Imperial College London
Department of Electrical and Electronic Engineering

New Methods for Deep Dictionary Learning and for Image Completion

Junjie Huang

October 2019

Supervised by Professor Pier Luigi Dragotti

Submitted in part fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London
and the Diploma of Imperial College London

Declaration of Originality

I declare that this thesis and the research it contains are the product of my own work under the guidance of my thesis supervisor Professor Pier Luigi Dragotti. All material that is not my own work has been properly acknowledged.

Junjie Huang

Copyright Declaration

Abstract

Digital imaging plays an essential role in many aspects of our daily life. However due to the hardware limitations of the imaging devices, the image measurements are usually inpaired and require further processing to enhance the quality of the raw images in order to enable applications on the user side. Image enhancement aims to improve the information content within image measurements by exploiting the properties of the target image and the forward model of the imaging device. In this thesis, we aim to tackle two specific image enhancement problems, that is, single image super-resolution and image completion.

First, we present a new Deep Analysis Dictionary Model (DeepAM) which consists of multiple layers of analysis dictionaries with associated soft-thresholding operators and a single layer of synthesis dictionary for single image super-resolution. To achieve an effective deep model, each analysis dictionary has been designed to be composed of an Information Preserving Analysis Dictionary (IPAD) which passes essential information from the input signal to output and a Clustering Analysis Dictionary (CAD) which generates discriminative feature representation. The parameters of the deep analysis dictionary model are optimized using a layer-wise learning strategy. We demonstrate that both the proposed deep dictionary design and the learning algorithm are effective. Simulation results show that the proposed method achieves comparable performance with Deep Neural Networks and other existing methods.

We then generalize DeepAM to a Deep Convolutional Analysis Dictionary Model (DeepCAM) by learning convolutional dictionaries instead of unstructured dictionaries. The convolutional dictionary is more suitable for processing high-dimensional signals

like images and has only a small number of free parameters. By exploiting the properties of a convolutional dictionary, we present an efficient convolutional analysis dictionary learning algorithm. The IPAD and the CAD parts are learned using variations of the proposed convolutional analysis dictionary learning algorithm. We demonstrate that DeepCAM is an effective multi-layer convolutional model and achieves better performance than DeepAM while using a smaller number of parameters.

Finally, we present an image completion algorithm based on dense correspondence between the input image and an exemplar image retrieved from Internet which has been taken at a similar position. The dense correspondence which is estimated using a hierarchical PatchMatch algorithm is usually noisy and with a large occlusion area corresponding to the region to be completed. By modelling the dense correspondence as a smooth field, an Expectation-Maximization (EM) based method is presented to interpolate a smooth field over the occlusion area which is then used to transfer image content from the exemplar image to the input image. Color correction is further applied to diminish the possible color differences between the input image and the exemplar image. Numerical results demonstrate that the proposed image completion algorithm is able to achieve photo realistic image completion results.

Acknowledgements

First and foremost, I would like to thank my doctoral thesis advisor Professor Pier Luigi Dragotti for his guidance, supervision and support not only on research but on all aspects. He has given me maximum freedom and full support. I am sincerely grateful for his consistent belief, trust and countless wise advice during my PhD study, especially during the difficult times. The research works would not even be possible without his vision, help and encouragement.

I would like to express my gratitude to my PhD examiners Dr. Tania Stathaki from Imperial College London and Professor Wenwu Wang from University of Surrey for their constructive and valuable feedback and suggestions.

I am grateful to the Communications and Signal Processing group in Imperial College London for offering me the scholarship to support my research. I would like to thank my friends in Imperial College London Cen, Cheng, Hamdi, Hengyan, Ivy, Jingyuan, John, Ling, Michael, Mingbo, Pingfan, Qi, Roxana, Shengxi, Stephanie, Wenqiang, Yang, Yang, Yuqi, Zheng for their help and support. You have made my PhD journey memorable and enjoyable.

I would like to thank my parents and grandparents for their constant and unconditional support and love. Last but not least, I would like to thank my wife Tianrui for always being by my side during every ups and downs of my PhD study. I would never be able to reach that far without you.

Junjie Huang, *London, UK.*

Contents

List of Tables

List of Figures

Nomenclature

List of Acronyms

CNN	Convolutional neural network
CSC	Convolutional sparse coding
DNN	Deep neural network
EM	Expectation Maximization
HR	High-resolution
LR	Low-resolution
MAP	Maximum a posteriori
MLP	Multilayer perceptron
NNF	Nearest neighbor field
OMP	Orthogonal matching pursuit
PSNR	Peak signal-to-noise ratio
ReLU	Rectified linear unit
RKHS	Reproducing kernel Hilbert space
ROI	Region of interests
SISR	Single image super-resolution
SR	Super-resolution
SVD	Singular vector decomposition

Notations

$\boldsymbol{X}$	Matrix: $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ with $m, n > 1$
$\boldsymbol{x}$	Vector: $\boldsymbol{x} \in \mathbb{R}^{m}$ with $m > 1$
$\boldsymbol{X}^T$	Transpose of matrix $\boldsymbol{X}$

$\boldsymbol{X}^{-1}$	Inverse of matrix of matrix $\boldsymbol{X}$
$\boldsymbol{X}^{\perp}$	Orthogonal complement matrix $\boldsymbol{X}$
$\boldsymbol{X}^{\dagger}$	Pseudo-inverse of matrix of matrix $\boldsymbol{X}$
$\det(\cdot)$	Matrix determinant
$\log(\cdot)$	Natural logarithm
$\lvert x \rvert$	Absolute value of a real number x
$\mathbb{R}$	The set of real numbers
$\mathbf{I}$	Identity matrix
$\mathbf{T}$	Toeplitz matrix
$\mathcal{H}_{\lambda}(\cdot)$	Hard-thresholding operator
$\mathcal{S}_{\lambda}(\cdot)$	Soft-thresholding operator
$\mathrm{rank}(\cdot)$	Matrix rank
$\lVert \boldsymbol{x} \rVert_0$	l_0-norm, number of non-zero entries in $\boldsymbol{x}$
$\lVert \boldsymbol{x} \rVert_1$	l_1-norm: $\sum_i \lvert \boldsymbol{x}(i) \rvert$
$\lVert \boldsymbol{x} \rVert_2$	l_2-norm: $\sqrt{\sum_i \boldsymbol{x}(i)^2}$
$\lVert \boldsymbol{X} \rVert_F$	Frobenius norm: $\sqrt{\sum_i \boldsymbol{X}(i,j)^2}$
$g_1[n] * g_2[n]$	Discrete-time convolution: $\sum_{m \in \mathbb{Z}} g_1[m] g_2[n-m]$
x_i	The i-th element of a vector $\boldsymbol{x}$
$x_{i,j}$	The element at the i-th row and the j-th column of a matrix $\boldsymbol{X}$

Chapter 1

Introduction

1.1 Motivations

Digital imaging devices play a pivotal role in connecting us with the real continuous world through the sampled discretized measurements. They enable us to capture scenes of the world into a form that we could further process and use as a tool to understand the world. Digital imaging appears almost everywhere and has significantly improved our quality of life in many aspects. It could be the digital camera on our mobile phone that keeps beautiful moments for us, it could be the magnetic resonance imaging (MRI) system in hospital that helps doctors diagnose diseases and save lives, it could be the space telescope that assist scientists to look through space and time and unravel the mystery of the universe.

Despite a broad range of applications, the quality of the acquired raw image measurements is restrained by hardware limitations of optics and sensors. The raw image measurements could be noisy, incomplete, distorted, or with insufficient resolution. The contents within the raw images are sometimes too inpaired to be directly used by the end users. It is therefore necessary and essential to enhance the quality of the raw

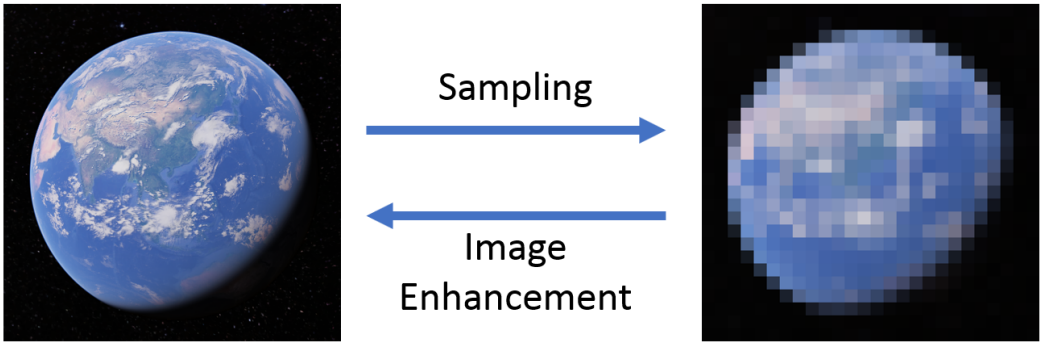


Figure 1.1: **Digital imaging and image enhancement.** The imaging devices enable us sense the continuous world into a discretized form, while there may contain insufficient/noisy information for us to fully understand the world. Image enhancement aims to recover or improve the information in the acquired image measurements to restore a better view of the real world.

images to facilitate its further use.

Image enhancement, which aims to recover or improve the information content within the image measurements, is a classical inverse problem in signal processing and computer vision. It serves as a tool to compensate the hardware limitations of the digital imaging systems and aims to achieve high quality images for real applications. According to the type of distortion incurred, image enhancement can be further categorized into image denoising, image completion, image deconvolution, and image super-resolution. Fig. 1.1 illustrates the digital imaging process and the objective of image enhancement.

Image enhancement has been intensively studied in the past decades. It has been advancing and evolving along with the hardware advancement and the emerging of new applications. There are two main categories of methods for tackling the image enhancement problems: model-based approaches and learning-based (or data-driven) approaches:

- Model-based algorithms are built from first principles and are established based on our understanding of the forward model of the imaging system and the prior model of the image signals to be recovered. The forward model describes the

image formation process of the imaging device and shows how the image measurements are related to the unknown high quality image. The prior model depicts the characteristics of the unknown high quality image based on our understanding. When the forward model and the prior model can be well modelled with proper mathematical formulations, a model-based method can lead to an effective solution. However, sometimes the forward model and the prior model are too difficult to be fully understood and expressed mathematically.

- Learning-based algorithms take an alternative direction by learning an inference model which tries to estimate a high quality image based on the image measurements from an external training dataset which contains sufficient high quality image samples or pairs of high quality and observed image samples. Note that an insightful understanding about the forward model and the prior model would also be beneficial for the learning-based algorithms. The learning-based algorithms usually require a training stage in which the parameters of the inference model are determined with the objective of minimizing the discrepancy between the enhanced image and the existing high quality image. The performance of a learning-based algorithm is closely related to the quality and quantity of the training dataset as well as the learning capability of the algorithm.

In this thesis, we are trying to push the boundaries of both the model-based direction and the learning-based direction on two specific image enhancement problems: single image super-resolution and image completion:

- We investigate a learning-based framework for single image super-resolution. Inspired by the recent success of Deep Neural Networks and the recent efforts to develop multi-layer dictionary models, we propose a Deep Analysis Dictionary framework with unstructured dictionaries and convolutional dictionaries. The proposed deep dictionary models lead to efficient and effective solutions for sin-

gle image super-resolution, and have a higher interpretability compared to deep neural networks.

- We take a model-based approach for image completion. Given an input image and an exemplar image, the dense correspondence between these two images is estimated and has a large occlusion area for the region to be completed. Based on a smoothness prior, we propose an Expectation-Maximization (EM) based nearest neighbour field (NNF) interpolation algorithm which interpolates a smooth and accurate NNF over the occluded region. Given the interpolated NNF, the exemplar content can be transferred from the exemplar image to the input image.

1.2 Outline of the Thesis

The remainder of this thesis is organised as follows:

In Chapter 2, we will first give an introduction to image enhancement, especially, image super-resolution and image completion starting with the problem formation, followed with discussions on the inherent difficulties and a review on the existing methods. Sparse models and deep neural networks are two classes of algorithms for solving image enhancement problems. The working principles and the different variations of sparse models and deep neural networks will be then reviewed.

In Chapter 3, we propose a Deep Analysis Dictionary Model (DeepAM) framework for single image super-resolution. An L-layer DeepAM consists of L layers of analysis dictionaries with associated soft-thresholding operators and a layer of synthesis dictionary. The forward model of DeepAM is a multi-layer matrix multiplications and element-wise soft-thresholding operations. To achieve an effective image super-resolution, each analysis dictionary is a combination of two sub-dictionaries: an Information Preserving Analysis Dictionary (IPAD) and a Clustering Analysis Dictionary (CAD). The IPAD

and soft-thresholding pair are designed to preserve essential information from the input, and the CAD and soft-thresholding pair aim to generate a discriminative representation. The proposed DeepAM provides an interpretable multi-layer non-linear dictionary model and could be helpful for the understanding of the workings of DNNs. We propose a layer-wise learning algorithm for learning the dictionaries and soft-thresholds in DeepAM. Simulation results show that our proposed DeepAM method achieves comparable performance with Deep Neural Networks (DNNs) based method and other existing methods.

In Chapter 4, we adapt the DeepAM framework from an unstructured dictionary model to a convolutional dictionary model. When the input signal is high-dimensional, an unstructured dictionary model requires a large number of parameters and has high computational complexity. A convolutional dictionary represents the convolution between a set of convolutional filters and a vector. It is a structured dictionary and can be expressed as a concatenation of Toeplitz matrices where each Toeplitz matrix corresponds to a filter. When the filters have compact support, the convolutional dictionary has a small number of parameters and requires a lower computational cost. We propose an efficient convolutional analysis dictionary learning algorithm by exploiting the properties of a convolutional dictionary. We propose a Deep Convolutional Analysis Dictionary Model (DeepCAM) framework which consists of multiple layers of convolutional analysis dictionaries and the corresponding soft-thresholding operations and a layer of convolutional synthesis dictionary. Based on the proposed convolutional analysis dictionary learning algorithm, we propose a layer-wise learning algorithm for learning the convolutional dictionaries and the soft-thresholds. Simulation results show that the proposed DeepCAM achieves satisfactory results.

In Chapter 5, we propose an image completion algorithm based on dense correspondence between the input image and an exemplar image retrieved from Internet. Contrary to traditional methods which register two images according to sparse correspon-

dence, we propose a hierarchical PatchMatch method that progressively estimates a dense correspondence, which is able to capture small deformations between images. The estimated dense correspondence has usually large occlusion areas that correspond to the regions to be completed. A nearest neighbor field (NNF) interpolation algorithm interpolates a smooth and accurate NNF over the occluded region. Given the calculated NNF, the correct image content from the exemplar image is transferred to the input image. Finally, as there could be a color difference between the completed content and the input image, a color correction algorithm is applied to remove the visual artifacts. Numerical results show that our proposed image completion method can achieve photo realistic image completion results.

Finally in Chapter 6, we conclude the thesis by summarizing the achievements and providing possible future research directions.

1.3 Publications

The materials presented in this thesis have led to the following publications:

To be Submitted

- J.J. Huang and P.L. Dragotti, "Learning Deep Analysis Dictionary - Part I: Unstructured Dictionary," **To be submitted to:** *IEEE Transactions on Signal Processing.*
- J.J. Huang and P.L. Dragotti, "Learning Deep Analysis Dictionary - Part II: Convolutional Dictionary," **To be submitted to:** *IEEE Transactions on Signal Processing.*

Peer-reviewed Journal

- J.J. Huang and P.L. Dragotti, "Photo Realistic Image Completion via Dense Correspondence," *IEEE Transactions on Image Processing*, vol 27, no. 11, pp. 5234-5247, November 2018.

Peer-reviewed Conferences

- J.J. Huang, and P.L. Dragotti, "A Deep Dictionary Model to Preserve and Disentangle Key Features in A signal," *In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'2019)*, Brighton, United Kingdom, April 2019
- J.J. Huang, and P.L. Dragotti, "A Deep Dictionary Model for Image Super-Resolution," *In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'2018)*, Calgary, Canada, April 2018.

Conference Abstract

- J.J. Huang, and P.L. Dragotti, "A Deep Analysis Dictionary Model," *In Proc. The Signal Processing with Adaptive Sparse Structured Representations workshop (SPARS'2019)*, Toulouse, France, July 2019.

Chapter 2

Background

In this chapter, we first give an introduction to image enhancement focusing on image super-resolution and image completion. The problem formulation and the inherent difficulties will be discussed and analyzed, and the existing methods will then be presented. In the second part, the sparse signal model will be reviewed. The sparse signal model is one of the most widely used tool for signal and image processing. It serves as the lens for us to see the world from the sparsity and simplicity perspective. Based on how signals are modelled, the synthesis, analysis and convolutional sparse model will be discussed. Finally, we will review Deep Neural Networks (DNNs) including the basics, the learning method and the variations. The forward model of a DNN is a simple cascade of linear transforms and element-wise non-linearities. With a deep structure, DNNs learn representations of different levels of abstraction and have been applied in many signal and image processing applications.

2.1 Image Enhancement

In an imaging system, the acquired image may not be perfect due to hardware limitations or the conditions under which the image is taken. The forward model of the

imaging system can generally be formulated as:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{2.1}$$

where $\boldsymbol{H}$ is a degradation matrix which approximates the imaging forward model as a linear process, $\boldsymbol{n}$ is the possible additive noise, and $\boldsymbol{y}$ and $\boldsymbol{x}$ are the acquired image and the desired high quality image, respectively.

Image enhancement aims to improve the information content of the acquired image signals and computationally compensate for the imperfections. It is an inverse problem with the objective of recovering a high quality image $\boldsymbol{x}$ from the observation $\boldsymbol{y}$. An inverse problem is well posed if the following three conditions are satisfied: (i) there exists a solution for every observed signal, (ii) the solution is unique, and (iii) the inverse of the degradation matrix is stable. However, image enhancement problems are usually ill-posed. Consequently, it is not enough to make use only of the forward model to solve the ill-posed inverse problem. A prior model describes the properties of the desired solution. A stable solution can be achieved if a proper prior model can be imposed to regularize the feasible solutions.

The image enhancement algorithms can be generally divided into model-based approaches and learning-based approaches according to how the prior model is established:

- Model-based approaches are based on mathematically defined prior models including smoothness, total variation, sparsity, low-rank and so on. The model-based approaches require an accurate modelling of the forward model and the prior information of the desired images. A prior model which can well represent the properties of the signals of interests will lead to accurate estimations, while an inappropriate choice of prior model will cause erroneous estimations. The model-based approaches usually have the advantage of simplicity and can solve

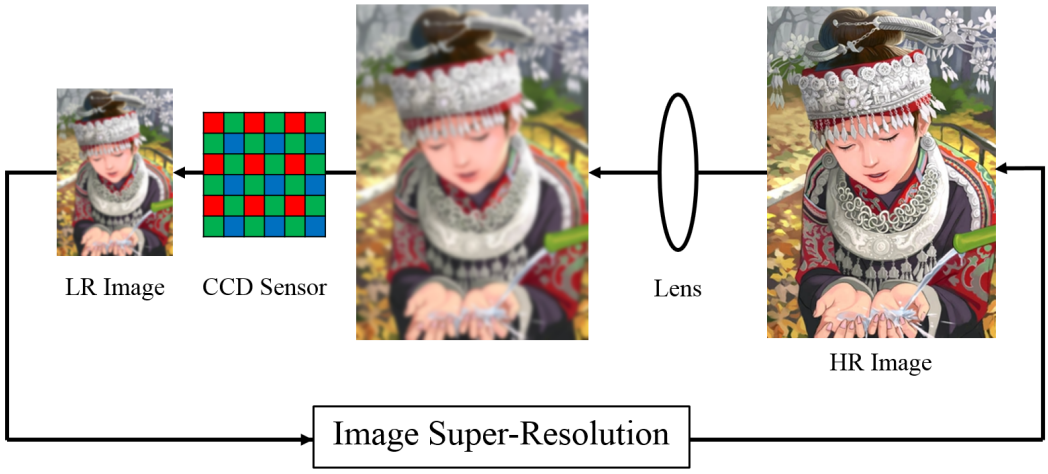


Figure 2.1: **The forward model of a digital camera and image super-resolution.** The captured image is assumed to be the blurred and down-sampled version of a high-resolution image. The objective of image super-resolution is to recover a high-resolution image from the observed low-resolution image.

the problem using a small number of parameters. However, it may not be easy to mathematically represent the forward model and the prior model well.

- Learning-based approaches exploit prior information from an external training dataset which defines a prior manifold in the solution space or defines a one-to-one mapping from the observed signal to the desired solution. When a pair-wise training dataset is used, a learning-based method tries to learn the non-linear inverse mapping $\boldsymbol{y} \rightarrow \boldsymbol{x}$ from a training dataset which consists of training data pairs $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$. The training dataset and the learning inference model are two important factors of the learning-based methods. It is essential to have a large training dataset with high quality training data. A learning-based inference model should be able to generate a feature representation that is both informative and discriminative for a robust estimation.

2.1.1 Image Super-Resolution

Digital cameras have become an indispensable part of our life for helping us capture moments forever. Image super-resolution aims to improve the resolution of a captured

image which is limited by the diffraction limit of the optical lenses and the density of the digital image sensors. Fig. 2.1 illustrates the forward model of a digital camera and the objective of image super-resolution. The upper part of Fig. 2.1 shows the block diagram of a digital camera which consists of an optical lens that focus light from a scene and digital image sensors, for example CCD sensors, that capture the focused light into digital image. The high-frequency details of the acquired image is limited by the optical lens which has a lens blur effect which is described by the Point Spread Function (PSF). The spatial resolution of the acquired LR image is limited by the density of the digital image sensors. The forward model of a digital camera can be expressed as:

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{B}\boldsymbol{x}, \tag{2.2}$$

where $\boldsymbol{x} \in \mathbb{R}^m$ is the vectorized high-resolution image, $\boldsymbol{B} \in \mathbb{R}^{m \times m}$ is the blurring operator which models the optical lens, $\boldsymbol{D} \in \mathbb{R}^{n \times m}$ is the down-sampling operator with $n < m$ which models the sampling operation occurred on digital image sensors, and $\boldsymbol{y} \in \mathbb{R}^n$ is the vectorized acquired low-resolution image.

The blurring operator $\boldsymbol{B}$ usually has a large condition number. The down-sampling operator $\boldsymbol{D}$ is a fat binary matrix in which each row has an entry with value 1 and other entries with value 0. With the blur and down-sampling operator $\boldsymbol{D}\boldsymbol{B}$, the observation of a digital camera is a blurred and down-sampled version of a scene.

The goal of image super-resolution is to restore the original spatial resolution and recover the missing high-frequency components from one or more blurred low-resolution (LR) images. The recovered high-resolution (HR) image should be as close to the ground-truth HR image as possible and should have sharp edges and natural looking textures. In particular, single image super-resolution (SISR) aims to restore the HR image from a single input LR image. Image super-resolution is an ill-posed inverse problem as the size of LR observation is much smaller than the desired HR output. Additional priors should be imposed to regularize the ill-posed problem. The prior

information can be hand-designed priors such as smoothness, sparsity and low-rank, and can also be learned priors which are inferred from an external training dataset.

Image super-resolution algorithms can be divided into patch-based methods and convolution-based methods. Both patch-based and convolution-based methods can learn from an external training dataset which contains LR and HR image pairs as training data.

As the number of HR pixels to be estimated is huge, it is difficult and computational inefficient to estimate the HR image as a whole. Patch-based methods [1,2,7–11] divide the LR image into small overlapping patches (e.g. 6×6 image regions) and infer a HR patch for each LR patch. The HR image can then be reconstructed by combining the estimated HR patches with patch overlapping. The method proposed by Zeyde *et al.* [2] is a synthesis dictionary based method with a coupled LR and HR dictionary. The LR dictionary is learned using K-SVD method [12] and has 1024 atoms, and the HR dictionary is learned using least squares. It assumes that a LR patch and its corresponding HR patch share the same sparse code which is retrieved using the OMP method [13]. The input LR feature is the concatenation of the intensity, the first-order derivatives, and the second-order derivatives of the LR data and is further compressed using Principle Component Analysis (PCA). The ANR method [7] and the A+ method [8] use the same feature representation as [2]. They apply a learned LR synthesis dictionary for LR patch clustering and have a regression model for each dictionary atom. The super-resolution algorithm finds the nearest neighbor atom for each input LR signal and apply the corresponding regression model for HR signal prediction. The dictionary has 1024 atoms and thus there are 1024 regression models. The A+ method [8] represented state-of-the-art before the emergence of methods based on deep convolutional neural networks.

Convolution-based methods [14–16] instead estimate all HR pixel values all together and are based on convolutional models such as Convolutional Neural Networks (CNN). The Super-Resolution Convolutional Neural Network (SRCNN) method [14] is the

first method that used convolutional neural network for single image super-resolution. SRCNN has 3 layers and uses 64 filters with spatial size 9×9, 32 filters with spatial size 1×1 and 32 filters with spatial size 5×5 for layer 1, 2 and 3, respectively. It takes the bicubic up-scaled image as input and is able to upscale the input LR image without dividing the input image into patches. There are two main strategies in single image super-resolution (SISR) using Convolutional Neural Networks, i.e. the early upsampling approach [15, 17] and the late upsampling approach [18, 19]. The early upsampling approach [15, 17] first upsamples the low-resolution (LR) image to the same resolution as the desired high-resolution (HR) one through bicubic interpolation and performs convolution on the upsampled image. The drawback of this approach is that the computational complexity is high during testing as the feature maps are of the same size as the HR image. The late upsampling approach [18,19] performs convolution on the input LR image and applies a deconvolution layer or a sub-pixel convolution layer at the last layer to predict all HR pixel values. The late upsampling approach has lower computational cost than the early one. After the emerging of the CNN-based image super-resolution algorithms, they become the dominant approaches for single image super-resolution task. This is mainly due to an end-to-end model parameter updating based on backpropagation algorithm [20] which enables an effective learning for a flexible network structure from a large dataset.

2.1.2 Image Completion

Image completion [21] tries to meet the increasing demand of editing personal photos, in particular by replacing an undesired image region (such as strangers, and construction sites) with a natural looking background which should be as close as possible to the real scene. The difficulty of this problem is directly related to the size of the Region-of-Interest (ROI) or "hole" to be completed which is assumed to be specified by the users. The larger the "hole", the higher the probability the missing content is non-stationary

Figure 2.2: **An example of image completion.** The objective of image completion is to remove the image content in the input image (left) indicated by the mask image (middle) and to fill it with a natural image background. The image completion result should be natural looking and consistent with the original image.

and this fact makes the image completion problem harder.

Single image completion algorithms [3, 22–31] exploit the self-similarity of image regions (usually small image patches) within the input image and cover the ROI with similar image content. Impressive completion results are demonstrated by recent works when ROI is relatively small and a sufficient number of repetitive patterns exists. The main limitation of the single image based approaches is that the completion results tend to deviate from the real scene and become less realistic as the size of ROI grows. This happens because the self-similarity assumption is less satisfied. Recent single image completion methods are mainly example based [3, 23, 24, 27, 28, 30, 31] rather than diffusion based [32, 33]. The diffusion based methods [32, 33] fill in the missing content on the input image by propagating information from the nearby regions with a smoothness constraint. Therefore, when the size of the ROI is too large or when the content of the missing region is non-static, diffusion based methods will not be able to recover an realistic image content. The example based methods exploit redundancy within the input image itself and transfer image patches from the image regions outside the "hole" to the "hole" region. The central issue here is to establish accurate patch correspondence in order to find similar patches within the input image. Fast patch matching methods [25, 26, 34–36] play an important role in these algorithms. In particular, PatchMatch [25, 26] is a patch-based fast algorithm for dense correspondence estimation. The space-time completion method [23] searches similar patches and

replaces the patches on the boundary of the "hole". By iteratively searching and updating, the "hole" gradually shrinks. The image melding method [3], which generates natural looking results, further extends the search space of the generalized PatchMatch method [26] with reflection, gain and bias. He and Sun [30] proposed to utilize the statistics of the offsets between similar patches returned by PatchMatch to accelerate the matching process. Since image completion can be considered as a multi-label discrete optimization problem, other discrete optimization techniques, such as Graph Cuts [37, 38] and Belief Propagation [39], are also employed to address the single image completion problem. Pathak *et al.* [40] and Iizuka *et al.* [5] take the data driven approach for image completion by learning from an external dataset using deep neural networks.

Internet-based image completion algorithms [4, 6, 41–43] instead search for suitable image content from existing similar images available on Internet and can, in this way, overcome the limitations of the self-similarity assumption. An image is called an exemplar image if it has been taken from a similar viewpoint as the input image but may differ in camera parameters, illumination conditions, and with possible occlusions. With the help of exemplar images from the Internet, suitable image regions can be transferred to the input image, and under these conditions, accurate and photo realistic restoration becomes possible. In the pioneering work of Hays and Efros [42], they proposed to find images which are semantically similar to the input image and perform context matching and blending using a graphical model. However, their retrieved images could be taken from a distinct location and not satisfy our definition of exemplar image. The resultant image may not be faithful to the real scene and the incorrect reconstructions may lead to unrealistic images. There are many papers e.g. [4, 6, 41] that search and apply exemplar images for image completion. Amirshahi and Kondo [41] proposed to find a single homography correspondence between two images from obtained sparse correspondence (i.e. the matched SIFT keypoints [44] between images) and transfer patches with respect to the locations specified by the homography model. The limita-

tion of the single homography model is that the relationship between matched SIFT keypoints may not be well modeled if two images do not share the same camera center or contain piece-wise planar scenes. Whyte *et al.* [6] use a geometrical registration and a color registration for image completion. The geometrical registration is performed using multiple planes to approximate matched SIFT keypoints correspondence. Affine transformation is then applied to each color channel for color registration. A recent work [4] proposed by Zhu *et al.* automatically finds 20 exemplar images from Internet through SIFT keypoints matching with multiple homographies and line segments matching. Each exemplar image is warped to the input image by a mesh-based warping which is assisted with both point and line constraints. A scoring algorithm helps to select as completed image the one with the highest score. Moving from the single homography model [41] to multiple homographies [4, 6], a more general scene correspondence can be more accurately approximated. However, the sparse correspondence they relied on is still a discretized representation of the continuous image structure and may not be able to fully capture the correspondence between images.

2.2 Sparse Modelling

Occam's razor [45] is one of the most influential principles in signal processing. It states that "among competing representations that predict equally well, the one with the fewest number of components should be selected". Based on concept of simplicity and parsimony, sparse signal model describes a signal of interests as a sparse combination of elementary signal components which are usually termed atoms. A dictionary is a matrix which contains all the atoms. The dictionary is essential to the sparse signal model as it defines the prototype signals. The dictionary can be divided into fixed dictionaries and learned dictionaries. The commonly used fixed dictionaries include Discrete Fourier Transform (DFT) matrix, wavelet transform [46, 47] and shearlet transform [48, 49]. These dictionaries usually have some optimized performance supported by theoretical

analysis and are associated with fast implementations, but they may not be adapted enough to a given set of signals. The learned dictionaries are trained using a set of signal samples by imposing a sparse constraint over the sparse representation. The learned dictionaries are more flexible and can adapt to the specific group of signal samples and can usually give a sparser representation than the fixed dictionaries.

Depending on the way of modeling signals, the sparse signal model [50] can be divided into synthesis or analysis model. In the following two subsections, we will briefly introduce the synthesis model and the analysis model.

2.2.1 Synthesis Sparse Model

A synthesis sparse model [50] represents a signal $\boldsymbol{x} \in \mathbb{R}^n$ (the subscript s represents synthesis sparse model) using a sparse vector $\boldsymbol{\gamma}_s \in \mathbb{R}^m$ with $m \geq n$ with respect to a synthesis dictionary $\boldsymbol{D} \in \mathbb{R}^{n \times m}$. The sparse vector indicates that the input signal $\boldsymbol{x}$ is a linear combination of a small number of atoms from the dictionary. Given a redundant synthesis dictionary $\boldsymbol{D}$, the sparse representation $\boldsymbol{\gamma}_s$ is the sparsest description (i.e. the least number of non-zero coefficients) of the input signal $\boldsymbol{x}$:

$$\boldsymbol{x} = \boldsymbol{D}\boldsymbol{\gamma}_s, \tag{2.3}$$

where $\|\boldsymbol{\gamma}_s\|_0 = k \ll m$ and $\|\cdot\|_0$ denotes the l_0 pseudo-norm which counts the number of non-zeros entries of a vector.

Fig. 2.3 shows an example of synthesis sparse signal model for image processing. The signal $\boldsymbol{x}$ is a linear combination of 3 atoms which are indicated by the sparse representation $\boldsymbol{\gamma}_s$.

Sparse representation has been widely used in image processing, signal processing, and machine learning. The sparse representation $\boldsymbol{\gamma}_s$ of a signal serves as a robust and dis-

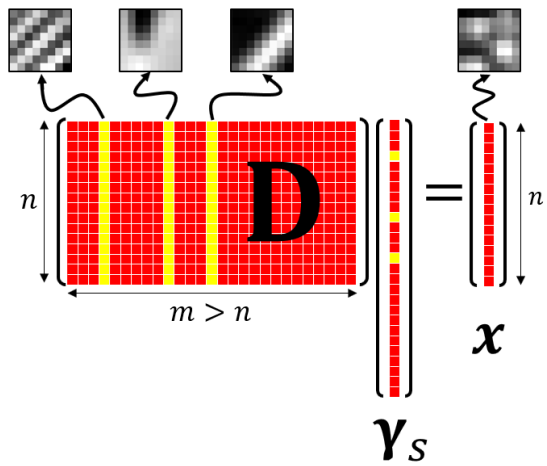


Figure 2.3: **A synthesis sparse signal model.** The input signal $\boldsymbol{x}$ is approximated using 3 atoms from the dictionary $\boldsymbol{D}$. The sparse representation has only 3 non-zero coefficients indicating the contribution from the corresponding activated atoms.

criminative representation. The signal components that cannot be well approximated by a small number of atoms, for example noise, will be suppressed by the sparsity constraint. The reconstructed signal using the obtained sparse representation will then contain less noise components. Finding a sparse representation given an over-complete dictionary and an input signal requires a non-linear operation. The linear non-separable signals within the original space could be linear separable in the sparse coefficient space. Therefore, the sparse representation possesses a stronger discriminative power and can be applied to both regression and classification problems.

Synthesis Sparse Pursuit:

Sparse pursuit aims to find the sparsest representation $\boldsymbol{\gamma}_s$ of the input signal $\boldsymbol{x}$ with respect to a given dictionary $\boldsymbol{D}$. The sparse pursuit problem can be formulated as:

$$\boldsymbol{\gamma}_s = \arg\min_{\boldsymbol{\gamma}} \|\boldsymbol{\gamma}\|_0, \quad \text{s.t. } \boldsymbol{x} = \boldsymbol{D}\boldsymbol{\gamma}. \tag{2.4}$$

Although the sparse representation is a robust and discriminative representation, sparse pursuit is an NP-hard (Non-deterministic Polynomial-time Hard) problem in the case of

arbitrary dictionaries. The constraint $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{\gamma}$ is an under-determined system and has infinite number of solutions. The l_0-norm further regularizes the solution to be sparse, however, it is non-convex and makes the whole problem a non-convex optimization problem.

In practice, sparse pursuit is solved using either greedy approaches [13,51,52] or convex relaxation approaches [53–56].

The greedy algorithms [13, 51, 52] gradually find k non-zero coefficients based on the correlation between the input signal and the atoms. For example, Orthogonal Matching Pursuit (OMP) [13] finds at each iteration a sparse coefficient with the aim of reducing the approximation error.

Convex relaxation algorithms approximate the non-convex problem in Eqn. (2.4) with a convex one by relaxing the non-convex l_0-norm to a convex l_1-norm:

$$\boldsymbol{\gamma}_s = \arg\min_{\boldsymbol{\gamma}} \|\boldsymbol{\gamma}\|_1, \quad \text{s.t. } \boldsymbol{x} = \boldsymbol{D}\boldsymbol{\gamma}, \tag{2.5}$$

where $\|\cdot\|_1$ denotes the l_1-norm with $\|\boldsymbol{\gamma}\|_1 = \sum_{j=1}^{m} |\gamma_j|$.

With augmented Lagrangian, the sparse pursuit problem can then be formulated as a constrained minimization problem:

$$\boldsymbol{\gamma}_s = \arg\min_{\boldsymbol{\gamma}} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{\gamma}\|_2^2 + \lambda\|\boldsymbol{\gamma}\|_1, \tag{2.6}$$

where λ is the regularization parameter which balances between the l_2-norm data fidelity term and the l_1-norm sparse term.

It is of interest to point out that under certain conditions, the l_1-norm minimization problem defined in Eqn. (2.6) is equivalent to the original l_0-norm minimization problem in Eqn. (2.4) [54, 57]. The convex quadratic problem can be solved using basis pursuit (BP) [54] algorithm or iterative algorithms, for example Iterative Soft-

Thresholding Algorithm (ISTA) [55, 56].

Synthesis Dictionary Learning:

Let us denote with $\boldsymbol{X} \in \mathbb{R}^{m \times N}$ a set of N training signal samples where each column is a training signal sample. The dictionary learning problem requires a joint estimation of the dictionary $\boldsymbol{D}$ and the sparse coefficient matrix $\boldsymbol{\Gamma}$, and can be formulated as follows:

$$\arg\min_{\boldsymbol{D},\boldsymbol{\Gamma}} \sum_i \|\boldsymbol{\Gamma}_{:,i}\|_0, \quad \text{s.t. } \boldsymbol{X} = \boldsymbol{D}\boldsymbol{\Gamma}. \tag{2.7}$$

The dictionary learning problem is a bi-linear matrix factorization problem with a sparsity constraint on the sparse representation. The synthesis dictionary learning algorithms [12, 58, 59] are usually based on an alternating minimization strategy and iterate between a sparse coding stage and a dictionary update stage.

In the sparse coding stage, the sparse representations $\boldsymbol{\Gamma}^k$ of the training data $\boldsymbol{X}$ are obtained through sparse pursuit with respect to the current dictionary $\boldsymbol{D}^{k-1}$:

$$\boldsymbol{\Gamma}^k = \arg\min_{\boldsymbol{\Gamma}} \sum_i \|\boldsymbol{\Gamma}_{:,i}\|_0, \quad \text{s.t. } \boldsymbol{X} = \boldsymbol{D}^{k-1}\boldsymbol{\Gamma}. \tag{2.8}$$

In the dictionary update stage, the dictionary is updated using the sparse coefficient matrix and the training data pair $(\boldsymbol{\Gamma}^k, \boldsymbol{X})$:

$$\boldsymbol{D}^k = \arg\min_{\boldsymbol{D}} \|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{\Gamma}^k\|_F^2. \tag{2.9}$$

The method of optimal directions (MOD) [58] computes the updated dictionary $\boldsymbol{D}_k$ as $\boldsymbol{D}_k = \boldsymbol{X}\boldsymbol{\Gamma}^{k\dagger}$ where $\boldsymbol{\Gamma}^{k\dagger}$ is the Moore-Penrose pseudoinverse. The K-SVD method [12] updates the dictionary atom by atom through a rank-1 approximation. For each atom, K-SVD algorithm first finds the data which has non-zero coefficients over this atom

and approximates the contribution of this pair of atom and sparse coefficient using a rank-1 matrix which has a closed form solution using Singular Value Decomposition (SVD). The Simultaneous Codeword Optimization (SimCo) [59] algorithm also iterates between a sparse coding stage which enforces a K-sparse constraint with hard-thresholding and a dictionary update stage which updates all dictionary atoms at the same time by performing dictionary update on manifold.

2.2.2 Analysis Sparse Model

Recently, the analysis sparse model [50, 60–65] has attracted increasing research interest. A redundant analysis dictionary $\boldsymbol{\Omega} \in \mathbb{R}^{m \times n}$ contains m row atoms $\{\boldsymbol{\omega}_i^T \in \mathbb{R}^{1 \times n}\}_{i=1}^m$ with $m > n$. The expectation is that an input signal $\boldsymbol{x} \in \mathbb{R}^n$ can be sparsely represented by a vector $\boldsymbol{\gamma}_a \in \mathbb{R}^m$ (the subscript a represents analysis sparse model) with respect to an analysis dictionary. That is, the analysis coefficients form a sparse vector with many zero entries:

$$\boldsymbol{\gamma}_a = \boldsymbol{\Omega}\boldsymbol{x} \quad \text{with } \|\boldsymbol{\gamma}_a\|_0 = p < m. \tag{2.10}$$

Fig. 2.4 shows an example of the analysis sparse signal model. Different from the synthesis sparse signal model, both the non-zero coefficients and the zero coefficients are informative for the representation of the input signal. With a proper analysis dictionary, the row atoms corresponding to the non-zero coefficients represent a low-dimensional subspace in which the input signal lies. This leads to a discriminative sparse representation for the input signal. From another perspective, the row atoms with zero coefficients define an orthogonal subspace of the input signal and therefore define its signal subspace which can be used to remove noisy components in the input signal.

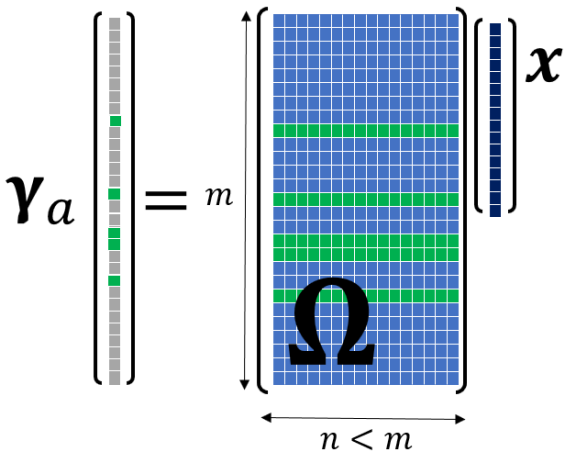


Figure 2.4: **A analysis sparse signal model.** The expectation is that the atoms of the analysis dictionary $\boldsymbol{\Omega}$ is able to sparsity the input signal. The sparse representation γ_a should be able to fully represent the information within the input signal and is more discriminative than the original signal.

Co-Sparse Analysis Model:

An analysis dictionary can be used to regularize the signal to be estimated. The row atoms of the analysis dictionary define the directions in the signal subspace where only a small portion of signals aligns to. Given an analysis dictionary learned from a set of training samples, a signal sampled from the same distribution as the training samples should be able to have sparse representation. An optimization problem with a sparsity constraint on the analysis coefficient $\boldsymbol{\Omega x}$ would favor the solution $\boldsymbol{x}$ that is within the manifold the training samples belong to. The analysis sparse pursuit aims to find a signal $\widehat{\boldsymbol{x}}$ whose inner product with $\boldsymbol{\Omega}$ is sparse and subject to a data fidelity constraint:

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{\Omega x}\|_0, \quad \text{s.t. } \boldsymbol{y} = \boldsymbol{H x} + \boldsymbol{n}, \tag{2.11}$$

where $\boldsymbol{H}$ is a linear matrix that relates the measured signal and the ground-truth signal, and $\boldsymbol{n}$ is a noise vector.

Let us define the co-support Λ of $\boldsymbol{\Omega}$ and $\boldsymbol{x}$ as a set of l row atoms of $\boldsymbol{\Omega}$ that $\boldsymbol{x}$ is orthogonal to and define matrix $\boldsymbol{\Omega}_\Lambda$ as the sub-matrix of $\boldsymbol{\Omega}$ constructed with atoms

indicated by Λ. The variable l is referred to the co-sparsity of $\boldsymbol{\Omega}$ and $\boldsymbol{x}$. By definition, the sub-matrix $\boldsymbol{\Omega}_\Lambda$ can annihilate the input signal $\boldsymbol{x}$:

$$\boldsymbol{\Omega}_\Lambda \boldsymbol{x} = \mathbf{0}. \tag{2.12}$$

The sub-matrix $\boldsymbol{\Omega}_\Lambda$ defines the subspace that $\boldsymbol{x}$ is orthogonal to, therefore we can retrieve the orthogonal complementary subspace that $\boldsymbol{x}$ belongs to. Let us assume the linear matrix is an identity matrix i.e. $\boldsymbol{H} = \mathbf{I}$. The signal $\widehat{\boldsymbol{x}}$ can therefore be recovered from a noisy observation $\boldsymbol{y}$ when the oracle co-support Λ is known:

$$\widehat{\boldsymbol{x}} = (\mathbf{I} - \boldsymbol{\Omega}_\Lambda^\dagger \boldsymbol{\Omega}_\Lambda)\boldsymbol{y}, \tag{2.13}$$

where $\mathbf{I}$ is an identity matrix, $\boldsymbol{\Omega}_\Lambda^\dagger$ is the pseudo-inverse of $\boldsymbol{\Omega}_\Lambda$, and the noisy signal is assumed to be in the form $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$ with $\boldsymbol{n}$ being a zero-mean i.i.d. Gaussian noise vector.

