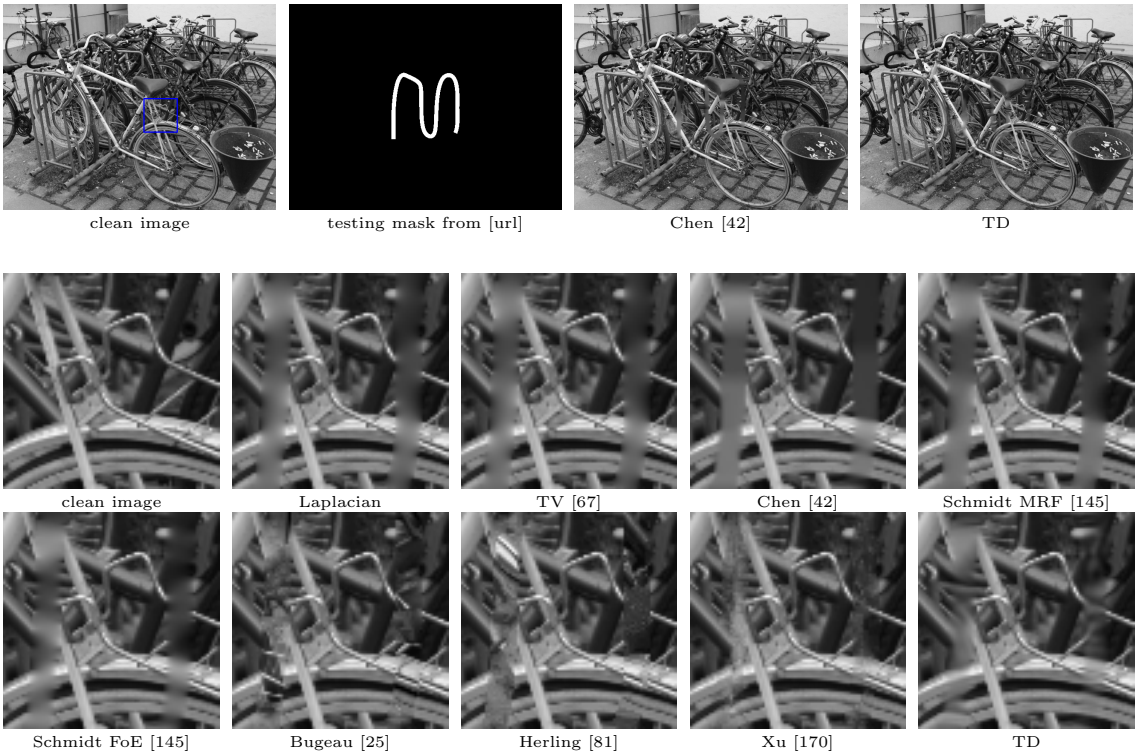


Figure 4.4: Qualitative results for inpainting a small connected image region.

Inpainting of a specific texture. Now we present experiments for texture inpainting, where we learn the specific texture within a single given image $\mathbf{I}$. The task is to inpaint a predefined image region within the same image (e.g. the blue regions indicated in Figure 4.5). In order to train a specific diffusion model we generate $K = 5$ training images $\mathbf{f}^j$, where pixels within in a mask $\mathbf{m}$ (Figure 4.5 indicates the training mask in green) are generated by linear diffusion. Thus we obtain the training set $\{\mathbf{f}^j, \mathbf{g}^j, \mathbf{m}^j\}_{j=1}^K$ (cf. Section 4.1.1.2), where $\mathbf{g}^j = \mathbf{I}$ and $\mathbf{m}^j = \mathbf{m}$ for $1 \leqslant j \leqslant K$. Based on this training set we learn a diffusion model. We first conduct a greedy training optimized by L-BFGS [105] running for 200 iterations in each stage. Next we jointly train the parameters of all stages. Because of the smoothness assumption and the local property of PDE based image inpainting methods, those type of methods are in general not well suited for the task of texture inpainting. Exemplar based image inpainting methods on the other hand are non-local and have achieved good performance for texture inpainting [25, 81, 170]. Anyhow, our experiments show that the proposed TD model is also able to perform texture inpainting even with small training datasets. The TD inpainting results and the exemplar based results are like two of a kind. We assume the reason is that there are many similarities between the training inpaiting domain and the testing inpainting domain.

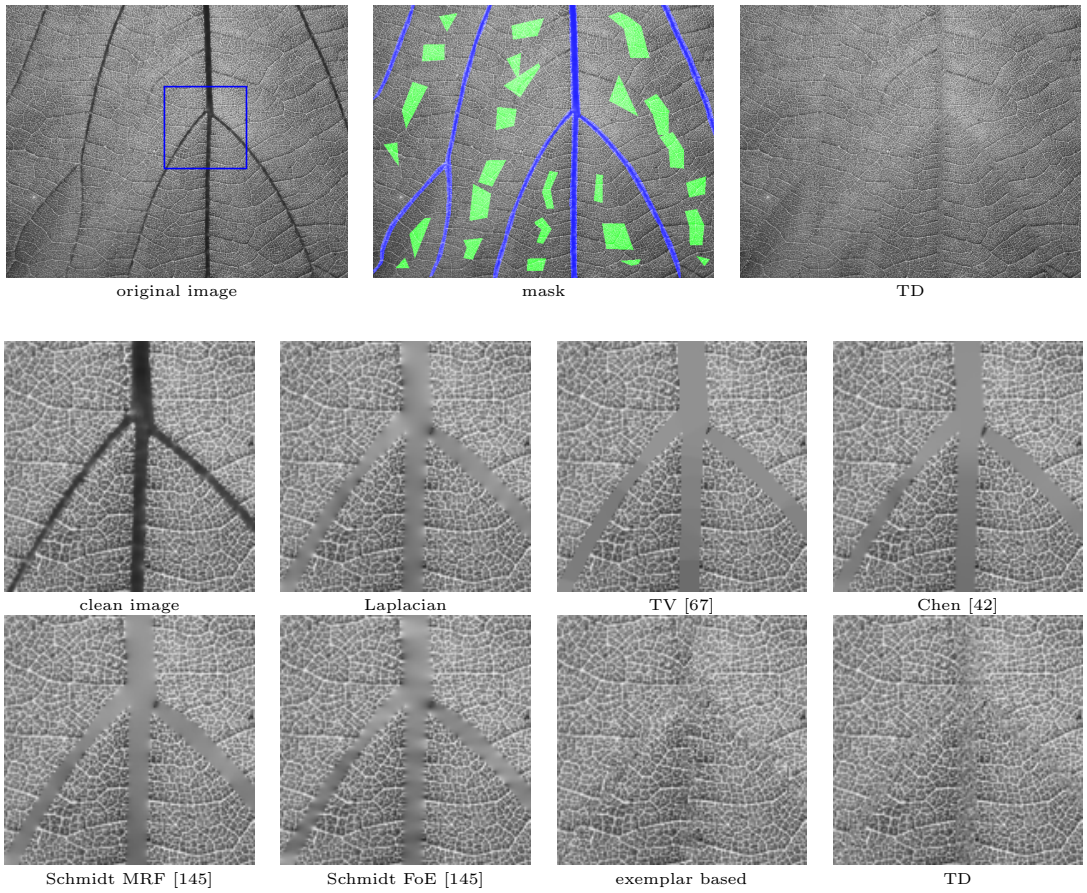


Figure 4.5: Qualitative results for texture inpainting. The image shows a closeup view of a leaf, where the inpainting regions are highlighted in blue in the mask at the top. The closeup views clearly show that the trained diffusion (TD) model is able to produce a natural-looking result, that is close to the result obtained with an exemplar based method. Other PDE based methods fail on this inpainting task.

Now, we continue to show the results of specific inpainting and associated experimental settings. Figure 4.5 provides some qualitative inpainting results. The size of the leaf-texture image is 500×625, see Figure 4.5. We generate many inpainting regions and the size of each inpainting region is about 800 pixels. In this experiment, we train a 15 stages model with 24 kernels with size of 19×19. In the joint training, the parameters of the model are optimized using 100 iterations of L-BFGS. In the tiger experiment illustrated in Figure 4.6, the test inpainting domain is the fence in the foreground of a 854×761 image. And the training inpaining regions are randomly generated regions and each inpainting region occupies around 1000 pixels. We train a 15 stages model with 24 kernels of size 19×19. In the joint training, the models is also optimized using 100 iterations of L-BFGS. From Figure 4.6, we can observe the result of our method is quite promising. The last experiment conducts on the wood image. As

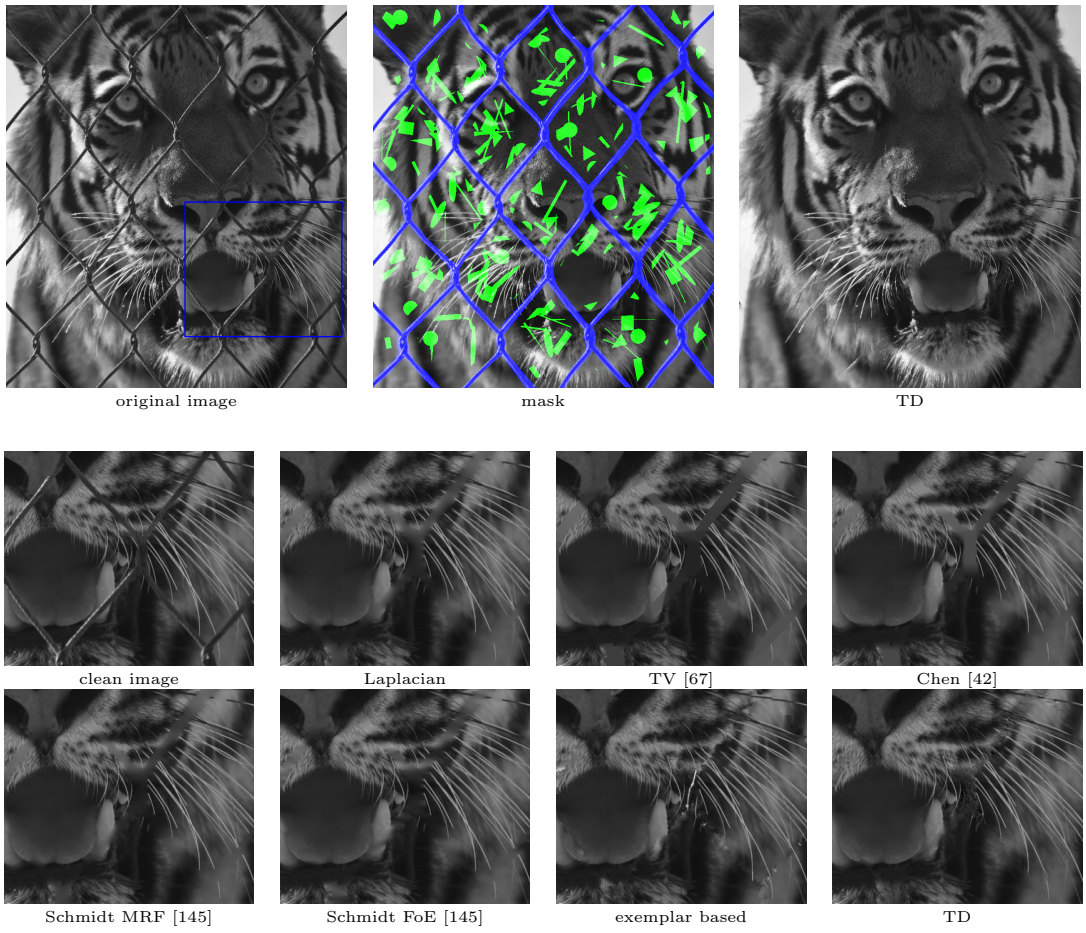


Figure 4.6: Qualitative results for texture inpainting. The image shows a tiger behind a fence, where the inpainting region is the fence in the foreground highlighted in blue in the mask at the top. The closeup views clearly show that the trained diffusion (TD) model is able to produce a natural-looking result, that is close to the result obtained with an exemplar based method.

shown in Figure 4.7, the size of image of wood is 640×480. Each inpainting region we generated occupies about 200 pixels. The trained model is 10 stage and have 24 kernels of size 13×13. The number of iteration of L-BFGS running is 100. Figure 4.7 shows that our method provides a piece of united wood after inpainting.

4.2 Light Field Image Processing

A Light Field (LF) [69, 101] is a 4D function that provides in addition to the spatial information, that corresponds to the information of a traditional 2D image, also directional information. The additional directional information includes information about the geometry of the observed scene, and thus gave rise to interesting applications,

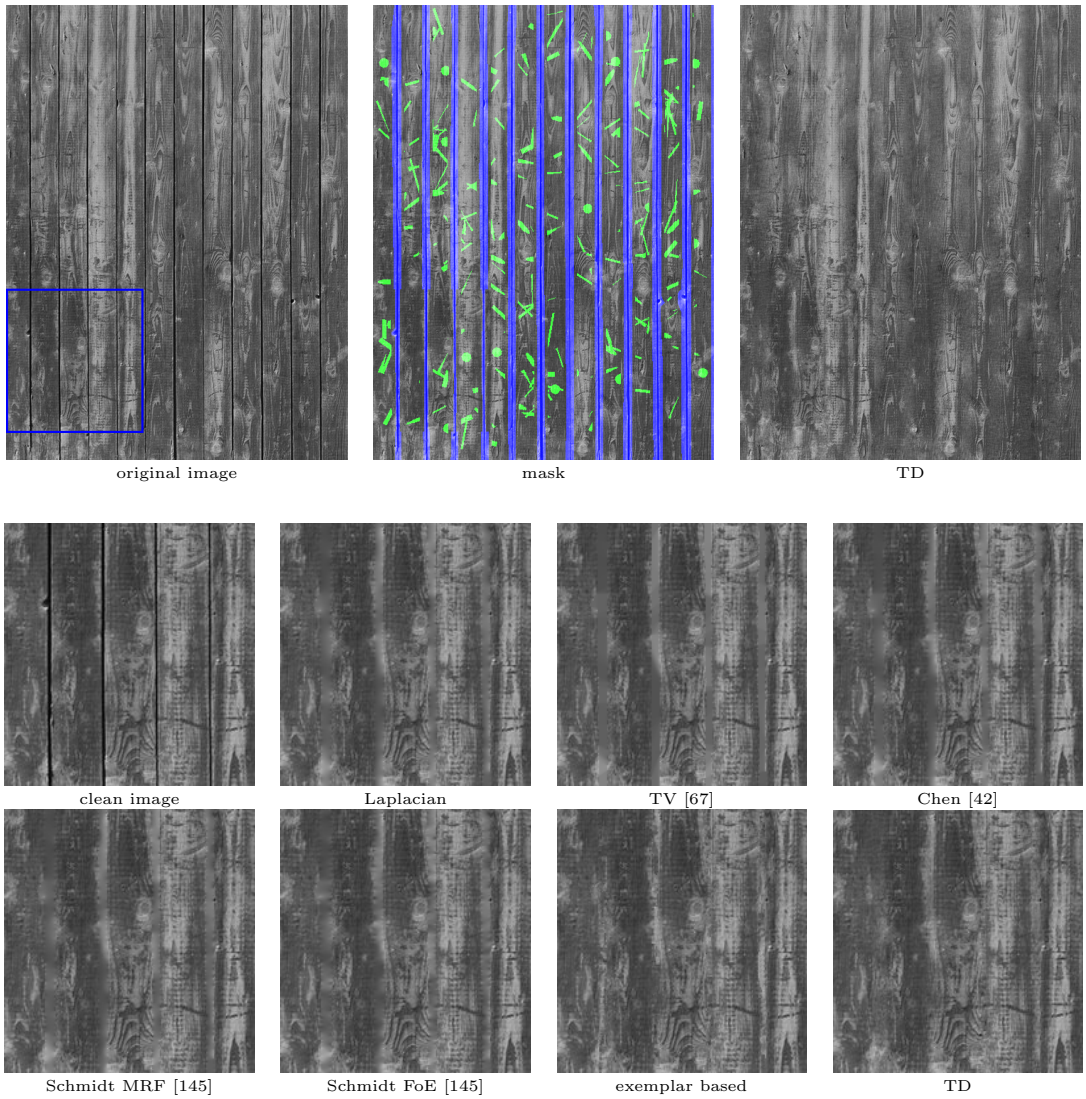
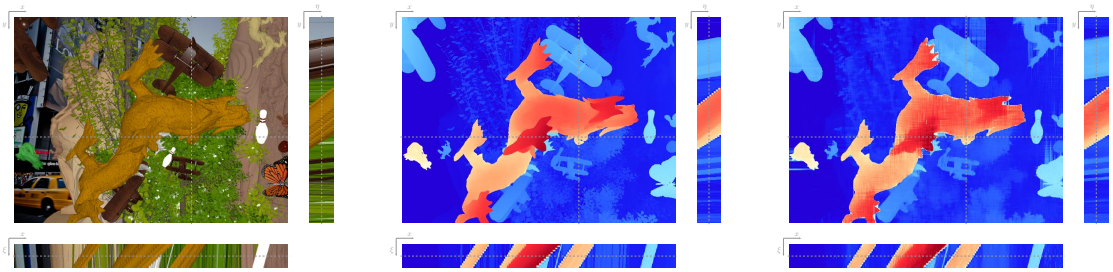


