INTERPRETABLE DEEP IMAGE DENOISER BY UNROLLING GRAPH LAPLACIAN REGULARIZER

SEYED ALIREZA HOSSEINI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

AUGUST 2025

Abstract

Image denoising is a fundamental problem in image restoration. Researchers have studied the problem for decades and proposed numerous algorithms. In the past ten years, deep learning has produced complex models that deliver high-quality denoised images, but these models require on large numbers of parameters, lack interpretability, and depend heavily on random parameter initialization. As a result, they frequently converge to poor-performing local minima.

This thesis proposes an image denoising neural net constructed by unrolling an iterative algorithm solving a maximum a posteriori (MAP) optimization problem regularized using a graph Laplacian prior. To guarantee a minimum level of performance, we initialize the network to a known (pseudo-)linear denoiser, which is mapped to a corresponding graph Laplacian matrix specifying the MAP problem, leveraging a previous linear algebraic theorem. The performance of the network is further enhanced by learning an appropriate perturbation matrix to augment the graph Laplacian via a lightweight convolutional neural net (CNN). This design bridges the gap between classical model-based methods with modern deep learning, eliminates the need for random initialization, reduces parameter count, and improves interpretability of the constructed network. Experiments show that our method demonstrates competitive image quality reconstruction compared to state-of-the-art deep learning models, while offering improved robustness, interpretability, and parameter efficiency.

Dedication

To all of us in the 2SLGBTQI+ Iranian community,
silenced and persecuted by a state that denies our existence,
yet steadfast in love, courage and hope.

In memory of Alireza Fazeli Monfared,[1]

[1]At twenty, Alireza Fazeli Monfared's life was taken for the simple act of living authentically.

Acknowledgements

This thesis could not have been completed without the support, feedback, and encouragement I received throughout my academic journey.

I would like to sincerely thank Dr. Gene Cheung for his guidance and support throughout my master's studies. His input and mentorship contributed to the completion of this work.

I would also like to thank my colleagues at the GISP Lab for the collaborative environment, valuable discussions, and their willingness to share knowledge. Their contributions enhanced both my research and learning experience.

I am deeply grateful to my parents for their unconditional love, sacrifices, and constant support. Their belief in me has been a steady source of strength.

I want to thank my partner for his presence, encouragement, and understanding, which have helped me stay grounded through every challenge. This journey would have been much harder without him.

I am also grateful to everyone who helped me along the way with their support, advice, or encouragement.

Table of Contents

List of Tables

List of Figures

With them the Seed of Wisdom did I sow,
And with my own hand labour'd it to grow:
And this was all the Harvest that I reap'd –
"I came like Water and like Wind I go."

— Omar Khayyam, an 11th-century Iranian poet and mathematician
Translated by Edward FitzGerald.

Chapter 1

Introduction

1.1 Background and Motivation

In the field of image processing and computational imaging, the task of *restoration* is to recover the original pristine image from an observed distorted one. Although the basic goal of denoising is to reduce additive noise without destroying image details, the significance of denoising goes beyond this singularly defined task. Specifically, an effective denoiser can be employed as a building block in a larger image restoration framework. One example is *Regularization by Denoising* (RED) [1], where a denoiser is used to define a more general regularizer for a targeted inverse imaging problem. Another example is *Plug-and-Play* (PnP) [2], which formulates a denoising subproblem within a larger image restoration problem—specifically those solved using the *Alternating Direction Method of Multipliers* (ADMM) framework [3]. Diffusion models can also incorporate denoisers into their generative forward process to produce meaningful images from a latent vector, typically modeled as a random Gaussian distribution.

1.2 Types of Image Noise and Focus of This Work

Noise in digital images refers to random or structured variations in pixel intensities that do not correspond to the true underlying scene. These variations may arise due to imperfections in the imaging sensor, electronic circuitry, or the transmission and storage of image data. Understanding the characteristics and mathematical models of different types of noise is crucial for designing effective denoising algorithms. Below, we review the most common types of noise encountered in image processing.

1.2.1 Additive White Gaussian Noise (AWGN)

Additive White Gaussian Noise (AWGN) is by far the most widely studied and utilized noise model in image denoising literature. It arises from various sources such as thermal noise in

image sensors, readout circuitry, and analog-to-digital conversion.

Mathematically, AWGN is modeled as

$$\mathbf{y} = \mathbf{x} + \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) \tag{1.1}$$

where $\mathbf{x}$ is the clean image, $\mathbf{y}$ is the observed noisy image, and $\mathbf{n}$ is a zero-mean Gaussian noise vector with covariance matrix $\sigma^2 \mathbf{I}$. The noise is assumed to be i.i.d. (independent and identically distributed) across pixels. AWGN is simple to simulate and analyze, making it the de facto standard for benchmarking denoising algorithms.

1.2.2 Impulse Noise (Salt-and-Pepper Noise)

Impulse noise appears as random occurrences of black and white pixels in the image and is often caused by malfunctioning pixel elements in camera sensors, faulty memory locations, or transmission errors in digital communication.

The corrupted image can be modeled as

$$y(i,j) = \begin{cases} x(i,j), & \text{with probability } 1-p \\ I_{\min}, & \text{with probability } \frac{p}{2} \\ I_{\max}, & \text{with probability } \frac{p}{2} \end{cases} \tag{1.2}$$

where p is the noise density, and $I_{\min}$ and $I_{\max}$ represent the minimum and maximum intensity values (e.g., 0 and 255 for 8-bit images). The non-Gaussian and non-additive nature of impulse noise renders it resistant to traditional filtering-based denoising methods.

1.2.3 Multiplicative Noise (Speckle)

Multiplicative noise, commonly known as speckle noise, is prevalent in coherent imaging systems such as ultrasound, laser, and Synthetic Aperture Radar (SAR). This type of noise arises due to interference between coherent wavefronts reflected from different surface points.

The noisy observation is modeled as

$$\mathbf{y} = \mathbf{x} \circ \mathbf{n}, \quad \mathbf{n} \sim \text{Gamma}(L, 1/L) \tag{1.3}$$

where $\circ$ denotes element-wise multiplication, and $\mathbf{n}$ is a multiplicative noise vector, often modeled using a Gamma distribution with mean 1 and variance $1/L$, where L represents the number of looks (a parameter controlling noise severity). The signal-dependent nature of multiplicative noise complicates the denoising task.

1.2.4 Structured Noise (e.g., Periodic or Stripe Noise)

Structured noise includes any noise pattern that exhibits regularity or spatial correlation. This could result from sensor defects, interference from power sources, or mechanical vibrations in scanning equipment. Examples include stripe noise, fixed-pattern noise, or periodic artifacts.

There is no universal model for structured noise, but one example specification is

$$\mathbf{y} = \mathbf{x} + \mathbf{n}_{\text{str}}, \quad \mathbf{n}_{\text{str}} = A\sin(2\pi f x + \phi) \tag{1.4}$$

where A, f, and ϕ are amplitude, frequency, and phase of a sinusoidal interference pattern. Removal of structured noise often requires spectral analysis or spatial-domain techniques that exploit the periodicity or correlation in the noise.

1.2.5 Poisson Noise (Shot Noise)

Poisson noise, also known as shot noise, is inherent in photon-limited imaging systems such as astronomy, medical imaging (e.g., PET), and low-light photography. It originates from the quantized nature of light and is signal-dependent.

The noisy observation follows

$$y(i,j) \sim \text{Poisson}(x(i,j)) \tag{1.5}$$

Unlike Gaussian noise, Poisson noise has a variance equal to its mean, i.e., $\text{Var}(y(i,j)) = x(i,j)$. This makes denoising more difficult, particularly in low-light conditions where signal levels are small and noise dominates. Variance-stabilizing transforms such as the Anscombe transform are often used to approximate Poisson noise as Gaussian.

Focus of This Work

In this thesis, we focus exclusively on the problem of denoising images corrupted by Additive White Gaussian Noise (AWGN), for the following reasons:

- AWGN is the most prevalent model used in benchmarking denoising algorithms, allowing for consistent and fair performance evaluation.
- It serves as a foundational testbed for developing and analyzing new denoising architectures, especially those based on iterative optimization and deep learning.
- Many techniques developed under the AWGN model can be generalized to handle more complex or realistic noise models with appropriate modifications.

Hence, this work adopts AWGN as the primary noise model, while acknowledging that future extensions may adapt the proposed methods to other types of noise.

1.2.6 Natural Noise

While much of the classical image denoising literature is built around synthetic noise models such as additive Gaussian noise, real-world noise observed in naturally acquired images often exhibits far more complex behavior. This type of noise, commonly referred to as *natural*

noise or *real image noise*, arises from a combination of physical processes in digital imaging pipelines, including photon shot noise, thermal readout noise, quantization errors, sensor-specific defects, and nonlinear transformations such as gamma correction and demosaicing. Unlike its synthetic counterparts, natural noise is not independent of the image content; it is typically signal-dependent, spatially correlated, non-stationary across the image, and highly dependent on the sensor characteristics and acquisition settings.

There is no universal analytical expression for modeling natural noise. Instead, the noisy image is often described as the output of a complex, nonlinear function that maps a clean image and the corresponding noise to the observed measurement. Formally, this can be expressed as $\mathbf{y} = f(\mathbf{x}, \mathbf{n})$, where $\mathbf{x}$ is the clean image, $\mathbf{n}$ represents the latent noise sources, and f encapsulates the full image formation pipeline of the camera, including both deterministic and stochastic elements. Because f is typically unknown and device-dependent, it is challenging to derive principled denoising algorithms in this setting.

To study and benchmark denoising algorithms under natural noise conditions, several datasets have been proposed in recent years. Among them, the Darmstadt Noise Dataset (DND), the Smartphone Image Denoising Dataset (SIDD), and the Nam dataset are widely used. These datasets provide pairs of real noisy and nearly noise-free images captured using professional or consumer-grade cameras under controlled conditions. Clean reference images are often generated by averaging multiple low-ISO or long-exposure frames, thereby minimizing the noise without introducing additional artifacts.

Denoising natural images presents several unique challenges. Since the noise is no longer additive, white, or Gaussian, models trained on synthetic data often fail to generalize. Additionally, because the noise distribution varies across scenes, lighting conditions, and devices, even small shifts in image acquisition parameters can significantly degrade performance. Unlike the AWGN setting, where pixel-level loss functions are often sufficient, natural noise scenarios may require more sophisticated training objectives or domain adaptation techniques to maintain robustness.

In this thesis, the focus remains on additive Gaussian noise, due to its well-established role as a canonical benchmark in the field and its analytical tractability for theoretical analysis. However, it is worth noting that the denoising framework proposed in this work, which leverages graph-based priors and interpretable deep unrolling, may offer extensibility to real-world settings where natural noise is present. In particular, the modularity of the approach and its reliance on structural assumptions rather than dataset-specific optimization may be advantageous when generalizing beyond the synthetic setting.

1.3 Overview of Denoising Methods

One approach to the denoising problem is *"model-based"*, which assumes mathematical models using which algorithms are derived. Examples include simple averaging filters, where the averaging weights are based on a uniform distribution, and Gaussian filter, where the averaging weights follow a Gaussian distribution. A Gaussian filter and an averaging filter smooth over

any signal using the same assumed distribution of weights. However, one can use different weights to smooth different signals, taking the signal structure into account. This approach helps prevent the loss of salient image details like edges while suppressing noise by averaging. *Bilateral filter* (BF) [4] and *Non-local Means* (NLM) [5] employ this strategy to preserve image details while reducing additive noise effectively.

Many model-based algorithms are also rooted in Bayesian inference, where they employ signal priors to regularize the ill-posed denoising problem. Examples of such priors include *total variation* (TV) [6], sparsity-promoting prior [7], low-rank prior [8], self-similarity prior [9] and autoregressive prior [10].

1.4 Strengths and Limitations of Existing Approaches

In the early 2000s, substantial research in image denoising led to the development of highly effective model-based denoisers. This progress prompted some researchers to consider the possibility that performance might have reached its practical limits, leading to the belief that "Denoising is Dead" [11]. However, the emergence of deep learning over the past decade has brought significant advancements in various fields, particularly in image processing and denoising. These developments have introduced new techniques for designing more effective deep-learning-based denoisers. Today, deep learning denoisers achieve high levels of performance in noise suppression, outperforming traditional model-based methods.

