Machine Learning for Image Inverse Problems and Novelty Detection

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Edward Reehorst, B.S., M.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2022

Dissertation Committee:

Philip Schniter, Advisor

Rizwan Ahmad

Lee Potter

Abstract

This dissertation addresses two separate engineering challenges: image-inverse problems and novelty detection.

First, we address image-inverse problems. We review Plug-and-Play (PnP) algorithms, where a proximal operator is replaced by a call of an arbitrary denoising algorithm. We apply PnP algorithms to compressive Magnetic Resonance Imaging (MRI). MRI is a non-invasive diagnostic tool that provides excellent soft-tissue contrast without the use of ionizing radiation. However, when compared to other clinical imaging modalities (e.g., CT or ultrasound), the data acquisition process for MRI is inherently slow, which motivates undersampling and thus drives the need for accurate, efficient reconstruction methods from undersampled datasets. We apply the PnP-ADMM algorithm to cardiac MRI and knee MRI data. For these algorithms, we developed learned denoisers that can process complex-valued MRI images. Our algorithms achieve state-of-the-art performance on both the cardiac and knee datasets.

Regularization by Denoising (RED), as proposed by Romano, Elad, and Milanfar, is a powerful image-recovery framework that aims to minimize an explicit regularization objective constructed from a plug-in image-denoising function. Experimental evidence suggests that RED algorithms are state-of-the-art. We claim, however, that explicit regularization does not explain the RED algorithms. In particular, we show

that many of the expressions in the paper by Romano et al. hold only when the denoiser has a symmetric Jacobian, and we demonstrate that such symmetry does not occur with practical denoisers such as non-local means, BM3D, TNRD, and DnCNN. To explain the RED algorithms, we propose a new framework called Score-Matching by Denoising (SMD), which aims to match a "score" (i.e., the gradient of a log-prior).

Novelty detection is the ability for a machine learning system to detect signals that are significantly different from samples seen during training. Detecting novelties is important in any problem where it is possible for the system to encounter data that is significantly different from the training data. In recent years, novelty detection in high dimensional signals has been an area of significant research, with many methods proposed. In order to leverage this tremendous body of work, we develop, Shift-Ensembled Novelty Detection (SEND), a framework for combining multiple novelty detection methods. Our experiments show SEND achieves state-of-the-art performance for novelty detection of images.

We also apply SEND to the problem of radio frequency (RF) waveform novelty detection. Recently, there has been tremendous research interest in utilizing self-supervised neural networks for novelty detection. These methods have largely been developed for image applications. In this work, we adapt SimCLR and CSI, two popular self-supervised learning methods, to operate with RF waveform data. Our experiments for RF waveform novelty detection show a significant increase in performance over state-of-the-art methods.

Dedicated to my wife, Kelsey, for your constant love and support.

Acknowledgments

I would like to thank and acknowledge several people and organizations who have played an important role in the completion of this dissertation. I would like to thank all of the Ohio State ECE faculty for their instruction and for developing an exciting place to learn. Thank you to my advisor, Phil Schniter, for your mentorship, helpful insights, and constant support. I also want to thank Rizwan Ahmad, Emre Ertin, Kiryung Lee, Yingbin Liang, and Lee Potter for lending their technical expertise by serving on exam committees over the course of my graduate education.

This work was funded by the National Science Foundation (under grant numbers 1527162, 1716388, 1955587), National Institute of Health (under grant number R01HL135489), the Center for Surveillance Research (under grant number 1539961), and Air Force Research Laboratory (under grant number FA8650-32-C-1174). This work would not have been possible without them.

At Air Force Research Lab, I would like to thank Joseph Meola, Jacob Martin, Nathan Wurst, Michael Rucci, Philip Maciejewski, Edmund Zelnio, Anne Pavy, Michael Wharton, and Christopher Ebersole for their encouragement and support during the writing of this dissertation.

Thank you to my friends and family for being supportive through this process. I could not have done it without you.

Vita

2016	B.S. Electrical and Computer Engineering, The Ohio State University
2022	M.S. Electrical and Computer Engineering, The Ohio State University
2016-2021	Graduate Research Assistant, The Ohio State University
2021-present	Electronics Engineer, Air Force Research Laboratory.

Publications

Research Publications

E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Trans. Comp. Imag.*, vol. 5, pp. 52–67, Mar. 2019

R. Ahmad, C. A. Bouman, G. T. Buzzard, S. Chan, S. Liu, E. T. Reehorst, and P. Schniter, "Plug and play methods for magnetic resonance imaging," *IEEE Signal Process. Mag.*, vol. 37, no. 1, pp. 105–116, 2020

S. Liu, E. Reehorst, P. Schniter, and R. Ahmad, "Free-breathing cardiovascular MRI using a Plug-and-Play method with learned denoiser," in *Proc. IEEE Int. Symp. Biomed. Imag.*, Apr. 2020

M. Wharton, E. T. Reehorst, and P. Schniter, "Compressive SAR image recovery and classification via CNNs," in *Proc. Asilomar Conf. Signals Syst. Comput.*, pp. 1081–1085, 2019

Fields of Study

Major Field: Electrical and Computer Engineering

Studies in:

Machine Learning
Signal Processing

Table of Contents

Page

Abstract ii

Dedication iv

Acknowledgments v

Vita vi

List of Tables xi

List of Figures xii

1. Introduction 1

    1.1 Image Inverse Problems 1
    1.2 Novelty Detection 3

2. Plug-and-Play Methods for Magnetic Resonance Imaging 5

    2.1 Introduction 5
    2.2 Image recovery in compressive MRI 8
    2.3 Signal Recovery and Denoising 10
    2.4 Plug-and-Play Methods 12
        2.4.1 Prox-based PnP 12
        2.4.2 The balanced FISTA approach 16
    2.5 Understanding PnP through Consensus Equilibrium 17
    2.6 Demonstration of PnP in MRI 20
        2.6.1 Parallel cardiac MRI 20
        2.6.2 Single-coil fastMRI knee data 24
    2.7 Conclusion 29

3. Regularization by Denoising: Clarifications and New Interpretations . . . 30

3.1 Introduction . . . 30
3.2 Background . . . 33
3.2.1 The MAP-Bayesian Interpretation . . . 33
3.2.2 ADMM . . . 33
3.2.3 Plug-and-Play ADMM . . . 34
3.2.4 Regularization by Denoising (RED) . . . 36
3.3 Clarifications on RED . . . 37
3.3.1 Preliminaries . . . 37
3.3.2 The RED Gradient . . . 38
3.3.3 Impossibility of Explicit Regularization . . . 39
3.3.4 Analysis of Jacobian Symmetry . . . 41
3.3.5 Jacobian Symmetry Experiments . . . 42
3.3.6 Local Homogeneity Experiments . . . 44
3.3.7 Hessian and Convexity . . . 47
3.3.8 Example RED-SD Trajectory . . . 48
3.3.9 Visualization of RED Cost and RED-Algorithm Gradient . 49
3.4 Score-Matching by Denoising . . . 49
3.4.1 Tweedie Regularization . . . 50
3.4.2 Tweedie Regularization as Kernel Density Estimation . . . 53
3.4.3 Score-Matching by Denoising . . . 54
3.4.4 Relation to Existing Work . . . 56
3.5 Fast RED Algorithms . . . 57
3.5.1 RED-ADMM . . . 57
3.5.2 Inexact RED-ADMM . . . 58
3.5.3 Majorization-Minimization and Proximal-Gradient RED . . 60
3.5.4 Dynamic RED-PG . . . 62
3.5.5 Accelerated RED-PG . . . 63
3.5.6 Convergence of RED-PG . . . 64
3.5.7 Algorithm Comparison: Image Deblurring . . . 67
3.6 Equilibrium View of RED Algorithms . . . 71
3.6.1 RED Equilibrium Conditions . . . 72
3.6.2 Interpreting the RED Equilibria . . . 74
3.7 Conclusion . . . 76

4. SEND: A Score Ensembling Strategy for Novelty Detection . . . 78

4.1 Introduction . . . 78
4.2 Background . . . 81
4.2.1 Low-Dimensional Novelty Detection . . . 81

4.2.2 High-Dimensional Novelty Detection . . . 81
4.3 Ensembling of Inlier Scores . . . 83
4.3.1 Generalized Likelihood Ratio Test . . . 85
4.3.2 Covariance Matrix Selection . . . 87
4.3.3 END Analysis: White Gaussian Case . . . 91
4.3.4 END Analysis: Redundant Scores . . . 92
4.4 Shift Ensembled Novelty Detection . . . 95
4.4.1 Inlier Scores . . . 96
4.4.2 Ensembles . . . 98
4.5 Experiments . . . 99
4.5.1 Setup . . . 99
4.5.2 Model Training . . . 99
4.5.3 Simple Ensemble Experiments . . . 100
4.5.4 Shift Ensemble Experiments . . . 101
4.5.5 Large Ensemble Experiments . . . 102
4.6 Summary . . . 102

5. Novelty Detection for RF Waveforms with Ensembled Contrastive Learning 104

5.1 Introduction . . . 104
5.2 Background . . . 106
5.2.1 Low-Dimensional Novelty Detection . . . 106
5.2.2 High-Dimensional Novelty Detection . . . 106
5.2.3 Self-Supervised Learning . . . 108
5.2.4 Self-Supervised Learning for Novelty Detection . . . 110
5.3 Contrastive Learning for RF Waveform Data . . . 114
5.3.1 Similarity Transforms . . . 114
5.3.2 Shifting Transforms . . . 117
5.4 SEND for RF Waveforms . . . 118
5.5 Experiments . . . 120
5.5.1 Datasets . . . 120
5.5.2 Evaluation . . . 121
5.5.3 Model Training . . . 121
5.5.4 Baseline Methods . . . 122
5.5.5 Novelty Detector Performance . . . 123
5.6 Summary . . . 124

Bibliography . . . 125

List of Tables

Table **Page**

2.1 rSNR (dB) of MRI cardiac cine recovery from four test datasets. . . . 25

2.2 rSNR and SSIM for fastMRI single-coil test data with $R = 4$. 29

3.1 Average Jacobian-symmetry error on 16×16 images 43

3.2 Average gradient error on 16×16 images 44

3.3 Average local-homogeneity error on 16×16 images 45

4.1 Results from the simple ensembling experiments. 100

4.2 Results from the shift ensembling experiments. 101

4.3 Results from the large ensembling experiments. 102

5.1 Performance of Linear classifier trained on SimCLR features for SIDLE and GNURadio PSK signals. 117

5.2 Novelty detection performance for various techniques on the SIDLE and GNURadio waveform datasets. Area under the receiver operating curve is labeled as AUC and detection probability is labeled DPR. DPR is evaluated at 1% false alarm rate for the SIDLE data and 5% false alarm rate for the GNURadio data. 123

List of Figures

Figure **Page**

2.1 The architecture of the CNN-based cardiac cine denoiser operating on spatiotemporal volumetric patches. 22

2.2 Two different views of the 3D sampling pattern used to retrospectively undersample one of the four test datasets at $R = 10$. The undersampling was performed only in the phase encoding direction and the pattern was varied across frames. In this example, the number of frequency encoding steps, phase encoding steps, and frames are 192, 144, and 25, respectively. 25

2.3 Results from cardiac cine Dataset #1 at $R = 10$. Top row: a representative frame from the fully sampled reference and various recovery methods. The green arrow points to an image feature that is preserved only by PnP-CNN and not by other methods. Middle row: error map $\times 6$. Bottom row: temporal frame showing the line drawn horizontally through the middle of the image in the top row, with the time dimension along the horizontal axis. The arrows point to the movement of the papillary muscles, which are more well-defined in PnP-CNN. . . . 26

2.4 NMSE versus iteration for two PnP and two CS algorithms on the cardiac cine recovery Dataset #3 at $R = 10$. 26

2.5 For PnP-CNN recovery of cardiac cine Dataset #3 at $R = 10$, the change in the final NMSE (after 100 iterations) as a function of σ^2/ν (left) and the NMSE versus iteration for several σ^2/ν (right). 27

2.6 NMSE versus time PnP ADMM with different numbers of CG and GD inner iterations on cardiac cine Dataset #3 at $R = 10$. 27

3.1 RED cost $C_{\mathsf{red}}(\boldsymbol{x}_k)$ and fixed-point error $\|\boldsymbol{A}^{\mathsf{T}}(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})/\sigma^2 + \lambda(\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_k))\|^2$ versus iteration k for $\{\boldsymbol{x}_k\}_{k=1}^K$ produced by the RED-SD algorithm from [5]. Although the fixed-point condition is asymptotically satisfied, the RED cost does not decrease with k. 48

3.2 Contours show RED cost $C_{\mathsf{red}}(\boldsymbol{x}_{\alpha,\beta})$ from (3.14) and arrows show RED-algorithm gradient field $\boldsymbol{g}(\boldsymbol{x}_{\alpha,\beta})$ from (3.46) versus (α, β), where $\boldsymbol{x}_{\alpha,\beta} = \widehat{\boldsymbol{x}} + \alpha\boldsymbol{e}_1 + \beta\boldsymbol{e}_2$ with randomly chosen $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. The subplots show that the minimizer of $C_{\mathsf{red}}(\boldsymbol{x}_{\alpha,\beta})$ is not the fixed-point $\widehat{\boldsymbol{x}}$, and that $C_{\mathsf{red}}(\cdot)$ may be non-smooth and/or non-convex. 50

3.3 PSNR versus runtime for RED-ADMM with TNRD denoising and I inner iterations. 59

3.4 PSNR versus iteration for RED algorithms with TNRD denoising when deblurring the starfish. 66

3.5 Fixed-point error versus iteration for RED algorithms with TNRD denoising when deblurring the starfish. 67

3.6 Update distance versus iteration for RED algorithms with TNRD denoising when deblurring the starfish. 68

3.7 PSNR versus iteration for RED algorithms with TDT denoising when deblurring the starfish. 69

3.8 Fixed-point error versus iteration for RED algorithms with TDT denoising when deblurring the starfish. 70

3.9 Update distance versus iteration for RED algorithms with TDT denoising when deblurring the starfish. 71

4.1 Example images of tanks used to train a DNN tank classifier and a test sample, that we would not like classified as a type of tank. Images are from the CIFAR-100 dataset [6]. 80

4.2 Example plots of (left) one-class classification and (right) multi-class novelty detection problems. 80

4.3 Diagrams of classic, low-dimensional (left) and deep, high-dimensional (right) novelty detection. Deep novelty detection utilizes a deep neural network to extract features before evaluating the score function $s(\cdot)$. . 81

4.4 Diagram of CSI ensembling technique. Our test sample $\boldsymbol{x}$ is transformed by n_r shifting transforms, $\{\boldsymbol{R}_i(\cdot)\}_{i=1}^{n_r}$. These shifted versions of the test sample are input to the feature encoder network $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$, which outputs corresponding feature vectors $\{\boldsymbol{z}_i\}_{i=1}^{n_r}$. Each feature vector is scored by n_s inlier scores, $\{s_i(\cdot)\}_{i=1}^{n_s}$. Finally, all the scores are ensembled to get one final score, which is thresholded to perform novelty detection. 83

4.5 Left: Raw "norm" and "cos" scores from [7] evaluated CIFAR-10 training inliers. Center: Quantile transformed versions of the scores from the left plot. Right: Plot of "norm" score (x-axis) versus "cos" score (y-axis) for CIFAR-10 test inliers and SVHN test novelties. Contours show the value of END score from (4.33). 86

4.6 Plots of eigen-score performance (in AUC) vs. eigenvalue for two 24-score experiments. We can see that there is a strong correlation between how informative a direction is for novelty detection with its eigenvalue. Left plot has CIFAR-10 as the inliers and SVHN samples as the novelties. The right plot has SVHN as the inliers and CIFAR-10 as the novelties. A line is fit in the log space of the eigenvalues. The R^2 value of this regression is given on each plot. 89

4.7 Sub-plots (a)-(c) show the AUC versus the novelty mean components α_1 and α_2. Sub-plots (d)-(e) show the AUC of END minus the AUC of the genie and single score detectors, respectively. 92

4.8 Performance of single score plotted against the END ensemble of redundant scores when (left) $\alpha = -0.6$ and (right) $\alpha = -2.0$. 93

4.9 Results from the redundant score experiment with three base scores. Sub-plots (a)-(c) show the AUC versus the novelty mean components α_1 and α_2. Sub-plots (d)-(e) show the AUC of END minus the AUC of the genie and the GLRT score with $\boldsymbol{\Sigma} = \widehat{\boldsymbol{\Sigma}}$, respectively. 94

5.1 Distribution of features where encoders are trained with (left) normal SimCLR and (right) SimCLR augmented with shifting transforms. We can see that by utilizing shifting transforms, novelty detection becomes easier. Figure from [8] 110

5.2 CSI uses two head networks, a SimCLR head, $\boldsymbol{z}_{\boldsymbol{\theta}}(\cdot)$, and a shift-classifier head, $\boldsymbol{\ell}_{\boldsymbol{\theta}}(\cdot)$, which both utilize a common feature encoder network, $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$. These networks are jointly trained to minimize $\mathcal{L}_{CSI}$. The final CSI score performs ensembling over the shift score, norm score, and cos score. 114

Chapter 1: Introduction

In this dissertation, we discuss two somewhat disjoint problems: image inverse problems and novelty detection.

1.1 Image Inverse Problems

Image inverse problems occur when we want to reconstruct an unknown image $\boldsymbol{x}^0 \in \mathbb{R}^n$, from measurements of the form

$$\boldsymbol{y} = \boldsymbol{g}(\mathcal{A}(\boldsymbol{x}^0)), \tag{1.1}$$

where $\boldsymbol{y} \in \mathbb{R}^m$ is our measurement vector, $\mathcal{A}(\cdot)$ is the measurement operator and $\boldsymbol{g}(\cdot)$ is some form of corruption, like noise. In this dissertation, we discuss a simplified problem where $\mathcal{A}(\cdot)$ is a linear operator and the corruption is additive white Gaussian noise. Under these assumptions, the model simplifies to

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^0 + \boldsymbol{w}, \tag{1.2}$$

where $\boldsymbol{A} \in \mathcal{R}^{m \times n}$ and $\boldsymbol{w} \in \mathbb{R}^m$ is white Gaussian noise. In many problems of interest, $\text{rank}(\boldsymbol{A}) < n$, resulting in an under-determined problem. Our linear model, (1.2), can apply to many problems including deblurring, denoising, inpainting, and compressive Magnetic Resonance Imaging (MRI).

In chapter 2, we will specifically be looking at compressive MRI. MRI is a valuable medical diagnostic tool because of its high soft-tissue contrast compared to other imaging methods. The main downsides of MRI are that machines are expensive and capture times are slow. For static imaging, these long capture times can be prohibitive from a cost and patient well-being standpoint. For dynamic applications, these capture times are simply not quick enough to capture fast moving processes, such as a cardiac cycle. The need for faster collection times leads to the practice of undersampling the data in an effort to speed up collection.

In this dissertation, we discuss image-inverse algorithms that utilize an image "denoiser" within an iterative reconstruction algorithm. This image denoiser can be any technique designed to recover $\boldsymbol{x}$ from noisy measurements of the form $\boldsymbol{r} = \boldsymbol{x} + \boldsymbol{w}$ where $\boldsymbol{w}$ is i.i.d. zero-mean Gaussian noise. These algorithms could be a classical algorithm, such as non-local means [9], or a machine learning approach such as Denoising Convolutional Neural Network (DnCNN) [10]. In chapter 2, we review the popular "Plug-and-Play" (PnP) framework and apply it to dynamic cardiac MRI and static knee MRI. In PnP algorithms, a proximal operator on the prior of a signal distribution is replaced by a call to an arbitrary denoiser. Our proposed MRI recovery algorithms utilize PnP-ADMM, with learned, complex-valued denoisers.

In chapter 3, we look at another class of methods based on the Regularization by Denoising (RED) [5] framework. With specific denoisers, Romano et al. [5] designed a regularization term that utilizes a denoiser, has a gradient that is easy to compute, and results in algorithms that perform well at standard inverse problems, such as inpainting, deblurring, and super-resolution. However, as we discuss in this dissertation, these claims are limited to a small number of denoisers. In fact, for

most denoisers tested, the gradient reported by [5] corresponds to a non-conservative vector field. In these cases there is no scalar function that has the corresponding gradient. In spite of these limitations, algorithms designed around the RED gradient perform well in practice. In chapter 3, we also discuss an alternative explanation for these RED algorithms from the standpoint of score-matching [11].

1.2 Novelty Detection

Novelty detection is the ability to detect a signal that is semantically different from some known distribution. Interest in this problem spans many application areas: autonomous vehicles, computer vision, Radio-Frequency (RF) waveforms, and fraud detection. The ability to detect signals that differ from some definition of normal is important to the design of trusted-AI systems.

Most novelty detection algorithms utilize an inlier score. This inlier score provides a confidence in a given sample being from the inlier distribution. We can threshold this score to perform novelty detection. If the inlier score is above the threshold, we deem the sample an inlier, and if the score is below the threshold, we label it a novelty. In chapter 4, we formulate Shift-Ensembled Novelty Detection (SEND), a framework for ensembling multiple inlier scores. By utilizing SEND, we can ensemble multiple popular, state-of-the-art inlier scores into one method. We test our method on several standard image datasets, where these ensembles are able to achieve state-of-the-art performance.

In chapter 5, we apply SEND and other recent advancements in novelty detection to the problem of RF waveform novelty detection. One of these recent advances is the utilization of self-supervised feature encoding networks [7,8,12] for novelty detection.

Research has shown that self-supervised techniques can provide better separation between inlier and novelty samples [7,8,13]. However, self-supervised learning techniques have primarily been developed for image data. A main contribution of this work is adapting these techniques to work with waveform data. We utilize this self-supervised feature encoder to evaluate several inlier scores, which are ensembled using SEND. In experiments on communication and radar waveform data, our SEND-based method significantly outperforms other state-of-the-art methods for RF waveform novelty detection.

Chapter 2: Plug-and-Play Methods for Magnetic Resonance Imaging

2.1 Introduction

In this chapter[1], we apply the plug-and-play framework [14] to the problem of compressive Magnetic Resonance Imaging (MRI). MRI uses radiofrequency waves to non-invasively evaluate the structure, function, and morphology of soft tissues. MRI has become an indispensable imaging tool for diagnosing and evaluating a host of conditions and diseases. Compared to other clinical imaging modalities (e.g., CT or ultrasound), however, MRI suffers from slow data acquisition. A typical clinical MRI exam consists of multiple scans and can take more than an hour to complete. For each scan, the patient may be asked to stay still for several minutes, with slight motion potentially resulting in image artifacts. Furthermore, dynamic applications demand collecting a series of images in quick succession. Due to the limited time window in many dynamic applications (e.g., contrast enhanced MR angiography), it is not feasible to collect fully sampled datasets. For these reasons, MRI data is often undersampled. Consequently, computationally efficient methods to recover

[1]This chapter is largely excerpted from the manuscript [2] co-authored with Rizwan Ahmad, Charles Bouman, Gregery Buzzard, Stanley Chan, Sizhuo Liu, and Philip Schniter.

high-quality images from undersampled MRI data have been actively researched for the last two decades.

The combination of parallel (i.e., multi-coil) imaging and compressive sensing (CS) has been shown to benefit a wide range of MRI applications [15,16], including dynamic applications, and has been included in the default image processing frameworks offered by several major MRI vendors. More recently, learning-based techniques (e.g., [17–21]) have been shown to outperform CS methods. Some of these techniques learn the entire end-to-end mapping from undersampled k-space or aliased images to recovered images (e.g., [19,22]). Considering that the forward model in MRI changes from one dataset to the next, such methods have to be either trained over a large and diverse data corpus or limited to a specific application. Other methods train scan-specific convolutional neural networks (CNN) on a fully-sampled region of k-space and then use it to interpolate missing k-space samples [20]. These methods do not require separate training data but demand a fully sampled k-space region. Due to the large number of unknowns in CNN, such methods require a fully sampled region that is larger than that typically acquired in parallel imaging, limiting the acceleration that can be achieved. Other supervised learning methods are inspired by classic variational optimization methods and iterate between data-consistency enforcement and a trained CNN, which acts as a regularizer [18]. Such methods require a large number of fully sampled, multi-coil k-space datasets, which may be difficult to obtain in many applications. Also, since CNN training occurs in the presence of dataset-specific forward models, generalization from training to test scenarios remains an open question [21]. Other learning-based methods have been proposed based on

bi-level optimization (e.g., [23]), adversarially learned priors (e.g., [24]), and autoregressive priors (e.g., [25]). Consequently, the integration of learning-based methods into physical inverse problems remains a fertile area of research. There are many directions for improvement, including recovery fidelity, computational and memory efficiency, robustness, interpretability, and ease-of-use.

This chapter focuses on "plug-and-play" (PnP) algorithms [14], which alternate image denoising with forward-model based signal recovery. PnP algorithms facilitate the use of state-of-the-art image models through their manifestations as image denoisers, whether patch-based (e.g., [26]) or deep neural network (DNN) based (e.g., [10]). The fact that PnP algorithms decouple image modeling from forward modeling has advantages in compressive MRI, where the forward model can change significantly among different scans due to variations in the coil sensitivity maps, sampling patterns, and image resolution. Furthermore, fully sampled k-space MRI data is not needed for PnP; the image denoiser can be learned from MRI image patches, or possibly even magnitude-only patches. The objective of this chapter is two-fold: i) to review recent advances in plug-and-play methods, and ii) to discuss their application to compressive MRI image reconstruction.

The remainder of the chapter is organized as follows. We first detail the inverse problem encountered in MRI reconstruction. We then review several PnP methods, where the unifying commonality is to iteratively call a denoising subroutine as one step of a larger optimization-inspired algorithm. We also review how the result of the PnP method can be interpreted as a solution to an equilibrium equation, allowing convergence analysis from the equilibrium perspective. We then apply PnP

with learned denoisers to dynamic cardiac MRI and static knee MRI datasets. This includes the development of 2D and 3D, complex-valued denoisers.

2.2 Image recovery in compressive MRI

In this section, we describe the standard linear inverse problem formulation in MRI. We acknowledge that more sophisticated formulations exist (see, e.g., [27] for a more careful modeling of physics effects). Briefly, the measurements are samples of the Fourier transform of the image, where the Fourier domain is often referred to as "k-space." The transform can be taken across two or three spatial dimensions and includes an additional temporal dimension in dynamic applications. Furthermore, measurements are often collected in parallel from $C \geq 1$ receiver coils. In dynamic parallel MRI with Cartesian sampling, the time-t k-space measurements from the i^{th} coil take the form

$$\boldsymbol{y}_i^{(t)} = \boldsymbol{P}^{(t)} \boldsymbol{F} \boldsymbol{S}_i \boldsymbol{x}^{(t)} + \boldsymbol{w}_i^{(t)}, \tag{2.1}$$

where $\boldsymbol{x}^{(t)} \in \mathbb{C}^N$ is the vectorized 2D or 3D image at discrete time t, $\boldsymbol{S}_i \in \mathbb{C}^{N \times N}$ is a diagonal matrix containing the sensitivity map for the i^{th} coil, $\boldsymbol{F} \in \mathbb{C}^{N \times N}$ is the 2D or 3D discrete Fourier transform (DFT), the sampling matrix $\boldsymbol{P}^{(t)} \in \mathbb{R}^{M \times N}$ contains M rows of the $N \times N$ identity matrix, and $\boldsymbol{w}_i^{(t)} \in \mathbb{C}^M$ is additive white Gaussian noise (AWGN). Often the sampling pattern changes across frames t. The MRI literature often refers to $R \triangleq N/M$ as the "acceleration rate." The AWGN assumption, which does not hold for the measured parallel MRI data, is commonly enforced using noise pre-whitening filters, which yields the model (2.1) but with diagonal "virtual" coil maps $\boldsymbol{S}_i$ [28]. Additional justification of the AWGN model can be found in [29].

MRI measurements are acquired using a sequence of measurement trajectories through k-space. These trajectories can be Cartesian or non-Cartesian in nature. Cartesian trajectories are essentially lines through k-space. In the Cartesian case, one k-space dimension (i.e., the frequency encoding) is fully sampled, while the other one or two dimensions (i.e., the phase encodings) are undersampled to reduce acquisition time. Typically, one line, or "readout," is collected after every RF pulse, and the process is repeated several times to collect adequate samples of k-space. Non-Cartesian trajectories include radial or spiral curves, which have the effect of distributing the samples among all dimensions of k-space. Compared to Cartesian sampling, non-Cartesian sampling provides more efficient coverage of k-space and yields an "incoherent" forward operator that is more conducive to compressed-sensing reconstruction [30]. But Cartesian sampling remains the method of choice in clinical practice, due to its higher tolerance to system imperfections and an extensive record of success.