Figure 4.7: Qualitative results for texture inpainting. The image shows wood planks, where we inpaint the space between neighboring planks highlighted in blue in the mask at the top. The closeup views clearly show that the trained diffusion (TD) model is able to produce a natural-looking result, that is close to the result obtained with an exemplar based method.

like for instance digital re-focusing [85, 121], digital viewpoint manipulation [121], or depth estimation [68, 76, 78, 88, 152, 159]. All of these tasks are basically impossible to realize given a single traditional 2D image, that only provides the spatial intensity information.

A LF is commonly described using the so-called two-plane parametrization. This type of parametrization defines a ray by the intersection points of two parallel planes. Those planes are referred to as image plane $\Omega \subseteq \mathbb{R}^2$ and lens plane $\Pi \subseteq \mathbb{R}^2$. Thus in mathematical



(a) LF **(b)** ground truth **(c)** proposed

Figure 4.8: Illustration of Light Field data. Figure 4.8a shows a sub-aperture image with vertical and horizontal EPIs. The EPIs correspond to the positions indicated with dashed lines in the sub-aperture image. Figure 4.8b shows the corresponding color-coded ground truth disparity field. Figure 4.8c shows the result of the proposed model.

terms the LF is given as

$$\mathcal{L} : \Omega \times \Pi \to \mathbb{R}, \qquad (\mathbf{p}, \mathbf{q}) \mapsto L(\mathbf{p}, \mathbf{q}) \,, \tag{4.47}$$

where $\mathbf{p} = (x, y)^\top \in \Omega$ and $\mathbf{q} = (\xi, \eta)^\top \in \Pi$ represent the spatial and directional coordinates.

There are different ways of visualizing the 4D LF. In this thesis we use the so-called Epipolar Plane Image (EPI) representation. In terms of Equation (4.49) an EPI is obtained by holding one spatial and one directional coordinate constant. For instance by choosing a certain y and a certain η we restrict the 4D LF to the 2D function

$$\Sigma_{y,\eta} : \mathbb{R}^2 \to \mathbb{R}, \qquad (x, \xi) \mapsto \mathcal{L}(x, y, \xi, \eta) \,, \tag{4.48}$$

that defines a horizontal EPI. In a similar way one can also define vertical EPIs. The EPI representation can be considered as a 2D slice through the 4D LF, and it illustrates the linear characteristic of the LF space. See Figure 4.8a for an illustration.

4.2.1 U-shaped Networks for Shape from Light Field

This section presents a novel technique for Shape from Light Field (SfLF), that utilizes deep learning strategies. Our model is based on a fully convolutional network, that involves two symmetric parts, an encoding and a decoding part, leading to a u-shaped network architecture. By leveraging a recently proposed LF dataset, we are able to effectively train our model using supervised training. To process an entire LF we split the LF data into the corresponding EPI representation and predict each EPI separately. This strategy provides good reconstruction results combined with a fast prediction time. In the experimental section we compare our method to the state of the art. The method performs well in terms of depth accuracy, and is able to outperform competing methods

in terms of prediction time by a large margin.

4.2.1.1 Related Work

One of the most important research topics in LF image processing is the development of efficient and reliable shape extraction methods. Those methods are the foundation of various applications, like for instance digital refocusing [85, 121], image segmentation [162], or super-resolution [19, 160], to name but a few. The main focus of research regarding SfLF lies on developing methods to accurately reconstruct the observed scene at depth discontinuities or occlusion boundaries. For this purpose various approaches have been proposed, including specialized multi-view stereo techniques [38, 78] and methods based on an EPI analysis [68, 159]. Wanner and Goldluecke [68, 159] used for example the 2D structure tensor to measure the direction of each position in the vertical and horizontal EPIs. The results are then fused using variational methods by incorporating additional global visibility constraints. In [78] Heber et al. proposed a variational multi-view stereo method based on a technique called Active Wavefront Sampling (AWS). Tao et al. [152] proposed a fusion method that uses correspondence and defocus cues. Chen et al. [38] introduced a bilateral consistency metric on the surface camera to indicate the probability of occlusions, which was further used for LF stereo matching. Heber and Pock [76] proposed a variational method, that shears the LF by applying a low-rank assumption, where the depth information is provided by the amount of shearing. Jeon et al. [88] proposed an algorithm for SfLF, that utilizes phase shift based subpixel displacements. In [77] Heber and Pock presented a method for SfLF that applies a conventional Convolutional Neural Network (CNN) in a sliding window fashion. Up to this point deep learning techniques were barely used in LF image processing. Utilizing trained models for SfLF is an interesting idea to address certain limitations of previous methods. On the one hand a trained model has the ability to learn how to handle occlusion and aliasing artifacts, and on the other hand a CNN also allows faster computation times.

The entire field of deep learning flourishes with innovations, one after another. However, the exploration of 4D LF data by CNNs is still limited. In this section we seize ideas presented in [77]. Furthermore this work also builds upon fully convolutional networks [108] and up-convolution-based approaches [52, 108, 178], i.e. the proposed network architecture consists of a contracting and an expanding path, that involve only convolutional layers. The former path compresses the information and simultaneously captures context, and the latter path extracts the information and upsamples it to the original size. The expanding path is more or less symmetric to the contracting path, yielding a u-shaped architecture, that can be trained in an end-to-end scheme.

4.2.1.2 Methodology

In this section we describe the methodology of the proposed approach. The success of the proposed CNN depends on leveraging a set of recent improvements, that include

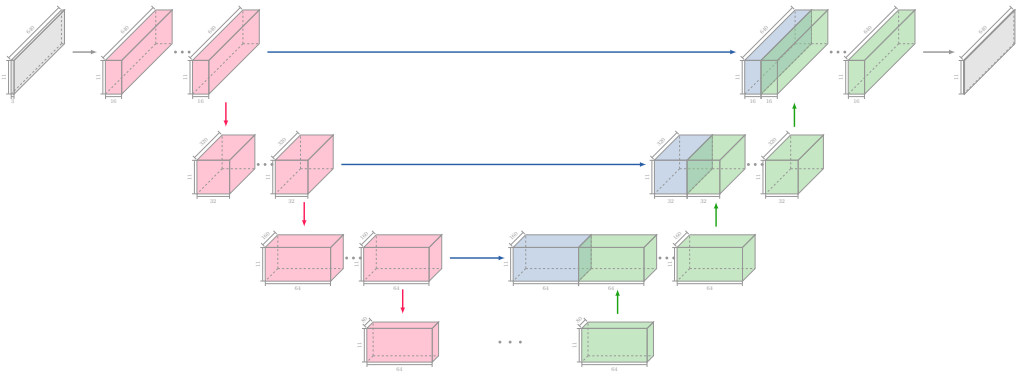


Figure 4.9: Illustration of the proposed u-shaped network architecture. The encoding and decoding parts of the network are highlighted in purple and green, respectively. The pinhole connections are marked in blue.

up-convolutions [108], no explicit pooling [148], and the Adam optimization method [92]. The section starts with a short introduction to CNNs, followed by the description of the used u-shaped network architecture. At the end of the section we provide details regarding the network training and the leveraged dataset.

Network Architecture. In contrast to methods that use natural images we are not able to exploit existing trained networks, i.e. we opt for designing our network entirely from scratch. However, not relying on pre-trained networks also allows to better adapt the network structure to the problem at hand. The proposed network is a fully convolutional network consisting of a contracting part and an expanding part. The first part acts as an encoder, that spatially compresses the image and thus reduces the input data to an essential feature representation. The bottom part processes the essential features, before the expanding part of the network decodes the simple feature representation to an output disparity image. The encoding and decoding parts of the network are basically symmetric leading to an u-shaped network architecture. An overview of the network structure is depicted in Figure 4.9, where the encoding and decoding parts of the network are highlighted in purple and green, respectively.

The u-shaped network uses down and up-convolutional layers for the encoding and decoding part, respectively. A down-convolution layer is obtained by increasing the stride of the convolution, i.e. it only computes a subset of all positions. This decreases the spatial resolution of the following layer, and simultaneously increases the spatial support of all subsequent layers. To increase the image resolution again we use so-called up-convolutional layers. Those layers use fractional strides to increase the resolution. Note that in the proposed u-shaped network we use the down and up-convolutional layers to decrease and increase only the spatial direction of the given EPI.

The basic building block of the overall network is a convolutional layer followed by a Rectified Linear Unit (ReLU) non-linearity [118], $\sigma(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x})$. We combine two

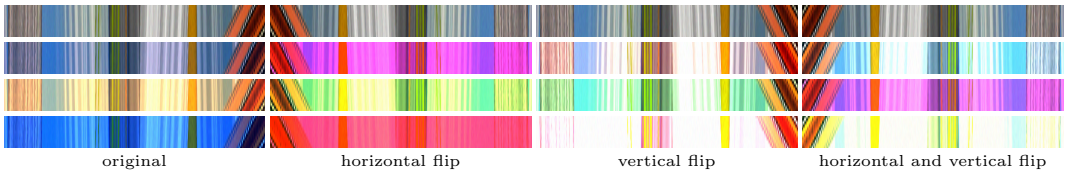


Figure 4.10: Illustration of the used data augmentation. The figure shows the original sample to the top left. The different columns represent horizontally and vertically flipped samples, and the different rows illustrate random brightness and color changes.

convolutional layers to one level. For the convolutional layers within one level, we use padding to compensate for the kernel size. This ensures that the output of one level has the same size as the input. In the first part of the network we use three of those levels, where we use down-convolutional layers after each level to increase the spatial support of subsequent layers. At each downsampling step we double the number of feature channels, except for the last level. The bottom part of the network consists of another level, that processes the compressed data. The decoding part of the network uses again three levels, but now we utilize up-convolutional layers before each level. Hence we up-convolve the whole coarse feature maps allowing to transfer high-level information to the fine prediction, and finally increase the image resolution back to the original size. All the involved convolutions use kernels of size 3×5, except for the down and up-convolutional layers that use 3×3 kernels. We also use so-called pinhole connections between the encoding and decoding part of the network, i.e. we concatenate the input of each level in the decoding part with the corresponding output feature map from the encoding path. We want to emphasis that the network structure involves only convolutional layers, i.e. we are not using any fully connected layers nor any pooling operations. A main advantage of avoiding fully connected layers is the ability to process EPIs of arbitrary resolutions.

Dataset. In order to train the proposed u-shape network a large amount of labeled training data is needed. Fortunately, we were allowed to use the synthetic dataset proposed in [77]. This dataset was generated using POV-Ray [POV-Ray] and comes with highly accurate ground truth depth fields. Moreover the dataset also provides a random scene generator that allows to generate the desired amount of LFs. We render 200 LFs with a spatial resolution of 640×480 and a directional resolution of 11×11, out of which 150 are used to generate training data and 50 are used for testing.

Data Augmentation. Data augmentation [56, 97] is a common way to combat overfitting and to improve the generalization of the trained model. It basically allows the model to become invariant to certain predefined image deformations. We perform excessive data augmentation, including brightness changes, color changes, and additive Gaussian noise. We also flip the EPIs horizontally and vertically, where each flipping results in a sign change of the disparity map. Our augmentation procedure results

in 8 times the original amount of image pairs. Although they are heavily correlated they allow to increase the robustness of the trained model. Figure 4.10 provides some augmentation examples.

Network Training. While Neural Networks (NNs) learned with back-propagation have been around for several decades [144], only recently the computational power and data has been available to fully exploit this training technique [97]. In order to train the proposed u-shaped network we use the tensorflow framework [2], where we use Adam [92] as the optimization method to minimize the ℓ_1 loss. Out of the 150 rendered LFs used for training we extract 20e3 EPIs. The extracted samples are then increased eightfold using data augmentation. To monitor overfitting we use a test set of 10e3 samples. In deep networks with many convolutional layers a good initialization of the weights is extremely important. Ideally the weights in the network should be initialized such that each feature map has approximately unit variance. This can be achieved by drawing the initial weights of a given node from a Gaussian distribution with standard deviation $\sqrt{2/n}$, where n denotes the number of incoming nodes [75]. After initializing the weights as suggested in [75] we train our model for 400 epochs, where we use a mini-batch size of 2^8 samples.

4.2.1.3 Experiments

We have performed an extensive analysis of our proposed model. We conducted synthetic and real world experiments. For the synthetic evaluation we used a recently presented LF dataset [77], where all LF scenes within the dataset have a directional resolution of 11×11, and a spatial resolution of 640×480. For the real world evaluation we used a LF captured with a Lytro camera as well as LFs from the Stanford Light Field Archive (SLFA). The used Lytro data provides a spatial resolution of 328×328 and a directional resolution of 7×7. LFs within the SLFA are captured using a multi-camera array [167] and contain 289 views on a 17×17 grid. We trained a u-shaped network based on the description in Section 4.2.1.2, where we use the same model for all the presented experiments. To obtain the final result we predict the horizontal and vertical EPIs and take the pointwise average of the two predictions.