To understand why deep learning-based methods perform better, we need to explore their underlying methodologies. Deep learning-based denoisers are *data-driven* methods that learn a mapping function from noisy images to clean images by minimizing a loss function on a training set containing pairs of noisy and clean images. These denoisers generally outperform model-based denoisers in almost all cases due to scaling—using larger datasets and more complex architectures. One might think that scaling can endlessly improve performance, but it comes with certain costs.

Although deep-learning-based denoisers are highly effective, the learned mappings can be extremely complex. This complexity is typically measured in terms of the number of learned parameters in a mapping. If the complexity is high, several challenges arise during the training process. First, the amount of data required to train such a denoiser would be large. Second, as the complexity of a deep-learning-based denoiser increases, it tends to *memorize* specific patterns in the training data instead of *learning* generalizable patterns. This causes the model to perform extremely well on the training data but poorly on new data it has not encountered before—a phenomenon known as *overfitting*.

In overfitting, the model memorizes specific patterns in the training data, including the covariate distribution of training examples. As a result, a large model may also perform poorly when it encounters data with different covariate distributions—a phenomenon known as *covariate shift* [12].

Deep learning models can map noisy inputs to clean outputs, but this process is often a black box. There is no clear mathematical intuition or explanation for why these models can

reduce noise and produce clean images. This lack of interpretability brings its own challenges. For example, interpretability can guide the creation of compact models with fewer parameters that still perform well, making them less vulnerable to overfitting and covariate shifts.

Further, network parameter initialization plays a crucial role in the performance of the final learned parameters during stochastic gradient descent [13]. Starting with a "bad" initialization can lead to getting stuck in poor local minimum, resulting in performance degradation. On the other hand, good initialization can reduce training time and allow the model to converge to an effective solution more quickly. Recently, several initialization schemes, such as Xavier initialization [14], have been proposed to address this issue.

1.5 Proposed Approach: A Hybrid Graph-based Denoising Network

Recently, an approach called *model-based deep learning* [15] has been introduced. This method combines the interpretability of model-based denoisers with the strong performance of deep learning-based denoisers. As the name suggests, it involves designing a deep-learning-based denoiser inspired by a model-based algorithm. Model-based algorithms often work through iterations, where the output of one step becomes the input for the next. By stacking these iterations like layers in a deep learning architecture, we can build a deep neural network that mimics the iterative process of the algorithm. This approach is known as *algorithm unrolling*.

Inspired by recent work in algorithm unrolling [15], in this thesis, we propose a novel image denoising neural net constructed by unrolling a graph-based algorithm, one that can be initialized to a known well-performing denoiser to guarantee a baseline level of performance. Specifically, a recent linear algebraic theorem in [16] shows that, under mild conditions, a (pseudo-)linear denoiser $\mathbf{\Psi}$ can be mapped to a graph filter that is a solution to a *maximum a posteriori* (MAP) optimization problem regularized using a graph smoothness prior [17] specified by graph Laplacian matrix $\mathbf{L}$. Leveraging this theorem, we first construct a neural network by unrolling an iterative algorithm solving the graph-regularized MAP problem specified by $\mathbf{L}$, where $\mathbf{L}$ is mapped from $\mathbf{\Psi}$ via the linear algbraic theorem. This ensures that the network performs on par to $\mathbf{\Psi}$ at a minimum. Then, we learn an appropriate perturbation matrix $\mathbf{P}$ to augment graph Laplacian $\mathbf{L}$ to $\mathbf{L} + \mathbf{P}$, via a lightweight convolutional neural net (CNN) in a data-driven manner to further improve performance.

1.6 Contributions of This Thesis

This design bridges the gap between classical model-based methods with modern deep learning, eliminates the need for random initialization, reduces parameter count, and improves interpretability of the constructed network. Simulation experiments show that our constructed network demonstrates competitive image quality reconstruction compared to state-of-the-art deep learning models such as DnCNN [18], while offering improved robustness, interpretability, and parameter efficiency.

1.7 Thesis Organization

The document is organized as follows. In Chapter 2, we formally define the denoising problem and provide a detailed discussion of model-based and deep learning-based approaches. This chapter also covers algorithm unrolling. In Chapter 3, we introduce key concepts and definitions in graph signal processing (GSP) [19], which form the foundation of the proposed method. In Chapter 4, we present our proposed image denoising network design and optimization. Finally, experimental results and conclusion are presented in Chapter 5 and 6, respectively.

Chapter 2

Related Works

2.1 Problem Definition

We begin by formally defining the denoising problem. Consider a target image represented as an $N \times N$ matrix, which is vectorized lexicographically into a vector $\mathbf{x} \in \mathbb{R}^{N^2}$. An image formation model for an observed noisy image $\mathbf{y} \in \mathbb{R}^{N^2}$ is

$$\mathbf{y} = \mathbf{x} + \mathbf{v}, \tag{2.1}$$

where $\mathbf{v} \in \mathbb{R}^{N^2}$ is a zero-mean i.i.d. Gaussian noise term, $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, and σ^2 is the noise variance. The goal of denoising is to recover $\mathbf{x}$ from $\mathbf{y}$. This can be done either with or without prior knowledge of σ^2.

A large portion of image denoising research has focused on *additive white Gaussian noise* (AWGN). However, other noise types like Poisson [20] and salt-and-pepper [21] have also been studied.

2.2 Model-based Image Restoration

Model-based methods assume mathematical models of the target signal, using which restoration algorithms are derived or designed. Specifically, for the image restoration problem, there are two main approaches in model-based schemes. The first approach is Bayesian inference, which aims to recover the image by minimizing an energy function that incorporates the noisy image and an assumed prior probability distribution. The second approach is based on direct filter designs to average image pixel values. We will review both approaches in detail in the following section.

2.2.1 Bilateral Filter

Uniform averaging of neighboring pixels results in blurring of important image details, such as object contours. To avoid this, the structure of the image itself should be considered when

applying averaging. *Signal-dependent* weighted averaging takes this structure into account, thereby limiting the loss of image details while suppressing noise by averaging only among "similar" samples in the signal. Specifically, in *bilateral filter* (BF) [4], similarity is defined in two ways: proximity between pixel physical locations in the image, and proximity between pixel values.

Mathematically, intensity x_j of each pixel j in the image is computed as a BF-weighted combination of intensity x_i of pixels i:

$$x_j = \sum_i \frac{w_{i,j}}{\sum_i w_{i,j}} x_i. \tag{2.2}$$

The BF weight $w_{i,j}$ between the target pixel i and its neighboring pixel j is defined as

$$w_{i,j} = \exp\left(-\frac{\|\mathbf{l}_i - \mathbf{l}_j\|_2^2}{\sigma_l^2}\right) \exp\left(-\frac{|x_i - x_j|^2}{\sigma_x^2}\right) \tag{2.3}$$

where $\mathbf{l}_i \in \mathbb{R}^2$ and $x_i \in \mathbb{R}$ denote the 2D grid location and intensity of pixel i, respectively.

BF is an example of a *pseudo-linear* denoiser; the weight matrix $\mathbf{W} = [w_{i,j}]_{N\times N}$ depend on the input signal itself and thus nonlinear. However, once $\mathbf{W}$ is fixed for a given signal, it becomes a linear operator on signal $\mathbf{x}$.

2.2.2 Sparsity Prior

One popular signal prior is a sparsity prior [7], which assumes that a target image patch $\mathbf{x}$ can be expressed as a sparse linear combination of atoms from an overcomplete dictionary $\mathbf{D}$, such that $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the code vector and $\|\boldsymbol{\alpha}\|_0$ is small. There are several methods to compute such a dictionary $\mathbf{D}$ from data, a process called *dictionary learning* [22].

Recall that in equation (2.1), we defined the problem for a single noisy image patch. This formulation can be extended to a set of image patches. Specifically, we stack m noisy patches as columns in a matrix $\mathbf{X} \in \mathbb{R}^{N\times m}$, where each column $\mathbf{x}_i$ in $\mathbf{X}$ represents a single noisy (vectorized) image patch. The problem of finding an overcomplete dictionary $\mathbf{D} \in \mathbb{R}^{N\times p}$, where $p > N$, to represent $\mathbf{X}$ sparsely can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{D},\mathbf{A}} \quad & \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 \\ \text{s.t.} \quad & \|\mathbf{a}_l\|_0 \le s, \quad l \in \{1, \dots, m\} \end{aligned}. \tag{2.4}$$

The variables are dictionary $\mathbf{D} \in \mathbb{R}^{N\times p}$ and a sparse representation matrix $\mathbf{A} \in \mathbb{R}^{p\times m}$, where each column l (represented by $\mathbf{a}_l$) has at most s non-zero elements. s is a parameter that specifies the sparsity count in each $\mathbf{a}_l$. The dictionary learning optimization problem in (2.4) is inherently difficult to solve, first due to its non-convexity caused by the ℓ_0 norm, and second because of the sparsity constraint, which make it NP-hard. Further, optimizing $\mathbf{D}$ and $\mathbf{A}$ at the same time makes (2.4) non-convex because of their multiplicative relationship.

We solve (2.4) in an alternating manner by addressing two sub-problems. The first sub-problem, known as *"dictionary learning"*, involves finding a dictionary $\mathbf{D}$ given a fixed

sparse representation $\mathbf{A}$. The second sub-problem focuses on finding a sparse representation $\mathbf{A}$ for a fixed dictionary $\mathbf{D}$.

There are many solutions to the dictionary learning problem, such as K-SVD [23]. The second sub-problem, often called *"sparse coding"*, assumes the dictionary $\mathbf{D}$ is known, and computes a sparse representation $\mathbf{A}$. Finding an optimal solution $\mathbf{A}$ for the sparse coding problem is NP-hard in the general case, but there are different algorithms that provide approximate solutions. One approach is to use greedy algorithms, such as *Orthogonal Matching Pursuit* (OMP) [24] or *Matching Pursuit* (MP) [25], which attempts to build sparse representation by selecting one atom at a time based on a local objective.

Another approach, called *Convex Relaxation*, replaces the pseudo-norm ℓ_0 with the convex ℓ_1 norm, considered the closest convex approximation of the ℓ_0 norm. This leads to the *Basis Pursuit* (BP) problem, which is known as Lasso [26] in statistics.

Specifically, for a given dictionary $\mathbf{D}$ and an input image patch $\mathbf{y}$, BP is the MAP problem with a convex ℓ_1-norm prior for code vector $\boldsymbol{\alpha}$ that promotes sparsity, formulated as

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{y} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1. \tag{2.5}$$

Solving (2.5) can be computationally expensive, since the ℓ_1-norm is not differentiable and thus there is no closed-form solution.

2.2.3 Total Variation Prior

Total variation (TV) [6] of an image can be defined as a measure of signal variation across a data kernel. There are various approaches to define a discretized [27] TV operator for digital images. The most common definition of total variation for digital images is as follows:

$$\mathrm{TV}(\mathbf{x}) = \sum_i \sum_{j \in \mathcal{N}_i} \|x_i - x_j\|_p^q \tag{2.6}$$

where $\mathcal{N}_i$ denotes a neighborhood centered at the i-th pixel (usually the horizontally and vertically adjacent pixels), and $\|\cdot\|_p^q$ represents the ℓ_p norm raised to the power of q.

Using the operator $\mathrm{TV}(\cdot)$ as a prior, the MAP-based denoising problem can be defined as follows:

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda \mathrm{TV}(\mathbf{x}). \tag{2.7}$$

Common choices for $\|\cdot\|_p$ are the ℓ_1- and ℓ_2-norms. As a prior, TV is popular due to its simplicity and effectiveness. However, it has some drawbacks. First, the ℓ_1-norm is non-differentiable and has no closed-form solution [28]. Second, resulting image textures tend to be over-smoothed. Third, TV approximates a slowly spatially-varying area with a piecewise constant signal, resulting in an unnatural staircase-like image with false gradients—known as the "staircase" effect.

2.2.4 Low-rank Prior

Consider stacking m similar ground truth image patches of N pixels each as columns in matrix $\mathbf{X} \in \mathbb{R}^{N \times m}$. Suppose $\mathbf{X}$ is corrupted by noise matrix $\mathbf{V}$, which we assume to be large but sparse, resulting in observation matrix $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{X} + \mathbf{V}. \tag{2.8}$$

In the *robust principal component analysis* (RPCA) framework [29], the denoising problem for a set of similar observed patches $\mathbf{Y}$ can be defined as

$$\min_{\mathbf{X},\mathbf{V}} \quad \|\mathbf{X}\|_* + \lambda \|\mathbf{V}\|_1, \qquad \text{s.t.} \quad \mathbf{Y} = \mathbf{X} + \mathbf{V} \tag{2.9}$$

where $\|\mathbf{X}\|_*$ denotes the *nuclear norm*—the sum of singular values (which are by definition non-negative) of matrix $\mathbf{X}$:

$$\|\mathbf{X}\|_* = \sum_i \sigma_i(\mathbf{X}), \tag{2.10}$$

where $\sigma_i(\mathbf{X})$ is the i-th singular value of $\mathbf{X}$. Nuclear norm is a convexification of matrix rank.