Since the sensitivity map, $\boldsymbol{S}_i$, is patient-specific and varies with the location of the coil with respect to the imaging plane, both $\boldsymbol{S}_i$ and $\boldsymbol{x}^{(t)}$ are unknown in practice. Although calibration-free methods have been proposed to estimate $\boldsymbol{S}_i\boldsymbol{x}^{(t)}$ (e.g., [31]) or to jointly estimate $\boldsymbol{S}_i$ and $\boldsymbol{x}^{(t)}$ (e.g., [32]), it is more common to first estimate $\boldsymbol{S}_i$ through a calibration procedure and then treat $\boldsymbol{S}_i$ as known in (2.1). Stacking $\{\boldsymbol{y}_i^{(t)}\}$, $\{\boldsymbol{x}^{(t)}\}$, and $\{\boldsymbol{w}_i^{(t)}\}$ into vectors $\boldsymbol{y}$, $\boldsymbol{x}$, and $\boldsymbol{w}$, and packing $\{\boldsymbol{P}^{(t)}\boldsymbol{F}\boldsymbol{S}_i\}$ into a known block-diagonal matrix $\boldsymbol{A}$, we obtain the linear inverse problem of recovering $\boldsymbol{x}$ from

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}, \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}), \tag{2.2}$$

where $\mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$ denotes a circularly symmetric complex-Gaussian random vector.

2.3 Signal Recovery and Denoising

The maximum likelihood (ML) estimate of $\boldsymbol{x}$ from $\boldsymbol{y}$ in (2.2) is $\widehat{\boldsymbol{x}}_{\mathsf{ml}} \triangleq \arg\max_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x})$, where $p(\boldsymbol{y}|\boldsymbol{x})$, the probability density of $\boldsymbol{y}$ conditioned on $\boldsymbol{x}$, is known as the "likelihood function." The ML estimate is often written in the equivalent form $\widehat{\boldsymbol{x}}_{\mathsf{ml}} = \arg\min_{\boldsymbol{x}}\{-\ln p(\boldsymbol{y}|\boldsymbol{x})\}$. In the case of σ^2-variance AWGN $\boldsymbol{w}$, we have that $-\ln p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 + \text{const}$, and so $\widehat{\boldsymbol{x}}_{\mathsf{ml}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$, which can be recognized as least-squares estimation. Although least-squares estimation can give reasonable performance when $\boldsymbol{A}$ is tall and well conditioned, this is rarely the case under moderate to high acceleration (i.e., $R > 2$). With acceleration, it is critically important to exploit prior knowledge of signal structure.

The traditional approach to exploiting such prior knowledge is to formulate and solve an optimization problem of the form

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 + \phi(\boldsymbol{x}) \right\}, \tag{2.3}$$

where the regularization term $\phi(\boldsymbol{x})$ encodes prior knowledge of $\boldsymbol{x}$. In fact, $\widehat{\boldsymbol{x}}$ in (2.3) can be recognized as the maximum a posteriori (MAP) estimate of $\boldsymbol{x}$ under the prior density model $p(\boldsymbol{x}) \propto \exp(-\phi(\boldsymbol{x}))$. To see why, recall that the MAP estimate maximizes the posterior distribution $p(\boldsymbol{x}|\boldsymbol{y})$. That is, $\widehat{\boldsymbol{x}}_{\mathsf{map}} \triangleq \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y}) = \arg\min_{\boldsymbol{x}}\{-\ln p(\boldsymbol{x}|\boldsymbol{y})\}$. Since Bayes' rule implies that $\ln p(\boldsymbol{x}|\boldsymbol{y}) = \ln p(\boldsymbol{y}|\boldsymbol{x}) + \ln p(\boldsymbol{x}) - \ln p(\boldsymbol{y})$, we have

$$\widehat{\boldsymbol{x}}_{\mathsf{map}} = \arg\min_{\boldsymbol{x}} \big\{ -\ln p(\boldsymbol{y}|\boldsymbol{x}) - \ln p(\boldsymbol{x}) \big\}. \tag{2.4}$$

Recalling that the first term in (2.3) (i.e., the "loss" term) was observed to be $-\ln p(\boldsymbol{y}|\boldsymbol{x})$ (plus a constant) under AWGN noise, the second term in (2.3) must

obey $\phi(\boldsymbol{x}) = -\ln p(\boldsymbol{x}) + \text{const}$. We will find this MAP interpretation useful in the sequel.

It is not easy to design good regularizers ϕ for use in (2.3). They must not only mimic the negative log signal-prior, but also enable tractable optimization. One common approach is to use $\phi(\boldsymbol{x}) = \lambda\|\boldsymbol{\Psi}\boldsymbol{x}\|_1$ with $\boldsymbol{\Psi}^{\mathsf{H}}$ a tight frame (e.g., a wavelet transform) and $\lambda > 0$ a tuning parameter [33]. Such regularizers are convex, and the ℓ_1 norm rewards sparsity in the transform outputs $\boldsymbol{\Psi}\boldsymbol{x}$ when used with the quadratic loss. One could go further and use the composite penalty $\phi(\boldsymbol{x}) = \sum_{l=1}^{D} \lambda_l\|\boldsymbol{\Psi}_l\boldsymbol{x}\|_1$. Due to the richness of data structure in MRI, especially for dynamic applications, utilizing multiple ($D > 1$) linear sparsifying transforms has been shown to improve recovery quality [34], but tuning multiple regularization weights $\{\lambda_l\}$ is a non-trivial problem [35].

Particular insight comes from considering the special case of $\boldsymbol{A} = \boldsymbol{I}$, where the measurement vector in (2.2) reduces to an AWGN-corrupted version of the image $\boldsymbol{x}$, i.e.,

$$\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{w}, \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}). \tag{2.5}$$

The problem of recovering $\boldsymbol{x}$ from noisy $\boldsymbol{z}$, known as "denoising," has been intensely researched for decades. While it is possible to perform denoising by solving a regularized optimization problem of the form (2.3) with $\boldsymbol{A} = \boldsymbol{I}$, today's state-of-the-art approaches are either algorithmic (e.g., [26]) or DNN-based (e.g., [10]). This begs an important question: can these state-of-the-art denoisers be leveraged for MRI signal reconstruction, by exploiting the connections between the denoising problem and (2.3)? As we shall see, this is precisely what the PnP methods do.

2.4 Plug-and-Play Methods

In this section, we review several approaches to PnP signal reconstruction. What these approaches have in common is that they recover $\boldsymbol{x}$ from measurements $\boldsymbol{y}$ of the form (2.2) by iteratively calling a sophisticated denoiser within a larger optimization or inference algorithm.

2.4.1 Prox-based PnP

To start, let us imagine how the optimization in (2.3) might be solved. Through what is known as "variable splitting," we could introduce a new variable, $\boldsymbol{v}$, to decouple the regularizer $\phi(\boldsymbol{x})$ from the data fidelity term $\frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$. The variables $\boldsymbol{x}$ and $\boldsymbol{v}$ could then be equated using an external constraint, leading to the constrained minimization problem

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{C}^N}{\arg\min} \underset{\boldsymbol{v}\in\mathbb{C}^N}{\min} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 + \phi(\boldsymbol{v}) \right\} \quad \text{subject to} \quad \boldsymbol{x}=\boldsymbol{v}. \tag{2.6}$$

Equation (2.6) suggests an algorithmic solution that alternates between separately estimating $\boldsymbol{x}$ and estimating $\boldsymbol{v}$, with an additional mechanism to asymptotically enforce the constraint $\boldsymbol{x}=\boldsymbol{v}$.

The original PnP method [14] is based on the alternating direction method of multipliers (ADMM) [36]. For ADMM, (2.6) is first reformulated as the "augmented Lagrangian"

$$\min_{\boldsymbol{x},\boldsymbol{v}} \max_{\boldsymbol{\lambda}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 + \phi(\boldsymbol{v}) + \mathrm{Re}\{\boldsymbol{\lambda}^{\mathsf{H}}(\boldsymbol{x}-\boldsymbol{v})\} + \frac{1}{2\nu}\|\boldsymbol{x}-\boldsymbol{v}\|^2 \right\}, \tag{2.7}$$

where $\boldsymbol{\lambda}$ are Lagrange multipliers and $\nu>0$ is a penalty parameter that affects the convergence speed of the algorithm, but not the final solution. With $\boldsymbol{u} \triangleq \nu\boldsymbol{\lambda}$, (2.7)

can be rewritten as

$$\min_{\boldsymbol{x},\boldsymbol{v}} \max_{\boldsymbol{u}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 + \phi(\boldsymbol{v}) + \frac{1}{2\nu}\|\boldsymbol{x}-\boldsymbol{v}+\boldsymbol{u}\|^2 - \frac{1}{2\nu}\|\boldsymbol{u}\|^2 \right\}. \tag{2.8}$$

ADMM solves (2.8) by alternating the optimization of $\boldsymbol{x}$ and $\boldsymbol{v}$ with gradient ascent of $\boldsymbol{u}$, i.e.,

$$\boldsymbol{x}_k = \boldsymbol{h}(\boldsymbol{v}_{k-1} - \boldsymbol{u}_{k-1}; \nu) \tag{2.9a}$$

$$\boldsymbol{v}_k = \text{prox}_{\phi}(\boldsymbol{x}_k + \boldsymbol{u}_{k-1}; \nu) \tag{2.9b}$$

$$\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + (\boldsymbol{x}_k - \boldsymbol{v}_k), \tag{2.9c}$$

where $\boldsymbol{h}(\boldsymbol{z};\nu)$ and $\text{prox}_{\phi}(\boldsymbol{z};\nu)$, known as "proximal maps" (see the tutorial [37]) are defined as

$$\text{prox}_{\phi}(\boldsymbol{z};\nu) \triangleq \arg\min_{\boldsymbol{x}} \left\{ \phi(\boldsymbol{x}) + \frac{1}{2\nu}\|\boldsymbol{x}-\boldsymbol{z}\|^2 \right\} \tag{2.10}$$

$$\boldsymbol{h}(\boldsymbol{z};\nu) \triangleq \arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|^2 + \frac{1}{2\nu}\|\boldsymbol{x}-\boldsymbol{z}\|^2 \right\} \tag{2.11}$$

$$= \text{prox}_{\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|^2/(2\sigma^2)}(\boldsymbol{z};\nu) \tag{2.12}$$

$$= \left(\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} + \frac{\sigma^2}{\nu}\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^{\mathsf{H}}\boldsymbol{y} + \frac{\sigma^2}{\nu}\boldsymbol{z}\right). \tag{2.13}$$

Under some weak technical constraints, it can be proven [36] that when ϕ is convex, the ADMM iteration (2.9) converges to $\widehat{\boldsymbol{x}}$, the global minimum of (2.3) and (2.6).

From the discussion in Section 2.3, we immediately recognize $\text{prox}_{\phi}(\boldsymbol{z};\nu)$ in (2.10) as the MAP denoiser of $\boldsymbol{z}$ under AWGN variance ν and signal prior $p(\boldsymbol{x}) \propto \exp(-\phi(\boldsymbol{x}))$. The key idea behind the original PnP work [14] was, in the ADMM recursion (2.9), to "plug in" a powerful image denoising algorithm like "block-matching and 3D filtering" (BM3D) [26] in place of the proximal denoiser $\text{prox}_{\phi}(\boldsymbol{x};\nu)$ from (2.10). If the

plug-in denoiser is denoted by "$\boldsymbol{f}$," then the PnP ADMM algorithm becomes

$$\boldsymbol{x}_k = \boldsymbol{h}(\boldsymbol{v}_{k-1} - \boldsymbol{u}_{k-1}; \nu) \tag{2.14a}$$

$$\boldsymbol{v}_k = \boldsymbol{f}(\boldsymbol{x}_k + \boldsymbol{u}_{k-1}) \tag{2.14b}$$

$$\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + (\boldsymbol{x}_k - \boldsymbol{v}_k). \tag{2.14c}$$

A wide variety of empirical results (see, e.g., [14,38,39]) have demonstrated that, when $\boldsymbol{f}$ is a powerful denoising algorithm like BM3D, the PnP algorithm (2.14) produces far better recoveries than the regularization-based approach (2.9). For parallel MRI, the advantages of PnP ADMM were demonstrated in [40]. Although the value of ν does not change the fixed point of the standard ADMM algorithm (2.9), it affects the fixed point of the PnP ADMM algorithm (2.14) through the ratio σ^2/ν in (2.13).

The success of PnP methods raises important theoretical questions. Since $\boldsymbol{f}$ is not in general the proximal map of any regularizer ϕ, the iterations (2.14) may not minimize a cost function of the form in (2.3), and (2.14) may not be an implementation of ADMM. It is then unclear if the iterations (2.14) will converge. And if they do converge, it is unclear what they converge to. The consensus equilibrium framework, which we discuss in Section 2.5, aims to provide answers to these questions.

The use of a generic denoiser in place of a proximal denoiser can be translated to non-ADMM algorithms, such as FISTA [41], primal-dual splitting (PDS) [42], and others, as in [43–45]. Instead of optimizing $\boldsymbol{x}$ as in (2.14), PnP FISTA [43] uses the iterative update

$$\boldsymbol{z}_k = \boldsymbol{s}_{k-1} - \frac{\nu}{\sigma^2}\boldsymbol{A}^{\mathsf{H}}(\boldsymbol{A}\boldsymbol{s}_{k-1} - \boldsymbol{y}) \tag{2.15a}$$

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{z}_k) \tag{2.15b}$$

$$\boldsymbol{s}_k = \boldsymbol{x}_k + \frac{q_{k-1} - 1}{q_k}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}), \tag{2.15c}$$

where (2.15a) is a gradient descent (GD) step on the negative log-likelihood $\frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ at $\boldsymbol{x} = \boldsymbol{s}_{k-1}$ with step-size $\nu \in (0, \sigma^2\|\boldsymbol{A}\|_2^{-2})$, (2.15b) is the plug-in replacement of the usual proximal denoising step in FISTA, and (2.15c) is an acceleration step, where it is typical to use $q_k = (1 + \sqrt{1 + 4q_{k-1}^2})/2$ and $q_0 = 1$.

PnP PDS [44] uses the iterative update

$$\boldsymbol{x}_k = \boldsymbol{f}\Big(\boldsymbol{x}_{k-1} - \frac{\nu}{\sigma^2}\boldsymbol{A}^{\mathsf{H}}\boldsymbol{v}_{k-1}\Big) \tag{2.16a}$$

$$\boldsymbol{v}_k = \gamma\boldsymbol{v}_{k-1} + (1-\gamma)\Big(\boldsymbol{A}(2\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) - \boldsymbol{y}\Big) \tag{2.16b}$$

where $\nu > 0$ is a stepsize, $\gamma \in (0, 1)$ is a relaxation parameter chosen such that $\gamma \leq \nu/(\nu + \sigma^2\|\boldsymbol{A}\|_2^{-2})$, and $\boldsymbol{f}(\boldsymbol{z})$ in (2.16a) is the plug-in replacement of the usual proximal denoiser $\text{prox}_{\phi}(\boldsymbol{z}; \nu)$.

Comparing PnP ADMM (2.14) to PnP FISTA (2.15) and PnP PDS (2.16), one can see that they differ in how the data fidelity term $\frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ is handled: PnP ADMM uses the proximal update (2.13), while PnP FISTA and PnP PDS use the GD step (2.15a). In most cases, solving the proximal update (2.13) is much more computationally costly than taking a GD step (2.15a). Thus, with ADMM, it is common to approximate the proximal update (2.13) using, e.g., several iterations of the conjugate gradient (CG) algorithm or GD, which should reduce the per-iteration complexity of (2.14) but may increase the number of iterations. But even with these approximations of (2.13), PnP ADMM is usually close to "convergence" after 10-50 iterations (e.g., see Figure 2.4).

An important difference between the aforementioned flavors of PnP is that the stepsize ν is constrained in FISTA but not in ADMM or PDS. Thus, PnP FISTA restricts the range of reachable fixed points relative to PnP ADMM and PnP PDS.

2.4.2 The balanced FISTA approach

In Section 2.3, when discussing the optimization problem (2.3), the regularizer $\phi(\boldsymbol{x}) = \lambda\|\boldsymbol{\Psi x}\|_1$ was mentioned as a popular option, where $\boldsymbol{\Psi}$ is often a wavelet transform. The resulting optimization problem,

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{A x}\|_2^2 + \lambda\|\boldsymbol{\Psi x}\|_1 \right\}, \tag{2.17}$$

is said to be stated in "analysis" form [16]. The proximal denoiser associated with (2.17) has the form

$$\text{prox}_{\phi}(\boldsymbol{z};\nu) = \arg\min_{\boldsymbol{x}} \left\{ \lambda\|\boldsymbol{\Psi x}\|_1 + \frac{1}{2\nu}\|\boldsymbol{x} - \boldsymbol{z}\|^2 \right\}. \tag{2.18}$$

When $\boldsymbol{\Psi}$ is orthogonal, it is well known that $\text{prox}_{\phi}(\boldsymbol{z};\nu) = \boldsymbol{f}_{\mathsf{tdt}}(\boldsymbol{z};\lambda\nu)$, where

$$\boldsymbol{f}_{\mathsf{tdt}}(\boldsymbol{z};\tau) \triangleq \boldsymbol{\Psi}^{\mathsf{H}}\text{soft-thresh}(\boldsymbol{\Psi z};\tau) \tag{2.19}$$

is the "transform-domain thresholding" denoiser with

$$[\text{soft-thresh}(\boldsymbol{u},\tau)]_n \triangleq \max\left\{0, \frac{|u_n| - \tau}{|u_n|}\right\} u_n. \tag{2.20}$$

The denoiser (2.19) is very efficient to implement, since it amounts to little more than computing forward and reverse transforms.

In practice, (2.17) yields much better results with *non*-orthogonal $\boldsymbol{\Psi}$, such as when $\boldsymbol{\Psi}^{\mathsf{H}}$ is a tight frame (see, e.g., the references in [46]). In the latter case, $\boldsymbol{\Psi}^{\mathsf{H}}\boldsymbol{\Psi} = \boldsymbol{I}$ with tall $\boldsymbol{\Psi}$. But, for general tight frames $\boldsymbol{\Psi}^{\mathsf{H}}$, the proximal denoiser (2.18) has no closed-form solution. What if we simply plugged the transform-domain thresholding denoiser (2.19) into an algorithm like ADMM or FISTA? How can we interpret the resulting approach? Interestingly, as we describe below, if (2.19) is used in PnP FISTA, then it does solve a convex optimization problem, although one with a different form than

(2.3). This approach was independently proposed in [33] and [46], where in the latter it was referred to as "balanced FISTA" (bFISTA) and applied to parallel cardiac MRI. Notably, bFISTA was proposed before the advent of PnP FISTA. More details are provided below.

The optimization problem (2.17) can be stated in constrained "synthesis" form as

$$\widehat{\boldsymbol{x}} = \boldsymbol{\Psi}^{\mathsf{H}}\widehat{\boldsymbol{\alpha}} \quad \text{for} \quad \widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}\in\text{range}(\boldsymbol{\Psi})}{\arg\min} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\Psi}^{\mathsf{H}}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1 \right\}, \tag{2.21}$$

where $\boldsymbol{\alpha}$ are transform coefficients. Then, as $\beta \to \infty$ below, (2.21) can be expressed in the unconstrained form

$$\widehat{\boldsymbol{x}} = \boldsymbol{\Psi}^{\mathsf{H}}\widehat{\boldsymbol{\alpha}} \quad \text{for} \quad \widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\arg\min} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\Psi}^{\mathsf{H}}\boldsymbol{\alpha}\|_2^2 + \frac{\beta}{2}\|\boldsymbol{P}_{\boldsymbol{\Psi}}^{\perp}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1 \right\} \tag{2.22}$$

with projection matrix $\boldsymbol{P}_{\boldsymbol{\Psi}}^{\perp} \triangleq \boldsymbol{I} - \boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathsf{H}}$. In practice, it is not possible to take $\beta \to \infty$ and, for finite values of β, the problems (2.21) and (2.22) are not equivalent. However, problem (2.22) under finite β is interesting to consider in its own right, and it is sometimes referred to as the "balanced" approach [47]. If we solve (2.22) using FISTA with step-size $\nu > 0$ (recall (2.15a)) and choose the particular value $\beta = 1/\nu$ then, remarkably, the resulting algorithm takes the form of PnP FISTA (2.15) with $\boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{f}_{\mathsf{tdt}}(\boldsymbol{z};\lambda)$. This particular choice of β is motivated by computational efficiency (since it leads to the use of $\boldsymbol{f}_{\mathsf{tdt}}$) rather than recovery performance. Still, as we demonstrate in Section 2.6, it performs relatively well.

2.5 Understanding PnP through Consensus Equilibrium

The success of the PnP methods in Section 2.4 raises important theoretical questions. For example, in the case of PnP ADMM, if the plug-in denoiser $\boldsymbol{f}$ is not the proximal map of any regularizer ϕ, then it is not clear what cost function is being

minimized (if any) or whether the algorithm will even converge. In this section, we show that many of these questions can be answered through the consensus equilibrium (CE) framework [1,45,48,49].

Rather than viewing (2.14) as minimizing some cost function, we can view it as seeking a solution $(\widehat{\boldsymbol{x}}_{\mathsf{pnp}}, \widehat{\boldsymbol{u}}_{\mathsf{pnp}})$ to

$$\widehat{\boldsymbol{x}}_{\mathsf{pnp}} = \boldsymbol{h}(\widehat{\boldsymbol{x}}_{\mathsf{pnp}} - \widehat{\boldsymbol{u}}_{\mathsf{pnp}}; \nu) \tag{2.23a}$$

$$\widehat{\boldsymbol{x}}_{\mathsf{pnp}} = \boldsymbol{f}(\widehat{\boldsymbol{x}}_{\mathsf{pnp}} + \widehat{\boldsymbol{u}}_{\mathsf{pnp}}), \tag{2.23b}$$

which, by inspection, must hold when (2.14) is at a fixed point. Not surprisingly, by setting $\boldsymbol{x}_k = \boldsymbol{x}_{k-1}$ in the PnP FISTA algorithm (2.15), it is straightforward to show that it too seeks a solution to (2.23). It is easy to show that the PnP PDS algorithm [44] seeks the same solution. With (2.23), the goal of the prox-based PnP algorithms becomes well defined! The pair (2.23) reaches a *consensus* in that the denoiser $\boldsymbol{f}$ and the data fitting operator $\boldsymbol{h}$ agree on the output $\widehat{\boldsymbol{x}}_{\mathsf{pnp}}$. The *equilibrium* comes from the opposing signs on the correction term $\widehat{\boldsymbol{u}}_{\mathsf{pnp}}$: the data-fitting subtracts it while the denoiser adds it.

Applying the $\boldsymbol{h}$ expression from (2.13) to (2.23a), we find that

$$\widehat{\boldsymbol{u}}_{\mathsf{pnp}} = \frac{\nu}{\sigma^2}\boldsymbol{A}^{\mathsf{H}}(\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{\mathsf{pnp}}), \tag{2.24}$$

where $\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{\mathsf{pnp}}$ is the k-space measurement error and $\boldsymbol{A}^{\mathsf{H}}(\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{\mathsf{pnp}})$ is its projection back into the image domain. We now see that $\widehat{\boldsymbol{u}}_{\mathsf{pnp}}$ provides a correction on $\widehat{\boldsymbol{x}}_{\mathsf{pnp}}$ for any components that are inconsistent with the measurements, but it provides no correction for errors in $\widehat{\boldsymbol{x}}_{\mathsf{pnp}}$ that lie outside the row-space of $\boldsymbol{A}$. Plugging (2.24) back into (2.23b), we obtain the image-domain fixed point equation

$$\widehat{\boldsymbol{x}}_{\mathsf{pnp}} = \boldsymbol{f}\left(\left(\boldsymbol{I} - \frac{\nu}{\sigma^2}\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}\right)\widehat{\boldsymbol{x}}_{\mathsf{pnp}} + \frac{\nu}{\sigma^2}\boldsymbol{A}^{\mathsf{H}}\boldsymbol{y}\right). \tag{2.25}$$

If $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}$, as in (2.2), and we define the PnP error as $\widehat{\boldsymbol{e}}_{\mathsf{pnp}} \triangleq \widehat{\boldsymbol{x}}_{\mathsf{pnp}} - \boldsymbol{x}$, then (2.25) implies

$$\widehat{\boldsymbol{x}}_{\mathsf{pnp}} = \boldsymbol{f}\left(\widehat{\boldsymbol{x}}_{\mathsf{pnp}} + \frac{\nu}{\sigma^2}\boldsymbol{A}^{\mathsf{H}}\left(\boldsymbol{w} - \boldsymbol{A}\widehat{\boldsymbol{e}}_{\mathsf{pnp}}\right)\right), \tag{2.26}$$

which says that the error $\widehat{\boldsymbol{e}}_{\mathsf{pnp}}$ combines with the k-space measurement noise $\boldsymbol{w}$ in such a way that the corresponding image-space correction $\widehat{\boldsymbol{u}}_{\mathsf{pnp}} = \frac{\nu}{\sigma^2}\boldsymbol{A}^{\mathsf{H}}(\boldsymbol{w} - \boldsymbol{A}\widehat{\boldsymbol{e}}_{\mathsf{pnp}})$ is canceled by the denoiser $\boldsymbol{f}(\cdot)$.

By viewing the goal of prox-based PnP as solving the equilibrium problem (2.23), it becomes clear that other solvers beyond ADMM, FISTA, and PDS can be used. For example, it was shown in [48] that the PnP CE condition (2.23) can be achieved by finding a fixed point of the system[2]

$$\underline{\boldsymbol{z}} = (2\mathcal{G} - \boldsymbol{I})(2\mathcal{F} - \boldsymbol{I})\underline{\boldsymbol{z}} \tag{2.27}$$

$$\underline{\boldsymbol{z}} = \begin{bmatrix}\boldsymbol{z}_1\\ \boldsymbol{z}_2\end{bmatrix}, \quad \mathcal{F}(\underline{\boldsymbol{z}}) = \begin{bmatrix}\boldsymbol{h}(\boldsymbol{z}_1;\nu)\\ \boldsymbol{f}(\boldsymbol{z}_2)\end{bmatrix}, \quad \text{and} \quad \mathcal{G}(\underline{\boldsymbol{z}}) = \begin{bmatrix}\frac{1}{2}(\boldsymbol{z}_1 + \boldsymbol{z}_2)\\ \frac{1}{2}(\boldsymbol{z}_1 + \boldsymbol{z}_2)\end{bmatrix}. \tag{2.28}$$

There exist many algorithms to solve (2.27). For example, one could use the Mann iteration [37]

$$\underline{\boldsymbol{z}}^{(k+1)} = (1-\gamma)\underline{\boldsymbol{z}}^{k} + \gamma(2\mathcal{G} - \boldsymbol{I})(2\mathcal{F} - \boldsymbol{I})\underline{\boldsymbol{z}}^{(k)}, \quad \text{with } \gamma \in (0,1), \tag{2.29}$$

when $\mathcal{F}$ is nonexpansive. The paper [48] also shows that this fixed point is equivalent to the solution of $\mathcal{F}(\underline{\boldsymbol{z}}) = \mathcal{G}(\underline{\boldsymbol{z}})$, in which case Newton's method or other root-finding methods could be applied.

The CE viewpoint also provides a path to proving the convergence of the PnP ADMM algorithm. Sreehari et al. [38] used a classical result from convex analysis to show that a sufficient condition for convergence is that i) $\boldsymbol{f}$ is non-expansive,

[2]The paper [48] actually considers the consensus equilibrium among $N > 1$ agents, whereas here we consider the simple case of $N = 2$ agents.

i.e., $\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$ for any $\boldsymbol{x}$ and $\boldsymbol{y}$, and ii) $\boldsymbol{f}(\boldsymbol{x})$ is a sub-gradient of some convex function, i.e., there exists φ such that $\boldsymbol{f}(\boldsymbol{x}) \in \partial\varphi(\boldsymbol{x})$. If these two conditions are met, then PnP ADMM (2.14) will converge to a global solution. Similar observations were made in other recent studies, e.g., [49,50]. That said, Chan et al. [39] showed that many practical denoisers do not satisfy these conditions, and so they designed a variant of PnP ADMM in which ν is decreased at every iteration. Under appropriate conditions on $\boldsymbol{f}$ and the rate of decrease, this latter method also guarantees convergence, although not exactly to a fixed point of (2.23) since ν is no longer fixed.

Similar techniques can be used to prove the convergence of other prox-based PnP algorithms. For example, under certain technical conditions, including non-expansiveness of $\boldsymbol{f}$, it was established [45] that PnP FISTA converges to the same fixed point as PnP ADMM.