We compare our model against the following state-of-the-art SfLF methods [76, 77, 88, 152, 159]. The method by Wanner and Goldluecke [159] analyzes the EPIs using the 2D structure tensor, before combining the obtained information using a variational framework. Tao et al. [152] proposed a fusion method that uses correspondence and defocus cues. Both local cues are combined to a global depth estimate by using a MRF model. Heber and Pock [76] proposed a variational multi-view stereo model based on low rank minimization. This model includes a matching term based on Robust Principal Component Analysis (RPCA), that can be interpreted as an all vs. all matching term. Jeon et al. [88] proposed an algorithm for SfLF, that utilizes phase shift based subpixel displacements. Besides the use of the phase shift theorem the

	Wanner [159]	Tao [152]	Heber [76]	Jeon [88]	Heber [77] (CNN)	proposed
RMSE	3.91	2.33	2.50	2.49	1.87	0.80
MAE	2.94	1.06	0.79	0.75	1.13	0.35
0.5%	22.00	16.32	8.47	9.64	17.96	7.34
0.2%	35.22	28.48	13.20	16.46	31.61	14.76
Time	3min 18s	23min 4s	4min 44s	2h 12min 30s	35s	2s
GPU	✓	✗	✓	✗	✓	✓

Table 4.3: Quantitative results for various Shape from Light Field methods averaged over 50 synthetic Light Fields. The table provides the Root Mean Squared Error, Mean Absolute Error, the percentage of pixels with a relative disparity error larger than 0.2% and 0.5%, and the computational time of the method.

algorithm is quite straightforward. They first calculate various cost volumes, that are processed using edge-preserving filtering, before extracting a disparity map based on the winner-takes-all strategy. To correct the obtained disparity map in weak textured regions they proceed with a multi-label optimization using graph cuts. At the end they refine the discrete disparity map to a continuous one using an iterative refinement scheme. In [77] Heber and Pock presented the first attempt to predict depth information for given LF data by utilizing deep learning strategies. Their network was trained in a sliding window setup to predict for each imaged scene point the orientation of the corresponding 2D hyperplane in the domain of the LF. This corresponds to estimating the line orientations in the horizontal and vertical EPIs simultaneously. They also use a 4D regularization step to overcome prediction errors in textureless or uniform regions, where they use a confidence measure to gauge the reliability of the estimate. This additional regularization step is not used in the following comparison, because such a post processing step can also be applied to the prediction of the proposed model. The method of Heber and Pock [77] works well but has drawbacks due to the sliding window scheme. First, the per-patch nature disallows to account for global output properties, and second, it leads to higher computational costs compared to the proposed approach. In what follows we will first provide some synthetic evaluations before presenting qualitative real world results.

Synthetic Evaluation. We start with the synthetic evaluation. Figure 4.11 provides a comparison of different state-of-the-art methods. Note that for all methods that rely on precomputed cost volumes [88, 152, 159], the number of labels is set to 200. Moreover we also set the necessary known disparity range for those methods based on the ground truth data. We can see that, despite the complexity of the scene, our model is able to predict accurate disparity results, that are on par with the competing methods. When comparing the results of the proposed model to the predictions obtained by the conventional CNN

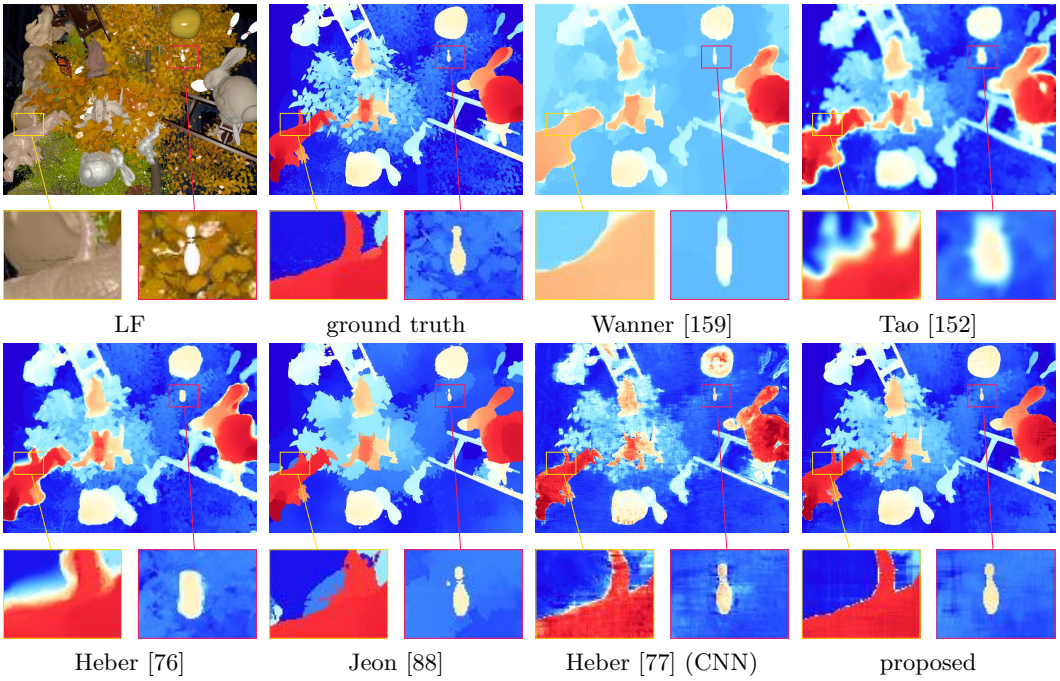


Figure 4.11: Comparison to state-of-the-art methods on the synthetic POV-Ray dataset. The figure shows the center view of the LF, the color-coded ground truth, the results for five state-of-the-art SfLF methods [76, 77, 88, 152, 159], followed by the result of the proposed method.

used in [77], we see that the proposed model provides better results in textureless regions. Also note that the proposed model is barely effected by depth discontinuities. Quantitative results in terms of Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are presented in Table 4.3. The table also shows the percentage of pixels with a relative disparity error larger than 0.2% and 0.5%. Besides the various disparity errors the table also provides the computation times for estimating a disparity map for one sub-aperture view of the LF. Moreover we also indication if a GPU implementation was used or not. The presented results represent the average over the 50 LFs used for testing. We observe that the proposed model was able to accurately learn the characteristics of this dataset. Furthermore, we also see that the proposed method is significantly better than all the competing methods in terms of the computation time. The presented method takes about 15 seconds to compute the disparity field for the entire LF (i.e. 121 views).

Real World Evaluation. We continue with the real world evaluation. Figure 4.12 provides a qualitative comparison to the methods by Tao et al. [152], Heber and Pock [76], and Jeon et al. [88]. To be able to compute results for the methods by Jeon et al. [88] and Tao et al. [152] in a reasonable time, it was necessary to reduce the directional resolution of the data to 11×11 and the number of labels to 75. The results show that although the proposed model was not trained on this dataset, nor have we performed

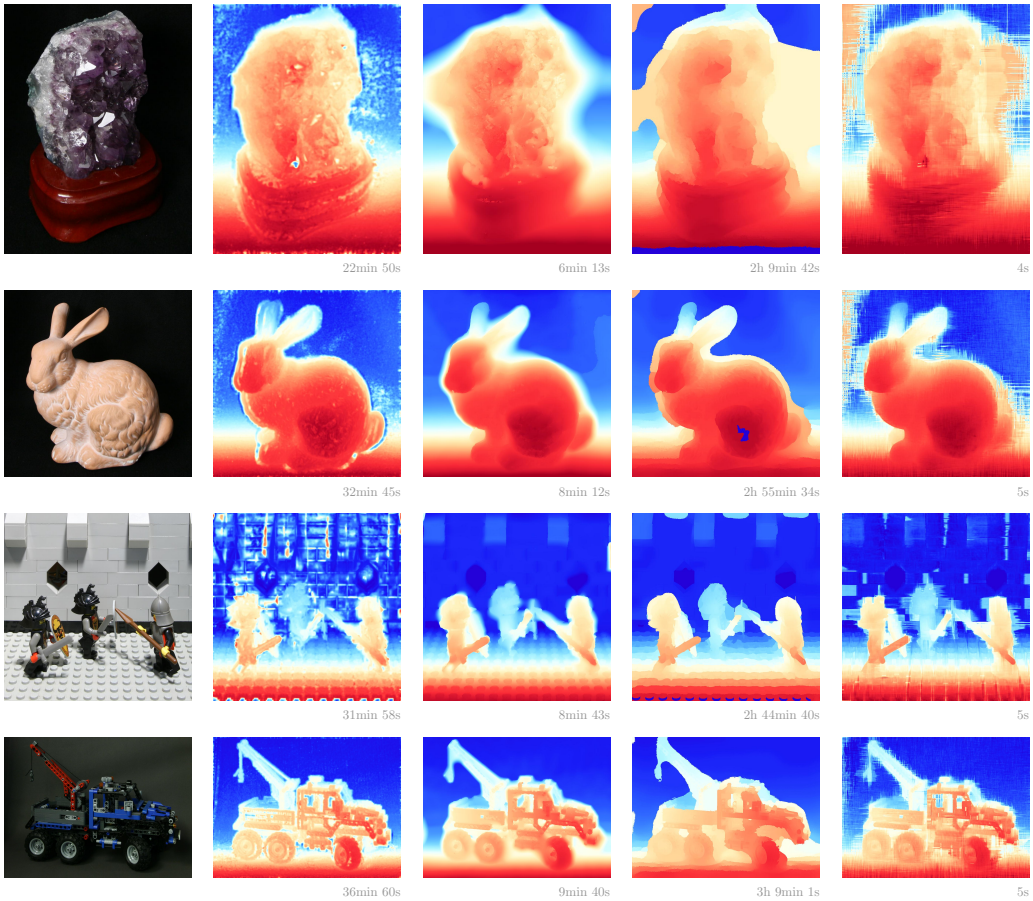


Figure 4.12: Qualitative comparison for Light Fields from the Stanford Light Field Archive. The figure shows from left to right the center view of the LF, followed by the results for the methods proposed by Tao et al. [152], Heber and Pock [76], and Jeon et al. [88]. The results to the right correspond to the proposed method.

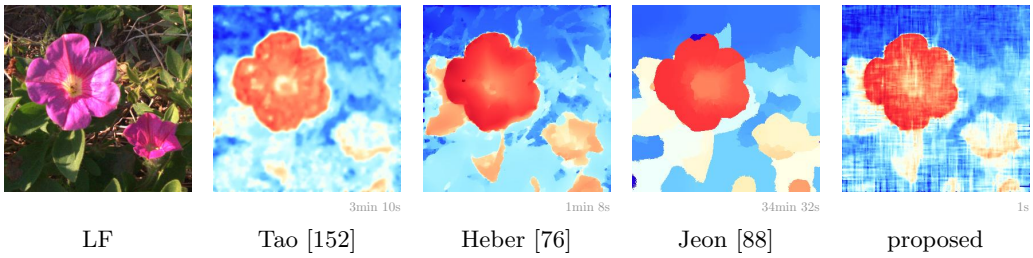


Figure 4.13: Qualitative comparison for a Light Field captured with a plenoptic camera. The figure shows from left to right the center view of the Light Field, followed by the results for the methods proposed by Tao et al. [152], Heber and Pock [76], and Jeon et al. [88]. The result to the right corresponds to the proposed method.

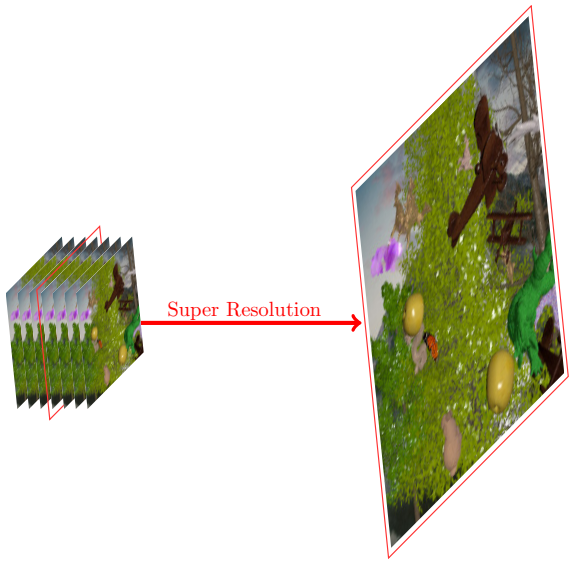


Figure 4.14: Illustration of Light Field Super Resolution. The figure shows to the left a set of low-resolution images, that are used as input, and to the right the high-resolution output image of the center view (marked in red).

any fine-tuning for this dataset, it allows to predict depth maps that are on par with the competing methods. However, the results are not perfect because the model produces streaking artifacts in homogeneous background regions. The main benefit of the proposed method is again the computational time of a few seconds. Also keep in mind that we are not using any post-processing, the results shown in the figure are the raw network predictions.

In Figure 4.13 we also present results for a LF captured with a Lytro camera. Note, that the Lytro data includes a significant amount of noise and outliers, for which the proposed u-shape network was not trained for. Nevertheless, the proposed model is able to predict a reasonable disparity field with clear depth discontinuities.

4.2.2 Light Field Superresolution

In this section we consider the problem of multi-view super-resolution, where the task is to reconstruct a high-resolution image based on a set of observed low-resolution input images taken from the same scene but from different viewpoints. The observed images are considered to be degraded observations of corresponding high-resolution ones. Thus the considered super-resolution task is aimed at increasing an image's apparent resolution, where existing details in correlated images are used to enhance the image.

The pixel correspondence problem and the super-resolution problem are interrelated in the sense that the result of one problem can help to solve the other one. Hence there exist various super-resolution approaches that rely on a pre-computed solution of the

correspondence problem. The proposed method avoids a dependency on pre-computed results by learning directly an end-to-end mapping between given low-resolution images and the desired high-resolution reconstruction result. For this purpose we adapted the RDM proposed in [40] and applied it to the target application of multi-view super-resolution. The main idea of RDMs is to learn a non-linear regularizer via supervised training.

We consider a special type of multi-view stereo data known as a LF. A LF is a densely sampled set of images captured from a regular grid of viewpoints located on a common 2D plane. It should be emphasized that the key difference to the general multi-view stereo setting is the dense and regular sampling of the viewpoints.

Recall that a 4D LF is commonly defined via the so-called two-plane parametrization. Let $\Omega \subseteq \mathbb{R}^2$ and $\Pi \subseteq \mathbb{R}^2$ be two parallel planes ($\Omega \neq \Pi$), then the LF is defined as

$$L : \Omega \times \Pi \to \mathbb{R}, \qquad (\mathbf{p}, \mathbf{q}) \mapsto L(\mathbf{p}, \mathbf{q}) \,, \tag{4.49}$$

where $\mathbf{p} = (x, y)^\top \in \Omega$ and $\mathbf{q} = (\xi, \eta)^\top \in \Pi$ represent spatial and directional coordinates. Note that Ω corresponds to the traditional image plane and Π is usually referred to as lens or focal plane. By choosing a certain directional coordinate η we can restrict the 4D LF to the 3D function

$$\Sigma_\eta : \Omega \times \Pi_\eta \to \mathbb{R}, \qquad (x, y, \xi) \mapsto L(x, y, \xi, \eta) \,, \tag{4.50}$$

where $\Pi_\eta = \{ \xi \,|\, (\xi, \eta) \in \Pi \}$. Note that Equation (4.50) defines a so-called EPI volume [21], which is an orthogonal 3D slice through the 4D LF.