The analysis sparse pursuit problem is difficult to solve. It requires a joint estimation of the signal $\widehat{\boldsymbol{x}}$ and the co-support $\widehat{\Lambda}$. Given a noisy observation $\boldsymbol{y}$ and an analysis dictionary $\boldsymbol{\Omega}$, the analysis sparse pursuit [60] aims to simultaneously find the co-support $\widehat{\Lambda}$ and estimate the signal $\widehat{\boldsymbol{x}}$ such that $\boldsymbol{\Omega}_{\widehat{\Lambda}}\widehat{\boldsymbol{x}} = \mathbf{0}$. The sparse pursuit problem for the co-sparse analysis model can be formulated as:

$$\begin{aligned} &\{\widehat{\boldsymbol{x}}, \widehat{\Lambda}\} = \arg\min_{\boldsymbol{x},\Lambda} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2, \\ &\text{s.t. } \boldsymbol{\Omega}_\Lambda \boldsymbol{x} = \mathbf{0} \text{ and } \mathrm{Rank}(\boldsymbol{\Omega}_\Lambda) = n - r_s, \end{aligned} \tag{2.14}$$

where the signal $\boldsymbol{x}$ is assumed to be within a subspace of $\mathbb{R}^n$ of dimension r_s.

Greedy analysis pursuit algorithms [60, 66] iteratively estimate the entries of the co-support set. Backward-Greedy algorithm [60] iteratively finds an atom that has the minimum correlation with the $\widehat{\boldsymbol{x}}$ estimated using Eqn. (2.13) and adds it into the co-

support set. The Optimized-Backward-Greedy algorithm [60] replaces the minimum correlation criterion and tries to find an atom that leads to the smallest decrease in signal energy when used for signal recovery as in Eqn. (2.13). Greedy Analysis Pursuit algorithm [66] iteratively removes an atom from the co-support set that has the maximum correlation with the current estimated signal.

Analysis Sparse Model and Element-wise Thresholding:

The analysis coefficients $\boldsymbol{\Omega x}$ is within the column space of the analysis dictionary. It is a linear transform of the input signal and usually not sparse. With an augmented Lagrangian, Eqn. (2.10) can be formulated into an unconstrained form which has a closed-form solution using hard-thresholding:

$$\begin{aligned} \boldsymbol{\gamma}_a &= \arg\min_{\boldsymbol{\gamma}} \frac{1}{2}\|\boldsymbol{\gamma} - \boldsymbol{\Omega x}\|_2^2 + \lambda\|\boldsymbol{\gamma}\|_0, \\ &= \mathcal{H}_\lambda(\boldsymbol{\Omega x}), \end{aligned} \tag{2.15}$$

where λ is the regularization parameter, and $\mathcal{H}_\lambda(\cdot)$ is an element-wise hard-thresholding operator with $\mathcal{H}_\lambda(a) = a\mathbf{1}_{|a|>\lambda}$.

By relaxing the l_0-norm to the l_1-norm, the sparse representation of a signal with respect to an analysis dictionary can be obtained by solving a l_1-norm minimization problem which has a closed-form solution using soft-thresholding:

$$\begin{aligned} \boldsymbol{\gamma}_a &= \arg\min_{\boldsymbol{\gamma}} \frac{1}{2}\|\boldsymbol{\gamma} - \boldsymbol{\Omega x}\|_2^2 + \lambda\|\boldsymbol{\gamma}\|_1, \\ &= \mathcal{S}_\lambda(\boldsymbol{\Omega x}), \end{aligned} \tag{2.16}$$

where λ is the regularization parameter, and $\mathcal{S}_\lambda(\cdot)$ is an element-wise soft-thresholding operator with $\mathcal{S}_\lambda(a) = \mathrm{sgn}(a)\max(|a| - \lambda, 0)$.

Hard-thresholding and the soft-thresholding are element-wise operations with low computational complexity. The sparse representation $\boldsymbol{\gamma}_a$ can be obtained efficiently using

a thresholding operation with threshold λ. However, the optimization problem in Eqn. (2.15) and Eqn. (2.16) does not impose a reconstruction constraint. The sparse representation $\boldsymbol{\gamma}_a$ may not be able to reconstruct the input signal. The quality of the sparse representation depends on the choice of the regularization parameter λ.

Fig. 2.4 shows an example of the analysis sparse signal model. The expectation is that the analysis coefficients $\boldsymbol{\gamma}_a$ are sparse, whilst preserving essential information for reconstruction. To achieve that, the analysis dictionary should be redundant and have a strong sparsifying ability. For example, when a signal $\boldsymbol{x}$ is within the span of a subspace defined by a small number of row atoms, it can be well represented by these group of atoms and all the other coefficients can be set to zero.

Analysis Dictionary Learning:

The analysis dictionary usually serves as a regularization term $\lambda||\boldsymbol{\Omega x}||_1$ in the optimization formulation and models the co-sparse prior which can be considered as an extension of the Total Variation (TV) prior.

Alternating minimization strategy can also be applied for analysis dictionary learning [60, 64, 65, 67, 68]. Analysis K-SVD [60] algorithm iterates between an analysis pursuit operation and a K-SVD dictionary update stage. Yaghoobi *et al.* [65] proposed a uniformly-normalized tight frame constraint for learning analysis operators. Analysis Simultaneous Codeword Optimization (ASimCO) algorithm [64] enforces a k-sparse constraint on the sparse-coding stage and updates multiple dictionary atoms simultaneously in the dictionary update stage. Sparsifying transform learning [67, 68] proposed to constrain the analysis operator to be full rank and well-conditioned.

The GeOmetric Analysis Operator Learning (GOAL) algorithm [63,69] learns the analysis dictionary by employing an alternative optimization strategy. It performs dictionary learning on manifolds by minimizing an objective function which promotes sparse

representation and also imposes full rank and linear independence constraints.

2.2.3 Convolutional Sparse Model

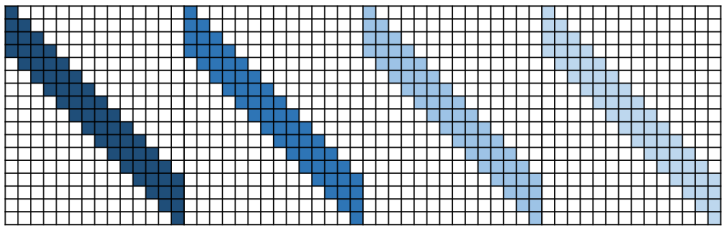
The sparse signal models that are based on unstructured dictionaries cannot handle high dimensional signals properly mainly due to the computational constraints on both the sparse pursuit process and the dictionary learning process. When processing high dimensional signals, the sparse representation methods usually divide the input signal into overlapping low-dimensional signal segments (i.e. patches) and perform sparse modelling on each low dimensional patch. This patch-based approach implicitly assumes that each patch is independent from the other patches and ignores the interactions between neighbouring patches. In the case of images, this may result in blocking artifacts on the processed signal due to the partition process.

Convolutional sparse models use convolutional dictionary. The convolutional dictionary makes the assumption that the filters are shift invariant and the statistics of the signal are similar at different locations. The convolutional dictionary explicitly models the interactions between the neighbouring signal segments through sharing dictionary atoms and sparse representation coefficients. Compared to an unstructured dictionary of the same size, the convolutional dictionary has a much smaller number of free parameters.

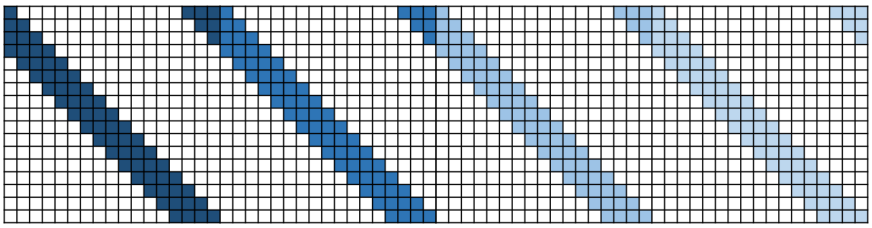
Convolutional Synthesis Sparse Model:

In the convolutional synthesis sparse model, a global signal $\boldsymbol{x} \in \mathbb{R}^n$ is represented as follows:

$$\boldsymbol{x} = \sum_{i=1}^{m} \boldsymbol{d}_i * \boldsymbol{\gamma}_{cs,i}, \quad \text{s.t.} \ \sum_i \|\boldsymbol{\gamma}_{cs,i}\|_0 \leq k, \tag{2.17}$$



(a) Convolutional synthesis dictionary with Toeplitz structure.



(b) Convolutional synthesis dictionary with circular structure.

Figure 2.5: **Convolutional synthesis dictionary with Toeplitz structure and circulant structure.** A convolutional synthesis dictionary $\mathcal{S}(\boldsymbol{D}, n)$ is a concatenation of Toeplitz matrices. Each Toeplitz matrix corresponds to a convolutional filter.

where $\boldsymbol{d}_i$ is the i-th filter, and $\boldsymbol{\gamma}_{cs,i}$ is the corresponding i-th sparse vector (the subscript cs represents convolutional synthesis sparse model).

In a convolutional model, the synthesis dictionary $\boldsymbol{D} = [\boldsymbol{d}_1, \cdots, \boldsymbol{d}_m] \in \mathbb{R}^{s \times m}$ is a collection of all the filters. The global sparse representation $\boldsymbol{\gamma}_{cs} = [\boldsymbol{\gamma}_{cs,1}; \cdots; \boldsymbol{\gamma}_{cs,m}]$ is the concatenation of the sparse vectors and is assumed to have a sparsity level k. The convolutional filters are usually assumed to have compact support i.e. $s \ll n$. Different from the unstructured case, the synthesis dictionary in a convolutional matrix and is not necessary that it is over-complete.

A convolution can be represented by multiplying a signal with a Toeplitz matrix which is a structured matrix with constant values along the diagonals:

$$\boldsymbol{d}_i * \boldsymbol{\gamma}_{sc,i} = \mathbf{T}(\boldsymbol{d}_i, n)\boldsymbol{\gamma}_{sc,i}, \tag{2.18}$$

where $\mathbf{T}(\boldsymbol{d}_i, n)$ is a Toeplitz matrix with n rows which is constructed using $\boldsymbol{d}_i$ as follows:

$$\mathbf{T}(\boldsymbol{d}_i, n) = \sum_{j=1}^{n} \boldsymbol{d}_i(j)\mathbf{T}_{Lj}, \tag{2.19}$$

where $\boldsymbol{d}_i(j)$ is the j-th coefficient of $\boldsymbol{d}_i$, and $\mathbf{T}_{Lj}$ is an indicator matrix with 1s on the j-th lower diagonal and 0s on other locations.

A convolutional synthesis dictionary is a concatenation of Toeplitz matrices in which each Toeplitz matrix is constructed using a filter. When the convolution between the filter and the input signal is assumed to be a circular convolution, each filter will lead to a circulant matrix which is a special case of Toeplitz matrix and assumes periodic boundary condition on the signals.

Fig. 2.5 shows two examples of the convolutional synthesis dictionary. In Fig. 2.5(a) the convolutional synthesis dictionary is a concatenation of Toeplitz matrices. The corresponding convolution operation is a discrete time convolution. In Fig. 2.5(b) the convolutional synthesis dictionary is a concatenation of circulant matrices which assume a circular convolution with periodic boundary condition. Circular convolution is usually applied to achieve a fast convolution.

With a matrix form representation, the global signal $\boldsymbol{x}$ can be represented by a global sparse representation $\boldsymbol{\gamma}_{cs}$ with respect to a convolutional synthesis dictionary $\mathcal{S}(\boldsymbol{D})$:

$$\begin{aligned} &\boldsymbol{x} = \sum_{i=1}^{m} \mathbf{T}(\boldsymbol{d}_i, n)\boldsymbol{\gamma}_{cs,i} = \mathcal{S}(\boldsymbol{D}, n)\boldsymbol{\gamma}_{cs}, \\ &\text{s.t. } \|\boldsymbol{\gamma}_{cs}\|_0 \leq k, \end{aligned} \tag{2.20}$$

where $\mathcal{S}(\boldsymbol{D}, n) = [\mathbf{T}(\boldsymbol{d}_1, n), \cdots, \mathbf{T}(\boldsymbol{d}_m, n)]$ denotes the convolutional synthesis dictionary which is a concatenation of Toeplitz matrices $\{\mathbf{T}(\boldsymbol{d}_i, n)\}_{i=1}^{m}$.

The Convolutional Sparse Coding (CSC) problem aims to find the global sparse representation $\boldsymbol{\gamma}_{cs}$ of an input signal $\boldsymbol{x}$ with respect to the convolutional dictionary $\mathcal{S}(\boldsymbol{D}, n)$

and can be expressed as:

$$\boldsymbol{\gamma}_{cs} = \arg\min_{\boldsymbol{\gamma}} \|\boldsymbol{\gamma}\|_0, \quad \text{s.t. } \boldsymbol{x} = \mathcal{S}(\boldsymbol{D}, n)\boldsymbol{\gamma}. \tag{2.21}$$

With a convex relaxation, the CSC problem can be represented as a constrained l_1 minimization problem:

$$\boldsymbol{\gamma}_{cs} = \arg\min_{\boldsymbol{\gamma}} \frac{1}{2}\|\boldsymbol{x} - \mathcal{S}(\boldsymbol{D})\boldsymbol{\gamma}\|_2^2 + \lambda\|\boldsymbol{\gamma}\|_1, \tag{2.22}$$

where λ is the regularization parameter.

When the convolution is assumed to be circular, the convolutional dictionary can be represented as circulant matrices which can be diagonalized by the Discrete Fourier Transform (DFT) matrix. The l_1-norm CSC problem is usually efficiently solved by applying Alternating Direction Method of Multipliers (ADMM) [70]. The optimization can be performed in the signal domain [71, 72] or in the Fourier domain [73, 74].

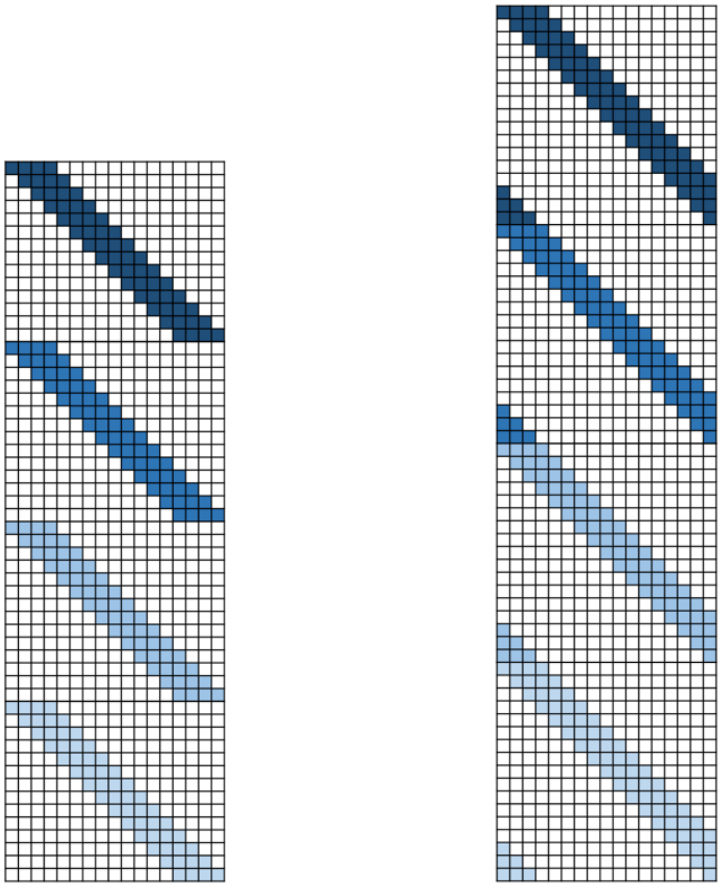
Convolutional Analysis Sparse Model:

The analysis sparse model can also be extended to the convolutional case. The analysis coefficients $\boldsymbol{\gamma}_{ca}$ are the result of convolving several filters with the input signal $\boldsymbol{x} \in \mathbb{R}^n$:

$$\begin{aligned} \boldsymbol{\gamma}_{ca} =& [\boldsymbol{\omega}_1 * \boldsymbol{x}; \cdots; \boldsymbol{\omega}_m * \boldsymbol{x}], \\ =& [\mathbf{T}(\boldsymbol{\omega}_1^T, n)\boldsymbol{x}; \cdots; \mathbf{T}(\boldsymbol{\omega}_m^T, n)\boldsymbol{x}], \end{aligned} \tag{2.23}$$

where $\boldsymbol{\omega}_i^T \in \mathbb{R}^s$ is the i-th filter with $s \ll n$, $\mathbf{T}(\boldsymbol{\omega}_i^T, n)$ is a Toeplitz matrix constructed using the i-th filter and is defined as follows:

$$\mathbf{T}(\boldsymbol{\omega}_i^T, n) = \sum_{j=1}^{n} \boldsymbol{\omega}_i(j)\mathbf{T}_{Uj}, \tag{2.24}$$



(a) Convolutional analysis dictionary with Toeplitz structure.

(b) Convolutional analysis dictionary with circular structure.

Figure 2.6: **Convolutional analysis dictionary with Toeplitz structure and circulant structure.** A convolutional analysis dictionary $\mathcal{H}(\boldsymbol{\Omega})$ is a concatenation of Toeplitz matrices. Each Toeplitz matrix corresponds to a convolutional filter.

where $\boldsymbol{\omega}_i(j)$ is the j-th coefficient of $\boldsymbol{\omega}_i$, and $\mathbf{T}_{Uj}$ is an indicator matrix with 1s on the j-th upper diagonal and 0s on other locations.

A convolutional analysis dictionary $\mathcal{H}(\boldsymbol{\Omega}, n) = [\mathbf{T}(\boldsymbol{\omega}_1^T, n); \cdots ; \mathbf{T}(\boldsymbol{\omega}_m^T, n)]$ is a concatenation of Toeplitz matrices along the column direction. When the convolution is a circulant convolution, circulant matrix will be used to represent the convolution operation. Fig. 2.6 shows two examples of convolutional analysis dictionary. In Fig. 2.6(a), convolutional analysis dictionary is a concatenation of Toeplitz matrices. In Fig. 2.6(b), the convolutional analysis dictionary is a concatenation of circulant matrices.

2.2.4 Convolutional Dictionary Learning:

When the convolutional dictionary is modelled as a concatenation of circulant matrices, convolutional dictionary learning can be operated on Fourier domain as circulant matrices can be diagonalized by the discrete Fourier transform (DFT) matrix.

Convolutional synthesis dictionary learning has attracted increasing research interests. Bristow *et al.* [73] proposed a fast convolutional synthesis dictionary learning algorithm which is based on Alternating Direction Method of Multipliers (ADMM) [70] and solves the convolution sub-problem in Fourier domain. Wohlberg [75] proposed to accelerate the convolutional synthesis sparse pursuit (i.e. convolutional sparse coding (CSC)) in Fourier domain. The circular boundary condition which is implicitly imposed with the use of Fourier domain operations may introduce reconstruction artifacts. Heide *et al.* [74] proposed a flexible framework for solving CSC problem which allows proper boundary conditions to be imposed rather than assuming a circular periodic boundary condition. Chun *et al.* [76] proposed a block proximal gradient method for convolutional dictionary learning which does not need hyper-parameter tuning.

Convolutional analysis dictionary learning has also started to attract more research

interests. Pfister and Bresler [68] proposed a filter bank sparsifying transform learning algorithm which is also based on Fourier domain representation and is able to learn a sparsifying convolutional analysis dictionary. Chun *et al.* [77] proposed a convolutional analysis operator learning framework which learns convolutional sparsifying regularizer.

2.3 Deep Neural Networks

Deep Neural Networks (DNNs) are architectures composed of multiple layers of linear transforms and non-linear operators. DNNs are highly non-linear and non-convex systems due to the non-linear layers and the multi-layer structure. Therefore, it is difficult to directly optimize all the parameters of a DNN. The parameters of the DNNs are usually optimized by backpropagation algorithm [20] which is based on gradient descent algorithms and the chain rule of derivatives.

With the help of massive labeled training data and powerful Graphics Processing Units (GPU), DNNs have achieved outstanding performance in many signal processing and computer vision tasks. The success of DNNs comes from its ability to learn representation from raw data and the end-to-end optimization strategy. DNNs learn representations from raw data and the feature representation at a deep layer is expressed in terms of the simpler representations at the shallow layers. Therefore, DNNs progressively learn more complex concepts and the feature representations possess with multiple levels of abstraction.

2.3.1 A Brief Introduction

The forward pass of a Deep Neural Network is a cascade of multiple layers of transformations. Let us consider an L-layer DNN, we denote with $\boldsymbol{\theta}_i$ the parameters of the i-th layer and with $\boldsymbol{x}_i$ the output of the i-th layer. The forward model of the i-th layer

of a DNN is a function of the output $\boldsymbol{x}_{i-1}$ from the previous layer and the parameters $\boldsymbol{\theta}_i$:

$$\boldsymbol{x}_i = f_i(\boldsymbol{x}_{i-1}, \boldsymbol{\theta}_i). \tag{2.25}$$

where $f_i(\cdot, \cdot)$ is assumed to be differentiable with respect to its inputs.

Therefore, the forward pass of the L-layer DNN can be expressed as:

$$\boldsymbol{x}_L = f_L\left(f_{L-1}\left(\cdots f_1\left(\boldsymbol{x}_0, \boldsymbol{\theta}_1\right)\cdots, \boldsymbol{\theta}_{L-1}\right), \boldsymbol{\theta}_L\right). \tag{2.26}$$

During the training stage, the parameters of a DNN $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i=1}^{L}$ will be learned from a set of training samples. A loss function $\mathcal{L}(\cdot, \cdot)$ defines the distance between the estimated signal $\boldsymbol{x}_L$ and the ground-truth signal $\boldsymbol{y}$. The learning objective is to minimize the loss E over a set of paired training samples $\{\boldsymbol{x}_{0,j}, \boldsymbol{y}_j\}_{j=1}^{N}$:

$$E = \frac{1}{N}\sum_{j=1}^{N} \mathcal{L}(\boldsymbol{x}_{L,j}, \boldsymbol{y}_j). \tag{2.27}$$

Backpropagation:

Backpropagation algorithm [20] is usually applied for learning the parameters of Deep Neural Networks. By exploiting the chain rule of derivatives, backpropagation algorithm enables us to simultaneously update all parameters across layers given that we have access to the derivatives of each parameter.

Given a set of paired training samples $\{\boldsymbol{x}_{0,j}, \boldsymbol{y}_j\}_{j=1}^{N}$, the loss E can be evaluated with the forward model of the DNN defined in Eqn. (2.26) and the loss function defined in Eqn. (2.27). The gradient descent based optimization methods try to iteratively update the parameters by stepping towards the negative direction of the gradient:

$$\boldsymbol{\theta}_i^t = \boldsymbol{\theta}_i^{t-1} - \eta\frac{\partial E}{\partial \boldsymbol{\theta}_i}, \tag{2.28}$$

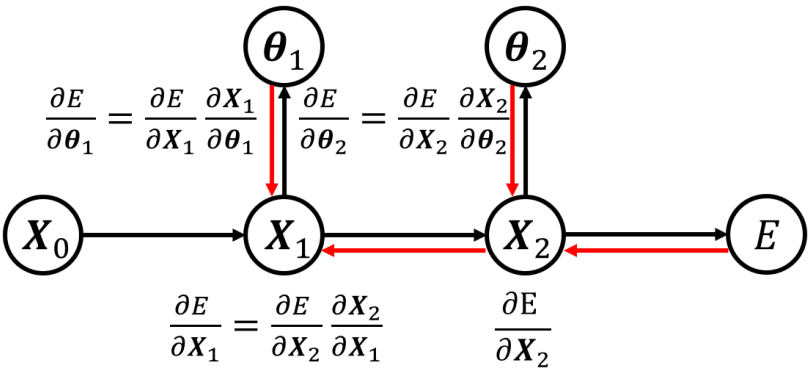


Figure 2.7: **The computation graph of a 2-layer neural network.** In the graph, $\boldsymbol{X}_0$ is the input signal, $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are the representation at the 1-st layer and 2-nd layer, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are the parameters of two layers, and E denotes the loss. The black arrows stand for the forward pass and the red arrows stand for the backpropagation.

where η is the step size (i.e. the learning rate).

Fig. 2.7 shows an example of the computation graph of a 2-layer neural network with both the forward pass and the backpropagation update. The backpropagation algorithm exploits the chain rule of the partial derivatives. Let us denote with $\boldsymbol{X}_i = [\boldsymbol{x}_{i,1}, \cdots, \boldsymbol{x}_{i,N}]$ the matrix containing the outputs at the i-layer for all training samples. The derivatives of the i-th layer parameter $\boldsymbol{\theta}_i$ can be evaluated as follows:

$$\frac{\partial E}{\partial \boldsymbol{\theta}_i} = \frac{\partial E}{\partial \boldsymbol{X}_i}\frac{\partial \boldsymbol{X}_i}{\partial \boldsymbol{\theta}_i}, \tag{2.29}$$

$$\frac{\partial E}{\partial \boldsymbol{X}_i} = \frac{\partial E}{\partial \boldsymbol{X}_{i+1}}\frac{\partial \boldsymbol{X}_{i+1}}{\partial \boldsymbol{X}_i}, \tag{2.30}$$

where the derivative $\partial \boldsymbol{X}_i / \partial \boldsymbol{\theta}_i$ and $\partial \boldsymbol{X}_{i+1} / \partial \boldsymbol{X}_i$ depend on the function $f_i(\cdot, \cdot)$ and $f_{i+1}(\cdot, \cdot)$, respectively.

Activation Function:

Deep Neural Networks consist of linear and non-linear layers. The function of the non-linear layers is to provide a non-linear mapping from the input signal to the desired output signal. When there is no non-linear layers, the DNN is a deep linear network

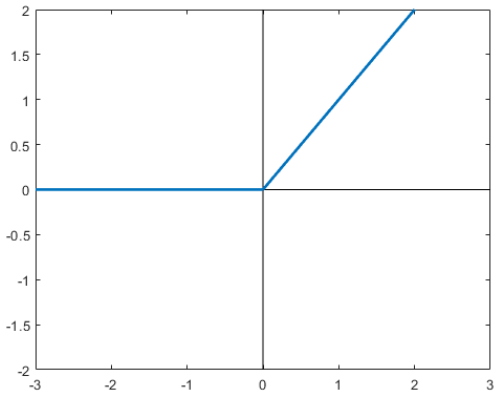


Figure 2.8: **Rectified Linear Unit (ReLU) activation function.**

which models a linear relationship from the input to the output. A deep linear network has no advantage over a shallow linear model in terms of expressive power. For computational consideration, the non-linear layers are usually characterized by activation functions which are usually element-wise operations and are (approximately) differentiable.

The Rectified Linear Unit (ReLU) is one of the most popular activation functions used in modern DNNs due to its simplicity and connections with sparse modelling. A ReLU activation function will keep the input when the input is positive, and will output zero otherwise:

$$\mathrm{ReLU}(x) = \max(x, 0). \tag{2.31}$$

The ReLU activation function and its derivatives can be efficiently computed. In the forward pass, only a comparison operation will be involved. The first derivative of the ReLU activation function is 1 when the input is positive and 0 otherwise.

The ReLU activation function produces sparse feature representations which lead to a better interpretation of the workings of DNNs. As a ReLU activation function output 0 or 1, it partitions the input space into two parts. The DNNs with ReLU activation functions (also known as a deep rectifier networks) partitions the signal space into a number of linear regions in which a linear transform is applied for prediction.

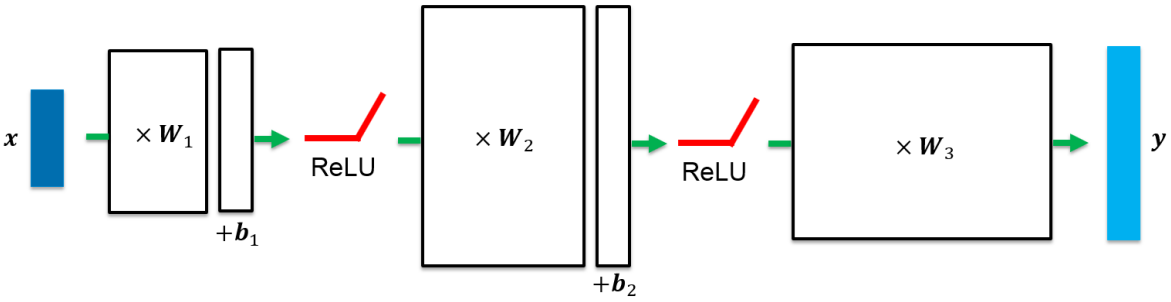


Figure 2.9: **A 3-layer multilayer perceptron with ReLU activation function.**

A non-zero coefficient can be interpreted as selecting (activating) a feature representation represented by a neuron. The ReLU activation function can be considered as a one-sided soft-thresholding function. This leads to a connection with the l_1-norm minimization problem.

There are many variations of ReLU activation function including leaky ReLU which has a slope $\alpha = 0.05$ when the input is negative and parametric ReLU which has learnable slope α. These modifications lead to improved performances.

2.3.2 Multilayer Perceptron

A Multilayer Perceptron (MLP) is a class of Artificial Neural Networks (ANNs) in which every neuron is connected to all the neurons in its next layer. There is at least one hidden layer (i.e. a layer of neurons with non-linear activation functions) in MLP. The connections between a neuron and all the neurons in the next layer can be represented by a vector where each value represents the weight. Therefore, a weight matrix $\boldsymbol{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}$ can be used to model the relationship between the neurons in the $(i-1)$-th layer and the neurons in the i-th layer with d_{i-1} and d_i neurons, respectively. There is a layer of activation functions between two layers as non-linearity. A bias vector $\boldsymbol{b}_i$ is used to represent the parameters of the activation functions for each neuron at layer i. Backpropagation algorithm [20] is usually applied for learning all weight matrices and bias vectors in a MLP.

Fig. 2.9 shows an example of a 3-layer MLP with ReLU activation functions. The parameters of the i-th layer include a weight matrix $\boldsymbol{W}_i$ and a bias vector $\boldsymbol{b}_i$. The weight matrix is the parameter of the linear layer, while the bias vector is the parameter of the ReLU activation function. The forward pass of the i-th layer of MLP can be expressed as follows:

$$\boldsymbol{x}_i = \mathrm{ReLU}(\boldsymbol{W}_i \boldsymbol{x}_{i-1} + \boldsymbol{b}_i). \tag{2.32}$$

The forward model of the MLP shown in Fig. 2.9 can be expressed as:

$$\boldsymbol{y} = \boldsymbol{W}_3 \mathrm{ReLU}(\boldsymbol{W}_2 \mathrm{ReLU}(\boldsymbol{W}_1 \boldsymbol{x}_0 + \boldsymbol{b}_1) + \boldsymbol{b}_2). \tag{2.33}$$

The MLP is usually referred to the "vanilla" neural networks. It has been widely used for theoretical study of DNNs. MLP is also an important tool for many applications where the input signal is low dimensional, such as simple classification or regression problem, or reinforcement learning problem. When the input signal is high dimensional, the number of parameters in a MLP would increase rapidly. This will result in a system with a huge number of parameters which is both computationally and memory inefficient.

2.3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [78, 79] are a class of Deep Neural Networks whose neurons are locally connected to the neurons in its next layer and share a small set of filters as the connection weights. CNNs are usually composed of convolutional layers, pooling layers, and fully connected layers. An example of convolutional neural network is shown in Fig. 2.10.

A convolutional layer contains a set of filters and the corresponding activation functions. Compared to a fully connected layer, a convolutional layer greatly reduces the number

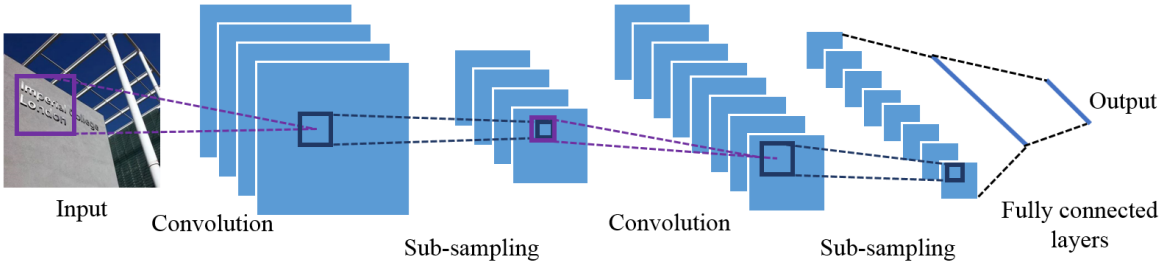


Figure 2.10: **An example of Convolutional Neural Networks.**

of parameters. Each filter will convolve with its feature representations output from the previous layer and generate a feature representation which will pass through an activation function for non-linear mapping. With the convolution operation, the feature representations at two adjacent layers can be considered as being locally "connected".

There is a close connection between the convolutional sparse model and CNN [72], since the operation of a convolution layer can be represented by multiplying the input feature with a convolutional dictionary.

A pooling layer is associated with a sub-sampling operation. The pooling layers are applied to reduce the spatial size of the feature representation and result in a reduction on the amount of computation. The pooling layer is also essential to extract translational invariant features for the image recognition application. Typical sub-sampling operations used in CNNs include max pooling and average pooling. The pooling operation is applied to a small spatial region on the feature maps, for example a 2×2 region. For example, a max pooling computes the max value within every small spatial region and retain the max value as the representation.

Chapter 3

Learning Deep Analysis Dictionary Models

3.1 Introduction

Deep Neural Networks (DNNs) [80] are complex architectures composed of a cascade of multiple linear and non-linear layers. Back-propagation algorithm [20] is usually applied to optimize the parameters of the linear transforms and the non-linearities within this highly non-linear and non-convex system. With the help of massive labeled training data and powerful Graphics Processing Units (GPU), DNNs have achieved outstanding performance in many signal processing and computer vision tasks. However, there still lacks a clear understanding about the workings of DNNs. The optimized DNNs are usually treated as black-box systems. Some natural questions are what are the functions of the linear transform and the non-linearities and what is the role of the "cascade" in DNNs.

Some recent works have tried to provide insights into the workings of DNNs. Bruna and Mallat [81, 82] proposed a Scattering Convolutional Network (SCN) by replacing the

learned filters with wavelet-like transforms. SCN provides feature representations which are translation and rotation invariant. Zeiler and Fergus [83] proposed a deconvolution technique to visualize the intermediate feature layers of a Convolutional Neural Network (CNN) trained for image classification. The filters in the first layer are Gabor like, and the deeper layer feature maps tend to be active only for certain objects. In [84], the authors suggested that an auto-encoder partitions the low-dimensional data manifold into cells and approximates each piece by a hyper-plane. The encoding capability limit of a DNN is described by the upper bound of the number of cells. Montufar *et al.* [85] have shown that the number of cells is exponentially larger than the degrees of freedom. Giryes *et al.* [86] theoretically analyzed the fully connected DNN with i.i.d. random Gaussian weights. They prove that a DNN with random Gaussian weights performs a distance-preserving embedding of the data.

Contrary to DNNs, the sparse representation framework [87] is much more established. Sparse representation over an over-complete dictionary can serve as an effective and robust representation in both classification and regression problems. Building a deep model using sparse representation over redundant synthesis dictionaries has facilitated interpretations of DNNs. Rubinstein and Elad [61] proposed an Analysis-Synthesis Thresholding (AST) model for image deblurring which consists of an analysis dictionary, element-wise hard-thresholding operators and a synthesis dictionary. The AST model can be interpreted as a fully connected DNN which is with one hidden layer and uses hard-thresholding as non-linearity. The Double-Sparsity model [88] proposes to learn a two-layer synthesis dictionary model. The first layer is a dictionary that models a fixed basis, while the second layer is a sparse dictionary with sparse atoms. The effective dictionary is the multiplication between the basis dictionary and the sparse dictionary. The Double-Sparsity model gives a more efficient and adaptive modeling and enables learning large dictionary from high-dimensional data. A Multi-Layer Convolutional Sparse Coding (ML-CSC) model is proposed in [72, 89] and gives a new interpretation on the working of the Convolutional Neural Networks (CNNs). The linear

models in CNNs are interpreted as synthesis dictionaries with convolutional structure and the function of the non-linearities is interpreted as a simplified sparse pursuit procedure. The ML-CSC model has multiple layers of synthesis dictionaries where the first dictionary is non-sparse while the following dictionaries are sparse. Tariyal *et al.* [90] proposed a greedy layer-wise deep dictionary learning method which performs synthesis dictionary learning layer-by-layer. A parametric approach is proposed in [91] to learn a deep dictionary for image classification tasks. The proposed dictionary learning method contains a forward pass which performs sparse coding with the given synthesis dictionaries and a backward pass which updates the dictionaries by gradient descent.

The contribution of this chapter is three-fold:

- We propose a Deep Analysis dictionary Model (DeepAM) framework which is composed of multiple layers of *analysis* dictionaries with associated soft-thresholding operators and a layer of synthesis dictionary. The forward model of DeepAM is a cascade of matrix multiplication and soft-thresholding operations which is similar to that of DNNs.

- We propose to characterize each analysis dictionary as a combination of two sub-dictionaries: an Information Preserving Analysis Dictionary (IPAD) and a Clustering Analysis Dictionary (CAD). The IPAD together with the soft-thresholding operator preserves the key information of the input data, while the CAD with the associated soft-thresholding operator generates a discriminative representation.

- We propose learning algorithms for DeepAM. To achieve the two different goals of IPAD and CAD, different learning criteria have been proposed. We validate our proposed DeepAM on single image super-resolution task. Simulation results show that the proposed deep dictionary model achieves comparable performance with a DNN which has the same structure but is optimized using back-propagation.

The rest of the chapter is organized as follows. Section 3.2 gives an overview of the pro-

posed deep analysis dictionary model. Section 3.3 analyzes the proposed deep analysis dictionary model and explains the rationals behind splitting each analysis dictionary into an information preserving and a clustering sub-dictionary. Section 3.4 introduces the learning algorithm for the deep analysis dictionary model. Section 3.5 presents simulation results on single image super-resolution task and Section 3.6 draws conclusions.

3.2 Overview

The motivation of this chapter is to build a deep dictionary model which is with multiple layers of dictionary and has interpretable linear transform and non-linearity in each layer. The single image super-resolution problem is taken as a case study for validation.

3.2.1 Image Super-Resolution

In image super-resolution, the low-resolution (LR) image $\boldsymbol{X}$ is assumed to be the blurred and down-sampled version of a high-resolution (HR) image $\boldsymbol{Y}$. The task of single image super-resolution (SISR) is to estimate a HR image $\widehat{\boldsymbol{Y}}$ from an observed LR image $\boldsymbol{X}$. SISR is ill-posed as there is ambiguity when recovering the HR signals from their low-dimensional observations.

Patch-based single image super-resolution is used as a sample application to validate the proposed method. Instead of estimating the HR image as a whole, the patch-based approaches [1, 2, 7–11] divide the LR image into overlapping patches and infer a HR patch for each LR patch. The HR image can then be reconstructed using the estimated HR patches by patch overlapping. The learning-based methods [1, 2, 7, 9–11, 14–16] learn the inference model from an external training dataset which contains LR and HR

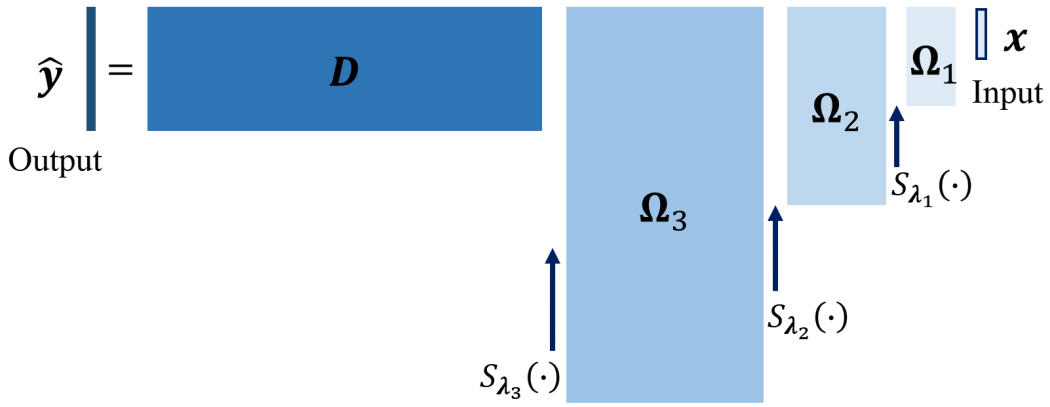


Figure 3.1: **A 3-layer deep analysis dictionary model.** There are 3 layers of analysis dictionaries $\{\boldsymbol{\Omega}_i\}_{i=1}^3$ with element-wise soft-thresholding operators $\{\mathcal{S}_{\boldsymbol{\lambda}_i}(\cdot)\}_{i=1}^3$ and a layer of synthesis dictionary $\boldsymbol{D}$. The output signal $\widehat{\boldsymbol{y}}$ is obtained through a cascade of matrix multiplications and soft-thresholding operations with input signal $\boldsymbol{x}$.

image pairs as training data. The patch-based methods extract LR-HR patch pairs $\{(\boldsymbol{x}_i^0, \boldsymbol{y}_i)\}_{i=1}^N$ from the training dataset. The size of the LR patches and the HR patches is assumed to be $p \times p$ and $(s \times p) \times (s \times p)$, respectively. The variable s represents the up-scaling factor. To gain illumination invariant property, the mean value of each patch is normally removed. For simplicity, let us denote $d_0 = p^2$ and $d_{L+1} = (s \times p)^2$. By vectorizing the image patches into vectors and grouping the training vectors into a matrix, we denote the input LR training data matrix as $\boldsymbol{\mathcal{X}}^0 = [\boldsymbol{x}_1^0, \cdots, \boldsymbol{x}_N^0] \in \mathbb{R}^{d_0 \times N}$ and its corresponding ground-truth HR training data matrix as $\boldsymbol{\mathcal{Y}} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_N] \in \mathbb{R}^{d_{L+1} \times N}$.

3.2.2 Deep Analysis Dictionary Model

We propose a Deep Analysis dictionary Model (DeepAM) which is made of multiple layers of analysis dictionary interleaved with soft-thresholding operations and a single synthesis dictionary. In an L-layer DeepAM, there are $L+1$ dictionaries and L layers that correspond to the non-linear operations. The first L dictionaries are analysis dictionaries and are denoted as $\left\{\boldsymbol{\Omega}_i \in \mathbb{R}^{d_i \times d_{i-1}}\right\}_{i=1}^L$ with $d_i \geq d_{i-1}$. The row atoms $\left\{\boldsymbol{\omega}_{i,j}^T\right\}_{j=1}^{d_i}$ of the analysis dictionary $\boldsymbol{\Omega}_i$ are assumed to be of unit norm. The non-

linear operator used in DeepAM is the element-wise soft-threshold $\{\mathcal{S}_{\boldsymbol{\lambda}_i}(\cdot)\}_{i=1}^{L}$[1] where $\boldsymbol{\lambda}_i \in \mathbb{R}^{d_i}$ denotes the threshold vector at layer i. The dictionary $\boldsymbol{D}$ in the last layer is a synthesis dictionary and is designed to optimize the regression task at hand. Fig. 3.1 shows an example of a 3-layer deep analysis dictionary model for image super-resolution task.

The forward model of an L-layer DeepAM can therefore be expressed as:

$$\widehat{\boldsymbol{y}} = \boldsymbol{D}\mathcal{S}_{\boldsymbol{\lambda}_L}\left(\boldsymbol{\Omega}_L\mathcal{S}_{\boldsymbol{\lambda}_{L-1}}\left(\cdots\boldsymbol{\Omega}_2\mathcal{S}_{\boldsymbol{\lambda}_1}\left(\boldsymbol{\Omega}_1\boldsymbol{x}^0\right)\cdots\right)\right), \tag{3.1}$$

where $\boldsymbol{x}^0$ and $\widehat{\boldsymbol{y}}$ is the input signal and the estimated output signal, respectively.

Let us denote with $\boldsymbol{x}^i = \mathcal{S}_{\boldsymbol{\lambda}_i}(\boldsymbol{\Omega}_i\boldsymbol{x}^{i-1})$ the output of the i-th layer. This means that the input signal $\boldsymbol{x}^{i-1}$ is multiplied with the analysis dictionary $\boldsymbol{\Omega}_i$ and then passed through the element-wise soft-thresholding operator $\mathcal{S}_{\boldsymbol{\lambda}_i}(\cdot)$. The thresholded output signal $\boldsymbol{x}^i$ will be a sparse signal and is expected to be better for predicting the ground-truth signal $\boldsymbol{y}$ than $\boldsymbol{x}^{i-1}$. After L layers of analysis dictionaries and thresholding, $\boldsymbol{x}^L$ is transformed to the HR signal domain via the synthesis dictionary. Note that the input LR signal lives in a lower dimensional space when compared to the target HR signal. It is essential for an inference model to be able to non-linearly transform the input data to a higher dimensional space. This is to be jointly achieved by the analysis dictionaries and the associated soft-thresholding operators.