A solution to (2.9) can be computed via soft-thresholding of singular values of $\mathbf{Y}$, computed using *singular value decomposition* (SVD) [30]:

$$\hat{\mathbf{X}} = \mathbf{U} S_\lambda(\mathbf{\Sigma}) \mathbf{V}^\top \tag{2.11}$$

where $\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ is SVD of $\mathbf{Y}$ and $S_\lambda(\cdot)$ is a soft-thresholding operator defined as follows:

$$S_\lambda(\sigma_i) = \begin{cases} \sigma_i - \lambda & \text{if} \quad \sigma_i - \lambda \geq 0, \\ \sigma_i + \lambda & \text{if} \quad \sigma_i + \lambda \leq 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

where σ_i is the i-th singular value of $\mathbf{Y}$.

The low-rank prior provides excellent performance; however, computational complexity remains an issue, since SVD has complexity $\mathcal{O}(N^3)$.

2.2.5 Self-similarity Prior

As the name suggests, the self-similarity prior [31] focuses on the similarity that the image has within itself. Natural images often exhibit self-similarity characteristics: similar pixel patches tend to recur non-locally throughout an image. More precisely, let $\mathbf{R}_j$ be an operator that extracts a patch centered at the j-th pixel. Further, let $\Omega(k)$ be a collection of pixels where similar patches to $\mathbf{R}_k$ are centered. Additionally, consider a distance metric $d(\cdot)$, which could be the ℓ_1- or ℓ_2-norm. With these definitions, the self-similarity prior can be expressed as:

$$\sum_k \sum_{j \in \Omega(k)} d\{\mathbf{R}_k \mathbf{x}, \mathbf{R}_j \mathbf{x}\}. \tag{2.13}$$

By fixing the distance metric, the MAP restoration problem can be defined as

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda \sum_k \sum_{j \in \Omega(k)} d\{\mathbf{R}_k \mathbf{x}, \mathbf{R}_j \mathbf{x}\}. \tag{2.14}$$

One can solve (2.14); however, the complexity of identifying non-locally located similar patches $\Omega(k)$ to each patch centered at k can be prohibitive.

2.2.6 BM3D

BM3D [32] is one of the top-performing model-based image denoising algorithms. Specifically, BM3D first performs block-matching to identify similar patches non-locally in an image to compose a 3D cube with layers of similar patches. Then, it transforms the entire cube into the 3D-DCT domain and performs collaborative filtering for denoising. Finally, it converts the cube back to the pixel domain for weighted averaging of different filtered outputs.

Although BM3D achieves excellent performance among model-based image denoising algorithms, it has two main shortcomings. First, the entire process of block matching, collaborative filtering, and aggregation is computationally intensive. Second, it struggles to handle complex textures in an image effectively.

2.3 Deep Learning Based Denoisers

We have overviewed model-based methods for image denoising. Now, we turn to designs of deep-learning-based image denoisers. Although modern image restoration models vary significantly in their architectures—utilizing different building blocks like CNNs [18] or Transformers [33, 34]—if we set aside these architectural differences, they all follow a common underlying principle.

Supervised deep learning-based methods are inherently data-driven, meaning that they rely on a labeled training dataset, denoted as $\mathcal{A}_n$. The labeled training dataset can be defined as follows:

$$\mathcal{A}_n = ((\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n)) \in (\mathcal{X} \times \mathcal{Y})^n \tag{2.15}$$

Here, $\mathcal{X} = \{\mathbf{x}_k\}_{k=1}^n$ represents the set of clean images, and $\mathcal{Y} = \{\mathbf{y}_k\}_{k=1}^n$ is the corresponding set of noisy images for each clean image. Specifically, $\mathbf{y}_k = \mathbf{x}_k + \mathbf{v}_k$, where $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$, since we are considering Gaussian noise.

Next, we define a parametric denoising model as $\hat{\mathbf{x}} = \mathcal{D}_\theta(\mathbf{y}, \sigma)$, where θ represents the set of trainable parameters that need to be optimized using the training dataset $\mathcal{A}_n$. There is no universal method to determine the best architecture for designing such a denoiser. Most researchers rely on trial and error to discover effective architectures, and there is limited theoretical justification for why a specific architecture works. In other words, these denoisers often lack interpretability.

The other consideration in designing such denoisers is determining *in what sense the trainable parameters can be optimized?* This involves defining a *loss function,* which imposes

a penalty to be minimized with respect to the learnable parameters θ, encouraging the output $\hat{\mathbf{x}}$ to resemble the clean image $\mathbf{x}$. There are various choices for this loss function, including the ℓ_1- and ℓ_2-norms, with the most commonly used being the ℓ_2-norm, *i.e.*, the mean squared error.

The final step is minimizing the loss function with respect to θ using *stochastic gradient descent* (SGD) [35] on small batches of the training set $(\mathbf{x}_i, \mathbf{y}_i)$. This involves backpropagating through the network and updating the parameters θ accordingly.

Next, we will discuss several representative deep-learning-based denoisers. In terms of convolutional neural networks, the most well-known architecture is DnCNN [18]. DnCNN learns the *residual mapping* instead of the latent clean image, where the *residual mapping* represents the difference between the noisy input and the latent clean image. In this case, the denoiser output is given by $\mathbf{y} = \mathbf{x} - R(\mathbf{x})$, where $R(\cdot)$ is a neural network that predicts the residual mapping. The DnCNN architecture consists of 17 layers, each containing a convolutional block, followed by batch normalization and a ReLU activation function.

With the emergence of the transformer architecture [36], researchers in computer vision and image processing explored its potential for learning tasks by dividing an image into smaller patches and treating these patches as tokens mapped into an embedding space [37]. In the context of denoising, this approach led to specialized architectures such as Restormer [34] and SwinIR [33].

In Restormer, the input noisy image first passes through a convolutional layer to extract a set of shallow features. These features are then processed through an encoder-decoder architecture, where each upsampling and downsampling block is itself a transformer block. Finally, the input noisy image is added to the output, following a residual learning strategy similar to the one discussed earlier.

2.4 Algorithm Unrolling

To address the limitations discussed in the previous section, one proposed solution is *model-based deep learning* [15]. While deep learning-based methods are purely data-driven, model-based approaches, as previously mentioned, rely on classical statistical modeling techniques. *Model-based deep learning* seeks to combine these two strategies, allowing us to leverage the strengths of both approaches simultaneously.

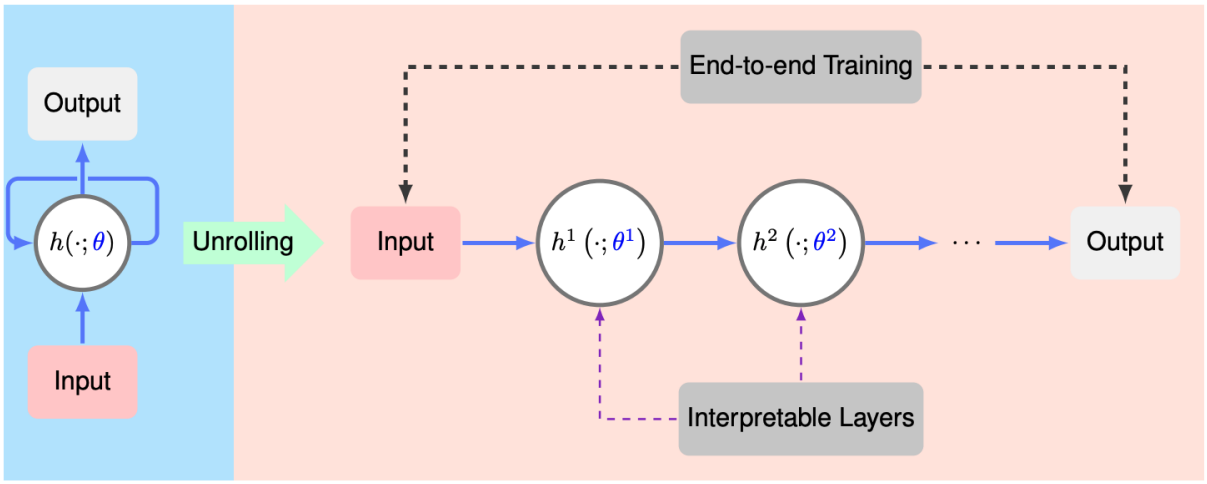


Figure 2.1: An overview of algorithm unrolling from [38]: A classical model-based algorithm $h(\cdot, \theta)$, which involves several iterations and parameters typically computed analytically (left), can be unrolled into a feed-forward neural network by stacking its iterations and interpreting each iteration as an interpretable neural layer. Moreover, the parameters in each iteration can be learned through backpropagation.

As mentioned earlier, selecting an appropriate architecture for a deep learning-based denoiser $\mathcal{D}_\theta(\mathbf{y}, \sigma)$ is often done in a blind manner—stacking simple building blocks and using trial and error until a well-performing denoiser is obtained. In contrast, model-based deep learning designs the deep learning architecture to resemble a model-based algorithm. Since classical model-based algorithms typically involve sequential steps, or iterations, this approach is often referred to as *"algorithm unrolling."* In this context, *"unrolling"* refers to unfolding each iteration into a neural layer to compose an interpretable feed-forward neural network. Given the constructed network, we can learn parameters end-to-end in a data-driven manner. See [38] for a detailed discussion.

Chapter 3

Preliminaries

We first review definitions and fundamental concepts in *graph signal processing* (GSP) [19]. We discuss a key theorem in [16] that we leverage heavily in our work. Finally, we review the concepts of *kernel functions* [39], which we develop in the sequel.

3.1 GSP Definitions

While traditional *digital signal processing* (DSP) focuses on signals residing on regular discrete kernels (*e.g.*, regularly sampled time line for audio, or 2D grid for images), there has been growing interest in processing signals on irregular kernels described by graphs, leading to the emergence of thed *Graph Signal Processing* (GSP) field [19]. GSP has garnered significant attention recently, as many core DSP concepts, such as sampling, Fourier transform, and filtering, have been successfully extended to the graph domain. GSP has been applied to many fields beyond image processing; it can be applied to network science, resource allocation, and biology, among others.

Here, we focus on the applications of GSP in image processing. In classical signal processing, an image is typically viewed as a discrete signal on a finite 2D grid. However, a graph is a more flexible data kernel, allowing definitions of edge weights and more general connectivity among pixels.

We begin by defining the key terminologies in GSP. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ is defined by a set of N vertices $\mathcal{V} = \{1, \ldots, N\}$ and a set of edges $\mathcal{E}$ connecting these nodes. Each edge $(i, j) \in \mathcal{E}$ can have a positive or negative edge weight, $w_{i,j} = W_{i,j}$. The *adjacency matrix* $\mathbf{W} \in \mathbb{R}^{N \times N}$ represents all edges. For an undirected graph, $\mathbf{W}$ is symmetric, *i.e.*, $W_{i,j} = W_{j,i}, \forall i, j$, while for a directed graph, $\mathbf{W}$ is asymmetric.

We define the *degree matrix* $\mathbf{D} \in \mathbb{R}^{N \times N}$ as a diagonal matrix, where each diagonal entry is the sum of the corresponding row of $\mathbf{W}$, i.e., $D_{i,i} = \sum_j W_{i,j}$. Then, the combinatorial graph Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$ is defined as $\mathbf{L} \triangleq \mathbf{D} - \mathbf{W}$. For graphs with self-loops (*i.e.*, $\exists i$ such that $W_{i,i} \neq 0$), the generalized graph Laplacian matrix $\mathcal{L} \in \mathbb{R}^{N \times N}$ is defined as $\mathcal{L} \triangleq \mathbf{D} - \mathbf{W} + (\mathbf{W})$.