There are different devices to capture LFs. The most promising device for this task is a LF or plenoptic camera [Lytro, Raytrix]. LF cameras came into the spotlight as interesting computational photography devices, that offer a new user experience for consumers because of features like digital refocusing capabilities [Lytro], i.e. the ability to adjust the focus settings after a single exposure. Moreover plenoptic cameras are also used in industrial applications like automated optical inspections and visual surface reconstruction [Raytrix]. A LF camera not only records spatial information, like a conventional camera, but it also records directional information. In order to capture the additional directional information a plenoptic camera leverages an array of micro-lenses placed in front of the image sensor. This leads to a trade-off between the spatial and the directional resolution, i.e. that angular samples being traded for spatial samples [66, 131]. One way to compensate the resulting loss of spatial resolution is the use of a higher resolution sensor (behind the micro-lens array), which obviously leads to higher data rates, a higher computation effort, more storage requirements, and higher costs. An attractive algorithmic solution to counteract the loss of spatial resolution, and simultaneously avoid the burden of a massive growth in data size and processing complexity is provided by various Light Field Super Resolution (LFSR) methods that

exploit the multi-view characteristic of the LF. Note that due to the fact that the correspondence problem and the super-resolution problem are highly interlinked, the proposed LFSR method has a high potential to improve various branches in the field of LF image processing.

4.2.2.1 Related Work

Super-resolution methods can be divided on the one hand into single-view and multi-view approaches, and on the other hand into learning based and variational approaches. Although there are super-resolution approaches for temporal image sequences, there is a paucity of super-resolution approaches that directly examine multi-view images. In this work we consider the task of LFSR where the related work is further limited. To the best of our knowledge, there exist only a few relevant publications.

Oberdoerster et al. [124] proposed a super-resolution approach for camera arrays by a custom design of the image sensor pixels. Heber and Pock [76] proposed a multi-view method for simultaneous depth estimation and superresolution based on low-rank techniques. A variational Bayesian framework was proposed in [19, 20] for restoration of high-resolution images from a LF given a predicted disparity map. Yoon et al. [175] adopted CNNs for LFSR, where they super-resolve sub-aperture images with the aid of neighboring sub-aperture images. Mitra and Veeraraghavan [115] designed a Gaussian mixture model for LF denoising, superresolution etc.. Wanner and Goldluecke [161] estimated disparity maps based on an EPI analysis and then applied a variational method to generate high-resolution images in both spatial and angular directions. Boominathan et al. [22] leveraged a hybrid imaging system, consisting of a high-resolution traditional camera and a LF camera and used a patch-based method for the task of LFSR.

Note that none of the above-mentioned methods tackled LFSR by using the entire information provided in the LF. In this section we present a method that goes into this direction. Our model is an adapted RDM [40], that is extended to train 3D convolutional filters. This extension allows us to tackle the problem of multi-view super-resolution. More specifically our model predicts a high-resolution image from the observed scene based on a given low-resolution EPI volume.

4.2.2.2 Methodology

This section presents a learning based method for LFSR. We train a novel 3D RDM, which can also be interpreted as a Recurrent Neural Network (RNN) (cf. Figure 4.17). The high-level interpretation of LFSR is illustrated in Figure 4.14, where the high-resolution image (shown on the right) is obtained by exploiting multiple low-resolution images (shown on the left). The principle of the 3D RDM is to efficiently learn a 3D regularizer based on supervised training.

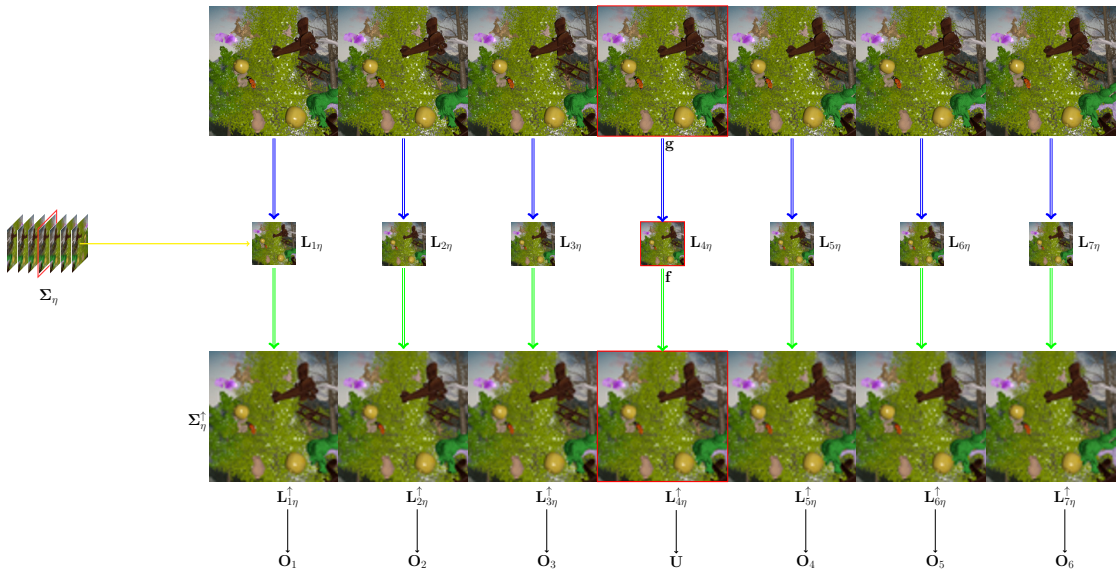


Figure 4.15: Illustration of the dataset generation. The figure shows the raw input on top and the Bicubic upsampled input data at the bottom. Blue arrows indicate downsampling operations and green arrows indicate upsampling operations.

Before presenting the proposed model we recall some background on RDMs and introduce several notations that are used throughout the section. First recall that the diffusion equation is a partial differential equation which describes the process of density (e.g.image intensity) changes. Let $\mathcal{U} : \mathbb{R}^n \times t \to \mathbb{R}$ be a function where $\mathbb{R}^n$ is the spatial domain and t is the time domain, then the diffusion equation is formulated as

$$\frac{\partial \mathcal{U}}{\partial t} = \operatorname{div}\left(\mathbf{\Gamma} \nabla \mathcal{U}\right), \tag{4.51}$$

where div is the divergence operator, and $\mathbf{\Gamma} \in \mathbb{R}^{n \times n}$ is called the diffusion tensor. In the case of isotropic diffusion, the diffusion tensor can be simplified as $\mathbf{\Gamma} = c\,\mathbf{I}$ where $c \in \mathbb{R}$ is called a diffusion coefficient, and $\mathbf{I}$ is the identity matrix. For anisotropic diffusion, the diffusion tensor is given by a dense symmetric matrix. Note that for isotropic diffusion, the propagation of diffusion in each direction is the same, cf. for instance the heat equation, whereas anisotropic diffusion propagates differently along different directions, cf. for example the Perona-Malik equation [165].

Chen and Pock [40] adapted Equation (4.51) to the following diffusion model in the discrete setting

$$\mathcal{U}_t = \mathcal{U}_{t-1} - \sum_{i \in [n_k]} \mathbf{k}_i^{t\top} * \boldsymbol{\phi}_i^t(\mathbf{k}_i^t * \mathcal{U}_{t-1}), \tag{4.52}$$

where $\mathcal{U}_t$ denotes an image at stage t, n_k[2] denotes the number of filters, $\mathbf{k}_i^t$ denotes a kernel

[2]Notation: $[n_k] := \{1, \ldots, n_k\}$

and $\boldsymbol{\phi}_i^t$ denotes a function representing a diffusion tensor, cf. $\boldsymbol{\Gamma}$ in Equation (4.51). One possible solution for 3D LFSR is to directly generalize the image $\mathcal{U}_t$ to an EPI volume and the 2D kernels $\mathbf{k}_i^t$ to their 3D counterparts. This would result in a model that outputs an entire EPI volume. However, our experiments showed that for the problem of LFSR, such a type of generalization is hard to train and generates undesired results. We assume that one reason is the complexity of the problem due to an exponential growth of variables. Moreover, due to the fact that the directional resolution in various LF capturing devices is usually quite low, boundary problems are somehow unavoidable. No matter which padding method is utilized, the large amount of errors from the convolution operation at the boundary results in an intricate problem.

Due to the fact the straight-forward generalization did not lead to satisfying results, we propose to split the complete training process into smaller sub-problems. Therefore the proposed method super-resolves only the center of the given low-resolution EPI volume. Note however that it is straightforward to adapt and train the proposed method to super-resolve any predefined sub-aperture image.

Let $\boldsymbol{\Sigma}_\eta$ be an EPI volume sampled with a spatial resolution of $m \times n$ and a directional resolution of d. Then the content of $\boldsymbol{\Sigma}_\eta$ can also be represented by listing the according sub-aperture images $\{\mathbf{L}_{\xi\eta}\}_{\xi\in[d]}$, where $\mathbf{L}_{\xi\eta} \in \mathbb{R}^{m\times n}$ denotes a sub-aperture image of the LF, i.e. a 2D slice of the LF where ξ and η are held constant. We apply bicubic spatial-upsampling to $\boldsymbol{\Sigma}_\eta \in \mathbb{R}^{m\times n\times d}$ and we denote the upsampled EPI volume as $\boldsymbol{\Sigma}_\eta^\uparrow \in \mathbb{R}^{mk\times nk\times d}$, where $k \in \mathbb{N}$ denotes the upsampling factor. Similar as before we can represent $\boldsymbol{\Sigma}_\eta^\uparrow$ as a list of upsampled sub-aperture images $\{\mathbf{L}_{\xi\eta}^\uparrow\}_{\xi\in[d]}$, that define the input of the proposed model. Compare Figure 4.15 for an illustration. Due to the fact that our model aims to predict the center view of the given EPI volume we will use a simplified notation. For this purpose we identify the upsampled sub-aperture images with the following list of images

$$\left\{\mathbf{O}_1, \ldots, \mathbf{O}_{\lfloor d/2\rfloor}, \mathbf{U}, \mathbf{O}_{\lceil d/2\rceil}, \ldots, \mathbf{O}_{(d-1)}\right\} \tag{4.53}$$

as shown in Figure 4.15. Next we define the vectorized representation of $\mathbf{O}_j$ and $\mathbf{U}$ as $\mathbf{o}_j := \mathrm{vec}(\mathbf{O}_j)$ and $\mathbf{u} := \mathrm{vec}(\mathbf{U})$, i.e. $\mathbf{o}_j, \mathbf{u} \in \mathbb{R}^{mnk^2}$. Given the low-resolution image $\mathbf{f}$ and the input images $\{\mathbf{o}_i, \mathbf{u}\}$, we aim to generate a high-resolution image $\mathbf{u}$, which is as close as possible to the ground truth $\mathbf{g}$. This is achieved by considering the model

$$\mathcal{E}(\mathbf{u}) = \lambda\mathcal{D}(\mathbf{u}, \mathbf{f}) + \mathcal{R}(\mathbf{u}), \tag{4.54}$$

where $\mathcal{D}$ and $\mathcal{R}$ denote the data term and the trained regularization term, respectively. λ is a trained weighting parameter, that allows to balance the influence of the data term w.r.t. the regularization. The learned RDM is obtained by considering the gradient of the energy functional $\mathcal{E}$ in Equation (4.54), where the final model is obtained as a series of gradient steps also called stages. For each stage, we learn the parameters of the model such that the recovered high-resolution image is as close as possible to the given ground truth.

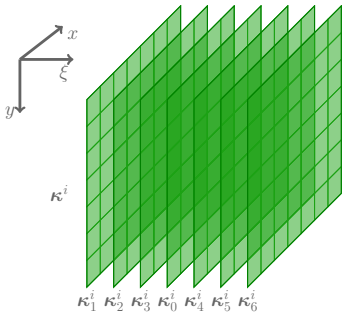


Figure 4.16: Illustration of the notation of a 3D kernel used within the trained regularizer (cf. (4.59)).

In what follows we will first provide details about the data term and the learned regularization term. Next we will describe the dataset used for supervised training. Finally we will outline the training procedure.

Data term. The data term enforces that the solution $\mathbf{u}$ is close to the observation $\mathbf{f}$. For simplicity we use the squared ℓ_2 norm to model the data term in Equation (4.54). The data term uses a downsampling operator $\mathbf{D}$ to compare a high-resolution image $\mathbf{u}$ to a low-resolution observation $\mathbf{f}$. Hence the data term can be written as

$$\mathcal{D}(\mathbf{u}) = \frac{1}{2}\|\mathbf{D}\mathbf{u} - \mathbf{f}\|_2^2 . \tag{4.55}$$

Due to the fact that we make use of the squared ℓ_2 norm, $\mathcal{D}$ is differentiable and the gradient is given as

$$\frac{\partial \mathcal{D}(\mathbf{u})}{\partial \mathbf{u}} = \mathbf{D}^\top (\mathbf{D}\mathbf{u} - \mathbf{f}) \ . \tag{4.56}$$

The gradient corresponds to the local reaction.

Regularization. Due to the fact that the given inverse problem is ill-posed, we make use of a trained regularizer $\mathcal{R}$ to constrain the space of possible high-resolution images. We formulate the regularizer as

$$\mathcal{R}(\mathbf{U}) = \sum_{i \in [n_k]} \boldsymbol{\rho}_i(\boldsymbol{\kappa}^i * \boldsymbol{\Sigma}^\uparrow_\eta) \, , \tag{4.57}$$

where $*$ denotes the 3D convolution operator, n_k is the number of penalty functions, $\boldsymbol{\rho}_i$ is the i^{th} penalty function and $\boldsymbol{\kappa}^i \in \mathbb{R}^{k_1 \times k_2 \times d}$ is the i^{th} 3D filter. Note that $\boldsymbol{\rho}_i$ and $\boldsymbol{\kappa}^i$ are the parameters to be learned. Using the notation presented in Equation (4.53) we can

rewrite Equation (4.57) as

$$\mathcal{R}(\mathbf{U}) = \sum_{i\in[n_k]} \boldsymbol{\rho}_i \left(\boldsymbol{\kappa}_0^i * \mathbf{U} + \sum_{j\in[d-1]} \boldsymbol{\kappa}_j^i * \mathbf{O}_j \right) , \tag{4.58}$$

where $\boldsymbol{\kappa}_j^i$ denotes a 2D slice through the 3D kernel $\boldsymbol{\kappa}^i$ at a sampling position of the third dimension (cf. Figure 4.16). More specifically, the kernel slices correspond to the images in Equation (4.53) in the following order

$$\left\{ \boldsymbol{\kappa}_1^i, \ldots, \boldsymbol{\kappa}_{\lfloor d/2 \rfloor}^i, \boldsymbol{\kappa}_0^i, \boldsymbol{\kappa}_{\lceil d/2 \rceil}^i, \ldots, \boldsymbol{\kappa}_{d-1}^i \right\} . \tag{4.59}$$