The proposed DeepAM framework is closely related to Deep Neural Networks (DNNs) with Rectified Linear Unit (ReLU) non-linearity[2]. As ReLU can be considered as the one-sided version of soft-thresholding, there exists an equivalence between a layer of analysis dictionary with soft-thresholding and a layer of Neural Networks with ReLU:

$$\boldsymbol{\Omega}_{i+1}\mathcal{S}_{\boldsymbol{\lambda}_i}\left(\boldsymbol{\Omega}_i\boldsymbol{x}^{i-1}\right) = \boldsymbol{\Omega}_{i+1}^{\mathrm{HC}}\mathrm{ReLU}(\boldsymbol{\Omega}_i^{\mathrm{VC}}\boldsymbol{x}^{i-1}, \boldsymbol{\lambda}_i^{\mathrm{VC}}), \tag{3.2}$$

[1]Soft-thresholding is defined as $\mathcal{S}_\lambda(a) = \mathrm{sgn}(a)\max(|a| - \lambda, 0)$.
[2]ReLU is defined as $\mathrm{ReLU}(a, \lambda) = \max(a - \lambda, 0)$.

where $\boldsymbol{\Omega}_{i+1}^{\mathrm{HC}} = [\boldsymbol{\Omega}_{i+1}, -\boldsymbol{\Omega}_{i+1}]$, $\boldsymbol{\Omega}_{i}^{\mathrm{VC}} = [\boldsymbol{\Omega}_{i}; -\boldsymbol{\Omega}_{i}]$ and $\boldsymbol{\lambda}_{i}^{\mathrm{VC}} = [\boldsymbol{\lambda}_{i}; \boldsymbol{\lambda}_{i}]$. The superscripts HC and VC stand for horizontal concatenate and vertical concatenate, respectively.

From Eqn. (3.2), a layer of analysis dictionary and soft-thresholding operation with n atoms can be represented with a layer of Neural Networks with ReLU and $2n$ neurons. For data which is symmetrically distributed around the origin, DeepAM with soft-thresholding can be more efficient than DNNs with ReLU. This observation will be validated numerically in Section 3.5.

The learning problem for DeepAM can be formulated as learning the parameters that minimize the mean squared error between the ground-truth data and the estimations:

$$\min_{\boldsymbol{\theta}} \left\| \boldsymbol{\mathcal{Y}} - \boldsymbol{D}\mathcal{S}_{\boldsymbol{\lambda}_L}\left(\cdots \boldsymbol{\Omega}_2 \mathcal{S}_{\boldsymbol{\lambda}_1}\left(\boldsymbol{\Omega}_1 \boldsymbol{\mathcal{X}}^0\right)\cdots\right)\right\|_F^2, \tag{3.3}$$

where $\boldsymbol{\theta} = \left\{\boldsymbol{D}, \{\boldsymbol{\Omega}_i\}_{i=1}^{L}, \{\boldsymbol{\lambda}_i\}_{i=1}^{L}\right\}$ denotes all the parameters of the L-layer DeepAM.

Optimizing Eqn. (3.3) directly can be very difficult as it involves non-convex matrix factorization and learning parameters of the non-linear operators. Standard tools for optimizing DNNs can be utilized, for example back-propagation algorithm [20]. However this would lead to effective but difficult to interpret results.

Our objective instead is to build a deep dictionary model with a higher interpretability through understanding the purpose of different components in the model. The analysis dictionary and threshold pairs play a key role in DeepAM as they determine the way effective features are generated. The synthesis dictionary instead can be learned using least squares once all the analysis dictionaries and thresholds are fixed. We propose a layer-wise learning strategy to learn the pair of analysis dictionary and soft-thresholding operators. In this way, we can obtain a system where the purpose of every component is easier to interpret.

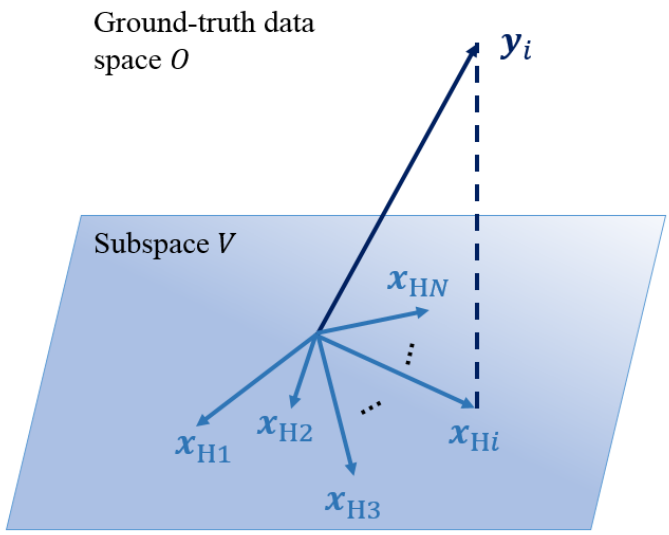


Figure 3.2: In image super-resolution, the input data spans a low-dimensional subspace V within the HR data space O. The objective is to estimate the unknown HR signal $\boldsymbol{y}_i$ based on the input LR signal $\boldsymbol{x}_{\mathrm{H}i}$. The dashed line represents the residual signal $\boldsymbol{r}_i = \boldsymbol{y}_i - \boldsymbol{x}_{\mathrm{H}i}$ which is orthogonal to the subspace spanned by the input data.

3.3 Analyzing the Deep Analysis Dictionary Model

To justify our approach, we begin by considering a 1-layer DeepAM system:

$$\widehat{\boldsymbol{y}}_i = \boldsymbol{D}\mathcal{S}_{\boldsymbol{\lambda}_1}(\boldsymbol{\Omega}_1 \boldsymbol{x}_i^0), \tag{3.4}$$

where $\boldsymbol{x}_i^0 \in \mathbb{R}^{d_0}$ is one of the elements of $\boldsymbol{\mathcal{X}}$, $\widehat{\boldsymbol{y}}_i \in \mathbb{R}^{d_2}$ with $d_2 > d_0$ and $\boldsymbol{\Omega}_1 \in \mathbb{R}^{d_1 \times d_0}$.

We assume, for the sake of argument, that the degradation model is linear. That is, there exists a degradation matrix $\boldsymbol{H} \in \mathbb{R}^{d_0 \times d_2}$ such that $\boldsymbol{x}_i^0 = \boldsymbol{H}\boldsymbol{y}_i$. Denote $\boldsymbol{x}_{\mathrm{H}i} = \boldsymbol{H}^\dagger \boldsymbol{x}_i^0 \in \mathbb{R}^{d_2}$ as the projection of the LR signal $\boldsymbol{x}^0$ onto the HR signal space with the pseudo-inverse matrix $\boldsymbol{H}^\dagger = \boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)^{-1}$.

As shown in Fig. 3.2, the LR signal $\boldsymbol{x}_{\mathrm{H}i}$ lies in a low-dimensional subspace $V \subset O$ of the ground-truth HR data space O. A linear operation will not be able to recover the components that are orthogonal to V (i.e. the dashed line in Fig. 3.2). It is therefore imperative to design the analysis dictionary $\boldsymbol{\Omega}_1$ and the non-linear soft-thresholding

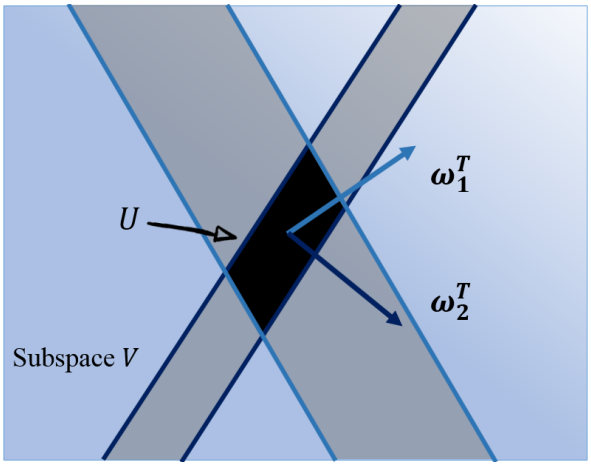


Figure 3.3: The analysis and soft-thresholding partitions the input data subspace V. There are two pairs of analysis atom and soft-thresholding operator. After soft-thresholding, the data in the gray region is with 1 zero coefficient and the data in the convex polyhedron U (i.e. the black region) is with all zero coefficients.

operation $\mathcal{S}_{\boldsymbol{\lambda}_1}(\cdot)$ in a way that facilitates the recovery of the information of $\boldsymbol{y}$ in $V^{\perp}$.

When we multiply $\boldsymbol{x}^0$ with $\boldsymbol{\Omega}_1$, the analysis dictionary atoms $\{\boldsymbol{\omega}_{1,j}^T\}_{j=1}^{d_1}$ project the input LR data onto specific directions. After soft-thresholding, the resulting signal $\boldsymbol{x}^1 = \mathcal{S}_{\boldsymbol{\lambda}_1}(\boldsymbol{\Omega}_1\boldsymbol{x}^0)$ is sparse and the end result is a partitioning of V as shown in Fig. 3.3. The analysis and soft-thresholding pairs partition the input data space into pieces (convex polyhedrons). Within each piece, the input data employs a linear transform for prediction. All the linear transforms are jointly determined by the synthesis dictionary $\boldsymbol{D}$.

We note that if we assume that all thresholds are large, there is a convex polyhedron U in which all input data will be set to zeros by the analysis and the thresholding operations, that is, $\mathcal{S}_{\boldsymbol{\lambda}_1}(\boldsymbol{\Omega}_1\boldsymbol{x}^0) = \mathbf{0}$, $\forall \boldsymbol{x}^0 \in U$ (see the central black region in Fig. 3.3). Therefore, the corresponding estimation $\widehat{\boldsymbol{y}}$ will be zero, and the information of the data within the convex polyhedron U will then be completely lost. This may lead to a large mean squared error for prediction.

This suggests that not all thresholds should be too large. The problem can be resolved if there is a set of analysis dictionary atoms with small thresholds. If we assume that

the signal subspace V has dimension K, then in order not to lose essential information there should be at least K pairs of analysis dictionary atoms and soft-thresholds for *information preservation*. These K analysis dictionary atoms are associated with K small soft-thresholds and are able to fully represent the input data space. Therefore the K pairs of analysis dictionary atoms and soft-thresholds together with the corresponding synthesis dictionary atoms provide a baseline estimation for the input data. The remaining analysis dictionary atoms can instead be associated with large thresholds. The outcome of these analysis and thresholding operations is a *clustering* of the input data. That is, the data within the same cluster has the same sparsity pattern and shares a linear model for prediction. The corresponding synthesis atoms then help recover the signal components within the orthogonal subspace $V^{\perp}$.

Based on the above analysis, we propose to divide an analysis dictionary $\boldsymbol{\Omega}$ into two sub-dictionaries $\boldsymbol{\Omega} = [\boldsymbol{\Omega}_{\mathrm{I}}; \boldsymbol{\Omega}_{\mathrm{C}}]$, and similarly divide each threshold vector into two parts $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_{\mathrm{I}}; \boldsymbol{\lambda}_{\mathrm{C}}]$. The Information Preserving Analysis Dictionary (IPAD) $\boldsymbol{\Omega}_{\mathrm{I}}$ with its thresholds $\boldsymbol{\lambda}_{\mathrm{I}}$ aims at passing key information of the input signal to the next layer. The Clustering Analysis Dictionary (CAD) $\boldsymbol{\Omega}_{\mathrm{C}}$ with its threshold $\boldsymbol{\lambda}_{\mathrm{C}}$ is to facilitate the separation of key feature in the signal. The CAD and thresholding operators provide a highly non-linear prediction. Fig. 3.4 shows an analysis dictionary and the soft-thresholding operation with the partition of the IPAD part and the CAD part.

In a multi-layer DeepAM, we adopt the same information preserving and clustering strategy. As depicted in Fig. 3.5, the analysis dictionary at each layer is composed of an IPAD part and a CAD part. The IPADs and thresholds $\{(\boldsymbol{\Omega}_{\mathrm{I}i}, \boldsymbol{\lambda}_{\mathrm{I}i})\}_{i=1}^{L}$ create a channel which transmits the information of the input signal to the CAD in each intermediate layer and to the final estimation. The feature representation $\boldsymbol{x}_{\mathrm{I}}^{L}$ generated by the last layer of IPAD and its thresholds should be able to well represent signal components of the HR signal which are within the input data subspace. The CADs and thresholds $\{(\boldsymbol{\Omega}_{\mathrm{C}i}, \boldsymbol{\lambda}_{\mathrm{C}i})\}_{i=1}^{L}$ are the main source of non-linearity in DeepAM. The

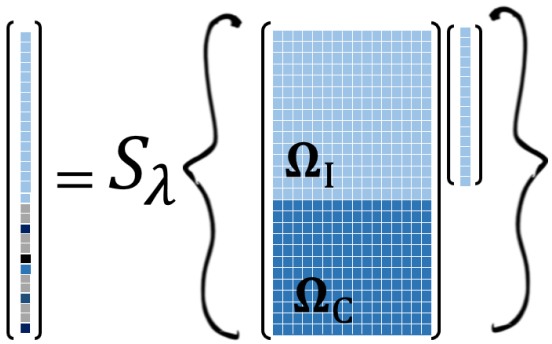


Figure 3.4: The analysis dictionary $\mathbf{\Omega}$ is designed to consist of an information preserving analysis dictionary $\mathbf{\Omega}_\mathrm{I}$ and a clustering analysis dictionary $\mathbf{\Omega}_\mathrm{C}$. The soft-thresholds corresponding to $\mathbf{\Omega}_\mathrm{C}$ are much higher than those used for $\mathbf{\Omega}_\mathrm{I}$ and result in a sparser representation.

feature representation $\boldsymbol{x}_\mathrm{C}^L$ generated by the last layer of CAD should be able to well represent the signal components of $\boldsymbol{y}$ which are orthogonal to the input data subspace. Therefore, the function of CAD and its thresholds can be interpreted as identifying the data with large energy in the subspace orthogonal to the input data subspace. A deep layer of CAD takes the feature representation generated by the IPAD and CAD of the previous layer as input and can generate a non-linear feature representation that cannot be attained by a single layer soft-thresholding with the same number of atoms. Therefore a DeepAM with deeper layers is expected to outperform a shallower one.

3.4 Learning a Deep Analysis Dictionary Model

In this section, we introduce the proposed learning algorithm for DeepAM. In view of the distinctive goals of the two pairs of sub-dictionary and thresholds, different learning criteria have been proposed for learning the IPAD and threshold pair and the CAD and threshold pair.

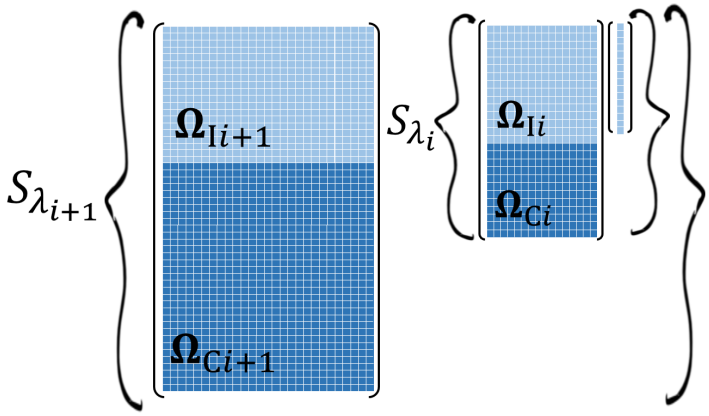


Figure 3.5: **Two consecutive layers in DeepAM.** The IPAD and threshold pairs create an information flow channel from input to output, and the CAD and threshold pairs combine information from the previous layer and generate a feature representation that can well represent the residual part.

3.4.1 Basic Analysis Dictionary Learning Algorithm

The IPAD and the CAD have different functions but also share some common learning objectives. There are four basic objectives for learning an analysis dictionary: (1) its atoms span a subspace of its input data space; (2) it is able to sparsify the input data; (3) there are no pairs of row atoms that are linearly dependent; (4) the row atoms are of unit norm.

Our proposed learning algorithm is an extension of the GeOmetric Analysis Operator Learning (GOAL) algorithm [63, 69] and we denote it as GOAL+. The four learning objectives can be attained by using corresponding constraints. The IPAD and the CAD are learned using modified versions of GOAL+ algorithm.

For simplicity, let us denote the analysis dictionary to be learned as $\mathbf{\Omega} \in \mathbb{R}^{m\times n}$, the j-th atom of $\mathbf{\Omega}$ as $\boldsymbol{\omega}_j^T$ and the training data as $\boldsymbol{\mathcal{X}} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{n\times N}$. We assume that the row atoms of $\mathbf{\Omega}$ span a K dimensional subspace $V \in \mathbb{R}^n$. Let us denote with $\boldsymbol{W} \in \mathbb{R}^{n\times K}$ the orthogonal basis of V, and with $\boldsymbol{W}_N \in \mathbb{R}^{n\times(n-K)}$ the orthogonal basis of the orthogonal complement $V^{\perp}$.

The first learning objective is that the learned analysis dictionary $\mathbf{\Omega}$ should span only

the subspace V. There are two main reasons for this requirement. First, the analysis dictionary $\boldsymbol{\Omega}$ which spans V can fully preserve the information of the input data that is within V. Second, if an atom $\boldsymbol{\omega}^T$ belongs to $V^{\perp}$, it is an unnecessary atom. This is because the coefficients $\boldsymbol{\omega}^T\boldsymbol{\mathcal{X}}$ will be zero since $V^{\perp}$ is in the null-space of $\boldsymbol{\mathcal{X}}$. We apply a logarithm determinant (log-det) term $h(\boldsymbol{\Omega})$ to impose the information preservation constraint:

$$h(\boldsymbol{\Omega}) = -\frac{1}{K\log K}\log\det\left(\frac{1}{m}\boldsymbol{W}^T\boldsymbol{\Omega}^T\boldsymbol{\Omega}\boldsymbol{W}\right). \tag{3.5}$$

Together with the unit norm constraint, the feasible set of the analysis dictionary $\boldsymbol{\Omega}$ is defined as $\Theta = \mathbb{S}_{n-1}^m \cap \boldsymbol{W}_N^{\perp}$ with $\mathbb{S}_{n-1}$ being the unit sphere in $\mathbb{R}^n$ and $\boldsymbol{W}_N^{\perp}$ being the orthogonal complement of $\boldsymbol{W}_N$. The feasible set Θ restricts the learned atoms in $\boldsymbol{\Omega}$ to be of unit norm and excludes the contributions from $\boldsymbol{W}_N$. Eqn. (3.5) is a generalization of the log-det constraint term applied in GOAL [63, 69]. Here, $\boldsymbol{W}$ defines a low-dimensional subspace of the input data space whose size K could be much smaller than the dimension of the input signal n as moving towards a deeper layer, while $\boldsymbol{W}$ in GOAL method [63, 69] defines a much larger subspace and is with $k = n$ or $k = n - 1$.

We achieve the constraint $\boldsymbol{\Omega}^T \in \Theta$ by performing orthogonal projection onto the tangent space $\mathrm{T}_{\boldsymbol{\Omega}}(\Theta)$ of the manifold Θ at location $\boldsymbol{\Omega}$. For a row atom $\boldsymbol{\omega}^T$, the operation of the orthogonal projection onto the tangent space $\mathrm{T}_{\boldsymbol{\omega}}(\Theta)$ can be represented by the projection matrix $\boldsymbol{P}_{\boldsymbol{\omega}}$:

$$\boldsymbol{P}_{\boldsymbol{\omega}} = \mathbf{I}_n - \boldsymbol{Q}_{\boldsymbol{\omega}}^{\dagger}\boldsymbol{Q}_{\boldsymbol{\omega}}, \tag{3.6}$$

where $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix, and $\boldsymbol{Q}_{\boldsymbol{\omega}} = [2\boldsymbol{\omega}^T; \boldsymbol{W}_N^T] \in \mathbb{R}^{(n-k+1)\times n}$.

Sparsifying ability is essential for both IPAD and CAD. When the input data is noisy, it is beneficial to apply a mild non-linear operation to the IPAD coefficients as a form of denoising. The function of CADs is to identify the data in $\boldsymbol{\mathcal{Y}}$ with large energy in the subspace orthogonal to the input data space. The sparsifying ability enables

Algorithm 1: GOAL+ Algorithm

Input: Training data, row number of the dictionary;
Initialize: Initialized $\mathbf{\Omega}^{(0)}$, $t = 0$;
while *halting criterion false* **do**
  $t \leftarrow t + 1$;
  Compute gradient of the objective function $\nabla f(\mathbf{\Omega}^{(t)})$;
  Orthogonal project $\nabla f(\mathbf{\Omega}^{(t)})$ onto the tangent space of manifold Θ at $\mathbf{\Omega}^{(t)}$: $\mathcal{G} \doteq \Pi_{\mathrm{T}_{\mathbf{\Omega}^{(t)}}(\Theta)}(\nabla f(\mathbf{\Omega}^{(t)}))$;
  Find new search direction $\mathcal{H}^{(t)} = -\mathcal{G} + \beta^{(t)}\mathcal{T}_{\mathcal{H}^{(t-1)}}$;
  Update $\mathbf{\Omega}^{(t+1)}$ along the search direction $\mathcal{H}^{(t)}$ using backtracking line search.
end
Output: Learned analysis dictionary $\mathbf{\Omega}$.

the analysis and the thresholding operation to select only a specific group of data. The sparsifying constraint is imposed by using a log-square function which is a good approximation of the l_0 norm:

$$g(\mathbf{\Omega}) = \frac{1}{Nm\log(1+\nu)} \sum_{i=1}^{N} \sum_{j=1}^{m} \log\left(1 + \nu(\boldsymbol{\omega}_j^T \boldsymbol{x}_i)^2\right), \tag{3.7}$$

where ν is a tunable parameter which controls the sparsifying ability of the learned dictionary.

Linearly dependent row atoms (e.g. $\boldsymbol{\omega}_i^T = \pm\boldsymbol{\omega}_j^T$) are undesirable in the learned dictionary. A logarithm barrier term $l(\mathbf{\Omega})$ is used to penalize linearly dependent row atoms:

$$l(\mathbf{\Omega}) = -\frac{1}{m(m-1)} \sum_{1 \le i < j \le m} \log\left(1 - \left(\boldsymbol{\omega}_i^T \boldsymbol{\omega}_j\right)^2\right). \tag{3.8}$$

Combining the information preservation constraint in Eqn. (3.5), feasible set constraint in Eqn. (3.6), sparsifying constraint in Eqn. (3.7), and linearly dependent penalty term in Eqn. (3.8), the objective function of GOAL+ algorithm is formulated as:

$$\mathbf{\Omega} = \arg\min_{\mathbf{\Omega}^T \in \Theta} f(\mathbf{\Omega}), \tag{3.9}$$

where $f(\mathbf{\Omega}) = g(\mathbf{\Omega}) + \kappa h(\mathbf{\Omega}) + \upsilon l(\mathbf{\Omega})$ with κ and υ being the regularization parameters.

The objective function defined in Eqn. (3.9) is optimized using a geometric conjugate gradient descent method [63, 92]. The analysis dictionary learning algorithm GOAL+ is summarized in **Algorithm 1**. At iteration t, the gradient of the objective function $\nabla f(\mathbf{\Omega}^{(t)})$ is computed and orthogonal projected on the tangent space of the manifold Θ at location $\mathbf{\Omega}^{(t)}$. The orthogonal projection of $\nabla f(\mathbf{\Omega})$ onto the tangent space $\mathrm{T}_{\mathbf{\Omega}}(\Theta)$ can be expressed as $\Pi_{\mathrm{T}_{\mathbf{\Omega}}(\Theta)}(\nabla f(\mathbf{\Omega})) = [\boldsymbol{P}_{\boldsymbol{\omega}_1}\nabla f(\boldsymbol{\omega}_1), \cdots, \boldsymbol{P}_{\boldsymbol{\omega}_{d_{\mathrm{I}i}}}\nabla f(\boldsymbol{\omega}_{d_{\mathrm{I}i}})]$. Let us denote $\mathcal{G} \doteq \Pi_{\mathrm{T}_{\mathbf{\Omega}^{(t)}}(\Theta)}(\nabla f(\mathbf{\Omega}^{(t)}))$, the search direction can be set as $\mathcal{H}^{(t)} = -\mathcal{G}$. In practice, the search direction is a combination of $\mathcal{G}$ and the previous search direction $\mathcal{T}_{\mathcal{H}^{(t-1)}}$. The updated analysis dictionary $\mathbf{\Omega}^{(t+1)}$ is then obtained by gradient descent with back-tracking line search along the search direction $\mathcal{H}^{(t)}$. The halting condition is when the analysis dictionary converges or when a pre-defined maximum number of iterations is reached. In summary, our optimization approach is similar to that in GOAL [63] except the orthogonal projection step as described in Eqn. (3.6) which represents the constraint introduced by the feasible set Θ. For a more detailed treatment of the geometric conjugate gradient descent we refer to [63, 92]. Now that the overall objectives of GOAL+ have been introduced, we can focus on how to tailor the optimization Eqn. (3.9) to achieve the objectives of IPAD and CAD respectively.

3.4.2 IPAD and Thresholds Pair Learning

The function of the IPAD and threshold pair $(\mathbf{\Omega}_{\mathrm{I}i}, \boldsymbol{\lambda}_{\mathrm{I}i})$ is to pass key information of the input data $\boldsymbol{\mathcal{X}}^0$ to a deeper layer. The learned IPADs create a channel that enables the information flow from the input signal to the estimated output signal.

IPAD Learning

The training data for learning $\boldsymbol{\Omega}_{\mathrm{I}i}$ is the i-th layer input training data $\boldsymbol{\mathcal{X}}^{i-1}$ (the $(i-1)$-th layer training data is obtained as $\boldsymbol{\mathcal{X}}^{i} = \mathcal{S}_{\boldsymbol{\lambda}_i}(\boldsymbol{\Omega}_i \boldsymbol{\mathcal{X}}^{i-1})$ for $i \geq 1$). Let us denote the rank of the first layer input training data $\boldsymbol{\mathcal{X}}^0$ as $k_0 = \mathrm{rk}(\boldsymbol{\mathcal{X}}^0)$ where $\mathrm{rk}(\cdot)$ outputs the rank of a matrix. The IPAD $\boldsymbol{\Omega}_{\mathrm{I}i} \in \mathbb{R}^{d_{\mathrm{I}i} \times d_{i-1}}$ is assumed to have $d_{\mathrm{I}i} \geq \mathrm{rk}(\boldsymbol{\mathcal{X}}^0)$ atoms to ensure that the learned IPAD can well represent the input data subspace.

By setting the training data as $\boldsymbol{\mathcal{X}}^{i-1}$, the i-th layer analysis dictionary $\boldsymbol{\Omega}_{\mathrm{I}i}$ can be learned using GOAL+. The orthonormal basis $\boldsymbol{W} \in \mathbb{R}^{d_{i-1} \times k_0}$ is set to be an arbitrary basis of $\boldsymbol{\mathcal{X}}^{i-1}$ that corresponds to the signal subspace of $\boldsymbol{\mathcal{X}}^0$. The orthogonal basis $\boldsymbol{W}_N \in \mathbb{R}^{d_{i-1} \times (d_{i-1} - k_0)}$ is set to span the orthogonal complement of the subspace spanned by $\boldsymbol{W}$.

Learning the Thresholds for IPAD

Given a learned IPAD $\boldsymbol{\Omega}_{\mathrm{I}i}$, the analysis coefficients $\boldsymbol{\alpha}^i = \boldsymbol{\Omega}_{\mathrm{I}i} \boldsymbol{x}^{i-1}$ contain sufficient information of $\boldsymbol{x}^{i-1}$. When $\boldsymbol{\alpha}^i$ is a redundant representation or when the input data $\boldsymbol{x}^{i-1}$ is noisy, applying a proper thresholding operation to $\boldsymbol{\alpha}^i$ can further enhance the robustness of the representation. We propose to apply soft-thresholding with small thresholds to the IPAD analysis coefficients as $\boldsymbol{z} = \mathcal{S}_{\boldsymbol{\lambda}_{\mathrm{I}i}}(\boldsymbol{\Omega}_{\mathrm{I}i} \boldsymbol{x}^{i-1})$ and interpret the soft-thresholding operation as a form of denoising.

There are related works in the literature about thresholding for redundant representations [93–95]. Elad [93] shows that simple shrinkage is effective for redundant representation and interpretes the simple shrinkage as the first iteration for solving Basis Pursuit Denoising (BPDN) problems. Raphan and Simoncelli [94] proposed a denoising scheme for redundant representations based on Stein's Unbiased Risk Estimator. Lin and Lee [95] proposed a Bayesian framework for finding the optimal l_1-norm regularization.

Let us consider a weighed l_1-norm regularized minimization problem:

$$\min_{\boldsymbol{z}} \frac{1}{2}||\boldsymbol{x} - \boldsymbol{\Omega}^T \boldsymbol{z}||_2^2 + \sum_{j=1}^{m} \lambda_j |z_j|, \tag{3.10}$$

where z_j is the j-th coefficient of the sparse vector $\boldsymbol{z}$, and λ_j is the corresponding regularization parameter.

Selecting the soft-threshold $\boldsymbol{\lambda}_{\mathrm{I}i}$ is equivalent to finding suitable regularization parameters in Eqn. (3.10) as the soft-thresholding operation $\mathcal{S}_{\boldsymbol{\lambda}}(\boldsymbol{\Omega}\boldsymbol{x})$ can be interpreted as the first iteration of the Iterative Soft-Thresholding Algorithm (ISTA) [55] for solving Eqn. (3.10):

$$\boldsymbol{z}^{(1)} = \mathcal{S}_{\boldsymbol{\lambda}} \left(\boldsymbol{z}^{(0)} + \boldsymbol{\Omega} \left(\boldsymbol{x} - \boldsymbol{\Omega}^T \boldsymbol{z}^{(0)} \right) \right), \tag{3.11}$$

where the initial sparse code $\boldsymbol{z}^{(0)} = \mathbf{0}$.

Lin and Lee [95] proposed a method to choose the optimal regularization parameters based on a Bayesian analysis. They assume that the data $\boldsymbol{x}$ is with additive i.i.d. zero mean Gaussian noise with variance σ^2:

$$P(\boldsymbol{x}|\boldsymbol{\Omega}^T, \boldsymbol{z}, \sigma^2) = \frac{1}{(2\pi\sigma)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} \left\| \boldsymbol{x} - \boldsymbol{\Omega}^T \boldsymbol{z} \right\|_2^2 \right), \tag{3.12}$$

and assume a Laplacian distribution prior for the sparse code $\boldsymbol{z}$ with parameters $\boldsymbol{\lambda}$:

$$P(\boldsymbol{z}|\boldsymbol{\lambda}) = \prod_{j=1}^{m} \frac{\lambda_j}{2} \exp\left(-\lambda_j |z_j| \right). \tag{3.13}$$

Empirically, we have found that the prior distribution $P(\boldsymbol{z})$ can be well characterized by an i.i.d. zero-mean Laplacian distribution. Based on the analysis in [95], the optimal regularization parameters for Eqn. (3.10) can be set as inversely proportional to the

variance of the Laplacian distribution:

$$\boldsymbol{\lambda} \propto \left[\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_m}\right]^T, \tag{3.14}$$

where σ_j is the variance of the Laplacian distribution for the j-th sparse code z_j.

From Eqn. (3.14), the soft-threshold associated with IPAD $\boldsymbol{\Omega}_{\mathrm{I}i}$ is:

$$\boldsymbol{\lambda}_{\mathrm{I}i} = \rho_{\mathrm{I}} \left[\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \cdots, \frac{1}{\sigma_{d_{\mathrm{I}i}}}\right]^T, \tag{3.15}$$

where ρ_{I} is a scaling parameter, and the variance σ_j of the j-th coefficient can be estimated using the obtained IPAD $\boldsymbol{\Omega}_{\mathrm{I}i}$ and its input data (i.e. σ_j is the mean absolute value of the inner product between the j-th atom of $\boldsymbol{\Omega}_{\mathrm{I}i}$ and the input data).

There is only a free parameter ρ to be determined. It can be obtained by solving a 1-dimensional search problem. The optimization problem for ρ is therefore formulated as:

$$\hat{\rho}_{\mathrm{I}} = \arg\min_{\rho \in \mathcal{D}} \left\| \boldsymbol{\mathcal{Y}} - \boldsymbol{G}\mathcal{S}_{\rho\boldsymbol{\lambda}} \left(\boldsymbol{\Omega}_{\mathrm{I}i}\boldsymbol{\mathcal{X}}^{i-1}\right) \right\|_F^2, \tag{3.16}$$

where $\boldsymbol{\lambda} = [1/\sigma_1, 1/\sigma_2, \cdots, 1/\sigma_{d_{\mathrm{I}i}}]^T$, $\boldsymbol{G} = \boldsymbol{\mathcal{Y}}\boldsymbol{Z}^T(\boldsymbol{Z}\boldsymbol{Z}^T)^{-1}$ with $\boldsymbol{Z} = \mathcal{S}_{\rho\boldsymbol{\lambda}}(\boldsymbol{\Omega}_{\mathrm{I}i}\boldsymbol{\mathcal{X}}^{i-1})$, and $\mathcal{D}$ is a discrete set of values.

The obtained pair $(\boldsymbol{\Omega}_{\mathrm{I}i}, \boldsymbol{\lambda}_{\mathrm{I}i})$ should be able to preserve the important information within the input signal and give no worse performance when compared to a linear model without any non-linearity.

3.4.3 Learning CAD and Threshold Pairs

The function of the clustering analysis dictionary and threshold pair $(\boldsymbol{\Omega}_{\mathrm{C}i}, \boldsymbol{\lambda}_{\mathrm{C}i})$ is to sparsify its input data and identify the data with large residual energy. The CAD and threshold pairs at shallow layers provide low-level feature representations for the CADs

at deeper layers.

CAD Learning

Different from IPAD, learning CAD $\boldsymbol{\Omega}_{\mathrm{C}i}$ requires supervision from both the input training data $\boldsymbol{\mathcal{X}}^{i-1}$ and the ground-truth training data $\boldsymbol{\mathcal{Y}}$.

Let us denote with $\boldsymbol{\mathcal{Y}}^i = \boldsymbol{D}^i\boldsymbol{\mathcal{X}}^{i-1}$ the middle resolution data and with $\boldsymbol{\mathcal{E}}^i = \boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{Y}}^i$ the residual data where $\boldsymbol{D}_i \in \mathbb{R}^{d_{L+1}\times d_i}$ is the synthesis dictionary of layer i which minimizes the mean squared error of the residual data and can be obtained by solving:

$$\boldsymbol{D}_i = \arg\min_{\boldsymbol{D}} ||\boldsymbol{D}\boldsymbol{\mathcal{X}}^{i-1} - \boldsymbol{\mathcal{Y}}||_F^2. \tag{3.17}$$

It has a closed-form solution given by:

$$\boldsymbol{D}_i = \boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{X}}^{i-1^T}\left(\boldsymbol{\mathcal{X}}^{i-1}\boldsymbol{\mathcal{X}}^{i-1^T}\right)^{-1}. \tag{3.18}$$

The learning objective for CAD $\boldsymbol{\Omega}_{\mathrm{C}i}$ is that its atoms should be able to project $\boldsymbol{\mathcal{X}}^{i-1}$ onto directions where the data with large residual has responses with large magnitude. To achieve that, we propose to first learn an analysis dictionary $\boldsymbol{\Psi}_i \in \mathbb{R}^{d_{\mathrm{C}i}\times d_{L+1}}$ in the ground-truth data domain which is able to simultaneously sparsify the middle resolution data $\boldsymbol{\mathcal{Y}}^i$ and the residual data $\boldsymbol{\mathcal{E}}^i$. That is, the atoms of $\boldsymbol{\Psi}_i$ are able to identify the data in $\boldsymbol{\mathcal{Y}}^i$ with large residual energy. The i-th layer CAD is then re-parameterized as:

$$\boldsymbol{\Omega}_{\mathrm{C}i} = \boldsymbol{\Psi}_i\boldsymbol{D}_i. \tag{3.19}$$

As a result, the learned CAD $\boldsymbol{\Omega}_{\mathrm{C}i}$ will have the same identification ability as $\boldsymbol{\Psi}_i$ since $\boldsymbol{\Omega}_{\mathrm{C}i}\boldsymbol{x}_j^{i-1} = \boldsymbol{\Psi}_i\boldsymbol{y}_j^i$.

We propose a consistent constraint for learning the analysis dictionary $\boldsymbol{\Psi}_i$. Each anal-

ysis atom $\boldsymbol{\psi}_l^T$ is enforced to be able to jointly sparsify $\boldsymbol{\mathcal{Y}}^i$ and $\boldsymbol{\mathcal{E}}^i$:

$$p(\boldsymbol{\Psi}) = c \sum_{j=1}^{N} \sum_{l=1}^{d_{\mathrm{C}i}} \log \left(1 + \nu \left((\boldsymbol{\psi}_l^T \boldsymbol{y}_j^i)^2 - (\boldsymbol{\psi}_l^T \boldsymbol{e}_j^i)^2 \right)^2 \right), \tag{3.20}$$

where $c = 1/N d_{\mathrm{C1}} \log(1+\nu)$, and ν is a tunable parameter.

The objective function for learning the analysis dictionary $\boldsymbol{\Psi}_i$ can be formulated as:

$$\boldsymbol{\Psi}_i = \arg \min_{\boldsymbol{\Psi}^T \in \Theta} f(\boldsymbol{\Psi}), \tag{3.21}$$

where $f(\boldsymbol{\Psi}) = g(\boldsymbol{\Psi}) + \kappa h(\boldsymbol{\Psi}) + \upsilon l(\boldsymbol{\Psi}) + \mu p(\boldsymbol{\Psi})$ with κ, υ and μ being the regularization parameters. Here, $g(\cdot)$, $h(\cdot)$ and $l(\cdot)$ are those defined in Eqn. (3.7), Eqn. (3.5) and Eqn. (3.8), respectively.

The input training data is set to $(\boldsymbol{\mathcal{X}}^{i-1}, \boldsymbol{\mathcal{Y}})$. Let us denote the rank of $\boldsymbol{\mathcal{Y}}$ as $k_L = \mathrm{rk}(\boldsymbol{\mathcal{Y}})$. The orthogonal basis $\boldsymbol{W} \in \mathbb{R}^{d_{L+1} \times k_L}$ is set to be an arbitrary basis of the signal subspace of $\boldsymbol{\mathcal{Y}}$. The orthonormal basis $\boldsymbol{W}_N \in \mathbb{R}^{d_{L+1} \times (d_{L+1} - k_L)}$ is set to be a basis spanning the orthogonal complement to the subspace spanned by $\boldsymbol{W}$.

The objective function in Eqn. (3.21) is optimized using GOAL+ algorithm. With the learned analysis dictionary $\boldsymbol{\Psi}_i$, the i-th layer CAD is obtained as in Eqn. (3.19).

Learning the Thresholds for CAD

The thresholds for CAD are crucial to the performance of DeepAM as the CAD and threshold pair is the main source of non-linearity in DeepAM. The atoms of the learned CAD project the input data onto directions where the data with large residual will have responses with large magnitude. After soft-thresholding, the coefficients should be as sparse as possible to achieve a strong discriminative power.

We propose to set the CAD thresholds in the form of:

$$\boldsymbol{\lambda}_{\mathrm{C}i} = \rho_{\mathrm{C}} \left[\sigma_1, \sigma_2, \cdots, \sigma_{d_{\mathrm{C}i}}\right], \tag{3.22}$$

where ρ_{C} is a scaling parameter, and σ_j is the variance of the Laplacian distribution for the j-th atom.

As discussed in previous section, the analysis coefficients can be well modelled by Laplacian distributions. By setting the CAD thresholds proportional to the variance of the analysis coefficients, the proportion of data that has been set to zero for each pair of atom and threshold will be the same. When the synthesis dictionary is applied for reconstruction, the synthesis atoms corresponding to the CAD atoms will be activated with a similar frequency.

With this simplification, the CAD thresholds can be learned in an efficient way. The scaling parameter ρ_{C} can be obtained using the same strategy as in Eqn. (3.16) by solving a 1-dimensional search problem. We will show that the learned CAD thresholds lead to an effective system in Section 3.5 simulation results.

3.4.4 Synthesis Dictionary Learning

The deep analysis dictionary learning can be considered as a layer-wise representation learning process in which the input data $\boldsymbol{\mathcal{X}}^0$ is consistently non-linearly transformed to a high dimensional feature representation $\boldsymbol{\mathcal{X}}^L$ with good descriptive and discriminative properties. The synthesis dictionary $\boldsymbol{D}$ models the linear transformation from $\boldsymbol{\mathcal{X}}^L$ to the desired HR counterpart $\boldsymbol{\mathcal{Y}}$. Similar to Eqn. (3.18), the synthesis dictionary is learned using least squares:

$$\boldsymbol{D} = \boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{X}}^{L^T}\left(\boldsymbol{\mathcal{X}}^L\boldsymbol{\mathcal{X}}^{L^T}\right)^{-1}. \tag{3.23}$$

Algorithm 2: DeepAM Learning Algorithm

Input: Training data pair $(\boldsymbol{\mathcal{X}}^0, \boldsymbol{\mathcal{Y}})$, the number of layers L, and the structure of DeepAM;
for $i \leftarrow 1$ **to** L **do**
  Learning $\boldsymbol{\Omega}_{\mathrm{I}i}$ using GOAL+ with training data $\boldsymbol{\mathcal{X}}^{i-1}$ and objective function Eqn. (3.9);
  Learning $\boldsymbol{\Omega}_{\mathrm{C}i}$ using GOAL+ with training data $(\boldsymbol{\mathcal{X}}^{i-1}, \boldsymbol{\mathcal{Y}})$ and objective function Eqn. (3.21);
  Learning thresholds $\boldsymbol{\lambda}_{\mathrm{I}i}$ and $\boldsymbol{\lambda}_{\mathrm{C}i}$;
  $\boldsymbol{\Omega}_i \leftarrow [\boldsymbol{\Omega}_{\mathrm{I}i}; \boldsymbol{\Omega}_{\mathrm{C}i}]$, $\boldsymbol{\lambda}_i \leftarrow [\boldsymbol{\lambda}_{\mathrm{I}i}; \boldsymbol{\lambda}_{\mathrm{C}i}]$;
  Update input training data as $\boldsymbol{\mathcal{X}}^i = \mathcal{S}_{\boldsymbol{\lambda}_i}(\boldsymbol{\Omega}_i \boldsymbol{\mathcal{X}}^{i-1})$;
end
Learning the synthesis dictionary $\boldsymbol{D}$ as in Eqn. (3.23).
Output: Learned DeepAM $\left\{\{\boldsymbol{\Omega}_i, \boldsymbol{\lambda}_i\}_{i=1}^{L}, \boldsymbol{D}\right\}$.

3.4.5 DeepAM Learning Algorithm

The overall learning algorithm for DeepAM is summarized in **Algorithm 2**. We adopt a layer-wise learning strategy for DeepAM. At each layer, two sub-dictionaries IPAD and CAD are independently learned and then combined to form the analysis dictionary, and the thresholds for IPAD and CAD are obtained with two different strategies. In this way, each pair of analysis dictionary and soft-thresholding operations consists of two pairs of linear transform and non-linear operation with different functionalities. Finally, the synthesis dictionary is learned using least squares.

3.5 Simulation Results

In this section, we report the implementation details and numerical results of our proposed DeepAM method and compare our proposed method with Deep Neural Networks learned using back-propagation [20] and with other single image super-resolution algorithms.

Parameters	ν	κ	υ	μ
IPAD	$100 \times d_{i-1}$	$d_{\mathrm{I}i}$	$0.01 \times d_{\mathrm{I}i}$	—
CAD	$100 \times d_{i-1}$	$0.01 \times d_{\mathrm{C}i}$	$0.01 \times d_{\mathrm{C}i}$	10

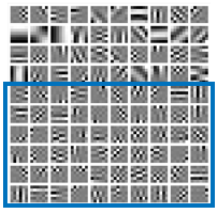
Table 3.1: Parameters setting of GOAL+ algorithm for learning the i-th layer IPAD $\mathbf{\Omega}_{\mathrm{I}i} \in \mathbb{R}^{d_{\mathrm{I}i} \times d_{i-1}}$ and CAD $\mathbf{\Omega}_{\mathrm{C}i} \in \mathbb{R}^{d_{\mathrm{C}i} \times d_{i-1}}$.