2.6 Demonstration of PnP in MRI

2.6.1 Parallel cardiac MRI

We now demonstrate the application of PnP methods to parallel cardiac MRI. Because the signal $\boldsymbol{x}$ is a cine (i.e., a video) rather than a still image, there are relatively few options available for sophisticated denoisers. Although algorithmic denoisers like BM4D [51] have been proposed, they tend to run very slowly, especially relative to the linear operators $\boldsymbol{A}$ and $\boldsymbol{A}^{\mathsf{H}}$. For this reason, we first trained an application specific CNN denoiser for use in the PnP framework. The architecture of the CNN denoiser, implemented and trained in PyTorch, is shown in Figure 2.1.

For training, we acquired 50 fully sampled, high-SNR cine datasets from eight healthy volunteers. Thirty three of those were collected on a 3 T scanner[3] and the remaining 17 were collected on a 1.5 T scanner. Out of the 50 datasets, 28, 7, 7, and 8 were collected in the short-axis, two-chamber, three-chamber, and four-chamber view, respectively. The spatial and temporal resolutions of the images ranged from 1.8 mm to 2.5 mm and from 34 ms to 52 ms, respectively. The image size ranged from 160×130 to 256×208 pixels and the number of frames ranged from 15 to 27. For each of the 50 datasets, the reference image series was estimated as the least-squares solution to (2.1), with the sensitivity maps $\boldsymbol{S}_i$ estimated from the time-averaged data using ESPIRiT [52]. We added zero-mean, complex-valued i.i.d. Gaussian noise to these "noise-free" reference images to simulate noisy images with SNR of 24 dB. Using a fixed stride of $30 \times 30 \times 10$, we decomposed the images into patches of size $55 \times 55 \times 15$. The noise-free and corresponding noisy patches were assigned as output and input to the CNN denoiser, with the real and imaginary parts processed as two separate channels. All 3D convolutions were performed using $3 \times 3 \times 3$ kernels. There were 64 filters of size $3 \times 3 \times 3 \times 2$ in the first layer, 64 filters of size $3 \times 3 \times 3 \times 64$ in the second through fourth layers, and 2 filters of size $3 \times 3 \times 3 \times 64$ in the last layer. We set the minibatch size to four and used the Adam optimizer [53] with a learning rate of 1×10^{-4} over 400 epochs. The training process was completed in 12 hours on a workstation equipped with a single NVIDIA GPU (GeForce RTX 2080 Ti).

For testing, we acquired four fully sampled cine datasets from two different healthy volunteers, with two image series in the short-axis view, one image series in the two-chamber view, and one image series in the four-chamber view. The spatial and

[3]The 3 T scanner was a Magnetom Prisma Fit from Siemens Healthineers in Erlangen, Germany and the 1.5 T scanner was a Magnetom Avanto from Siemens Healthineers in Erlangen, Germany.

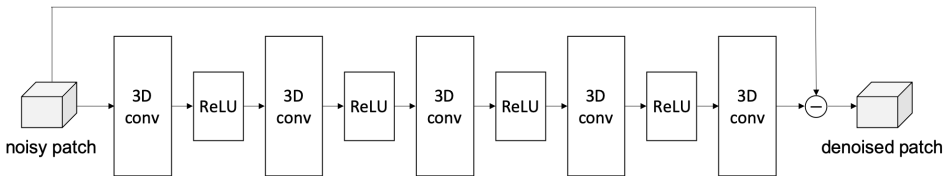


Figure 2.1: The architecture of the CNN-based cardiac cine denoiser operating on spatiotemporal volumetric patches.

temporal resolutions of the images ranged from 1.9 mm to 2 mm and from 37 ms to 45 ms, respectively. For the four datasets, the space-time signal vector, $\boldsymbol{x}$, in (2.2) had dimensions of $192 \times 144 \times 25$, $192 \times 144 \times 25$, $192 \times 166 \times 16$, and $192 \times 166 \times 16$, respectively, with the last dimension representing the number of frames. The datasets were retrospectively downsampled at acceleration rates, R, of 6, 8, and 10 using pseudo-random sampling [54]. A representative sampling pattern used to undersample one of the datasets is shown in Figure 2.2. The data were compressed to $C = 12$ virtual coils for faster computation [55]. The measurements were modeled as described in (2.1), with the sensitivity maps, $\boldsymbol{S}_i$, estimated from the time-averaged data using ESPIRiT.

For compressive MRI recovery, we used PnP ADMM from (2.14) with $\boldsymbol{f}(\cdot)$ as the CNN-based denoiser described above; we will refer to the combination as PnP-CNN. We employed a total of 100 ADMM iterations, and in each ADMM iteration, we performed four steps of CG to approximate (2.13), for which we used $\sigma^2 = 1 = \nu$. (See Figure 2.5 for the effect of σ^2/ν on the final NMSE and the convergence rate.) We compared this PnP method to three CS-based methods: CS with an undecimated wavelet transform (CS-UWT), CS with total variation (TV),[4] and a low-rank plus

[4]Note that sometimes UWT and TV are combined [15].

sparse (L+S) method (see, e.g., [56]). We also compared to PnP-UWT and the transform-learning [57] method LASSI [58].

For PnP-UWT, we used PnP FISTA from (2.15) with $\boldsymbol{f}(\cdot)$ implemented as $\boldsymbol{f}_{\mathsf{tdt}}$ given in (2.19), i.e., bFISTA. A three-dimensional single-level Haar UWT was used as $\boldsymbol{\Psi}$ in (2.19). For CS-TV, we used a 3D finite-difference operator for $\boldsymbol{\Psi}$ in the regularizer $\phi(\boldsymbol{x}) = \|\boldsymbol{\Psi}\boldsymbol{x}\|_1$, and for CS-UWT, we used the aforementioned UWT instead. For both CS-TV and CS-UWT, we used monotone FISTA [59] to solve the resulting convex optimization problem (2.3). For L+S, the method by Otazo et al. [56] was used. The regularization weights for CS-UWT, PnP-UWT, CS-TV, and L+S were manually tuned to maximize the reconstruction SNR (rSNR)[5] for Dataset #3 at $R = 10$. For LASSI we used the authors' implementation at `https://gitlab.com/ravsa19/lassi`, and we did our best to manually tune all available parameters.

The rSNR values are summarized in Table 2.1. For all four datasets and three acceleration rates, PnP-CNN exhibited the highest rSNR with respect to the fully sampled reference. Also, compared to the CS methods and PnP-UWT, which uses a more traditional denoiser based on soft-thresholding of UWT coefficients, PnP-CNN was better at preserving anatomical details of the heart; see Figure 2.3. The performance of PnP-UWT was similar to that of CS-UWT. Figure 2.4 plots NMSE as a function of the number of iterations for the CS and PnP methods. Since the CS methods were implemented using CPU computation and the PnP methods were implemented using GPU computation, a direct runtime comparison was not possible. We did, however, compare the per-iteration runtime of PnP ADMM for two different

[5] rSNR is defined as $\|\boldsymbol{x}\|^2/\|\widehat{\boldsymbol{x}} - \boldsymbol{x}\|^2$, where $\boldsymbol{x}$ is the true image and $\widehat{\boldsymbol{x}}$ is the estimate.

denoisers: the CNN and UWT-based $\boldsymbol{f}_{\mathsf{tdt}}$ described earlier in this section. When the CNN denoiser was replaced with the UWT-based $\boldsymbol{f}_{\mathsf{tdt}}$, the per-iteration runtime changed from 2.05 s to 2.1 s, implying that the two approaches have very similar computational costs.

For PnP-CNN, Figure 2.5 shows the dependence of the final NMSE ($= \text{rSNR}^{-1}$) and of the convergence rate on σ^2/ν for one of the testing datasets included in this study. Overall, final NMSE varies less than 0.5 dB for $\sigma^2/\nu \in [0.5, 2]$ for all four datasets and all three acceleration rates. Figure 2.6 compares CG and GD when solving (2.13). To this end, NMSE vs. runtime is plotted for different numbers of CG or GD inner-iterations for Dataset #3 at $R = 10$. For GD, the step-size was manually optimized. Figure 2.6 suggests that it is best to use a 1 to 4 inner iterations of either GD or CG; using more inner iterations slows the convergence rate without improving the final performance.

The results in this section, although preliminary, highlight the potential of PnP methods for MRI recovery of cardiac cines. By optimizing the denoiser architecture, the performance of PnP-CNN may be further improved.

2.6.2 Single-coil fastMRI knee data

In this section, we investigate recovery of 2D knee images from the single-coil fastMRI dataset [60]. This dataset contains fully-sampled k-space data that are partitioned into 34 742 training slices and 7 135 testing slices. The Cartesian sampling patterns from [60] were used to achieve acceleration rate $R = 4$.

We evaluated PnP using the ADMM algorithm with a learned DnCNN [10] denoiser. To accommodate complex-valued images, DnCNN was configured with two

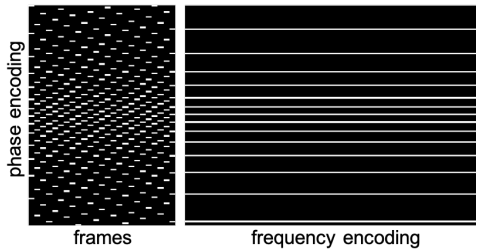


Figure 2.2: Two different views of the 3D sampling pattern used to retrospectively undersample one of the four test datasets at $R = 10$. The undersampling was performed only in the phase encoding direction and the pattern was varied across frames. In this example, the number of frequency encoding steps, phase encoding steps, and frames are 192, 144, and 25, respectively.

Acceleration	CS-UWT	CS-TV	L+S	LASSI	PnP-UWT	PnP-CNN
Dataset #1 (short-axis)						
$R = 6$	30.10	29.03	30.97	27.09	30.18	**31.82**
$R = 8$	28.50	27.35	29.65	25.91	28.60	**31.25**
$R = 10$	26.94	25.78	28.29	24.98	27.06	**30.46**
Dataset #2 (short-axis)						
$R = 6$	29.23	28.27	29.73	25.87	29.29	**30.81**
$R = 8$	27.67	26.65	28.23	24.54	27.75	**30.17**
$R = 10$	26.12	25.11	26.89	23.61	26.22	**29.21**
Dataset #3 (two-chamber)						
$R = 6$	27.33	26.38	27.83	24.97	27.38	**29.36**
$R = 8$	25.63	24.63	26.30	23.52	25.69	**28.50**
$R = 10$	24.22	23.24	24.93	22.51	24.28	**27.49**
Dataset #4 (four-chamber)						
$R = 6$	30.41	29.63	30.62	27.62	30.60	**32.19**
$R = 8$	28.68	27.76	29.00	26.33	28.94	**31.42**
$R = 10$	27.09	26.18	27.60	25.24	27.37	**30.01**

Table 2.1: rSNR (dB) of MRI cardiac cine recovery from four test datasets.

input and two output channels. The denoiser was then trained using only the central slices of the 3 T scans without fat-suppression from the training set, comprising a total of 267 slices (i.e., $< 1\%$ of the total training data). The training-noise variance and the PnP ADMM tuning parameter σ^2/ν were manually adjusted in an attempt to maximize rSNR.

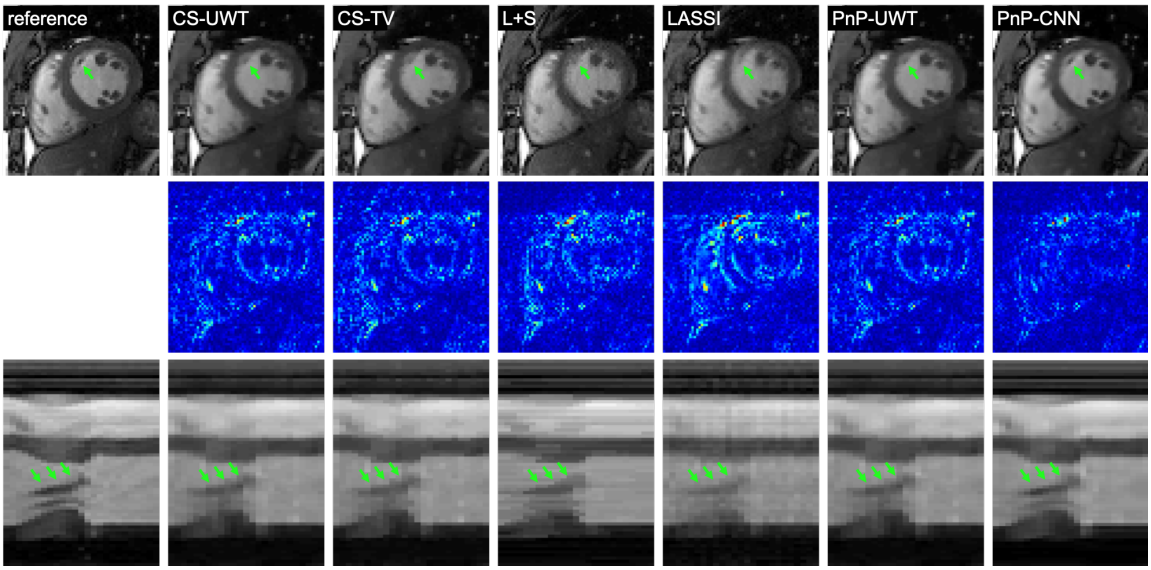


Figure 2.3: Results from cardiac cine Dataset #1 at $R = 10$. Top row: a representative frame from the fully sampled reference and various recovery methods. The green arrow points to an image feature that is preserved only by PnP-CNN and not by other methods. Middle row: error map $\times 6$. Bottom row: temporal frame showing the line drawn horizontally through the middle of the image in the top row, with the time dimension along the horizontal axis. The arrows point to the movement of the papillary muscles, which are more well-defined in PnP-CNN.

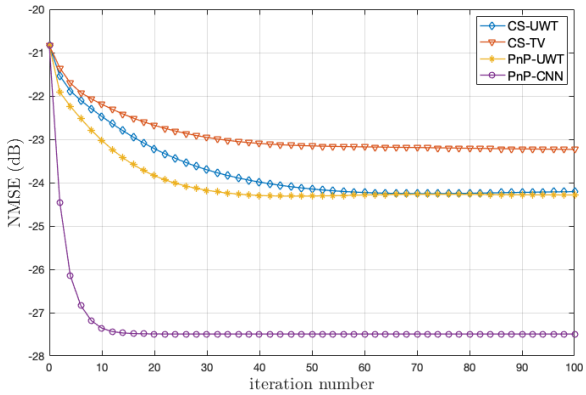


Figure 2.4: NMSE versus iteration for two PnP and two CS algorithms on the cardiac cine recovery Dataset #3 at $R = 10$.

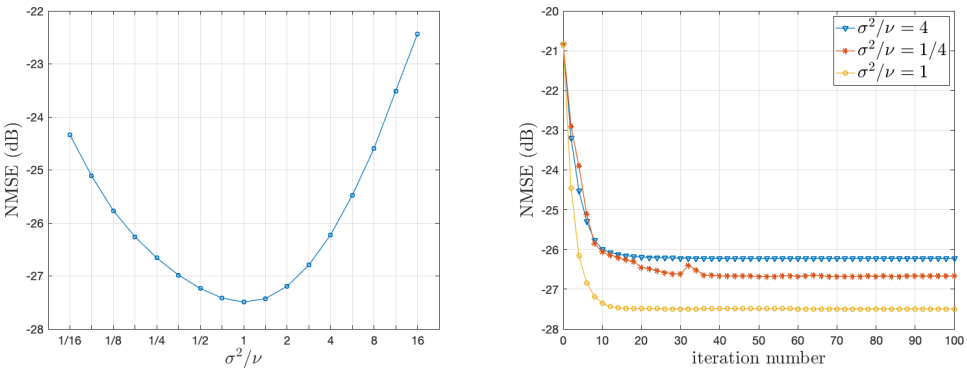


Figure 2.5: For PnP-CNN recovery of cardiac cine Dataset #3 at $R = 10$, the change in the final NMSE (after 100 iterations) as a function of σ^2/ν (left) and the NMSE versus iteration for several σ^2/ν (right).

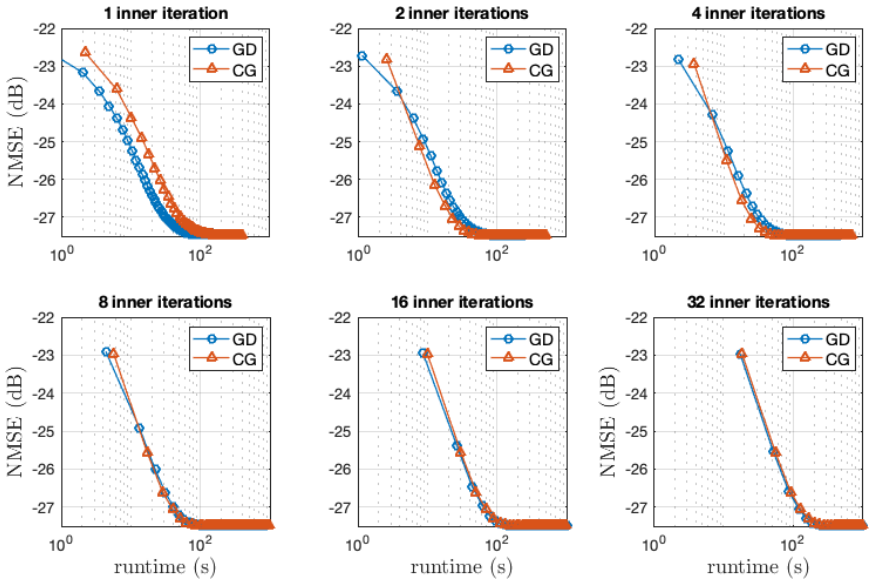


Figure 2.6: NMSE versus time PnP ADMM with different numbers of CG and GD inner iterations on cardiac cine Dataset #3 at $R = 10$.

PnP was then compared to the TV and U-Net baseline methods described and configured in [60]. For example, 128 channels were used for the U-Net's first layer, as recommended in [60]. We then trained three versions of the U-Net. The first version

was trained on the full fastMRI training set[6] with random sampling masks. The second U-Net was trained on the full fastMRI training set, but with a fixed sampling mask. The third U-Net was trained using only the central slices of the 3 T scans without fat-suppression (i.e., the same data used to train the DnCNN denoiser) and with a fixed sampling mask.

To evaluate performance, we used the central slices of the non-fat-suppressed 3 T scans from the validation set, comprising a total of 49 slices. The evaluation considered both random sampling masks and the same fixed mask used for training. The resulting average rSNR and SSIM scores are summarized in Table 2.2. The table shows that PnP-CNN performed similarly to the U-Nets and significantly better than TV. In particular, PnP-CNN achieved the highest rSNR score with both random and fixed testing masks, and the U-Net gave slightly higher SSIM scores in both tests. Among the U-Nets, the version trained with a fixed sampling mask and full data gave the best rSNR and SSIM performance when testing with the same mask, but its performance dropped considerable when testing with random masks. Meanwhile, the U-Net trained with the smaller data performed significantly worse than the other U-Nets, with either fixed or random testing masks. And although this latter U-Net used exactly the same training data as the PnP-CNN method, it was not competitive with PnP-CNN. Although preliminary, these results suggest that i) PnP methods are much less sensitive to deviations in the forward model between training and testing, and that ii) PnP methods are effective with relatively small training datasets.

[6]The full fastMRI training set includes 1.5 T and 3 T scans, with and without fat suppression, and an average of 36 slices per volume.

	Random testing masks		Fixed testing mask	
	rSNR (dB)	SSIM	rSNR (dB)	SSIM
CS-TV	17.56	0.647	18.16	0.654
U-Net: Random training masks, full training data	20.76	**0.772**	20.72	0.768
U-Net: Fixed training mask, full training data	19.63	0.756	20.82	**0.770**
U-Net: Fixed training mask, smaller training data	18.90	0.732	19.67	0.742
PnP-CNN	**21.16**	0.758	**21.14**	0.754

Table 2.2: rSNR and SSIM for fastMRI single-coil test data with $R = 4$.

2.7 Conclusion

PnP methods present an attractive avenue for compressive MRI recovery. In contrast to traditional CS methods, PnP methods can exploit richer image structure by using state-of-the-art denoisers. To demonstrate the potential of such methods for MRI reconstruction, we used PnP to recover cardiac cines and knee images from highly undersampled datasets. With application-specific CNN-based denoisers, PnP was able to significantly outperform traditional CS methods and to perform on par with modern deep-learning methods, but with considerably less training data. The time is ripe to investigate the potential of PnP methods for a variety of MRI applications.

Chapter 3: Regularization by Denoising: Clarifications and New Interpretations

3.1 Introduction

In this chapter[7], we consider the problem of recovering a (vectorized) image $\boldsymbol{x}^0 \in \mathbb{R}^N$ from noisy linear measurements $\boldsymbol{y} \in \mathbb{R}^M$ of the form

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^0 + \boldsymbol{e}, \tag{3.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ is a known linear transformation and $\boldsymbol{e}$ is noise. This problem is of great importance in many applications and has been studied for several decades.

One of the most popular approaches to image recovery is the "variational" approach, where one poses and solves an optimization problem of the form

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \big\{ \ell(\boldsymbol{x}; \boldsymbol{y}) + \lambda \rho(\boldsymbol{x}) \big\}. \tag{3.2}$$

In (3.2), $\ell(\boldsymbol{x}; \boldsymbol{y})$ is a loss function that penalizes mismatch to the measurements, $\rho(\boldsymbol{x})$ is a regularization term that penalizes mismatch to the image class of interest, and $\lambda > 0$ is a design parameter that trades between loss and regularization. A prime advantage of the variational approach is that, in many cases, efficient optimization methods can be readily applied to (3.2).

[7]Work presented in this chapter is largely excerpted from the manuscript [1] co-authored with Philip Schniter.

A key question is: How should one choose the loss $\ell(\cdot;\boldsymbol{y})$ and regularization $\rho(\cdot)$ in (3.2)? As discussed in the sequel, the MAP-Bayesian interpretation suggests that they should be chosen in proportion to the negative log-likelihood and negative log-prior, respectively. The trouble is that accurate prior models for images are lacking.

Recently, a breakthrough was made by Romano, Elad, and Milanfar in [5]. Leveraging the long history (e.g., [61,62]) and recent advances (e.g., [10,63]) in image denoising algorithms, they proposed the *regularization by denoising* (RED) framework, where an explicit regularizer $\rho(\boldsymbol{x})$ is constructed from an image denoiser $\boldsymbol{f}:\mathbb{R}^N\to\mathbb{R}^N$ using the simple and elegant rule

$$\rho_{\mathsf{red}}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\big(\boldsymbol{x}-\boldsymbol{f}(\boldsymbol{x})\big). \tag{3.3}$$

Based on this framework, they proposed several recovery algorithms (based on steepest descent, ADMM, and fixed-point methods, respectively) that yield state-of-the-art performance in deblurring and super-resolution tasks.

In this chapter, we provide some clarifications and new interpretations of the excellent RED algorithms from [5]. Our work was motivated by an interesting empirical observation: With many practical denoisers $\boldsymbol{f}(\cdot)$, the RED algorithms do not minimize the RED variational objective "$\ell(\boldsymbol{x};\boldsymbol{y})+\lambda\rho_{\mathsf{red}}(\boldsymbol{x})$." As we establish in the sequel, the RED regularization (3.3) is justified only for denoisers with symmetric Jacobians, which unfortunately does not cover many state-of-the-art methods such as non-local means (NLM) [9], BM3D [26], TNRD [63], and DnCNN [10]. In fact, we are able to establish a stronger result: For non-symmetric denoisers, there exists no regularization $\rho(\cdot)$ that explains the RED algorithms from [5].

In light of these (negative) results, there remains the question of how to explain/understand the RED algorithms from [5] when used with non-symmetric denoisers. In response, we propose a framework called *score-matching by denoising* (SMD), which aims to match the "score" (i.e., the gradient of the log-prior) rather than to design any explicit regularizer. We then show tight connections between SMD, kernel density estimation [64], and constrained minimum mean-squared error (MMSE) denoising. In addition, we provide new interpretations of the RED-ADMM and RED-FP algorithms proposed in [5], and we propose novel RED algorithms with faster convergence. Inspired by [48], we show that the RED algorithms seek to satisfy a consensus equilibrium condition that allows a direct comparison to the plug-and-play ADMM algorithms discussed in chapter 2.

The remainder of the chapter is organized as follows. In Section 3.2 we provide more background on RED and related algorithms such as plug-and-play ADMM [14]. In Section 3.3, we discuss the impact of Jacobian symmetry on RED and test whether this property holds in practice. In Section 3.4, we propose the SMD framework. In Section 3.5, we present new interpretations of the RED algorithms from [5] and new algorithms based on accelerated proximal gradient methods. In Section 3.6, we perform an equilibrium analysis of the RED algorithms, and, in Section 3.7, we conclude.

3.2 Background

3.2.1 The MAP-Bayesian Interpretation

For use in the sequel, we briefly discuss the Bayesian maximum a posteriori (MAP) estimation framework [65]. The MAP estimate of $\boldsymbol{x}$ from $\boldsymbol{y}$ is defined as

$$\widehat{\boldsymbol{x}}_{\mathsf{map}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y}), \tag{3.4}$$

where $p(\boldsymbol{x}|\boldsymbol{y})$ denotes the probability density of $\boldsymbol{x}$ given $\boldsymbol{y}$. Notice that, from Bayes rule $p(\boldsymbol{x}|\boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})/p(\boldsymbol{y})$ and the monotonically increasing nature of $\ln(\cdot)$, we can write

$$\widehat{\boldsymbol{x}}_{\mathsf{map}} = \arg\min_{\boldsymbol{x}} \big\{ -\ln p(\boldsymbol{y}|\boldsymbol{x}) - \ln p(\boldsymbol{x}) \big\}. \tag{3.5}$$

MAP estimation (3.5) has a direct connection to variational optimization (3.2): the log-likelihood term $-\ln p(\boldsymbol{y}|\boldsymbol{x})$ corresponds to the loss $\ell(\boldsymbol{x};\boldsymbol{y})$ and the log-prior term $-\ln p(\boldsymbol{x})$ corresponds to the regularization $\lambda\rho(\boldsymbol{x})$. For example, with additive white Gaussian noise (AWGN) $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \sigma_e^2 \boldsymbol{I})$, the log-likelihood implies a quadratic loss:

$$\ell(\boldsymbol{x};\boldsymbol{y}) = \frac{1}{2\sigma_e^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{3.6}$$

Equivalently, the normalized loss $\ell(\boldsymbol{x};\boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$ could be used if σ_e^2 were absorbed into λ.

3.2.2 ADMM

A popular approach to solving (3.2) is through ADMM [36], which we now review. Using variable splitting, (3.2) becomes

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \big\{ \ell(\boldsymbol{x};\boldsymbol{y}) + \lambda\rho(\boldsymbol{v}) \big\} \;\; \text{s.t. } \boldsymbol{x} = \boldsymbol{v}. \tag{3.7}$$

Algorithm 1 ADMM [36]

Require: $\ell(\cdot;\boldsymbol{y}), \rho(\cdot), \beta, \lambda, \boldsymbol{v}_0, \boldsymbol{u}_0$, and K

1: **for** $k = 1, 2, \ldots, K$ **do**
2: $\quad \boldsymbol{x}_k = \arg\min_{\boldsymbol{x}}\{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{v}_{k-1} + \boldsymbol{u}_{k-1}\|^2\}$
3: $\quad \boldsymbol{v}_k = \arg\min_{\boldsymbol{v}}\{\lambda\rho(\boldsymbol{v}) + \frac{\beta}{2}\|\boldsymbol{v} - \boldsymbol{x}_k - \boldsymbol{u}_{k-1}\|^2\}$
4: $\quad \boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \boldsymbol{x}_k - \boldsymbol{v}_k$
5: **end for**
6: Return $\boldsymbol{x}_K$

Using the augmented Lagrangian, problem (3.7) can be reformulated as

$$\min_{\boldsymbol{x},\boldsymbol{v}} \max_{\boldsymbol{p}} \left\{ \ell(\boldsymbol{x};\boldsymbol{y}) + \lambda\rho(\boldsymbol{v}) + \boldsymbol{p}^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{v}) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{v}\|^2 \right\} \tag{3.8}$$

using Lagrange multipliers (or "dual" variables) $\boldsymbol{p}$ and a design parameter $\beta > 0$. Using $\boldsymbol{u} \triangleq \boldsymbol{p}/\beta$, (3.8) can be simplified to

$$\min_{\boldsymbol{x},\boldsymbol{v}} \max_{\boldsymbol{u}} \left\{ \ell(\boldsymbol{x};\boldsymbol{y}) + \lambda\rho(\boldsymbol{v}) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{v} + \boldsymbol{u}\|^2 - \frac{\beta}{2}\|\boldsymbol{u}\|^2 \right\}. \tag{3.9}$$

The ADMM algorithm solves (3.9) by alternating the minimization of $\boldsymbol{x}$ and $\boldsymbol{v}$ with gradient ascent of $\boldsymbol{u}$, as specified in Algorithm 1. ADMM is known to converge under convex $\ell(\cdot;\boldsymbol{y})$ and $\rho(\cdot)$, and other mild conditions (see [36]).