By rewriting the convolutions in Equation (4.58) using linear operators $\mathbf{K}_i$, $i \in \{0\} \cup [d-1]$, we obtain the final version of the regularizer as

$$\mathcal{R}(\mathbf{u}) = \sum_{i\in[n_k]} \boldsymbol{\rho}_i \left(\mathbf{K}_0^i \mathbf{u} + \sum_{j\in[d-1]} \mathbf{K}_j^i \mathbf{o}_j \right) , \tag{4.60}$$

where $\mathbf{K}_0^i \mathbf{u}$ is equal to the 2D convolution $\boldsymbol{\kappa}_0^i * \mathbf{U}$ and so forth. The gradient of $\mathcal{R}(\mathbf{u})$ w.r.t. $\mathbf{u}$ is then given as

$$\frac{\partial \mathcal{R}(\mathbf{u})}{\partial \mathbf{u}} = \sum_{i\in[n_k]} {\mathbf{K}_0^i}^\top \boldsymbol{\phi}_i \left(\mathbf{K}_0^i \mathbf{u} + \sum_{j\in[d-1]} \mathbf{K}_j^i \mathbf{o}_j \right) , \tag{4.61}$$

where $\boldsymbol{\phi}_i(\mathbf{x}) := \frac{\partial \boldsymbol{\rho}_i(\mathbf{x})}{\partial \mathbf{x}}$. The functions $\boldsymbol{\phi}_i$ are formulated as linear combinations of a set of Radial Basis Functions (RBFs) whose values depend on the distance from the origin. RBF methods [26, 59] and RBF networks [111, 128] have become crucial techniques for approximating multidimensional scattered data. In this section, we make use of so-called Gaussian RBFs, where we use n_k different influence functions. By combining Equation (4.56) and (4.61), we can summarize the proposed RDM as

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} = & -\lambda \mathbf{D}^\top (\mathbf{D}\mathbf{u} - \mathbf{f}) \\ & - \sum_{i\in[n_k]} {\mathbf{K}_0^i}^\top \boldsymbol{\phi}_i \left(\mathbf{K}_0^i \mathbf{u} + \sum_{j\in[d-1]} \mathbf{K}_j^i \mathbf{o}_j \right) . \end{aligned} \tag{4.62}$$

By introducing the superscript t referring to different stages, we obtain

$$\begin{aligned} \mathbf{u}^t = & \mathbf{u}^{t-1} - \lambda^t \mathbf{D}^\top \left(\mathbf{D}\mathbf{u}^{t-1} - \mathbf{f} \right) \\ & - \sum_{i\in[n_k]} {\mathbf{K}_0^{it}}^\top \boldsymbol{\phi}_i^t \left(\mathbf{K}_0^{it} \mathbf{u}^{t-1} + \sum_{j\in[d-1]} \mathbf{K}_j^{it} \mathbf{o}_j \right) . \end{aligned} \tag{4.63}$$

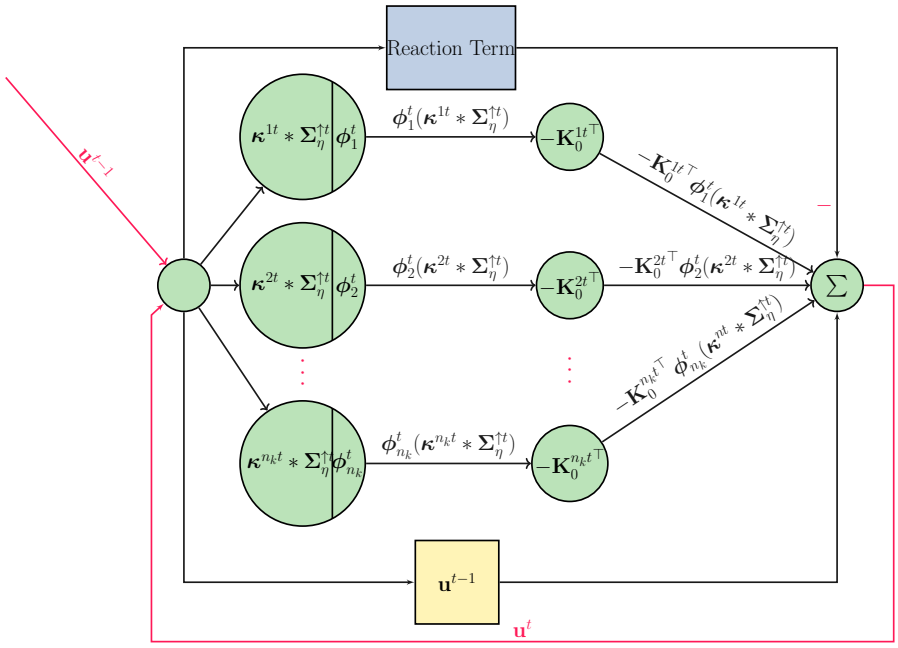


Figure 4.17: Illustration of the RNN interpretation of the proposed RDM.

For each stage t, we aim to learn the set of parameters $\Theta^t = \{\boldsymbol{\kappa}^{it}, \phi_i^t, \lambda^t\}$. Note that the overall model can be interpreted as a RNN as shown in Figure 4.17.

Dataset. In order to train the proposed model using supervised training, we first leverage the synthetic dataset proposed in [77]. For our current purpose we use the random scene generator to create 50 LFs with a spatial resolution of 480×639. For the test dataset, we use another 50 LFs from the POV-Ray dataset [77] with the same spatial resolution of 480×639. In addition, we also test on the HCI dataset [HCI Dataset]. For the real-world experiment, we use 11 LFs as training dataset and 2 LFs as test dataset from the SLFA [Stanford Light Field Archive]. The directional resolution of all LFs used in this section is 7×1.

We first use bicubic interpolation to downsample every sub-aperture image. The low-resolution image $\mathbf{f}$ (cf. Equation (4.55)) is the downsampled sub-aperture image of the center view. Next we generate the input images $\{\mathbf{o}_i, \mathbf{u}\}$ by upsampling the downsampled sub-aperture images again using bicubic interpolation.

Training. In this section we outline the learning approach. We use a PSNR [40] based learning approach and a SSIM [176] based learning approach separately to estimate the parameters $\Theta^t = \{\boldsymbol{\kappa}^{it}, \phi_i^t, \lambda^t\}$ for all stages $t \in [T]$. We start with a greedy training, where we learn the parameters at the stage t and then we use the optimal parameters to super-resolve the sub-aperture image of the center view $\mathbf{u}^t$. Note that we do not update

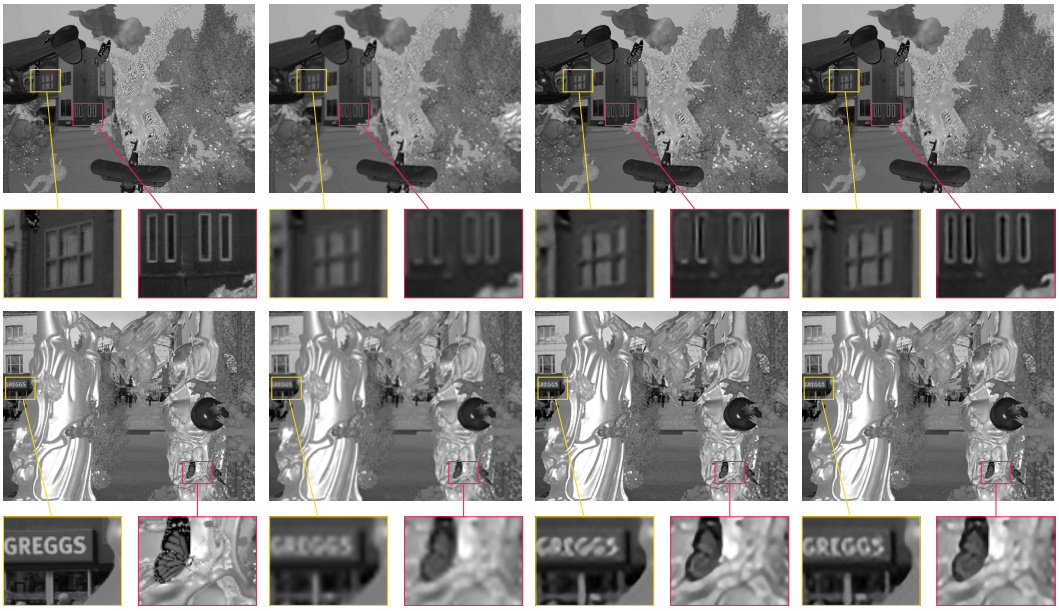


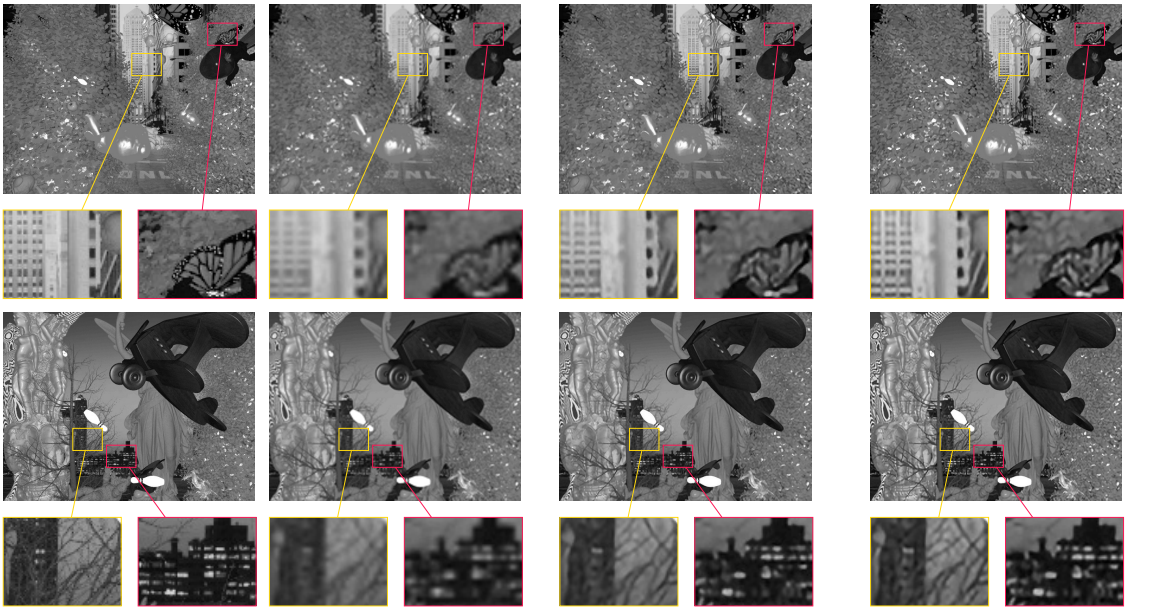
Figure 4.18: Illustration of the results I. From left to right, top to bottom, the figure shows the original high-resolution image, the bicubic upsampled image, the upsampled image by 2D RDM and the upsampled image by our trained model learned on PSNR.

$\mathbf{o}_j$, $j \in [d-1]$. We keep the original information of $\mathbf{o}_j$ which is exploited by the 3D kernels $\boldsymbol{\kappa}^{it}$, $i \in [n_k]$. We proceed by learning the parameters of the stage $t+1$, till we reach the maximum number of stages T. The result of the greedy training is used to initialize the model for the following joint training, where we train all stages simultaneously. The objective of the joint training is to only minimize the energy of stage T, which is associated with the final output. Let $\{\boldsymbol{\Sigma}_\eta^j, \mathbf{g}^j\}_{j\in[\mathrm{n_{trn}}]}$ denote the $\mathrm{n_{trn}}$ training samples, where $\boldsymbol{\Sigma}_\eta^j$ is the j^{th} low-resolution EPI volume with the center image $\mathbf{f}^j$, and $\mathbf{g}^j$ is the j^{th} ground truth high-resolution center image. Then the greedy training minimizes

$$\mathcal{L}(\Theta^t) = \sum_{j\in[\mathrm{n_{trn}}]} \ell^j(\Theta^t)\,, \tag{4.64}$$

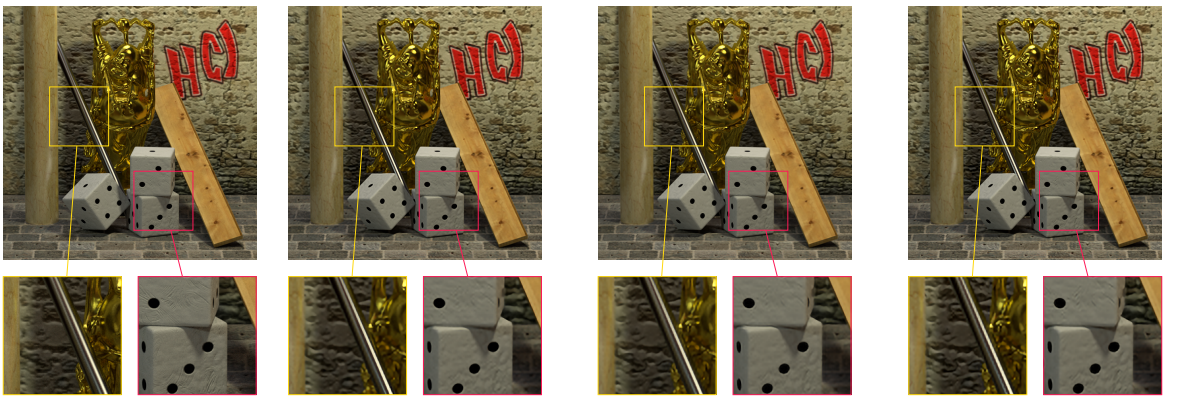
where $\ell^j(\Theta^t)$ indicates the loss of the j^{th} training sample measured either by PSNR or by SSIM when using parameters Θ^t. For simplicity we will omit the superscripts for the training samples in the following calculation. The PSNR loss of each sample is defined as

$$\ell(\mathbf{u}^t, \mathbf{g}) = \frac{1}{2}\|\mathbf{u}^t - \mathbf{g}\|_2^2\,, \tag{4.65}$$



High-resolution image | Bicubic upsampled | Our model trained on PSNR | Our model trained on SSIM

Figure 4.19: Illustration of the results II. From left to right, top to bottom, the figure shows the original high-resolution image, the bicubic upsampled image, and the upsampled images by our trained model learned on PSNR and on SSIM.



High-resolution image | 2D RDM (PSNR:30.083 SSIM:0.980) | Our model trained on PSNR (PSNR:30.487 SSIM:0.982) | Our model trained on SSIM (PSNR:30.063 SSIM:0.979)

Figure 4.20: Illustration of the results III. From left to right, top to bottom, the figure shows the original high-resolution image, the 2D RDM, and the upsampled images by our trained model trained on PSNR and on SSIM.

and the SSIM loss of each sample is defined as

$$\ell(\mathbf{u}^t, \mathbf{g}) = 1 - \mathcal{S}(\mathbf{u}^t, \mathbf{g}), \tag{4.66}$$

where $\mathcal{S}(\mathbf{u}^t, \mathbf{g})$ denotes the SSIM measure of $\mathbf{u}^t$ and $\mathbf{g}$ [158]. Because $\mathcal{S}(\mathbf{u}^t, \mathbf{g})$ reaches its maximum of one only if $\mathbf{u}^t = \mathbf{g}$, we reverse the sign of $\mathcal{S}(\mathbf{u}^t, \mathbf{g})$ to minimize the energy $\ell(\mathbf{u}^t, \mathbf{g})$. We will show in the experimental section that the RDM learned with SSIM loss allows to reconstruct sharper high-resolution results than the one learned with the PSNR based loss.