Note that $\mathbf{L}$ (or $\mathcal{L}$) is provably positive semi-definite (PSD) if all edge weights are non-negative [40]. By the spectral theorem [41], a real and symmetric $\mathbf{L}$ (or $\mathcal{L}$) can be eigen-decomposed into $\mathbf{L} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^\top$. This eigen-decomposition allows us to perform frequency analysis on graph signals. Specifically, the k-th eigen-pair $(\lambda_k, \mathbf{v}_k)$, where λ_k is the k-th eigenvalue and $\mathbf{v}_k$ is its corresponding eigenvector, represents the k-th graph frequency and Fourier mode for $\mathcal{G}$. Consequently, we can transform a graph signal into the Fourier domain using $\tilde{\mathbf{x}} = \mathbf{V}^\top\mathbf{x}$ and define the Graph Fourier Transform coefficients as $\tilde{x}_k = \mathbf{v}_k^\top\mathbf{x}$ for a signal $\mathbf{x}$.

3.2 Regularization in GSP

There are various graph smoothness measures that can be defined for a graph signal, including the *Graph Laplacian Regularizer* (GLR) [17] and *Graph Total Variation* (GTV) [42]. Given a graph $\mathcal{G}$ represented by its graph Laplacian matrix $\mathbf{L}$, the GLR for a graph signal $\mathbf{x} \in \mathbb{R}^N$ can be defined as

$$\mathbf{x}^\top\mathbf{L}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} w_{i,j}(x_i - x_j)^2 = \sum_k \lambda_k \tilde{x}_k^2. \tag{3.1}$$

Minimizing the GLR results in a smooth signal, where connected samples (x_i, x_j) are similar. This is equivalent to promoting a low-pass signal in the spectral domain.

Similarly, one can define GTV [42] as

$$\|\mathbf{x}\|_{\text{GTV}} = \sum_{(i,j)\in\mathcal{E}} w_{ij}|x_i - x_j|. \tag{3.2}$$

Edge weights $w_{i,j}$'s can be defined in a *signal-dependent* manner, similar to how Bilateral Filter weights [4] are defined, *i.e.*, $w_{i,j}$'s are used to filter a signal $\mathbf{x}$, while the definition of $w_{i,j}$'s depends on $\mathbf{x}$. For example, non-negative $w_{i,j}$ can be computed as

$$w_{i,j} = \exp\left(-\frac{|x_i - x_j|^2}{\sigma_x^2}\right), \tag{3.3}$$

where $\sigma_x > 0$ is a parameter. This results in a *signal-dependent* GLR (SDGLR) [43], meaning that $\mathbf{L}$ becomes a function of the signal, denoted as $\mathbf{L}(\mathbf{x})$. Typically, a signal estimate $\mathbf{x}'$ is first used to defined $\mathbf{L}(\mathbf{x}')$, a new signal $\mathbf{x}$ is computed via graph filter using GLR $\mathbf{x}^\top\mathbf{L}(\mathbf{x}')\mathbf{x}$ as signal prior, then $\mathbf{L}$ is updated based on $\mathbf{x}$, and so on.

More generally, any feature vector $\mathbf{f}_i$ for pixel i, extracted from the signal $\mathbf{x}$, can be used to compute edge weights based on feature distance. We discuss this in detail in the sequel.

3.3 Signal Denoising with GLR

The application of different graph-based priors in image restoration has been studied [17, 44, 28]. Similar to Total Variation (TV), a graph-based prior assumes that the signal is smooth

with respect to a data kernel described by a graph. For denoising, a MAP formulation with GLR as signal prior can be written as

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \mu\, \mathbf{x}\mathcal{L}\mathbf{x}, \tag{3.4}$$

where $\mathcal{L}$ is a generalized graph Laplacian matrix, and $\mu \in \mathbb{R}_+$ is a positive weight parameter.

Given that $\mathcal{L}$ is provably PSD [40], equation (3.4) is an unconstrained quadratic programming (QP) optimization problem with a closed-form solution:

$$(\mathbf{I} + \mu\mathcal{L})\hat{\mathbf{x}} = \mathbf{y}. \tag{3.5}$$

Since $\mathcal{L}$ is PSD and symmetric, the matrix $(\mathbf{I} + \mu\mathcal{L})$ is positive definite (PD) and symmetric for every $\lambda > 0$. Thus, assuming $\mathcal{L}$ is sparse, one can solve the linear system in (3.5) in linear time using the *Conjugate Gradient* (CG) algorithm [45].

3.4 Mapping between Pseudo-linear Matrix and MAP Solution Graph Filter

We review a theorem in [16] that connects a *pseudo-linear* denoiser $\mathbf{\Psi}$ to a graph filter that is a solution to the MAP problem (3.4). By pseudo-linear, we mean that the computation of $\mathbf{\Psi}$ itself may be non-linear, but once $\mathbf{\Psi}$ is computed, the filtering process is $\mathbf{\Psi}\mathbf{y}$—a matrix-vector mutiplication—for observed noisy signal $\mathbf{y}$, and hence a linear operation when $\mathbf{\Psi}$ is fixed. An example of a pseudo-linear denoiser is bilateral filter (BF) [4], where BF weights are signal-dependent.

Specifically, any (pseudo)-linear denoiser $\mathbf{\Psi}$, under mild conditions, has a one-to-one mapping to the solution filter of a MAP denoising problem, regularized by a graph Laplacian regularizer (GLR) [17]. The theorem is as follows:

Theorem 1. *A (pseudo)-linear denoiser $\mathbf{\Psi} \in \mathbb{R}^{N\times N}$ is a solution to the MAP problem if $\mathbf{L} = \mu^{-1}(\mathbf{\Psi}^{-1} - \mathbf{I})$, assuming that the matrix $\mathbf{\Psi}$ is non-expansive, symmetric, and positive definite (PD).*

See [16] for a formal proof.

3.5 Kernel Functions

Let $\mathcal{X}$ be a non-empty set. A function $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a kernel function iff it satisfies the following properties [39]

1. It is symmetric, *i.e.*, $\kappa(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{y}, \mathbf{x}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$, and

2. It is positive semi-definite (PSD), *i.e.*, for any finite set of n points $\{\mathbf{x}_i\}_{i=1}^n$ and real coefficients $\{c_i\}_{i=1}^n$, it satisfies

$$\sum_i \sum_j c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0. \tag{3.6}$$

By the *Mercer's Theorem* [39], a function $\kappa(\mathbf{x}, \mathbf{y})$ is a kernel iff it can be written as

$$\kappa(\mathbf{x}, \mathbf{y}) = \sum_i^\infty \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{y}) \tag{3.7}$$

where non-negative λ_i's are *eigenvalues*, and ψ_i's are *eigenfunctions* in the following sense:

$$\int \kappa(\mathbf{x}, \mathbf{y}) \psi_i(\mathbf{y}) d\mathbf{y} = \lambda_i \psi_i(\mathbf{x}). \tag{3.8}$$

These eigenfunctions are orthogonal, *i.e.*,

$$\int \psi_i(\mathbf{x}) \psi_j(\mathbf{x}) d\mathbf{x} = \delta_{i-j}. \tag{3.9}$$

Eigenfunctions ψ_i can be derived by solving (3.8), also known as *Fredholm integral equation* of the second kind [46]. However, (3.8) is analytically solvable only for a few special kernels.

(3.7) allows us to define a mapping ψ from $\mathcal{X}$ to a (possibly) infinite-dimensional feature space:

$$\psi(\mathbf{x}) = \sum_i^\infty \sqrt{\lambda_i} \psi_i(\mathbf{x}). \tag{3.10}$$

Given ψ, we can also write the kernel function in terms of inner product:

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \psi(\mathbf{x}), \psi(\mathbf{y}) \rangle. \tag{3.11}$$

As an inner product, $\kappa(\mathbf{x}, \mathbf{y})$ satisfies

1. Symmetry: $\kappa(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{y}, \mathbf{x}), \quad \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{X}$
2. Linearity: $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$, $\alpha \in \mathbb{R}$,
$$\kappa(\alpha \mathbf{x}, \mathbf{y}) = \alpha \kappa(\mathbf{x}, \mathbf{y}),$$
$$\kappa(\mathbf{x} + \mathbf{z}, \mathbf{y}) = \kappa(\mathbf{x}, \mathbf{y}) + \kappa(\mathbf{z}, \mathbf{y})$$
3. Positive definiteness: $\forall \mathbf{x} \in \mathcal{X}$,
$$\kappa(\mathbf{x}, \mathbf{x}) \geq 0 \qquad \text{(non-negativity)}$$
$$\kappa(\mathbf{x}, \mathbf{x}) = 0 \text{ iff } \mathbf{x} = \mathbf{0} \qquad \text{(definiteness)}$$

We provide some example kernel functions below:

1. Linear kernel: $\kappa(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^N$
2. Gaussian kernel: $\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}\right), \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^N, \sigma > 0$

Chapter 4

Proposed Method

Our proposal is based on model-based deep learning [38]; specifically, we first construct an iterative image denoising algorithm derived from a MAP optimization formulation regularized using GLR as done in (3.4), then unroll the iterations into neural layers for data-driven parameter tuning. While unrolling GLR/GTV has been studied in previous works [47, 48, 49], these methods come with two major drawbacks. First, the matrix inversion operation used in [47] to derive an optimal graph filter is complex and computationally expensive, making backpropagation challenging. Second, previous unrolled GLR and GTV methods used random or zero initialization for network parameters, which does not guarantee good performance, especially given the non-convex nature of the cost functions and the multiple local minima encountered in conventional *stochastic gradient descent* (SGD) [35].

We leverage Theorem 1 to address the initialization problem. As stated, under mild conditions, any (pseudo-)linear denoiser $\mathbf{\Psi}$ can be mapped to a solution graph filter of a MAP problem regularized by GLR. This allows us to initialize an unrolled graph-based denoising network, specifically the graph Laplacian matrix $\mathbf{L}$, using a reliable and well-established (pseudo-)linear denoiser $\mathbf{\Psi}$, and then further optimize sparse matrix $\mathbf{L}$ from that starting point. Doing so means that we begin with a performance-guaranteed denoiser $\mathbf{\Psi}$, and progressively improve its performance via parameter tuning using SGD, mitigating the common problem of poor local minima due to random initialization.

4.1 Problem Formulation

Denote by $\mathbf{L}$ a graph Laplacian derived from a (pseudo-)linear denoiser $\mathbf{\Psi}$, *i.e.*, according to Theorem 1, $\mathbf{L} = \mu^{-1}(\mathbf{\Psi}^{-1} - \mathbf{I}_N)$. Suppose we perturb $\mathbf{L}$ with symmetric matrix $\mathbf{P}$, *i.e.*, $\tilde{\mathbf{L}} = \mathbf{L} + \mathbf{P}$. Using GLR $\mathbf{x}^\top \tilde{\mathbf{L}} \mathbf{x}$ as regularizer for signal denoising, the resulting denoised signal

$\mathbf{x}^*$ given noisy observation $\mathbf{y}$ is

$$(\mathbf{I} + \mu\tilde{\mathbf{L}})\mathbf{x}^* = \mathbf{y} \tag{4.1}$$

$$(\mathbf{I} + \mu(\mathbf{L} + \mathbf{P}))\mathbf{x}^* = \mathbf{y} \tag{4.2}$$

$$(\mathbf{I} + \mu\mathbf{L})\mathbf{x}^* + \mu\mathbf{P}\mathbf{x}^* = \mathbf{y} \tag{4.3}$$

$$\mathbf{x}^* + \underbrace{(\mathbf{I} + \mu\mathbf{L})^{-1}}_{\mathbf{\Psi}}\mu\mathbf{P}\mathbf{x}^* \overset{(a)}{=} \underbrace{(\mathbf{I} + \mu\mathbf{L})^{-1}}_{\mathbf{\Psi}}\mathbf{y} \tag{4.4}$$

$$(\mathbf{I} + \mu\mathbf{\Psi}\mathbf{P})\mathbf{x}^* = \mathbf{\Psi}\mathbf{y} \tag{4.5}$$

where (a) is possible since $\mathbf{L}$ is PSD, and thus $\mathbf{I} + \mu\mathbf{L}$ is PD and invertible. (4.5) is a linear system, and thus $\mathbf{x}^*$ can be computed via *conjugate gradient* (CG) [45]—an iterative algorithm—in linear time, assuming that coefficient matrix $\mathbf{I} + \mu\mathbf{\Psi}\mathbf{P}$ is sparse, symmetric and PD (to be discussed). Iterations of CG can be unrolled into neural layers to compose a feed-forward network for data-driven parameter tuning. Note that unlike [47], *no matrix inversion is required to compute solution* $\mathbf{x}^*$ *in the unrolled feed-forward network.*

4.2 Interpretation of (4.5)

(4.5) provides an interesting interpretation: perturbing graph Laplacian $\mathbf{L}$ additively by $\mathbf{P}$ translates to a *cascade of two consecutive filters*—the original denoiser $\mathbf{\Psi}$, followed by a solution graph filter $(\mathbf{I} + \mu\mathbf{\Psi}\mathbf{P})^{-1}$, defined by symmetric *augmented Laplacian* $\mathbf{\Psi}\mathbf{P}$, that is a solution to the MAP optimization:

$$\min_{\mathbf{x}} \|\mathbf{\Psi}\mathbf{y} - \mathbf{x}\|_2^2 + \mu\,\mathbf{x}^\top(\mathbf{\Psi}\mathbf{P})\mathbf{x}. \tag{4.6}$$

Note that (4.6) is a valid MAP optimization iff $\mathbf{\Psi}\mathbf{P}$ is PSD. Given that $\mathbf{\Psi}$ must be PD to satisfy conditions for Theorem 1, it means $\mathbf{P}$ must be PSD. We revisit this *PSD cone constraint* $\mathbf{P} \succeq 0$ in the sequel.