3.2.3 Plug-and-Play ADMM

Importantly, line 3 of Algorithm 1 can be recognized as variational *denoising* of $\boldsymbol{x}_k + \boldsymbol{u}_{k-1}$ using regularization $\lambda\rho(\boldsymbol{x})$ and quadratic loss $\ell(\boldsymbol{x};\boldsymbol{r}) = \frac{1}{2\nu}\|\boldsymbol{x} - \boldsymbol{r}\|^2$, where $\boldsymbol{r} = \boldsymbol{x}_k + \boldsymbol{u}_{k-1}$ at iteration k. By "denoising," we mean recovering $\boldsymbol{x}^0$ from noisy measurements $\boldsymbol{r}$ of the form

$$\boldsymbol{r} = \boldsymbol{x}^0 + \boldsymbol{e}, \quad \boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \nu\boldsymbol{I}), \tag{3.10}$$

for some variance $\nu > 0$.

Image denoising has been studied for decades (see, e.g., the overviews [61,62]), with the result that high performance methods are now readily available. Today's state-of-the-art denoisers include those based on image-dependent filtering algorithms (e.g., BM3D [26]) or deep neural networks (e.g., TNRD [63], DnCNN [10]). Most of these denoisers are not variational in nature, i.e., they are not based on any explicit regularizer $\lambda\rho(\boldsymbol{x})$.

Leveraging the denoising interpretation of ADMM, Venkatakrishnan, Bouman, and Wolhberg [14] proposed to replace line 3 of Algorithm 1 with a call to a sophisticated image denoiser, such as BM3D, and dubbed their approach *Plug-and-Play* (PnP) ADMM. Numerical experiments show that PnP-ADMM works very well in most cases. However, when the denoiser used in PnP-ADMM comes with no explicit regularization $\rho(\boldsymbol{x})$, it is not clear what objective PnP-ADMM is minimizing, making PnP-ADMM convergence more difficult to characterize. Similar PnP algorithms have been proposed using primal-dual methods [44] and FISTA [43] in place of ADMM. See chapter 2 for a detailed review of PnP methods and their convergence properties.

Approximate message passing (AMP) algorithms [66] also perform denoising at each iteration. In fact, when $\boldsymbol{A}$ is large and i.i.d. Gaussian, AMP constructs an internal variable statistically equivalent to $\boldsymbol{r}$ in (3.10) [67]. While the earliest instances of AMP assumed separable denoising (i.e., $[\boldsymbol{f}(\boldsymbol{x})]_n = f(x_n)\ \forall n$ for some f) later instances, like [68,69], considered non-separable denoising. The paper [70] by Metzler, Maleki, and Baraniuk proposed to plug an image-specific denoising algorithm, like BM3D, into AMP. Vector AMP, which extends AMP to the broader class of "right

rotationally invariant" random matrices, was proposed in [71], and VAMP with image-specific denoising was proposed in [72]. Rigorous analyses of AMP and VAMP under non-separable denoisers were performed in [73] and [74], respectively.

3.2.4 Regularization by Denoising (RED)

As discussed in the Introduction, Romano, Elad, and Milanfar [5] proposed a radically new way to exploit an image denoiser, which they call *regularization by denoising* (RED). Given an arbitrary image denoiser $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^N$, they proposed to construct an explicit regularizer of the form

$$\rho_{\mathsf{red}}(\boldsymbol{x}) \triangleq \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})) \tag{3.11}$$

to use within the variational framework (3.2). The advantage of using an explicit regularizer is that a wide variety of optimization algorithms can be used to solve (3.2) and their convergence can be tractably analyzed.

In [5], numerical evidence is presented to show that image denoisers $\boldsymbol{f}(\cdot)$ are *locally homogeneous* (LH), i.e.,

$$(1+\epsilon)\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}\big((1+\epsilon)\boldsymbol{x}\big) \;\forall \boldsymbol{x} \tag{3.12}$$

for sufficiently small $\epsilon \in \mathbb{R} \setminus 0$. For such denoisers, Romano et al. claim [5, Eq.(28)] that $\rho_{\mathsf{red}}(\cdot)$ obeys the gradient rule

$$\nabla \rho_{\mathsf{red}}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x}). \tag{3.13}$$

If $\nabla \rho_{\mathsf{red}}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})$, then any minimizer $\widehat{\boldsymbol{x}}$ of the variational objective under quadratic loss,

$$\frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda \rho_{\mathsf{red}}(\boldsymbol{x}) \triangleq C_{\mathsf{red}}(\boldsymbol{x}), \tag{3.14}$$

must yield $\nabla C_{\mathsf{red}}(\widehat{\boldsymbol{x}}) = \mathbf{0}$, i.e., must obey

$$\mathbf{0} = \frac{1}{\sigma^2}\boldsymbol{A}^{\mathsf{T}}(\boldsymbol{A}\widehat{\boldsymbol{x}} - \boldsymbol{y}) + \lambda(\widehat{\boldsymbol{x}} - \boldsymbol{f}(\widehat{\boldsymbol{x}})). \tag{3.15}$$

Based on this line of reasoning, Romano et al. proposed several iterative algorithms that find an $\widehat{\boldsymbol{x}}$ satisfying the fixed-point condition (3.15), which we will refer to henceforth as "RED algorithms."

3.3 Clarifications on RED

In this section, we first show that the gradient expression (3.13) holds if and only if the denoiser $\boldsymbol{f}(\cdot)$ is LH and has Jacobian symmetry (JS). We then establish that many popular denoisers lack JS, such as the median filter (MF) [75], non-local means (NLM) [9], BM3D [26], TNRD [63], and DnCNN [10]. For such denoisers, the RED algorithms cannot be explained by $\rho_{\mathsf{red}}(\cdot)$ in (3.11). We also show a more general result: When a denoiser lacks JS, there exists no regularizer $\rho(\cdot)$ whose gradient expression matches (3.13). Thus, the problem is not the specific form of $\rho_{\mathsf{red}}(\cdot)$ in (3.11) but rather the broader pursuit of explicit regularization.

3.3.1 Preliminaries

We first state some definitions and assumptions. In the sequel, we denote the ith component of $\boldsymbol{f}(\boldsymbol{x})$ by $f_i(\boldsymbol{x})$, the gradient of $f_i(\cdot)$ at $\boldsymbol{x}$ by

$$\nabla f_i(\boldsymbol{x}) \triangleq \left[\frac{\partial f_i(\boldsymbol{x})}{\partial x_1} \quad \cdots \quad \frac{\partial f_i(\boldsymbol{x})}{\partial x_N}\right]^{\mathsf{T}}, \tag{3.16}$$

and the Jacobian of $\boldsymbol{f}(\cdot)$ at $\boldsymbol{x}$ by

$$J\boldsymbol{f}(\boldsymbol{x}) \triangleq \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial f_1(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_N} \\ \frac{\partial f_2(\boldsymbol{x})}{\partial x_1} & \frac{\partial f_2(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\boldsymbol{x})}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N(\boldsymbol{x})}{\partial x_1} & \frac{\partial f_N(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f_N(\boldsymbol{x})}{\partial x_N} \end{bmatrix}. \tag{3.17}$$

Without loss of generality, we take $[0, 255]^N \subset \mathbb{R}^N$ to be the set of possible images. A given denoiser $\boldsymbol{f}(\cdot)$ may involve decision boundaries $\mathcal{D} \subset [0, 255]^N$ at which its behavior changes suddenly. We assume that these boundaries are a closed set of measure zero and work instead with the open set $\mathcal{X} \triangleq (0, 255)^N \setminus \mathcal{D}$, which contains almost all images.

We furthermore assume that $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^N$ is *differentiable* on $\mathcal{X}$, which means [76, p.212] that, for any $\boldsymbol{x} \in \mathcal{X}$, there exists a matrix $\boldsymbol{J} \in \mathbb{R}^{N \times N}$ for which

$$\lim_{\boldsymbol{w} \to \boldsymbol{0}} \frac{\|\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{w}) - \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{J}\boldsymbol{w}\|}{\|\boldsymbol{w}\|} = 0. \tag{3.18}$$

When $\boldsymbol{J}$ exists, it can be shown [76, p.216] that $\boldsymbol{J} = J\boldsymbol{f}(\boldsymbol{x})$.

3.3.2 The RED Gradient

We first recall a result that was established in [5].

Lemma 1 (Local homogeneity [5])**.** *Suppose that denoiser $\boldsymbol{f}(\cdot)$ is locally homogeneous. Then $[J\boldsymbol{f}(\boldsymbol{x})]\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x})$.*

Proof. Our proof is based on differentiability and avoids the need to define a directional derivative. From (3.18), we have

$$0 = \lim_{\epsilon \to 0} \frac{\|\boldsymbol{f}(\boldsymbol{x} + \epsilon\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}) - [J\boldsymbol{f}(\boldsymbol{x})]\boldsymbol{x}\epsilon\|}{\|\epsilon\boldsymbol{x}\|} \ \forall \boldsymbol{x} \in \mathcal{X} \tag{3.19}$$

$$= \lim_{\epsilon \to 0} \frac{\|(1+\epsilon)\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}) - [J\boldsymbol{f}(\boldsymbol{x})]\boldsymbol{x}\epsilon\|}{\|\epsilon\boldsymbol{x}\|} \ \forall \boldsymbol{x} \in \mathcal{X} \tag{3.20}$$

$$= \lim_{\epsilon \to 0} \frac{\|\boldsymbol{f}(\boldsymbol{x}) - [J\boldsymbol{f}(\boldsymbol{x})]\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \ \forall \boldsymbol{x} \in \mathcal{X}, \tag{3.21}$$

where (3.20) follows from local homogeneity (3.12). Equation (3.21) implies that $[J\boldsymbol{f}(\boldsymbol{x})]\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}) \ \forall \boldsymbol{x} \in \mathcal{X}$. □

We now state one of the main results of this section.

Lemma 2 (RED gradient)**.** *For $\rho_{red}(\cdot)$ defined in (3.11),*

$$\nabla \rho_{red}(\boldsymbol{x}) = \boldsymbol{x} - \frac{1}{2}\boldsymbol{f}(\boldsymbol{x}) - \frac{1}{2}[J\boldsymbol{f}(\boldsymbol{x})]^{\mathsf{T}}\boldsymbol{x}. \tag{3.22}$$

Proof. For any $\boldsymbol{x} \in \mathcal{X}$ and $n = 1, \ldots, N$,

$$\frac{\partial \rho_{\mathsf{red}}(\boldsymbol{x})}{\partial x_n} = \frac{\partial}{\partial x_n}\frac{1}{2}\sum_{i=1}^{N}\left(x_i^2 - x_i f_i(\boldsymbol{x})\right) \tag{3.23}$$

$$= \frac{1}{2}\frac{\partial}{\partial x_n}\left(x_n^2 - x_n f_n(\boldsymbol{x}) + \sum_{i\neq n} x_i^2 - \sum_{i\neq n} x_i f_i(\boldsymbol{x})\right) \tag{3.24}$$

$$= \frac{1}{2}\left(2x_n - f_n(\boldsymbol{x}) - x_n\frac{\partial f_n(\boldsymbol{x})}{\partial x_n} - \sum_{i\neq n} x_i\frac{\partial f_i(\boldsymbol{x})}{\partial x_n}\right) \tag{3.25}$$

$$= x_n - \frac{1}{2}f_n(\boldsymbol{x}) - \frac{1}{2}\sum_{i=1}^{N} x_i\frac{\partial f_i(\boldsymbol{x})}{\partial x_n} \tag{3.26}$$

$$= x_n - \frac{1}{2}f_n(\boldsymbol{x}) - \frac{1}{2}\left[[J\boldsymbol{f}(\boldsymbol{x})]^{\mathsf{T}}\boldsymbol{x}\right]_n, \tag{3.27}$$

using the definition of $J\boldsymbol{f}(\boldsymbol{x})$ from (3.17). Collecting $\{\frac{\partial \rho_{\mathsf{red}}(\boldsymbol{x})}{\partial x_n}\}_{n=1}^{N}$ into the gradient vector (3.13) yields (3.22). □

Note that the gradient expression (3.22) differs from (3.13).

Lemma 3 (Clarification on (3.13))**.** *Suppose that the denoiser $\boldsymbol{f}(\cdot)$ is locally homogeneous. Then the RED gradient expression (3.13) holds if and only if $J\boldsymbol{f}(\boldsymbol{x}) = [J\boldsymbol{f}(\boldsymbol{x})]^{\mathsf{T}}$.*

Proof. If $J\boldsymbol{f}(\boldsymbol{x}) = [J\boldsymbol{f}(\boldsymbol{x})]^{\mathsf{T}}$, then the last term in (3.22) becomes $-\frac{1}{2}[J\boldsymbol{f}(\boldsymbol{x})]\boldsymbol{x}$, which equals $-\frac{1}{2}\boldsymbol{f}(x)$ by Lemma 1, in which case (3.22) agrees with (3.13). But if $J\boldsymbol{f}(\boldsymbol{x}) \neq [J\boldsymbol{f}(\boldsymbol{x})]^{\mathsf{T}}$, then (3.22) differs from (3.13). □

3.3.3 Impossibility of Explicit Regularization

For denoisers $\boldsymbol{f}(\cdot)$ that lack Jacobian symmetry (JS), Lemma 3 establishes that the gradient expression (3.13) does not hold. Yet (3.13) leads to the fixed-point

condition (3.15) on which all RED algorithms in [5] are based. The fact that these algorithms work well in practice suggests that "$\nabla\rho(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})$" is a desirable property for a regularizer $\rho(\boldsymbol{x})$ to have. But the regularization $\rho_{\mathsf{red}}(\boldsymbol{x})$ in (3.11) does not lead to this property when $\boldsymbol{f}(\cdot)$ lacks JS. Thus an important question is:

> *Does there exist some other regularization $\rho(\cdot)$ for which $\nabla\rho(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})$ when $\boldsymbol{f}(\cdot)$ is non-JS?*

The following theorem provides the answer.

Theorem 1 (Impossibility)**.** *Suppose that denoiser $\boldsymbol{f}(\cdot)$ has a non-symmetric Jacobian. Then there exists no regularization $\rho(\cdot)$ for which $\nabla\rho(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})$.*

Proof. To prove the theorem, we view $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^N$ as a vector field. Theorem 4.3.8 in [77] says that a vector field $\boldsymbol{f}$ is *conservative* if and only if there exists a continuously differentiable potential $\overline{\rho} : \mathcal{X} \to \mathbb{R}$ for which $\nabla\overline{\rho} = \boldsymbol{f}$. Furthermore, Theorem 4.3.10 in [77] says that if $\boldsymbol{f}$ is conservative, then the Jacobian $J\boldsymbol{f}$ is symmetric. Thus, by the contrapositive, if the Jacobian $J\boldsymbol{f}$ is *not* symmetric, then no such potential $\overline{\rho}$ exists.

To apply this result to our problem, we define

$$\rho(\boldsymbol{x}) \triangleq \frac{1}{2}\|\boldsymbol{x}\|^2 - \overline{\rho}(\boldsymbol{x}) \tag{3.28}$$

and notice that

$$\nabla\rho(\boldsymbol{x}) = \boldsymbol{x} - \nabla\overline{\rho}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x}). \tag{3.29}$$

Thus, if $J\boldsymbol{f}(\boldsymbol{x})$ is non-symmetric, then $J[\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})] = \boldsymbol{I} - J\boldsymbol{f}(\boldsymbol{x})$ is non-symmetric, which means that there exists no ρ for which (3.29) holds. □

Thus, the problem is not the specific form of $\rho_{\mathsf{red}}(\cdot)$ in (3.11) but rather the broader pursuit of explicit regularization. We note that the notion of conservative vector fields was discussed in [38, App. A] in the context of PnP algorithms, whereas here we discuss it in the context of RED.

3.3.4 Analysis of Jacobian Symmetry

The previous sections motivate an important question: Do commonly-used image denoisers have sufficient JS?

For some denoisers, JS can be studied analytically. For example, consider the "transform domain thresholding" (TDT) denoisers of the form

$$\boldsymbol{f}(\boldsymbol{x}) \triangleq \boldsymbol{W}^{\mathsf{T}} \boldsymbol{g}(\boldsymbol{W}\boldsymbol{x}), \tag{3.30}$$

where $\boldsymbol{g}(\cdot)$ performs componentwise (e.g., soft or hard) thresholding and $\boldsymbol{W}$ is some transform, as occurs in the context of wavelet shrinkage [78], with or without cycle-spinning [79]. Using $g_n'(\cdot)$ to denote the derivative of $g_n(\cdot)$, we have

$$\frac{\partial f_n(\boldsymbol{x})}{\partial x_q} = \sum_{i=1}^{N} w_{in} g_i'\left(\sum_{j=1}^{N} w_{ij} x_j\right) w_{iq} = \frac{\partial f_q(\boldsymbol{x})}{\partial x_n}, \tag{3.31}$$

and so the Jacobian of $\boldsymbol{f}(\cdot)$ is perfectly symmetric.

Another class of denoisers with perfectly symmetric Jacobians are those that produce MAP or MMSE optimal $\widehat{\boldsymbol{x}}$ under some assumed prior $\widehat{p}_{\mathsf{x}}$. In the MAP case, $\widehat{\boldsymbol{x}}$ minimizes (over $\boldsymbol{x}$) the cost $c(\boldsymbol{x};\boldsymbol{r}) = \frac{1}{2\nu}\|\boldsymbol{x}-\boldsymbol{r}\|^2 - \ln \widehat{p}_{\mathsf{x}}(\boldsymbol{x})$ for noisy input $\boldsymbol{r}$. If we define $\phi(\boldsymbol{r}) \triangleq \min_{\boldsymbol{x}} c(\boldsymbol{x};\boldsymbol{r})$, known as the Moreau-Yosida envelope of $-\ln \widehat{p}_{\mathsf{x}}$, then $\widehat{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{r}) = \boldsymbol{r} - \nu\nabla\phi(\boldsymbol{r})$, as discussed in [37] (See also [80] for insightful discussions in the context of image denoising.) The elements in the Jacobian are therefore $[J\boldsymbol{f}(\boldsymbol{r})]_{n,q} = \frac{\partial f_n(\boldsymbol{r})}{\partial r_q} = \delta_{n-q} - \nu\frac{\partial^2\phi(\boldsymbol{r})}{\partial r_q \partial r_n}$, and so the Jacobian matrix is symmetric. In

the MMSE case, we have that $\boldsymbol{f}(\boldsymbol{r}) = \boldsymbol{r} - \nabla\rho_{\mathsf{TR}}(\boldsymbol{r})$ for $\rho_{\mathsf{TR}}(\cdot)$ defined in (3.52) (see Lemma 4), and so $[J\boldsymbol{f}(\boldsymbol{r})]_{n,q} = \delta_{n-q} - \frac{\partial^2\rho_{\mathsf{TR}}(\boldsymbol{r})}{\partial r_q \partial r_n}$, again implying that the Jacobian is symmetric. But it is difficult to say anything about the Jacobian symmetry of *approximate* MAP or MMSE denoisers.

Now let us consider the more general class of denoisers

$$\boldsymbol{f}(x) = \boldsymbol{W}(\boldsymbol{x})\boldsymbol{x}, \tag{3.32}$$

sometimes called "pseudo-linear" [62]. For simplicity, we assume that $\boldsymbol{W}(\cdot)$ is differentiable on $\mathcal{X}$. In this case, using the chain rule, we have

$$\frac{\partial f_n(\boldsymbol{x})}{\partial x_q} = w_{nq}(\boldsymbol{x}) + \sum_{i=1}^{N} \frac{\partial w_{ni}(\boldsymbol{x})}{\partial x_q} x_i, \tag{3.33}$$

and so the following are sufficient conditions for Jacobian symmetry.

1. $\boldsymbol{W}(\boldsymbol{x})$ is symmetric $\forall \boldsymbol{x} \in \mathcal{X}$,
2. $\sum_{i=1}^{N} \frac{\partial w_{ni}(\boldsymbol{x})}{\partial x_q} x_i = \sum_{i=1}^{N} \frac{\partial w_{qi}(\boldsymbol{x})}{\partial x_n} x_i \ \forall \boldsymbol{x} \in \mathcal{X}$.

When $\boldsymbol{W}$ is $\boldsymbol{x}$-invariant (i.e., $\boldsymbol{f}(\cdot)$ is linear) and symmetric, both of these conditions are satisfied. This latter case was exploited for RED in [50]. The case of non-linear $\boldsymbol{W}(\cdot)$ is more complicated. Although $\boldsymbol{W}(\cdot)$ can be symmetrized (see [81,82]), it is not clear whether the second condition above will be satisfied.

3.3.5 Jacobian Symmetry Experiments

For denoisers that do not admit a tractable analysis, we can still evaluate the Jacobian of $\boldsymbol{f}(\cdot)$ at $\boldsymbol{x}$ numerically via

$$\frac{f_i(\boldsymbol{x} + \epsilon\boldsymbol{e}_n) - f_i(\boldsymbol{x} - \epsilon\boldsymbol{e}_n)}{2\epsilon} \triangleq \big[\widehat{J\boldsymbol{f}}(\boldsymbol{x})\big]_{i,n}, \tag{3.34}$$

	TDT	MF	NLM	BM3D	TNRD	DnCNN
$e^J_{\boldsymbol{f}}(\boldsymbol{x})$	5.36e-21	1.50	0.250	1.22	0.0378	0.0172

Table 3.1: Average Jacobian-symmetry error on 16×16 images

where $\boldsymbol{e}_n$ denotes the nth column of $\boldsymbol{I}_N$ and $\epsilon > 0$ is small ($\epsilon = 1 \times 10^{-3}$ in our experiments). For the purpose of quantifying JS, we define the normalized error metric

$$e^J_{\boldsymbol{f}}(\boldsymbol{x}) \triangleq \frac{\big\|\widehat{J\boldsymbol{f}}(\boldsymbol{x}) - [\widehat{J\boldsymbol{f}}(\boldsymbol{x})]^{\mathsf{T}}\big\|_F^2}{\|\widehat{J\boldsymbol{f}}(\boldsymbol{x})\|_F^2}, \tag{3.35}$$

which should be nearly zero for a symmetric Jacobian.

Table 3.1 shows[8] the average value of $e^J_{\boldsymbol{f}}(\boldsymbol{x})$ for 17 different image patches[9] of size 16×16, using denoisers that assumed a noise variance of 25^2. The denoisers tested were the TDT from (3.30) with the 2D Haar wavelet transform and soft-thresholding, the median filter (MF) [75] with a 3×3 window, non-local means (NLM) [9], BM3D [26], TNRD [63], and DnCNN [10]. Table 3.1 shows that the Jacobians of all but the TDT denoiser are far from symmetric.

Jacobian symmetry is of secondary interest; what we really care about is the accuracy of the RED gradient expressions (3.13) and (3.22). To assess gradient accuracy, we numerically evaluated the gradient of $\rho_{\mathsf{red}}(\cdot)$ at $\boldsymbol{x}$ using

$$\frac{\rho_{\mathsf{red}}(\boldsymbol{x} + \epsilon\boldsymbol{e}_n) - \rho_{\mathsf{red}}(\boldsymbol{x} - \epsilon\boldsymbol{e}_n)}{2\epsilon} \triangleq \big[\widehat{\nabla\rho_{\mathsf{red}}}(\boldsymbol{x})\big]_n \tag{3.36}$$

[8]Matlab code for the experiments is available at `http://www2.ece.ohio-state.edu/~schniter/RED/index.html`.

[9]We used the center 16×16 patches of the standard Barbara, Bike, Boats, Butterfly, Cameraman, Flower, Girl, Hat, House, Leaves, Lena, Parrots, Parthenon, Peppers, Plants, Raccoon, and Starfish test images.

$e_{\boldsymbol{f}}^{\nabla}(\boldsymbol{x})$	TDT	MF	NLM	BM3D	TNRD	DnCNN
$\nabla\rho_{\mathsf{red}}(\boldsymbol{x})$ from (3.13)	0.381	0.904	0.829	0.790	0.416	1.76
$\nabla\rho_{\mathsf{red}}(\boldsymbol{x})$ from (3.38)	0.381	1.78e-21	0.0446	0.447	0.356	1.69
$\nabla\rho_{\mathsf{red}}(\boldsymbol{x})$ from (3.22)	4.68e-19	1.75e-21	1.32e-20	4.80e-14	3.77e-19	6.76e-13

Table 3.2: Average gradient error on 16×16 images

and compared the result to the analytical expressions (3.13) and (3.22). Table 3.2 reports the normalized gradient error

$$e_{\boldsymbol{f}}^{\nabla}(\boldsymbol{x}) \triangleq \frac{\|\nabla\rho_{\mathsf{red}}(\boldsymbol{x}) - \widehat{\nabla\rho_{\mathsf{red}}}(\boldsymbol{x})\|^2}{\|\widehat{\nabla\rho_{\mathsf{red}}}(\boldsymbol{x})\|^2} \tag{3.37}$$

for the same ϵ, images, and denoisers used in Table 3.1. The results in Table 3.2 show that, for all tested denoisers, the numerical gradient $\widehat{\nabla\rho_{\mathsf{red}}}(\cdot)$ closely matches the analytical expression for $\nabla\rho_{\mathsf{red}}(\cdot)$ from (3.22), but not that from (3.13). The mismatch between $\widehat{\nabla\rho_{\mathsf{red}}}(\cdot)$ and $\nabla\rho_{\mathsf{red}}(\cdot)$ from (3.13) is partly due to insufficient JS and partly due to insufficient LH, as we establish below.

3.3.6 Local Homogeneity Experiments

Recall that the TDT denoiser has a symmetric Jacobian, both theoretically and empirically. Yet Table 3.2 reports a disagreement between the $\nabla\rho_{\mathsf{red}}(\cdot)$ expressions (3.13) and (3.22) for TDT. We now show that this disagreement is due to insufficient local homogeneity (LH).

To do this, we introduce yet another RED gradient expression,

$$\nabla\rho_{\mathsf{red}}(\boldsymbol{x}) \overset{\mathsf{LH}}{=} \boldsymbol{x} - \frac{1}{2}[J\boldsymbol{f}(\boldsymbol{x})]\boldsymbol{x} - \frac{1}{2}[J\boldsymbol{f}(\boldsymbol{x})]^{\mathsf{T}}\boldsymbol{x}, \tag{3.38}$$

which results from combining (3.22) with Lemma 1. Here, $\overset{\mathsf{LH}}{=}$ indicates that (3.38) holds under LH. In contrast, the gradient expression (3.13) holds under *both* LH and

	TDT	MF	NLM	BM3D	TNRD	DnCNN
$e_{\boldsymbol{f}}^{\mathsf{LH},1}(\boldsymbol{x})$	2.05e-8	0	1.41e-8	7.37e-7	2.18e-8	1.63e-8
$e_{\boldsymbol{f}}^{\mathsf{LH},2}(\boldsymbol{x})$	0.0205	2.26e-23	0.0141	3.80e4	2.18e-2	0.0179

Table 3.3: Average local-homogeneity error on 16×16 images

Jacobian symmetry, while the gradient expression (3.22) holds in general (i.e., even in the absence of LH and/or Jacobian symmetry). We also introduce two normalized error metrics for LH,

$$e_{\boldsymbol{f}}^{\mathsf{LH},1}(\boldsymbol{x}) \triangleq \frac{\left\|\boldsymbol{f}((1+\epsilon)\boldsymbol{x}) - (1+\epsilon)\boldsymbol{f}(\boldsymbol{x})\right\|^2}{\|(1+\epsilon)\boldsymbol{f}(\boldsymbol{x})\|^2} \tag{3.39}$$

$$e_{\boldsymbol{f}}^{\mathsf{LH},2}(\boldsymbol{x}) \triangleq \frac{\left\|[\widehat{J\boldsymbol{f}}(\boldsymbol{x})]\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})\right\|^2}{\|\boldsymbol{f}(\boldsymbol{x})\|^2}. \tag{3.40}$$

which should both be nearly zero for LH $\boldsymbol{f}(\cdot)$. Note that $e_{\boldsymbol{f}}^{\mathsf{LH},1}$ quantifies LH according to definition (3.12) and closely matches the numerical analysis of LH in [5]. Meanwhile, $e_{\boldsymbol{f}}^{\mathsf{LH},2}$ quantifies LH according to Lemma 1 and to how LH is actually used in the gradient expressions (3.13) and (3.38).