Recall Equation (4.63) which shows one step of gradient descent. To simplify notation we define the 3D convolution between the 3D kernel $\boldsymbol{\kappa}^i$ and the bicubic upsampled EPI volume $\boldsymbol{\Sigma}_\eta^\uparrow$ as

$$\mathbf{s}^t = \mathbf{K}_0^{it}\mathbf{u}^{t-1} + \sum_{j\in[d-1]} \mathbf{K}_j^{it}\mathbf{o}_j\,. \tag{4.67}$$

We will describe now how to calculate the gradient of $\mathbf{u}^t$ w.r.t. $\mathbf{K}_j^{it},\ j \in [d-1]$. All the remaining gradients can be easily derived based on the work presented in [40, 43, 176]. According to Equation (4.63), we obtain

$$\frac{\partial \mathbf{u}^t}{\partial \mathbf{K}_j^{it}} = \frac{\partial\left(\mathbf{K}_j^{it}\mathbf{o}_j\right)}{\partial \mathbf{K}_j^{it}} \frac{\partial \boldsymbol{\phi}_i^t\left(\mathbf{s}^t\right)}{\partial \mathbf{s}^t} \mathbf{K}_0^{it}. \tag{4.68}$$

Note that $\mathbf{K}_j^{it}$ is represented by the coefficients of n_b DCT basis kernels, i.e. $\mathbf{K}_j^{it} = \sum_{k\in[n_b]} \nu_k^{ijt}\mathbf{B}_k$[4]. Let $\boldsymbol{\nu}^{ijt} = [\nu_1^{ijt}\ \cdots\ \nu_{\mathrm{n_{bs}}}^{ijt}]^\top$, then we have

$$\frac{\partial\left(\mathbf{K}_j^{it}\mathbf{o}_j\right)}{\partial \boldsymbol{\nu}^{ijt}} = \begin{bmatrix} \mathbf{o}_j^\top\mathbf{B}_1^\top \\ \vdots \\ \mathbf{o}_j^\top{\mathbf{B}_{n_b}}^\top \end{bmatrix}. \tag{4.69}$$

By combining Equation (4.68) and (4.69) we obtain the gradient of $\mathbf{u}^t$ w.r.t. the parameters of the kernel $\mathbf{K}_j^{it}$ as

$$\frac{\partial \mathbf{u}^t}{\partial \boldsymbol{\nu}^{ijt}} = \begin{bmatrix} \mathbf{o}_j^\top\mathbf{B}_1^\top \\ \vdots \\ \mathbf{o}_j^\top{\mathbf{B}_{n_b}}^\top \end{bmatrix} \frac{\partial \boldsymbol{\phi}_i^t\left(\mathbf{s}^t\right)}{\partial \mathbf{s}^t} \mathbf{K}_0^{it}. \tag{4.70}$$

[4] $\mathbf{B}_k\mathbf{g} = \mathbf{b}_k * \mathbf{g}$, where $\mathbf{b}_k$ is the k^{th} DCT basis.

		PSNR		SSIM
	Bicubic	2D RDM	3D RDM	3D RDM
PSNR	26.316	27.384	27.747	27.486
SSIM	0.883	0.925	0.933	0.933

Table 4.4: Quantitative results for different super-resolution methods based on the POV-Ray dataset. The table proves on the one hand a comparison between 2D and 3D RDMs, and on the other hand it also shows the difference between PSNR and SSIM based training.

4.2.2.3 Experiments

In all experiments we use a magnification factor of 3. In [40], Chen and Pock investigated the influence of the model capacity including the number of training samples, filter size etc.. We stick to the model capacity where 2D RDM achieved the best result for super-resolution [40]. We compare with the corresponding 2D RDM [40] with the same model capacity except that we exploit the directional information. We use 48 influence functions combined with 48 3D kernels of size $7 \times 7 \times 7$. The weighting parameter λ^t is initialized to zero. Following the work of [40], we initialize the kernels as the DCT basis kernels and the influence function as $c_1 \frac{x}{1+x^2},\ c_1 \in \mathbb{R}^+$, whose integral has the form of $\frac{1}{2} c_1 \log(1 + x^2) + c_3,\ \ c_3 \in \mathbb{R}$. For all the RDMs, we first conduct a greedy training with 5 stages followed by a joint training.

Synthetic Experiment. We run Limited-memory BFGS (L-BFGS) with a fixed number of 250 iterations in the greedy training and also with 250 iterations in the joint training. First we conduct a comparison with the corresponding 2D RDM [40]. In addition, we compare 3D RDMs based on the PSNR learning approach with the one based on the SSIM learning approach. We train both the 2D RDM and the 3D RDMs using the same model capacity and the same training dataset consisting of 50 LF images.

We test the trained 2D and 3D models on a test dataset consisting of 50 LF images and we evaluated the results based on the PSNR and the SSIM. The average PSNR and SSIM results are presented in Table 4.4. The PSNR based evaluation shows that the proposed 3D RDM is able to improve upon the 2D RDM. As a baseline we also list the result of bicubic upsampling. We also observe that PSNR based training achieves better PSNR than SSIM based training. The 3D RDMs are able to outperform the 2D RDM, but there is no quantitative difference in SSIM between PSNR and SSIM based training in this case. It was shown in [175], that methods based on pre-computed depth-maps [115, 161] perform worse than bicubic upsampling. Because of this we did not incorporate those methods in our evaluation.

Figure 4.18 provides qualitative results, where we compare our model to the baseline of bicubic upsampling and to the 2D RDM. We can observe a clear qualitative improvement

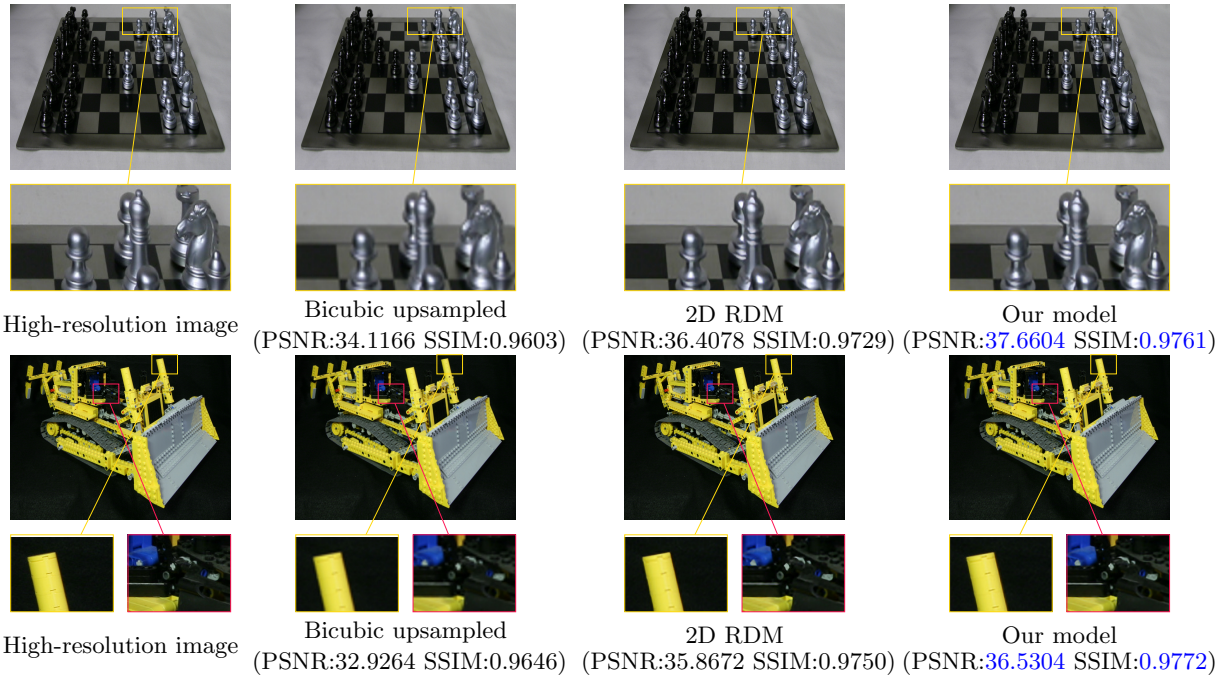


Figure 4.21: Illustration of the real-world results. From left to right, top to bottom, the figure shows the original high-resolution image, the bicubic upsampled image, the upsampled image by 2D RDM and the upsampled image by our trained model learned on SSIM. The close-up images show that the proposed model can reconstruct a high-resolution scene preserving more details.

in the results provided by our model. When considering the close-up views, we recognize that the 2D model is hallucinating additional details, that do not correspond to real scene details. The 3D model on the other hand is able to reconstruct real details from the observed scene.

Figure 4.19 compares the 3D RDM trained based on the PSNR measure with the one based on the SSIM measure. Here we can observe that the 3D RDM trained based on the SSIM provides slightly sharper results.

Next we show a result of the HCI LF dataset. Figure 4.20 shows the superresolved images by the 2D RDM, the 3D RDM learned with PSNR loss and the 3D RDM learned with SSIM loss. In this experiment, we apply the above-mentioned models to each channel of the RGB images. The 3D RDM learned with PSNR reaches the highest PSNR value of 30.487. The PSNR value of the 2D RDM is on par with the 3D RDM learned with SSIM loss. Although the SSIM values of the 2D RDM and 3D RDM are quite similar, we do observe more details in the superresolved image provided by the 3D RDM learned with SSIM loss (cf. scratches on the dices).

Real-world Experiment. We show real-world results from the SLFA, to demonstrate the applicability of our method to real-world data. In the SLFA, there are 13 LFs and we

divided this dataset into a training and a test dataset including 11 and 2 LFs, respectively. Note, that our trained 3D RDM has the same settings as the 2D RDM for single image super-resolution in [6]. Since the 2D RDM in [6] was trained on real-world images and achieved the best result without overfitting in this experiment, we compare with the 2D RDM published online[3]. The two LFs in the test dataset are called 'Chess' and 'Lego Bulldozer'. As shown by Figure 4.21, 3D RDM outperforms 2D RDM by a convincing margin. And from the close-up images we can observe more details from the results of 3D RDM than the one of 2D RDM.

This section presented a 3D RDM for LFSR. The RDM can be seen as a RNN. If we neglect the reaction term, then one stage of the RDM is composed of a convolutional layer followed by a trainable non-linearity. The reason of super-resolution of one sub-aperture image is that it provides better optimal solutions without using a large dataset in the training. Thus, super-resolution of EPI volume can combine several such models to avoid the curse of dimensionality in the training.

We investigated the use of 3D RDMs to restore a high-resolution image based on a set of observed low-resolution sub-aperture images. Single image super-resolution approaches rely on hallucinating additional information learned from the training set, whereas the proposed LFSR approach allows to extract information from the different views in order to reconstruct real details from the scene.

In our experimental section we demonstrated that the proposed 3D RDMs can improve upon 2D RDMs. Our experimental results showed quantitative and qualitative improvements based on synthetic and real-world datasets.

Finally it should be mentioned that the proposed model is applicable to a large range of LF image processing problems like e.g. LF denoising or LF inpainting. Furthermore it also provides an attractive opportunity for data size reduction for multi-view systems.

[3] `http://www.icg.tugraz.at/Members/Chenyunjin/about-yunjin-chen`

5 Summary and Conclusion

5.1 Conclusion

In this thesis we studied Machine Learning (ML) methods for Computer Vision (CV) problems. We first briefly reviewed the background of ML and CV. In the part of ML, we introduced the evolution of ML and the Convolutional Neural Network (CNN), one of the most important ML models. In the part of optimization, we reviewed first-order methods and second-order methods, separately. We discussed the heuristics for the Primal-Dual Algorithm (PDCP) and based on this, we presented the Online PDCP. In addition, we described the Reaction-Diffusion Model (RDM) and the Fields of Experts (FoE). In Section 4, we showed that the RDM, Recurrent Neural Network (RNN) and FoE share common properties.

Some classical ML algorithms, e.g.SVMs, or dimension reduction, are able to tackle large-scale real-world problems. Therefore, they are frequently used in different fields, from medicine to industry. It is crucial to solve those ML problems in an efficient and accurate way. We picked up several common ML problems, like dimensionality reduction, SVMs, feature selection, multi-task learning, matrix completion and matrix factorization. We illustrated how to solve the above-mentioned ML problems using first-order methods. We presented a detailed derivation for each problem and provided a comprehensive comparison among the first-order methods. Our results show the advantage to solve ML problems with primal-dual models because PDCP can stably converge to the optimal solution. Moreover, in some cases, PDCP even attains an empirical convergence rate which is better than $O(1/n^2)$.

Section 4 of this thesis was devoted to the applications of ML to CV problems, where we distinguished between single-view image processing and Light Field (LF) image processing. In single-view image processing, we proposed a trained diffusion model for image inpainting, where we learned the filters and influence functions based on the Structural Similarity Image Measure (SSIM). The trained diffusion models

were optimized by a greedy training followed by a joint training. We compared our method with state-of-the-art inpainting methods for the task of inpainting 80% and 90% random missing pixels and for the task of inpainting small connected image regions. The comparison showed our method provides competitive performance. Next, we trained our models to inpaint a specific texture. As a PDE based inpainting method, our models surprisingly fulfill also this inpainting task.

In the LF processing, we first considered the problem of Shape from Light Field (SfLF). We trained a novel u-shape CNN comprising an encoding and a decoding part. By comparing to other methods, we showed that our method can predict depth with a similar precision but more efficient. Then we demonstrated LF superresolution. We trained a 3D RDM to recover high-resolution center views of the LF. We evaluated the results provided by a 2D RDM and a 3D RDM. The evaluation showed that the 3D RDM is superior to the 2D RDM.

5.2 Direction to Future Work

In this thesis, we discussed ML methods in CV with a strong focus on optimization strategies. We also demonstrated several models applied to CV problems. Our models are efficient and effective, but there are still interesting directions that can be explored in future work. First, one could consider to use different second-order methods to train or fine-tune a ML model. Second, regarding LF image processing we only used a 2D or 3D subspace of the entire 4D LF domain as input to our models. Hence one could consider different strategies to utilize the entire 4D information of the LF, e.g. one could generalize the presented RDM to 4D . Finally, one could also consider expanding our models to so-called Hybrid Imaging Systems [22] consisting of a plenoptic camera and a traditional camera.