3.5.1 Implementation Details

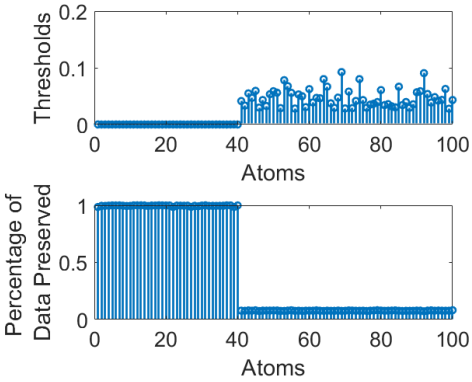
We use the standard 91 training images [1] as training dataset and use *Set5* [1] and *Set14* [2] as the testing dataset. The Peak Signal-to-Noise Ratio (PSNR)[3] is used as the evaluation metric. The color images have been converted from RGB color space to YCbCr color space and image super-resolution is only applied to the luminance component. The low-resolution (LR) images are obtained by down-sampling the ground-truth high-resolution (HR) images by a factor $s = 2$ using Matlab function imresize. The size of the low-resolution patches is set to $p = 6$ for the purpose of better visualization and easier interpretation. The size of the high-resolution patches is then 12×12. Around $N = 3 \times 10^5$ LR-HR patch pairs have been collected for training. During testing, full patch overlapping is applied to reconstruct the HR images.

Table 3.1 shows the parameters setting of GOAL+ algorithm for learning the i-th layer Information Preserving Analysis Dictionary (IPAD) and Clustering Analysis Dictionary (CAD). A grid search has been used to find the regularization parameters in the objection functions in Eqn. (3.9) and Eqn. (3.21). Both the IPAD and the CAD are initialized with i.i.d. Gaussian random entries. We apply batch training for GOAL+ algorithm. The training data has been divided into batches with the size of $N_b = 10^4$. During training, the GOAL+ algorithm is sequentially applied to batches until the learned dictionary converges. For each batch, $\tau = 100$ iterations of conjugate gradient descent is performed to update the dictionary. The discrete set $\mathcal{D}$ used for searching the scaling parameter of the thresholds is set to be $\mathcal{D} = [\cdots, 10^{-2}, 2\times10^{-2}, \cdots, 10^{-1}, 2\times10^{-1}, \cdots]$.

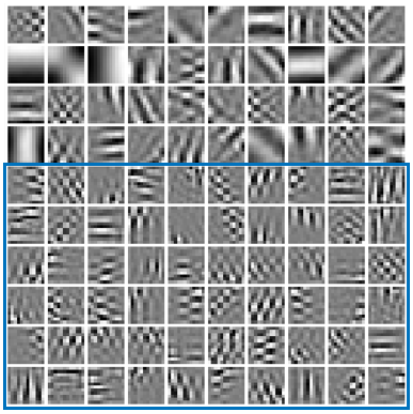
[3]PSNR$=10\log(\frac{255^2}{\sqrt{MSE}})$, where MSE is the mean squared error between the ground-truth HR image and the estimated HR image



(a) $\boldsymbol{\Omega}_1 \in \mathbb{R}^{36\times 100}$.



(b) $\boldsymbol{\lambda}_1 \in \mathbb{R}^{100}$ and the percentage of data preserved after thresholding.



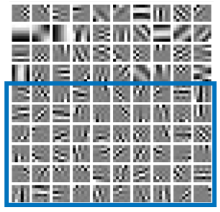
(c) $\boldsymbol{D} \in \mathbb{R}^{100\times 144}$.

Figure 3.6: **An example of a learned 1-layer DeepAM.** Each atom is displayed as a 2D patch. The atoms within blue box are the clustering atoms. In $\boldsymbol{\Omega}_1$, the first 40 atoms are the information preserving atoms and the remaining 60 atoms are the clustering atoms.

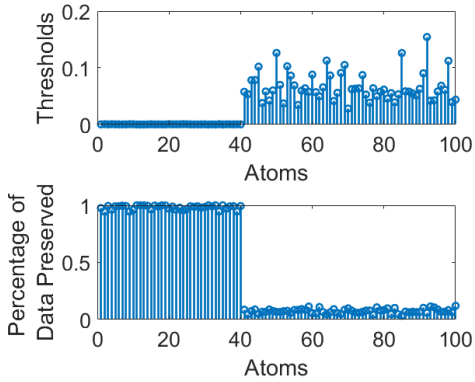
3.5.2 Visualization of the Learned DeepAM

In this section, we will show and analyze the learned DeepAM with the implementation details as described in the previous section.

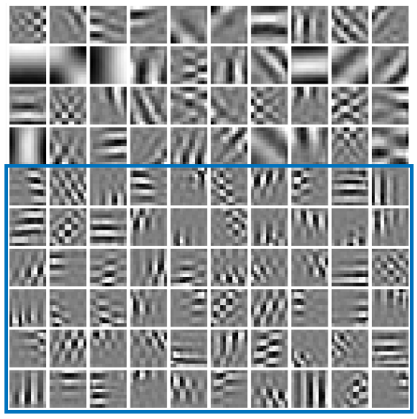
Figure 3.6 shows an example of a learned 1-layer DeepAM. It contains an analysis dictionary $\boldsymbol{\Omega}_1$, thresholds $\boldsymbol{\lambda}_1$ and a synthesis dictionary $\boldsymbol{D}$. Each atom is displayed in a 2D patch in which black and white corresponds to the smallest and the largest value, respectively. The number of the information preserving atoms is set to be 40 which is larger than the rank of the input data. The thresholds depicted in Figure



(a) $\boldsymbol{\Omega}_1 \in \mathbb{R}^{36\times 100}$.



(b) $\boldsymbol{\lambda}_1 \in \mathbb{R}^{100}$ and the percentage of data preserved after thresholding.



(c) $\boldsymbol{D} \in \mathbb{R}^{100\times 144}$.

Figure 3.7: **The 1-layer DeepAM further fine-tuned using backpropagation.** The dictionary atoms seem more localized. The thresholds are in general larger than those in Fig. 3.6(b).

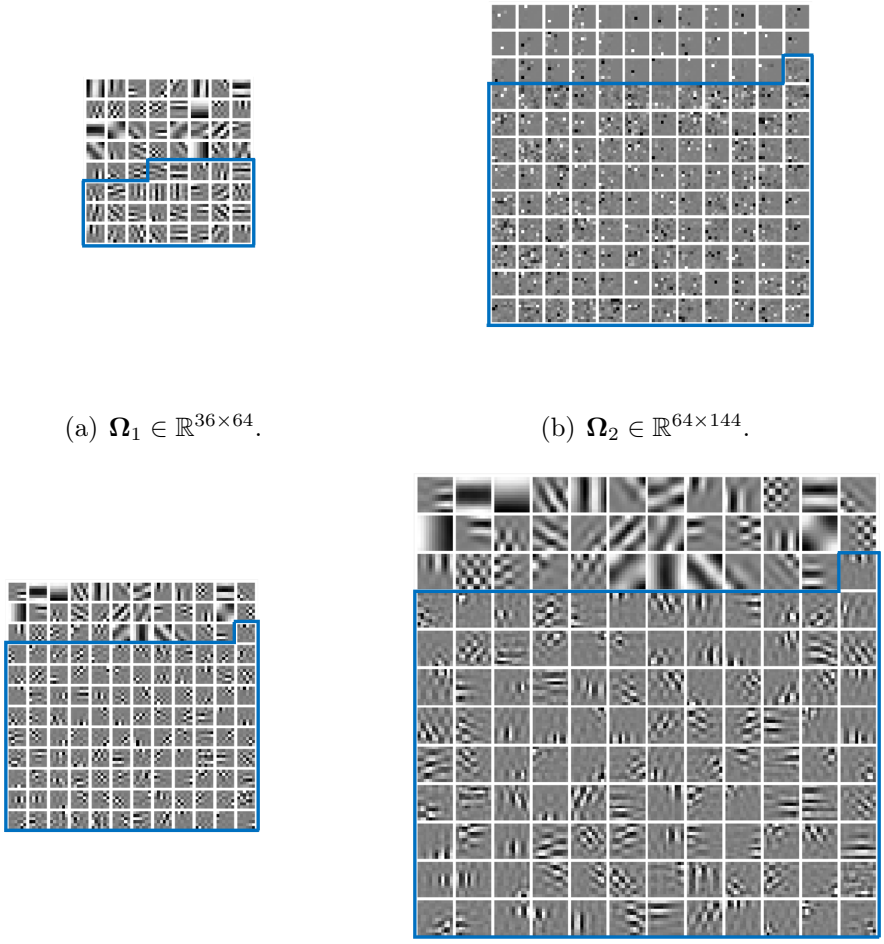
3.6(b) show a clear bimodal behaviour. The first 40 thresholds are close to zero, while the remaining 60 thresholds are relatively large. After thresholding, almost all coefficients corresponding to IPAD are non-zeros, and the percentage of non-zero coefficients of different CAD atoms are similar and are around 8%. This indicates that modelling the distribution of the analysis coefficients as a Laplacian distribution is a good approximation. The atoms within blue box are the clustering atoms. The atoms in IPAD shown in Figure 3.6(a) seem like the LR versions of their corresponding synthesis atoms in Figure 3.6(c). The CAD atoms look like directional filters and are more localized. There is little low-frequency information. The corresponding synthesis atoms are correlated to the CAD atoms, however, they are not the HR counterpart.

The inner product between the HR projection of a CAD atom and its corresponding synthesis atom $\langle \boldsymbol{H}^{\dagger}\boldsymbol{\omega}, \boldsymbol{d}\rangle$ is nearly zero. This shows that the synthesis atoms which correspond to the CAD part are nearly orthogonal to the LR data subspace.

Back-propagation [20] can be used to further update the parameters in our learned DeepAM. The back-propagation update is implemented using Pytorch with Adam optimizer [96], batch size 1024, initial learning rate 10^{-3}, learning rate decay step 20, and decay rate 0.1. The parameter setting has been tuned to achieve the best performance.

Figure 3.7 shows the 1-layer DeepAM after updating using back-propagation [20]. With back-propagation, the performance of DeepAM has a rapid improvement with the first 5 epochs and converges within 20 epochs. After back-propagation update, the average PSNR evaluated on *Set 5* has improved by approximately 0.3 dB. We can find that the different characteristics of the IPAD part and the CAD part remain on the updated DeepAM. There are subtle differences on the updated dictionaries. In general, the IPAD atoms have no visible change, while the CAD atoms have become more localized. The thresholds continue to have a bimodal behaviour. There is only a slight change on the percentage of non-zero coefficients of different atoms.

Figure 3.8 shows the dictionaries within a learned 2-layer DeepAM including two analysis dictionaries $\boldsymbol{\Omega}_1$, $\boldsymbol{\Omega}_2$ and a synthesis dictionary $\boldsymbol{D}$. The first analysis dictionary $\boldsymbol{\Omega}_1$ is similar to that in Figure 3.6(a), while its CAD part mainly contains directional filters due to a smaller number of clustering atoms. The second analysis dictionary $\boldsymbol{\Omega}_2$ is shown in Figure 3.8(b) and is a sparse dictionary where the sparse atoms can be considered as indicating a weighted combination of the first layer analysis dictionary atoms if the soft-thresholding operation is neglected. The effective dictionary $\boldsymbol{\Omega}_{21} = \boldsymbol{\Omega}_2\boldsymbol{\Omega}_1$ shown in Figure 3.8(c) can partially show the effective atoms applied to the input LR data whose IPAD part is similar to those in $\boldsymbol{\Omega}_1$ and CAD part contains more localized atoms when compared to those in $\boldsymbol{\Omega}_1$. Similar observations can be found in a deeper analysis dictionary in DeepAM. The synthesis dictionary has similar characteristics as



(a) $\mathbf{\Omega}_1 \in \mathbb{R}^{36\times 64}$.

(b) $\mathbf{\Omega}_2 \in \mathbb{R}^{64\times 144}$.

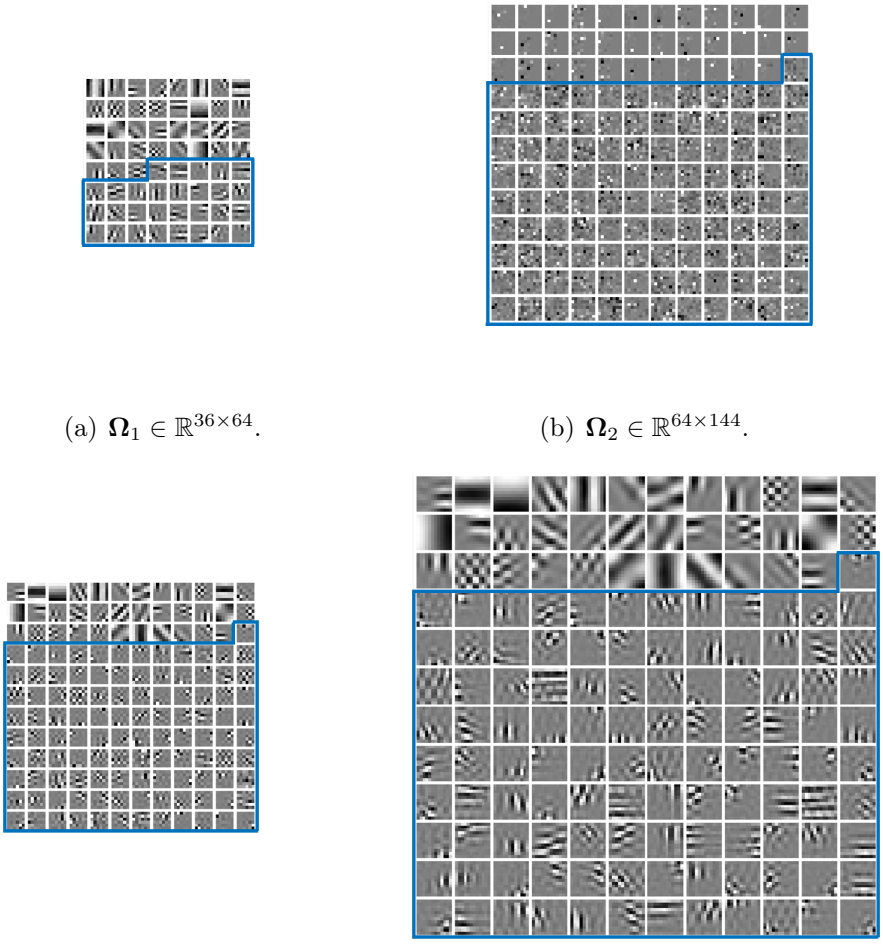
(c) $\mathbf{\Omega}_2\mathbf{\Omega}_1 \in \mathbb{R}^{36\times 144}$.

(d) $\boldsymbol{D} \in \mathbb{R}^{144\times 144}$.

Figure 3.8: **An example of a learned 2-layer DeepAM.** Each atom is displayed as a 2D patch. The atoms within blue box are the clustering atoms.

the one in the 1-layer DeepAM.

Figure 3.9 shows the dictionaries of the back-propagation updated 2-layer DeepAM. The back-propagation slightly updates the dictionaries and converges within 20 epochs. The average PSNR evaluated on *Set 5* has been rapidly improved 0.2 dB with the first 5 epochs and achieved a 0.3 dB improvement after convergence. We can find similar observations on the updated dictionaries of the 2-layer DeepAM as those in the 1-layer DeepAM. After back-propagation, there is still a clear difference between the IPAD atoms and the CAD atoms. The IPAD atoms did not change significantly, while the



(a) $\mathbf{\Omega}_1 \in \mathbb{R}^{36\times 64}$.

(b) $\mathbf{\Omega}_2 \in \mathbb{R}^{64\times 144}$.

(c) $\mathbf{\Omega}_2\mathbf{\Omega}_1 \in \mathbb{R}^{36\times 144}$.

(d) $\boldsymbol{D} \in \mathbb{R}^{144\times 144}$.

Figure 3.9: **The dictionaries of the 2-layer DeepAM further fine-tuned using backpropagation.**

CAD atoms in $\mathbf{\Omega}_1$ and the effective dictionary $\mathbf{\Omega}_{21}$ have become more localized.

3.5.3 Comparison with Deep Neural Networks

In this section, we compare our proposed DeepAM method with Deep Neural Networks (DNNs). The number of IPAD atoms in each layer is set to be 35 which is the rank of the input LR data since a DeepAM with more CAD atoms provides better performance when the input data is noiseless. For comparison, DNNs are learned with the same training data using gradient descent with back-propagation. Let us denote DNN-R and

Model Size	$36 \times 256 \times 144$			$36 \times 256 \times 256 \times 144$			$36 \times 256 \times 256 \times 256 \times 144$		
Method	DNN-R	DNN-S	DeepAM	DNN-R	DNN-S	DeepAM	DNN-R	DNN-S	DeepAM
Set 5	35.83	36.26	35.90	36.14	36.50	36.12	36.19	36.54	36.22
Set 14	31.80	32.06	31.83	32.00	32.23	31.96	32.01	32.25	32.02

Table 3.2: Average PSNR (dB) by different methods evaluated on *Set 5* [1] and *Set 14* [2].

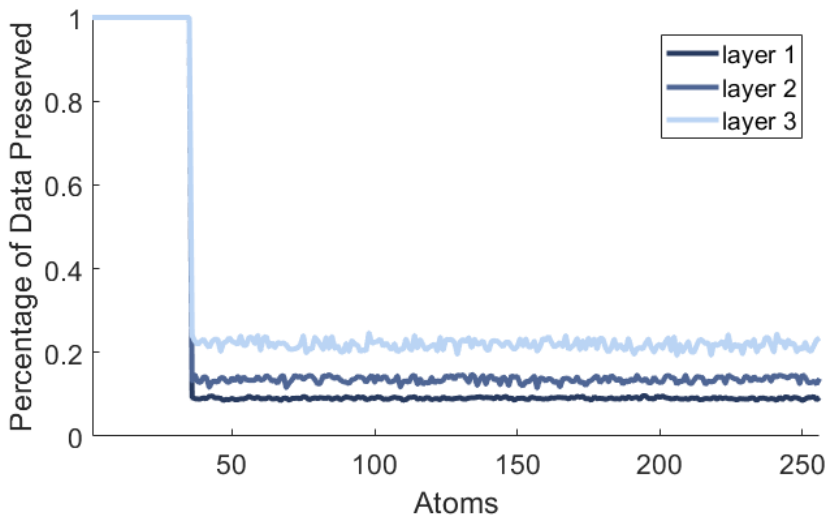


Figure 3.10: The percentage of data preserved after thresholding for the atoms in 3 different layers of the 3-layer DeepAM in Table 3.2.

DNN-S as the DNN with ReLU as non-linearity and soft-thresholding as non-linearity, respectively. The forward model of DNN-S is the same as our DeepAM method. The implementation of DNNs is based on Pytorch with Adam optimizer [96], batch size 1024, initial learning rate 5×10^{-3}, learning rate decay step 50, and decay rate 0.1. The total number of epochs for training is 250. The parameter setting has been tuned to achieve to the best performance. The parameters of the DNNs are initialized using the default method in Pytorch.

Table 3.2 reports the average PSNR (dB) of DNN-R method, DNN-S method and the proposed DeepAM method evaluated on *Set 5* [1] and *Set14* [2]. There are three different model sizes which are corresponding to DNNs with 1 hidden layer, 2 hidden layers and 3 hidden layers where the number of neurons in each layer is fixed to be 256.

With a deeper model, the performance of our proposed DeepAM method improves even when the size of the final feature representation is the same. This indicates that the depth of DeepAM is important for the final performance. The feature representation at a shallow layer provides useful information for a deeper layer and helps the final prediction made in the last layer. Figure 3.10 further shows the percentage of non-zero

coefficients for each atom in 3 different layers of the 3-layer DeepAM in Table 3.2. We can find that the percentage of non-zero coefficients have a bilateral phenomenon in all three layers which is the same as that shown in Figure 3.6(b). After thresholding, the percentage of non-zero coefficients corresponding to CAD atoms are almost the same in each layer. The percentage of non-zero coefficients for CAD atoms in layer 1, 2 and 3 is around 9%, 14% and 22%, respectively. This means the feature representation becomes more non-sparse with the increase of layers. A denser signal representation is helpful for modelling more complex signals which require more synthesis atoms for a good reconstruction quality. This could be the reason for an improved performance with the increase of depth.

Our proposed DeepAM method achieves a similar average PSNR when compared to the DNN-R method over different model sizes. The DNN-S method achieves the highest average PSNR which is around 0.3 dB and 0.2 dB higher than that of our proposed DeepAM method when evaluated in *Set 5* and *Set 14*, respectively. The slightly lower performance of DeepAM when compared to DNN-S should be an acceptable and reasonable result since DeepAM does not utilize joint optimization as DNNs and the thresholds are set according to simple principles. On the other hand, the simulation result validates the effectiveness of our proposed DeepAM method and shows that the simultaneous information preserving and clustering idea could be a good interpretation of the workings of DNNs. It also indicates that our DeepAM method can be further improved. The better performance of DNN-S also suggests that DNNs with soft-thresholding as non-linearity can be more effective for image enhancement applications.

As shown in the previous section, back-propagation can be used to further improve the learned DeepAM. Figure 3.11 shows average PSNR of the 3-layer DeepAM in Table 3.2 which is updated using back-propagation and the performance of the baseline DNN-S. The performance of DeepAM has been improved significantly and outperforms DNN-S

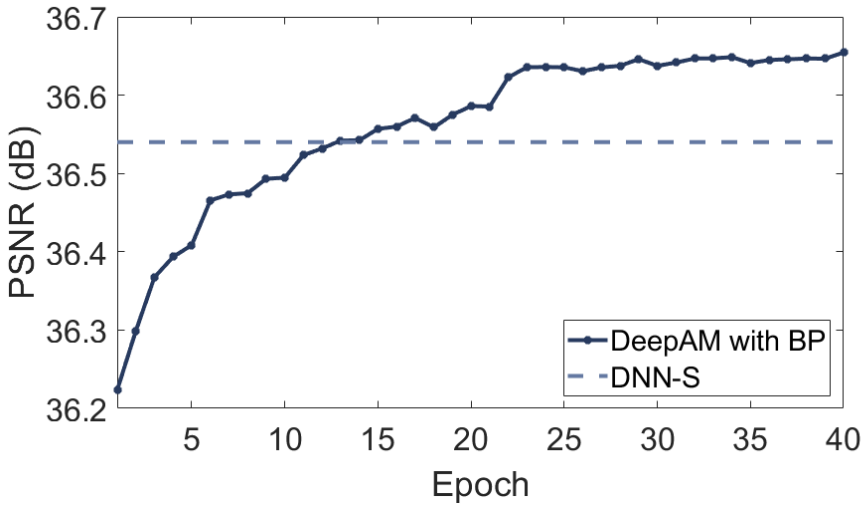


Figure 3.11: The average PSNR (dB) of the DeepAM updated using back-propagation evaluated on *Set 5* [1].

within 15 epoches. The converged performance of DeepAM is around 0.1 dB higher than that of DNN-S. The result shows that the parameters of DeepAM are not far from the converged parameters. It also suggests that the DeepAM learning algorithm has the potential to be a good initialization method for DNNs.

3.5.4 Comparison with Single Image Super-Resolution Methods

In this section, we will compare our proposed DeepAM method with some existing single image super-resolution methods including Bicubic interpolation, sparse coding (SC) based method [2], Anchored Neighbor Regression (ANR) method [7], Adjusted Anchored Neighborhood Regression (A+) method [8], and Super-Resolution Convolutional Neural Network (SRCNN) method [14].

In Table 3.3, $\text{DeepAM}_{\text{bp}}$ and DeepAM represents the 3-layer DeepAM (the model size is $36 \times 256 \times 256 \times 256 \times 64$) that is with and without back-propagation, respectively. The input data is the intensity of the LR image patches. Instead of predicting 12×12 HR for each input LR 6×6 patch, the output of DeepAM is the central 8×8 HR

Images	Bicubic	SC [2]	ANR [7]	A+ [8]	SRCNN [14]	DeepAM	$\text{DeepAM}_{\text{bp}}$
baboon	24.85	25.47	25.55	25.66	25.64	25.65	**25.70**
barbara	27.89	28.52	28.45	**28.51**	28.41	28.49	28.43
bridge	26.64	27.62	27.61	27.86	27.79	27.82	**27.93**
coastguard	29.19	30.26	30.37	30.37	30.29	30.44	**30.53**
comic	25.77	27.37	27.50	28.01	27.98	27.77	**28.12**
face	34.70	35.46	35.53	**35.64**	35.51	35.62	35.62
flowers	30.15	32.06	32.07	32.81	32.82	32.45	**32.86**
foreman	35.49	38.34	38.24	**39.39**	38.68	38.89	39.34
lenna	34.54	36.08	36.19	36.47	36.37	36.46	**36.51**
man	29.12	30.32	30.34	30.76	30.70	30.57	**30.78**
monarch	32.70	35.53	35.49	36.80	36.97	36.06	**37.00**
pepper	35.01	36.64	36.51	37.08	36.90	36.87	**37.11**
ppt3	26.61	29.03	28.70	29.82	**30.13**	29.13	29.70
zebra	30.44	33.07	32.92	33.46	33.15	33.34	**33.71**
Average	30.22	31.84	31.82	32.33	32.24	32.11	**32.38**

Table 3.3: PSNR (dB) of different methods evaluated on *Set 14* [2]. (The best result in each row is in blod.)

(a) Input LR image.

(b) DeepAM (PSNR = 33.34 dB).

(c) $\text{DeepAM}_{\text{bp}}$ (PSNR = 33.71 dB).

Figure 3.12: The input LR and the reconstructed HR image of DeepAM and $\text{DeepAM}_{\text{bp}}$.

patch since the input LR patch does not contain sufficient information to predict the boundary pixels. DeepAM achieves a comparable performance to the existing methods.

Its average PSNR is around 0.3 dB higher than that of the SC method and the ANR method, while it is around 0.2 dB lower than that of the A+ method and is around 0.1 dB lower than that of the SRCNN method. $\text{DeepAM}_{\text{bp}}$ achieves the highest average PSNR. Figure 3.12 shows an example of the input LR image and the reconstructed HR images using DeepAM and $\text{DeepAM}_{\text{bp}}$.

3.6 Summary

In this chapter, we proposed a Deep Analysis Dictionary Model (DeepAM) framework which consists of multiple layers of analysis dictionary and soft-thresholding operators and a layer of synthesis dictionary. Each analysis dictionary has been designed to contain two sub-dictionaries: an Information Preserving Analysis Dictionary (IPAD) and a Clustering Analysis Dictionary (CAD). The IPAD and threshold pairs are to pass key information from input to deeper layers. The function of the CAD and threshold pairs is to facilitate discrimination of key features. We proposed an extension of GOAL method [63] to perform dictionary learning for both the IPAD and the CAD. The thresholds have been efficiently set according to simple principles, while leading to effective models. Simulation results show that our proposed DeepAM achieves comparable performance with DNNs and other existing single image super-resolution methods.

Chapter 4

Learning Deep Convolutional Analysis Dictionary Models

4.1 Introduction

Convolutional dictionary learning has attracted increasing interests in signal and image processing communities as it leads to a more elegant framework for high-dimensional signal analysis. The convolutional dictionary [68, 71–77, 89, 97–100] is a global model and takes the high-dimensional signal as input for joint sparse representation and processing, whereas traditional patch-based approaches [1, 2, 12, 60, 63, 101, 102] usually divide the high-dimensional signal into overlapping low-dimensional patches, perform sparse representation on each patch independently, and estimate the high-dimensional output signal through patch averaging.

A convolutional dictionary models the convolution between a set of convolutional filters and a signal. It is a structured dictionary and can be represented as a concatenation of Toeplitz matrices where each Toeplitz matrix is constructed using the taps of a filter. The convolutional dictionaries make the assumption that the filters are shift invariant

and with compact support. Therefore, a convolutional dictionary is effective for processing high-dimensional signals while also restraining the number of free parameters.

To achieve efficient convolutional dictionary learning, the convolutional dictionary is usually modelled as a concatenation of circulant matrices [68, 73–77] by assuming a periodic boundary condition on the signals. As all circulant matrices share the same set of eigenvectors which is the discrete Fourier transform (DFT) matrix, a circular convolution can be therefore represented as a multiplication in Fourier domain and can be efficiently implemented using Fast Fourier Transform (FFT). However, using a circulant matrix to approximate a general Toeplitz matrix may lead to boundary artifacts [74, 103, 104] especially when the boundary region is large.

A multi-layer convolutional dictionary model is able to represent multiple levels of abstraction of the input signal. Due to the associativity property of convolution, multiplying two convolutional dictionaries results in a convolutional dictionary whose corresponding filters are the convolution of two set of filters and have support size that is larger than the original filters. In the Multi-Layer Convolutional Sparse Coding (ML-CSC) model in [89, 99, 105], there are multiple layers of convolutional synthesis dictionaries. By increasing the number of layers, the global dictionary which is the multiplication of the convolutional dictionaries is able to represent more complex structures with simple convolutional filters. The ML-CSC model [99] provides theoretical insights on the conditions for success layered sparse pursuit and uses it as a way to interpret the forward pass of Deep Neural Networks (DNNs) as a layered sparse pursuit.

Convolutional Neural Networks (CNNs) [78,79,106] have been widely used in processing images and achieve state-of-the-art performances in many applications. A CNN consists of a cascade of convolution operations and element-wise non-linear operations. The Rectified Linear Unit (ReLU) activation function is one of the most popular non-linear operators used in DNNs. With the multi-layer convolution structure, a deeper layer in a CNN receives information from a corresponding wider region on the input signal.

The ReLU operator provides a non-linear transformation to convolution representation and leads to a sparse feature representation. The parameters of a CNN are usually optimized using the backpropagation algorithm [20] with stochastic gradient descent. As a consequence of that it may be difficult to interpret the functions of each learned convolutional kernel and its corresponding non-linear operator.

In this chapter, we propose a Deep Convolutional Analysis Dictionary Model (DeepCAM) which consists of multiple layers of convolutional analysis dictionaries and soft-thresholding operations and a layer of convolutional synthesis dictionary. The motivation is to use DeepCAM as a tool to interpret the workings of CNNs from the sparse representation perspective. Single image super-resolution [15, 17–19] is used a sample application to validate the propose model design. The input low-resolution image is used as the input of DeepCAM without being partitioned into patches. At each layer of DeepCAM, the input signal is multiplied with a convolutional analysis dictionary and then passed through soft-thresholding operators as a non-linear transformation. At the last layer, the convolutional synthesis dictionary is used to predict the high-resolution image.

The contribution of this chapter is two-fold:

- We propose a Deep Convolutional Analysis Dictionary Model (DeepCAM) for single image super-resolution. The proposed architecture is made of multiple layers of convolutional analysis dictionary and soft-thresholding and a single layer of convolutional synthesis dictionary. At each layer, the convolutional analysis dictionary and the soft-thresholding operation are designed to achieve simultaneous information preservation and discriminative representation generation.

- We propose a convolutional analysis dictionary learning method by explicitly modelling the convolutional dictionary with a Toeplitz structure. By exploiting the properties of Toeplitz matrices, the convolutional analysis dictionary can be

efficiently learned from a set of training samples. Simulation results on single image super-resolution are used to validate our proposed DeepCAM and convolutional dictionary learning method.

The rest of the chapter is organized as follows. Section 4.2 introduces convolutional analysis dictionary. In Section 4.5, we propose an efficient convolutional analysis dictionary learning algorithm by exploiting the properties of a convolution dictionary. Section 4.4 presents the proposed Deep Convolutional Analysis Dictionary Model (DeepCAM) and the learning algorithm. Section 4.6 presents simulation results on single image super-resolution task and Section 4.7 draws conclusions.

4.2 Convolutional Analysis Dictionary

An analysis dictionary $\mathbf{\Omega} \in \mathbb{R}^{m \times n}$ contains m row atoms $\{\boldsymbol{\omega}_i^T \in \mathbb{R}^n\}_{i=1}^m$ and is usually assumed to be over-complete with $m \geq n$. Given a signal of interests $\boldsymbol{\alpha} \in \mathbb{R}^n$, the analysis dictionary $\mathbf{\Omega}$ should be able to sparsify $\boldsymbol{\alpha}$ while preserving its essential information. That is, the analysis coefficients $\mathbf{\Omega}\boldsymbol{\alpha}$ are sparse but still contain sufficient information for further processing.

The focus of this paper is on learning convolutional analysis dictionaries which model the convolution between a signal and a set of filters. The filters' taps depend on the rows of the analysis dictionary. In what follows, we first show how we build our convolutional analysis dictionary from a unstructured analysis dictionary. We then study strategies to learn a convolutional analysis dictionary from a set of training samples.

The convolution can be represented as a multiplication with a Toeplitz matrix. Let us assume that the input signal is 1-dimensional for simplicity. The convolution between

an atom $\boldsymbol{\omega}_i \in \mathbb{R}^n$ and an input signal $\boldsymbol{x} \in \mathbb{R}^l$ with $l > n$ can be expressed as:

$$\boldsymbol{\omega}_i * \boldsymbol{x} = \mathbf{T}(\boldsymbol{\omega}_i^T, l)\boldsymbol{x}, \tag{4.1}$$

where $*$ denotes the convolution operator, and $\mathbf{T}(\boldsymbol{\omega}_i^T, l)$ is a Toeplitz matrix with l columns which is constructed using $\boldsymbol{\omega}_i$ as follows:

$$\mathbf{T}(\boldsymbol{\omega}_i^T, l) = \sum_{j=1}^{l} \boldsymbol{\omega}_i(j)\mathbf{T}_j, \tag{4.2}$$

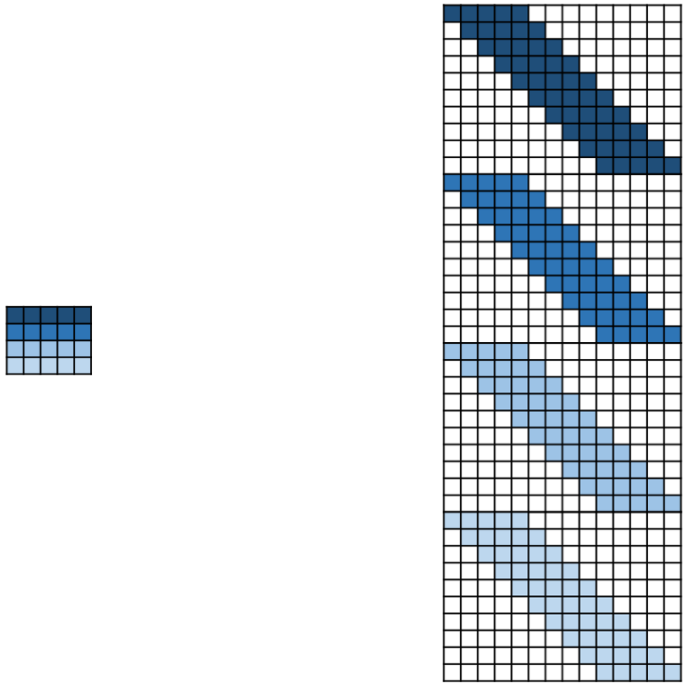
where $\boldsymbol{\omega}_i(j)$ is the j-th coefficient of $\boldsymbol{\omega}_i$, and $\mathbf{T}_j \in \mathbb{R}^{p\times l}$ with $p = l-n+1$ is an indicator matrix with 1s on the j-th upper diagonal and 0s on other locations.

Given an analysis dictionary $\boldsymbol{\Omega} \in \mathbb{R}^{m\times n}$ and an input signal $\boldsymbol{x} \in \mathbb{R}^l$, their convolution can be expressed as a matrix multiplication where $\boldsymbol{\Omega}$ is converted to a convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ which can be represented as a concatenation of Toeplitz matrices along the column direction:

$$\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l) = [\mathbf{T}(\boldsymbol{\omega}_1^T, l); \cdots ; \mathbf{T}(\boldsymbol{\omega}_m^T, l)]. \tag{4.3}$$

Instead of assuming a circulant structure, we model the convolutional operation with a Toeplitz matrix. The represented convolution operation will be performed only within the input signal. That is, convolutional operation is performed without padding at the boundaries.

Fig. 4.1 shows an example of how we build a convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ which is in a form of a concatenation of m Toeplitz matrices $\{\mathbf{T}(\boldsymbol{\omega}_i^T, l)\}_{i=1}^m$ from the unstructured analysis dictionary $\boldsymbol{\Omega} \in \mathbb{R}^{m\times n}$ with $m = 4, n = 5$ and $l = 12$. Note that the analysis dictionary $\boldsymbol{\Omega}$ in Fig. 4.1(a) is not over-complete, while the convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ has more rows than columns.

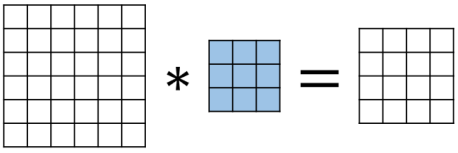


(a) An analysis dictionary $\mathbf{\Omega}$.

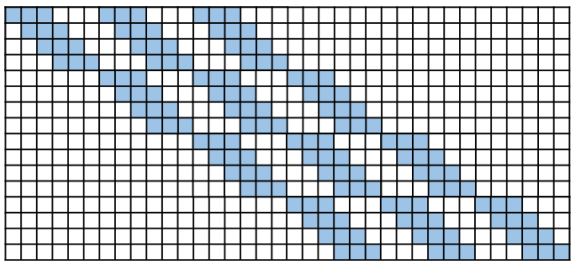
(b) The convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\mathbf{\Omega}, l)$.

Figure 4.1: **An analysis dictionary and the corresponding convolutional dictionary.** A convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\mathbf{\Omega}, l)$ with $l = 12$ is a concatenation of $m = 4$ Toeplitz matrices $\{\mathbf{T}(\boldsymbol{\omega}_i^T, l)\}_{i=1}^m$.

The above description of convolutional analysis dictionary applies to 1-dimensional input signals, but it can be extended to multi-dimensional signals, like for example images. A convolutional analysis dictionary will then be in the form of a concatenation of doubly block Toeplitz matrices (i.e. a matrix with block Toeplitz structure where each block is a Toeplitz matrix). Similar to Eqn. (4.2), the doubly block Toeplitz structure can be represented with corresponding indicator matrices. Fig. 4.2(a) shows an example of 2-dimensional convolution between a 6×6 image region and a 3×3 filter and Fig. 4.2(b) shows the corresponding convolutional analysis dictionary.



(a) The convolution operation between a 6×6 image region and a 3×3 filter.



(b) The corresponding Toeplitz matrix.

Figure 4.2: **A example of convolution between a 2-dimensional convolutional filter with 2-dimensional data and the convolution can be represented by a doubly block Toeplitz matrix.**

4.3 Learning Convolutional Analysis Dictionaries

In this section, we propose an efficient convolutional analysis dictionary learning algorithm by exploiting the properties of the Toeplitz structure within the dictionary.

For simplicity, let us assume that the input data is made of 1-dimensional vectors. Therefore, the convolutional analysis dictionary is a concatenation of Toeplitz matrices as we discussed in Section **??**. The proposed learning algorithm can be easily extended to the multi-dimensional case where the convolutional analysis dictionary is a concatenation of doubly block Toeplitz matrices.

Let us assume that convolution is performed between an analysis dictionary $\boldsymbol{\Omega} \in \mathbb{R}^{m \times n}$ and an input signal $\boldsymbol{x} \in \mathbb{R}^{l}$ with $l > n$. Therefore, the convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ will be of size $q \times l$ with $q = m(l - n + 1)$. Given that $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ is built using $\boldsymbol{\Omega}$, it has the same number of free parameters as $\boldsymbol{\Omega}$ despite $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ being a much bigger matrix. This means that if we were to optimize $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ directly we would end up with a computationally inefficient approach.

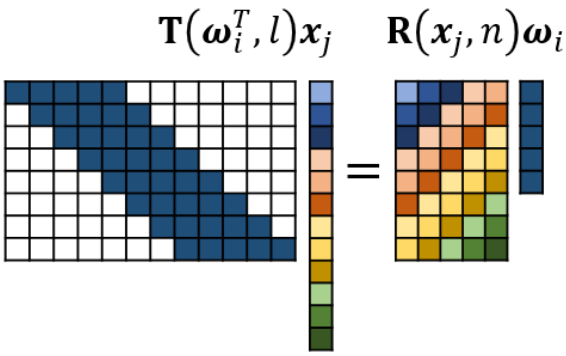


Figure 4.3: **Convolution expressed using right dual matrix.** Convolution can be represented as multiplying with a Toeplitz matrix or multiplying with a right dual matrix. The right dual matrix is not a sparse matrix.

Symbol	$\boldsymbol{\Omega}$	$\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$	$\mathbf{T}(\boldsymbol{\omega}_i^T, l)$	$\mathbf{R}(\boldsymbol{u}_{Si}, n)$	$\boldsymbol{U}_S$	$\boldsymbol{U}_N$	$\boldsymbol{V}$	$\boldsymbol{P}_S$	$\boldsymbol{P}_{\boldsymbol{\omega}}$
Size	$m \times n$	$pm \times l$	$p \times l$	$p \times n$	$l \times K$	$l \times (l-K)$	$pm \times K$	$l \times l$	$n \times n$

Table 4.1: A list of symbols and their dimensions. For simplicity, in the table we denote $p = l - n + 1$ with $l \gg n$.

We mitigate this issue by first observing that as we have assumed that the convolutional filters are with compact support $n \ll l$, the convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ has many zero entries. It is therefore inefficient to evaluate $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)\boldsymbol{x}$ by directly multiplying $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ with $\boldsymbol{x}$. As illustrated in Fig. 4.3, with the commutativity property of convolution (i.e. $\boldsymbol{a} * \boldsymbol{b} = \boldsymbol{b} * \boldsymbol{a}$), the matrix multiplication between a Toeplitz matrix $\mathbf{T}(\boldsymbol{\omega}_i^T, l) \in \mathbb{R}^{p \times l}$ with $p = l - n + 1$ and an input vector $\boldsymbol{x} \in \mathbb{R}^l$ can be efficiently implemented as:

$$\mathbf{T}(\boldsymbol{\omega}_i^T, l)\boldsymbol{x} = \mathbf{R}(\boldsymbol{x}_j, n)\boldsymbol{\omega}_i, \tag{4.4}$$

where $\mathbf{R}(\boldsymbol{x}, n) \in \mathbb{R}^{p \times n}$ is the right dual matrix of $\mathbf{T}(\boldsymbol{\omega}_i^T, l)$ and is defined as:

$$\mathbf{R}(\boldsymbol{x}, n) = \sum_{j=1}^{L} \boldsymbol{x}(j)\mathbf{R}_j, \tag{4.5}$$

where $\boldsymbol{x}(j)$ is the j-th coefficient of $\boldsymbol{x}$, and $\mathbf{R}_j \in \mathbb{R}^{p \times n}$ is an indicator matrix with 1s on the j-th skew-diagonal and 0s on other locations. Note that the right dual matrix $\mathbf{R}(\boldsymbol{x}, n)$ is a matrix without non-zero entries and has $n \ll l$ columns.

Secondly, whenever possible, we will pose the optimization problem using $\boldsymbol{\Omega}$ whilst imposing the constraints associated with the structured matrix $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ as the actual analysis dictionary learning problem.

We want the convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ to satisfy four properties: (i) its row atoms span the input data space; (ii) it is able to sparsify the input data; (iii) the row atoms are of unit norm; (iv) there are no pairs of row atoms in $\boldsymbol{\Omega}$ that are linearly dependent.

Different from the unstructured analysis dictionary learning case, we propose to use two sets of input training data with different sizes. Let us denote the super-patch training data as $\boldsymbol{\mathcal{X}}_S = [\boldsymbol{x}_{S1}, \boldsymbol{x}_{S2}, \cdots, \boldsymbol{x}_{SN_1}] \in \mathbb{R}^{l \times N_1}$ and denote the small patch training data as $\boldsymbol{\mathcal{X}} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{N_2}] \in \mathbb{R}^{n \times N_2}$. Both the super-patch and small patch training datasets are extracted from an external training dataset. The super-patch training data will be used to impose the property (i) which is a global property of the convolutional dictionary. The small patch training data will be used to impose property (ii).

The first learning objective is that the convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ should be able to span the input data space in order to preserve the information within the input super-patch training data $\boldsymbol{\mathcal{X}}_S$. The super-patch training dataset defines the subspace covered by the input data. Let us denote with $\boldsymbol{U}_S \in \mathbb{R}^{l \times K}$ the orthogonal basis covering the signal subspace of the input super-patch data $\boldsymbol{\mathcal{X}}_S$ where we assume that this subspace has dimension K, we also denote with $\boldsymbol{U}_N \in \mathbb{R}^{l \times (l-K)}$ the orthogonal basis of the orthogonal complement to the signal subspace of $\boldsymbol{\mathcal{X}}_S$. These two bases will be used to impose that the row space of the learned convolutional analysis dictionary spans the input data space while being orthogonal to the null-space of $\boldsymbol{\mathcal{X}}_S$. The information preservation constraint can be interpreted as a rank constraint on the convolutional analysis dictionary which is usually achieved by imposing a logarithm

determinant constraint:

$$h(\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)) = -\frac{1}{K \log K} \log \det \left(\frac{1}{q} \boldsymbol{U}_S^T \boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)^T \boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l) \boldsymbol{U}_S \right). \tag{4.6}$$

The size of the convolutional analysis dictionary can be huge, especially when the input data is multi-dimensional. Therefore it would be computationally inefficient to evaluate Eqn. (4.6) and its first order derivative directly. By exploiting the properties of the convolutional analysis dictionary, we propose an efficient reformulation of Eqn. (4.6) which is based on the analysis dictionary $\boldsymbol{\Omega}$.