4.3 Implementation of (4.5)

In practical implementation, (4.5) means that one can learn a *sparse* symmetric perturbation matrix $\mathbf{P}$ from data, initialized to the zero matrix, to further improve performance from the default denoiser $\mathbf{\Psi}$. Note that $\mathbf{P}$ can be designed to be sparse and still be effective; dense image denoisers $(\mathbf{I} + \mu\mathbf{L})^{-1}$ derived from GLR regularization based on sparse graph Laplacian matrices $\mathbf{L}$ have long been demonstrated to have good performance [47, 48]. (In contrast, denoisers like $\mathbf{\Psi}$ are not sparse in general.) Thus, parameterization of $\mathbf{P}$ can potentially be more efficiently implemented than parameterizing perturbation matrix $\mathbf{P}$ in $\mathbf{\Psi} + \mathbf{P}$ directly.

4.4 Parameterizing Perturbation Matrix P

We design the parameterization of the pertubation matrix $\mathbf{P}$ to satisfy the following requirements:

1. $\mathbf{P}$ should be symmetric for $\tilde{\mathbf{L}} = \mathbf{L} + \mathbf{P}$ to remain symmetric;
2. $\mathbf{P}$ should be sparse towards parameter efficiency; and
3. $\mathbf{P}$ must be PSD to ensure $\mathbf{\Psi P}$ is PSD, so that MAP optimization (4.6) is convex with linear system (4.5) as a solution.

One way to satisfy the PSD cone constraint $\mathbf{P} \succeq 0$ is to enforce $\mathbf{P} = \alpha \mathbf{L}^+$ to be a scaled Laplacian matrix $\mathbf{L}^+$ (for non-negative scalar $\alpha \geq 0$) corresponding to a positive graph $\mathcal{G}^+(\mathcal{V}, \mathcal{E}^+)$ [40], *i.e.*, $\mathbf{L}^+ = \text{diag}(\mathbf{W}^+\mathbf{1}) - \mathbf{W}^+$. Here, $\mathbf{W}^+ \in \mathbb{R}^{N \times N}$ is an adjacency matrix for a positive undirected graph $\mathcal{G}^+$, *i.e.*, $W^+_{i,j} = W^+_{j,i} \geq 0, \forall i, j$. Undirected graph $\mathcal{G}^+$ implies $\mathbf{W}^+$ is symmetric, and thus $\mathbf{L}^+$ and $\mathbf{P}$ are also symmetric.

Specifically, to ensure that each edge weight $W^+_{i,j} = w^+_{i,j}$ for edge $(i,j) \in \mathcal{E}^+$ is non-negative, we can define $w^+_{i,j}$ as a function of non-negative feature distance $d_{i,j} \geq 0$ between pixels i and j using the exponential kernel, *i.e.*,

$$w^+_{i,j} = \exp\left(-d_{i,j}\right) \tag{4.7}$$

$$d_{i,j} = (\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{M} (\mathbf{f}_i - \mathbf{f}_j) \tag{4.8}$$

where $\mathbf{M} \in \mathbb{R}^{K \times K}$, $\mathbf{M} \succeq 0$, is a PSD *metric matrix*, so that *Mahalanobis distance* (feature distance) $d_{i,j} \geq 0$. $\mathbf{f}_i \in \mathbb{R}^K$ is a K-dimensional *feature vector* for node i, where $K \ll N$, learned from data, for example via a shallow *convolutional neural net* (CNN), as done in previous unrolled GLR / GTV work [47, 48]. From (4.7), it is clear that $w^+_{i,j} \in (0, 1]$. Parameterization of $\mathbf{P}$ will thus mean parameterizing CNNs required to compute features $\mathbf{f}_i$ and metric matrix $\mathbf{M}$.

4.5 Sparsifying $\mathbf{\Psi}$

To solve linear system (4.5) in linear time via CG, coefficient matrix $\mathbf{I} + \mu \mathbf{\Psi P}$ must be sparse, meaning that the number of nonzero matrix entries should be $\mathcal{O}(N)$. However, as previously discussed, (pseudo-)linear denoiser $\mathbf{\Psi}$ is dense in general, meaning that $\mathbf{\Psi P}$ can be still dense even if $\mathbf{P}$ is sparse. One remedy is to sparsify $\mathbf{\Psi}$ before computing $\mathbf{I} + \mu \mathbf{\Psi P}$ as input to CG. Specifically, we can first compute sparse $\tilde{\mathbf{\Psi}} = \text{th}_\epsilon(\mathbf{\Psi})$, where each entry $\tilde{\Psi}_{i,j}$ is a hard-thresholding operation of $\Psi_{i,j}$ for pre-chosen threshold $\epsilon > 0$.

4.6 Modifying $\mathbf{\Psi}$ to Satisfy Theorem 1 Conditions

A known well-performing denoiser $\mathbf{\Psi}$ may not satisfy the required conditions for us to employ for initialization in our unrolled graph denoising network. According to Theorem 1, denoiser $\mathbf{\Psi}$ must satisfy the three specified conditions:

1. $\mathbf{\Psi}$ must be symmetric;
2. $\mathbf{\Psi}$ must be positive definite (PD); and
3. $\mathbf{\Psi}$ must be non-expansive.

We address the problem of modifying a subset of linear denoisers, denoted by $\{\mathbf{\Psi}\}_{\text{init}}$, to ensure it satisfies all three conditions required by Theorem 1. A common and popular subset of (pseudo-)linear denoisers is *kernel denoisers* [50]. We define kernel denoiser as follows. We first assume a known strictly positive *kernel function* (discussed in Section 3.5) $\kappa : \mathbb{R}^K \times \mathbb{R}^K \mapsto \mathbb{R}_+$ that quantifies strictly positive values of affinity (or similarity) between features vectors $\mathbf{f}_i$ and $\mathbf{f}_j$ of dimension K representing two pixels i and j. The pairwise kernel values for a collection of N pixels are ordered into a *kernel matrix* $\mathbf{K} \in \mathbb{R}^{N \times N}$, *i.e.*, $\mathbf{K}_{i,j} = \kappa(\mathbf{f}_i, \mathbf{f}_j)$, where $\{\mathbf{f}_i\}_{i=1}^N$ represents a set of features. Note that given κ is a kernel function, matrix $\mathbf{K}$ must be symmetric and PSD [39]. A kernel denoiser is then defined as $\mathbf{\Psi}_\kappa \triangleq \mathbf{D}^{-1}\mathbf{K}$, where $\mathbf{D}$ is a diagonal matrix, typically used so that $\mathbf{\Psi}_\kappa$ is *row-stochastic* (sums of row entries equal to 1). (In one special case, $\mathbf{D} = \mathbf{I}$ and $\mathbf{\Psi}_\kappa = \mathbf{K}$.) Note that $\mathbf{\Psi}_\kappa$ is not symmetric in general, and hence it does not satisfy the first required condition. An example of kernel denoiser is the *bilateral filter* (BF) [4].

Given a kernel denoiser $\mathbf{\Psi}_\kappa$, we pre-process it to satisfy the three requirements for Theorem 1 as follows. We first compute a *shifted* kernel matrix $\mathbf{K}'$ as $\mathbf{K}' \leftarrow \mathbf{D}\mathbf{\Psi}_\kappa + \delta\mathbf{I}$ for a small positive parameter $\delta > 0$. This increases the matrix's non-negative eigenvalues by $\delta > 0$, so that $\mathbf{K}'$ has strictly positive eigenvalues. We then input $\mathbf{K}'$ to the *Sinkhorn-Knopp* (SK) algorithm [51], which outputs a symmetric *doubly-stochastic* matrix $\mathbf{K}''$ (*i.e.*, sums of rows and columns equal to 1). It is known that, given a symmetric, strictly positive matrix $\mathbf{K}'$, the SK Algorithm converges surely to a symmetric doubly stochastic matrix [52]. Being doubly stochastic, $\mathbf{K}''$ has eigenvalues no larger than 1, and thus non-expansive. Therefore, $\mathbf{K}''$ satisfies all three conditions required by Theorem 1.

4.7 CNN Architectures

We employ two CNNs to optimize the scalar GLR weight factor μ and parameters that define the matrix-valued perturbation matrix $\mathbf{P}$ in (4.5), each designed with distinct architectural details suited to their respective output domains.

Motivated by the designs in [48, 47], we use a lightweight CNN, denoted by CNN_μ, to estimate μ, which follows a typical feature-extraction pipeline: three convolutional layers with progressively increasing channel depth ($32 \rightarrow 64 \rightarrow 128$), each followed by batch

normalization and a leaky ReLU activation. Spatial dimensionality is reduced via strided convolutions, and global feature aggregation is achieved through an adaptive average pooling layer. The resulting feature vector is passed through a two-layer fully connected module, ending with a sigmoid activation scaled to a predefined range $[\mu_{\min}, \mu_{\max}]$ to enforce positivity and boundedness of μ values. This architectural choice ensures stable estimates with low variance, suitable for controlling the balance between data fidelity and prior regularization.

Next, following the discussion in Section 4.4, we design $\text{CNN}_{\mathbf{f}}$ to extract feature vector $\mathbf{f}_i$ for each pixel i as follows. $\text{CNN}_{\mathbf{f}}$ employs an adaptive depth mechanism controlled by a tunable parameter K, allowing flexible adjustment of model capacity based on the denoising complexity. It uses LeakyReLU activation and Bath Normalization (BN) in all layers to improve training stability. In contrast, prior unrolled methods such as DGLR [47] and DGTV [48] typically avoid normalization layers and rely on standard activation like ReLU, which can lead to unstable training and slower convergence. This is due to the fact that BN mitigates internal covariate shift and smooths the optimization landscape, while LeakyReLU prevents neuron inactivity and maintains gradient flow.

Together, these two modules form the adaptive prior estimation mechanism in our model, where CNN_{μ} provides global scalar regularization control, and $\text{CNN}_{\mathbf{f}}$ encodes local structural information for denoising guidance.

4.8 Overall Model Architecture

The overall architecture of the proposed method is illustrated in Figure 4.3. The model takes the noisy input image $\mathbf{y}$ and processes it through two main convolutional branches. The first branch, denoted as CNN_f, is responsible for extracting feature representations from $\mathbf{y}$. These features are then used to compute the perturbation matrix based on the Mahalanobis distance, which encodes pairwise relationships between image pixels and is later used in the regularization term.

In parallel, the second branch, CNN_{μ}, processes the same noisy input $\mathbf{y}$ to estimate the regularization parameter μ, which adaptively balances the data-fidelity term and the regularization term in the reconstruction process.

The outputs of CNN_f and CNN_{μ}, together with the initial denoiser matrix $\mathbf{\Psi}$, are integrated into a linear system as expressed in Equation (4.5). This system is efficiently solved using the Conjugate Gradient (CG) method to produce the denoised image.

The loss function is computed by comparing the reconstructed image with the ground truth image, and gradients are propagated back through all components of the architecture. As shown in Figure 4.3, the red dashed arrows indicate the backpropagation flow across the different blocks, enabling end-to-end training of the network parameters.