A

List of Acronyms

AI	Artificial Intelligence. vii
AWS	Active Wavefront Sampling. 104
BFGS	Broyden-Fletcher-Goldfarb-Shanno algorithm. 44–46
CNN	Convolutional Neural Network. viii, xvii, 12, 22, 23, 104, 105, 108, 109, 113, 125, 126
CV	Computer Vision. vii, 85, 125, 126
DCT	Discrete Cosine Transform. 96, 121
DNN	Deep Neural Network. vii
EPI	Epipolar Plane Image. viii, 103–108, 112, 113, 115, 119, 121, 123
FBS	Forward-backward Splitting Method. vii, 36–38, 44, 47, 49–51, 63, 67, 70, 75, 80, 81
FISTA	Fast Iterative Shrinkage-thresholding Algorithm. vii, xvii, xviii, 36, 38, 39, 44, 47, 49–53, 55–59, 61–68, 70, 71, 73–75, 78–81
FoE	Fields of Experts. 25, 26, 84, 86, 87, 96, 125
GPU	Graphics Processing Unit. 17
GSIM	Gradient Similarity Image Measure. 96–98

ISTA Iterative Shrinkage-thresholding Algorithm. 38

L-BFGS Limited-memory BFGS. 44, 46, 122
LF Light Field. viii, xix, xxi, 5, 6, 101–104, 106–113, 115, 118, 121–123, 125, 126
LFSR Light Field Super Resolution. vii, viii, 6, 112–114, 123

MAE Mean Absolute Error. 109
ML Machine Learning. vii, xxi, 3–5, 12, 17, 18, 22, 26, 35, 37, 46–51, 54, 74, 80, 81, 125, 126
MRF Markov Random Field. 86, 96, 107

NN Neural Network. xvii, 5, 18–22, 107

OSGA Optimal Subgradient Algorithm. vii, 36, 39, 40, 44, 47, 49–53, 56, 58, 62, 65, 67, 73, 78, 80, 81

PCA Principal Component Analysis. 69
PDCP Primal-Dual Algorithm. vii, xvii, xviii, 36, 37, 42–44, 47, 49–53, 55, 56, 58, 61, 62, 64–68, 71–73, 76–81, 125
PDE Partial Differential Equation. 85, 86, 98, 100
PoE Product of Experts. 25
PSNR Peak Signal to Noise Ratio. xxi, 85, 89, 96–98, 118–120, 122

RBF Radial Basis Function. 117
RDM Reaction-Diffusion Model. vii, viii, xxi, 6, 83–85, 87, 112–115, 117–120, 122, 123, 125, 126
ReLU Rectified Linear Unit. 21, 105
RMSE Root Mean Squared Error. 109
RNN Recurrent Neural Network. vii, xix, 114, 118, 125
RNNs Recurrent Neural Networks. 22
ROF Rudin Osher Fatemi. 86
RPCA Robust Principal Component Analysis. 107

SfLF	Shape from Light Field. vii, xxi, 103, 104, 107–109, 126
SLFA	Stanford Light Field Archive. xix, 107, 110
SR	Super Resolution. xvii, 14–16, 18
SSIM	Structural Similarity Image Measure. viii, xxi, 85, 89, 90, 96–98, 118, 119, 122, 125
SVM	Support Vector Machine. vii, 5, 33, 48
TV	Total Variation. 86

B

List of Publications

My work at the Institute for Computer Graphics and Vision led to the following peer-reviewed publications. For the sake of completeness of this Thesis, they are listed in chronological order along with the respective abstracts.

B.1 2014

A Comparison of First-order Algorithms for Machine Learning

Yu Wei and Pock Thomas
In: *Proceedings of The 38th Annual Workshop of the Austrian Association for Pattern Recognition (OAGM), 2014*
IST Austria

Abstract: Using an optimization algorithm to solve a machine learning problem is one of mainstreams in the field of science. In this work, we demonstrate a comprehensive comparison of some state-of-the-art first-order optimization algorithms for convex optimization problems in machine learning. We concentrate on several smooth and non-smooth machine learning problems with a loss function plus a regularizer. The overall experimental results show the superiority of primal-dual algorithms in solving a machine learning problem from the perspectives of the ease to construct, running time and accuracy.

B.2 2015

On learning optimized reaction diffusion processes for effective image restoration

Yunjin Chen, Wei Yu and Thomas Pock
In: *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*
June 2015, Boston, USA

Abstract: For several decades, image restoration remains an active research topic in low-level computer vision and hence new approaches are constantly emerging. However, many recently proposed algorithms achieve state-of-the-art performance only at the expense of very high computation time, which clearly limits their practical relevance. In this work, we propose a simple but effective approach with both high computational efficiency and high restoration quality. We extend conventional nonlinear reaction diffusion models by several parametrized linear filters as well as several parametrized influence functions. We propose to train the parameters of the filters and the influence functions through a loss based approach. Experiments show that our trained nonlinear reaction diffusion models largely benefit from the training of the parameters and finally lead to the best reported performance on common test datasets for image restoration. Due to their structural simplicity, our trained models are highly efficient and are also well-suited for parallel computation on GPUs.

My responsibility: *The influence functions can be formulated as basis functions, for instance, Gaussian radial basis or triangular-shaped basis functions. For triangular-shaped basis functions, we used the technique of slice transform to represent influence functions. It is space and time consumable to calculate the gradient w.r.t. influence functions represented by triangular-shaped basis functions. I used linear algebra to simplify this process. Thus, the time complexity reduced from $O(kn)$ to $O(n)$, where n is the number of pixels in one image and k is the number of the bins shaping the influence function. The time complexity when using Gaussian radial basis functions is $O(dn)$ where d is the number of Gaussian radial basis functions. This reduces the time of training and testing dramatically. And the results in the application of image denoising is on par with the one trained by models using Gaussian radial basis functions.*

Learning Reaction-Diffusion Models for Image Inpainting

Wei Yu, Stefan Heber and Thomas Pock
In: *37th German Conference on Pattern Recognition, GCPR 2015*
October 2015, Aachen, Germany

Abstract: In this paper we present a trained diffusion model for image inpainting based on the structural similarity measure. The proposed diffusion model uses several parametrized linear filters and influence functions. Those parameters are learned in a loss based approach, where we first perform a greedy training before conducting a joint training to further improve the inpainting performance. We provide a detailed comparison to state-of-the-art inpainting algorithms based on the TUM-image inpainting database. The experimental results show that the proposed diffusion model is efficient and achieves superior performance. Moreover, we also demonstrate that the proposed method has a texture preserving property, that makes it stand out from previous PDE based methods.

B.3 2016

U-shaped Networks for Shape from Light Field

Stefan Heber, Wei Yu and Thomas Pock
In: *Proceedings of the British Machine Vision Conference (BMVC)*
September 2016, York, UK

Abstract: This paper presents a novel technique for Shape from Light Field (SfLF), that utilizes deep learning strategies. Our model is based on a fully convolutional network, that involves two symmetric parts, an encoding and a decoding part, leading to a u-shaped network architecture. By leveraging a recently proposed Light Field (LF) dataset, we are able to effectively train our model using supervised training. To process an entire LF we split the LF data into the corresponding Epipolar Plane Image (EPI) representation and predict each EPI separately. This strategy provides good reconstruction results combined with a fast prediction time. In the experimental section we compare our method to the state of the art. The method performs well in terms of depth accuracy, and is able to outperform competing methods in terms of prediction time by a large margin.

My responsibility: *I performed some preliminary experiments that showed unsatisfactorily streaking artefacts in 3D Light Field Super Resolution (LFSR). Because of this observation we changed our planes, and started to tackle the task of shape estimation based on LF data. For this task the streaking artefacts are not a big problem. I helped to implement the model using the tensorflow framework. I was monitoring the training process and evaluated the models. The u-shaped models performed surprisingly well on this task.*

B.4 2017

Reaction-Diffusion Models for Light Field Super-Resolution

Wei Yu, Stefan Heber and Thomas Pock
Submitted to ICCV 2017

Abstract: This paper considers the problem of multi-view superresolution in the Light Field (LF) setting. We introduce a novel Light Field Super Resolution (LFSR) method that is based on Reaction-Diffusion Models (RDMs). The main idea of RDMs is to learn a non-linear regularizer via supervised training. The proposed LFSR model allows to reconstruct a high-resolution image given a set of observed low-resolution sub-aperture images. More specifically the proposed method examines 3D subsets of a 4D LF called Epipolar Plane Image (EPI) volumes and generates a highresolution image of the observed scene. An important aspect of our model is the use of 3D convolutions, that allow to propagate information from two spatial and one directional dimension of the LF. Our experimental results show that the proposed method can achieve a far higher restoration quality than competing methods. Furthermore our quantitative experiments demonstrate a significant gain in PSNR.

Bibliography

[url] The tum-image inpainting database. (page)

[2] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. (page)

[3] Ahmadi, A. A., Olshevsky, A., Parrilo, P. A., and Tsitsiklis, J. N. (2013). Np-hardness of deciding convexity of quartic polynomials and related problems. *Mathematical Programming*, 137(1-2):453–476. (page)

[4] Ahookhosh, M. (2014). Optimal subgradient algorithms with application to large-scale linear inverse problems. Unpublished manuscript. (page)

[5] Alexander, D. C., Zikic, D., Zhang, J., Zhang, H., and Criminisi, A. (2014). Image quality transfer via random forest regression: applications in diffusion mri. In *MICCAI*, pages 225–232. Springer. (page)

[6] Alpaydin, E. (2004). Introduction to machine learning (adaptive computation and machine learning series). (page)

[Amazon Machine Learning] Amazon Machine Learning. Amazon machine learning. `https://aws.amazon.com/machine-learning/`. (page)

[8] Avron, H., Kale, S., Kasiviswanathan, S. P., and Sindhwani, V. (2012). Efficient and practical stochastic subgradient descent for nuclear norm regularization. In *ICML*. icml.cc / Omnipress. (page)

[Azure Machine Learning] Azure Machine Learning. Azure machine learning. `https://azure.microsoft.com/en-us/services/machine-learning/`. (page)

[10] Bauschke, H. and Combettes, Patrick, L. (2011). *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer. (page)

[11] Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202. (page)

[12] Beck, A. and Teboulle, M. (2012). Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22(2):557–580. (page)

[13] Belkin, M., Niyogi, P., and Sindhwani, V. (2005). On manifold regularization. In *AISTATS*. Citeseer. (page)

[14] Ben-Hur, A. and Weston, J. (2010). A user's guide to support vector machines. In Carugo, O. and Eisenhaber, F., editors, *Data Mining Techniques for the Life Sciences*, volume 609 of *Methods in Molecular Biology*, pages 223–239. Humana Press. (page)

[15] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *PAMI*, 35(8):1798–1828. (page)

[16] Bennett, K. P. and Parrado-Hernández, E. (2006). The interplay of optimization and machine learning research. *J. Mach. Learn. Res.*, 7:1265–1281. (page)

[17] Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. (2000). Image inpainting. In *Conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co. (page)

[18] Bertalmio, M., Vese, L., Sapiro, G., and Osher, S. (2003). Simultaneous structure and texture image inpainting. In *Computer Vision and Pattern Recognition, 2003*, volume 2, pages II–707–12 vol.2. (page)

[19] Bishop, T. E. and Favaro, P. (2012). The light field camera: Extended depth of field, aliasing, and superresolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):972–986. (page)

[20] Bishop, T. E., Zanetti, S., and Favaro, P. (2009). Light field superresolution. In *Computational Photography (ICCP), 2009 IEEE International Conference on*, pages 1–9. IEEE. (page)

[21] Bolles, R. C., Baker, H. H., and Marimont, D. H. (1987). Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1(1):7–55. (page)

[22] Boominathan, V., Mitra, K., and Veeraraghavan, A. (2014). Improving resolution and depth-of-field of light field cameras using a hybrid imaging system. In *Computational Photography (ICCP), 2014 IEEE International Conference on*, pages 1–10. IEEE. (page)

[23] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA. (page)

[24] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140. (page)

[25] Bugeau, A., Bertalmío, M., Caselles, V., and Sapiro, G. (2010). A comprehensive framework for image inpainting. *Image Processing, IEEE Transactions on*, 19(10):2634–2645. (page)

[26] Buhmann, M. D. (2003). *Radial Basis Functions*. Cambridge University Press, New York, NY, USA. (page)

[27] Burges, C. J. C. (2010). Dimension reduction: A guided tour. *Foundations and Trends in Machine Learning*, 2(4). (page)

[28] Cai, J.-F., Candès, E. J., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM J. on Optimization*, 20(4):1956–1982. (page)

[29] Candes, E. J., Romberg, J. K., and Tao, T. (2006). Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223. (page)

[30] Caselles, V. (2011). Exemplar-based image inpainting and applications. *SIAM News*, 44(10):1–3. (page)

[31] Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145. (page)

[32] Chan, T. F. and Shen, J. (2001a). Local inpainting models and TV inpainting. *SIAM J. Appl. Math.*, 62(3):1019–1043. (page)

[33] Chan, T. F. and Shen, J. (2001b). Nontexture inpainting by curvature-driven diffusions. *Journal of Visual Communication and Image Representation*, 12(4):436–449. (page)

[34] Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27. (page)

[35] Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178. (page)

[36] Chapelle, O., Schölkopf, B., Zien, A., et al. (2006). Semi-supervised learning. (page)

[37] Charbonnier, P., Blanc-Féraud, L., Aubert, G., and Barlaud, M. (1997). Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing*, 6(2):298–311. (page)

[38] Chen, C., Lin, H., Yu, Z., Bing Kang, S., and Yu, J. (2014a). Light field stereo matching using bilateral statistics of surface cameras. (page)

[39] Chen, G. H.-G. and Rockafellar, R. T. (1997). Convergence rates in forward–backward splitting. *SIAM J. on Optimization*, 7(2):421–444. (page)

[40] Chen, Y. and Pock, T. (2016). Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1. (page)

[41] Chen, Y., Ranftl, R., and Pock, T. (2014b). A bi-level view of inpainting-based image compression. *arXiv preprint arXiv:1401.4112*. (page)

[42] Chen, Y., Ranftl, R., and Pock, T. (2014c). Insights into analysis operator learning: From patch-based sparse models to higher order mrfs. *Image Processing, IEEE Transactions on*, 23(3):1060–1072. (page)

[43] Chen, Y., Yu, W., and Pock, T. (2015). On learning optimized reaction diffusion processes for effective image restoration. In *CVPR*, pages 5261–5269. (page)

[44] Cherkassky, V. and Ma, Y. (2004). Practical selection of svm parameters and noise estimation for svm regression. *Neural Networks*, 17:113–126. (page)

[45] Coates, A. and Ng, A. Y. (2012). Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, pages 561–580. Springer. (page)

[46] Coates, A., Ng, A. Y., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223. (page)

[47] Criminisi, A., Pérez, P., and Toyama, K. (2004). Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200–1212. (page)

[48] Criminisi, A., Robertson, D., Konukoglu, E., Shotton, J., Pathak, S., White, S., and Siddiqui, K. (2013). Regression forests for efficient anatomy detection and localization in computed tomography scans. *Medical image analysis*, 17(8):1293–1303. (page)

[49] De Bie, T., Lanckriet, G. R. G., and Cristianini, N. (2003). Convex tuning of the soft margin parameter. Technical Report UCB/CSD-03-1289, EECS Department, University of California, Berkeley. (page)

[50] Domingos, P. (2012). A few useful things to know about machine learning. *ACMCOM*, 55(10):78–87. (page)

[51] Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306. (page)

[52] Dosovitskiy, A., Springenberg, J. T., and Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1538–1546. (page)

[53] Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 272–279, New York, NY, USA. ACM. (page)

[54] Duchi, J. and Singer, Y. (2009). Efficient online and batch learning using forward backward splitting. *J. Mach. Learn. Res.*, 10:2899–2934. (page)

[55] Efros, A. and Leung, T. (1999). Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038 vol.2. (page)

[56] Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 2366–2374. Curran Associates, Inc. (page)

[57] Esedoglu, S. and Shen, J. (2002). Digital inpainting based on the Mumford-Shah-Euler image model. *European Journal of Applied Mathematics*, null:353–370. (page)

[58] Esser, E., Zhang, X., and Chan, T. F. (2010). A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM J. Img. Sci.*, 3(4):1015–1046. (page)

[59] Figueiredo, M. A. (2000). On gaussian radial basis function approximations: Interpretation, extensions, and learning strategies. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 618–621. IEEE. (page)

[60] Fodor, I. (2002). A survey of dimension reduction techniques. (page)

[61] Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612. (page)

[62] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139. (page)

[63] Fujino, A., Ueda, N., and Saito, K. (2005). A hybrid generative/discriminative approach to semi-supervised classifier design. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 764. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. (page)

[64] Galić, I., Weickert, J., Welk, M., Bruhn, A., Belyaev, A., and Seidel, H.-P. (2008). Image compression with anisotropic diffusion. *Journal of Mathematical Imaging and Vision*, 31(2-3):255–269. (page)