With the definition of the right dual matrix, the multiplication between $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ and the j-th orthogonal basis element $\boldsymbol{u}_{Sj}$ can be expressed as:

$$\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l) \boldsymbol{u}_{Sj} = \mathrm{vec}\left(\mathbf{R}(\boldsymbol{u}_{Sj}, n) \boldsymbol{\Omega}^T \right), \tag{4.7}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization operation $\mathrm{vec}(\boldsymbol{A}) : \mathbb{R}^{m_1 \times \cdots \times m_{D_a}} \to \mathbb{R}^{\prod_{k=1}^{D_a} m_k}$, the vectorization operation for 2-dimensional signal can be expressed as:

$$\mathrm{vec}(\boldsymbol{A}) = \sum_{i=1}^{n} \boldsymbol{e}_i \otimes \boldsymbol{A} \boldsymbol{e}_i, \tag{4.8}$$

where $\boldsymbol{e}_i$ is the i-th canonical basis vector of $\mathbb{R}^n$, that is, $\boldsymbol{e}_i = [0, \cdots, 0, 1, 0, \cdots, 0]^T \in \mathbb{R}^n$ (with 1 on the i-th location), and $\otimes$ denotes the Kroneckers product.

The information preservation constraint in Eqn. (4.6) can therefore be reformulated and expressed based on the analysis dictionary $\boldsymbol{\Omega}$ as:

$$h(\boldsymbol{\Omega}) = -\frac{1}{K \log K} \log \det \left(\frac{1}{q} \boldsymbol{V}^T \boldsymbol{V} \right), \tag{4.9}$$

where $\boldsymbol{V} = [\boldsymbol{v}_i, \cdots, \boldsymbol{v}_K]$ with $\boldsymbol{v}_i = \mathrm{vec}(\mathbf{R}(\boldsymbol{u}_{Si}, n) \boldsymbol{\Omega}^T)$.

The gradient of $h(\mathbf{\Omega})$ can be expressed as:

$$\frac{\partial}{\partial \mathbf{\Omega}} h(\mathbf{\Omega}) = -\frac{2}{K\log(K)} \left(\sum_{i=1}^{K} \mathbf{\Omega}\mathbf{\Sigma}_{Si} \right) \mathbf{\Sigma}_V^{-1}, \tag{4.10}$$

where $\mathbf{\Sigma}_{Si} = \mathbf{R}(\boldsymbol{u}_{Si}, n)^T \mathbf{R}(\boldsymbol{u}_{Si}, n)$ and $\mathbf{\Sigma}_V = \boldsymbol{V}^T \boldsymbol{V}$.

With the information preservation constraint in Eqn. (4.9), the learned $\boldsymbol{\mathcal{H}}(\mathbf{\Omega}, l)$ is constrained to span the signal subspace defined by $\boldsymbol{U}_S$. However, we still need to exclude the null-space components of the training data from $\boldsymbol{\mathcal{H}}(\mathbf{\Omega}, l)$. Specifically, the Toeplitz matrix $\mathbf{T}(\boldsymbol{\omega}_i^T, l)$ should not be within the subspace spanned by $\boldsymbol{U}_N$ to avoid a zero response when multiplying with $\boldsymbol{\mathcal{X}}_S$.

Therefore we define the feasible set of the convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\mathbf{\Omega}, l)$ as $\Theta_{\mathcal{H}} = \mathbb{S}_{l-1}^q \cap \boldsymbol{U}_N^{\perp}$ with $\mathbb{S}_{l-1}$ being the unit sphere in $\mathbb{R}^l$ and $\boldsymbol{U}_N^{\perp}$ being the orthogonal complement of $\boldsymbol{U}_N$. The unit sphere constraint ensure that the unit norm condition is satisfied. The feasible set $\Theta_{\mathcal{H}}$ is defined in $\mathbb{R}^l$, while we wish to have a feasible set for $\mathbf{\Omega}$ which is defined in $\mathbb{R}^n$ with $n \ll l$ and can be more efficiently implemented.

The operation of orthogonal projection onto the complementary subspace of $\boldsymbol{U}_N$ can be represented by the projection matrix $\boldsymbol{P}_S \in \mathbb{R}^{l \times l}$ given by:

$$\boldsymbol{P}_S = \mathbf{I}_l - \boldsymbol{U}_N^{T\dagger} \boldsymbol{U}_N^T, \tag{4.11}$$

where $\mathbf{I}_l \in \mathbb{R}^{l \times l}$ is the identity matrix and $\boldsymbol{U}_N^{T\dagger}$ is the pseudo-inverse of $\boldsymbol{U}_N^T$.

The orthogonal projection operation is achieved by multiplying the convolutional analysis dictionary with the projection matrix. The projection is applied on the rows of $\boldsymbol{\mathcal{H}}(\mathbf{\Omega}, l)$. With the definition of the right dual matrix, the orthogonal projection operation can be expressed in terms of the analysis dictionary atom $\boldsymbol{\omega}_i^T$ as:

$$\boldsymbol{P}_S \mathbf{T}(\boldsymbol{\omega}_i^T, l)^T = \left[\mathbf{R}(\boldsymbol{p}_{S1}, n)\boldsymbol{\omega}_i, \cdots, \mathbf{R}(\boldsymbol{p}_{Sl}, n)\boldsymbol{\omega}_i \right]^T, \tag{4.12}$$

where $\boldsymbol{p}_{Sj}^T$ denotes the j-th row of $\boldsymbol{P}_S$.

We note that, after the projection, the Toeplitz structure within $\mathbf{T}(\boldsymbol{\omega}_i^T, l)\boldsymbol{P}_S^T$ may not be preserved and needs to be imposed again. The Toeplitz matrix closest to $\mathbf{T}(\boldsymbol{\omega}_i^T, l)\boldsymbol{P}_S^T$ can be obtained by averaging over the diagonal elements. The orthogonal projection operation and the averaging operation can be jointly represented and applied to the atoms of the analysis dictionary. Let us define a vector $\boldsymbol{p}_j$ whose inner product with $\boldsymbol{\omega}_i$ equals the average value of the j-th diagonal elements of $\mathbf{T}(\boldsymbol{\omega}_i^T, l)\boldsymbol{P}_S^T$, it can be expressed as:

$$\boldsymbol{p}_j^T = \frac{1}{n}\sum_{k=j}^{j+n-1} \boldsymbol{r}_{Sk,k-j+1}^T, \tag{4.13}$$

where $\boldsymbol{r}_{Sk,i}^T$ denotes the i-th row of $\mathbf{R}(\boldsymbol{p}_{Sk}, n)$.

The matrix $\boldsymbol{P} = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n]^T \in \mathbb{R}^{n\times n}$ therefore represents simultaneously the orthogonal projection operation and the averaging operation. Let us denote the feasible set of the analysis dictionary $\boldsymbol{\Omega}$ as $\Theta_\Omega = \mathbb{S}_{n-1}^m \cap \boldsymbol{W}_N^\perp$ where $\mathbb{S}_{n-1}^m$ represents unit sphere in $\mathbb{R}^n$ and $\boldsymbol{W}_N^\perp$ represents the orthogonal complementary subspace of the null-space of $\boldsymbol{\mathcal{X}}$. The operation of the orthogonal projection onto the tangent space $\mathcal{T}_{\boldsymbol{\omega}}(\Theta_\Omega)$ can be represented by a projection matrix $\boldsymbol{P}_{\boldsymbol{\omega}}$:

$$\boldsymbol{P}_{\boldsymbol{\omega}} = \boldsymbol{P}\left(\mathbf{I}_n - \boldsymbol{Q}_{\boldsymbol{\omega}}^\dagger \boldsymbol{Q}_{\boldsymbol{\omega}}\right), \tag{4.14}$$

where $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix, and $\boldsymbol{Q}_{\boldsymbol{\omega}} = [2\boldsymbol{\omega}^T; \boldsymbol{W}_N^T] \in \mathbb{R}^{(n-k+1)\times n}$.

The sparsifying property of the convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}, l)$ over the super-patch training data can be achieved by imposing the sparsifying property of the analysis dictionary $\boldsymbol{\Omega}$ over the small patch training data. The rationale is that the row atoms of the convolutional analysis dictionary only operate on local regions of the input signal as illustrated in Eqn. (4.4) and Fig. 4.3. Similar to [63, 69, 107], the sparsifying constraint is imposed by using a log-square function which promotes the

analysis dictionary atoms to sparsify the small patch training data:

$$g(\mathbf{\Omega}) = \frac{1}{N_2 m \log(1+\nu)} \sum_{i=1}^{N_2} \sum_{j=1}^{m} \log\left(1 + \nu(\boldsymbol{\omega}_j^T \boldsymbol{x}_i)^2\right), \tag{4.15}$$

where ν is a tunable parameter which controls the sparsifying ability of the learned dictionary.

The linearly dependent penalty and the unit norm constraint can also be imposed on the analysis dictionary $\mathbf{\Omega}$. Linearly dependent row atoms (e.g. $\boldsymbol{\omega}_i^T = \pm\boldsymbol{\omega}_j^T$) are penalized by using a logarithm barrier term $l(\mathbf{\Omega})$:

$$l(\mathbf{\Omega}) = -\frac{1}{m(m-1)} \sum_{1 \le i < j \le m} \log\left(1 - \left(\boldsymbol{\omega}_i^T \boldsymbol{\omega}_j\right)^2\right). \tag{4.16}$$

We observe that, by exploiting the Toeplitz structure, we have been able to impose the desired proprieties of a convolutional analysis dictionary by imposing constraints on the lower-dimensional analysis dictionary $\mathbf{\Omega}$. This will reduce computational costs and memory requirements.

Combining the information preservation constraint in Eqn. (4.9), feasible set constraint in Eqn. (4.14), sparsifying constraint in Eqn. (4.15), and linearly dependent penalty term in Eqn. (4.16), the objective function for the convolutional analysis dictionary problem can be expressed as:

$$\mathbf{\Omega} = \arg \min_{\mathbf{\Omega}^T \in \Theta_{\mathbf{\Omega}}} f(\mathbf{\Omega}), \tag{4.17}$$

where $f(\mathbf{\Omega}) = g(\mathbf{\Omega}) + \kappa h(\mathbf{\Omega}) + \upsilon l(\mathbf{\Omega})$ with κ and υ being the regularization parameters.

The proposed convolutional analysis dictionary learning algorithm ConvGOAL+ is summarized in **Algorithm 3**. The objective function in Eqn. (4.17) is optimized using a geometric conjugate gradient descent method. For a more detailed introduction

Algorithm 3: ConvGOAL+ Algorithm

Input: the number of convolutional filters m, the support size n, training data $\boldsymbol{\mathcal{X}}_S \in \mathbb{R}^{l \times N_1}$ and $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n \times N_1}$;
Initialize: Initialized $\boldsymbol{\Omega}^{(0)} \in \mathbb{R}^{m \times n}$, $t = 0$;
Find the orthogonal basis $\boldsymbol{U}_S$ and $\boldsymbol{U}_N$ for the input data subspace of $\boldsymbol{\mathcal{X}}_S$ and its orthogonal complementary subspace, respectively;
Find the orthogonal basis $\boldsymbol{W}_N$ for the orthogonal complementary subspace of $\boldsymbol{\mathcal{X}}$;
while *halting criterion false* **do**
  $t \leftarrow t + 1$;
  Compute gradient of the objective function $\nabla f(\boldsymbol{\Omega}^{(t)})$;
  Orthogonal project $\nabla f(\boldsymbol{\Omega}^{(t)})$ onto the tangent space of manifold Θ_Ω at $\boldsymbol{\Omega}^{(t)}$;
  Update $\boldsymbol{\Omega}^{(t+1)}$ along the search direction using backtracking line search.
end
Output: Learned analysis dictionary $\boldsymbol{\Omega}$.

about geometric conjugate gradient descent, we refer to [63, 92, 107].

4.4 Deep Convolutional Analysis Dictionary Model

In this section, we propose a Deep Convolutional Analysis Dictionary Model (DeepCAM) which consists of multiple layers of analysis dictionary and soft-thresholding operations and a single layer of synthesis dictionary. DeepCAM is a convolutional extension of the Deep Analysis dictionary Model (DeepAM) [107]. Different from DeepAM which is patch-based, DeepCAM performs convolution operation and element-wise soft-thresholding at image level on all layers without dividing the input image into patches.

When it comes to Single Image Super-Resolution (SISR), Convolutional Neural Networks are designed using two main strategies: the early upsampling approach [15, 17] and the late upsampling approach [18, 19]. The early upsampling approach [15, 17] first upsamples the low-resolution (LR) image to the same resolution as the desired high-resolution (HR) image through bicubic interpolation and then performs convolution on the upsampled image. The drawback of this approach is that this leads to a

large number of model parameters and a high computational complexity during testing as the feature maps are of the same size as the HR image. The late upsampling approach [18, 19] performs convolution on the input LR image and applies a deconvolution layer [18] or a sub-pixel convolution layer [19] at the last layer to predict the HR image. The late upsampling approach has smaller number of parameters and lower computational cost than the early upsampling one.

SISR is used as a sample application to validate our proposed design. We utilize a similar strategy as the late upsampling approach. The LR image is used as input to DeepCAM without bicubic interpolation. At each layer, the convolution and soft-thresholding operations are applied to the corresponding input signal. For SISR with up-sampling factor s, the synthesis dictionary consists of s^2 atoms. The convolution between the synthesis dictionary and its input signal yields s^2 output channels which correspond to s^2 sub-sampled version of the HR image. The final predicted HR image can then be obtained by reshaping and combing the s^2 output channels.

The parameters of a L-layer DeepCAM include L layers of analysis dictionary and soft-thresholds pair $\{(\boldsymbol{\Omega}_i, \boldsymbol{\lambda}_i)\}_{i=1}^{L}$ and a single synthesis layer modelled with dictionary $\boldsymbol{D}$. The atoms of the dictionaries represent filters. Let us denote with d_i the number of filters at the i-th layer, denote with $p_i \times p_i$ the spatial support size of the convolutional filters, and denote with $n_i = p_i^2 d_{i-1}$ the support size of the i-th layer filter. Therefore, the size of the parameters can be denoted with the analysis dictionary $\{\boldsymbol{\Omega}_i \in \mathbb{R}^{d_i \times n_i}\}_{i=1}^{L}$, the soft-thresholds $\{\boldsymbol{\lambda}_i \in \mathbb{R}^{d_i}\}_{i=1}^{L}$ and the synthesis dictionary $\boldsymbol{D} \in \mathbb{R}^{d_{L+1} \times n_{L+1}}$. There are $d_{L+1} = s^2$ output channels at the last layer and each output channel corresponds to a sub-sampled version of the HR image.

Fig. 4.4 shows an example of the forward model of a 2-layer DeepCAM for 2× image super-resolution. The input LR image denoted with $\boldsymbol{X}_0$ passes through multiple layers of convolution with the analysis dictionary and soft-thresholding. There are 4 synthesised HR sub-images $\widehat{\boldsymbol{Y}}$ which are obtained by convolving the last layer analysis

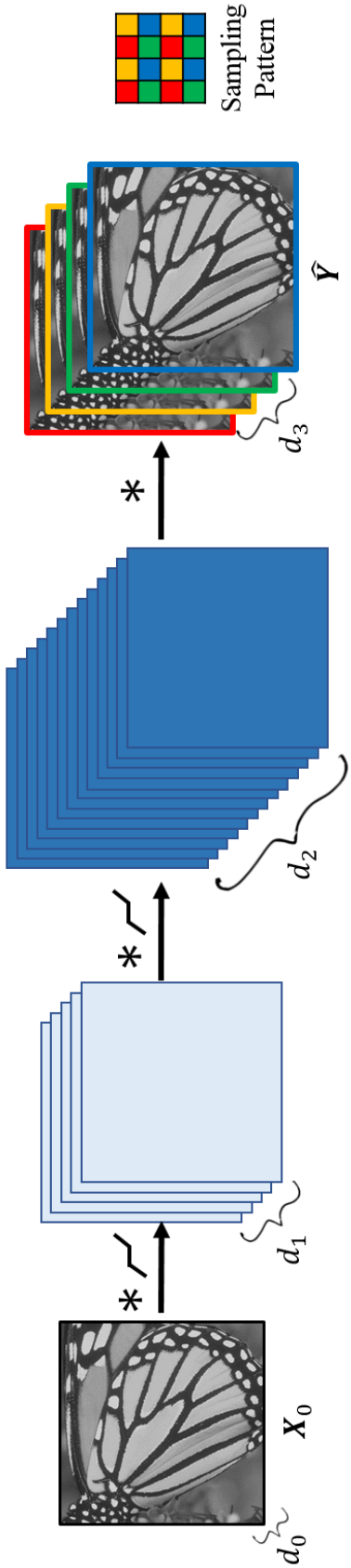


Figure 4.4: **A 2-layer Deep Convolutional Analysis dictionary Model for 2× single image super-resolution.** There are 2 layers of analysis dictionaries $\{\boldsymbol{\Omega}_i\}_{i=1}^2$ with element-wise soft-thresholding operators $\{\mathcal{S}_{\boldsymbol{\lambda}_i}(\cdot)\}_{i=1}^2$ and a layer of synthesis dictionary $\boldsymbol{D}$. The input image is a gray image with a single channel. The estimated $d_3 = 4$ HR images $\widehat{\boldsymbol{Y}}$ is obtained through a cascaded convolution operation and soft-thresholding operation with input LR image $\boldsymbol{X}_0$. The final predicted HR image is obtained by reshaping $\widehat{\boldsymbol{Y}}$ according to the sampling pattern.

feature maps with the synthesis dictionary $\boldsymbol{D}$ and will be rearranged to generate the final predicted HR image according to the sampling pattern.

Let us denote with $\boldsymbol{X}_{i-1} \in \mathbb{R}^{W_{i-1} \times H_{i-1} \times d_{i-1}}$ the input signal at the i-th layer, and denote with $\boldsymbol{\omega}_{i,j}^T \in \mathbb{R}^{p_i^2 d_{i-1}}$ the j-th atom of $\boldsymbol{\Omega}_i$. The convolution and soft-thresholding operations corresponding to the j-th atom and threshold pair $(\boldsymbol{\omega}_{i,j}, \boldsymbol{\lambda}_i(j))$ can be expressed as:

$$\begin{aligned} \boldsymbol{F}_{i,j} &= \mathrm{mat}(\boldsymbol{\omega}_{i,j}) * \boldsymbol{X}_{i-1}, \\ \boldsymbol{X}_{i,j} &= \mathcal{S}_{\boldsymbol{\lambda}_i(j)}(\boldsymbol{F}_{i,j}), \end{aligned} \tag{4.18}$$

where $\mathrm{mat}(\cdot)$ represents the operation which reshapes a vector of length $p_i^2 d_{i-1}$ to a tensor with size $p_i \times p_i \times d_{i-1}$, $\boldsymbol{F}_{i,j} \in \mathbb{R}^{W_i \times H_i}$ is the convolution result, $\mathcal{S}_\lambda(\cdot)$ denotes the element-wise soft-thresholding operation with threshold λ, and $\boldsymbol{X}_{i,j} \in \mathbb{R}^{W_i \times H_i}$ is the sparse representation after thresholding.

Fig. 4.5 illustrates the convolution and the soft-thresholding operation described in Eqn. (4.18). The convolution linearly transforms the input signal $\boldsymbol{X}_{i-1}$ to a 2-D representation $\boldsymbol{F}_{i,j}$. An element-wise soft-thresholding operation is then applied to every element on $\boldsymbol{F}_{i,j}$ and generates a sparse 2-D representation $\boldsymbol{X}_{i,j}$.

By stacking the d_i sparse 2-D representation $\{\boldsymbol{X}_{i,j}\}_{j=1}^{d_i}$, the i-th layer output signal can be represented as $\boldsymbol{X}_i \in \mathbb{R}^{W_i \times H_i \times d_i}$. For simplicity, let us denote the i-th layer convolution and soft-thresholding operation as:

$$\boldsymbol{X}_i = \mathcal{S}(\boldsymbol{\Omega}_i * \boldsymbol{X}_{i-1}, \boldsymbol{\lambda}_i). \tag{4.19}$$

When the convolution of $\boldsymbol{\Omega}_i$ and $\boldsymbol{X}_{i-1}$ is represented by a convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_i, l)$ with $l = W_{i-1} H_{i-1} d_{i-1}$, the convolution and soft-thresholding operations

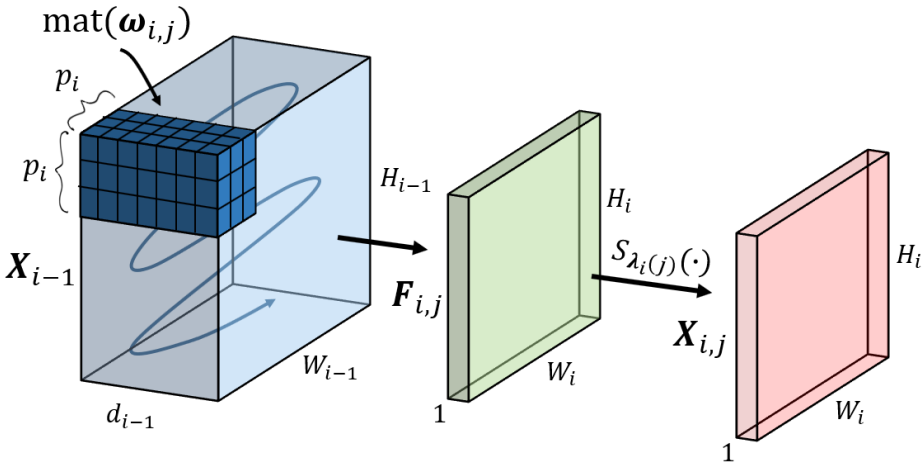


Figure 4.5: **The convolution and soft-thresholding operation corresponding to the atom and threshold pair** $(\boldsymbol{\omega}_{i,j}, \boldsymbol{\lambda}_i(j))$**.** The input signal $\boldsymbol{X}_{i-1}$ is of size $H_{i-1} \times W_{i-1} \times d_{i-1}$. The atom $\boldsymbol{\omega}_{i,j} \in \mathbb{R}^{p_i^2 d_{i-1}}$ represents a convolutional filter with spatial support size $p_i \times p_i$ and d_{i-1} channels. The convolution of $\boldsymbol{\omega}_{i,j}$ and $\boldsymbol{X}_{i-1}$ results in a matrix $\boldsymbol{F}_{i,j}$ of size $W_i \times H_i$. An element-wise soft-thresholding operation $\mathcal{S}_{\boldsymbol{\lambda}_i(j)}(\cdot)$ is applied to every element of $\boldsymbol{F}_{i,j}$ and results in $\boldsymbol{X}_{i,j}$.

can be represented as follows:

$$\boldsymbol{X}_i = \text{mat}\left(\mathcal{S}_{\boldsymbol{\lambda}_i \otimes \mathbf{1}}\left(\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_i, l)\text{vec}(\boldsymbol{X}_{i-1})\right)\right), \tag{4.20}$$

where $\mathbf{1}$ is a all ones vector of size $W_i H_i$, and $\otimes$ is the Kronecker product.

The forward model of a L-layer DeepCAM is a cascade of convolution and soft-thresholding operations and can be expressed as:

$$\widehat{\boldsymbol{Y}} = \boldsymbol{D} * \mathcal{S}\left(\left(\cdots \mathcal{S}\left(\boldsymbol{\Omega}_1 * \boldsymbol{X}_0, \boldsymbol{\lambda}_1\right) \cdots\right), \boldsymbol{\lambda}_L\right), \tag{4.21}$$

where $\widehat{\boldsymbol{Y}}$ denotes the estimated d_{L+1} HR images.

The analysis dictionary and soft-threshold pairs should be able to generate feature representations that contain the essential information of the input LR image while being sufficiently discriminative to predict the missing high frequency information in the LR image.

4.5 Learning A Deep Convolutional Analysis Dictionary Model

In this section, we will introduce the proposed learning algorithm for learning both the convolutional analysis dictionary and the soft-thresholds in DeepCAM.

We adopt a joint *Information Preserving* and *Clustering* strategy for multi-layer analysis dictionary model construction as proposed in DeepAM [107]. At each layer, the analysis dictionary $\boldsymbol{\Omega}_i$ has been divided into two sub-dictionaries: an Information Preserving Analysis Dictionary (IPAD) $\boldsymbol{\Omega}_{\mathrm{I}i}$ and a Clustering Analysis Dictionary (CAD) $\boldsymbol{\Omega}_{\mathrm{C}i}$. The IPAD and soft-threshold pair $(\boldsymbol{\Omega}_{\mathrm{I}i}, \boldsymbol{\lambda}_{\mathrm{I}i})$ will generate feature maps that can preserve the information from the input image. The CAD and soft-threshold pair $(\boldsymbol{\Omega}_{\mathrm{C}i}, \boldsymbol{\lambda}_{\mathrm{C}i})$ will generate feature maps with strong discriminative power that can facilitate the prediction of the HR image. To achieve this goal, there should be a sufficient number of IPAD and CAD atoms.

4.5.1 Learning IPAD and Threshold Pair

The Information Preserving Analysis Dictionary (IPAD) and soft-threshold pair are used to preserve the essential information within the input image. The IPAD will be learned using the proposed convolutional analysis dictionary learning method. The thresholds will be set according to the method used in DeepAM [107].

A multi-layer convolutional analysis dictionary naturally possesses a multi-scale property. The product of two convolutional dictionaries leads to a convolutional dictionary whose equivalent filters are given by the convolution of the filters in the two dictionaries due to the associativity property of convolution (i.e. $\boldsymbol{a} * (\boldsymbol{b} * \boldsymbol{c}) = (\boldsymbol{a} * \boldsymbol{b}) * \boldsymbol{c}$).

Let us denote with $\boldsymbol{\mathcal{H}}_i$ the i-th layer convolutional analysis dictionary constructed using convolutional filters with patch size p_i. The effective convolutional analysis dictionary

$\boldsymbol{\mathcal{H}}^{(i)} = \boldsymbol{\mathcal{H}}_i \boldsymbol{\mathcal{H}}_{i-1} \cdots \boldsymbol{\mathcal{H}}_1$ has filters with spatial patch size:

$$p_{\text{eff},i} = \sum_{j=1}^{i} p_j - (i-1). \tag{4.22}$$

An example of a two-layer convolutional analysis dictionary is shown in Fig. 4.6. The effective dictionary $\boldsymbol{\mathcal{H}}^{(i)}$ has an effective patch size that increases with the number of layers and can achieve a large patch size even with convolutional analysis dictionaries of filters with small patch size.

When the support size of a convolutional analysis dictionary is small, its row atoms can only receive local information from the whole input signal. With an increased effective patch size, the row atoms of the convolutional analysis dictionary at a deeper layer will receive information from a larger segment of the input signal.

For each spatial location at the synthesised HR images $\widehat{\boldsymbol{Y}}$, there is a corresponding super-patch region on each layer which contributes all the information for predicting a HR pixel. Let us denote $p_{S,i}$ as the super-patch size at the i-th layer. It can be expressed in terms of the patch size of the convolutional filters:

$$p_{S,i} = p_{L+1} + \sum_{j=i}^{L} (p_j - 1). \tag{4.23}$$

Fig. 4.7 shows the super-patches at different layers for a 2-layer DeepCAM. Note that the super-patch size at a shallower layer is larger than that in a deeper layer.

In the proposed convolutional analysis dictionary learning method ConvGOAL+, there are two sets of training data: the super-patch training data $\boldsymbol{\mathcal{X}}_S$ and the small patch training data $\boldsymbol{\mathcal{X}}$. The super-patch data $\boldsymbol{\mathcal{X}}_S$ is used to impose the rank constraint. The small patch data $\boldsymbol{\mathcal{X}}$ has the same support as the filters and is used to impose the sparsifying and linear independent constraints.

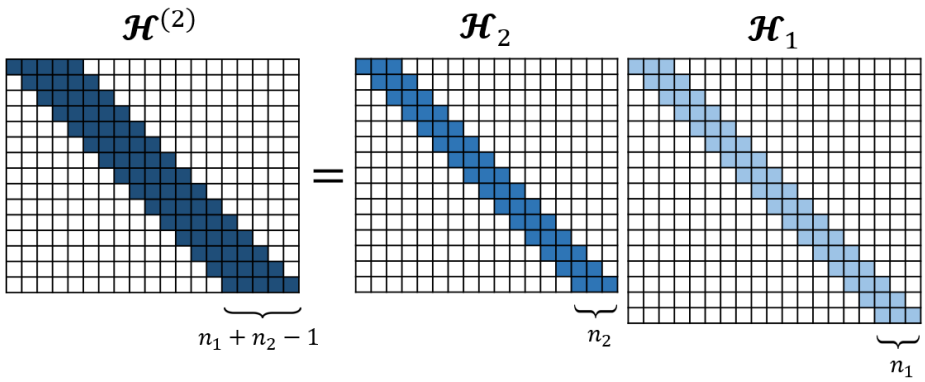


Figure 4.6: **Effective convolutional analysis dictionary.** With a two-layer convolutional analysis dictionary, the effective convolutional analysis dictionary $\boldsymbol{\mathcal{H}}^{(2)} = \boldsymbol{\mathcal{H}}_2\boldsymbol{\mathcal{H}}_1$ is still a convolutional analysis dictionary and with support size $n_2 + n_1 - 1$.

The patch size of the super-patch training data for convolutional analysis dictionary learning should be no smaller than $p_{S,i}$. Otherwise, we can not ensure that the learned convolutional analysis dictionary will be able to utilize all information within the super-patch for predicting the corresponding HR pixel values.

At the i-th layer, let us define the support size of the super-patch training data as $S_i = p_{S,i}^2 d_{i-1}$. The super-patch training data, the small patch training dataset and the ground-truth training dataset are denoted as $\boldsymbol{\mathcal{X}}_S^{i-1} \in \mathbb{R}^{S_i \times N_1}$, $\boldsymbol{\mathcal{X}}^{i-1} \in \mathbb{R}^{n_i \times N_2}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{s^2 \times N_2}$, respectively.

Let us denote $d_{\mathrm{I}i}$ as the number of atoms in $\boldsymbol{\Omega}_{\mathrm{I}i}$. With $\boldsymbol{\Omega}_{\mathrm{I}i} \in \mathbb{R}^{d_{\mathrm{I}i} \times n_i}$, we will have $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_{\mathrm{I}i}, S_i) \in \mathbb{R}^{d_{\mathrm{I}i}(p_{S,i}-p_i+1)^2 \times S_i}$. From the degree of freedom perspective, there should be at least $S_1 = p_{S,1}^2 d_0$ rows in $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_{\mathrm{I}i}, S_i)$ to ensure that information from $\boldsymbol{\mathcal{X}}_S^0$ will be preserved. This leads to:

$$d_{\mathrm{I}i} \geq \frac{p_{S,1}^2 d_0}{(p_{S,i} - p_i + 1)^2}. \tag{4.24}$$

Eqn. (4.24) indicates that there should be more atoms for information preservation in a deeper layer of a DeepCAM. For example, when $d_0 = 1$, $p_1 = 2$, $p_2 = 3$, and $p_3 = 4$ in a 2-layer DeepCAM, there should be at least 2 atoms in the 1-st layer, and 4 atoms in the 2-nd layer for information preservation.

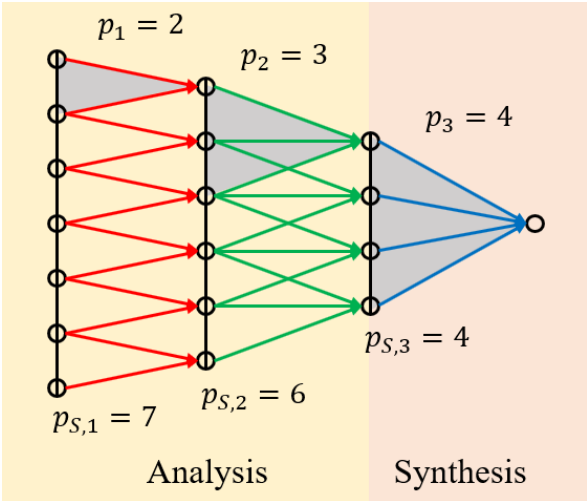


Figure 4.7: **Super-patches at different layers in a 2-layer DeepCAM.** A synthesised pixel value corresponds to a super-patch on the i-th layer with patch size $p_{S,i}$. In this example, the convolutional filters are with size $p_1 = 2$, $p_2 = 3$, and $p_3 = 4$. The super-patch size at layer 1, 2, and 3 is 7, 6, and 4, respectively.

Given $d_{\mathrm{I}i}$ atoms, the super-patch training data $\boldsymbol{\mathcal{X}}_S^{i-1}$ and the small patch training data $\boldsymbol{\mathcal{X}}^{i-1}$, the IPAD $\boldsymbol{\Omega}_{\mathrm{I}i}$ is learned using ConvGOAL+ algorithm. The convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_{\mathrm{I}i}, S_i)$ will be able to preserve essential information from the input LR image.

The soft-thresholds $\boldsymbol{\lambda}_{\mathrm{I}i}$ should be set properly. As in [107], the inner product between an analysis atom $\boldsymbol{\omega}_{i,j}$ and the small patch training samples $\boldsymbol{\mathcal{X}}^{i-1}$ can be well modelled by a Laplacian distribution with diversity σ_j. The sparse representation after thresholding can be modelled as the solution to a l_1-norm regularized minimization problem with regularization parameter $\boldsymbol{\lambda}_{\mathrm{I}i}$. With a Bayesian analysis, the soft-thresholds associated with IPAD $\boldsymbol{\Omega}_{\mathrm{I}i}$ can be set being inversely proportional to the diversities [107]:

$$\boldsymbol{\lambda}_{\mathrm{I}i} = \rho_{\mathrm{I}} \left[\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \cdots, \frac{1}{\sigma_{d_{\mathrm{I}i}}} \right]^T, \tag{4.25}$$

where ρ_{I} is a scaling parameter, and the diversity σ_j of the j-th coefficient can be estimated using the obtained IPAD $\boldsymbol{\Omega}_{\mathrm{I}i}$ and the small patch training data $\boldsymbol{\mathcal{X}}^{i-1}$.

The free parameter ρ_{I} can be determined by solving a 1-dimensional search problem.

The optimization problem for ρ_{I} is therefore formulated as:

$$\rho_{\mathrm{I}} = \arg\min_{\rho\in\mathcal{D}} \left\|\boldsymbol{\mathcal{Y}} - \boldsymbol{G}\mathcal{S}_{\rho\boldsymbol{\lambda}\otimes\mathbf{1}}\left(\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_{\mathrm{I}i}, S_i)\mathrm{vec}(\boldsymbol{\mathcal{X}}_S^{i-1})\right)\right\|_F^2, \tag{4.26}$$

where $\boldsymbol{\lambda} = [1/\sigma_1, 1/\sigma_2, \cdots, 1/\sigma_{d_{\mathrm{I}i}}]^T$, $\mathbf{1}$ is an all ones vector of size $(p_{S,i} - p_i + 1)^2$, $\otimes$ is the Kronecker product, $\boldsymbol{G} = \boldsymbol{\mathcal{Y}}\boldsymbol{Z}^T(\boldsymbol{Z}\boldsymbol{Z}^T)^{-1}$ with $\boldsymbol{Z} = \mathcal{S}_{\rho\boldsymbol{\lambda}\otimes\mathbf{1}}\left(\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_{\mathrm{I}i}, S_i)\mathrm{vec}(\boldsymbol{\mathcal{X}}_S^{i-1})\right)$, and $\mathcal{D}$ is a discrete set of values.

The obtained IPAD and threshold pair $(\boldsymbol{\Omega}_{\mathrm{I}i}, \boldsymbol{\lambda}_{\mathrm{I}i})$ should be able to preserve the essential information within the input image and give no worse performance when compared to a linear convolution without non-linearities.

4.5.2 Learning CAD and Threshold Pair

The function of a Clustering Analysis Dictionary (CAD) is to perform a linear transformation to its input signal such that the responses are highly correlated to the amount of residual energy. Soft-thresholding which is used as a non-linearity sets to zero the data with relatively small responses. The signals with large residual energy will then be identified.

The number of atoms in $\boldsymbol{\Omega}_{\mathrm{C}i}$ is essential to the performance of DeepCAM. Similar to the discussions in Section 4.5.1, with $d_{\mathrm{C}i}$ atoms in $\boldsymbol{\Omega}_{\mathrm{C}i}$, the size of the convolutional analysis dictionary $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_{\mathrm{C}i}, S_i)$ will be $d_{\mathrm{C}i}(p_{S,i} - p_i + 1)^2 \times S_i$. For each super-patch region on the LR image, the number of coefficients for discriminative feature representation should not decrease over layers. That is, we would like to have more atoms in $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_{\mathrm{C}i}, S_i)$ than those in $\boldsymbol{\mathcal{H}}(\boldsymbol{\Omega}_{\mathrm{C}i-1}, S_{i-1})$. Therefore, the number of CAD atoms should meet the condition:

$$d_{\mathrm{C}i} \geq d_{\mathrm{C}i-1}\frac{(p_{S,i-1} - p_{i-1} + 1)^2}{(p_{S,i} - p_i + 1)^2}. \tag{4.27}$$

Different from the unstructured deep dictionary model, it is not straightforward to set

the dictionary sizes. Eqn. (4.24) and Eqn. (4.27) provide a guideline on how to set the number of atoms in order to generate representations that are both information preserving and discriminative.

Let us denote with $\boldsymbol{\mathcal{Y}}^i \in \mathbb{R}^{s^2 p_i^2 \times N_2}$ the corresponding HR patch training data of $\boldsymbol{\mathcal{X}}^{i-1}$. A synthesis dictionary $\boldsymbol{D}_i \in \mathbb{R}^{s^2 p_i^2 \times n_i}$ can be learned to map $\boldsymbol{\mathcal{X}}^{i-1}$ to $\boldsymbol{\mathcal{Y}}^i$ by solving:

$$\boldsymbol{D}_i = \arg\min_{\boldsymbol{D}} ||\boldsymbol{D}\boldsymbol{\mathcal{X}}^{i-1} - \boldsymbol{\mathcal{Y}}^i||_F^2. \tag{4.28}$$

It has a closed-form solution:

$$\boldsymbol{D}_i = \boldsymbol{\mathcal{Y}}^i \boldsymbol{\mathcal{X}}^{i-1^T} \left(\boldsymbol{\mathcal{X}}^{i-1} \boldsymbol{\mathcal{X}}^{i-1^T} \right)^{-1}. \tag{4.29}$$

Given $\boldsymbol{D}_i$, we define the middle resolution (MR) and the residual data as $\widehat{\boldsymbol{\mathcal{Y}}}^i = \boldsymbol{D}_i \boldsymbol{\mathcal{X}}^{i-1}$ and $\boldsymbol{\mathcal{E}}^i = \boldsymbol{\mathcal{Y}}^i - \widehat{\boldsymbol{\mathcal{Y}}}^i$, respectively. The MR data is a linear transformation of the input small patch training data. The residual data contains the information about the residual energy.

We propose to learn an analysis dictionary $\boldsymbol{\Psi}_i \in \mathbb{R}^{d_{\mathrm{C}i} \times s^2 p_i^2}$ in the ground-truth data domain. If $\boldsymbol{\Psi}_i$ is able to simultaneously sparsify the middle resolution data $\widehat{\boldsymbol{\mathcal{Y}}}^i$ and the residual data $\boldsymbol{\mathcal{E}}^i$, the atoms within the learned $\boldsymbol{\Psi}_i$ will then be able to identify the data in $\widehat{\boldsymbol{\mathcal{Y}}}^i$ with large residual energy and the i-th layer CAD is then re-parameterized as:

$$\boldsymbol{\Omega}_{\mathrm{C}i} = \boldsymbol{\Psi}_i \boldsymbol{D}_i. \tag{4.30}$$

An additional consistent constraint proposed in [107] is applied to impose the simultaneous sparsifying property. Each analysis atom $\boldsymbol{\psi}_k^T$ is enforced to be able to jointly

sparsify $\widehat{\boldsymbol{\mathcal{Y}}}^i$ and $\boldsymbol{\mathcal{E}}^i$:

$$p(\boldsymbol{\Psi}) = c\sum_{j=1}^{N_2}\sum_{k=1}^{d_{\mathrm{C}i}} \log\left(1+\nu\left(\left(\boldsymbol{\psi}_k^T\widehat{\boldsymbol{y}}_j^i\right)^2-\left(\boldsymbol{\psi}_k^T\boldsymbol{e}_j^i\right)^2\right)^2\right), \tag{4.31}$$

where $c = 1/N_2 d_{\mathrm{C}i}\log(1+\nu)$, and ν is a tunable parameter.

The objective function for learning the analysis dictionary $\boldsymbol{\Psi}_i$ can then be formulated as:

$$\boldsymbol{\Psi}_i = \arg\min_{\boldsymbol{\Psi}^T\in\Theta} f(\boldsymbol{\Psi}), \tag{4.32}$$

where $f(\boldsymbol{\Psi}) = g(\boldsymbol{\Psi})+\kappa h(\boldsymbol{\Psi})+\upsilon l(\boldsymbol{\Psi})+\mu p(\boldsymbol{\Psi})$ with κ, υ and μ being the regularization parameters. The functions $g(\cdot)$, $h(\cdot)$ and $l(\cdot)$ are those defined in Eqn. (4.15), Eqn. (4.9) and Eqn. (4.16), respectively.

The objective function in Eqn. (4.32) is optimized using ConvGOAL+ algorithm. With the learned analysis dictionary $\boldsymbol{\Psi}_i$, the i-th layer CAD is then obtained as in Eqn. (4.30).

As proposed in DeepAM [107], it is both effective and efficient to set CAD soft-thresholds being proportional to the diversity of the analysis coefficients. The CAD soft-thresholds are therefore defined as follows:

$$\boldsymbol{\lambda}_{\mathrm{C}i} = \rho_{\mathrm{C}}\left[\sigma_1,\sigma_2,\cdots,\sigma_{d_{\mathrm{C}i}}\right], \tag{4.33}$$

where ρ_{C} is a scaling parameter, and σ_j is the diversity of the Laplacian distribution for the j-th atom.

The free parameter ρ_{C} can be learned using a similar approach to the one used to solve Eqn. (**??**). As the analysis coefficients can be well modelled by Laplacian distributions, the proportion of data that has been set to zero for each pair of atom and threshold

will be the same. The optimization problem for ρ_{C} is formulated as:

$$\rho_{\mathrm{C}} = \arg\min_{\rho \in \mathcal{D}} \left\| \boldsymbol{\mathcal{Y}} - \boldsymbol{G}\mathcal{S}_{[\boldsymbol{\lambda}_I;\rho\boldsymbol{\lambda}\otimes\mathbf{1}]} \left(\boldsymbol{\mathcal{H}}([\boldsymbol{\Omega}_{\mathrm{I}i};\boldsymbol{\Omega}_{\mathrm{C}i}], S_i)\mathrm{vec}(\boldsymbol{\mathcal{X}}_S^{i-1}) \right) \right\|_F^2, \tag{4.34}$$

where $\boldsymbol{\lambda} = [\sigma_1, \sigma_2, \cdots, \sigma_{d_{\mathrm{C}i}}]^T$, $\mathbf{1}$ is a all ones vector of size $(p_{S,i} - p_i + 1)^2$, $\otimes$ is the Kronecker product, $\boldsymbol{G} = \boldsymbol{\mathcal{Y}}\boldsymbol{Z}^T(\boldsymbol{Z}\boldsymbol{Z}^T)^{-1}$ with $\boldsymbol{Z} = \mathcal{S}_{[\boldsymbol{\lambda}_I;\rho\boldsymbol{\lambda}\otimes\mathbf{1}]} \left(\boldsymbol{\mathcal{H}}([\boldsymbol{\Omega}_{\mathrm{I}i};\boldsymbol{\Omega}_{\mathrm{C}i}], S_i)\mathrm{vec}(\boldsymbol{\mathcal{X}}_S^{i-1}) \right)$, and $\mathcal{D}$ is a discrete set of values.