In the architecture shown in Figure 4.3, the learnable parameters are distributed across the different modules. In the feature extraction branch CNN_f, convolutional filter weights are trained to produce the feature vectors $\mathbf{f}_i$ for each pixel i. These features are then used in Eq. (4.8) to compute the metric matrix $\mathbf{M}$ via the Mahalanobis distance. In the

regularization branch CNN_μ, convolutional filter weights are learned to map the noisy input $\mathbf{y}$ to the spatially varying regularization parameters μ_i, which adaptively balance the data-fidelity and regularization terms in Eq. (4.5). Overall, the trainable parameters consist of the convolutional weights of CNN_f and CNN_μ, which determine $\mathbf{f}_i$, μ, and metric matrix $\mathbf{M}$ as defined in Eqs. (4.8) and (4.5), and are jointly optimized in an end-to-end fashion through backpropagation, as indicated by the red dashed arrows in Figure 4.3.

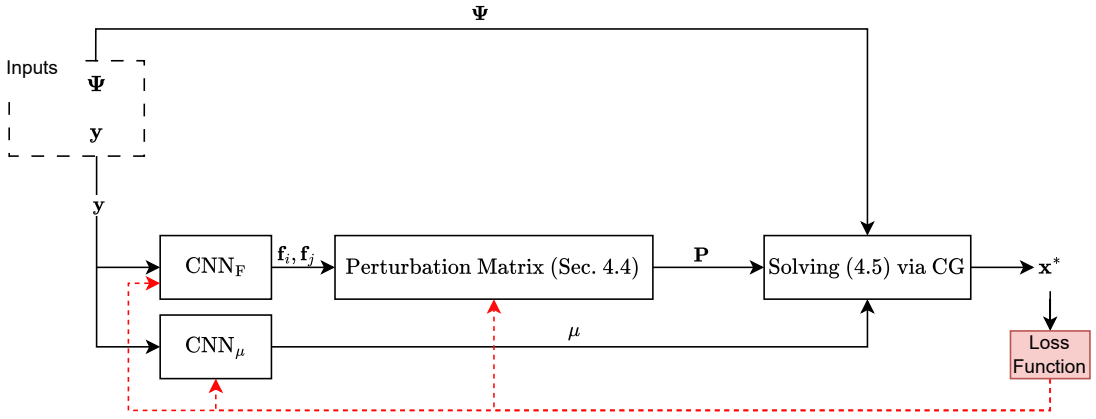


Figure 4.3: Overall model architecture: The noisy image $\mathbf{y}$ is fed into CNN_f to extract features, followed by computation of the perturbation matrix using Mahalanobis distance. In parallel, $\mathbf{y}$ is also fed into CNN_μ to estimate the regularization parameter μ. Using these parameters along with the initial denoiser matrix $\mathbf{\Psi}$, the linear equation (4.5) is solved via the CG method. The loss is then computed against the ground truth, and backpropagation updates the parameters of each block. Red dashed arrows indicate the backpropagation flow.

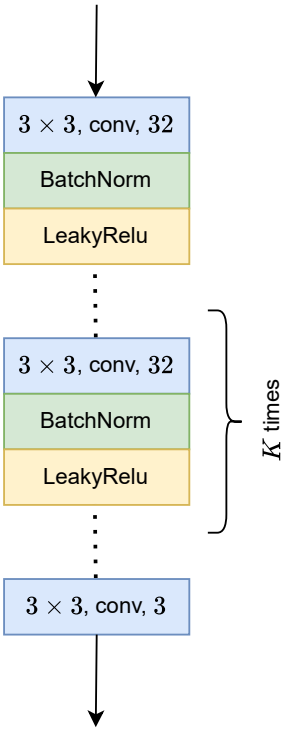


Figure 4.1: CNN architecture $\text{CNN}_{\mathbf{f}}$ for producing feature maps.

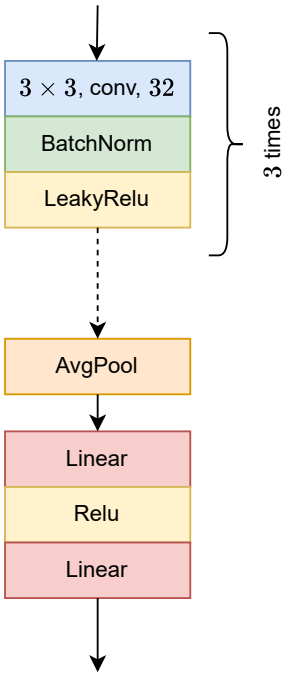


Figure 4.2: CNN architecture CNN_{μ} for regularization parameter.

Chapter 5

Experiments

We report experimental results in this chapter. First, we show that our new implementation in this thesis—parameterizing a single perturbation matrix $\mathbf{P}$ in (4.5)—has improved image quality over our earlier version published in a conference paper in 2024 [53]. Then, we show that optimizing perturbation matrix $\mathbf{P}$ in a data-driven manner improves performance over the baseline model initialized with denoiser $\mathbf{\Psi}$. This validates the effectiveness of our initialized model plus perturbation, and confirms its practical benefit in guiding the optimization towards better solutions.

5.1 Dataset Description

We use the DIV2K dataset for training, which consists of 800 high-resolution RGB images. From each image, we randomly extract 10 patches of size 36×36, resulting in a total of 8000 training samples. All patches are processed in the RGB color space, and no data augmentation is applied. During training, *additive white Gaussian noise* (AWGN) with standard deviations $\sigma \in \{15, 25, 50, 75\}$ is applied to the clean patches to simulate various noise levels and improve the model's generalization.

For evaluation, we employ four standard benchmark datasets commonly used in image denoising: McM, Kodak, Urban100, and CBSD68. All test images are used in full resolution without cropping or resizing. The same noise levels are applied to the test images, and denoising performance is measured using the average peak signal-to-noise ratio (PSNR) across each dataset.

5.2 Evaluation Metrics

In this thesis, the quality of the reconstructed images is assessed using two widely adopted metrics in image processing and computer vision: **Peak Signal-to-Noise Ratio (PSNR)** and **Structural Similarity Index Measure (SSIM)**.

5.2.1 Peak Signal-to-Noise Ratio (PSNR)

PSNR quantifies the difference between a reference image I and a reconstructed (or degraded) image $\hat{I}$ based on the pixel-wise Mean Squared Error (MSE). It is expressed in decibels (dB) and defined as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{\text{MSE}} \right) \tag{5.1}$$

where MAX_I is the maximum possible pixel value of the image (255 for 8-bit images), $\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[I(i,j) - \hat{I}(i,j) \right]^2$, and m and n are the image dimensions. A higher PSNR value indicates better fidelity of the reconstructed image to the reference image. However, PSNR is a purely mathematical measure and does not always align with human perception of visual quality.

5.2.2 Structural Similarity Index Measure (SSIM)

SSIM [54] evaluates the perceptual similarity between two images by considering changes in structural information, luminance, and contrast. Given two image patches x and y, SSIM is defined as:

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{5.2}$$

where μ_x and μ_y are the mean intensities of x and y, σ_x^2 and σ_y^2 are the variances, σ_{xy} is the covariance between x and y, and C_1 and C_2 are small constants to stabilize the division. SSIM values range from -1 to 1, where 1 indicates perfect structural similarity. Unlike PSNR, SSIM correlates more closely with the human visual system by focusing on structure-preserving characteristics.

5.3 Comparison with Early Version [53]

In a nutshell, our earlier implementation [53] first computes the graph Laplacian matrix $\mathbf{L}$ corresponding to a known denoiser $\mathbf{\Psi}$ according to Theorem 1, approximated via truncated Taylor's series expansion due to complexity concerns. Computed $\mathbf{L}$ is then used to define the linear system (3.5) that is a solution the MAP denoising problem regularized using GLR (3.4). Unrolling CG used to solve (3.5) defined by $\mathbf{L}$ into neural layers results in a neural net, which is the denoising model. [53] considered specifically a generalized bilateral filter [4] as initializing denoiser $\mathbf{\Psi}$, one that is amenable to direct parameterization of σ's, berfore conversion to $\mathbf{L}$, that determine the relative weights of domain and range filters in a data-driven manner.

In contrast, in this thesis our model can be more generally initialized using *any* given known denoiser $\mathbf{\Psi}$ that satisfies the three requirements in Theorem 1. Further, thanks to derivation that results in linear system (4.5), no approximation of matrix inverse (via truncated Taylor's series) is necessary, reducing algorithm complexity. To evaluate the impact

of this architectural design change, we tested both implementations on a fixed set of randomly sampled image patches and measured the resulting PSNR. Table 5.1 presents the results for the *McM* dataset under a Gaussian noise level of $\sigma = 25$. The ICIP'24 baseline includes three variants with increasing model complexity: M (learning only the metric matrix in the generalized bilateral filter), M+C (additionally learning the Taylor series coefficients), and M+C+CGD (further incorporating the learning of conjugate gradient descent parameters). As shown, all three variants of the ICIP'24 method yield PSNR values significantly lower than our model, both before and after training.

The results consistently showed that the new implementation in this thesis significantly improved performance, confirming that the new learned perturbation component plays a critical role in enhancing the filtering behavior beyond what is achievable using fixed, handcrafted features.

The untrained version of our model ("Init") achieves a PSNR of 29.46 dB, already outperforming the most advanced ICIP'24 variant (M+C+CGD), which reaches only 28.96 dB. After training, our model improves further to 30.19 dB, surpassing the best ICIP'24 result by more than 1.2 dB. This substantial margin highlights the effectiveness of our perturbation-based optimization design.

5.4 Improvement from Initialization

We show that optimizing parameters of the perturbation matrix $\mathbf{P}$ improves image quality over the baseline model initialized to denoiser $\mathbf{\Psi}$. As shown in Table 5.2, we compare the performance of the initialized model with the parameter-optimized version across four standard datasets (*McM*, *Kodak*, *Urban*, and *CBSD68*) and multiple noise levels ($\sigma = 15$, 25, 50, 75). For example, on the McM dataset with $\sigma = 50$, the untrained model achieves 25.01 dB, while the trained version reaches 27.60 dB—an improvement of 2.59 dB. Similar gains are observed across all datasets, confirming the effectiveness of the added perturbation matrix $\mathbf{P}$ and the positive influence of the learned parameters.

Although we restrict our experiments to a single perturbation matrix $\mathbf{P}$, the positive results suggest that further improvement may be possible by learning multiple perturbation matrices, which we leave for future investigation.

5.5 Model Performance Analysis

Figure 5.1 provides a visual comparison of various denoising methods on a representative image from the McM dataset with noise level $\sigma = 25$. The noisy input exhibits severe degradation (PSNR = 21.05 dB, SSIM = 0.384), making it a challenging case for restoration.

Our proposed method, even prior to training, produces a strong initial estimate with a PSNR of 28.89 dB and SSIM of 0.776, outperforming the traditional NLM method [5] (28.52 dB, 0.7631) and achieving competitive quality without any learned parameters. After training, the model further improves to 30.13 dB and 0.8327 SSIM, surpassing the classical

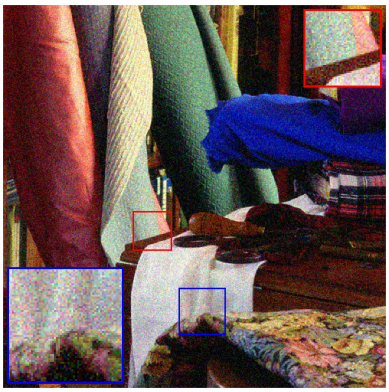
Table 5.1: PSNR (dB) comparison between our proposed method and the previous method on the *McM* dataset with noise level $\sigma = 25$. The ICIP'24 method has three training variants that differ in the number of parameters, denoted as M, M+C, and M+C+CGD. More details about the ICIP'24 method can be found in [53].

Model	**Proposed Model**		**ICIP'24**		
PSNR (dB)	Init	Trained	M	M + C	M + C + CGD
	29.46	30.19	28.16	28.69	28.96

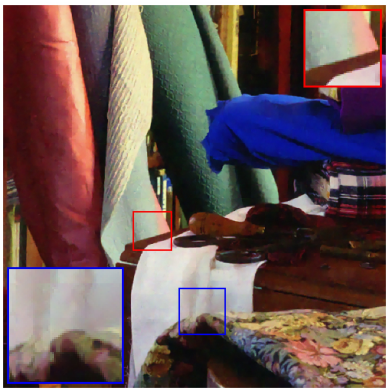
Table 5.2: PSNR (dB) comparison for different methods on four datasets at varying noise levels σ.