[65] Gao, J., Shi, Q., and Caetano, T. S. (2012). Dimensionality reduction via compressive sensing. *Pattern Recogn. Lett.*, 33(9):1163–1170. (page)

[66] Georgeiv, T., Zheng, K. C., Curless, B., Salesin, D., Nayar, S., and Intwala, C. (2006). Spatio-angular resolution tradeoff in integral photography. In *In Eurographics Symposium on Rendering*, pages 263–272. (page)

[67] Getreuer, P. (2012). Total variation inpainting using split bregman. *Image Processing On Line*, 2:147–157. (page)

[68] Goldluecke, B. and Wanner, S. (2013). The variational structure of disparity and regularization of 4d light fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (page)

[69] Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. (1996). The lumigraph. In *SIGGRAPH*, pages 43–54. (page)

[70] Grossauer, H. (2004). A combined pde and texture synthesis approach to inpainting. In *Computer Vision-ECCV 2004*, pages 214–224. Springer. (page)

[71] Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. A. (2008). *Feature extraction: foundations and applications*, volume 207. Springer. (page)

[72] Hansen, E. and Walster, G. W. (2003). *Global optimization using interval analysis: revised and expanded*, volume 264. CRC Press. (page)

[73] Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition. (page)

[HCI Dataset] HCI Dataset. HCI Dataset. `https://hci.iwr.uni-heidelberg.de/hci/softwares/light_field_analysis`. (page)

[75] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852. (page)

[76] Heber, S. and Pock, T. (2014). Shape from light field meets robust PCA. In *Proceedings of the 13th European Conference on Computer Vision*. (page)

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

[78] Heber, S., Ranftl, R., and Pock, T. (2013). Variational Shape from Light Field. In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*. (page)

[79] Hel-Or, Y. and Shaked, D. (2008). A discriminative approach for wavelet denoising. *Image Processing, IEEE Transactions on*, 17(4):443–457. (page)

[80] Herculano-Houzel, S. (2009). The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in human neuroscience*, 3:31. (page)

[81] Herling, J. and Broll, W. (2012). Pixmix: A real-time approach to high-quality diminished reality. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 141–150. IEEE. (page)

[82] Hinton, G. E. (1999). Products of experts. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 1–6. IET. (page)

[83] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800. (page)

[84] Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243. (page)

[85] Isaksen, A., McMillan, L., and Gortler, S. J. (2000). Dynamically reparameterized light fields. In *SIGGRAPH*, pages 297–306. (page)

[86] Jenkinson, J., Grigoryan, A., Hajinoroozi, M., Diaz Hernandez, R., Peregrina Barreto, H., Ortiz Esquivel, A., Altamirano, L., and Chavushyan, V. (2014). Machine learning and image processing in astronomy with sparse data sets. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 200–203. IEEE. (page)

[87] Jenkinson, J., Grigoryan, A. M., and Agaian, S. S. (2015). Enhancement of galaxy images for improved classification. In *IS&T/SPIE Electronic Imaging*, pages 93990X–93990X. International Society for Optics and Photonics. (page)

[88] Jeon, H. G., Park, J., Choe, G., Park, J., Bok, Y., Tai, Y. W., and Kweon, I. S. (2015). Accurate depth map estimation from a lenslet light field camera. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1547–1555. (page)

[89] Jia, H., Murphey, Y. L., Shi, J., and Chang, T.-S. (2004). An intelligent real-time vision system for surface defect detection. In *PR*, volume 3, pages 239–242. IEEE. (page)

[90] Kaelbling, L. (2003). Jmlr special issue on variable and feature selection. (page)

[91] Karmitsa, N., Bagirov, A., and Makela, M. M. (2009). Empirical and theoretical comparisons of several nonsmooth minimization methods and software. Technical report, Technical Report 959, Turku Centre for Computer Science, Turku. (page)

[92] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980. (page)

[93] Kokaram, A. C., Morris, R. D., Fitzgerald, W. J., and Rayner, P. J. (1995). Interpolation of missing data in image sequences. *Image Processing, IEEE Transactions on*, 4(11):1509–1519. (page)

[94] Komodakis, N. and Tziritas, G. (2007). Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *Trans. Img. Proc.*, 16(11):2649–2661. (page)

[95] Kong, X., Li, K., Yang, Q., Wenyin, L., and Yang, M.-H. (2013). A new image quality metric for image auto-denoising. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2888–2895. IEEE. (page)

[96] Kontschieder, P., Fiterau, M., Criminisi, A., and Rota Bulo, S. (2015). Deep neural decision forests. In *ICCV*, pages 1467–1475. (page)

[97] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc. (page)

[98] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551. (page)

[99] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. (page)

[100] Levin, A., Zomet, A., and Weiss, Y. (2003). Learning how to inpaint from global image statistics. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 305–312 vol.1. (page)

[101] Levoy, M. and Hanrahan, P. (1996). Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 31–42, New York, NY, USA. ACM. (page)

[102] Lions, P. L. and Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer. Anal.*, 16(6):964–979. (page)

[103] Liu, A., Lin, W., and Narwaria, M. (2012). Image quality assessment based on gradient similarity. *Image Processing, IEEE Transactions on*, 21(4):1500–1512. (page)

[104] Liu, D., Sun, X., Wu, F., Li, S., and Zhang, Y.-Q. (2007). Image compression with edge-based inpainting. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(10):1273–1287. (page)

[105] Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528. (page)

[106] Liu, H. and Motoda, H. (2007). *Computational methods of feature selection*. CRC Press. (page)

[107] Liu, H. and Motoda, H. (2012). *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media. (page)

[108] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440. (page)

[Lytro] Lytro. Lytro. `https://www.lytro.com/`. (page)

[110] Maeireizo, B., Litman, D., and Hwa, R. (2004). Co-training for predicting emotions with spoken dialogue data. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 28. Association for Computational Linguistics. (page)

[111] Mai-Duy, N. and Tran-Cong, T. (2003). Approximation of function and its derivatives using radial basis function networks. *Applied Mathematical Modelling*, 27(3):197 – 220. (page)

[112] Malhi, A. and Gao, R. X. (2004). Pca-based feature selection scheme for machine defect classification. *Instrumentation and Measurement, IEEE Transactions on*, 53(6):1517–1525. (page)

[113] Masnou, S. and Morel, J.-M. (1998). Level lines based disocclusion. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pages 259–263 vol.3. (page)

[114] Matthias, S. (2001). Learning with labeled and unlabeled data. *Inst. Adapt. Neural Comput.* (page)

[115] Mitra, K. and Veeraraghavan, A. (2012). Light field denoising, light field superresolution and stereo camera based refocussing using a gmm light field patch prior. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 22–28. IEEE. (page)

[MNIST] MNIST. Mnist. `http://yann.lecun.com/exdb/mnist/`. (page)

[117] Mumford, D. and Shah, J. (1989). Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. on Pure and Applied Mathematics*, 42(5):577–685. (page)

[118] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Fuernkranz, J. and Joachims, T., editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress. (page)

[119] Nemirovsky, A. and Yudin, D. (1983). *Problem complexity and method efficiency in optimization*. Wiley-Interscience series in discrete mathematics. Wiley, Chichester, New York. A Wiley-Interscience publication. (page)

[120] Neumaier, A. (2016). Osga: a fast subgradient algorithm with optimal complexity. *Mathematical Programming*, 158(1):1–21. (page)

[121] Ng, R. (2006). *Digital Light Field Photography*. Phd thesis, Stanford University. (page)

[122] Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134. (page)

[123] Nikolaidis, N. and Pitas, I. (2001). Digital image processing in painting restoration and archiving. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 586–589. IEEE. (page)

[124] Oberdörster, A., Favaro, P., and Lensch, H. P. (2014). Anamorphic pixels for multi-channel superresolution. In *Computational Photography (ICCP), 2014 IEEE International Conference on*, pages 1–10. IEEE. (page)

[125] Ochs, P., Chen, Y., Brox, T., and Pock, T. (2014). ipiano: Inertial proximal algorithm for nonconvex optimization. *SIAM Journal on Imaging Sciences*, 7(2):1388–1419. (page)

[126] Osuna, E., Freund, R., and Girosi, F. (1997). Training support vector machines: an application to face detection. In *CVPR*, pages 130–136. IEEE. (page)

[127] Pardalos, P. M. and Vavasis, S. A. (1992). Open questions in complexity theory for numerical optimization. *Mathematical Programming*, 57(1):337–339. (page)

[128] Park, J. and Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural Comput.*, 3(2):246–257. (page)

[129] Passty, G. B. (1979). Ergodic convergence to a zero of the sum of monotone operators in hilbert space. 72. (page)

[130] Pele, O., Taskar, B., Globerson, A., and Werman, M. (2013). The pairwise piecewise-linear embedding for efficient non-linear classification. In Dasgupta, S. and Mcallester, D., editors, *Proceedings of the 30th International Conference on Machine*

Learning (ICML-13), volume 28, pages 205–213. JMLR Workshop and Conference Proceedings. (page)

[131] Perwass, C. and Wietzke, L. (2012). Single lens 3d-camera with extended depth-of-field. In *Proc. SPIE*, volume 8291, pages 829108–829108–15. (page)

[132] Pham, T. P., Ng, H. T., and Lee, W. S. (2005). Word sense disambiguation with semi-supervised learning. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1093. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. (page)

[133] Pierucci, F., Harchaoui, Z., and Malick, J. (2014). A smoothing approach for composite conditional gradient with nonsmooth loss. (page)

[POV-Ray] POV-Ray. Pov-ray. `http://www.povray.org`. (page)

[Raytrix] Raytrix. Raytrix. `https://www.raytrix.de/`. (page)

[136] Recht, B., Fazel, M., and Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 52(3):471–501. (page)

[137] Rehman, A. and Wang, Z. (2011). Ssim-based non-local means image denoising. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 217–220. IEEE. (page)

[138] Rennie, J. D. M. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 713–719, New York, NY, USA. ACM. (page)

[139] Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. (page)

[140] Roth, S. and Black, M. (2005). Fields of experts: a framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 860–867 vol. 2. (page)

[141] Roth, S. and Black, M. (2007). Steerable random fields. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. (page)

[142] Roth, S. and Black, M. J. (2009). Fields of experts. *International Journal of Computer Vision*, 82(2):205–229. (page)

[143] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268. (page)

[144] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA. (page)

[145] Schmidt, U., Gao, Q., and Roth, S. (2010). A generative perspective on MRFs in low-level vision. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1751–1758. IEEE. (page)

[146] Shalev-Shwartz, S. and Zhang, T. (2013). Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *CoRR*, abs/1309.2375. (page)

[147] Smola, A. J., Vishwanathan, S. V. N., and Le, Q. V. (2007). Bundle methods for machine learning. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *NIPS*. MIT Press. (page)

[148] Springenberg, J., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2015). Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*. (page)

[149] Srebro, N., Rennie, J. D. M., and Jaakola, T. S. (2005). Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press. (page)

[Stanford Light Field Archive] Stanford Light Field Archive. Stanford Light Field Archive. `http://lightfield.stanford.edu/lfs.html`. (page)

[151] Stefan Heber, W. Y. and Pock, T. (2016). U-shaped networks for shape from light field. In *Proc. British Machine Vision Conf.* (page)

[152] Tao, M. W., Hadap, S., Malik, J., and Ramamoorthi, R. (2013). Depth from combining defocus and correspondence using light-field cameras. In *International Conference on Computer Vision (ICCV)*. (page)

[TensorFlow] TensorFlow. Tensorflow. `https://www.tensorflow.org/`. (page)

[154] Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288. (page)

[155] Tiefenbacher, P., Bogischef, V., Merget, D., and Rigoll, G. (2015). Subjective and objective evaluation of image inpainting quality. In *Proc. International Conference on Image Processing, ICIP 2015*, Qubec City, Canada. IEEE. to appear. (page)

[156] Van der Maaten, L. J. P., Postma, E. O., and van den Herik, H. J. (2008). Dimensionality Reduction: A Comparative Review. (page)

[157] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511. IEEE. (page)

[158] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612. (page)

[159] Wanner, S. and Goldluecke, B. (2012a). Globally consistent depth labeling of 4D lightfields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (page)

[160] Wanner, S. and Goldluecke, B. (2012b). Spatial and angular variational super-resolution of 4d light fields. In *European Conference on Computer Vision (ECCV)*. (page)

[161] Wanner, S. and Goldluecke, B. (2014). Variational light field analysis for disparity estimation and super-resolution. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):606–619. (page)

[162] Wanner, S., Straehle, C., and Goldluecke, B. (2013). Globally consistent multi-label assignment on the ray space of 4d light fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (page)

[163] Watson, G. (1992). Characterization of the subdifferential of some matrix norms. *Linear Algebra and its Applications*, 170(0):33 – 45. (page)

[Watson Developer Cloud] Watson Developer Cloud. Watson developer cloud. `https://www.ibm.com/watson/developercloud/`. (page)

[165] Wei, G. W. (1999). Generalized perona-malik equation for image restoration. *IEEE Signal processing letters*, 6(7):165–167. (page)

[166] Weiss, S. M. and Kapouleas, I. (1990). An empirical comparison of pattern recognition, neural nets and machine learning classification methods. *Readings in machine learning*, pages 177–183. (page)

[167] Wilburn, B., Joshi, N., Vaish, V., Talvala, E.-V., Antunez, E., Barth, A., Adams, A., Horowitz, M., and Levoy, M. (2005). High performance imaging using large camera arrays. *ACM Trans. Graph.*, 24(3):765–776. (page)

[168] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259. (page)

[169] WOODBURY, M. (1950). Inverting modified matrices. Technical report, Princeton University. (page)

[170] Xu, Z. and Sun, J. (2010). Image inpainting by patch propagation using patch sparsity. *Image Processing, IEEE Transactions on*, 19(5):1153–1165. (page)

[171] Yang, H., Xu, Z., King, I., and Lyu, M. R. (2010). Online learning for group lasso. In *ICML*, pages 1191–1198. (page)

[172] Yang, T., Jin, R., Mahdavi, M., and Zhu, S. (2012). An efficient primal-dual prox method for non-smooth optimization. *CoRR*, abs/1201.5283. (page)

[173] Yao, Y., Marcialis, G. L., Pontil, M., Frasconi, P., and Roli, F. (2003). Combining flat and structured representations for fingerprint classification with recursive neural networks and support vector machines. *PR*, 36(2):397–406. (page)

[174] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics. (page)

[175] Yoon, Y., Jeon, H. G., Yoo, D., Lee, J. Y., and Kweon, I. S. (2015). Learning a deep convolutional network for light-field image super-resolution. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 57–65. (page)

[176] Yu, W., Heber, S., and Pock, T. (2015). Learning reaction-diffusion models for image inpainting. In *Pattern Recognition - 37th German Conference, GCPR 2015, Aachen, Germany, October 7-10, 2015, Proceedings*, pages 356–367. (page)

[177] Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67. (page)

[178] Zeiler, M. D. and Fergus, R. (2014). *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, chapter Visualizing and Understanding Convolutional Networks, pages 818–833. Springer International Publishing, Cham. (page)

[179] Zhang, X., Burger, M., and Osher, S. (2011). A unified primal-dual algorithm framework based on bregman iteration. *Journal of Scientific Computing*, 46(1):20–46. (page)

[180] Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458. (page)