4.5.3 Synthesis Dictionary Learning

At the last layer, the synthesis dictionary $\boldsymbol{D} \in \mathbb{R}^{s^2 \times n_{L+1}}$ will transform the L-th layer deep convolutional representation $\boldsymbol{\mathcal{X}}^L \in \mathbb{R}^{n_{L+1} \times N_2}$ to the ground-truth training data $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{s^2 \times N_2}$. The synthesis dictionary can be learned using least squares:

$$\boldsymbol{D} = \boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{X}}^{L^T} \left(\boldsymbol{\mathcal{X}}^L \boldsymbol{\mathcal{X}}^{L^T} \right)^{-1}. \tag{4.35}$$

By convolving the learned synthesis dictionary with the L-th layer feature maps, there will result in s^2 estimated HR images which can be reshaped and combined to form the final estimated HR image.

4.5.4 DeepCAM Learning Algorithm

The overall learning algorithm for DeepCAM is summarized in **Algorithm 4**. We adopt a layer-wise learning strategy for DeepCAM. At each layer, the IPAD and the CAD are sequentially learned using ConvGOAL+ algorithm with different objective functions. The soft-thresholds are then learned using Eqn. (4.26) and Eqn. (4.34). The learned analysis dictionary and the soft-thresholds are the concatenation of the IPAD part and the CAD part. After all the analysis dictionaries and soft-thresholds

Algorithm 4: DeepCAM Learning Algorithm

Input: Training data pair $(\boldsymbol{\mathcal{X}}_S^0, \boldsymbol{\mathcal{Y}})$, the number of layers L, the number of filters, and the filter size;
for $i \leftarrow 1$ **to** L **do**
  Learning $\boldsymbol{\Omega}_{\mathrm{I}i}$ using ConvGOAL+ with training data $(\boldsymbol{\mathcal{X}}_S^{i-1}, \boldsymbol{\mathcal{X}}^{i-1})$ and objective function Eqn. (4.17);
  Learning $\boldsymbol{\Omega}_{\mathrm{C}i}$ using ConvGOAL+ with training data $(\boldsymbol{\mathcal{X}}_S^{i-1}, \boldsymbol{\mathcal{X}}^{i-1}, \boldsymbol{\mathcal{Y}})$ and objective function Eqn. (4.32);
  Learning thresholds $\boldsymbol{\lambda}_{\mathrm{I}i}$ and $\boldsymbol{\lambda}_{\mathrm{C}i}$;
  $\boldsymbol{\Omega}_i \leftarrow [\boldsymbol{\Omega}_{\mathrm{I}i}; \boldsymbol{\Omega}_{\mathrm{C}i}]$, $\boldsymbol{\lambda}_i \leftarrow [\boldsymbol{\lambda}_{\mathrm{I}i}; \boldsymbol{\lambda}_{\mathrm{C}i}]$;
  Update the super-patch training data as $\boldsymbol{\mathcal{X}}_S^i = \mathcal{S}(\boldsymbol{\Omega}_i * \boldsymbol{\mathcal{X}}_S^{i-1}, \boldsymbol{\lambda}_i)$;
  Extract the small patch training data $\boldsymbol{\mathcal{X}}^i$ from $\boldsymbol{\mathcal{X}}_S^i$;
end
Learning the synthesis dictionary $\boldsymbol{D}$ as in Eqn. (4.35).
Output: Learned DeepCAM $\left\{\{\boldsymbol{\Omega}_i, \boldsymbol{\lambda}_i\}_{i=1}^L, \boldsymbol{D}\right\}$.

have been learned, the synthesis dictionary is learned using least squares.

4.6 Simulation Results

In this section, we report the implementation details and numerical results of our proposed DeepCAM method and compare our proposed method with other single image super-resolution algorithms.

4.6.1 Implementation Details

The standard 91 training images [1] are used as the training dataset and the *Set5* [1] and the *Set14* [2] are used as the testing datasets. The color images have been converted from the RGB color space to the YCbCr color space. Image super-resolution is only performed on the luminance channel Y. The low-resolution (LR) images are generated by down-sampling the high-resolution (HR) image with the Matlab function *imresize*.

Parameters	ν	κ	υ	μ
IPAD	$10 \times n_i$	$100 \times d_{\mathrm{I}i}$	$0.01 \times d_{\mathrm{I}i}$	—
CAD	$10 \times n_i$	$0.1 \times d_{\mathrm{C}i}$	$0.01 \times d_{\mathrm{C}i}$	100

Table 4.2: Parameters setting of GOAL+ algorithm for learning the i-th layer IPAD $\mathbf{\Omega}_{\mathrm{I}i} \in \mathbb{R}^{d_{\mathrm{I}i} \times n_i}$ and CAD $\mathbf{\Omega}_{\mathrm{C}i} \in \mathbb{R}^{d_{\mathrm{C}i} \times n_i}$.

The Peak Signal-to-Noise Ratio (PSNR)[1] is used as the evaluation metric.

Table 4.2 shows the parameters setting of ConvGOAL+ algorithm for learning the i-th layer Information Preserving Analysis Dictionary (IPAD) and Clustering Analysis Dictionary (CAD). Both the IPAD and the CAD are initialized with i.i.d. Gaussian random entries. We apply batch training for ConvGOAL+ algorithm. The training data has been equally divided into $N_b = 10$ batches. During training, the ConvGOAL+ algorithm is sequentially applied to batches until the learned dictionary converges or all batches have been used for training. The spatial size of the super-patches $p_{S,i}$ used for training is set to the minimum value as indicated by Eqn. (4.23) for the purpose of minimizing the training computational complexity. The number of IPAD atoms is set to $d_{\mathrm{I}i} = d_{\min} + 1$ where $d_{\min}$ is the minimum integer defined in Eqn. (4.24). The number of CAD atoms is then set to $d_{\mathrm{C}i} = d_i - d_{\mathrm{I}i}$ with a pre-defined total number of channel d_i. For each batch, $\tau = 100$ iterations of conjugate gradient descent is performed to update the dictionary. The discrete set $\mathcal{D}$ used for searching the scaling parameter of the thresholds is set to be $\mathcal{D} = [\cdots, 10^{-2}, 2 \times 10^{-2}, \cdots, 10^{-1}, 2 \times 10^{-1}, \cdots]$.

4.6.2 Analysis of the Learned DeepCAM

In this section, we will analyze the learned DeepCAM in terms of the number of layers, learned soft-thresholds, extracted feature maps.

Table 4.3 shows the PSNR (dB) of the learned DeepCAM with different number of layers evaluated on *Set5* [2]. For DeepCAM with different number of layers, the spatial

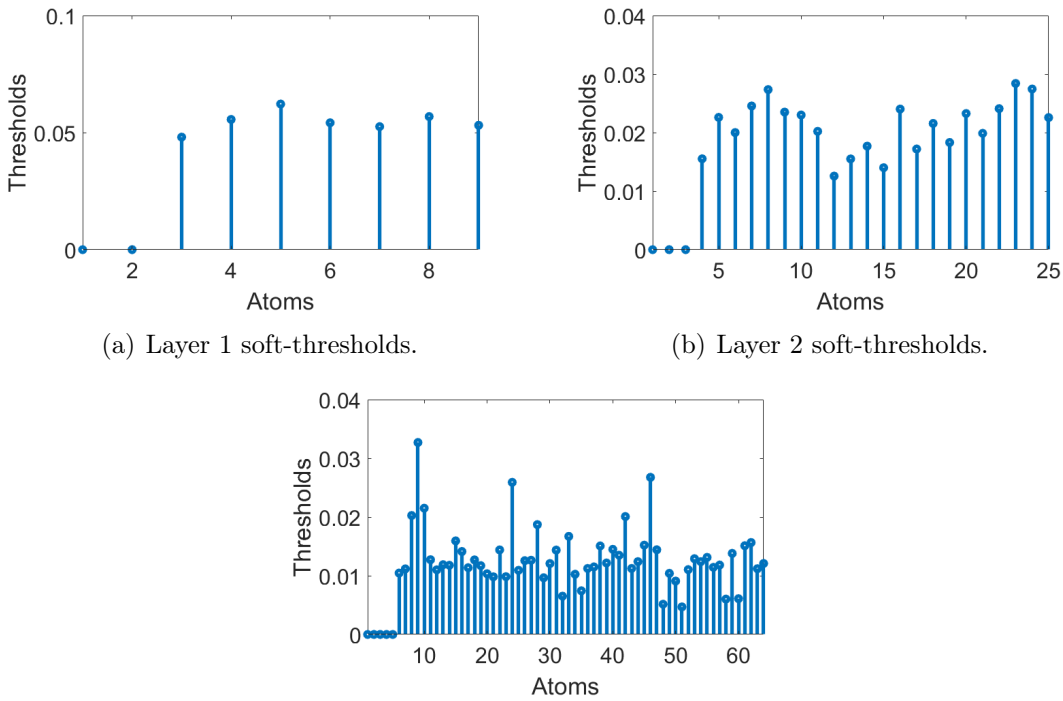
[1] PSNR$=10\log(\frac{255^2}{\sqrt{MSE}})$, where MSE is the mean squared error between the ground-truth HR image and the estimated HR image

Layer Number	1	2	3
Filter Number	[64]	[16,64]	[9,25,64]
baby	38.03	38.21	38.30
bird	39.74	40.19	40.60
butterfly	31.70	31.99	32.17
head	35.54	35.58	35.61
woman	34.76	34.83	34.84
Average	35.95	36.16	36.30

Table 4.3: PSNR (dB) of DeepCAM with different number of layers. For all DeepCAM, the spatial filter size at all layers is 3×3. The maximum number of filters at the last layer is set to 64. $[d_1, d_2]$ denotes that there are d_1 filters at the first layer, and d_2 filter at the second layer.

filter size at all layers is set to 3×3 and the maximum number of filters at the last layer is set to 64. The effective filter size for the DeepCAM with 1, 2, and 3 layers is therefore 5, 7, and 9, respectively. We can find that the DeepCAM with more layers achieves higher average PSNR. From 1 layer to 2 layers, there is around 0.2 dB improvement on the average PSNR. Especially, the PSNR of the testing image "bird" has been improved by around 0.5 dB. With 3 layers, further improvements can be observed on all testing images. The simulation results indicate that the proposed IPAD and CAD design of the DeepCAM can lead to a successful deep model and the proposed learning algorithms for both convolutional analysis dictionary and soft-thresholds is effective. The improved performance of DeepCAM with more layers can be due to two reasons. First, a deeper model has more non-linear layers and is therefore with a stronger expressive power. Second, different from the unstructured deep dictionary model, a deeper convolutional dictionary model has an increased effective filter size which helps include more information for prediction and therefore improves prediction performance.

Fig. 4.8 shows the soft-thresholds of a 3-layer DeepCAM with 9, 25 and 64 filters at layer 1, 2 and 3, respectively. We can observe that the soft-thresholds have a bimodal behaviour. That is, the thresholds corresponding to IPAD are relatively small, while the thresholds corresponding to CAD are relatively large. The small soft-thresholds



(a) Layer 1 soft-thresholds.

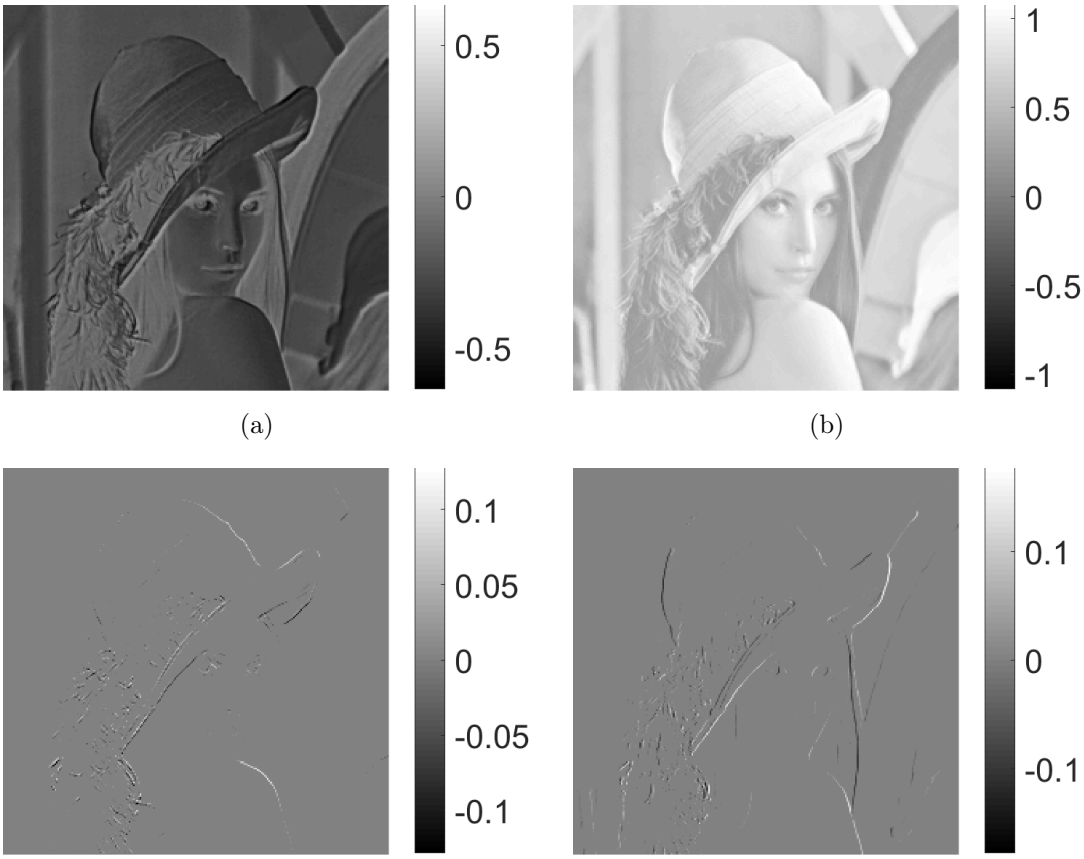
(b) Layer 2 soft-thresholds.

(c) Layer 3 soft-thresholds.

Figure 4.8: The soft-thresholds in layer 1, 2 and 3 of DeepCAM. There is a bimodal behaviour on the thresholds. The thresholds correspond to IPAD are relatively small, while the thresholds correspond to CAD are relatively large.

of IPAD help preserve information from the input image. The large soft-thresholds in CAD help generate the features that could effectively identify input image regions with large estimation error. Another observation is that the amplitude of the soft-thresholds decreases over layers. This will lead to denser representations at a deeper layer which can represent more complex signals.

Due to different learning objectives of IPAD and CAD, their resultant feature maps contains different information. Fig. 4.9 shows the first layer feature maps of a 3-layer DeepCAM. The first 2 feature maps correspond to IPAD. We can find that these two feature maps, especially, the feature map in Fig. 4.9(b) represent detailed structural information of the input LR image. The feature maps in Fig.4.9(c) - 4.9(d) correspond to CAD and have zero responses on most regions due to relatively large soft-thresholds. There contains different directional edges corresponding to regions that require non-linear estimations. A combination of these feature maps from both IPAD and CAD



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

Figure 4.9: The first layer feature maps of DeepCAM. The feature maps in 4.9(a) - 4.9(b) corresponding to IPAD and contain detailed structural information about the input LR image. The feature maps in 4.9(c) - 4.9(d) corresponding to CAD. They contains directional edges.

forms an informative and discriminative feature representation for predicting the HR image.

4.6.3 Comparison with Single Image Super-Resolution Methods

In this section, we compare our proposed DeepCAM method with the DeepAM method [107] and some existing single image super-resolution methods including bicubic interpolation, sparse coding (SC) based method [2], Anchored Neighbor Regression (ANR) method [7], Adjusted Anchored Neighborhood Regression (A+) method [8], and Super-

Method	SC [2]	ANR [7]	A+ [8]	SRCNN [14]	DeepAM	DeepCAM
Parameters	65,536	1,064,896	1,064,896	8,128	156,672	34,740

Table 4.4: Number of free parameters in different single image super-resolution methods.

Resolution Convolutional Neural Network (SRCNN) method [14].

The SC-based method [2], Anchored Neighbor Regression (ANR) method [7], Adjusted Anchored Neighborhood Regression (A+) method [8] are patch-based single image super-resolution method. SC method [2] is based on synthesis sparse representation and has a LR synthesis dictionary with 1024 atoms and a corresponding HR synthesis dictionary. The input feature is the compressed 1-st and 2-nd order derivatives of the image patch and is obtained using Principal Component Analysis (PCA). The ANR method and A+ method [7, 8] are based on clustering and assign each cluster a linear regression model. They use the same feature as in [2] while require a huge number of free parameters. The SRCNN method [14] is a convolutional neural network method with 2 convolutional layers of 64 and 32 filters. The spatial filter size is 9×9, 1×1 and 5×5, respectively.

DeepCAM used for comparison is a 3-layer DeepCAM. For the convolutional analysis dictionary, the spatial filter size is 3×3 at all layers and the filter number is 9, 25, 100 for layer 1, 2 and 3, respectively. The convolutional synthesis dictionary is with spatial filter size 5×5. Therefore, the effective filter size of DeepCAM is 11×11.

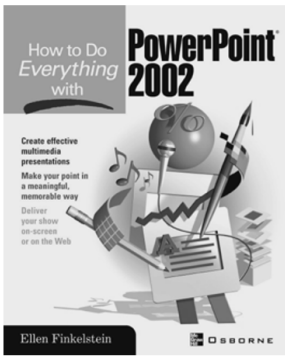
Table 4.4 shows the number of free parameters in different single image super-resolution methods. The SC method requires a relatively small number of parameters which mainly comes from two synthesis dictionaries. The ANR method [7] and the A+ method [8] have around 1 million free parameters because there are 1024 regressors with size of 36×28. The SRCNN method [14] has the least number of parameters, while there are more parameters in its improved versions which has more layers, more filters and with larger filter size. The DeepAM has more around 160,000 parameters.

Images	Bicubic	SC [2]	ANR [7]	A+ [8]	SRCNN [14]	DeepAM [107]	CNN-BP$_{80}$	CNN-BP$_{40}$	DeepCAM	DeepCAM$_{\text{bp}}$
baby	36.93	38.11	38.31	38.39	38.16	38.48	38.31	38.31	38.18	38.20
bird	36.76	39.87	40.00	41.11	40.58	41.01	41.10	41.06	40.68	40.78
butterfly	27.58	30.96	30.76	32.37	32.58	31.44	32.23	32.56	32.39	32.50
head	34.71	35.47	35.54	35.64	35.51	35.64	35.57	35.59	35.50	35.55
woman	32.31	34.59	34.68	35.44	35.07	35.20	34.84	35.16	34.93	35.25
Average	33.66	35.80	35.86	36.59	36.38	36.35	36.41	36.54	36.34	36.45

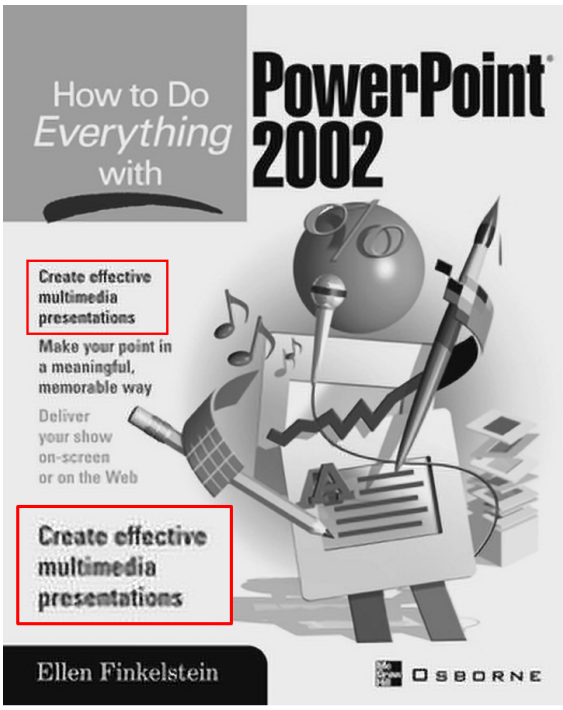
Table 4.5: PSNR (dB) of different methods evaluated on *Set5* [1].

Images	Bicubic	SC [2]	ANR [7]	A+ [8]	SRCNN [14]	DeepAM [107]	CNN-BP$_{80}$	CNN-BP$_{40}$	DeepCAM	DeepCAM$_{bp}$
baboon	24.85	25.47	25.55	25.66	25.64	25.65	25.54	25.63	25.52	25.55
barbara	27.87	28.50	28.43	28.49	28.53	28.49	27.88	28.14	28.45	28.40
bridge	26.64	27.63	27.62	27.87	27.74	27.82	27.82	27.87	27.81	27.84
c.guard	29.16	30.23	30.34	30.34	30.43	30.44	30.28	30.37	30.40	30.40
comic	25.75	27.34	27.47	27.98	28.17	27.77	27.29	27.82	27.72	27.98
face	34.69	35.45	35.52	35.63	35.57	35.62	35.54	35.55	35.48	35.51
flowers	30.13	32.04	32.06	32.80	32.95	32.45	32.65	32.82	32.57	32.78
foreman	35.55	38.41	38.31	39.45	37.43	38.89	39.31	39.24	39.12	39.23
lenna	34.52	36.06	36.17	36.45	36.46	36.46	36.35	36.39	36.16	36.25
man	29.11	30.31	30.33	30.74	30.78	30.57	30.68	30.71	30.58	30.70
monarch	32.68	35.50	35.46	36.77	37.11	36.06	36.62	36.84	36.56	36.85
pepper	35.02	36.64	36.51	37.08	36.89	36.87	36.99	36.99	36.86	36.97
ppt3	26.58	29.00	28.67	29.79	30.31	29.13	28.02	29.18	29.61	29.67
zebra	30.41	33.05	32.91	33.45	33.14	33.34	32.45	33.04	33.33	33.28
Average	30.21	31.83	31.81	32.32	32.22	32.11	31.96	32.18	32.16	32.24

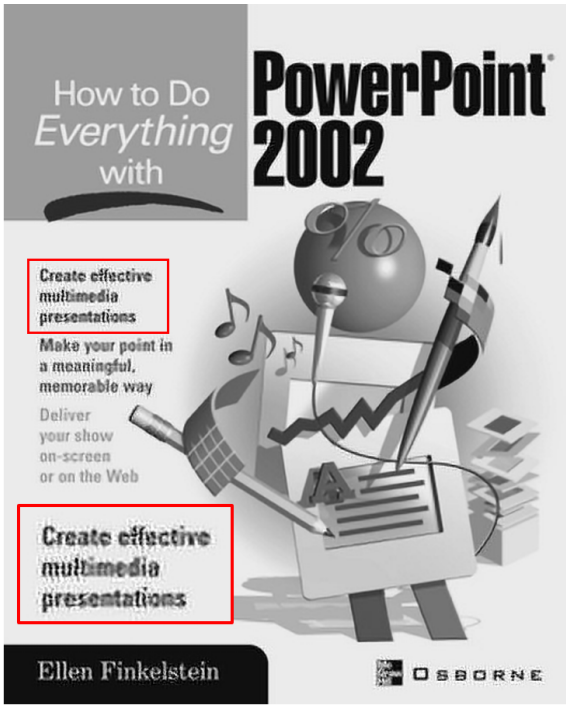
Table 4.6: PSNR (dB) of different methods evaluated on *Set14* [2].



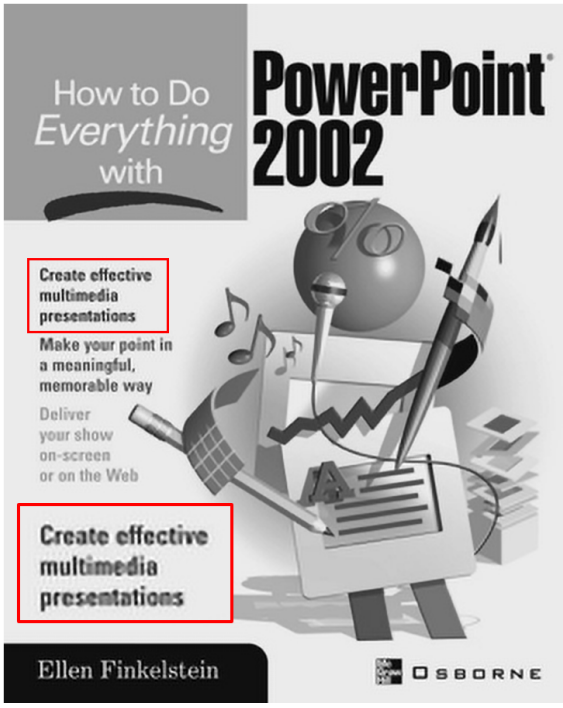
(a) LR image.



(b) CNN-BP$_{40}$ (PSNR = 29.18 dB).

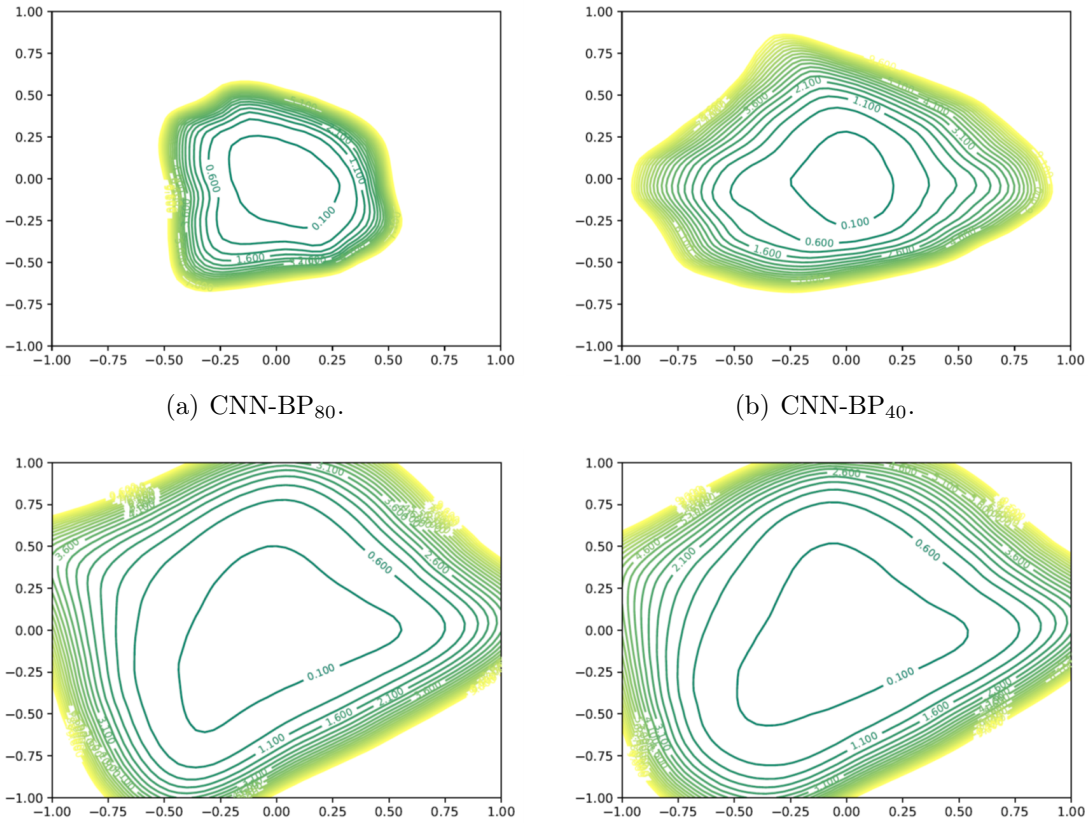


(c) CNN-BP$_{80}$ (PSNR = 28.02 dB).



(d) DeepCAM (PSNR = 29.65 dB).

Figure 4.10: Examples of reconstructed HR images by different methods. DeepCAM achieves the better result than the backpropagation trained CNN. A region with characters on the reconstructed image have been marked using red rectangle and zoomed in.



(a) CNN-BP$_{80}$.

(b) CNN-BP$_{40}$.

(c) DeepCAM.

(d) DeepCAM$_{\text{bp}}$.

Figure 4.11: Visualize the 2D surface of minima obtained by different methods. The sharpness of minimizers correlates well with generalization error. A wider, and flatter minimizer usually has better generalization ability. The minimizers of DeepCAM and DeepCAM$_{\text{bp}}$ are flatter and wider than that of CNN-BP$_{40}$ and CNN-BP$_{80}$.

This is because the dictionaries are not structured and there are 3 layers of analysis dictionaries. The proposed DeepCAM has only around 35,000 free parameters since each structured convolutional dictionary shares a small number of free parameters though it has a huge size.

We denote CNN-BP$_{DS}$ as the deep neural network with the same structure as DeepCAM and trained using backpropagation with learning rate decay step DS and total $5 \times DS$ epochs for training. The implementation of DNNs is based on Pytorch with Adam optimizer [96], batch size 1, initial learning rate 1×10^{-3}, and decay rate 0.1. The training data has been arranged into 36×36 and 72×72 patch pairs.

Table 4.5 shows the evaluation results of different methods on *Set5*. DeepCAM achieves

around 0.5 dB higher PSNR than SC method and ANR method, and has similar performance as SRCNN method and DeepAM, while has around 0.2 dB lower PSNR than that of A+ method. The parameters setting of CNN-BP$_{40}$ has been tuned to achieve the best performance on *Set5*. CNN-BP$_{80}$ and CNN-BP$_{40}$ have been used for comparison and achieves around 0.1 dB and 0.2 dB higher PSNR than DeepCAM. DeepCAM$_{\text{bp}}$ represents the backpropagation fine-tuned DeepCAM with with Adam optimizer [96], batch size 1, initial learning rate 1×10^{-4}, and with total 20 epochs. DeepCAM$_{\text{bp}}$ achieves an improved performance than DeepCAM. It average PSNR is comparable to that of A+ method and CNN-BP.

Table 4.6 shows the evaluation results of different single image super-resolution methods on *Set14*. Similar results can be observed as in Table 4.5. The average PSNR of DeepCAM is around 0.3 dB higher than that of SC method and ANR method, while it is around 0.15 dB lower than that of A+ method. DeepCAM achieves similar performance as SRCNN method, DeepAM and CNN-BP$_{40}$. DeepCAM$_{\text{bp}}$ achieves improved performance as DeepCAM.

It is interesting to note that DeepCAM significantly outperforms CNN-BP$_{40}$ and CNN-BP$_{80}$ on "ppt3" and "zebra" which contain sharp edges with small scales. Especially, DeepCAM achieves around 0.4 dB and 1.6 dB higher PSNR on "ppt3" than CNN-BP$_{40}$ and CNN-BP$_{80}$, respectively. Figure 4.10 shows the input LR image and the reconstructed HR images of the testing image "ppt3" using CNN-BP$_{80}$ and DeepCAM method. We can find that CNN-BP$_{80}$ cannot well reconstruct the characters. A possible reason is that CNN-BP has a weaker generalization ability on the unseen testing data.

Li *et al.* [108] proposes a visualization method to visualize the loss surface landscape around the minimizer of a deep model. The sharpness of the loss surface landscape of a minimizer is well correlated to the generalization ability. That is, a wider and flatter minimizer has better generalization ability. From Figure 4.11, we can find that CNN-BP$_{80}$ has the sharpest and the most narrow 2D loss surface, CNN-BP$_{40}$ has a

wider and flatter one while is still less wider than that of DeepCAM. We can also find that performing backpropagation fine-tuning on DeepCAM does not change much on the 2D loss surface. The visualization in Figure 4.11 correlates well with the simulation results in Table 4.6. The reason that DeepCAM possesses a stronger generalization ability could due to our information preserving and clustering design which encodes a general rule for image super-resolution.

4.7 Summary

In this chapter, we proposed a convolutional analysis dictionary learning algorithm by exploiting the properties of the Toeplitz structure within the convolution matrices. The proposed algorithm can impose the global rank property on learned convolutional analysis dictionaries while performing learning on the low-dimensional signals. We then proposed a Deep Convolutional Analysis Dictionary Model (DeepCAM) framework which consists of multiple layers of convolutional analysis dictionary and soft-threshold pairs and a single layer of convolutional synthesis dictionary. Similar to the DeepAM, the convolutional analysis dictionaries are designed to be made of an information preserving analysis dictionary (IPAD) and a clustering analysis dictionary (CAD). The IPAD preserves the information from the input image, while the CAD generates discriminative feature maps for image super-resolution. Simulation results show that our proposed DeepCAM achieves comparable performance with other existing single image super-resolution methods.

Chapter 5

Photo Realistic Image Completion via Dense Correspondences

5.1 Introduction

In this chapter, we propose to tackle the Internet-based image completion problem using a robustly estimated dense correspondence. The advantage of utilizing dense correspondence for image completion is two-fold. First, dense correspondence gives us a complete landscape of the correspondence between images, which captures some local deformations that might be missed by sparse correspondence. With the dense correspondence, the Internet-based image completion problem becomes a surface fitting problem with a segment of the original surface missing. Therefore, our objective becomes finding a surface which is consistent with the acquired correspondence outside the "hole" region under a smoothness constraint. Second, dense correspondence provides a rich set of color correspondence between the input image and the retrieved exemplar image. This helps remove color discrepancy on the completed region. This is in contrast with the color correspondence provided by sparse correspondence which is usually not sufficient for robust color correction.

A hierarchical framework is proposed to progressively achieve image completion based on dense correspondence. The hierarchical structured PatchMatch imposes smoothness constraint on the estimated dense correspondence. However, the obtained dense correspondence is usually noisy and contains outliers. We propose to use an Expectation-Maximization (EM) based approach with kernel ridge regression to jointly denoise the obtained correspondence and interpolate the correspondence in the "hole" region at each hierarchical level. Within the hierarchical framework, the EM model parameters of the current level are used to initialize the parameters of the next level and this leads to a faster convergence. At the final level, a completed image is obtained by transferring image content with respect to the interpolated dense correspondence and performing color correction to remove the possible color differences between two images.

The rest of the chapter is organized as follows. Section 5.2 gives an overview of our proposed image completion method. Section 5.3 introduces the proposed image completion using dense correspondence framework, and Section 5.4 describes color correction based on estimated dense correspondence. Section 5.5 presents the results of our extensive experimental work and Section 5.6 draws conclusions.

5.2 Overview

Our algorithm makes the same assumptions as other Internet-based image completion algorithms [4, 6, 41]. Specifically, we assume that there are many images on Internet similar to the image that needs to be completed and some similar exemplar images are already available in that we rely on existing searching engines to find exemplar images. This is particularly true for images of famous landmarks. Moreover, we assume that a region-of-interest (ROI) is given by the user indicating a region that requires completion.

The overview of the proposed image completion method is illustrated in Fig. 5.1. The

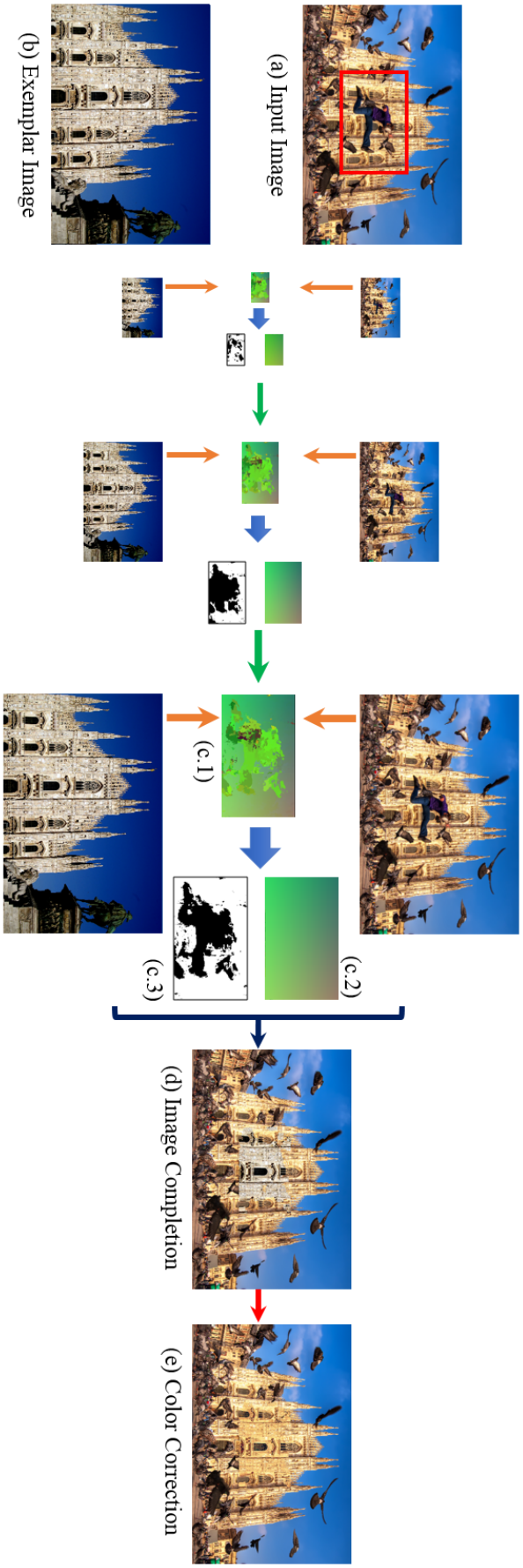


Figure 5.1: **Overview of the proposed image completion algorithm.** (a) The input image with a region-of-interest. (b) A retrieved exemplar image. (c) Our proposed framework progressively estimates dense correspondence (e.g. (c.1)) from coarse-to-fine pyramid levels using a hierarchical PatchMatch and interpolates a smooth dense correspondence (e.g. (c.2)) over the occluded region based on the estimated inlier correspondences (e.g. (c.3)). (d) Image completion based on the interpolated dense correspondence. (e) Color correction based on the dense correspondence on the estimated inlier region. (Please note that the dense correspondence has four dimensions at each location, the color representation only shows the first three dimensions.)

required ROI is a rectangle region rather than a detailed contour. However, we are not trying to replace the whole ROI but only the occluded part. Gaussian image pyramids are built for both the input image and the retrieved exemplar image. We then progressively estimate dense correspondence from coarse-to-fine pyramid levels. An Expectation-Maximization (EM) algorithm jointly performs inliers/outliers estimation and dense correspondence interpolation (Fig. 5.1 (c)) to obtain a smooth dense correspondence over the "hole" region. The current level dense correspondence and model parameters of the EM algorithm will be passed to the next pyramid level. The "hole" can then be filled using the image content from the exemplar image. More precisely, the interpolated dense correspondence indicates the correct pixel location on the exemplar image of a pixel on the input image. By transferring the corresponding pixel values from the exemplar image, the undesired image content in the input image can be replaced (Fig. 5.1 (d)). However, it is very likely that the color between the input image and the retrieved exemplar image is incompatible (for example, in Fig. 5.1 (a) and (b)). Based on the established dense correspondence, our color correction algorithm is applied to fit a smooth B-spline color transfer function for each color channel in RGB color space. It corrects the color differences and improves visual quality as shown in Fig. 5.1 (e).

5.3 Image Completion using Dense correspondence

Let us denote the input image with $\mathcal{I}^1$ and the exemplar image with $\mathcal{E}^1$. We generate from $\mathcal{I}^1$ an image pyramid with decreasing resolutions $\{\mathcal{I}^k\}_{k=1}^K$, scaled down by a factor 2^{k-1}. Similarly, an image pyramid $\{\mathcal{E}^k\}_{k=1}^K$ is generated from $\mathcal{E}^1$.

The kth level nearest neighbor field (NNF) $\mathcal{F}^k$ relates every image patch on $\mathcal{I}^k$ with its nearest neighbor (NN) patch on $\mathcal{E}^k$. As the same scene on $\mathcal{I}^k$ and $\mathcal{E}^k$ may differ in viewpoint, we assume there are $D = 4$ degrees of freedom for the NN patch, i.e.

position (u, v), scale s, and orientation θ. Thus, $\mathcal{F}^k$ has the same size as $\mathcal{I}^k$ and has 4 matching parameters at each location. We denote with $\mathcal{I}_r^k(\boldsymbol{p})$ the $(2r+1) \times (2r+1)$ patch centered at $\boldsymbol{p} = (x, y)$ on $\mathcal{I}^k$, with $\mathcal{F}^k(\boldsymbol{p})$ the matching parameter (u, v, s, θ) of the NN patch for patch $\mathcal{I}_r^k(\boldsymbol{p})$, and with $\mathcal{E}_r^k(\mathcal{F}^k(\boldsymbol{p}))$ the NN patch on $\mathcal{E}^k$ at location (u, v) with patch radius $s \times r$, and orientation θ.

5.3.1 Basic PatchMatch

In this section, we briefly review PatchMatch and the variations we have introduced to make it more suitable to our problem. For a detailed description of the classical PatchMatch, please refer to [25, 26].

There are three key steps in PatchMatch, i.e. random initialization, neighboring propagation, and random search. As natural images are highly structured, good matching parameters in a randomly initialized NNF can be propagated to its spatial neighbors with minor adjustment. For example, a patch $\mathcal{I}_r^1(p)$ can update its matching parameter $\mathcal{F}^1(\boldsymbol{p})$ by using those of its spatial neighbors Ψ_p:

$$\mathcal{F}^1(\boldsymbol{p}) = \arg\min_{\mathcal{F}^1(p_i)} \left\{ S\left(\mathcal{F}^1(p_i)\right) | p_i \in p \cup \Psi_p \right\}, \tag{5.1}$$

where $S(\mathcal{F}^1(p_i))$ is the matching cost between $\mathcal{I}_r^1(p)$ and $\mathcal{E}_r^1(\mathcal{F}^1(p_i) + \boldsymbol{\omega})$ with $\boldsymbol{\omega}$ being the affine adjustment.

Propagation is specified by how the spatial neighbors Ψ_p is defined. There are two kinds of propagation in PatchMatch [25], i.e. type 1: $\Psi_p = \{\boldsymbol{p} - \mathbf{1}_h, \boldsymbol{p} - \mathbf{1}_v\}$ and type 2: $\Psi_p = \{\boldsymbol{p} + \mathbf{1}_h, \boldsymbol{p} + \mathbf{1}_v\}$, where $\mathbf{1}_h = (1, 0)$ and $\mathbf{1}_v = (0, 1)$. Bailer *et al.* [109] proposed that propagation should be performed in two more directions (i.e. type 3: $\Psi_p = \{\boldsymbol{p} - \mathbf{1}_h, \boldsymbol{p} + \mathbf{1}_v\}$ and type 4: $\Psi_p = \{\boldsymbol{p} + \mathbf{1}_h, \boldsymbol{p} - \mathbf{1}_v\}$) so that good matching parameters can be propagated to any position on the input image. This is the strategy

adopted in this chapter.

After propagation has been performed at each position, random search is applied to avoid matching parameters being trapped in local minima. For a location $\boldsymbol{p}$, a small number of matching parameters will be randomly sampled near $\mathcal{F}^1(\boldsymbol{p})$ in the parameter space within a predefined maximum random search range. The sampling radius shrinks by 2 until the range is smaller than 1. If one of them can provide lower matching cost, $\mathcal{F}^1(\boldsymbol{p})$ will be updated with it. The problem of the random search in classical PatchMatch is that random search is intensively applied in every location with the same sampling radius. This introduces a huge number of ineffective computation, since a non-adaptive sampling radius will generate many irrelevant matching parameters with high matching costs. In Section 5.3.4, we propose an adaptive random search which adaptively selects candidate matching parameters based on their estimated reliability.

5.3.2 Feature Representation and Distance Metric

In optical flow estimation, two images are temporally adjacent and have no obvious color difference. However, in our setting the exemplar images are normally different from the input image in viewpoint, illumination and color condition.

The commonly used patch feature descriptors in optical flow include RGB, intensity invariant color representation, gradients, SIFT descriptor [44], and Census transform [110]. Matching patches using RGB will result in very noisy NNF as RGB is vulnerable to the aforementioned image variations. SIFT descriptor can provide better performance, however, this is a computationally expensive solution as a huge number of patches need to be evaluated in PatchMatch. Census transform is a binary descriptor and is robust to illumination changes. The advantage of binary descriptors [111–113] is that a patch is represented by a binary string and matching is easy to compute, since Hamming distance instead of l_2 distance is applied for feature distance evaluation.

We utilize BRIEF descriptor [112] as feature representation. It is one of the most widespread binary descriptors and has comparable performance with floating-point descriptors [114]. Census transform [110] is a special case of BRIEF. According to BRIEF descriptor [112], a small number of binary tests with randomly generated test locations can yield good discrimination power. A binary test τ for a patch is defined as:

$$\tau(c, \boldsymbol{p}, \boldsymbol{q}) = \begin{cases} 1 & \text{if } c(\boldsymbol{p}) < c(\boldsymbol{q}), \\ 0 & \text{otherwise}, \end{cases} \tag{5.2}$$

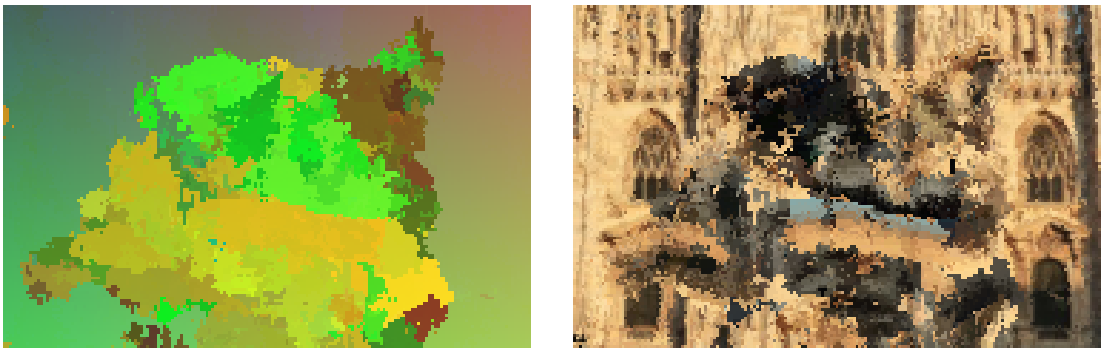
where c represents one of the feature channels, $\boldsymbol{p}$ and $\boldsymbol{q}$ are two positions on the patch.