The middle row of Table 3.2 reports the average gradient error of the gradient expression (3.38), and Table 3.3 reports average LH error for the metrics $e_{\boldsymbol{f}}^{\mathsf{LH},1}$ and $e_{\boldsymbol{f}}^{\mathsf{LH},2}$. There we see that the average $e_{\boldsymbol{f}}^{\mathsf{LH},1}$ error is small for all denoisers, consistent with the experiments in [5]. But the average $e_{\boldsymbol{f}}^{\mathsf{LH},2}$ error is several orders of magnitude larger (for all but the MF denoiser). We also note that the value of $e_{\boldsymbol{f}}^{\mathsf{LH},2}$ for BM3D is several orders of magnitude higher than for the other denoisers. This result is consistent with Fig. 3.2, which shows that the cost function associated with BM3D is much less smooth than that of the other denoisers. As discussed below, these

seemingly small imperfections in LH have a significant effect on the RED gradient expressions (3.13) and (3.38).

Starting with the TDT denoiser, Table 3.2 shows that the gradient error on (3.38) is large, which can only be caused by insufficient LH. The insufficient LH is confirmed in Table 3.3, which shows that the value of $e_{\boldsymbol{f}}^{\mathsf{LH},2}(\boldsymbol{x})$ for TDT is non-negligible, especially in comparison to the value for MF.

Continuing with the MF denoiser, Table 3.1 indicates that its Jacobian is far from symmetric, while Table 3.3 indicates that it is LH. The gradient results in Table 3.2 are consistent with these behaviors: the $\nabla\rho_{\mathsf{red}}(\boldsymbol{x})$ expression (3.38) is accurate on account of LH being satisfied, but the $\nabla\rho_{\mathsf{red}}(\boldsymbol{x})$ expression (3.13) is inaccurate on account of a lack of JS.

The results for the remaining denoisers NLM, BM3D, TNRD, and BM3D show a common trend: they have non-trivial levels of *both* JS error (see Table 3.1) and LH error (see Table 3.3). As a result, the gradient expressions (3.13) and (3.38) are *both* inaccurate (see Table 3.2).

In conclusion, the experiments in this section show that the RED gradient expressions (3.13) and (3.38) are very sensitive to small imperfections in LH. Although the experiments in [5] suggested that many popular image denoisers are approximately LH, our experiments suggest that their levels of LH are insufficient to maintain the accuracy of the RED gradient expressions (3.13) and (3.38).

3.3.7 Hessian and Convexity

From (3.26), the (n, j)th element of the Hessian of $\rho_{\mathsf{red}}(\boldsymbol{x})$ equals

$$\frac{\partial^2 \rho_{\mathsf{red}}(\boldsymbol{x})}{\partial x_n \partial x_j} = \frac{\partial}{\partial x_j}\left(x_n - \frac{1}{2} f_n(\boldsymbol{x}) - \frac{1}{2}\sum_{i=1}^{N} x_i \frac{\partial f_i(\boldsymbol{x})}{\partial x_n}\right) \tag{3.41}$$

$$= \delta_{n-j} - \frac{1}{2}\frac{\partial f_n(\boldsymbol{x})}{\partial x_j} - \frac{1}{2}\frac{\partial f_j(\boldsymbol{x})}{\partial x_n} - \frac{1}{2} x_j \frac{\partial^2 f_j(\boldsymbol{x})}{\partial x_n \partial x_j} - \frac{1}{2}\sum_{i \neq j} x_i \frac{\partial^2 f_i(\boldsymbol{x})}{\partial x_n \partial x_j} \tag{3.42}$$

$$= \delta_{n-j} - \frac{1}{2}\frac{\partial f_n(\boldsymbol{x})}{\partial x_j} - \frac{1}{2}\frac{\partial f_j(\boldsymbol{x})}{\partial x_n} - \frac{1}{2}\sum_{i=1}^{N} x_i \frac{\partial^2 f_i(\boldsymbol{x})}{\partial x_n \partial x_j}. \tag{3.43}$$

where $\delta_k = 1$ if $k = 0$ and otherwise $\delta_k = 0$. Thus, the Hessian of $\rho_{\mathsf{red}}(\cdot)$ at $\boldsymbol{x}$ equals

$$H\rho_{\mathsf{red}}(\boldsymbol{x}) = \boldsymbol{I} - \frac{1}{2} J\boldsymbol{f}(\boldsymbol{x}) - \frac{1}{2}[J\boldsymbol{f}(\boldsymbol{x})]^{\mathsf{T}} - \frac{1}{2}\sum_{i=1}^{N} x_i H f_i(\boldsymbol{x}). \tag{3.44}$$

This expression can be contrasted with the Hessian expression from [5, (60)], which reads

$$\boldsymbol{I} - J\boldsymbol{f}(\boldsymbol{x}). \tag{3.45}$$

Interestingly, (3.44) differs from (3.45) even when the denoiser has a symmetric Jacobian $J\boldsymbol{f}(\boldsymbol{x})$. One implication is that, even if eigenvalues of $J\boldsymbol{f}(\boldsymbol{x})$ are limited to the interval $[0, 1]$, the Hessian $H\rho_{\mathsf{red}}(\boldsymbol{x})$ may not be positive semi-definite due to the last term in (3.44), with possibly negative implications on the convexity of $\rho_{\mathsf{red}}(\cdot)$. That said, the RED algorithms do not actually minimize the variational objective $\ell(\boldsymbol{x}; \boldsymbol{y}) + \lambda\rho_{\mathsf{red}}(\boldsymbol{x})$ for common denoisers $\boldsymbol{f}(\cdot)$ (as established in Section 3.3.8), and so the convexity of $\rho_{\mathsf{red}}(\cdot)$ may not be important in practice. We investigate the convexity of $\rho_{\mathsf{red}}(\cdot)$ numerically in Section 3.3.9.

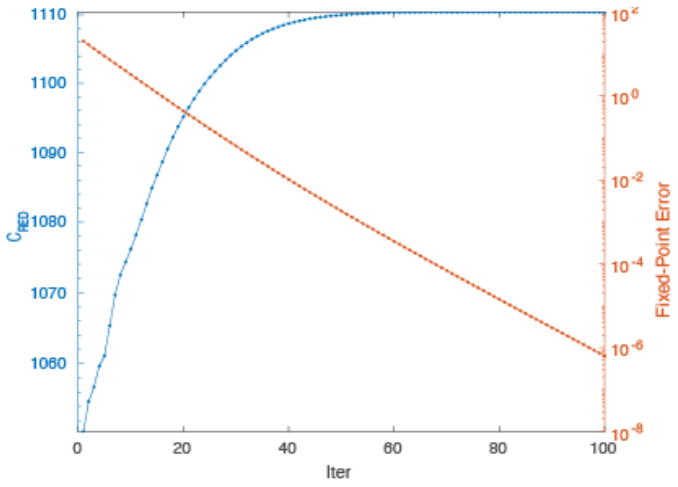


Figure 3.1: RED cost $C_{\mathsf{red}}(\boldsymbol{x}_k)$ and fixed-point error $\|\boldsymbol{A}^{\mathsf{T}}(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})/\sigma^2 + \lambda(\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_k))\|^2$ versus iteration k for $\{\boldsymbol{x}_k\}_{k=1}^K$ produced by the RED-SD algorithm from [5]. Although the fixed-point condition is asymptotically satisfied, the RED cost does not decrease with k.

3.3.8 Example RED-SD Trajectory

We now provide an example of how the RED algorithms from [5] do not necessarily minimize the variational objective $\ell(\boldsymbol{x};\boldsymbol{y}) + \lambda\rho_{\mathsf{red}}(\boldsymbol{x})$.

For a trajectory $\{\boldsymbol{x}_k\}_{k=1}^K$ produced by the steepest-descent (SD) RED algorithm from [5], Fig. 3.1 plots, versus iteration k, the RED Cost $C_{\mathsf{red}}(\boldsymbol{x}_k)$ from (3.14) and the error on the fixed-point condition (3.15), i.e., $\|\boldsymbol{g}(\boldsymbol{x}_k)\|^2$ with

$$\boldsymbol{g}(\boldsymbol{x}) \triangleq \frac{1}{\sigma^2}\boldsymbol{A}^{\mathsf{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) + \lambda\big(\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})\big). \tag{3.46}$$

For this experiment, we used the 3×3 median-filter for $\boldsymbol{f}(\cdot)$, the Starfish image, and noisy measurements $\boldsymbol{y} = \boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$ with $\sigma^2 = 20$ (i.e., $\boldsymbol{A} = \boldsymbol{I}$ in (3.14)).

Figure 3.1 shows that, although the RED-SD algorithm asymptotically satisfies the fixed-point condition (3.15), the RED cost function $C_{\mathsf{red}}(\boldsymbol{x}_k)$ does not decrease with k, as would be expected if the RED algorithms truly minimized the RED cost $C_{\mathsf{red}}(\cdot)$.

This behavior implies that any optimization algorithm that monitors the objective value $C_{\mathsf{red}}(\boldsymbol{x}_k)$ for, say, backtracking line-search (e.g., the FASTA algorithm [83]), is difficult to apply in the context of RED.

3.3.9 Visualization of RED Cost and RED-Algorithm Gradient

We now show visualizations of the RED cost $C_{\mathsf{red}}(\boldsymbol{x})$ from (3.14) and the RED algorithm's gradient field $\boldsymbol{g}(\boldsymbol{x})$ from (3.46), for various image denoisers. For this experiment, we used the Starfish image, noisy measurements $\boldsymbol{y} = \boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ with $\sigma^2 = 100$ (i.e., $\boldsymbol{A} = \boldsymbol{I}$ in (3.14) and (3.46)), and λ optimized over a grid (of 20 values logarithmically spaced between 0.0001 and 1) for each denoiser, so that the PSNR of the RED fixed-point $\widehat{\boldsymbol{x}}$ is maximized.

Figure 3.2 plots the RED cost $C_{\mathsf{red}}(\boldsymbol{x})$ and the RED algorithm's gradient field $\boldsymbol{g}(\boldsymbol{x})$ for the TDT, MF, NLM, BM3D, TNRD, and DnCNN denoisers. To visualize these quantities in two dimensions, we plotted values of $\boldsymbol{x}$ centered at the RED fixed-point $\widehat{\boldsymbol{x}}$ and varying along two randomly chosen directions. The figure shows that the minimizer of $C_{\mathsf{red}}(\boldsymbol{x})$ does not coincide with the fixed-point $\widehat{\boldsymbol{x}}$, and that the RED cost $C_{\mathsf{red}}(\cdot)$ is not always smooth or convex.

3.4 Score-Matching by Denoising

As discussed in Section 3.2.4, the RED algorithms proposed in [5] are explicitly based on gradient rule

$$\nabla \rho(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x}). \tag{3.47}$$

This rule appears to be useful since these algorithms work very well in practice. But Section 3.3 established that $\rho_{\mathsf{red}}(\cdot)$ from (3.11) does not usually satisfy (3.47). We

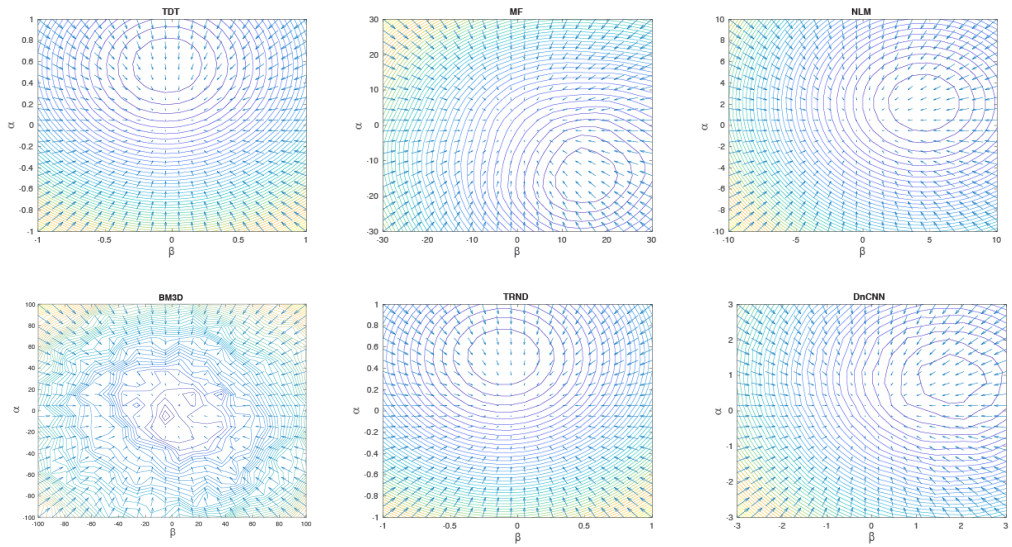


Figure 3.2: Contours show RED cost $C_{\mathsf{red}}(\boldsymbol{x}_{\alpha,\beta})$ from (3.14) and arrows show RED-algorithm gradient field $\boldsymbol{g}(\boldsymbol{x}_{\alpha,\beta})$ from (3.46) versus (α, β), where $\boldsymbol{x}_{\alpha,\beta} = \widehat{\boldsymbol{x}} + \alpha \boldsymbol{e}_1 + \beta \boldsymbol{e}_2$ with randomly chosen $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. The subplots show that the minimizer of $C_{\mathsf{red}}(\boldsymbol{x}_{\alpha,\beta})$ is not the fixed-point $\widehat{\boldsymbol{x}}$, and that $C_{\mathsf{red}}(\cdot)$ may be non-smooth and/or non-convex.

are thus motivated to seek an alternative explanation for the RED algorithms. In this section, we explain them through a framework that we call *score-matching by denoising* (SMD).

3.4.1 Tweedie Regularization

As a precursor to the SMD framework, we first propose a technique based on what we will call *Tweedie regularization*.

Recall the measurement model (3.10) used to define the "denoising" problem, repeated in (3.48) for convenience:

$$\boldsymbol{r} = \boldsymbol{x}^0 + \boldsymbol{e}, \quad \boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \nu \boldsymbol{I}). \tag{3.48}$$

To avoid confusion, we will refer to $\boldsymbol{r}$ as "pseudo-measurements" and $\boldsymbol{y}$ as "measurements." From (3.48), the likelihood of $\boldsymbol{x}^0$ is $p(\boldsymbol{r}|\boldsymbol{x}^0;\nu) = \mathcal{N}(\boldsymbol{r};\boldsymbol{x}^0,\nu\boldsymbol{I})$.

Now, suppose that we model the true image $\boldsymbol{x}^0$ as a realization of a random vector $\boldsymbol{x}$ with prior pdf $\widehat{p}_{\mathsf{x}}$. We write "$\widehat{p}_{\mathsf{x}}$" to emphasize that the model distribution may differ from the true distribution p_{x} (i.e., the distribution from which the image $\boldsymbol{x}$ is actually drawn). Under this prior model, the MMSE denoiser of $\boldsymbol{x}$ from $\boldsymbol{r}$ is

$$\mathrm{E}_{\widehat{p}_{\mathsf{x}}}\{\boldsymbol{x}|\boldsymbol{r}\} \triangleq \widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{r}), \tag{3.49}$$

and the likelihood of observing $\boldsymbol{r}$ is

$$\widehat{p}_{\mathsf{r}}(\boldsymbol{r};\nu) \triangleq \int_{\mathbb{R}^N} p(\boldsymbol{r}|\boldsymbol{x};\nu)\widehat{p}_{\mathsf{x}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \tag{3.50}$$

$$= \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{r};\boldsymbol{x},\nu\boldsymbol{I})\widehat{p}_{\mathsf{x}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}. \tag{3.51}$$

We will now define the *Tweedie regularizer* (TR) as

$$\rho_{\mathsf{TR}}(\boldsymbol{r};\nu) \triangleq -\nu \ln \widehat{p}_{\mathsf{r}}(\boldsymbol{r};\nu). \tag{3.52}$$

As we now show, $\rho_{\mathsf{TR}}(\cdot)$ has the desired property (3.47).

Lemma 4 (Tweedie)**.** *For $\rho_{\mathsf{TR}}(\boldsymbol{r};\nu)$ defined in (3.52),*

$$\nabla\rho_{\mathsf{TR}}(\boldsymbol{r};\nu) = \boldsymbol{r} - \widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{r}), \tag{3.53}$$

where $\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\cdot)$ is the MMSE denoiser from (3.49).

Proof. Equation (3.53) is a direct consequence of a classical result known as Tweedie's formula [84,85]. A short proof, from first principles, is now given for completeness.

$$\frac{\partial}{\partial r_n}\rho_{\mathsf{TR}}(\boldsymbol{r};\nu) = -\nu\frac{\partial}{\partial r_n}\ln\int_{\mathbb{R}^N}\widehat{p}_{\mathsf{x}}(\boldsymbol{x})\mathcal{N}(\boldsymbol{r};\boldsymbol{x},\nu\boldsymbol{I})\,\mathrm{d}\boldsymbol{x} \tag{3.54}$$

$$= -\frac{\nu\int_{\mathbb{R}^N}\widehat{p}_{\mathsf{x}}(\boldsymbol{x})\frac{\partial}{\partial r_n}\mathcal{N}(\boldsymbol{r};\boldsymbol{x},\nu\boldsymbol{I})\,\mathrm{d}\boldsymbol{x}}{\int_{\mathbb{R}^N}\widehat{p}_{\mathsf{x}}(\boldsymbol{x})\mathcal{N}(\boldsymbol{r};\boldsymbol{x},\nu\boldsymbol{I})\,\mathrm{d}\boldsymbol{x}} \tag{3.55}$$

$$= \frac{\int_{\mathbb{R}^N}\widehat{p}_{\mathsf{x}}(\boldsymbol{x})\mathcal{N}(\boldsymbol{r};\boldsymbol{x},\nu\boldsymbol{I})(r_n - x_n)\,\mathrm{d}\boldsymbol{x}}{\int_{\mathbb{R}^N}\widehat{p}_{\mathsf{x}}(\boldsymbol{x})\mathcal{N}(\boldsymbol{r};\boldsymbol{x},\nu\boldsymbol{I})\,\mathrm{d}\boldsymbol{x}} \tag{3.56}$$

$$= r_n - \int_{\mathbb{R}^N} x_n\frac{\widehat{p}_{\mathsf{x}}(\boldsymbol{x})\mathcal{N}(\boldsymbol{r};\boldsymbol{x},\nu\boldsymbol{I})}{\int_{\mathbb{R}^N}\widehat{p}_{\mathsf{x}}(\boldsymbol{x}')\mathcal{N}(\boldsymbol{r};\boldsymbol{x}',\nu\boldsymbol{I})\,\mathrm{d}\boldsymbol{x}'}\,\mathrm{d}\boldsymbol{x} \tag{3.57}$$

$$= r_n - \int_{\mathbb{R}^N} x_n\,\widehat{p_{\mathsf{x}|\mathsf{r}}}(\boldsymbol{x}|\boldsymbol{r};\nu)\,\mathrm{d}\boldsymbol{x} \tag{3.58}$$

$$= r_n - [\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{r})]_n, \tag{3.59}$$

where (3.56) used $\frac{\partial}{\partial r_n}\mathcal{N}(\boldsymbol{r};\boldsymbol{x},\nu\boldsymbol{I}) = \mathcal{N}(\boldsymbol{r};\boldsymbol{x},\nu\boldsymbol{I})(x_n - r_n)/\nu$. Stacking (3.59) for $n = 1,\ldots,N$ in a vector yields (3.53). □

Thus, if the TR regularizer $\rho_{\mathsf{TR}}(\cdot;\nu)$ is used in the optimization problem (3.14), then the solution $\widehat{\boldsymbol{x}}$ must satisfy the fixed-point condition (3.15) associated with the RED algorithms from [5], albeit with an MMSE-type denoiser. This restriction will be removed using the SMD framework in Section 3.4.3.

It is interesting to note that the gradient property (3.53) holds even for non-homogeneous $\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\cdot)$. This generality is important in applications under which $\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\cdot)$ is known to lack LH. For example, with a binary image $\boldsymbol{x}\in\{0,1\}^N$ modeled by $\widehat{p}_{\mathsf{x}}(\boldsymbol{x}) = \prod_{n=1}^N 0.5(\delta(x_n)+\delta(x_n-1))$, the MMSE denoiser takes the form $[\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{x})]_n = 0.5 + 0.5\tanh(x_n/\nu)$, which is not LH.

3.4.2 Tweedie Regularization as Kernel Density Estimation

We now show that TR arises naturally in the data-driven, non-parametric context through kernel-density estimation (KDE) [64].

Recall that, in most imaging applications, the true prior p_{x} is unknown, as is the true MMSE denoiser $\boldsymbol{f}_{\mathsf{mmse},\nu}(\cdot)$. There are several ways to proceed. One way is to design "by hand" an approximate prior $\widehat{p}_{\mathsf{x}}$ that leads to a computationally efficient denoiser $\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\cdot)$. But, because this denoiser is not MMSE for $\boldsymbol{x} \sim p_{\mathsf{x}}$, the performance of the resulting estimates $\widehat{\boldsymbol{x}}$ will suffer relative to $\boldsymbol{f}_{\mathsf{mmse},\nu}$.

Another way to proceed is to approximate the prior using a large corpus of training data $\{\boldsymbol{x}_t\}_{t=1}^T$. To this end, an approximate prior could be formed using the empirical estimate

$$\widehat{p}_{\mathsf{x}}(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T} \delta(\boldsymbol{x} - \boldsymbol{x}_t), \tag{3.60}$$

but a more accurate match to the true prior p_{x} can be obtained using

$$\widetilde{p}_{\mathsf{x}}(\boldsymbol{x};\nu) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{N}(\boldsymbol{x};\boldsymbol{x}_t,\nu\boldsymbol{I}) \tag{3.61}$$

with appropriately chosen $\nu > 0$, a technique known as kernel density estimation (KDE) or Parzen windowing [64]. Note that if $\widetilde{p}_{\mathsf{x}}$ is used as a surrogate for p_{x}, then the MAP optimization problem becomes

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{r}} \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{r}-\boldsymbol{y}\|^2 - \ln \widetilde{p}_{\mathsf{x}}(\boldsymbol{r};\nu) \tag{3.62}$$

$$= \arg\min_{\boldsymbol{r}} \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{r}-\boldsymbol{y}\|^2 + \lambda\rho_{\mathsf{TR}}(\boldsymbol{r};\nu) \text{ for } \lambda = \frac{1}{\nu}, \tag{3.63}$$

with $\rho_{\mathsf{TR}}(\cdot;\nu)$ from (3.50)-(3.52) constructed using $\widehat{p}_{\mathsf{x}}$ from (3.60). In summary, TR arises naturally in the data-driven approach to image recovery when KDE is used to smooth the empirical prior.

3.4.3 Score-Matching by Denoising

A limitation of the above TR framework is that it results in denoisers $\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}$ with symmetric Jacobians. (Recall the discussion of MMSE denoisers in Section 3.3.4.) To justify the use of RED algorithms with non-symmetric Jacobians, we introduce the *score-matching by denoising* (SMD) framework in this section.

Let us continue with the KDE-based MAP estimation problem (3.62). Note that $\widehat{\boldsymbol{x}}$ from (3.62) zeros the gradient of the MAP optimization objective and thus obeys the fixed-point equation

$$\frac{1}{\sigma^2}\boldsymbol{A}^{\mathsf{T}}(\boldsymbol{A}\widehat{\boldsymbol{x}}-\boldsymbol{y})-\nabla\ln\widetilde{p}_{\mathsf{x}}(\widehat{\boldsymbol{x}};\nu)=\boldsymbol{0}. \tag{3.64}$$

In principle, $\widehat{\boldsymbol{x}}$ in (3.64) could be found using gradient descent or similar techniques. However, computation of the gradient

$$\nabla\ln\widetilde{p}_{\mathsf{x}}(\boldsymbol{r};\nu)=\frac{\nabla\widetilde{p}_{\mathsf{x}}(\boldsymbol{r};\nu)}{\widetilde{p}_{\mathsf{x}}(\boldsymbol{r};\nu)}=\frac{\sum_{t=1}^{T}(\boldsymbol{x}_t-\boldsymbol{r})\mathcal{N}(\boldsymbol{r};\boldsymbol{x}_t,\nu\boldsymbol{I})}{\nu\sum_{t=1}^{T}\mathcal{N}(\boldsymbol{r};\boldsymbol{x}_t,\nu\boldsymbol{I})} \tag{3.65}$$

is too expensive for the values of T typically needed to generate a good image prior $\widetilde{p}_{\mathsf{x}}$.

A tractable alternative is suggested by the fact that

$$\nabla\ln\widetilde{p}_{\mathsf{x}}(\boldsymbol{r};\nu)=\frac{\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{r})-\boldsymbol{r}}{\nu} \tag{3.66}$$

$$\text{for }\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{r})=\frac{\sum_{t=1}^{T}\boldsymbol{x}_t\mathcal{N}(\boldsymbol{r};\boldsymbol{x}_t,\nu\boldsymbol{I})}{\sum_{t=1}^{T}\mathcal{N}(\boldsymbol{r};\boldsymbol{x}_t,\nu\boldsymbol{I})}, \tag{3.67}$$

where $\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{r})$ is the MMSE estimator of $\boldsymbol{x}\sim\widehat{p}_{\mathsf{x}}$ from $\boldsymbol{r}=\boldsymbol{x}+\mathcal{N}(\boldsymbol{0},\nu\boldsymbol{I})$. In particular, if we can construct a good approximation to $\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\cdot)$ using a denoiser $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ in a computationally efficient function class $\mathcal{F}\triangleq\{\boldsymbol{f}_{\boldsymbol{\theta}}:\boldsymbol{\theta}\in\boldsymbol{\Theta}\}$, then we can efficiently approximate the MAP problem (3.62).

This approach can be formalized using the framework of *score matching* [11], which aims to approximate the "score" (i.e., the gradient of the log-prior) rather than the prior itself. For example, suppose that we want to want to approximate the score $\nabla \ln \widetilde{p}_{\mathsf{x}}(\cdot;\nu)$. For this, Hyvärinen [11] suggested to first find the best mean-square fit among a set of computationally efficient functions $\boldsymbol{\psi}(\cdot;\boldsymbol{\theta})$, i.e., find

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathrm{E}_{\widetilde{p}_{\mathsf{x}}} \left\{ \left\| \boldsymbol{\psi}(\boldsymbol{x};\boldsymbol{\theta}) - \nabla \ln \widetilde{p}_{\mathsf{x}}(\boldsymbol{x};\nu) \right\|^2 \right\}, \tag{3.68}$$

and then to approximate the score $\nabla \ln \widetilde{p}_{\mathsf{x}}(\cdot;\nu)$ by $\boldsymbol{\psi}(\cdot;\widehat{\boldsymbol{\theta}})$. Later, in the context of denoising autoencoders, Vincent [86] showed that if one chooses

$$\boldsymbol{\psi}(\boldsymbol{x};\boldsymbol{\theta}) = \frac{\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \boldsymbol{x}}{\nu} \tag{3.69}$$

for some function $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot) \in \mathcal{F}$, then $\widehat{\boldsymbol{\theta}}$ from (3.68) can be equivalently written as

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathrm{E}_{\widehat{p}_{\mathsf{x}}} \left\{ \left\| \boldsymbol{f}_{\boldsymbol{\theta}}\big(\boldsymbol{x} + \mathcal{N}(0,\nu\boldsymbol{I})\big) - \boldsymbol{x} \right\|^2 \right\}. \tag{3.70}$$

In this case, $\boldsymbol{f}_{\widehat{\boldsymbol{\theta}}}(\cdot)$ is the MSE-optimal denoiser, averaged over $\widehat{p}_{\mathsf{x}}$ and constrained to the function class $\mathcal{F}$.

Note that the denoiser approximation error can be directly connected to the score-matching error as follows. For any denoiser $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ and any input $\boldsymbol{x}$,

$$\begin{aligned} &\left\| \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{x}) \right\|^2 \\ &= \nu^2 \left\| \frac{\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \boldsymbol{x}}{\nu} - \nabla \ln \widetilde{p}_{\mathsf{x}}(\boldsymbol{x};\nu) \right\|^2 \qquad (3.71) \\ &= \nu^2 \left\| \boldsymbol{\psi}(\boldsymbol{x};\boldsymbol{\theta}) - \nabla \ln \widetilde{p}_{\mathsf{x}}(\boldsymbol{x};\nu) \right\|^2 \qquad (3.72) \end{aligned}$$

where (3.71) follows from (3.66) and (3.72) follows from (3.69). Thus, matching the score is directly related to matching the MMSE denoiser.