Dataset	σ	NLM	BM3D	CDnCNN	Ours (initialization)	Ours (Trained)
McM	15	31.99	33.69	33.45	32.36	33.35
	25	29.05	30.70	31.52	29.46	30.91
	50	24.05	25.86	28.62	25.01	27.60
	75	20.73	22.48	25.10	21.90	25.67
Kodak	15	31.70	34.25	34.60	32.55	33.68
	25	29.13	31.47	32.14	30.06	31.10
	50	24.88	27.35	28.95	26.43	27.86
	75	22.05	24.31	25.10	23.73	26.26
Urban	15	30.13	32.92	31.96	30.93	31.73
	25	27.27	29.96	29.60	28.11	28.75
	50	22.16	25.39	26.36	24.00	25.21
	75	18.74	22.15	23.34	21.30	23.14
CBSD68	15	30.55	33.32	33.90	31.67	32.93
	25	28.01	30.30	31.24	29.03	30.25
	50	24.03	26.03	27.95	25.33	26.93
	75	21.23	23.13	24.49	22.72	25.15

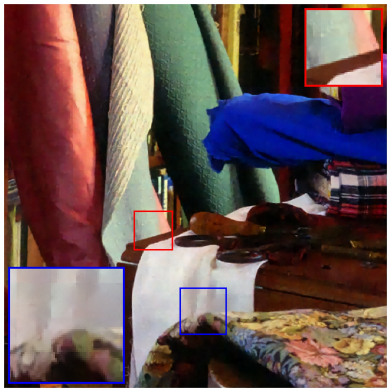
(a) Ground truth
(PSNR (dB) | SSIM)



(b) Noisy ($\sigma = 25$)
(21.05 | 0.384)



(c) Initialization
(28.89 | 0.776)



(d) Trained model
(30.13 | 0.8327)

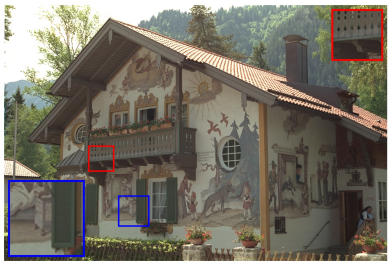
(e) BM3D
(30.03 | 0.8176)

(f) DnCNN
(**30.77** | **0.8509**)

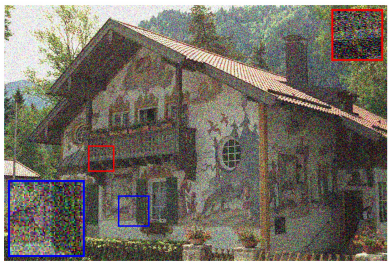
(g) NLM
(28.52 | 0.7631)

Figure 5.1: Visual comparison of different methods on McM (second image).

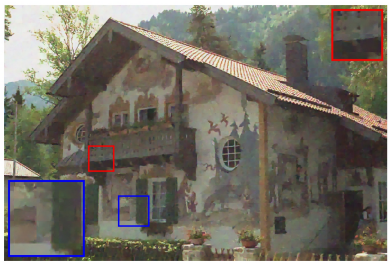
BM3D method [32] (30.03 dB, 0.8176) and approaching the performance of the deep-learning-based DnCNN model [18] (30.77 dB, 0.8509).



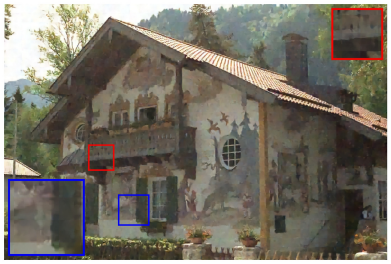
(a) Ground truth
(PSNR (dB) | SSIM)



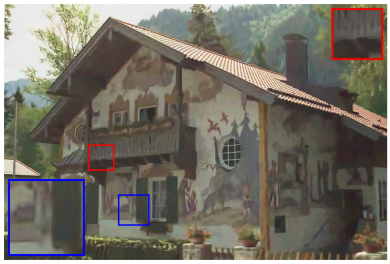
(b) Noisy ($\sigma = 50$)
(14.78 | 0.208)



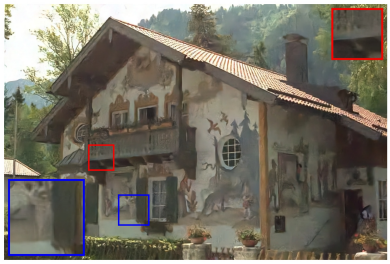
(c) Initialization
(25.01 | 0.7047)



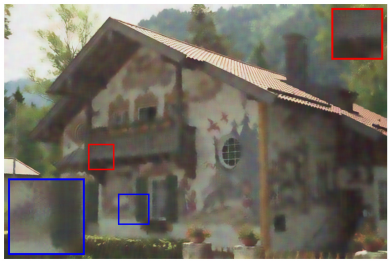
(d) Trained model
(26.21 | 0.7381)



(e) BM3D
(25.62 | 0.7405)



(f) DnCNN
(**26.97** | **0.7754**)



(g) NLM
(23.33 | 0.5760)

Figure 5.2: Visual comparison of different methods on Kodak (24th image).

Figure 5.3 illustrates the visual performance of different denoising methods on the 24$^{\text{th}}$ image of the Kodak dataset under severe Gaussian noise ($\sigma = 50$). The noisy image exhibits strong degradation with a PSNR of 14.78 dB and SSIM of 0.208.

The specific test images shown in Figures 5.1 and 5.3 were selected because they contain diverse and challenging visual structures that highlight different aspects of denoising performance. The McM example (second image in the dataset) includes fine-grained textures and smooth regions in the same scene, making it suitable for assessing the model's ability to preserve high-frequency details while avoiding over-smoothing in flat areas. The Kodak

example (24th image in the dataset) contains sharp edges, distinct object boundaries, and strong contrast transitions, which are useful for evaluating the preservation of structural information and avoidance of ringing or edge artifacts. By selecting these representative and visually challenging cases, we provide a qualitative comparison that complements the quantitative PSNR and SSIM results.

Our model, when applied without any training, produces an initial output with 25.01 dB PSNR and 0.7047 SSIM—indicating that the initialized model already performs reasonably well in a challenging noisy setting. After training, the performance improves further to 26.21 dB and 0.7381 SSIM, outperforming BM3D (25.62 dB, 0.7405) and significantly surpassing NLM (23.33 dB, 0.5760).

Although DnCNN achieves the best result in this case (26.97 dB, 0.7754), our model remains competitive while relying on a simpler and more interpretable architecture. The consistent gain from initialization to trained output confirms the robustness and effectiveness of our approach, even under high noise levels.

Table 5.2 reports PSNR (in dB) for different denoising methods on four benchmark datasets (*McM*, *Kodak*, *Urban*, *CBSD68*) across four noise levels ($\sigma \in \{15, 25, 50, 75\}$). Our method is evaluated both at initialization and after full training. The results clearly show that our initialization alone outperforms traditional non-learning-based methods such as NLM across all settings. For example, on Kodak at $\sigma = 50$, the initialization reaches 26.43 dB compared to NLM's 24.88 dB. After training, our method achieves PSNRs that are consistently higher than BM3D and approaches the deep learning-based DnCNN. Notably, on McM at $\sigma = 75$, our trained model reaches 25.67 dB, outperforming BM3D (22.48 dB) and coming within 0.57 dB of DnCNN.

Overall, the improvement from initialization to trained model ranges from 0.8 dB to over 2.6 dB, depending on the dataset and noise level. This confirms two key aspects of our method: (i) the initialization provides a strong starting point that captures essential image structures, and (ii) the training process effectively refines the solution and closes the gap with fully-trained deep networks, despite our model being significantly simpler and more interpretable.

5.5.1 Robustness to Statistical Mismatch

To evaluate robustness against *statistical mismatch*—where test-time noise distributions deviate from training conditions—we trained both our proposed method and DnCNN on synthetic Gaussian noise with $\sigma = 25$, and evaluated them on the McM dataset with noise levels ranging from $\sigma = 25$ to 70.

As shown in Figure 5.4, both models exhibit PSNR degradation as noise increases. However, the proposed method consistently generalizes better under mismatch. At the training noise level ($\sigma = 25$), DnCNN slightly outperforms our model (31.52 dB vs. 30.91 dB), but at the cost of a significantly higher parameter count and computational complexity.

Beyond the training noise level ($\sigma > 25$), DnCNN's performance degrades more rapidly, indicating inferior robustness to unseen noise statistics:

Table 5.3: Comparison of PSNR (dB) under statistical mismatch: models trained on noise level $\sigma = 25$ and tested on $\sigma = 45$.

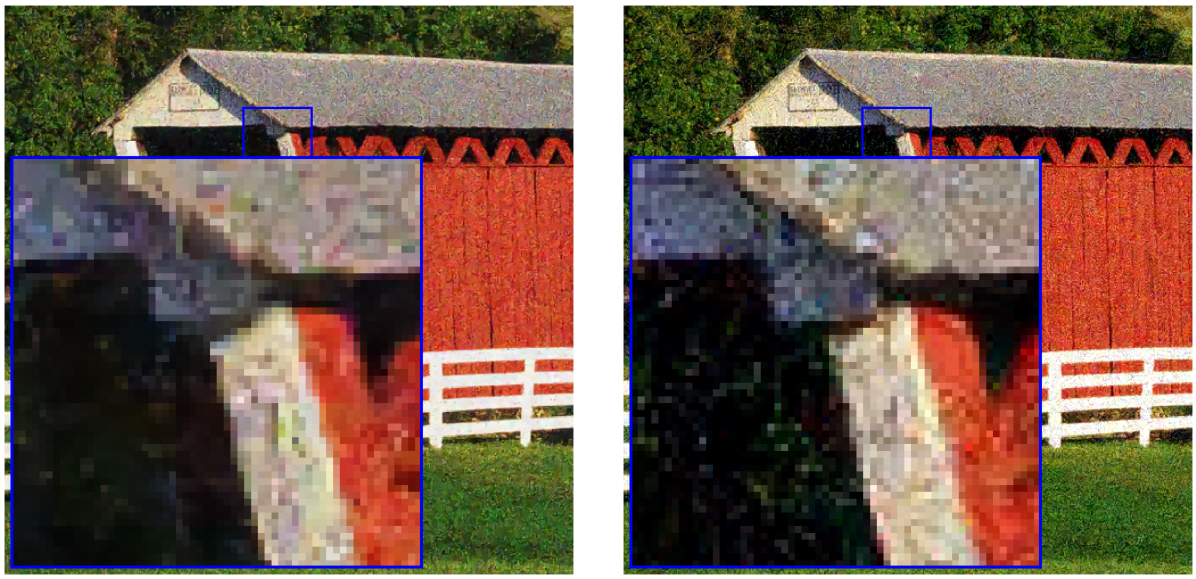
Method	# Parameters	$\sigma_{\text{train}} = 25$	
		$\sigma_{\text{test}} = 45$	SSIM
Ours	0.25 M	24.12	0.539
CDnCNN	0.56M	22.81	0.431

- At $\sigma = 35$, our method achieves 27.83 dB vs. 27.21 dB for DnCNN.
- At $\sigma = 45$, the gap widens to 24.12 dB vs. 22.81 dB.
- At $\sigma = 70$, our model still reaches 18.45 dB, while DnCNN falls to 17.19 dB.

This trend confirms that although DnCNN may perform slightly better under matched noise conditions, it fails to generalize under distribution shift—precisely where our method excels. Combined with its lower computational footprint, the proposed model offers a more robust and efficient solution for practical denoising tasks with uncertain or varying noise levels.

Table 5.3 compares the performance of our model with CDnCNN under a covariate shift scenario, where both models are trained on image patches with noise level $\sigma = 25$ and tested on the *McM* dataset with $\sigma = 45$. Our model achieves significantly better generalization under statistical distribution shift, while using less than half the number of parameters compared to DnCNN. This can be attributed to the use of shallow CNNs for both regularization parameter estimation and feature extraction, which improves robustness and reduces overfitting.

In addition to improved generalization, the reduced parameter count of the proposed model offers tangible computational and storage benefits. With less than half the number of parameters of DnCNN, the model size is reduced proportionally, resulting in approximately a twofold decrease in storage requirements when using 32-bit floating-point representation. For instance, a network with 0.30 million parameters occupies only about 1.2 MB compared to 2.6 MB for a model with 0.65 million parameters. This reduction not only lowers the persistent storage cost but also decreases memory usage during inference and accelerates model loading, making the method more suitable for resource-constrained environments.



(a) Proposed method output (PSNR: 24.13)

(b) DnCNN output (PSNR: 22.15)

Figure 5.3: Visual comparison between the proposed method and DnCNN, both trained on noise level $\sigma = 25$ and tested on a noisy image with noise level $\sigma = 45$.