In order to gain color invariant property, RGB color image is converted to CIE La*b* color space where L is the illuminance channel, and a* and b* are chrominance channels. Two gradient channels $\nabla_{\mathrm{x}}\mathrm{L}$ and $\nabla_{\mathrm{y}}\mathrm{L}$ have also been included to account for edges. Therefore the final feature channel is $\mathcal{C} = (\mathrm{L}, \mathrm{a}^*, \mathrm{b}^*, \nabla_{\mathrm{x}}\mathrm{L}, \nabla_{\mathrm{y}}\mathrm{L})$.

There are n_b binary tests that are generated $\{\tau(c_i, \boldsymbol{p}_i, \boldsymbol{q}_i)\}_{i=1}^{n_b}$. For each binary test $\tau(c, \boldsymbol{p}, \boldsymbol{q})$, c is randomly selected from feature channel set $\mathcal{C}$, and $\boldsymbol{p}$ and $\boldsymbol{q}$ are randomly sampled from discrete locations on a $(2r+1) \times (2r+1)$ patch. After the n_b binary tests are determined, all the patches will use these tests to construct the BRIEF descriptor. The BRIEF descriptor of a patch can be expressed as a binary string being the binarization of $\sum_{i=1}^{n_\mathrm{b}} 2^{i-1}\tau(c_i, \boldsymbol{p}_i, \boldsymbol{q}_i)$. Thus, the distance metric $S(\cdot)$ in Eqn. (5.1) measures the Hamming distance between the BRIEF descriptors of two patches.

5.3.3 Nearest Neighbor Field Interpolation

NNF interpolation is necessary to produce a smooth and accurate NNF over the occluded region. With the basic PatchMatch using BRIEF descriptor, the acquired NNF



(a) Estimated NNF.

(b) Reconstructed image based on estimated NNF.

Figure 5.2: (a) An example of the estimated nearest neighbour field. (b) The reconstructed image based on the estimated NNF. The estimated NNF is noisy and with a large region with occlusions. The quality of the NNF can be reflected in the reconstructed image.

is still relatively noisy and with a large outlier region in the occluded part (see Fig. 5.2 for an example). Only with a well refined NNF, the corresponding pixel values on the exemplar image can be faithfully transferred to the input image.

We apply an Expectation-Maximization (EM) approach similar to [115–117] to jointly estimate inliers/outliers in the observed NNF and interpolate NNF within ROI using kernel ridge regression. There are two reasons for that. First, traditional approaches, such as affine transform, and thin-plate spline model, are vulnerable to noise and outliers. The EM algorithm can be used to identify the noisy data and outliers from the observed data. In turn, a reliable NNF interpolation model can be constructed using only the inlier data points. Second, the kernel ridge regression can flexibly adapt to the variations in NNF. In this way, the flexibility gained through dense correspondence is preserved.

Likelihood Formulation

We assume that there are N matched pixel-wise correspondences $\{(\boldsymbol{p}_i, \mathcal{F}^k(\boldsymbol{p}_i))\}_{i=1}^N$ within the ROI of pyramid level k. To simplify notation, we denote $(\boldsymbol{x}_i, \boldsymbol{y}_i) = (\boldsymbol{p}_i, \mathcal{F}^k(\boldsymbol{p}_i))$

for $i = 1, 2, ..., N$, with $\mathbf{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_N)^{\mathrm{T}} \in \mathbb{R}^{N\times 2}$ and $\mathbf{Y} = (\boldsymbol{y}_1, ..., \boldsymbol{y}_N)^{\mathrm{T}} \in \mathbb{R}^{N\times D}$.

The NNF generated by PatchMatch is assumed to be a mixture of Gaussian distributed inliers and uniformly distributed outliers. Moreover the components of the NNF are assumed to be independent. An indicator $z_i \in \{0, 1\}$ is defined for each data pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ to indicate it being an inlier ($z_i = 1$) or an outlier ($z_i = 0$). The fitting error of inliers are assumed to follow a Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with zero mean and variance $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times D}$. The outliers are the data pairs which cannot be well described by NNF interpolation function $\boldsymbol{f}$. We assume the outliers are uniformly distributed among the parameter space. Under the above assumptions, the likelihood function is given as follows:

$$p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{N} \sum_{z_i} p(\boldsymbol{y}_i, z_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \prod_{i=1}^{N} \left(\gamma \frac{\exp(-d_i)}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}} + \frac{1-\gamma}{V} \right),$$

where model parameter $\boldsymbol{\theta} = \{\boldsymbol{f}, \gamma, \boldsymbol{\Sigma}\}$, $d_i = \frac{1}{2}(\boldsymbol{y}_i - \boldsymbol{f}(\boldsymbol{x}_i))^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{y}_i - \boldsymbol{f}(\boldsymbol{x}_i))$, γ represents the percentage of inliers, $\det(\cdot)$ is the determinant of a matrix, and V is the volume of the NNF parameter space.

The underlying interpolation function should be smooth. Let us assume a smooth prior on the NNF interpolation function as $p(\boldsymbol{f}) \propto \exp\left(-\frac{\lambda}{2}\phi(\boldsymbol{f})\right)$ where $\phi(\boldsymbol{f})$ is a smoothness function, and λ is a regularization parameter. With the likelihood defined in Eqn. (5.3), the smooth prior on $\boldsymbol{f}$, and assuming a uniform prior on parameter γ and $\boldsymbol{\Sigma}$, the posterior distribution of the model parameter can be estimated via Bayes rule as $p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta})p(\boldsymbol{f})$. The optimal model parameter $\boldsymbol{\theta}^*$ is then estimated from a Maximum A Posteriori (MAP) of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta})p(\boldsymbol{f}). \tag{5.3}$$

EM Algorithm

Let us define $p_i = p(z_i = 1|\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{\theta})$ and $P = \sum_{i=1}^{N} p_i$. Replacing Eqn. (5.3) into Eqn. (5.3), taking a negative logarithm of it and removing the terms independent of $\boldsymbol{\theta}$ yields the negative log-likelihood function:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{N} p_i d_i + \frac{DP}{2} \ln \det(\boldsymbol{\Sigma}) - P \ln \gamma - (N - P) \ln(1 - \gamma) + \frac{\lambda}{2} \phi(\boldsymbol{f}).$$

We apply an EM algorithm to iteratively estimate model parameter $\boldsymbol{\theta} = \{\boldsymbol{f}, \gamma, \boldsymbol{\Sigma}\}$. At iteration $t + 1$, the EM algorithm iterates between an Expectation step (E-step), which computes the posteriori probability of the latent variable $p(z_i|\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{\theta}_t)$ using the parameter $\boldsymbol{\theta}_t = \{\boldsymbol{f}_t, \gamma_t, \boldsymbol{\Sigma}_t\}$ from previous iteration, and a Maximization step (M-step), which estimates a new parameter $\boldsymbol{\theta}_{t+1}$ that minimizes the negative log-likelihood $Q(\boldsymbol{\theta})$.

During the E-step, the posteriori probability of the latent variable z_i can be estimated using Bayes rule with the model parameter $\boldsymbol{\theta}_t$:

$$p_i = \frac{\gamma_t \exp\left(-d_{i,t}\right)}{\gamma_t \exp\left(-d_{i,t}\right) + \frac{1-\gamma_t}{V} \sqrt{\det(2\pi \boldsymbol{\Sigma}_t)}}, \tag{5.4}$$

where $d_{i,t} = \frac{1}{2} \left(\boldsymbol{y}_i - \boldsymbol{f}_t(\boldsymbol{x}_i)\right)^{\mathrm{T}} \boldsymbol{\Sigma}_t^{-1} \left(\boldsymbol{y}_i - \boldsymbol{f}_t(\boldsymbol{x}_i)\right)$. In the M-step, model parameter is updated so as to minimize the negative log-likelihood. To exclude the influence of the outliers, we binarize the posteriori probability such that $p_i = 1$ if $p_i > \varphi$ ($\varphi = 0.5$), $p_i = 0$ otherwise. We take the derivative of Eqn. (5.4) with respect to γ and $\boldsymbol{\Sigma}$, respectively, and set them to zero. The updated parameters are then given by:

$$\begin{cases} \gamma_{t+1} = \frac{\mathrm{tr}(\mathbf{P})}{N}, \\ \boldsymbol{\Sigma}_{t+1}(i,i) = \frac{(\mathbf{Y}(:,i) - \mathbf{Z}(:,i))^{\mathrm{T}} \mathbf{P} (\mathbf{Y}(:,i) - \mathbf{Z}(:,i))}{\mathrm{tr}(\mathbf{P})}, \end{cases} \tag{5.5}$$

where $\mathbf{P} = \text{diag}(p_1, ..., p_N)$ is a $N \times N$ diagonal matrix with diagonal values specified by $p_1, ..., p_N$, $\mathbf{Y}(:, i)$ is the ith column of $\mathbf{Y}$, $\text{tr}(\cdot)$ returns the trace of a matrix, and $\mathbf{Z} = (\boldsymbol{f}_t(\boldsymbol{x}_1), ..., \boldsymbol{f}_t(\boldsymbol{x}_N))^{\mathrm{T}} \in \mathbb{R}^{N \times D}$ are the estimated NNF using $\boldsymbol{f}_t$.

We group the terms in Eqn. (5.4) related to the NNF interpolation function $\boldsymbol{f}$ and define an energy function $E(\boldsymbol{f})$ function as:

$$E(\boldsymbol{f}) = \frac{1}{2}\sum_{i=1}^{N} p_i \left(\boldsymbol{y}_i - \boldsymbol{f}(\boldsymbol{x}_i)\right)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{y}_i - \boldsymbol{f}(\boldsymbol{x}_i)\right) + \frac{\lambda}{2}\phi(\boldsymbol{f}). \tag{5.6}$$

Eqn. (5.6) is a special form of Tikhonov regularization. The interpolation function $\boldsymbol{f}_{t+1}$ is updated by minimizing the energy function $E(\boldsymbol{f})$ with respect to $\boldsymbol{f}$. Let the smoothness function be defined as $\phi(\boldsymbol{f}) = ||\boldsymbol{f}||^2_{\mathcal{H}}$ where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS). The reproducing kernel is selected as Gaussian. According to the representer theorem [118], the NNF interpolation function $\boldsymbol{f}$ is in the form of weighted sum of kernel products:

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{n=1}^{N} k(\boldsymbol{x}, \boldsymbol{x}_n)\boldsymbol{w}_n, \tag{5.7}$$

where $k(\boldsymbol{a}, \boldsymbol{b}) = \exp\left(-\frac{||\boldsymbol{a}-\boldsymbol{b}||^2}{\beta}\right)$ is a Gaussian reproducing kernel with filter range defined by β, and $\boldsymbol{w}_n \in \mathbb{R}^D$ is the weight associated with $k(\cdot, \boldsymbol{x}_n)$.

The norm in RKHS $\mathcal{H}$ can be expressed as $||\boldsymbol{f}||^2_{\mathcal{H}} = \widetilde{\mathbf{W}}^{\mathrm{T}}\mathbf{K}\widetilde{\mathbf{W}}$ with $\widetilde{\mathbf{W}} = \left(\boldsymbol{w}_1^{\mathrm{T}}, ..., \boldsymbol{w}_N^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{ND \times 1}$ being the coefficient column vector and $\mathbf{K} \in \mathbb{R}^{ND \times ND}$ being a $N \times N$ block matrix where the $(i, j)^{th}$ block has size $D \times D$ and its entries are with value $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. By replacing $\boldsymbol{f}$ in Eqn. (5.6) with the form in Eqn. (5.7), the energy function can be expressed in matrix form as follows:

$$E(\boldsymbol{f}) = \frac{1}{2}\left(\widetilde{\mathbf{Y}} - \mathbf{K}\widetilde{\mathbf{W}}\right)^{\mathrm{T}} \widetilde{\mathbf{P}} \left(\widetilde{\mathbf{Y}} - \mathbf{K}\widetilde{\mathbf{W}}\right) + \frac{\lambda}{2}\widetilde{\mathbf{W}}^{\mathrm{T}}\mathbf{K}\widetilde{\mathbf{W}}, \tag{5.8}$$

where $\widetilde{\mathbf{Y}} = \left(\boldsymbol{y}_1^{\mathrm{T}}, ..., \boldsymbol{y}_N^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{ND \times 1}$ is a column vector, $\widetilde{\mathbf{P}} = \mathbf{P} \otimes \boldsymbol{\Sigma}^{-1}$ with $\otimes$ being

Kronecker product.

Taking the derivative of Eqn. (5.8) with respect to $\widetilde{\mathbf{W}}$ and setting it to zero leads to the following expression for the coefficient matrix $\widetilde{\mathbf{W}}$:

$$\widetilde{\mathbf{W}} = (\mathbf{K} + \lambda \widetilde{\mathbf{P}}^{-1})^{-1} \widetilde{\mathbf{Y}}. \tag{5.9}$$

Though a closed form solution exists for $\widetilde{\mathbf{W}}$, it is too expensive to compute the inverse of $(\mathbf{K} + \lambda \widetilde{\mathbf{P}}^{-1})$ which can be decomposed into D matrix inverses of size $N \times N$. In order to reduce the computational complexity, a fast approximation method [116, 119] suggests that we can use a subset of the observed data as control points $\{\widetilde{\boldsymbol{x}}_m\}_{m=1}^{M}$ with $M \ll N$ and this reduces the size of the matrix to $M \times M$. With this fast approximation, the computational complexity is reduced from $O(N^3)$ to $O(M^3)$. The NNF interpolation function is then only represented by M kernel products:

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{m=1}^{M} k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}_m) \boldsymbol{w}_m. \tag{5.10}$$

By using M control points $\{\widetilde{\boldsymbol{x}}_m\}_{m=1}^{M}$, the coefficient matrix $\widetilde{\mathbf{W}} \in \mathbb{R}^{MD \times 1}$ are determined using the closed form expression:

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{U}}^{\mathrm{T}} \widetilde{\mathbf{P}}_{\mathbf{I}} \widetilde{\mathbf{U}} + \lambda \widetilde{\mathbf{Q}})^{-1} \widetilde{\mathbf{U}}^{\mathrm{T}} \widetilde{\mathbf{P}}_{\mathbf{I}} \widetilde{\mathbf{Y}}, \tag{5.11}$$

where $\widetilde{\mathbf{U}} \in \mathbb{R}^{ND \times MD}$ is an $N \times M$ block matrix with the $(i,j)^{th}$ block being a $D \times D$ matrix whose entries are with value $k(\boldsymbol{x}_i, \widetilde{\boldsymbol{x}}_j)$, $\widetilde{\mathbf{P}}_{\mathbf{I}} = \mathbf{P} \otimes \mathbf{I}_D$ ($\mathbf{I}_D$ is a $D \times D$ identity matrix), and $\widetilde{\mathbf{Q}} = \mathbf{Q} \otimes \boldsymbol{\Sigma}$ with $\mathbf{Q} \in \mathbb{R}^{M \times M}$ being the intra Gram matrix and $\mathbf{Q}(i,j) = k(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j)$.

The NNF interpolation function $\boldsymbol{f}$ is thus defined by the selected M control points $\{\widetilde{\boldsymbol{x}}_m\}_{m=1}^{M}$ as well as the coefficient matrix $\widetilde{\mathbf{W}}$. If the control points are fixed through

the EM algorithm, the NNF interpolation function $\boldsymbol{f}$ is solely determined by $\widetilde{\mathbf{W}}$. The interpolation function $\boldsymbol{f}_{t+1}$ at iteration $t+1$ is thus obtained through Eqn. (5.11) by using the covariance matrix $\boldsymbol{\Sigma}_{t+1}$ from Eqn. (5.5) and the updated probability matrix $\mathbf{P}$. One may argue that it is only necessary to find the interpolation function for position parameters (u, v). This can save half computation. However, the other two parameters (s, θ) provide important information for posteriori probability estimation and a well estimated probability matrix $\mathbf{P}$ leads to a faster convergence of the EM algorithm.

The stopping criterion of the EM algorithm is activated when the negative log-likelihood converges or the maximum iteration $\mathcal{T}_{\mathrm{EM}}$ has been reached.

5.3.4 Hierarchical PatchMatch with NNF Interpolation

We propose to progressively estimate a smooth NNF within the ROI using a hierarchical PatchMatch and interpolate an accurate NNF over the occluded region. The flow diagram of the hierarchical PatchMatch is shown in Fig. 5.3. The two main reasons for using the hierarchical framework are as follows:

- First, the basic PatchMatch generally produces noisy NNF as its objective is to find nearest neighbor patches rather than a smooth NNF. The coarse-to-fine scheme can impose smoothness constraint on NNF. With a fixed patch size, a patch in a higher pyramid level has relatively larger spatial range and imposes stronger spatial smoothness constraints. Proceeding to lower pyramid levels, the NNF is refined with a relatively smaller patch size and has higher matching accuracy. The lower level NNF is initialized by up-sampling previous level NNF. Good matching parameters can be passed through the hierarchy, while erroneous matching parameters are unlikely to be retained on NNF at different pyramid levels.

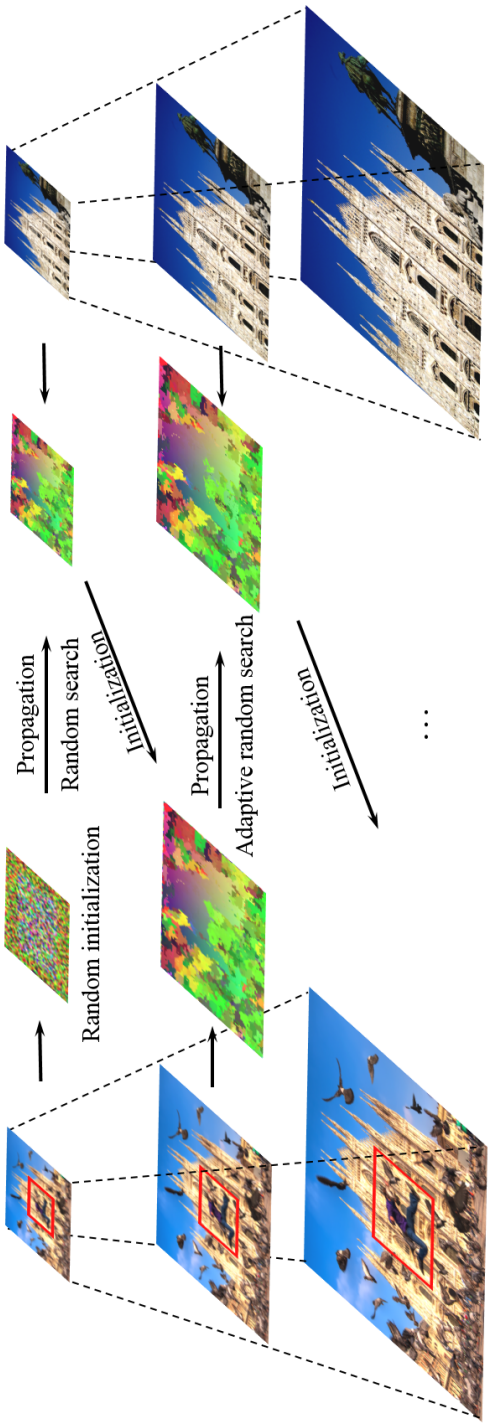


Figure 5.3: **Hierarchical PatchMatch flow diagram.** Image pyramids for the input image and the exemplar image are built. The dense correspondence is estimated from the higher to the lower pyramid level. Dense correspondence at lower pyramid level is initialized using the result from previous level. Once a reference NNF predicted by previous level interpolation function is available, random search is performed in an adaptive way to reduce complexity. (The hole region is marked with red rectangle.)

- Second, NNF interpolation can benefit from the hierarchical PatchMatch. From Eqn. (5.11), the estimation of $\mathbf{P}$ and $\mathbf{\Sigma}$ is essential to obtain a reliable $\widetilde{\mathbf{W}}$ which corresponds to a smooth and accurate NNF. The selection of control points can also affect the quality of NNF interpolation. As the overall problem is non-convex, a good initialization plays a key role in the EM algorithm. The EM algorithm can have a "warm" start by using the model parameters of the previous level for initialization. This gives a faster convergence. Moreover, the control points can be selected based on their probability of being inliers using the model parameters estimated in the previous level. A better control points selection scheme will improve NNF interpolation accuracy and make full use of the limited number of control points.

With the $(k+1)^{th}$ level NNF $\mathcal{F}^{k+1}$, the kth level NNF $\mathcal{F}^k$ is initialized by up-sampling $\mathcal{F}^k$ by a factor 2. The spatial range of $\mathcal{F}^k$ has been doubled compared to $\mathcal{F}^{k+1}$, while the scale and the rotation range are kept the same. In order to faithfully pass the matching parameters between different pyramid levels, the up-sampling method applied is the nearest neighbor interpolation:

$$\mathcal{F}^k(2\boldsymbol{p} - (i,j)) = (2u, 2v, s, \theta), \tag{5.12}$$

where $\mathcal{F}^{k+1}(\boldsymbol{p}) = (u, v, s, \theta)$, and $i, j = \{0, 1\}$.

The complexity of PatchMatch algorithm is proportional to the number of checked matching parameters from neighboring patches and randomly sampled parameters. With the initialized NNF $\mathcal{F}^k$, propagation and random search are performed in a way similar to the one described in Section 5.3.1. With the existence of the ROI, only the pixels in ROI will be processed. Moreover two approaches are proposed to reduce the complexity of random search. First, random search can be performed in an adaptive way. Once the interpolation function of the previous level is successfully estimated, it can be used to predict a reference NNF $\mathcal{F}^k_{\text{ref}}$ to guide the random search. For a

randomly generated parameter $\mathcal{F}^k_{\text{rnd}}(\boldsymbol{p})$ around $\mathcal{F}^k(\boldsymbol{p})$, if it is too far from the reference NNF (i.e. $\exists i \in \{1, 2, ..., D\} : \boldsymbol{d}_{\text{rnd}}(i) > \boldsymbol{\tau}_{\text{rnd}}(i)$ where $\boldsymbol{d}_{\text{rnd}} = \left|\mathcal{F}^k_{\text{rnd}}(\boldsymbol{p}) - \mathcal{F}^k_{\text{ref}}(\boldsymbol{p})\right|$ and $\boldsymbol{\tau}_{\text{rnd}} \in \mathbb{R}^D$ is the adaptive search threshold), it will unlikely be a good candidate and thus is abandoned. This leads to an early termination for the unreliable matching parameters and saves computation. Second, random search is conducted for the patches with edges which are determined via Canny edge detection. A random search is then performed only on patches which contains pixels with edge value larger than a threshold τ_{edge}. For the patches extracted from smooth region, random search would have a high probability of introducing wrong matching parameters which have even lower matching cost than the correct one. This can reduce complexity as well as avoid incorrect matching parameters being adopted.

The model parameters at level k are initialized using the model parameters from level $k+1$. Instead of using the exact value of $\boldsymbol{\Sigma}$ at previous level, it is re-scaled by a factor $\kappa > 1$ for initialization to avoid being trapped in a local minimum. The random selection of control points directly affects the results of the NNF interpolation algorithm. If outliers are picked as control points, the NNF interpolation could become less effective. When the EM algorithm of the previous level produces a reliable NNF interpolation function, the probability p_i of a data pair being inlier at current level can be predicted from Eqn. (5.4). Control points are selected from the data pairs with inlier probability $p_i > \rho$.

5.3.5 Image Completion

A smooth NNF $\mathcal{F}$ over the occluded region can be obtained through our proposed hierarchical PatchMatch and NNF interpolation:

$$\mathcal{F} = \mathbf{W}\mathbf{U}^{\text{T}}, \tag{5.13}$$

Algorithm 5: Image Completion via Dense Correspondence

```
Input: Input image I^1 with a ROI, exemplar image E^1;
Build image pyramids {I^k}_{k=1}^K and {E^k}_{k=1}^K;
Randomly initialize the coarsest level NNF F^K;
for i ← 1 to L do
    Perform PatchMatch between I^{K+1-i} and E^{K+1-i} within ROI;
    Interpolate NNF within ROI using the EM algorithm in Section 5.3.3;
    if i < K then
        Initialize the next level NNF F^{K-i} and EM algorithm model parameter
         θ^{K-i};
    end
end
Perform image completion to I^1as in Section 5.3.4.
Output: Completed image I^c.
```

where $\mathbf{W} = (\boldsymbol{w}_1, ..., \boldsymbol{w}_M) \in \mathbb{R}^{D\times M}$ is the interpolation coefficient matrix, and $\mathbf{U} \in \mathbb{R}^{N\times M}$ is the inter Gram matrix with $\mathbf{U}(i,j) = k(\boldsymbol{x}_i, \widetilde{\boldsymbol{x}}_j)$ and $\widetilde{\boldsymbol{x}}_j$ being the control points from pyramid level 1.

Let us define a mask $\mathcal{M}$ within the ROI which has value 1 on the locations with $p_i \geq \tau_p$ and value 0 otherwise. As the BRIEF descriptor is robust to small occlusions, the boundary of $\mathcal{M}$ may still contain image content that needs to be replaced. To include all the occluded image region, the mask $\mathcal{M}$ is eroded a few times. Let denote with $\mathcal{M}_i$ the region with mask value i for $i = 0, 1$. The mask region $\mathcal{M}_0$ indicates the image content on the input image which should be replaced. Every pixel $\boldsymbol{p} \in \mathcal{M}_0$ on the input image is replaced by the corresponding pixel value on location $\mathcal{E}^1(\mathcal{F}(p))$ from the exemplar image. The completed input image is denoted by $\mathcal{I}^c$.

Algorithm 5 summarizes our proposed image completion method based on dense correspondence.

5.4 Color Correction

Though images are taken near the same landmark, different camera parameters, shooting time, and shooting angles could result in significant color difference between the input image and the selected exemplar images. In order to have a visually pleasant result, color correction needs to be performed.

Different from the color transfer methods [120,121] where color correspondences are not available, pixel-wise color correspondences $\mathcal{D} = \{(x_{ci}, y_{ci})\}_{i=1}^{|\mathcal{M}_1|}$ have been established by the computed dense correspondence $\mathcal{F}^1$ within $\mathcal{M}_1$ with (x_{ci}, y_{ci}) being the corresponding color values on input image and exemplar image. A global color correction model can be used to minimize the color differences in a way similar to NRDC [122]. For each RGB color channel, a color transfer curve can be fitted using color correspondence $\mathcal{D}$ and applied to $\mathcal{I}^c$ within $\mathcal{M}_0$. The color transfer curve $\boldsymbol{f}_c$ is modeled as a piece-wise cubic spline with L knots:

$$\boldsymbol{f}_c(x) = \sum_{i=1}^{L} c(i) B(x - i), \tag{5.14}$$

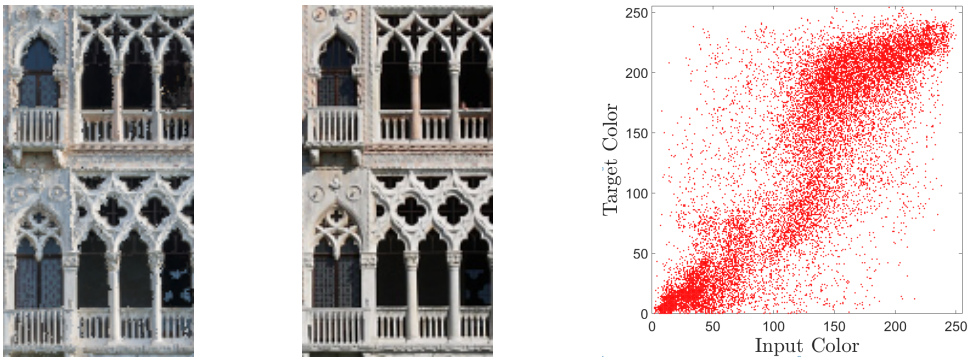
where $B(x)$ is the cubic B-spline basis function, and $c(i)$ are the B-spline coefficients.

The B-spline coefficient matrix $\mathbf{C} = (c(1), c(2), ..., c(L))^{\mathrm{T}}$ can be obtained in a least squares manner:

$$\mathbf{C} = (\mathbf{B}^{\mathrm{T}}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{T}, \tag{5.15}$$

where $\mathbf{B} \in \mathbb{R}^{|\mathcal{M}_1| \times L}$ is the B-spline basis matrix constructed using the input color values and $\mathbf{T} \in \mathbb{R}^{|\mathcal{M}_1|}$ is the target color value matrix containing the target color values in $\mathcal{M}_1$.

However, color correspondence is generally noisy as shown in Fig. 5.4, even when the estimated dense correspondence is accurate. The noise could be due to subtle differences in the images (for example, shadows) or intrinsically noisy image (i.e. the



(a) Input. (b) Target. (c) Color correspondence.

Figure 5.4: **An example of color correspondence.** (a) Reconstructed image segment with the estimated NNF. (b) The corresponding original image segment. (c) Their color correspondence (red channel) is noisy and with many outliers. The objective is to fit a color transfer curve which can faithfully map the color in (a) to that in (b).

same input color corresponds to a few different colors on the captured image). If a B-spline curve is applied to fit the noisy color correspondence, the transfer curve will be distorted with reduced dynamic range. This will greatly affect the color transfer curve estimation.

To further improve the global model, we adopt an EM-based algorithm for color correction as in the case of NNF interpolation. The color correspondence can be re-arranged as $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_1 \cup ... \cup \mathcal{D}_{255}$ with $\mathcal{D}_m = \{(m, y_{m,i})\}_{i=1}^{N_m}$ for input color value $m = 0, ..., 255$. From the observation, $\mathcal{D}_m$ can also be modeled as a mixture of Gaussian distributed inliers and uniformly distributed outliers. We need to estimate the percentage and variance of inliers $\boldsymbol{\theta}_m = (\gamma_m, \sigma_m^2)$ for $\mathcal{D}_m$.

During the E-step, the probability of color pair $(m, y_{m,i})$ being an inlier can be estimated using:

$$p_{m,i} = \frac{\gamma_m \exp\left(-\frac{\|y_{m,i} - \boldsymbol{f}_c(m)\|^2}{2\sigma_m^2}\right)}{\gamma_m \exp\left(-\frac{\|y_{m,i} - \boldsymbol{f}_c(m)\|^2}{2\sigma_m^2}\right) + \frac{1-\gamma_m}{256}\left(2\pi\sigma_m^2\right)^{\frac{1}{2}}}. \tag{5.16}$$

During the M-step, $\boldsymbol{\theta}_m = (\gamma_m, \sigma_m^2)$ is updated as:

Algorithm 6: Color Correction

```
Input: Completed image I^c and masks {M_i}_{i=0}^1;
for each RGB color channel do
    Fit a B-spline curve with color correspondence D;
    for i = 1 : T_out do
        for m = 0 : 255 do
            for j = 1 : T_in do
                E-step: update posterior probability as for Eqn.(5.16);
                M-step: update model parameteras for Eqn.(5.17);
            Remove color pairs with low probability from D;
        Fit a B-spline curve with updated D;
    Apply color transfer curve to image region on I^c within M_0 for image
      correction.
Output: Color corrected image O.
```

$$\begin{cases} \gamma_m = \frac{\mathrm{tr}(\mathbf{P}_m)}{N_m}, \\ \sigma_m^2 = \frac{(\mathbf{Y}_m - \mathbf{Z}_m)^{\mathrm{T}} \mathbf{P}_m (\mathbf{Y}_m - \mathbf{Z}_m)}{\mathrm{tr}(\mathbf{P}_m)}, \end{cases} \tag{5.17}$$

where $\mathbf{P}_m = \mathrm{diag}(p_{m,1}, ..., p_{m,N_m})$ is a diagonal matrix, $\mathbf{Y}_m = \left(y_{m,1}, ..., y_{m,N_m}\right)^{\mathrm{T}}$ and $\mathbf{Z}_m = (\boldsymbol{f}_c(m), ..., \boldsymbol{f}_c(m))^{\mathrm{T}} \in \mathbb{R}^{N_m}$ are column vectors.

For each input color value $m \in \{0, ..., 255\}$, the EM algorithm is applied for $\mathcal{T}_{\mathrm{in}}$ iterations and followed by removing the input and target color pairs with low probability being inliers (i.e. $p_{m,i} < \tau_{pc}$) from $\mathcal{D}$. When the outliers in every input color values have been removed, the color transfer curve is re-estimated with the updated color correspondence $\mathcal{D}$. After the whole process has been iterated for $\mathcal{T}_{\mathrm{out}}$ iterations, the color transfer function is applied to image regions on $\mathcal{I}^c$ within $\mathcal{M}_0$. **Algorithm 6** summarizes our proposed color correction algorithm.

5.5 Numerical Results

In this section, we report the implementation details and numerical results of our proposed image completion method and compare them to other commonly used methods. Testing images are from [4].

5.5.1 Image Completion via Dense Correspondence

For PatchMatch, the NNF parameters (u, v, s, θ) take discrete values. The scale parameter s has been discretized into 256 discrete scales. The minimum and maximum scale is 0.33 and 3.00, respectively. The scale ratio between consecutive scales is fixed. Similarly, there are 90 discrete orientations for the orientation parameter θ. The start angle is $-45°$, and the end angle is $45°$ with the angle step being $1°$. In random search, the adaptive search threshold is set to $\boldsymbol{\tau}_{\text{rnd}} = (10, 10, 25, 9)^{\text{T}}$, the edge threshold to determine edge patches is set to $\tau_{\text{edge}} = 60$. The size of the image pyramid K is the largest integer such that $2^{-K} \times \max(m, n) \geq 32$ for $\mathcal{I}^1 \in \mathbb{R}^{m \times n}$. For example, if the image size is 800×1024, then $K = 5$.

The patch size is selected as 7×7 (i.e. patch radius is $r = 3$). The length of the BRIEF descriptor is set to be $n_b = 128$. When the pixels on patch $\mathcal{E}_r^k(\mathcal{F}^k(p))$ does not fall on a regular grid of image $\mathcal{E}^k$, bi-linear interpolation is applied to compute its pixel values. Its BRIEF descriptor is then extracted from the interpolated pixel values.

Data normalization has been applied to each column dimension of $\mathbf{X}$ and $\mathbf{Y}$ so that each parameter has zero mean and unit variance. It enables us to fix the Gaussian kernel filter range β in NNF interpolation for different "hole" sizes. The EM algorithm will stop if it does not converge within $\mathcal{T}_{\text{EM}} = 50$ iterations. When the model parameter of the previous level is not available, the percentage of inliers is initialized as $\gamma = 0.5$, otherwise the model parameter is initialized using the parameter of the previous level.

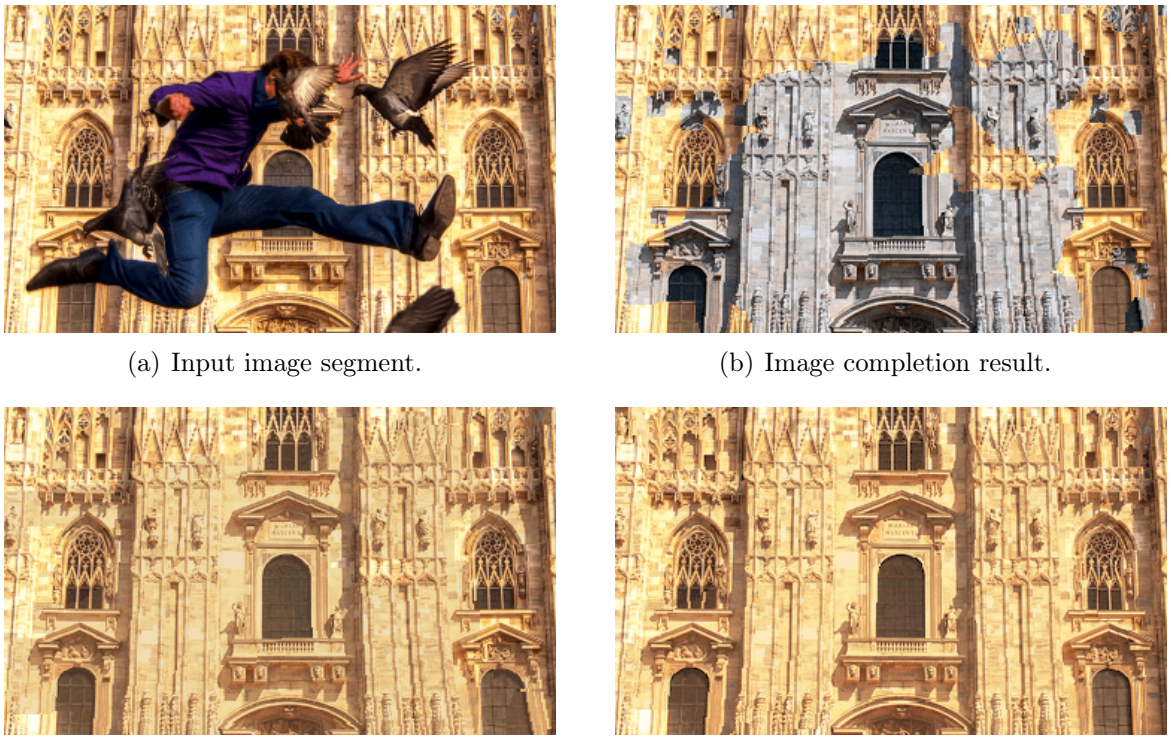
The re-scaling factor for the initialization of the model parameter across pyramid level was set to $\kappa = 2$. The regularization parameter λ is set to 10. A data pair will be selected as a control point only if it is very likely an inlier (i.e. its inlier probability estimated using the previous level interpolation function exceeds $\rho = 0.99$). The threshold value for identifying mask region is set to $\tau_p = 0.7$ through cross-validation. A data point with the confidence score larger than 0.7 is considered to be reliably estimated inlier.

It is essential to have some good matching parameters found at the initial pyramid level so that these good parameters can be propagated at consecutive pyramid levels. With more iterations, a larger number of random searches will be applied to exploit the parameter space in order to increase the probability of obtaining some good matches. The total number of iterations applied at the first level was selected as 16. Each type of propagation is performed 4 times. For the other levels, the number of iteration for propagation is reduced to 4, i.e. each type propagation is performed only once.

For NNF interpolation, the number of control points M is around 0.5% of the total number of points N within the ROI. The filter range of the Gaussian kernel is selected as $\beta = 100$. There are two reasons to set the filter range to such a large value. First, the filter range β controls the smoothness of the interpolation result. As the NNF is generally noisy, a large β can remove the noisy high frequency components. Second, a small filter range is not able to interpolate the NNF within a large occluded region as the Gaussian kernel will return a very small value when the input location is far from the control points.

5.5.2 Color Correction

The B-spline for color correction has $L = 7$ knots. The maximum inner $\mathcal{T}_{\text{in}}$ and outer $\mathcal{T}_{\text{out}}$ iterations for the EM algorithm is 3 and 5, respectively. The threshold value for



(a) Input image segment. (b) Image completion result.

(c) Color correction with B-spline. (d) Proposed color correction.

Figure 5.5: **Color correction comparison.** (a) Image segment within the ROI. (b) Image completion result by our method before color correction. (c) Color correction result by directly fitting a B-spline curve for each color channel using the obtained color correspondence. (d) Color correction result of our proposed method. (Better view in electronic version.)

defining inliers is set to $\tau_{pc} = 0.7$ via cross-validation.

In this section, we make visual comparisons between the result without color correction, with color correction by direct B-spline fitting, and with our proposed color correction method. As shown in Fig. 5.5 (b), though the completed texture is highly coherent with the original image, there is an apparent color difference between the completed region and the original image in the image completion result. If a B-spline curve is fitted for every color channel using the obtained color correspondence, the color difference in 5.5 (c) is alleviated, however, the completed region is still a bit darker and can be identified. This is caused by using a noisy color correspondence for color correction. Fig 5.5 (d) shows our proposed color correction result where the completed region has no visible color difference compared to the input image in Fig 5.5 (a). This validates

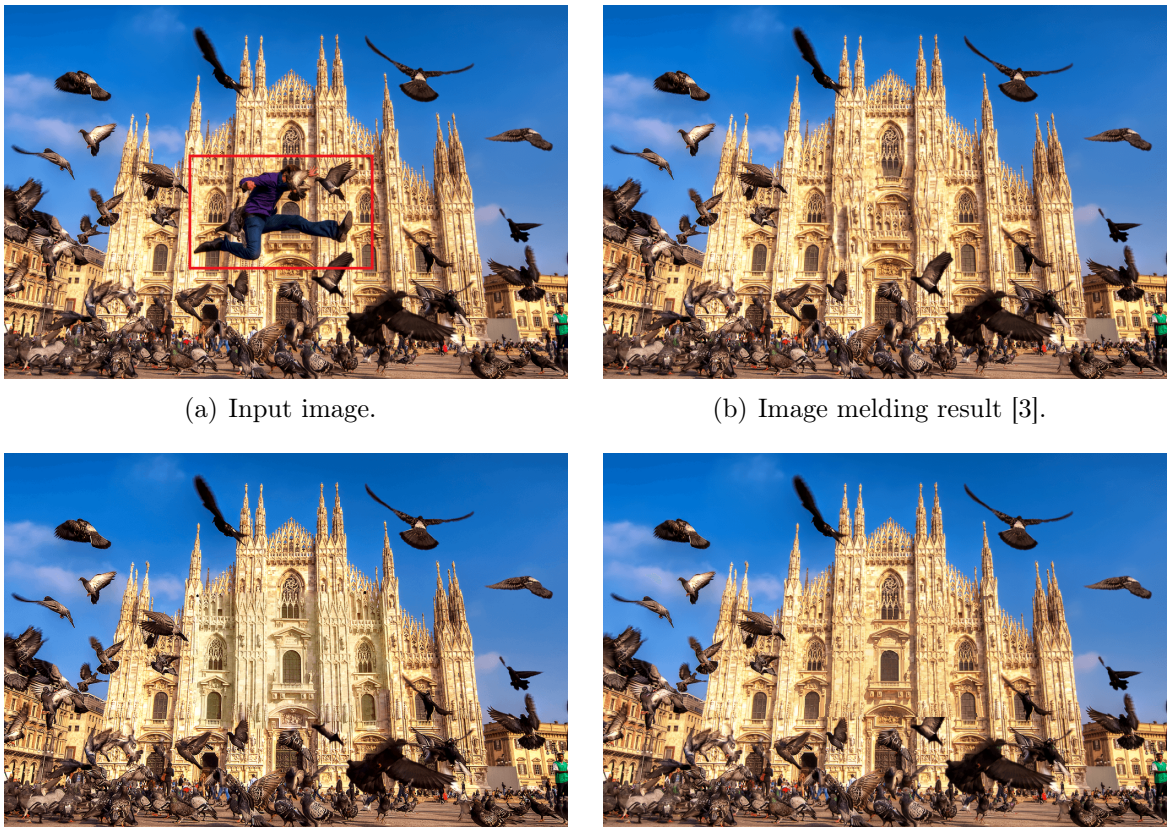
the effectiveness of our proposed color correction method for image completion.

5.5.3 Comparison with State-of-the-Art Methods

In this section, we compare our proposed image completion via dense correspondence method with state-of-the-art image completion methods [3, 4]. Image melding [3] is a single image completion method based on patch correspondence within the input image itself. Zhu *et al.*'s method [4] is a recently proposed Internet-based image completion method based on sparse and line correspondence. Image melding method requires a detailed mask to identify the unwanted image region, while Zhu *et al.*'s method and our proposed method only need a rectangle to mark ROI. The Matlab realization of image melding is from authors' website. The image completion results of [4] are provided by the authors.

Fig. 5.6 and Fig. 5.7 present two set of image completion results for detailed comparison. Fig. 5.8 - Fig. 5.11 show more comparison results. For completeness, we also show, in Fig. 5.13, the exemplar images used by our image completion method.

Fig. 5.6 (a) shows an image which is relatively easy to complete since the background is a large plane where the correspondence can be well modeled by a perspective transform. We would like to remove the stranger from the photo. In Fig. 5.6 (b), image melding method reconstructs a distorted image region since there are not enough repetitive image patches. This reveals the shortcoming of image completion methods based on a single image. In Fig. 5.6 (c), Zhu *et al.*'s method utilizes exemplar images from Internet for image completion. The completed image region is essentially coherent with the original image content in the input image. However, there is a visible miss-alignment on the top boundary between completed region and the input image. The completed image region also has a slightly different color style compared with the input image as there is no color correction performed in [4]. Fig. 5.6 (d) shows the



(a) Input image.