Plugging the score approximation (3.69) into the fixed-point condition (3.64), we get

$$\frac{1}{\sigma^2}\boldsymbol{A}^{\mathsf{T}}(\boldsymbol{A}\widehat{\boldsymbol{x}}-\boldsymbol{y})+\lambda\big(\widehat{\boldsymbol{x}}-\boldsymbol{f}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{x}})\big)=\boldsymbol{0} \text{ for } \lambda=\frac{1}{\nu}, \tag{3.73}$$

which matches the fixed-point condition (3.15) of the RED algorithms from [5]. Here we emphasize that $\mathcal{F}$ may be constructed in such a way that $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ has a non-symmetric Jacobian, which is the case for many state-of-the-art denoisers. Also, $\boldsymbol{\theta}$ does not need to be optimized for (3.73) to hold. Finally, $\widehat{p_{\mathsf{x}}}$ need not be the empirical prior (3.60); it can be any chosen prior [86]. Thus, the score-matching-by-denoising (SMD) framework offers an explanation of the RED algorithms from [5] that holds for generic denoisers $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$, whether or not they have symmetric Jacobians, are locally homogeneous, or MMSE. Furthermore, it suggests a rationale for choosing the regularization weight λ and, in the context of KDE, the denoiser variance ν.

3.4.4 Relation to Existing Work

Tweedie's formula (3.53) has connections to Stein's Unbiased Risk Estimation (SURE) [87], as discussed in, e.g., [88, Thm. 2] and [89, Eq. (2.4)]. SURE has been used for image denoising in, e.g., [90]. Tweedie's formula was also used in [91] to interpret autoencoding-based image priors. In our work, Tweedie's formula is used to provide an interpretation for the RED algorithms through the construction of the explicit regularizer (3.52) and the approximation of the resulting fixed-point equation (3.64) via score matching.

Recently, Alain and Bengio [92] studied the contractive auto-encoders, a type of autoencoder that minimizes squared reconstruction error plus a penalty that tries to make the autoencoder as simple as possible. While previous works such as [93]

conjectured that such auto-encoders minimize an energy function, Alain and Bengio showed that they actually minimize the norm of a score (i.e., match a score to zero). Furthermore, they showed that, when the coder and decoder do not share the same weights, it is not possible to define a valid energy function because the Jacobian of the reconstruction function is not symmetric. The results in [92] parallel those in this chapter, except that they focus on auto-encoders while we focus on variational image recovery. Another small difference is that [92] uses the small-ν approximation

$$\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{x}) = \boldsymbol{x} + \nu \nabla \ln \widehat{p}_{\mathsf{x}}(\boldsymbol{x}) + o(\nu), \tag{3.74}$$

whereas we use the exact (Tweedie's) relationship (3.53), i.e.,

$$\widehat{\boldsymbol{f}}_{\mathsf{mmse},\nu}(\boldsymbol{x}) = \boldsymbol{x} + \nu \nabla \ln \widetilde{p}_{\mathsf{x}}(\boldsymbol{x}), \tag{3.75}$$

where is $\widetilde{p}_{\mathsf{x}}$ the "Gaussian blurred" version of $\widehat{p}_{\mathsf{x}}$ from (3.51).

3.5 Fast RED Algorithms

In [5], Romano et al. proposed several ways to solve the fixed-point equation (3.15). Throughout this chapter, we have been referring to these methods as "RED algorithms." In this section, we provide new interpretations of the RED-ADMM and RED-FP algorithms from [5] and we propose new RED algorithms based on accelerated proximal gradient methods.

3.5.1 RED-ADMM

The ADMM approach was summarized in Algorithm 1 for an arbitrary regularizer $\rho(\cdot)$. To apply ADMM to RED, line 3 of Algorithm 1, known as the "proximal update," must be specialized to the case where $\rho(\cdot)$ obeys (3.13) for some denoiser $\boldsymbol{f}(\cdot)$.

Algorithm 2 RED-ADMM with I Inner Iterations [5]

Require: $\ell(\cdot;\boldsymbol{y}), \boldsymbol{f}(\cdot), \beta, \lambda, \boldsymbol{v}_0, \boldsymbol{u}_0, K$, and I

1: **for** $k = 1, 2, \ldots, K$ **do**
2: $\quad \boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{v}_{k-1} + \boldsymbol{u}_{k-1}\|^2\}$
3: $\quad \boldsymbol{z}_0 = \boldsymbol{v}_{k-1}$
4: $\quad$ **for** $i = 1, 2, \ldots, I$ **do**
5: $\quad\quad \boldsymbol{z}_i = \frac{\lambda}{\lambda+\beta}\boldsymbol{f}(\boldsymbol{z}_{i-1}) + \frac{\beta}{\lambda+\beta}(\boldsymbol{x}_k + \boldsymbol{u}_{k-1})$
6: $\quad$ **end for**
7: $\quad \boldsymbol{v}_k = \boldsymbol{z}_I$
8: $\quad \boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \boldsymbol{x}_k - \boldsymbol{v}_k$
9: **end for**
10: Return $\boldsymbol{x}_K$

To do this, Romano et al. [5] proposed the following. Because $\rho(\cdot)$ is differentiable, the proximal solution $\boldsymbol{v}_k$ must obey the fixed-point relationship

$$\mathbf{0} = \lambda\nabla\rho(\boldsymbol{v}_k) + \beta(\boldsymbol{v}_k - \boldsymbol{x}_k - \boldsymbol{u}_{k-1}) \tag{3.76}$$

$$= \lambda\big(\boldsymbol{v}_k - \boldsymbol{f}(\boldsymbol{v}_k)\big) + \beta(\boldsymbol{v}_k - \boldsymbol{x}_k - \boldsymbol{u}_{k-1}) \tag{3.77}$$

$$\Leftrightarrow\ \boldsymbol{v}_k = \frac{\lambda}{\lambda+\beta}\boldsymbol{f}(\boldsymbol{v}_k) + \frac{\beta}{\lambda+\beta}(\boldsymbol{x}_k + \boldsymbol{u}_{k-1}). \tag{3.78}$$

An approximation to $\boldsymbol{v}_k$ can thus be obtained by iterating

$$\boldsymbol{z}_i = \frac{\lambda}{\lambda+\beta}\boldsymbol{f}(\boldsymbol{z}_{i-1}) + \frac{\beta}{\lambda+\beta}(\boldsymbol{x}_k + \boldsymbol{u}_{k-1}) \tag{3.79}$$

over $i = 1, \ldots, I$ with sufficiently large I, initialized at $\boldsymbol{z}_0 = \boldsymbol{v}_{k-1}$. This procedure is detailed in lines 3-6 of Algorithm 2. The overall algorithm is known as RED-ADMM.

3.5.2 Inexact RED-ADMM

Algorithm 2 gives a faithful implementation of ADMM when the number of inner iterations, I, is large. But using many inner iterations may be impractical when the denoiser is computationally expensive, as in the case of BM3D or TNRD. Furthermore, the use of many inner iterations may not be necessary.

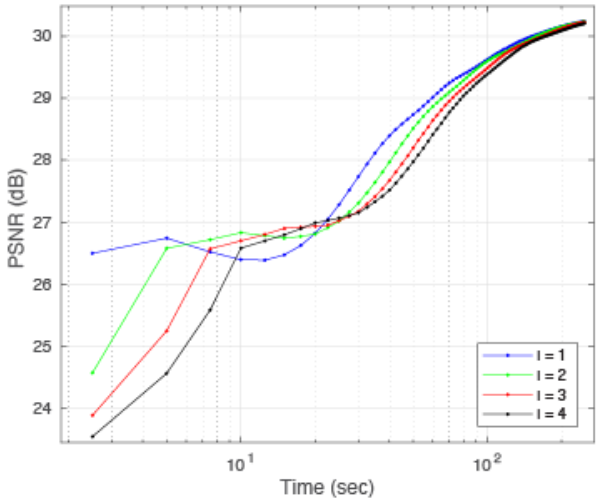


Figure 3.3: PSNR versus runtime for RED-ADMM with TNRD denoising and I inner iterations.

Algorithm 3 RED-ADMM with $I = 1$

Require: $\ell(\cdot;\boldsymbol{y}), \boldsymbol{f}(\cdot), \beta, \lambda, \boldsymbol{v}_0, \boldsymbol{u}_0$, and K

1: **for** $k = 1, 2, \dots, K$ **do**
2: $\quad \boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{v}_{k-1} + \boldsymbol{u}_{k-1}\|^2\}$
3: $\quad \boldsymbol{v}_k = \frac{\lambda}{\lambda+\beta}\boldsymbol{f}(\boldsymbol{v}_{k-1}) + \frac{\beta}{\lambda+\beta}(\boldsymbol{x}_k + \boldsymbol{u}_{k-1})$
4: $\quad \boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \boldsymbol{x}_k - \boldsymbol{v}_k$
5: **end for**
6: Return $\boldsymbol{x}_K$

For example, Fig. 3.3 plots PSNR trajectories versus runtime for TNRD-based RED-ADMM with $I = 1, 2, 3, 4$ inner iterations. For this experiment, we used the deblurring task described in Section 3.5.7, but similar behaviors can be observed in other applications of RED. Figure 3.3 suggests that $I = 1$ inner iterations gives the fastest convergence. Note that [5] also used $I = 1$ when implementing RED-ADMM.

With $I = 1$ inner iterations, RED-ADMM simplifies down to the 3-step iteration summarized in Algorithm 3. Since Algorithm 3 looks quite different than standard

ADMM (recall Algorithm 1), one might wonder whether there exists another interpretation of Algorithm 3. Noting that line 3 can be rewritten as

$$\boldsymbol{v}_k = \boldsymbol{v}_{k-1} - \frac{1}{\lambda+\beta}\big[\lambda\nabla\rho(\boldsymbol{v}_{k-1}) + \beta(\boldsymbol{v}_{k-1} - \boldsymbol{x}_k - \boldsymbol{u}_{k-1})\big] \tag{3.80}$$

$$= \boldsymbol{v}_{k-1} - \frac{1}{\lambda+\beta}\nabla\left[\lambda\rho(\boldsymbol{v}) + \frac{\beta}{2}\|\boldsymbol{v} - \boldsymbol{x}_k - \boldsymbol{u}_{k-1}\|^2\right]_{\boldsymbol{v}=\boldsymbol{v}_{k-1}} \tag{3.81}$$

we see that the $I = 1$ version of inexact RED-ADMM replaces the proximal step with a gradient-descent step under stepsize $1/(\lambda+\beta)$. Thus the algorithm is reminiscent of the proximal gradient (PG) algorithm [94,95]. We will discuss PG further in the sequel.

3.5.3 Majorization-Minimization and Proximal-Gradient RED

We now propose a proximal-gradient approach inspired by *majorization minimization* (MM) [96]. As proposed in [97], we use a quadratic upper-bound,

$$\overline{\rho}(\boldsymbol{x};\boldsymbol{x}_k) \triangleq \rho(\boldsymbol{x}_k) + [\nabla\rho(\boldsymbol{x}_k)]^{\mathsf{T}}\big(\boldsymbol{x} - \boldsymbol{x}_k\big) + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2, \tag{3.82}$$

on the regularizer $\rho(\boldsymbol{x})$, in place of $\rho(\boldsymbol{x})$ itself, at the kth algorithm iteration. Note that if $\rho(\cdot)$ is convex and $\nabla\rho(\cdot)$ is L_ρ-Lipschitz, then $\overline{\rho}(\boldsymbol{x};\boldsymbol{x}_k)$ "majorizes" $\rho(\boldsymbol{x})$ at $\boldsymbol{x}_k$ when $L \geq L_\rho$, i.e.,

$$\overline{\rho}(\boldsymbol{x};\boldsymbol{x}_k) \geq \rho(\boldsymbol{x}) \;\forall \boldsymbol{x} \in \mathcal{X} \tag{3.83}$$

$$\overline{\rho}(\boldsymbol{x}_k;\boldsymbol{x}_k) = \rho(\boldsymbol{x}_k). \tag{3.84}$$

The majorized objective can then be minimized using the *proximal gradient* (PG) algorithm [94,95] (also known as forward-backward splitting) as follows. From (3.82),

Algorithm 4 RED-PG Algorithm

Require: $\ell(\cdot;\boldsymbol{y}), \boldsymbol{f}(\cdot), \lambda, \boldsymbol{v}_0, L > 0$, and K

1: **for** $k = 1, 2, \ldots, K$ **do**
2: $\quad \boldsymbol{x}_k = \arg\min_{\boldsymbol{x}}\{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda L}{2}\|\boldsymbol{x} - \boldsymbol{v}_{k-1}\|^2\}$
3: $\quad \boldsymbol{v}_k = \frac{1}{L}\boldsymbol{f}(\boldsymbol{x}_k) - \frac{1-L}{L}\boldsymbol{x}_k$
4: **end for**
5: Return $\boldsymbol{x}_K$

note that the majorized objective can be written as

$$\begin{aligned}
&\ell(\boldsymbol{x};\boldsymbol{y}) + \lambda\overline{\rho}(\boldsymbol{x};\boldsymbol{x}_k) \\
&\quad = \ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda L}{2}\left\|\boldsymbol{x} - \left(\boldsymbol{x}_k - \frac{1}{L}\nabla\rho(\boldsymbol{x}_k)\right)\right\|^2 + \text{const} \qquad (3.85)\\
&\quad = \ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda L}{2}\Bigg\|\boldsymbol{x} - \underbrace{\left(\boldsymbol{x}_k - \frac{1}{L}\big(\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_k)\big)\right)}_{\triangleq\, \boldsymbol{v}_k}\Bigg\|^2 + \text{const}, \qquad (3.86)
\end{aligned}$$

where (3.86) follows from assuming (3.47), which is the basis for all RED algorithms. The RED-PG algorithm then alternately updates $\boldsymbol{v}_k$ as per the gradient step in (3.86) and updates $\boldsymbol{x}_{k+1}$ according to the proximal step

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}}\left\{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda L}{2}\|\boldsymbol{x} - \boldsymbol{v}_k\|^2\right\}, \qquad (3.87)$$

as summarized in Algorithm 4. Convergence is guaranteed if $L \geq L_\rho$; see [94,95] for details.

We now show that RED-PG with $L = 1$ is identical to the "fixed point" (FP) RED algorithm proposed in [5]. First, notice from Algorithm 4 that $\boldsymbol{v}_k = \boldsymbol{f}(\boldsymbol{x}_k)$ when $L = 1$, in which case

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}}\left\{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x}_{k-1})\|^2\right\}. \qquad (3.88)$$

For the quadratic loss $\ell(\boldsymbol{x};\boldsymbol{y}) = \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2$, (3.88) becomes

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2 + \frac{\lambda}{2}\|\boldsymbol{x}-\boldsymbol{f}(\boldsymbol{x}_{k-1})\|^2 \right\} \tag{3.89}$$

$$= \Big(\frac{1}{\sigma^2}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \lambda\boldsymbol{I}\Big)^{-1}\Big(\frac{1}{\sigma^2}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{y} + \lambda\boldsymbol{f}(\boldsymbol{x}_{k-1})\Big), \tag{3.90}$$

which is exactly the RED-FP update [5, (37)]. Thus, (3.88) generalizes [5, (37)] to possibly non-quadratic[10] loss $\ell(\cdot;\boldsymbol{y})$, and RED-PG generalizes RED-FP to arbitrary $L > 0$. More importantly, the PG framework facilitates algorithmic acceleration, as we describe below.

The RED-PG and inexact RED-ADMM-$I=1$ algorithms show interesting similarities: both alternate a proximal update on the loss with a gradient update on the regularization, where the latter term manifests as a convex combination between the denoiser output and another term. The difference is that RED-ADMM-$I=1$ includes an extra state variable, $\boldsymbol{u}_k$. The experiments in Section 3.5.7 suggest that this extra state variable is not necessarily advantageous.

3.5.4 Dynamic RED-PG

Recalling from (3.86) that $1/L$ acts as a stepsize in the PG gradient step, it may be possible to speed up PG by decreasing L, although making L too small can prevent convergence. If $\rho(\cdot)$ was known, then a line search could be used, at each iteration k, to find the smallest value of L that guarantees the majorization of $\rho(\boldsymbol{x})$ by $\overline{\rho}(\boldsymbol{x};\boldsymbol{x}_k)$ [94]. However, with a non-LH or non-JS denoiser, it is not possible to evaluate $\rho(\cdot)$, preventing such a line search.

[10]The extension to non-quadratic loss is important for applications like phase-retrieval, where RED has been successfully applied [98].

Algorithm 5 RED-DPG Algorithm

Require: $\ell(\cdot;\boldsymbol{y}), \boldsymbol{f}(\cdot), \lambda, \boldsymbol{v}_0, L_0 > 0, L_\infty > 0$, and K

1: **for** $k = 1, 2, \ldots, K$ **do**

2: $\quad \boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda L_{k-1}}{2}\|\boldsymbol{x} - \boldsymbol{v}_{k-1}\|^2\}$

3: $\quad L_k = \left(\frac{1}{L_\infty} + (\frac{1}{L_0} - \frac{1}{L_\infty})\frac{1}{\sqrt{k+1}}\right)^{-1}$

4: $\quad \boldsymbol{v}_k = \frac{1}{L_k}\boldsymbol{f}(\boldsymbol{x}_k) - \frac{1-L_k}{L_k}\boldsymbol{x}_k$

5: **end for**

6: Return $\boldsymbol{x}_K$

We thus propose to vary L_k (i.e., the value of L at iteration k) according to a fixed schedule. In particular, we propose to select L_0 and L_∞, and smoothly interpolate between them at intermediate iterations k. One interpolation scheme that works well in practice is summarized in line 3 of Algorithm 5. We refer to this approach as "dynamic PG" (DPG). The numerical experiments in Section 3.5.7 suggest that, with appropriate selection of L_0 and L_∞, RED-DPG can be significantly faster than RED-FP.

3.5.5 Accelerated RED-PG

Another well-known approach to speeding up PG is to apply momentum to the $\boldsymbol{v}_k$ term in Algorithm 4 [94], often known as "acceleration." An accelerated PG (APG) approach to RED is detailed in Algorithm 6. There, the momentum in line 5 takes the same form as in FISTA [41]. The numerical experiments in Section 3.5.7 suggest that RED-APG is the fastest among the RED algorithms discussed above.

By leveraging the principle of vector extrapolation (VE) [99], a different approach to accelerating RED algorithms was recently proposed in [100]. Algorithmically, the approach in [100] is much more complicated than the PG-DPG and PG-APG methods proposed above. In fact, we have been unable to arrive at an implementation of [100]

Algorithm 6 RED-APG Algorithm

Require: $\ell(\cdot;\boldsymbol{y}), \boldsymbol{f}(\cdot), \lambda, \boldsymbol{v}_0, L > 0$, and K

1: $t_0 = 1$
2: **for** $k = 1, 2, \ldots, K$ **do**
3: $\quad \boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda L}{2}\|\boldsymbol{x} - \boldsymbol{v}_{k-1}\|^2\}$
4: $\quad t_k = \frac{1+\sqrt{1+4t_{k-1}^2}}{2}$
5: $\quad \boldsymbol{z}_k = \boldsymbol{x}_k + \frac{t_{k-1}-1}{t_k}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$
6: $\quad \boldsymbol{v}_k = \frac{1}{L}\boldsymbol{f}(\boldsymbol{z}_k) - \frac{1-L}{L}\boldsymbol{z}_k$
7: **end for**
8: Return $\boldsymbol{x}_K$

that reproduces the results in that paper, and the authors have not been willing to share their implementation with us. Thus, we cannot comment further on the difference in performance between our PG-DPG and PG-APG schemes and the one in [100].

3.5.6 Convergence of RED-PG

Recalling Theorem 1, the RED algorithms do not minimize an explicit cost function but rather seek fixed points of (3.15). Therefore, it is important to know whether they actually converge to any one fixed point. Below, we use the theory of non-expansive and α-averaged operators to establish the convergence of RED-PG to a fixed point under certain conditions.

First, an operator $\boldsymbol{B}(\cdot)$ is said to be *non-expansive* if its Lipschitz constant is at most 1 [101]. Next, for $\alpha \in (0, 1)$, an operator $\boldsymbol{P}(\cdot)$ is said to be α-*averaged* if

$$\boldsymbol{P}(\boldsymbol{x}) = \alpha \boldsymbol{B}(\boldsymbol{x}) + (1 - \alpha)\boldsymbol{x} \tag{3.91}$$

for some non-expansive $\boldsymbol{B}(\cdot)$. Furthermore, if $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are α_1 and α_2-averaged, respectively, then [101, Prop. 4.32] establishes that the composition $\boldsymbol{P}_2 \circ \boldsymbol{P}_1$ is α-averaged with

$$\alpha = \frac{2}{1 + \frac{1}{\max\{\alpha_1, \alpha_2\}}}. \tag{3.92}$$

Recalling RED-PG from Algorithm 4, let us define an operator called $\boldsymbol{T}(\cdot)$ that summarizes one algorithm iteration:

$$\begin{aligned}
&\boldsymbol{T}(\boldsymbol{x}) \\
&\triangleq \arg\min_{\boldsymbol{z}} \left\{ \ell(\boldsymbol{z}; \boldsymbol{y}) + \tfrac{\lambda L}{2} \left\| \boldsymbol{z} - \left(\tfrac{1}{L} \boldsymbol{f}(\boldsymbol{x}) - \tfrac{1-L}{L} \boldsymbol{x} \right) \right\|^2 \right\} && (3.93) \\
&= \operatorname{prox}_{\ell/(\lambda L)} \left(\tfrac{1}{L} (\boldsymbol{f}(\boldsymbol{x}) - (1-L)\boldsymbol{x}) \right) && (3.94)
\end{aligned}$$

Lemma 5. *If $\ell(\cdot)$ is proper, convex, and continuous; $\boldsymbol{f}(\cdot)$ is non-expansive; and $L > 1$, then $\boldsymbol{T}(\cdot)$ from (3.94) is α-averaged with $\alpha = \max\{\frac{2}{1+L}, \frac{2}{3}\}$.*

Proof. First, because $\ell(\cdot)$ is proper, convex, and continuous, we know that the proximal operator $\operatorname{prox}_{\ell/(\lambda L)}(\cdot)$ is α-averaged with $\alpha = 1/2$ [101]. Then, by definition, $\frac{1}{L}\boldsymbol{f}(\boldsymbol{z}) - \frac{1-L}{L}\boldsymbol{z}$ is α-averaged with $\alpha = 1/L$. From (3.94), $\boldsymbol{T}(\cdot)$ is the composition of these two α-averaged operators, and so from (3.92) we have that $\boldsymbol{T}(\cdot)$ is α-averaged with $\alpha = \max\{\frac{2}{1+L}, \frac{2}{3}\}$. □

With Lemma 5, we can prove the convergence of RED-PG.

Theorem 2. *If $\ell(\cdot)$ is proper, convex, and continuous; $\boldsymbol{f}(\cdot)$ is non-expansive; $L > 1$; and $\boldsymbol{T}(\cdot)$ from (3.94) has at least one fixed point, then RED-PG converges.*

Proof. From (3.94), we have that Algorithm 4 is equivalent to

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \boldsymbol{T}(\boldsymbol{x}_k) && (3.95) \\
&= \alpha \boldsymbol{B}(\boldsymbol{x}_k) + (1-\alpha)\boldsymbol{x}_k && (3.96)
\end{aligned}$$

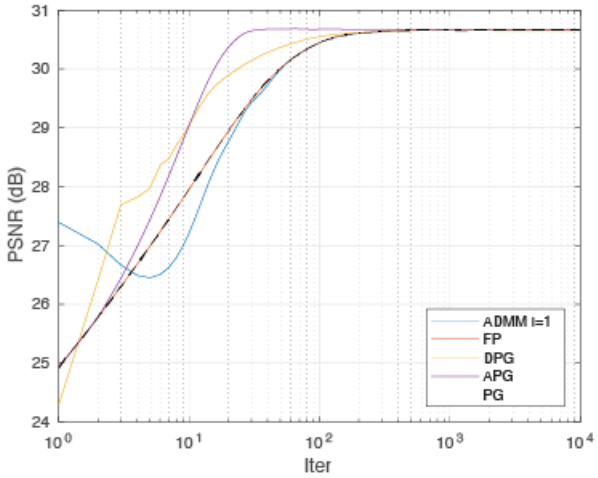


Figure 3.4: PSNR versus iteration for RED algorithms with TNRD denoising when deblurring the starfish.

where $\boldsymbol{B}(\cdot)$ is an implicit non-expansive operator that must exist under the definition of α-averaged operators from (3.91). The iteration (3.96) can be recognized as a Mann iteration [37], since $\alpha \in (0, 1)$. Thus, from [101, Thm. 5.14], $\{\boldsymbol{x}_k\}$ is a convergent sequence, in that there exists a fixed point $\boldsymbol{x}_\star \in \mathbb{R}^N$ such that $\lim_{k\to\infty} \|\boldsymbol{x}_k - \boldsymbol{x}_\star\| = 0$. □

We note that similar Mann-based techniques were used in [45,48] to prove the convergence of PnP-based algorithms. Also, we conjecture that similar techniques may be used to prove the convergence of other RED algorithms, but we leave the details to future work. Experiments in Section 3.5.7 numerically study the convergence behavior of several RED algorithms with different image denoisers $\boldsymbol{f}(\cdot)$.

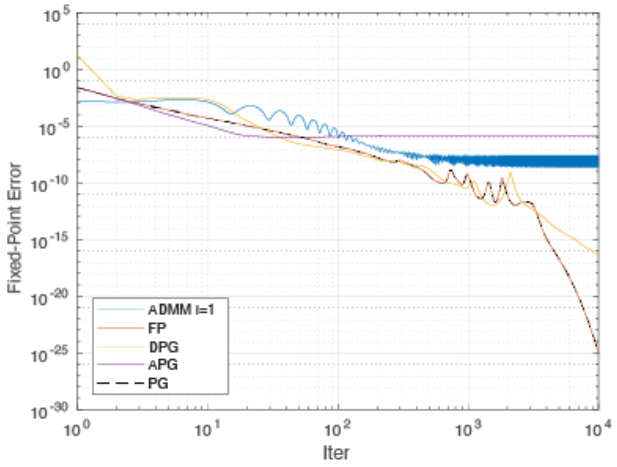


Figure 3.5: Fixed-point error versus iteration for RED algorithms with TNRD denoising when deblurring the starfish.

3.5.7 Algorithm Comparison: Image Deblurring

We now compare the performance of the RED algorithms discussed above (i.e., inexact ADMM, FP, DPG, APG, and PG) on the image deblurring problem considered in [5, Sec. 6.1]. For these experiments, the measurements $\boldsymbol{y}$ were constructed using a 9×9 uniform blur kernel for $\boldsymbol{A}$ and using AWGN with variance $\sigma^2 = 2$. As stated earlier, the image $\boldsymbol{x}$ is normalized to have pixel intensities in the range $[0, 255]$.

For the first experiment, we used the TNRD denoiser. The various algorithmic parameters were chosen based on the recommendations in [5]: the regularization weight was $\lambda = 0.02$, the ADMM penalty parameter was $\beta = 0.001$, and the noise variance assumed by the denoiser was $\nu = 3.25^2$. The proximal step on $\ell(\boldsymbol{x}; \boldsymbol{y})$, given in (3.90), was implemented with an FFT. For RED-DPG we used[11] $L_0 = 0.2$ and

[11]Matlab code for these experiments is available at `http://www2.ece.ohio-state.edu/~schniter/RED/index.html`.

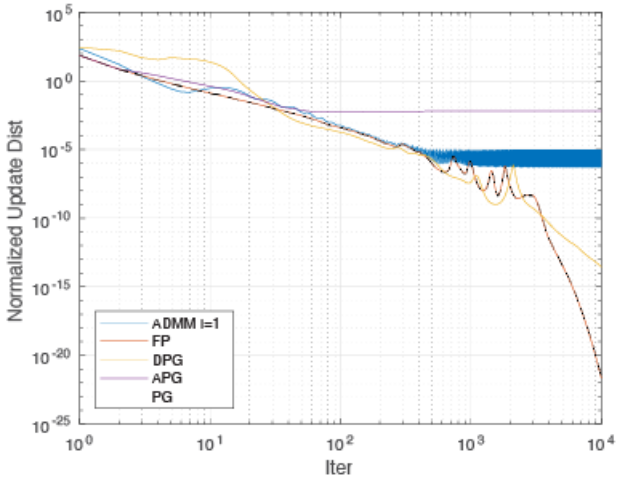


Figure 3.6: Update distance versus iteration for RED algorithms with TNRD denoising when deblurring the starfish.

$L_\infty = 2$, for RED-APG we used $L = 1$, and for RED-PG we used $L = 1.01$ since Theorem 2 motivates $L > 1$.