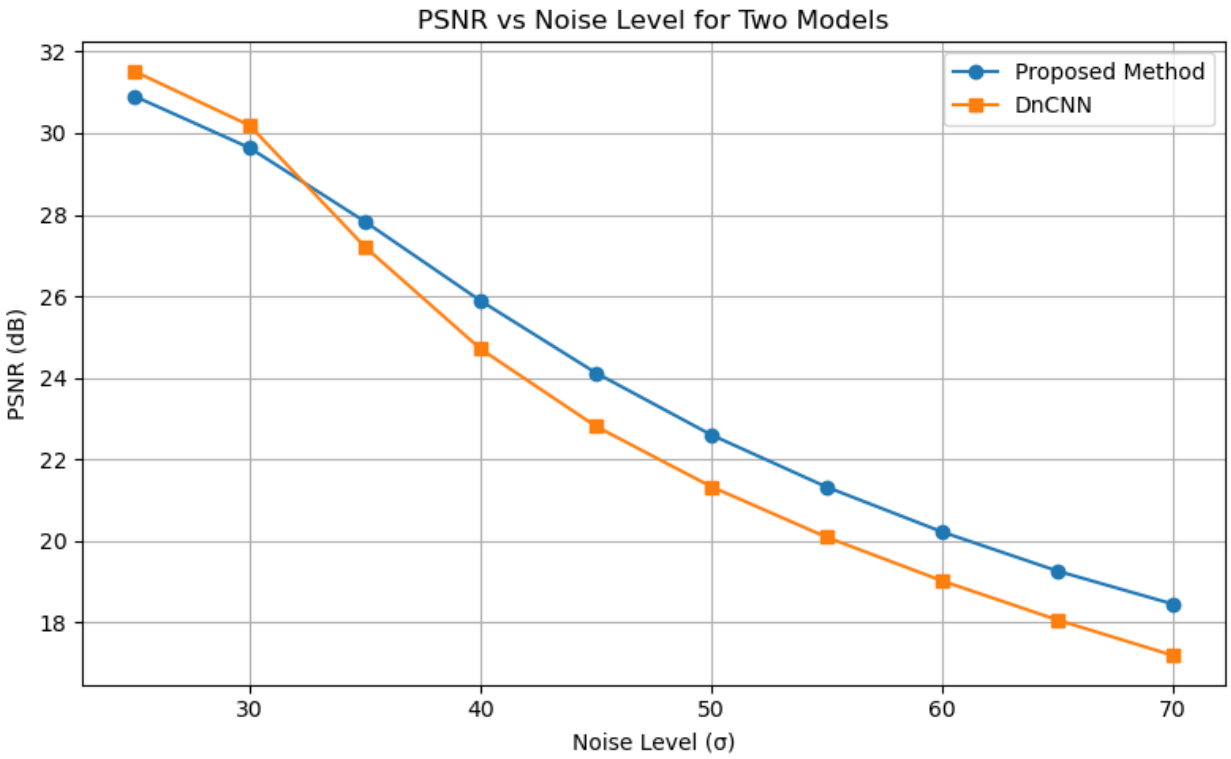


Figure 5.4: PSNR vs. noise level (σ) for the proposed method and DnCNN. Both models were trained at $\sigma = 25$ and tested on the McM dataset with varying noise levels.

Chapter 6

Conclusion

While existing "black-box" deep neural networks have achieved state-of-the-art results in image denoising, their performance often depends on the choice of initialization. Poor initialization often leads to unstable training and suboptimal local minimum convergence. This sensitivity becomes particularly problematic when the test data exhibits differing noise characteristics. Consequently, designing effective initialization schemes is critical to improving generalization and robustness in deep denoising models.

In this thesis, we propose a novel image denoising model that incorporates a performance-guaranteed initialization, integrating graph algorithm unrolling with data-driven parameter learning. Specifically, we first initialize an unrolled network to a known (pseudo-)linear denoiser $\mathbf{\Psi}$, leveraging a previous theorem in [16] that, under mild conditions, maps a denoiser $\mathbf{\Psi}$ to a graph filter that is a solution to a MAP denoising problem regularized using the graph Laplacian regularizer (GLR) [17] defined by Laplacian $\mathbf{L}$. After initialization, to improve performance further using training data, our model employs two separate convolutional neural networks: i) one to learn per-pixel features to construct an appropriate perturbation matrix $\mathbf{P}$ that augments the mapped graph Laplacian $\mathbf{L}$, and ii) another to estimate the regularization parameter μ in a spatially adaptive way. To solve the linear system corresponding to the solution of the graph-regularized MAP problem defined by $\mathbf{L}$, we unroll the iterative conjugate gradient algorithm into neural layers, composing our denoising neural net. Experimental results show encouraging image denoising results: outperforming model-based schemes such as NLM and BM3D [32], while being competitive to deep-learning models such as DnCNN [18].

For future work, we will explore the use of multiple perturbation matrices, building on our finding that even a single learned perturbation matrix can significantly enhance performance. We hypothesize that introducing a set of such matrices may lead to further improvements by better capturing the variability in the data. In addition, we plan to investigate learning the parameters of the conjugate gradient (CG) solver itself, as our preliminary experiments suggest that this can lead to faster convergence and improved reconstruction accuracy.

Bibliography

[1] Yaniv Romano, Michael Elad, and Peyman Milanfar. "The little engine that could: Regularization by denoising (RED)". In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.

[2] Stanley H Chan, Xiran Wang, and Omar A Elgendy. "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications". In: *IEEE Transactions on Computational Imaging* 3.1 (2016), pp. 84–98.

[3] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[4] Carlo Tomasi and Roberto Manduchi. "Bilateral filtering for gray and color images". In: *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, pp. 839–846.

[5] Antoni Buades, Bartomeu Coll, and J-M Morel. "A non-local algorithm for image denoising". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 2. Ieee. 2005, pp. 60–65.

[6] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[7] Michael Elad and Michal Aharon. "Image denoising via learned dictionaries and sparse representation". In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 1. IEEE. 2006, pp. 895–900.

[8] Ivan Markovsky. *Low rank approximation: algorithms, implementation, applications*. Vol. 906. Springer, 2012.

[9] Linwei Fan et al. "Adaptive texture-preserving denoising method using gradient histogram and nonlocal self-similarity priors". In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.11 (2018), pp. 3222–3235.

[10] Xiangjun Zhang and Xiaolin Wu. "Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation". In: *IEEE transactions on image processing* 17.6 (2008), pp. 887–896.

[11] Priyam Chatterjee and Peyman Milanfar. "Is denoising dead?" In: *IEEE Transactions on Image Processing* 19.4 (2009), pp. 895–911.

[12] Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. "Trained transformers learn linear models in-context". In: *arXiv preprint arXiv:2306.09927* (2023).

[13] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning". In: *International conference on machine learning*. PMLR. 2013, pp. 1139–1147.

[14] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[15] Nir Shlezinger et al. "Model-based deep learning". In: *Proceedings of the IEEE* 111.5 (2023), pp. 465–499.

[16] Niruhan Viswarupan et al. "Mixed Graph Signal Analysis of Joint Image Denoising/Interpolation". In: *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2024, pp. 9431–9435.

[17] Jiahao Pang and Gene Cheung. "Graph Laplacian regularization for image denoising: Analysis in the continuous domain". In: *IEEE Transactions on Image Processing* 26.4 (2017), pp. 1770–1785.

[18] Kai Zhang et al. "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising". In: *IEEE transactions on image processing* 26.7 (2017), pp. 3142–3155.

[19] Antonio Ortega et al. "Graph signal processing: Overview, challenges, and applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[20] Wuttipong Kumwilaisak et al. "Image denoising with deep convolutional neural and multi-directional long short-term memory networks under Poisson noise environments". In: *IEEE Access* 8 (2020), pp. 86998–87010.

[21] Bo Fu et al. "A salt and pepper noise image denoising method based on the generative classification". In: *Multimedia Tools and Applications* 78.9 (2019), pp. 12043–12053.

[22] Bogdan Dumitrescu and Paul Irofti. *Dictionary learning algorithms and applications*. Springer, 2018.

[23] Michal Aharon, Michael Elad, and Alfred Bruckstein. "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation". In: *IEEE Transactions on signal processing* 54.11 (2006), pp. 4311–4322.

[24] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition". In: *Proceedings of 27th Asilomar conference on signals, systems and computers*. IEEE. 1993, pp. 40–44.

[25] Stéphane G Mallat and Zhifeng Zhang. "Matching pursuits with time-frequency dictionaries". In: *IEEE Transactions on signal processing* 41.12 (1993), pp. 3397–3415.

[26] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.1 (1996), pp. 267–288.

[27] Xavier Bresson, Tony F Chan, et al. "Fast dual minimization of the vectorial total variation norm and applications to color image processing". In: *Inverse problems and imaging* 2.4 (2008), pp. 455–484.

[28] Fei Chen, Gene Cheung, and Xue Zhang. "Fast & robust image interpolation using gradient graph Laplacian regularizer". In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2021, pp. 1964–1968.

[29] Emmanuel J Candès et al. "Robust principal component analysis?" In: *Journal of the ACM (JACM)* 58.3 (2011), pp. 1–37.

[30] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. "A singular value thresholding algorithm for matrix completion". In: *SIAM Journal on optimization* 20.4 (2010), pp. 1956–1982.

[31] Jiahao Pang et al. "Redefining self-similarity in natural images for denoising using graph signal gradient". In: *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*. IEEE. 2014, pp. 1–8.

[32] Kostadin Dabov et al. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095.

[33] Jingyun Liang et al. "Swinir: Image restoration using swin transformer". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 1833–1844.

[34] Syed Waqas Zamir et al. "Restormer: Efficient transformer for high-resolution image restoration". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 5728–5739.

[35] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[36] A Vaswani. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017).

[37] Alexey Dosovitskiy. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[38] Vishal Monga, Yuelong Li, and Yonina C Eldar. "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing". In: *IEEE Signal Processing Magazine* 38.2 (2021), pp. 18–44.

[39] J Shawe-Taylor. "Kernel Methods for Pattern Analysis". In: *Cambridge University Press google schola* 2 (2004), pp. 181–201.

[40] Gene Cheung et al. "Graph spectral image processing". In: *Proceedings of the IEEE* 106.5 (2018), pp. 907–930.

[41] Gilbert Strang. *Introduction to linear algebra*. SIAM, 2022.

[42] Yuanchao Bai et al. "Graph-based blind image deblurring from a single photograph". In: *IEEE transactions on image processing* 28.3 (2018), pp. 1404–1418.

[43] Xianming Liu et al. "Random walk graph Laplacian-based smoothness prior for soft decoding of JPEG images". In: *IEEE Transactions on Image Processing* 26.2 (2016), pp. 509–524.

[44] Yeganeh Gharedaghi, Gene Cheung, and Xianming Liu. "Retinex-based image denoising/contrast enhancement using gradient graph Laplacian regularizer". In: *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2023, pp. 2710–2714.

[45] Jonathan Richard Shewchuk et al. "An introduction to the conjugate gradient method without the agonizing pain". In: (1994).

[46] Esmail Babolian and Danial Hamedzadeh. "A splitting iterative method for solving second kind integral equations in reproducing kernel spaces". In: *Journal of Computational and Applied Mathematics* 326 (2017), pp. 204–216.

[47] Jin Zeng et al. "Deep graph Laplacian regularization for robust denoising of real images". In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition workshops*. 2019, pp. 0–0.

[48] Huy Vu, Gene Cheung, and Yonina C Eldar. "Unrolling of deep graph total variation for image denoising". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2050–2054.

[49] Rino Yoshida et al. "Unrolling graph total variation for light field image denoising". In: *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2022, pp. 2162–2166.

[50] Peyman Milanfar. "A tour of modern image filtering: New insights and methods, both practical and theoretical". In: *IEEE signal processing magazine* 30.1 (2012), pp. 106–128.

[51] Philip A Knight. "The Sinkhorn–Knopp algorithm: convergence and applications". In: *SIAM Journal on Matrix Analysis and Applications* 30.1 (2008), pp. 261–275.

[52] Philip A Knight. "The Sinkhorn–Knopp algorithm: convergence and applications". In: *SIAM Journal on Matrix Analysis and Applications* 30.1 (2008), pp. 261–275.

[53] Seyed Alireza Hosseini et al. "Constructing an Interpretable Deep Denoiser by Unrolling Graph Laplacian Regularizer". In: *2024 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2024, pp. 1431–1437.

[54] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.