(b) Image melding result [3].

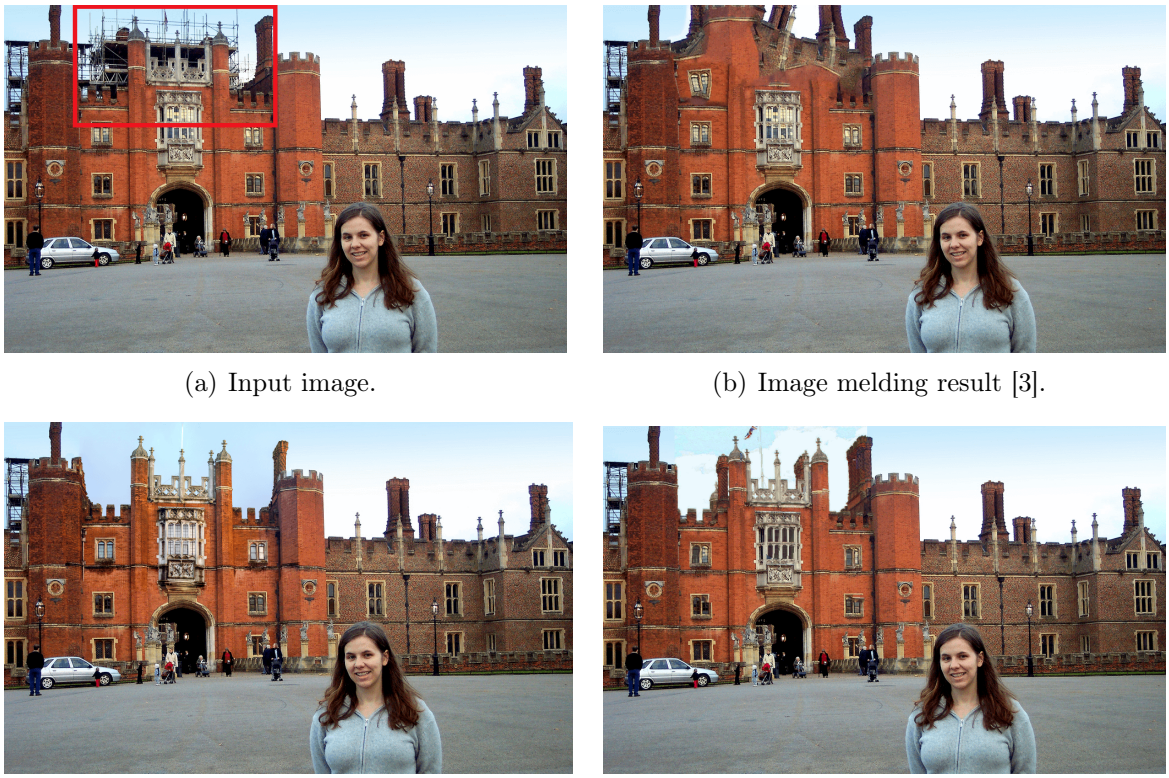
(c) Zhu *et al.*'s result [4].

(d) Our completion result.

Figure 5.6: **Comparison with state-of-the-art image completion methods for the image taken near the Duomo of Milan.** (a) Input image with a labeled ROI (red rectangles). (b) Image melding result [3]. (c) Image completion result from [4]. (d) our proposed image completion via dense correspondence result. (Better view in electronic version.)

image completion result by our proposed method. With dense correspondence and NNF interpolation, the completed image content is highly consistent with input image. Though the exemplar has a distinct color style as shown in Fig. 5.5 (b), our final image completion result has no visible color difference thanks to our color correction algorithm.

In Fig. 5.7 (a), there is a construction site on the Hampton Palace. We would like to restore its normal look. As shown in Fig. 5.7 (b), image melding is not able to reconstruct a natural-looking building due to the large size of the "hole" and non-stationary image content. In Fig. 5.7 (c), Zhu *et al.*'s method faithfully reconstructs a natural looking image. In Fig. 5.7 (d), our image completion result is also faithful.



(a) Input image.

(b) Image melding result [3].

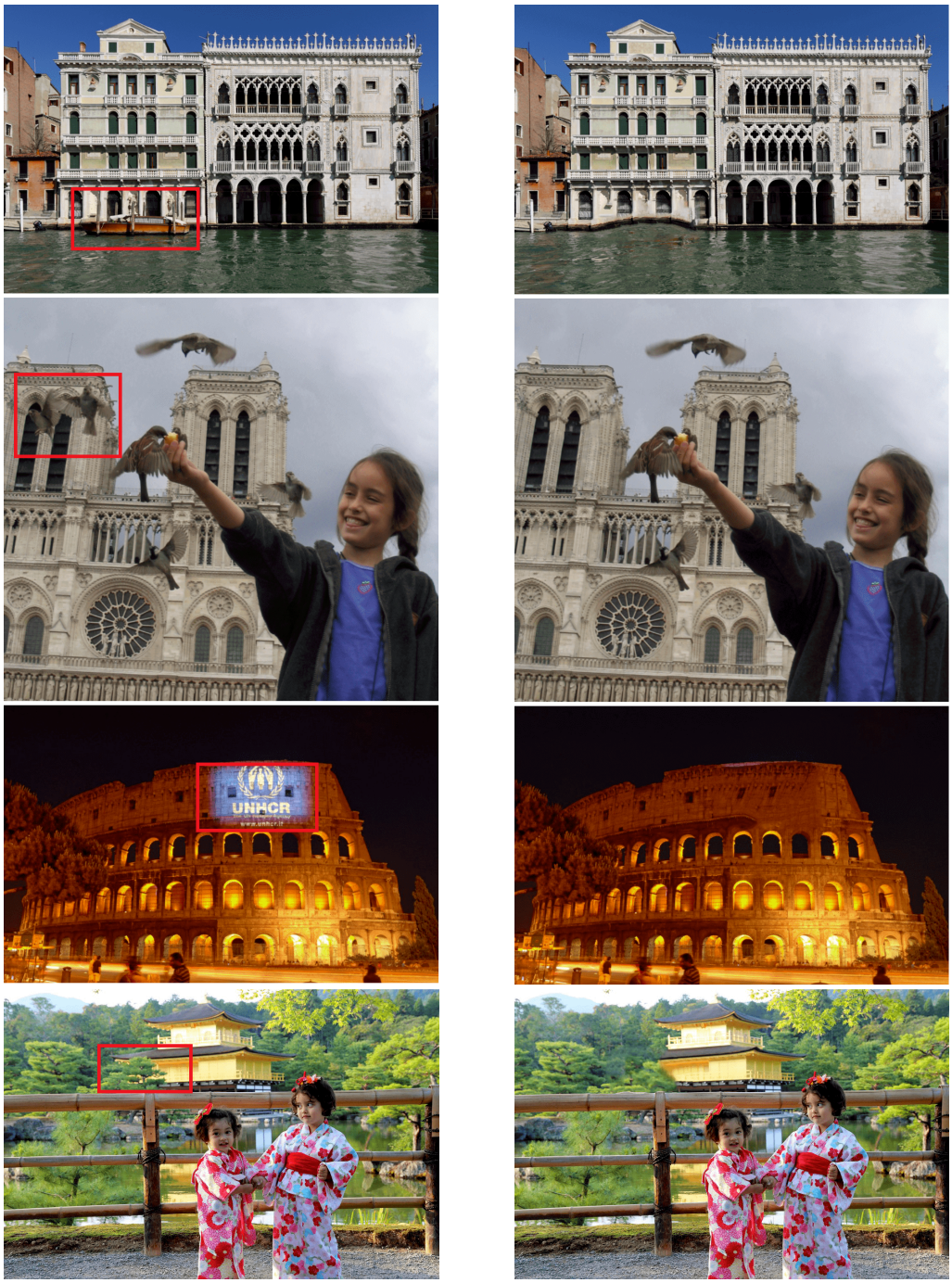
(c) Zhu *et al.*'s result [4].

(d) Our completion result.

Figure 5.7: **Comparison with state-of-the-art image completion methods for the image taken near the Hampton Palace.** (a) Input image with a labeled ROI (red rectangles). (b) Image melding result [3]. (c) Image completion result from [4]. (d) our proposed image completion via dense correspondence result. (Better view in electronic version.)

When making a comparison between the result in Fig. 5.7 (c) and Fig. 5.7 (d), we can find that our result has an indistinguishable color while there is a slight color difference in the result of [4]. The building in our completed result (shown in Fig. 5.7 (d)) has almost the same orientation as that in the input image (Fig. 5.7 (a)), while the building in Fig. 5.7 (c) is slightly tilted.

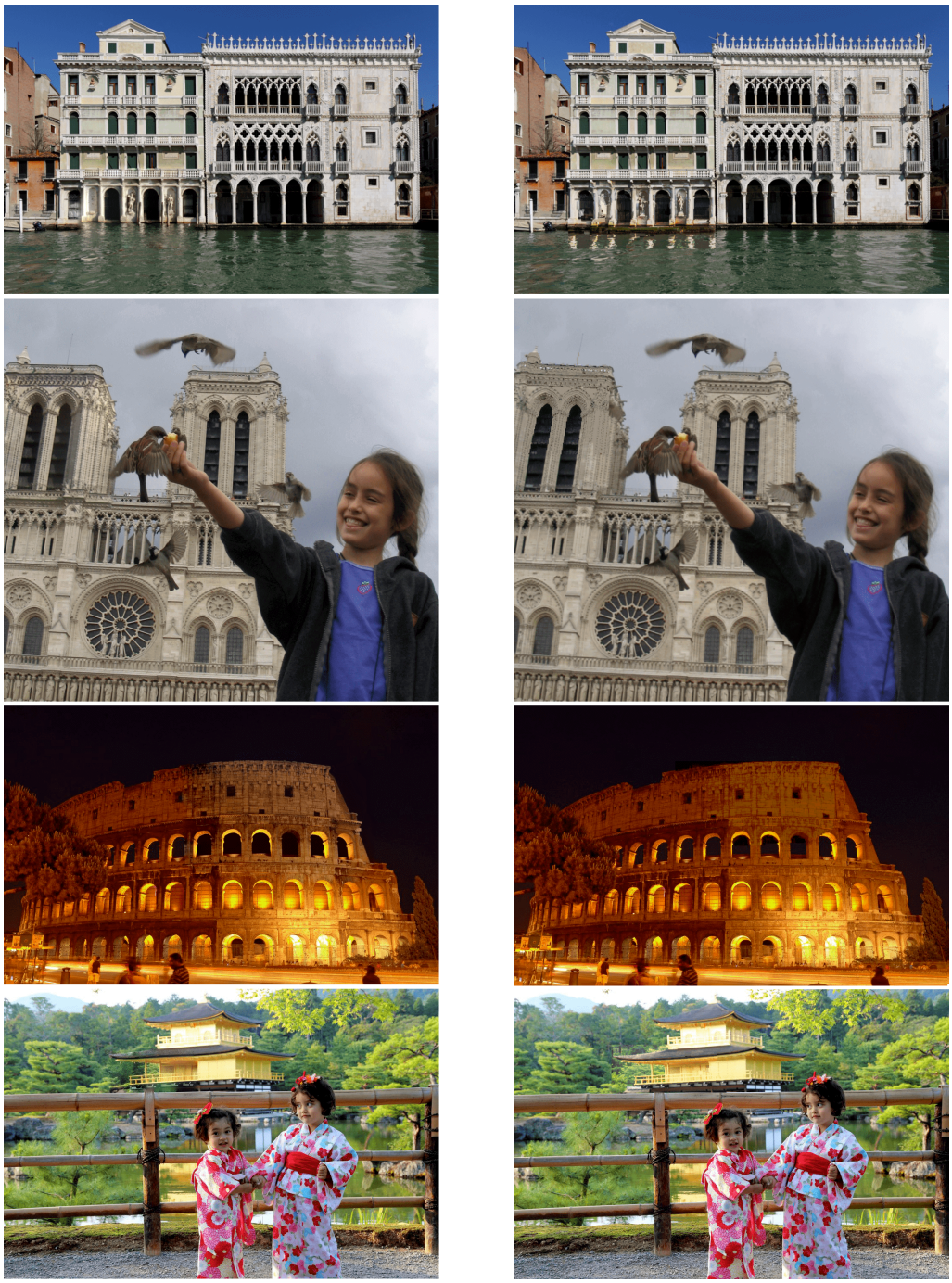
In the first row of Fig. 5.8 and Fig. 5.9, image melding still fails to reconstruct a natural looking image. Zhu *et al.*'s method faithfully reconstructs the image content behind the boat. However, the reconstructed image region seems to have a larger scale compared to the input image and has a slight miss-alignment. The image completion result by our proposed method feels more realistic. It is interesting to find that image



(a) Input image. (b) Image melding result [3].

Figure 5.8: **Comparison with state-of-the-art image completion methods.** (a) Input images with a labeled ROI (red rectangles). (b) Image melding results [3]. The input images from top to bottom are taken near Palazzo Santa Sofia, Notre-Dame de Paris, Colosseum, and Kinkaku-ji, respectively. (Better view in electronic version.)

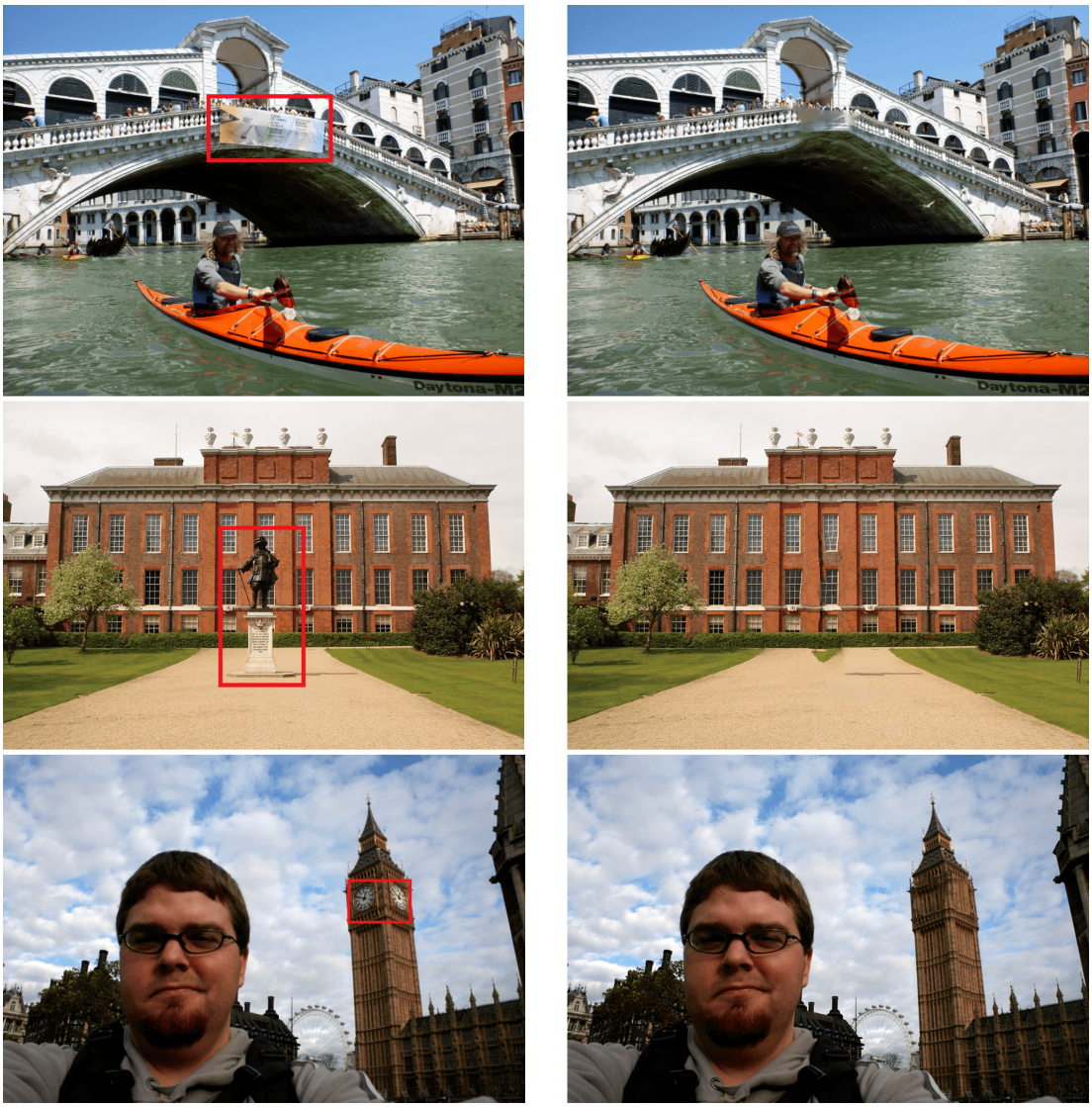
melding method achieves a better completion result for the second input image in Fig. 5.8 - 5.9 (the image taken near the Notre-Dame de Paris). This is because the Notre-



(a) Zhu *et al.*'s result [4].

(b) Our completion result.

Figure 5.9: **Comparison with state-of-the-art image completion methods.** (a) Image completion results from [4]. (b) our proposed image completion via dense correspondence results. The input images from top to bottom are taken near Palazzo Santa Sofia, Notre-Dame de Paris, Colosseum, and Kinkaku-ji, respectively. (Better view in electronic version.)



(a) Input image. (b) Image melding result [3].

Figure 5.10: **Comparison with state-of-the-art image completion methods.**. (a) Input images with a labeled ROI (red rectangles). (b) Image melding results [3]. The input images from top to bottom are taken near Rialto Bridge, Kensington Palace, and Big Ben, respectively. (Better view in electronic version.)

Dame de Paris is symmetric and image melding method makes use of image content from the right side. Single image based method also has the advantage of consistent color across the reconstructed image. For the input image taken near the Colosseum, image melding method completes the "hole" with slight distortion. The result by Zhu *et al.*'s method has a visible inconsistency in the color rendering. Our proposed method reconstructs the content within the ROI realistically. In the last row, Zhu *et al.*'s

(a) Zhu *et al.*'s result [4].

(b) Our completion result.

Figure 5.11: **Comparison with state-of-the-art image completion methods.** (a) Image completion results from [4]. (b) our proposed image completion via dense correspondence results. The input images from top to bottom are taken near Rialto Bridge, Kensington Palace, and Big Ben, respectively. (Better view in electronic version.)

method and our method have similar completion results. Three further examples are shown in Fig. 5.10 and Fig. 5.11 where we can appreciate a behaviour of the three algorithms similar to what seen in Fig. 5.8 - 5.9. Image melding tends to struggle when the region to be completed is large and there is a lack of similar patches in the image. It performs well otherwise. The other two methods, i.e. [4] and ours, perform consistently well with our method often providing a more consistent rendering.

Figure 5.12: **Sample image completion results by a deep neural network based method [5].**

Figure 5.13: **The exemplar images used for image completion.**

The data-driven methods of [5, 40] are powerful and are able to learn a single and generally effective model for image completion from a large dataset. In Fig 5.12, sample image completion results using [5] are shown. We conjecture that the unsatisfactory results can be due to the inconsistency between the training dataset and the testing image. Our proposed method is a model-based approach so it does not suffer from potentional training/testing mismatches and the prior information (i.e. the smoothness of the dense correspondence and the color transfer curve) is important to provide the much better image completion results. We also note that our proposed method relies on the use of "side information" provided by the retrieved exemplar images, whereas the other two approaches do not need an exemplar image. This also clarify why we perform much better.

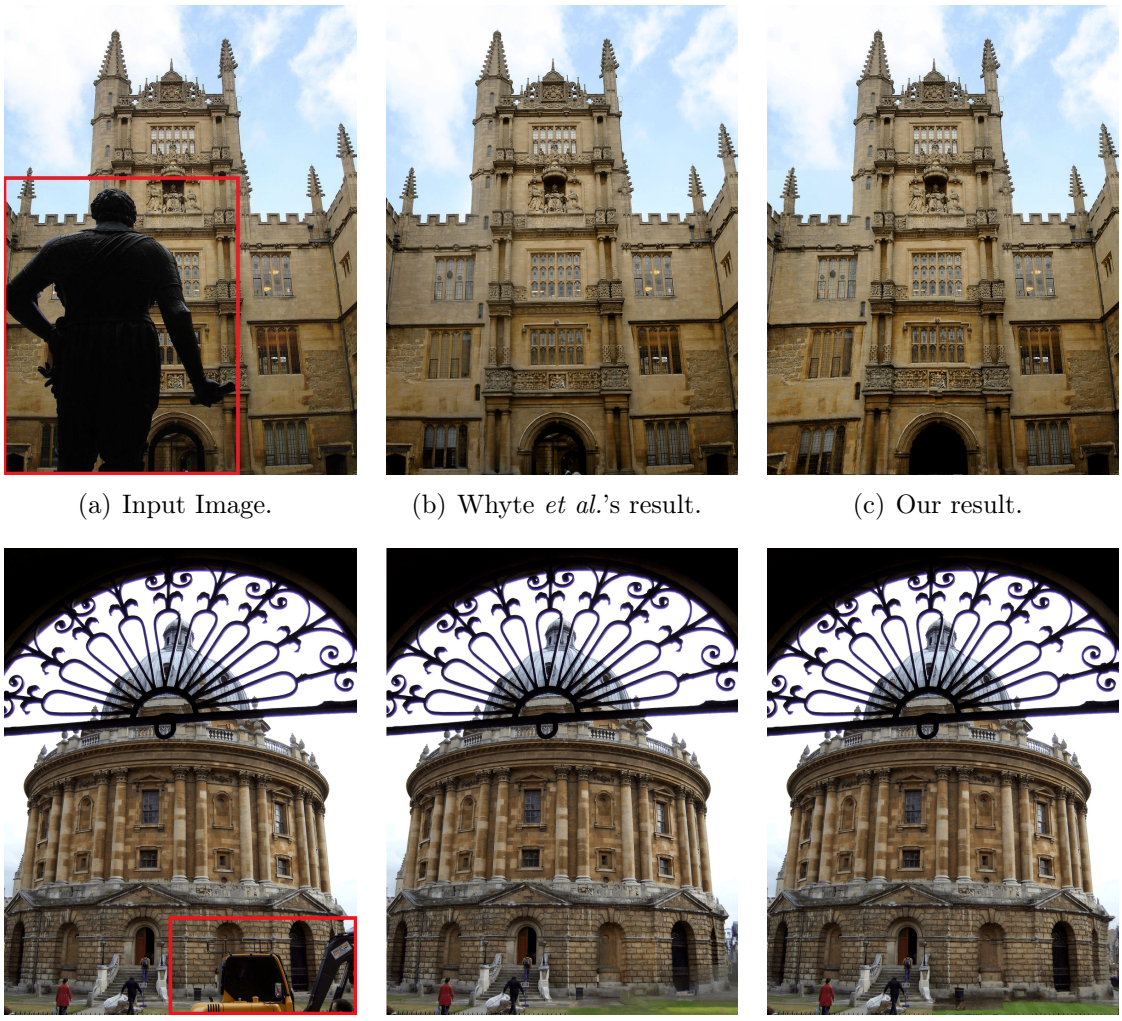
5.5.4 Subjective Evaluation for Image Completion

In order to perform subjective evaluation, we have surveyed 34 people about their preference over different image completion results. Each respondent was asked to make a pairwise comparison for 9 pairs of image completion results. The 9 pairs of image completion results are randomly selected from the results of image melding (method 1), Zhu *et al.*'s method (method 2), and our proposed method (method 3) shown in Fig. 5.6 - Fig. 5.11. For method i and j, with $i, j = 1, 2, 3$ and $i \neq j$, we have obtained 102 pairwise comparison results. This gives us a winning matrix $\mathbf{\Xi} \in \mathbb{R}^{3\times 3}$:

$$\mathbf{\Xi} = \begin{bmatrix} 0 & 0.95 & 0.83 \\ 0.05 & 0 & 0.75 \\ 0.17 & 0.25 & 0 \end{bmatrix}, \tag{5.18}$$

where $\mathbf{\Xi}(i, j)$ indicates the fraction of respondents that think the result of method j is better than that of method i with $i, j \in \{1, 2, 3\}$.

From the winning matrix $\mathbf{\Xi}$, we can find that about 83% and 75% of the respondents think the image completion results of our proposed method are better than those of image melding and Zhu *et al.*'s method, respectively. There are also some interesting results from the survey. We thought that image melding achieves the best result for the Notre-Dame de Paris case, while most respondents think Zhu *et al.*'s method produces a better result. This could be due to a brighter color and more detailed structure in the result of Zhu *et al.*'s method. Another example is that most respondents prefer our result in the Big Ben case over Zhu *et al.*'s. This may be due to a slightly miss-alignment on the Zhu *et al.*'s result.



(a) Input Image. (b) Whyte *et al.*'s result. (c) Our result.

(d) Input Image. (e) Whyte *et al.*'s result. (f) Our result.

Figure 5.14: **Further image completion results evaluated on the images from [6].**

5.5.5 Computation Complexity and Further Examples

Table 5.5.5 shows the computation time of our proposed image completion method. We have implemented our method using C++. The evaluation is performed on a PC with Intel Core i7 3.4 GHz CPU. From Table I, the computation time of our image completion algorithm is generally linear with the number of pixels in the ROI region, while may fluctuate depending on the complexity of the image content. To show that our method works well also in different conditions, we have also performed image completion on two images from [6]. The results are shown in Fig. 5.14.

Test Image	Duomo	Hampton Palace	Palazzo Santa Sofia	Notre Dame	Colosseum	Kinkaku-ji	Rialto Bridge	Kensington Palace	Big Ben
ROI Size	81.4	68.0	35.1	19.5	48.0	30.5	27.9	82.5	3.2
Time (s)	27.58	64.34	13.28	10.91	44.19	10.08	9.01	31.57	5.76

Table 5.1: Computation time (s) of the proposed algorithm. The ROI size is the number of pixels in the rectangle ROI in 10^3 pixels.

5.6 Summary

In this chapter, we have proposed a novel Internet-based image completion method. Instead of correlating the input image with the retrieved exemplar image based on sparse correspondence, we propose to make use of dense correspondence to relate every pixel within the ROI to a pixel on the exemplar image. This enables accurate image content transfer from the exemplar image and reliable color correction to remove color difference between the completed part and the input image.

A hierarchical framework is built to perform dense correspondence estimation, image completion, and color correction jointly. The dense correspondence estimation is based on a hierarchical variation of the PatchMatch method. Contrary to other approaches, BRIEF descriptor is adopted to handle the inter image differences. As the estimated NNF is in general noisy and with large occlusion within the "hole", an EM-based method is applied to identify reliable inliers from the estimated dense correspondence and interpolate a smooth NNF over the occluded region. A global model is applied for color correction. As a noisy color correspondence is observed, color correction is also based on a similar EM algorithm to remove outlier color correspondences. To demonstrate our method, we made a comparison with the state-of-the-art image completion methods. From the numerical results, we can see that our proposed image completion method achieves photo realistic results on a wide range of images.

Chapter 6

Conclusion

6.1 Summary

In this thesis we have considered two specific image enhancement problems and pushed the boundaries on both the learning-based direction and the model-based direction. For single image super-resolution, we presented a new learning-based multi-layer dictionary model using either unstructured dictionary or convolutional dictionary. The proposed model design provides useful insights on the workings of Deep Neural Networks (DNNs) and enables us to bridge the established sparse representation theory with DNNs. Finally, we presented a model-based image completion algorithm which transfers image contents from an exemplar image to the input image by using an interpolated smooth nearest neighbor field.

In Chapter 3, we presented a Deep Analysis Dictionary Model (DeepAM) which consists of multiple layers of linear and non-linear transforms with an application on single image super-resolution. A L-layer DeepAM is composed of L layers of analysis dictionary and element-wise soft-thresholding pairs and a single layer of synthesis dictionary. The forward pass of DeepAM is efficient and contains matrix multiplication with the

analysis dictionaries and the element-wise soft-thresholding operations. To have an effective deep model, we propose to divide each analysis dictionary into an Information Preserving Analysis Dictionary (IPAD) and a Clustering Analysis Dictionary (CAD) to preserve the essential information from the input signal and generate discriminative feature representations, respectively. Based on different learning objectives, we proposed learning algorithms for both the IPAD and its soft-thresholds pair and the CAD and its thresholds pair. We demonstrated in simulation results that the learned DeepAM achieves comparable performance as the Deep Neural Networks (DNNs) which has the same structure and is optimized using backpropagation and other existing single image super-resolution methods. The model design of DeepAM provides some insights on the workings of the fully connected DNNs.

In Chapter 4, we generalized DeepAM by replacing its linear transforms from unstructured dictionaries to convolutional dictionaries. A L layer Deep Convolutional Analysis Dictionary Model (DeepCAM) consists of L layers of convolutional analysis dictionary and element-wise soft-thresholding pairs and a single layer of convolutional synthesis dictionary. Each convolutional analysis dictionary is also composed of a convolutional IPAD sub-dictionary and a convolutional CAD convolutional analysis sub-dictionary. Based on the convolutional structure within DeepCAM, the minimum number of IPAD and CAD atoms at each layer has been derived based on information preserving and discriminative feature generation requirement. The convolutional IPAD and CAD are learned using variations of our proposed convolutional analysis dictionary learning algorithm which is able to achieve efficient learning by exploiting the structural properties of convolutional dictionaries. Simulation results show that the learned DeepCAM achieves a better performance than DeepAM while using a smaller number of parameters.

In Chapter 5, we considered the Internet-based image completion problem whose objective is to transfer image contents from an exemplar image to the input image and

the transferred contents should be consistent with the input image. We presented an image completion algorithm based on dense correspondence which models pixel-wise correspondences between two images and therefore has the potential to generate highly accurate image content transfer. A hierarchical PatchMatch algorithm is proposed to progressively generate the nearest neighbor field (NNF) between the input image and the exemplar image. The estimated NNF is usually noisy and with a large occlusion area corresponding to the region to be replaced. By modelling NNF as a smooth field, we propose to an Expectation-Maximization algorithm to interpolate a smooth NNF over the occlusion area which is then used to transfer contents. Color correction is further applied to diminish the color differences between the transferred contents and the original input image. Numerical results show that the proposed image completion method can achieve photo realistic results.

6.2 Future research

We conclude this thesis with some future research directions.

1. **Joint deep dictionary learning algorithm** – We presented a layer-wise algorithm for learning the Deep Analysis Dictionary Model (DeepAM). The multi-layer analysis dictionaries and soft-thresholds are learned sequentially from shallow layers to deep layers and are therefore independently optimized. Simulation results in Chapter 3 demonstrate that backpropagation algorithm can be applied to further improve a learned DeepAM with minor changes. An efficient learning algorithm which is able to learn all components jointly and incorporate the desired properties of the analysis dictionaries and the soft-thresholds will be useful for a deep dictionary model.

2. **General non-linear functions** – We have applied soft-thresholding operators as the non-linear functions in DeepAM. This is Justified if a sparse modelling point

of view is used. As discussed in Chapter 3, a DeepAM with soft-thresholding can be considered as a deep model with rectified linear unit (ReLU) as non-linear functions, while it is still not clear how to learn DeepAM with other non-linear functions. It would be very interesting to generalize DeepAM with soft-thresholding to DeepAM with other non-linear functions such as Leaky ReLU function, Sigmoid function, and Tanh function. This would enable us to further improve the performance of the learned DeepAM and apply DeepAM on other applications like for example classification tasks.

3. **Image completion with multiple exemplar images** – We presented an image completion algorithm based on an single exemplar image. This imposes some requirements on the quality of this exemplar image, like for example having no occlusions. A more robust approach is to use multiple exemplar images for image completion. Based on the contents of the exemplar images, the transferred image contents can come from different exemplar images. This will lead to a more flexible framework and better image completion results.

Bibliography

[1] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[2] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International Conference on Curves and Surfaces*. Springer, 2010, pp. 711–730.

[3] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image melding: Combining inconsistent images using patch-based synthesis." *ACM Trans. Graph.*, vol. 31, no. 4, pp. 82–1, 2012.

[4] Z. Zhu, H.-Z. Huang, Z.-P. Tan, K. Xu, and S.-M. Hu, "Faithful completion of images of scenic landmarks using internet images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 8, pp. 1945–1958, 2016.

[5] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 107, 2017.

[6] O. Whyte, J. Sivic, and A. Zisserman, "Get out of my picture! internet-based inpainting," in *Proceedings of the 20th British Machine Vision Conference, London*, 2009.

[7] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1920–1927.

[8] ——, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 111–126.

[9] J.-J. Huang, W.-C. Siu, and T.-R. Liu, "Fast image interpolation via random forests," *IEEE Transactions on Image Processing*, vol. 24, no. 10, pp. 3232–3245, 2015.

[10] J.-J. Huang and W.-C. Siu, "Learning hierarchical decision trees for single image super-resolution," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[11] J.-J. Huang, T. Liu, P. L. Dragotti, and T. Stathaki, "SRHRF+: Self-example enhanced single image super-resolution using hierarchical random forests," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on New Trends in Image Restoration and Enhancement*, July 2017.

[12] M. Aharon, M. Elad, A. Bruckstein *et al.*, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, p. 4311, 2006.

[13] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[14] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision*. Springer, 2014, pp. 184–199.

[15] ——, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[16] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[17] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[18] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European conference on computer vision.* Springer, 2016, pp. 391–407.

[19] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[21] C. Guillemot and O. Le Meur, "Image inpainting: Overview and recent advances," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 127–144, 2014.

[22] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on image processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[23] Y. Wexler, E. Shechtman, and M. Irani, "Space-time completion of video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 463–476, 2007.

[24] C.-W. Fang and J.-J. J. Lien, "Rapid image completion system using multiresolution patch-based directional and nondirectional approaches," *IEEE Transactions on Image Processing*, vol. 18, no. 12, pp. 2769–2779, 2009.

[25] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "Patchmatch: a randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics-TOG*, vol. 28, no. 3, p. 24, 2009.

[26] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, "The generalized patchmatch correspondence algorithm," in *European Conference on Computer Vision*. Springer, 2010, pp. 29–43.

[27] T.-H. Kwok, H. Sheung, and C. C. Wang, "Fast query for exemplar-based image completion," *IEEE Transactions on Image Processing*, vol. 19, no. 12, pp. 3106–3115, 2010.

[28] Z. Xu and J. Sun, "Image inpainting by patch propagation using patch sparsity," *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1153–1165, 2010.

[29] O. Le Meur, M. Ebdelli, and C. Guillemot, "Hierarchical super-resolution-based inpainting," *IEEE Transactions on Image Processing*, vol. 22, no. 10, pp. 3779–3790, 2013.

[30] K. He and J. Sun, "Image completion approaches using the statistics of similar patches," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2423–2435, 2014.

[31] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf, "Image completion using planar structure guidance," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 129, 2014.

[32] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[33] S. Gepshtein and Y. Keller, "Image completion by diffusion maps and spectral relaxation," *IEEE Transactions on Image Processing*, vol. 22, no. 8, pp. 2983–2994, 2013.

[34] C. Barnes, F.-L. Zhang, L. Lou, X. Wu, and S.-M. Hu, "Patchtable: efficient patch queries for large datasets and applications," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 97, 2015.

[35] K. He and J. Sun, "Computing nearest-neighbor fields via propagation-assisted kd-trees," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 111–118.

[36] S. Korman and S. Avidan, "Coherency sensitive hashing," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1607–1614.

[37] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, "Image completion with structure propagation," *ACM Transactions on Graphics (ToG)*, vol. 24, no. 3, pp. 861–868, 2005.

[38] Y. Liu and V. Caselles, "Exemplar-based image inpainting using multiscale graph cuts," *IEEE Transactions on Image Processing*, vol. 22, no. 5, pp. 1699–1711, 2013.

[39] N. Komodakis and G. Tziritas, "Image completion using efficient belief propagation via priority scheduling and dynamic pruning," *IEEE Transactions on Image Processing*, vol. 16, no. 11, pp. 2649–2661, 2007.

[40] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[41] H. Amirshahi and S. Kondo, "An image completion algorithm using occlusion-free images from internet photo sharing sites," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 91, no. 10, pp. 2918–2927, 2008.

[42] J. Hays and A. A. Efros, "Scene completion using millions of photographs," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 4.

[43] Q. Shan, B. Curless, Y. Furukawa, C. Hernandez, and S. M. Seitz, "Photo uncrop," in *European Conference on Computer Vision*. Springer, 2014, pp. 16–31.

[44] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[45] E. Sober, *Ockham's razors*. Cambridge University Press, 2015.

[46] N. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals," *Applied and Computational Harmonic Analysis*, vol. 10, no. 3, pp. 234–253, 2001.

[47] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 123–151, 2005.

[48] D. Labate, W.-Q. Lim, G. Kutyniok, and G. Weiss, "Sparse multidimensional representation using shearlets," in *Wavelets XI*, vol. 5914. International Society for Optics and Photonics, 2005, p. 59140U.

[49] G. Easley, D. Labate, and W.-Q. Lim, "Sparse directional image representations using the discrete shearlet transform," *Applied and Computational Harmonic Analysis*, vol. 25, no. 1, pp. 25–46, 2008.

[50] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse Problems*, vol. 23, no. 3, p. 947, 2007.

[51] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[52] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[53] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[54] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

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

[56] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[57] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition."

[58] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 5. IEEE, 1999, pp. 2443–2446.

[59] W. Dai, T. Xu, and W. Wang, "Simultaneous codeword optimization (simco) for dictionary update and learning," *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6340–6353, 2012.

[60] R. Rubinstein, T. Peleg, and M. Elad, "Analysis k-svd: A dictionary-learning algorithm for the analysis sparse model," *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 661–677, 2013.

[61] R. Rubinstein and M. Elad, "Dictionary learning for analysis-synthesis thresholding," *IEEE Transactions on Signal Processing*, vol. 62, no. 22, pp. 5962–5972, 2014.

[62] Y. Chen, R. Ranftl, and T. Pock, "Insights into analysis operator learning: From patch-based sparse models to higher order MRFs," *IEEE Transactions on Image Processing*, vol. 23, no. 3, pp. 1060–1072, 2014.

[63] S. Hawe, M. Kleinsteuber, and K. Diepold, "Analysis operator learning and its application to image reconstruction," *IEEE Transactions on Image Processing*, vol. 22, no. 6, pp. 2138–2150, 2013.

[64] J. Dong, W. Wang, W. Dai, M. D. Plumbley, Z.-F. Han, and J. Chambers, "Analysis simco algorithms for sparse analysis model based dictionary learning," *IEEE Transactions on Signal Processing*, vol. 64, no. 2, pp. 417–431, 2015.

[65] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Transactions on Signal Processing*, vol. 61, no. 9, pp. 2341–2355, 2013.

[66] S. Nam, M. E. Davies, M. Elad, and R. Gribonval, "The cosparse analysis model and algorithms," *arXiv preprint arXiv:1106.4987*, 2011.

[67] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, 2012.

[68] L. Pfister and Y. Bresler, "Learning filter bank sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 67, no. 2, pp. 504–519, 2018.

[69] M. Kiechle, T. Habigt, S. Hawe, and M. Kleinsteuber, "A bimodal co-sparse analysis model for image processing," *International Journal of Computer Vision*, vol. 114, no. 2-3, pp. 233–247, 2015.

[70] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[71] V. Papyan, J. Sulam, and M. Elad, "Working locally thinking globally: Theoretical guarantees for convolutional sparse coding," *IEEE Transactions on Signal Processing*, vol. 65, no. 21, pp. 5687–5701, 2017.

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

[73] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 391–398.

[74] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5135–5143.

[75] B. Wohlberg, "Efficient convolutional sparse coding," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 7173–7177.

[76] I. Y. Chun and J. A. Fessler, "Convolutional dictionary learning: Acceleration and convergence," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1697–1712, 2017.

[77] ——, "Convolutional analysis operator learning: Acceleration, convergence, application, and neural networks," *arXiv preprint arXiv:1802.05584*, 2018.

[78] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, 1990, pp. 396–404.

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

[80] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

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

[82] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[83] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision.* Springer, 2014, pp. 818–833.

[84] N. Lei, Z. Luo, S.-T. Yau, and D. X. Gu, "Geometric understanding of deep learning," *arXiv preprint arXiv:1805.10451*, 2018.

[85] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 2924–2932.

[86] R. Giryes, G. Sapiro, and A. M. Bronstein, "Deep neural networks with random Gaussian weights: A universal classification strategy?" *IEEE Transactions Signal Processing*, vol. 64, no. 13, pp. 3444–3457, 2016.

[87] M. Elad, "Sparse and redundant representations: From theory to applications in signal and image processing," *Springer*, 2010.

[88] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[89] J. Sulam, V. Papyan, Y. Romano, and M. Elad, "Multi-layer convolutional sparse modeling: Pursuit and dictionary learning," *arXiv preprint arXiv:1708.08705*, 2017.

[90] S. Tariyal, A. Majumdar, R. Singh, and M. Vatsa, "Deep dictionary learning," *IEEE Access*, vol. 4, pp. 10 096–10 109, 2016.

[91] S. Mahdizadehaghdam, A. Panahi, H. Krim, and L. Dai, "Deep dictionary learning: A parametric network approach," *arXiv preprint arXiv:1803.04022*, 2018.

[92] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds.* Princeton University Press, 2009.

[93] M. Elad, "Why simple shrinkage is still relevant for redundant representations?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5559–5569, 2006.

[94] M. Raphan and E. P. Simoncelli, "Optimal denoising in redundant representations," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1342–1352, 2008.

[95] Y. Lin and D. D. Lee, "Bayesian l 1-norm sparse learning," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 5. IEEE, 2006, pp. V–V.

[96] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[97] J.-F. Cai, H. Ji, Z. Shen, and G.-B. Ye, "Data-driven tight frame construction and image denoising," *Applied and Computational Harmonic Analysis*, vol. 37, no. 1, pp. 89–105, 2014.

[98] B. Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 301–315, 2015.

[99] V. Papyan and M. Elad, "Multi-scale patch-based image restoration," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 249–261, 2016.

[100] A. Aberdam, J. Sulam, and M. Elad, "Multi-layer sparse coding: The holistic way," *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 1, pp. 46–77, 2019.

[101] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[102] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, 2012.

[103] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, "Learning convolutional feature hierarchies for visual recognition," in *Advances in Neural Information Processing Systems*, 2010, pp. 1090–1098.

[104] I. Y. Chun and J. A. Fessler, "Convolutional analysis operator learning: Acceleration, convergence, application, and neural networks," *arXiv preprint arXiv:1802.05584*, 2018.

[105] V. Papyan, Y. Romano, J. Sulam, and M. Elad, "Theoretical foundations of deep learning via sparse representations: A multilayer sparse model and its connection to convolutional neural networks," *IEEE Signal Processing Magazine*, vol. 35, no. 4, pp. 72–89, 2018.

[106] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[107] J.-J. Huang and P. L. Dragotti, "Learning deep analysis dictionary and thresholding models-part I: Unstructured dictionary," *submitted to IEEE Transactions on Signal Processing*, 2019.

[108] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Advances in Neural Information Processing Systems*, 2018, pp. 6389–6399.

[109] C. Bailer, B. Taetz, and D. Stricker, "Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4015–4023.

[110] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *European Conference on Computer Vision*. Springer, 1994, pp. 151–158.

[111] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[112] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "Brief: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2012.

[113] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE conference on.* Ieee, 2012, pp. 510–517.

[114] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2011, pp. 2564–2571.

[115] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[116] J. Ma, J. Zhao, J. Tian, X. Bai, and Z. Tu, "Regularized vector field learning with sparse approximation for mismatch removal," *Pattern Recognition*, vol. 46, no. 12, pp. 3519–3532, 2013.

[117] J. Ma, J. Zhao, J. Tian, A. L. Yuille, and Z. Tu, "Robust point matching via vector field consensus," *IEEE Transactions on Image Processing*, vol. 23, no. 4, pp. 1706–1721, 2014.

[118] C. A. Micchelli and M. Pontil, "On learning vector-valued functions," *Neural computation*, vol. 17, no. 1, pp. 177–204, 2005.

[119] G. Donato and S. Belongie, "Approximate thin plate spline mappings," in *European Conference on Computer Vision.* Springer, 2002, pp. 21–31.

[120] F. Pitié, A. C. Kokaram, and R. Dahyot, "Automated colour grading using colour distribution transfer," *Computer Vision and Image Understanding*, vol. 107, no. 1, pp. 123–137, 2007.

[121] O. Frigo, N. Sabater, V. Demoulin, and P. Hellier, "Optimal transportation for example-guided color transfer," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 655–670.

[122] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski, "Non-rigid dense correspondence with applications for image enhancement," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 70, 2011.