Figure 3.4 shows

$$\text{PSNR}_k \triangleq -10 \log_{10} \left(\frac{1}{N256^2} \|\boldsymbol{x} - \widehat{\boldsymbol{x}}_k\|^2 \right)$$

versus iteration k for the starfish test image. In the figure, the proposed RED-DPG and RED-APG algorithms appear significantly faster than the RED-FP and RED-ADMM-$I = 1$ algorithms proposed in [5]. For example, RED-APG reaches PSNR $= 30$ in 15 iterations whereas RED-FP and inexact RED-ADMM-$I = 1$ take about 50 iterations.

Figure 3.5 shows the fixed-point error

$$\frac{1}{N} \left\| \frac{1}{\sigma^2} \boldsymbol{A}^H (\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}) + \lambda (\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_k)) \right\|^2$$

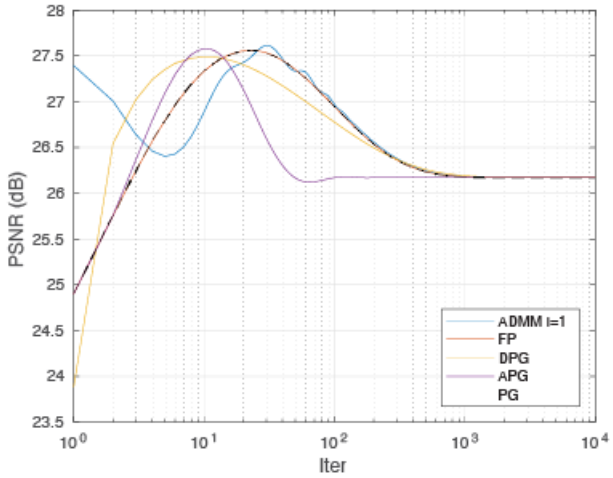


Figure 3.7: PSNR versus iteration for RED algorithms with TDT denoising when deblurring the starfish.

verus iteration k. All but the RED-APG and RED-ADMM algorithms appear to converge to the solution set of the fixed-point equation (3.15). The RED-APG and RED-ADMM algorithms appear to approximately satisfy the fixed-point equation (3.15), but not exactly satisfy (3.15), since the fixed-point error does not decay to zero.

Figure 3.6 shows the update distance $\frac{1}{N}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2$ vs. iteration k for the algorithms under test. For most algorithms, the update distance appears to be converging to zero, but for RED-APG and RED-ADMM it does not. This suggests that the RED-APG and RED-ADMM algorithms are converging to a limit cycle rather than a unique limit point.

Next, we replace the TNRD denoiser with the TDT denoiser from (3.30) and repeat the previous experiments. For the TDT denoiser, we used a Haar-wavelet based orthogonal discrete wavelet transform (DWT) $\boldsymbol{W}$, with the maximum number of

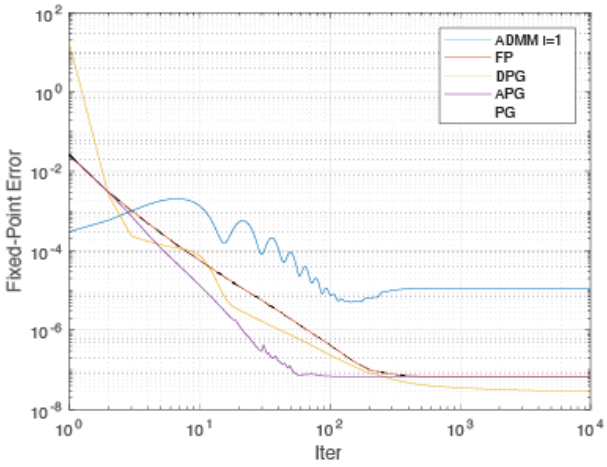


Figure 3.8: Fixed-point error versus iteration for RED algorithms with TDT denoising when deblurring the starfish.

decomposition levels, and a soft-thresholding function $\boldsymbol{g}(\cdot)$ with threshold value 0.001. Unlike the TNRD denoiser, this TDT denoiser is the proximal operator associated with a convex cost function, and so we know that it is $\frac{1}{2}$-averaged and non-expansive.

Figure 3.7 shows PSNR versus iteration with TDT denoising. Interestingly, the final PSNR values appear to be nearly identical among all algorithms under test, but more than 1 dB worse than the values around iteration 20. Figure 3.8 shows the fixed-point error vs. iteration for this experiment. There, the errors of most algorithms converge to a value near 10^{-7}, but then remain at that value. Noting that RED-PG satisfies the conditions of Theorem 2 (i.e., convex loss, non-expansive denoiser, $L > 1$), it should converge to a fixed-point of (3.15). Therefore, we attribute the fixed-point error saturation in Fig. 3.8 to issues with numerical precision. Figure 3.9 shows the normalized distance versus iteration with TDT denoising. There, the distance decreases to zero for all algorithms under test.

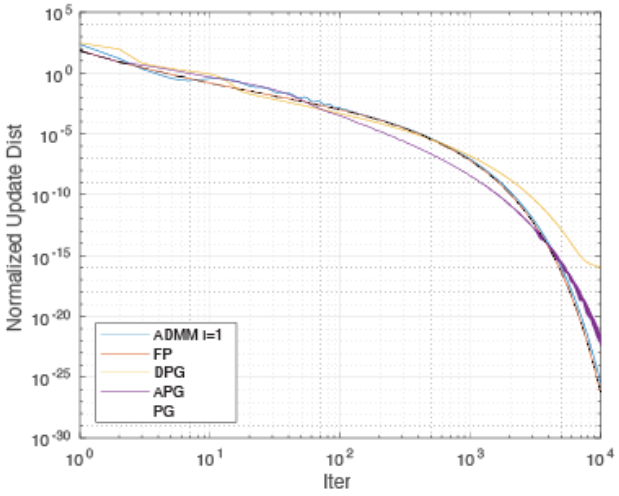


Figure 3.9: Update distance versus iteration for RED algorithms with TDT denoising when deblurring the starfish.

We emphasize that the proposed RED-DPG, RED-APG, and RED-PG algorithms seek to solve exactly the same fixed-point equation (3.15) sought by the RED-SD, RED-ADMM, and RED-FP algorithms proposed in [5]. The excellent quality of the RED fixed-points was firmly established in [5], both qualitatively and quantitatively, in comparison to existing state-of-the-art methods like PnP-ADMM [14]. For further details on these comparisons, including examples of images recovered by the RED algorithms, we refer the interested reader to [5].

3.6 Equilibrium View of RED Algorithms

Like the RED algorithms, PnP-ADMM [14] repeatedly calls a denoiser $\boldsymbol{f}(\cdot)$ in order to solve an inverse problem. In [48], Buzzard, Sreehari, and Bouman show that PnP-ADMM finds a "consensus equilibrium" solution rather than a minimum of any

explicit cost function. By consensus equilibrium, we mean a solution $(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{u}})$ to

$$\widehat{\boldsymbol{x}} = F(\widehat{\boldsymbol{x}} + \widehat{\boldsymbol{u}}) \tag{3.97a}$$

$$\widehat{\boldsymbol{x}} = G(\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{u}}) \tag{3.97b}$$

for some functions $F, G : \mathbb{R}^N \to \mathbb{R}^N$. For PnP-ADMM, these functions are [48]

$$F_{\mathsf{pnp}}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}} \left\{ \ell(\boldsymbol{x}; \boldsymbol{y}) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{v}\|^2 \right\} \tag{3.98}$$

$$G_{\mathsf{pnp}}(\boldsymbol{v}) = \boldsymbol{f}(\boldsymbol{v}). \tag{3.99}$$

3.6.1 RED Equilibrium Conditions

We now show that the RED algorithms also find consensus equilibrium solutions, but with $G \neq G_{\mathsf{pnp}}$. First, recall ADMM Algorithm 1 with explicit regularization $\rho(\cdot)$. By taking iteration $k \to \infty$, it becomes clear that the ADMM solutions must satisfy the equilibrium condition (3.97) with

$$F_{\mathsf{admm}}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}} \left\{ \ell(\boldsymbol{x}; \boldsymbol{y}) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{v}\|^2 \right\} \tag{3.100}$$

$$G_{\mathsf{admm}}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}} \left\{ \lambda\rho(\boldsymbol{x}) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{v}\|^2 \right\}, \tag{3.101}$$

where we note that $F_{\mathsf{admm}} = F_{\mathsf{pnp}}$.

The RED-ADMM algorithm can be considered as a special case of ADMM Algorithm 1 under which $\rho(\cdot)$ is differentiable with $\nabla\rho(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})$, for a given denoiser $\boldsymbol{f}(\cdot)$. We can thus find $G_{\mathsf{red\text{-}admm}}(\cdot)$, i.e., the RED-ADMM version of $G(\cdot)$ satisfying the equilibrium condition (3.97b), by solving the right side of (3.101) under $\nabla\rho(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})$. Similarly, we see that the RED-ADMM version of $F(\cdot)$ is identical to the ADMM version of $F(\cdot)$ from (3.100). Now, the $\widehat{\boldsymbol{x}} = G_{\mathsf{red\text{-}admm}}(\boldsymbol{v})$ that solves

the right side of (3.101) under differentiable $\rho(\cdot)$ with $\nabla\rho(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})$ must obey

$$\mathbf{0} = \lambda\nabla\rho(\widehat{\boldsymbol{x}}) + \beta(\widehat{\boldsymbol{x}} - \boldsymbol{v}) \tag{3.102}$$

$$= \lambda\big(\widehat{\boldsymbol{x}} - \boldsymbol{f}(\widehat{\boldsymbol{x}})\big) + \beta(\widehat{\boldsymbol{x}} - \boldsymbol{v}), \tag{3.103}$$

which we note is a special case of (3.15). Continuing, we find that

$$\mathbf{0} = \lambda\big(\widehat{\boldsymbol{x}} - \boldsymbol{f}(\widehat{\boldsymbol{x}})\big) + \beta(\widehat{\boldsymbol{x}} - \boldsymbol{v}) \tag{3.104}$$

$$\Leftrightarrow \mathbf{0} = \frac{\lambda+\beta}{\beta}\widehat{\boldsymbol{x}} - \frac{\lambda}{\beta}\boldsymbol{f}(\widehat{\boldsymbol{x}}) - \boldsymbol{v} \tag{3.105}$$

$$\Leftrightarrow \boldsymbol{v} = \left(\frac{\lambda+\beta}{\beta}\boldsymbol{I} - \frac{\lambda}{\beta}\boldsymbol{f}\right)(\widehat{\boldsymbol{x}}) \tag{3.106}$$

$$\Leftrightarrow \widehat{\boldsymbol{x}} = \left(\frac{\lambda+\beta}{\beta}\boldsymbol{I} - \frac{\lambda}{\beta}\boldsymbol{f}\right)^{-1}(\boldsymbol{v}) = G_{\mathsf{red\text{-}admm}}(\boldsymbol{v}), \tag{3.107}$$

where $\boldsymbol{I}$ represents the identity operator and $(\cdot)^{-1}$ represents the functional inverse. In summary, RED-ADMM with denoiser $\boldsymbol{f}(\cdot)$ solves the consensus equilibrium problem (3.97) with $F = F_{\mathsf{admm}}$ from (3.100) and $G = G_{\mathsf{red\text{-}admm}}$ from (3.107).

Next we establish an equilibrium result for RED-PG. Defining $\boldsymbol{u}_k = \boldsymbol{v}_k - \boldsymbol{x}_k$ and taking $k \to \infty$ in Algorithm 4, it can be seen that the fixed points of RED-PG obey (3.97a) for

$$F_{\mathsf{red\text{-}pg}}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}}\left\{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda L}{2}\|\boldsymbol{x} - \boldsymbol{v}\|^2\right\}. \tag{3.108}$$

Furthermore, from line 3 of Algorithm 4, it can be seen that the RED-PG fixed points also obey

$$\widehat{\boldsymbol{u}} = \frac{1}{L}\left(\boldsymbol{f}(\widehat{\boldsymbol{x}}) - \widehat{\boldsymbol{x}}\right) \tag{3.109}$$

$$\Leftrightarrow \widehat{\boldsymbol{x}} - \widehat{\boldsymbol{u}} = \widehat{\boldsymbol{x}} - \frac{1}{L}\left(\boldsymbol{f}(\widehat{\boldsymbol{x}}) - \widehat{\boldsymbol{x}}\right) \tag{3.110}$$

$$= \left(\frac{L+1}{L}\boldsymbol{I} - \frac{1}{L}\boldsymbol{f}\right)(\widehat{\boldsymbol{x}}) \tag{3.111}$$

$$\Leftrightarrow \widehat{\boldsymbol{x}} = \left(\frac{L+1}{L}\boldsymbol{I} - \frac{1}{L}\boldsymbol{f}\right)^{-1}(\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{u}}), \tag{3.112}$$

which matches (3.97b) when $G = G_{\text{red-pg}}$ for

$$G_{\text{red-pg}}(\boldsymbol{v}) = \left(\frac{L+1}{L}\boldsymbol{I} - \frac{1}{L}\boldsymbol{f}\right)^{-1}(\boldsymbol{v}). \tag{3.113}$$

Note that $G_{\text{red-pg}} = G_{\text{red-admm}}$ when $L = \beta/\lambda$.

3.6.2 Interpreting the RED Equilibria

The equilibrium conditions provide additional interpretations of the RED algorithms. To see how, first recall that the RED equilibrium $(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{u}})$ satisfies

$$\widehat{\boldsymbol{x}} = F_{\text{red-pg}}(\widehat{\boldsymbol{x}} + \widehat{\boldsymbol{u}}) \tag{3.114a}$$

$$\widehat{\boldsymbol{x}} = G_{\text{red-pg}}(\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{u}}), \tag{3.114b}$$

or an analogous pair of equations involving $F_{\text{red-admm}}$ and $G_{\text{red-admm}}$. Thus, from (3.108), (3.109), and (3.114a), we have that

$$\widehat{\boldsymbol{x}} = F_{\text{red-pg}}\left(\widehat{\boldsymbol{x}} + \frac{1}{L}(\boldsymbol{f}(\widehat{\boldsymbol{x}}) - \widehat{\boldsymbol{x}})\right) \tag{3.115}$$

$$= F_{\text{red-pg}}\left(\frac{L-1}{L}\widehat{\boldsymbol{x}} + \frac{1}{L}\boldsymbol{f}(\widehat{\boldsymbol{x}})\right) \tag{3.116}$$

$$= \arg\min_{\boldsymbol{x}}\left\{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda L}{2}\left\|\boldsymbol{x} - \frac{L-1}{L}\widehat{\boldsymbol{x}} - \frac{1}{L}\boldsymbol{f}(\widehat{\boldsymbol{x}})\right\|^2\right\}. \tag{3.117}$$

When $L = 1$, this simplifies down to

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}}\left\{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda}{2}\left\|\boldsymbol{x} - \boldsymbol{f}(\widehat{\boldsymbol{x}})\right\|^2\right\}. \tag{3.118}$$

Note that (3.118) is reminiscent of, although in general not equivalent to,

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}}\left\{\ell(\boldsymbol{x};\boldsymbol{y}) + \frac{\lambda}{2}\left\|\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x})\right\|^2\right\}, \tag{3.119}$$

which was discussed as an "alternative" formulation of RED in [5, Sec. 5.2].

Insights into the relationship between RED and PnP-ADMM can be obtained by focusing on the simple case of

$$\ell(\boldsymbol{x};\boldsymbol{y}) = \frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{y}\|^2, \tag{3.120}$$

where the overall goal of variational image recovery would be the denoising of $\boldsymbol{y}$. For PnP-ADMM, (3.90) and (3.98) imply

$$F_{\mathsf{pnp}}(\boldsymbol{v}) = \frac{1}{1+\lambda\sigma^2}\boldsymbol{y} + \frac{\lambda\sigma^2}{1+\lambda\sigma^2}\boldsymbol{v}, \tag{3.121}$$

and so the equilibrium condition (3.97a) implies

$$\widehat{\boldsymbol{x}}_{\mathsf{pnp}} = \frac{1}{1+\lambda\sigma^2}\boldsymbol{y} + \frac{\lambda\sigma^2}{1+\lambda\sigma^2}(\widehat{\boldsymbol{x}}_{\mathsf{pnp}} + \widehat{\boldsymbol{u}}_{\mathsf{pnp}}) \tag{3.122}$$

$$\Leftrightarrow \widehat{\boldsymbol{u}}_{\mathsf{pnp}} = \frac{\widehat{\boldsymbol{x}}_{\mathsf{pnp}} - \boldsymbol{y}}{\lambda\sigma^2}. \tag{3.123}$$

Meanwhile, (3.99) and the equilibrium condition (3.97b) imply

$$\widehat{\boldsymbol{x}}_{\mathsf{pnp}} = \boldsymbol{f}(\widehat{\boldsymbol{x}}_{\mathsf{pnp}} - \widehat{\boldsymbol{u}}_{\mathsf{pnp}}) \tag{3.124}$$

$$= \boldsymbol{f}\left(\frac{\lambda\sigma^2-1}{\lambda\sigma^2}\widehat{\boldsymbol{x}}_{\mathsf{pnp}} + \frac{1}{\lambda\sigma^2}\boldsymbol{y}\right). \tag{3.125}$$

In the case that $\lambda = 1/\sigma^2$, we have the intuitive result that

$$\widehat{\boldsymbol{x}}_{\mathsf{pnp}} = \boldsymbol{f}(\boldsymbol{y}), \tag{3.126}$$

which corresponds to direct denoising of $\boldsymbol{y}$. For RED, $\widehat{\boldsymbol{u}}_{\mathsf{red}}$ is algorithm dependent, but $\widehat{\boldsymbol{x}}_{\mathsf{red}}$ is always the solution to (3.15), where now $\boldsymbol{A} = \boldsymbol{I}$ due to (3.120). That is,

$$\boldsymbol{y} - \widehat{\boldsymbol{x}}_{\mathsf{red}} = \lambda\sigma^2\big(\widehat{\boldsymbol{x}}_{\mathsf{red}} - \boldsymbol{f}(\widehat{\boldsymbol{x}}_{\mathsf{red}})\big). \tag{3.127}$$

Taking $\lambda = 1/\sigma^2$ for direct comparison to (3.126), we find

$$\boldsymbol{y} - \widehat{\boldsymbol{x}}_{\mathsf{red}} = \widehat{\boldsymbol{x}}_{\mathsf{red}} - \boldsymbol{f}(\widehat{\boldsymbol{x}}_{\mathsf{red}}). \tag{3.128}$$

Thus, whereas PnP-ADMM reports the denoiser output $\boldsymbol{f}(\boldsymbol{y})$, *RED reports the* $\widehat{\boldsymbol{x}}$ *for which the denoiser residual* $\boldsymbol{f}(\widehat{\boldsymbol{x}}) - \widehat{\boldsymbol{x}}$ *negates the measurement residual* $\boldsymbol{y} - \widehat{\boldsymbol{x}}$. This $\widehat{\boldsymbol{x}}$ can be expressed concisely as

$$\widehat{\boldsymbol{x}} = (2\boldsymbol{I} - \boldsymbol{f})^{-1}(\boldsymbol{y}) = G_{\mathsf{red\text{-}pg}}(\boldsymbol{y})\big|_{L=1}. \tag{3.129}$$

3.7 Conclusion

The RED paper [5] proposed a powerful new way to exploit plug-in denoisers when solving imaging inverse problems. In fact, experiments in [5] suggest that the RED algorithms are state-of-the-art. Although [5] claimed that the RED algorithms minimize an optimization objective containing an explicit regularizer of the form $\rho_{\mathsf{red}}(\boldsymbol{x}) \triangleq \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{x}))$ when the denoiser is LH, we showed that the denoiser must also be Jacobian symmetric for this explanation to hold. We then provided extensive numerical evidence that practical denoisers like the median filter, non-local means, BM3D, TNRD, or DnCNN lack sufficient Jacobian symmetry. Furthermore, we established that, with non-JS denoisers, the RED algorithms cannot be explained by explicit regularization of any form.

None of our negative results dispute the fact that the RED algorithms work very well in practice. But they do motivate the need for a better understanding of RED. In response, we showed that the RED algorithms can be explained by a novel framework called *score-matching by denoising* (SMD), which aims to match the "score" (i.e., the gradient of the log-prior) rather than design any explicit regularizer. We then established tight connections between SMD, kernel density estimation, and constrained MMSE denoising.

On the algorithmic front, we provided new interpretations of the RED-ADMM and RED-FP algorithms proposed in [5], and we proposed novel RED algorithms with much faster convergence. Finally, we performed a consensus-equilibrium analysis of the RED algorithms that lead to additional interpretations of RED and its relation to PnP-ADMM.

Chapter 4: SEND: A Score Ensembling Strategy for Novelty Detection

4.1 Introduction

There are many applications where we want to be able to detect whether a signal was drawn from a known distribution or from a different, unknown distribution. This problem is called novelty detection, and a motivation is as follows. Advancements in deep learning have made the classification of high-dimensional signals, such as images [102], audio [103], and Radio-Frequency waveforms [104], a largely solved problem when the test sample is consistent with the training distribution. As an example, let us say we wanted to classify different types of tanks. We gather a large dataset of images of tanks and train a deep neural network (DNN) to perform classification. When our model is deployed, it does a great job of classifying tanks. But, as illustrated in Fig. 4.1, what if the system is asked to classify a non-tank vehicle, such as a school bus? A standard DNN classifier will only be able to return a type of tank, which is obviously incorrect. The ability to recognize the school bus as a novelty has obvious merit in this type of scenario. Novelty detection is particularly important in problems like our tank example, where the operational domain of the system is complex and dynamic.

In this dissertation, we will focus on a data-driven approach to novelty detection. We assume we have access to a training set $X = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i$ is the ith training signal and $y_i \in \{1 \dots K\}$ is the corresponding class label. We denote the number of known classes as $K \geq 1$. We will call a signal coming from the known class(es) an inlier and any other signal a novelty. Given the training set, our goal is to learn a classifier that can distinguish between an inlier and a novelty. This problem is different from standard binary classification because we do not have any examples of novelties in the training set.

In our formulation of the problem, we allow the number of known classes K to be any positive integer. If $K = 1$, this problem is called one-class classification [105]. In one-class classification, we are only given training examples from a single class and must use that data to learn a function to distinguish between signals from that class and any novelties. When $K > 1$, we call the problem multi-class novelty detection [106]. In this problem, the training data will be labeled and come from multiple classes. Open set classification [107] is a related problem where the goal is to perform multi-class novelty detection as well as classify inlier samples. Open set classification could be achieved by combining a multi-class novelty detection algorithm, like those discussed here, and a classification algorithm. One-class classification and multi-class classification are illustrated in Fig. 4.2. Our experiments will be performed on the multi-class novelty detection problem, but the developed method would also be compatible with the one-class problem.

Almost all novelty detection methods score test samples with an "inlier score." The higher this score, the more confident the model is that the test sample is an inlier. In order to perform novelty detection, they threshold this score. Any test sample with

Figure 4.1: Example images of tanks used to train a DNN tank classifier and a test sample, that we would not like classified as a type of tank. Images are from the CIFAR-100 dataset [6].

Figure 4.2: Example plots of (left) one-class classification and (right) multi-class novelty detection problems.

a score below the threshold is classified as a novelty, and a sample with a score higher than the threshold is classified as an inlier. In this work, we propose Shift-Ensembled Novelty Detection (SEND), a principled way to combine multiple scores. SEND is compatible with most inlier scores. In our experiments, we demonstrate SEND with several popular inlier score functions and show the ensembled score gives better performance than the individual scores themselves, in most cases. This technique allows us to leverage the large number of novelty detection techniques that have been developed by the community.

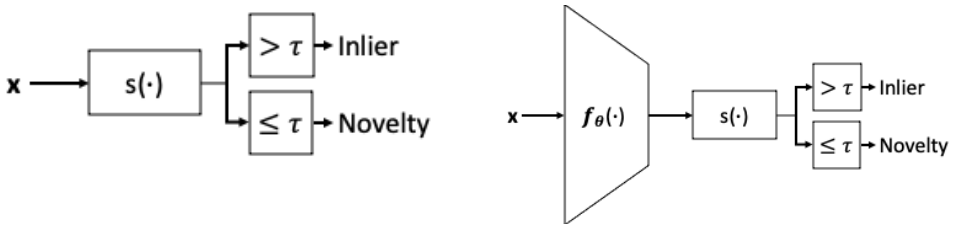


Figure 4.3: Diagrams of classic, low-dimensional (left) and deep, high-dimensional (right) novelty detection. Deep novelty detection utilizes a deep neural network to extract features before evaluating the score function $s(\cdot)$.

4.2 Background

4.2.1 Low-Dimensional Novelty Detection

When our signals $\boldsymbol{x}_i$ are low dimensional, novelty detection is essentially a solved problem [108]. Classical machine learning methods such as Mahalanobis distance [109], nearest neighbor distance [110], kernel density estimation [64], one-class support vector machine (OC-SVM) [111], and support vector data description [112] are techniques that perform well in low-dimensional settings.

These methods assume a notion of similarity between data points that is defined as some distance function, whether that is ℓ_2 distance, Mahalanobis distance, or some kernel function, such as the radial basis function. These assumptions work well in low dimensions but break down in high dimensions due to the curse of dimensionality.

4.2.2 High-Dimensional Novelty Detection

Novelty detection of high-dimensional signals is an active area of research. Most novelty detection solutions for high-dimensional signals utilize a DNN to encode the signal into a lower-dimensional space, after which low-dimensional novelty detection

approaches can be used [108]. In multi-class novelty detection, a common choice is to use the features output by the penultimate layer of a supervised DNN classifier. Novelty detection is then performed by thresholding a scoring function that takes these low-dimensional features as an input. We will call the feature encoder $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$, which is parameterized by DNN weights $\boldsymbol{\theta}$, and the score function $s(\cdot)$. Fig. 4.3 shows the difference between classic, low-dimensional novelty detection and deep, high-dimensional novelty detection.

Addressing the one-class classification problem, Sohn et al. [8] train a feature encoder using a contrastive loss [12]. These features are then scored using Mahalanobis distance or OC-SVM. They identify a problem with naïvely utilizing contrastively trained features for novelty detection. Contrastive training encourages the feature space to "fill-up" with inlier samples. For novelty detection, this is problematic because a lack of separation between novelties and inliers results in poor novelty detection performance. They propose to use smaller batch sizes and "shifting transforms," such as image rotations, during network training to fix this problem.

In parallel work [7], Tack et al. also train a feature encoder network with a contrastive loss and utilize shifting transforms. Their technique, called Contrastive Shifting Instances (CSI), uses a common backbone network and two additional "head" networks. One is the "contrastive head" and the other the "shift-classifier head." The contrastive head is trained using the contrastive loss, while the shift-classifier head learns to classify which shift was used on a given sample. At test time, CSI utilizes the contrastive features, as well as the activations of the shift-classifier head, to generate multiple inlier scores. These inlier scores are subsequently ensembled using a heuristic technique. CSI also introduces ensembling over the shifts of the test image,

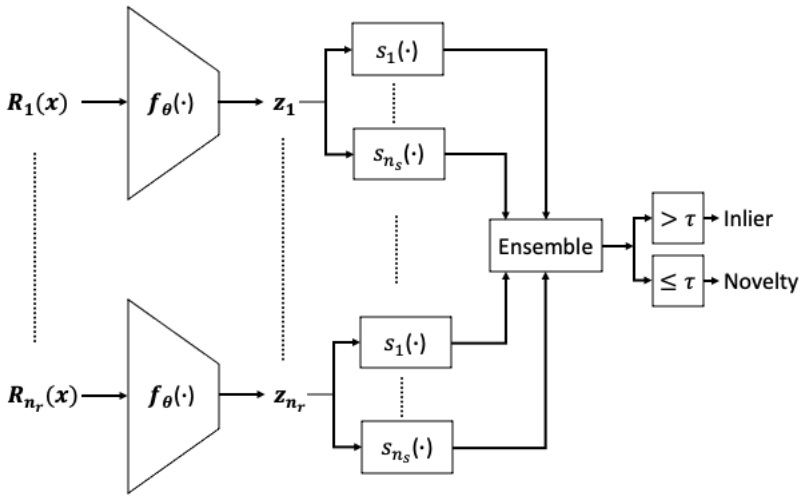


Figure 4.4: Diagram of CSI ensembling technique. Our test sample $\boldsymbol{x}$ is transformed by n_r shifting transforms, $\{\boldsymbol{R}_i(\cdot)\}_{i=1}^{n_r}$. These shifted versions of the test sample are input to the feature encoder network $\boldsymbol{f_\theta}(\cdot)$, which outputs corresponding feature vectors $\{\boldsymbol{z}_i\}_{i=1}^{n_r}$. Each feature vector is scored by n_s inlier scores, $\{s_i(\cdot)\}_{i=1}^{n_s}$. Finally, all the scores are ensembled to get one final score, which is thresholded to perform novelty detection.

$\{\boldsymbol{R}_j(\boldsymbol{x})\}_{j=1}^{n_r}$. For example, if using four image rotations as the shifting transform, CSI will evaluate each test image rotated by 0°, 90°, 180°, and 270°. The three scores from each of the four rotations are combined using a weighted sum. Fig. 4.4 illustrates the ensembling over multiple shifting transforms and inlier scores. The experiments in [7] show a significant benefit to ensembling inlier scores versus utilizing a single score. For our image experiments, we follow [7,8] and utilize image rotations of 0°, 90°, 180°, and 270°. We denote these rotations as $\boldsymbol{R}_1(\cdot) \ldots \boldsymbol{R}_4(\cdot)$, respectively.

4.3 Ensembling of Inlier Scores

We are motivated by the observation [7] that novelty detection performance can be improved by ensembling several scores. We will refer to the scores in the ensemble

as "base scores." Let us assume we have n_s different base scores $\{s_i(\cdot)\}_{i=1}^{n_s}$. Using the base scores, we can formulate a hypothesis test [113], where H_0 is the hypothesis that the test sample is an inlier and H_1 is the hypothesis that the test sample is a novelty.

One challenge when designing an ensembling technique for novelty detection is that different inlier base scores can have very different distributions. To illustrate this, we show histograms of the "norm" and "cos" scores from [7] evaluated on images from the CIFAR-10 dataset in the left plot of Fig. 4.5. These scores are described in more detail in Section 4.4.1 below. To calibrate the inlier distributions of the various base scores, we propose to use the quantile transform [114], which transforms the distribution of each score to be approximately Gaussian with zero mean and unit variance. The quantile transform for the ith base score is defined as

$$q_i(s) \triangleq \Phi^{-1}(\widehat{F}_i(s)) \tag{4.1}$$

$$\widehat{F}_i(s) \triangleq \frac{1}{n}\sum_{t=1}^{n} \mathbb{1}(s \leq s_i(\boldsymbol{x}_t)), \tag{4.2}$$

where $\Phi^{-1}(\cdot)$ is the inverse Gaussian CDF, $\widehat{F}_i(\cdot)$ is the empirical CDF of base score i evaluated on the training set, and $\mathbb{1}(\cdot)$ is the indicator function that has a value of 1 if the statement is true, and 0 otherwise. The center plot of Fig. 4.5 shows the same norm and cos distributions after processing by the quantile transform.

We model the quantile transformed base score using the random variable

$$r_i \triangleq q_i(s_i(\boldsymbol{x})). \tag{4.3}$$

This allows us to define conditional inlier and novelty distributions,

$$p(r_i|H_0) \triangleq p(q_i(s_i(\boldsymbol{x}))|\boldsymbol{x} \text{ is an inlier}) \tag{4.4}$$

$$p(r_i|H_1) \triangleq p(q_i(s_i(\boldsymbol{x}))|\boldsymbol{x} \text{ is a novelty}). \tag{4.5}$$

The quantile transform ensures that

$$p(r_i|H_0) \approx \mathcal{N}(r_i; 0, 1). \tag{4.6}$$

Defining $\boldsymbol{r} \triangleq [r_1, \ldots, r_{n_s}]^T$, we approximate the joint score distribution for the inliers as jointly Gaussian, i.e.,

$$p(\boldsymbol{r}|H_0) \approx \mathcal{N}(\boldsymbol{r}; \mathbf{0}, \boldsymbol{\Sigma}), \tag{4.7}$$

where $\boldsymbol{\Sigma}$ is the covariance of $\boldsymbol{r}$ under H_0. This is an approximation because the quantile transform only enforces that every entry in $\boldsymbol{r}$ is marginally Gaussian. However, we can see in the right plot of Fig. 4.5 that, for our working example, the transformed inliers have an approximately jointly Gaussian distribution. We handle the selection of the covariance matrix, $\boldsymbol{\Sigma}$, below.

4.3.1 Generalized Likelihood Ratio Test

If we had full knowledge of the distributions $p(\boldsymbol{r}|H_0)$ and $p(\boldsymbol{r}|H_1)$, we would be motivated to perform novelty detection using a likelihood ratio test (LRT) [113]

$$\mathrm{LR}(\boldsymbol{r}) \triangleq \frac{p(\boldsymbol{r}|H_0)}{p(\boldsymbol{r}|H_1)}, \quad \mathrm{LR}(\boldsymbol{r}) \underset{H_1}{\overset{H_0}{\gtrless}} \tau. \tag{4.8}$$

This is because the LRT is Neyman-Pearson optimal, meaning that the probability of detection is maximized for a fixed false-alarm rate [113]. Unfortunately, we have no data from which to estimate $p(\boldsymbol{r}|H_1)$, due to there being no novelties in the training set. Our approach will be to parameterize $p(\boldsymbol{r}|H_1)$ and formulate a composite hypothesis test [115].

We will assume that each base score has been designed such that the mean score under H_0 is no smaller than the mean score under H_1. Since the quantile transform

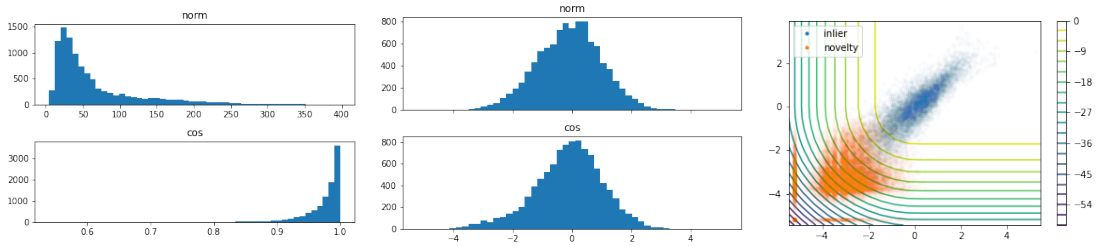


Figure 4.5: Left: Raw "norm" and "cos" scores from [7] evaluated CIFAR-10 training inliers. Center: Quantile transformed versions of the scores from the left plot. Right: Plot of "norm" score (x-axis) versus "cos" score (y-axis) for CIFAR-10 test inliers and SVHN test novelties. Contours show the value of END score from (4.33).

is monotonic, the quantile transformed scores, r_i, will also have this characteristic. In this case, the distributions $p(r_i|H_1)$ will have a non-positive mean.

Other than this non-positive mean property, we know very little about the novelty distribution $p(\boldsymbol{r}|H_1)$. For the sake of tractability, we will assume covariance homogeneity between novelties and inliers, which gives

$$p(\boldsymbol{r}|H_0) \approx \mathcal{N}(\boldsymbol{r}; \mathbf{0}, \boldsymbol{\Sigma}) \tag{4.9}$$

$$p(\boldsymbol{r}|H_1, \boldsymbol{\alpha}) \approx \mathcal{N}(\boldsymbol{r}; \boldsymbol{\alpha}, \boldsymbol{\Sigma}), \quad \alpha_i < 0 \; \forall i \in \{1 \dots n_s\} \tag{4.10}$$

where $\boldsymbol{\alpha}$ is unknown. We will discuss the selection of $\boldsymbol{\Sigma}$ below. Using (4.9) and (4.10) we find the $\log \mathrm{LR}$ as

$$\log \mathrm{LR} \triangleq -\tfrac{1}{2}\boldsymbol{r}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{r} + \tfrac{1}{2}(\boldsymbol{r}-\boldsymbol{\alpha})\boldsymbol{\Sigma}^{-1}(\boldsymbol{r}-\boldsymbol{\alpha}) \tag{4.11}$$

$$= -\boldsymbol{\alpha}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{r} + C, \tag{4.12}$$

where C is a constant that absorbs terms that do not depend on $\boldsymbol{r}$. A uniformly most powerful (UMP) test exists if the Neyman-Pearson test statistic is independent of the unknown [113]. Whenever $n_s > 1$, a UMP test does not exist because the test

statistic (4.12) changes with the direction of the unknown, $\boldsymbol{\alpha}$. When $n_s = 1$, the property $\alpha_1 < 0$ implies that a UMP exists and manifests as $r_1 \underset{H_1}{\overset{H_0}{\gtrless}} \tau$.

Because $\boldsymbol{\alpha}$ is non-positive but otherwise unknown, we can formulate a generalized likelihood ratio test (GLRT) [113] of the form

$$\mathrm{GLR}(\boldsymbol{r}) \triangleq \frac{p(\boldsymbol{r}|H_0)}{\max_{\boldsymbol{\alpha}:\alpha_i \leq 0} p(\boldsymbol{r}|H_1, \boldsymbol{\alpha})}, \quad \mathrm{GLR}(\boldsymbol{r}) \underset{H_1}{\overset{H_0}{\gtrless}} \tau. \tag{4.13}$$

Let us define the (non-positive) mean vector that maximizes the denominator of $\mathrm{GLR}(\boldsymbol{r})$ as

$$\boldsymbol{\alpha}^* \triangleq \arg\max_{\boldsymbol{\alpha}:\alpha_i \leq 0} \log p(\boldsymbol{r}|H_1, \boldsymbol{\alpha}), \tag{4.14}$$

$$= \arg\min_{\boldsymbol{\alpha}:\alpha_i \leq 0} (\boldsymbol{r} - \boldsymbol{\alpha})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{r} - \boldsymbol{\alpha}). \tag{4.15}$$

Because (4.15) is a quadratic program, we can solve for $\boldsymbol{\alpha}^*$ using a convex solver such as CVX [116]. We can then write the log GLR in terms of $\boldsymbol{r}$ and $\boldsymbol{\alpha}^*$ as

$$\log \mathrm{GLR}(\boldsymbol{r}) = \log p(\boldsymbol{r}|H_0) - \log p(\boldsymbol{r}|H_1, \boldsymbol{\alpha} = \boldsymbol{\alpha}^*) \tag{4.16}$$

$$= -\frac{1}{2} \boldsymbol{r}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{r} + \frac{1}{2} (\boldsymbol{r} - \boldsymbol{\alpha}^*)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{r} - \boldsymbol{\alpha}^*) \tag{4.17}$$

$$= \frac{1}{2} (\boldsymbol{\alpha}^*)^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\alpha}^* - \boldsymbol{r}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\alpha}^* \tag{4.18}$$

$$= (\tfrac{1}{2} \boldsymbol{\alpha}^* - \boldsymbol{r})^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\alpha}^*. \tag{4.19}$$

We then define the GLRT score as

$$s_{\mathrm{GLRT}}(\boldsymbol{r}) \triangleq (\tfrac{1}{2} \boldsymbol{\alpha}^* - \boldsymbol{r})^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\alpha}^*. \tag{4.20}$$

4.3.2 Covariance Matrix Selection

We now discuss the choice of the covariance matrix, $\boldsymbol{\Sigma}$, in the GLRT score (4.20). One choice would be to use the sample covariance from the (inlier) training samples,

$$\widehat{\boldsymbol{\Sigma}} \triangleq \frac{1}{n-1} \sum_{t=1}^{n} (\boldsymbol{r}^t - \widehat{\boldsymbol{\mu}})(\boldsymbol{r}^t - \widehat{\boldsymbol{\mu}})^T + \sigma \mathbf{I}, \tag{4.21}$$

where $\{\boldsymbol{r}^t\}_{t=1}^n$ are the quantile transformed base scores, evaluated on the training set, $\widehat{\boldsymbol{\mu}}$ is the sample mean of $\{\boldsymbol{r}^t\}_{t=1}^n$, and σ is a small constant to ensure $\widehat{\boldsymbol{\Sigma}}$ is full rank. For our experiments, we choose $\sigma = 10^{-6}$. In order to study this choice, we perform an eigenvalue decomposition of the sample covariance matrix, giving

$$\widehat{\boldsymbol{\Sigma}} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T, \tag{4.22}$$

where the columns of the orthonormal matrix $\boldsymbol{V}$ are the eigenvectors and $\boldsymbol{\Lambda}$ contains the non-negative eigenvalues along the diagonal and zeros elsewhere. With the eigenvalue-decomposition of (4.22), we can write the GLRT score as

$$\begin{aligned} s_{\text{GLRT}}(\boldsymbol{r}) &= (\tfrac{1}{2}\boldsymbol{\alpha}^* - \boldsymbol{r})^T \widehat{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\alpha}^* && (4.23)\\ &= (\tfrac{1}{2}\boldsymbol{\alpha}^* - \boldsymbol{r})^T \boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^T \boldsymbol{\alpha}^* && (4.24)\\ &= (\boldsymbol{V}^T(\tfrac{1}{2}\boldsymbol{\alpha}^* - \boldsymbol{r}))^T \boldsymbol{\Lambda}^{-1}\boldsymbol{V}^T \boldsymbol{\alpha}^* && (4.25)\\ &= \sum_{l=1}^{n_s} \frac{(\boldsymbol{v}_i^T(\frac{1}{2}\boldsymbol{\alpha}^* - \boldsymbol{r}) \cdot \boldsymbol{v}_i^T \boldsymbol{\alpha}^*)}{\lambda_i}. && (4.26) \end{aligned}$$

In this form, we see how the lth eigenvalue scales the $\boldsymbol{v}_l$ directional component of $s_{\text{GLRT}}(\cdot)$.

We want to see how score performance in these sub-spaces correlates with eigenvalue. We now define what we call the "eigen-scores."

$$s_{\text{eig}}^l(\boldsymbol{r}) \triangleq \boldsymbol{v}_l^T \boldsymbol{r}, \tag{4.27}$$

where $\boldsymbol{v}_l$ is the lth column of $\boldsymbol{V}$, whose corresponding eigen-value is $\lambda_l = [\boldsymbol{\Lambda}]_{l,l}$. For two example experiments, which will be discussed in more detail below, we plot the AUC performance of score s_{eig}^l versus eigenvalue λ_l, for $l \in \{1 \dots 24\}$ in Fig. 4.6. Due to the inherent sign ambiguity of the eigen-vectors, $\boldsymbol{v}_l$, we will take the negative of

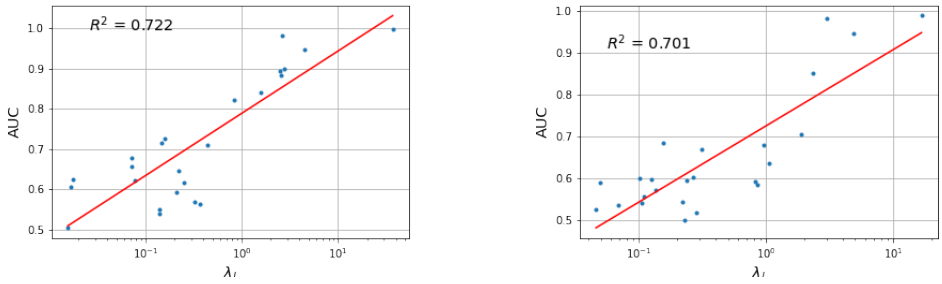


Figure 4.6: Plots of eigen-score performance (in AUC) vs. eigenvalue for two 24-score experiments. We can see that there is a strong correlation between how informative a direction is for novelty detection with its eigenvalue. Left plot has CIFAR-10 as the inliers and SVHN samples as the novelties. The right plot has SVHN as the inliers and CIFAR-10 as the novelties. A line is fit in the log space of the eigenvalues. The R^2 value of this regression is given on each plot.

the eigen-score if its AUC is less than one half. Both experiments show a strong correlation between AUC performance and eigenvalue size.

The existence of an eigen-score with a large eigenvalue implies that the base scores in $\boldsymbol{r}$ are strongly correlated in the $\boldsymbol{v}_l$ direction. The larger the eigenvalue the stronger the correlation. Fig. 4.6 then implies that, as this correlation increases, the novelty detection performance of that base score also tends to increase. In other words, base scores that are more correlated tend to perform better for novelty detection than those that are less correlated.

With this knowledge, selecting the sample covariance matrix for the GLRT score would devalue these directions that are most informative. Instead, we can get better performance by choosing $\boldsymbol{\Sigma} = \mathbf{I}$. This choice equally weights the contributions of each eigenvector direction.

A further benefit of this approach is that it drastically simplifies the calculation of $\boldsymbol{\alpha}^*$ from (4.15) to

$$[\boldsymbol{\alpha}^*]_i = r_i^- \triangleq \begin{cases} r_i & \text{if } r_i < 0 \\ 0 & \text{otherwise} \end{cases}. \tag{4.28}$$

This removes the need to perform a convex optimization to solve (4.15). Using $\boldsymbol{\Sigma} = \mathbf{I}$ and (4.28), the GLRT score (4.20) simplifies to

$$\begin{aligned} s_{\text{GLRT}}(\boldsymbol{r}) &= (\tfrac{1}{2}\boldsymbol{r}^- - \boldsymbol{r})^T \boldsymbol{r}^- && (4.29)\\ &= \sum_{i=1}^{n_s} (\tfrac{1}{2} r_i^- - r_i) r_i^- && (4.30)\\ &= \sum_{i=1}^{n_s} \tfrac{1}{2} (r_i^-)^2 && (4.31)\\ &= -\tfrac{1}{2} \|\boldsymbol{r}^-\|^2, && (4.32) \end{aligned}$$

where (4.31) results by observing that if r_i is positive, then r_i^- is zero, while if r_i is negative, then $r_i^- = r_i$. We call this final method Ensembled Novelty Detection (END).

$$s_{\text{END}}(\boldsymbol{r}) \triangleq -\tfrac{1}{2} \|\boldsymbol{r}^-\|^2. \tag{4.33}$$

In Section 4.5, we compare $\boldsymbol{\Sigma} = \mathbf{I}$ to $\boldsymbol{\Sigma} = \widehat{\boldsymbol{\Sigma}}$ in practical experiments to show the superiority of $\boldsymbol{\Sigma} = \mathbf{I}$.

The right plot of Fig. 4.5 shows quantile transformed "norm" and "cos" scores, $\boldsymbol{r}$, for test inliers and test novelties, together with contours of the END score. From this plot, we can see that the END score does a good job of separating the inliers from the novelties. It is clear from the figure that thresholding the END score provides better classification of the inliers and novelties than thresholding either base score individually.

4.3.3 END Analysis: White Gaussian Case

In this section, we will study the performance of END in a simple Gaussian setting:

$$p(\boldsymbol{r}|H_0) = \mathcal{N}(\boldsymbol{r}; \mathbf{0}, \mathbf{I}) \tag{4.34}$$

$$p(\boldsymbol{r}|H_1, \boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{r}; \boldsymbol{\alpha}, \mathbf{I}). \tag{4.35}$$

We draw 100 000 samples from each distribution and repeat the experiment for a dense grid of $\boldsymbol{\alpha} = [\alpha_1, \alpha_2]^T$. We will compare END to a single score, defined as

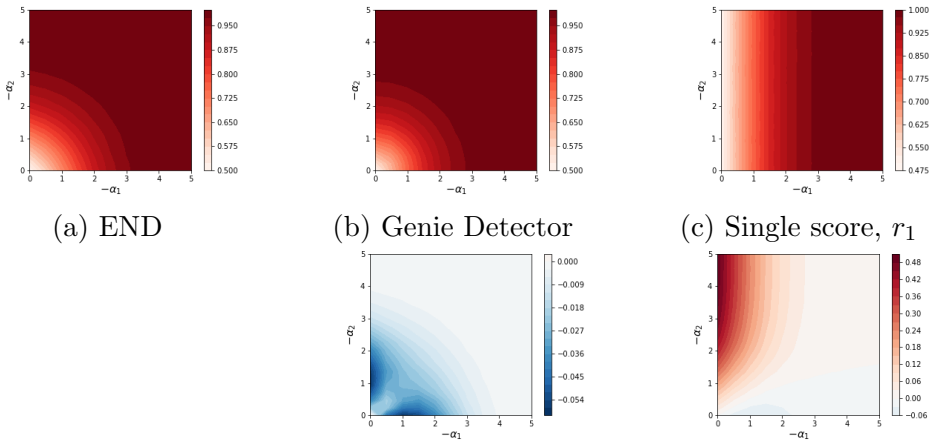
$$s_1(\boldsymbol{r}) = r_1, \tag{4.36}$$

and a genie detector that knows $\boldsymbol{\alpha}$ and then uses the Neyman-Pearson optimal log LR from (4.12)

$$s_{\text{gen}}(\boldsymbol{r}) \triangleq -\boldsymbol{\alpha}^T \boldsymbol{r}. \tag{4.37}$$

In Fig. 4.7, we plot the AUC versus α_1 and α_2 for each detector. We also plot the difference between END and the genie detector, as well as the difference between END and the single-score detector. Not surprisingly, the plots show that the genie detector performs the best because it has additional knowledge (of $\boldsymbol{\alpha}$) is it uses optimally. However, Fig. 4.7d shows that there is only a small decrease in performance for END when compared with the genie detector.

Fig. 4.7e shows single-score vs END-ensembled AUC performance. The first take-away from this figure is that, for most values of $\boldsymbol{\alpha}$, the AUC performance of END is much better than that of the single-score detector. However, along the $\alpha_2 = 0$ axis we see a light blue area where the single score, r_1, performs slightly better than the END ensemble. This makes sense because, along this line, the r_2 score is uninformative,



(a) END

(b) Genie Detector

(c) Single score, r_1

(d) Difference between END and genie detector performance

(e) Difference between END and single score performance

Figure 4.7: Sub-plots (a)-(c) show the AUC versus the novelty mean components α_1 and α_2. Sub-plots (d)-(e) show the AUC of END minus the AUC of the genie and single score detectors, respectively.

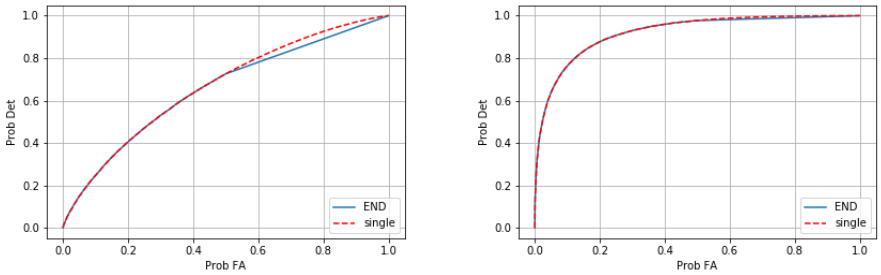
but END will still try to utilize it. We also note that, along the $\alpha_2 = 0$ line, the single score is equivalent to the optimal genie score. This plot suggests that the advantages of ensembling via END are far greater than the disadvantages.

4.3.4 END Analysis: Redundant Scores

Next, we discuss the performance of END with redundant base scores. First, we study the simple case where we have two identical base scores, $r_1 = r_2 = r$. In this case, the single-score detector $s(\boldsymbol{r}) = r_1$ is Neyman-Pearson optimal. Meanwhile, END simplifies to

$$s_{\text{END}}(\boldsymbol{r}) = -\tfrac{1}{2}\|\boldsymbol{r}^-\|^2 \tag{4.38}$$

$$= -(r^-)^2. \tag{4.39}$$

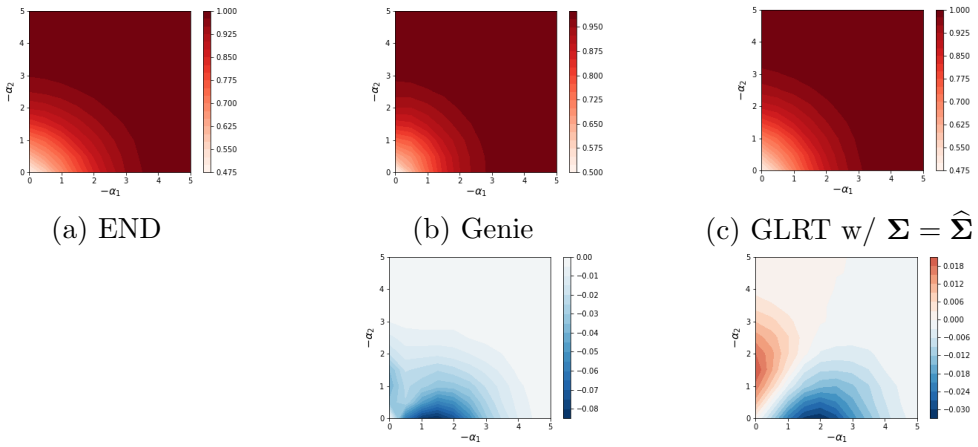


(a) $\alpha = -0.6$ (b) $\alpha = -2.0$

Figure 4.8: Performance of single score plotted against the END ensemble of redundant scores when (left) $\alpha = -0.6$ and (right) $\alpha = -2.0$.

Because $r^- \leq 0$ from (4.28), thresholding $s_{\text{END}}(\boldsymbol{r}_)$ from (4.39) is equivalent to thresholding r^-. For negative thresholds τ, thresholding r^- is equivalent to thresholding the original base score r. But positive thresholds τ cannot be applied to r^-, nor $s_{\text{END}}(\boldsymbol{r})$ from (4.39). Therefore, the ROC curve for $s_{\text{END}}(\boldsymbol{r})$ with redundant base scores will be equivalent to that of the single score detector up to a false alarm rate of 0.5, which is achieved by a threshold of $\tau = 0$. Due to positive thresholds not being available to END, false alarm rates greater than one half can be obtained only by forming a randomized rule [113] that mixes the detectors that correspond to false alarm rates of 0.5 and 1. This can be seen from the ROC curve in Fig. 4.8. In most practical cases, we are interested in probability of false alarm < 0.5, and in those cases the redundant END score is equivalent to the single score, which is Neyman-Pearson optimal.

In a second study of redundant scores, we consider three base scores, where two of the three are identical. We draw samples of $[r_1, r_2]$ from distributions (4.34) and (4.35) and set $r_3 = r_2$. This time, we construct a genie score that knows to ignore r_3 and also knows $[\alpha_1, \alpha_2]$, which it uses to formulate the Neyman-Pearson optimal



(a) END

(b) Genie

(c) GLRT w/ $\mathbf{\Sigma} = \widehat{\mathbf{\Sigma}}$

(d) Difference between END performance and genie detector performance

(e) Difference between performance of END and GLRT with $\mathbf{\Sigma} = \widehat{\mathbf{\Sigma}}$

Figure 4.9: Results from the redundant score experiment with three base scores. Sub-plots (a)-(c) show the AUC versus the novelty mean components α_1 and α_2. Sub-plots (d)-(e) show the AUC of END minus the AUC of the genie and the GLRT score with $\mathbf{\Sigma} = \widehat{\mathbf{\Sigma}}$, respectively.

score in (4.37) with $\boldsymbol{r} = [r_1, r_2]^T$. We repeat the experiments from Fig. 4.7, but now with END from (4.33), the genie score, and $s_{\text{GLRT}}(\boldsymbol{r})$ from (4.20) with $\mathbf{\Sigma} = \widehat{\mathbf{\Sigma}}$. Recall that END is the GLRT score from (4.20) with $\mathbf{\Sigma} = \mathbf{I}$.

In Fig. 4.9, when the non-redundant base score, r_1, is more informative than the redundant score, r_2, the GLRT score with $\mathbf{\Sigma} = \widehat{\mathbf{\Sigma}}$ performs better than END. However, when the redundant base score, r_2, is significantly more informative than r_1, which occurs in the top left of the plots, END performs better. The eigen-score experiments in Fig. 4.6 suggest that, in practice, redundant scores tend to perform much better than non-redundant scores. This suggests that the top left region in Fig. 4.9 is the region of practical interest.

4.4 Shift Ensembled Novelty Detection

In this section, we once again take inspiration from CSI [7]. As seen in Fig. 4.4 above, CSI utilizes several base scores along with "shifting" transforms, such as image rotations, to make a larger ensemble of scores. In the case of image rotations, CSI evaluates each base score on the test image rotated by 0°, 90°, 180°, and 270°, giving an ensemble with four times as many scores.

We thus propose to combine our score ensembling strategy, END, with shifting transforms, like those used in CSI, to obtain an enhanced novelty detection framework, which we call Shift-Ensembled Novelty Detection (SEND). Suppose we have n_s base scores, indexed by i, and n_r shifting transforms, indexed by j, giving a total ensemble of $n_s n_r$ scores. For SEND, we define a version of the quantile transform indexed by i and j,

$$q_{ij}(s) \triangleq \Phi^{-1}(\widehat{F}_{ij}(s)) \tag{4.40}$$

$$\widehat{F}_{ij}(s) \triangleq \frac{1}{n}\sum_{t=1}^{n} \mathbb{1}(s \leq s_i(\boldsymbol{R}_j(\boldsymbol{x}_t))) \tag{4.41}$$

where $\boldsymbol{R}_j(\cdot)$ is the jth shifting transform and $\widehat{F}_{ij}$ is the sample CDF for score function i and shift j on the inlier training data $\{x_t\}_{t=1}^n$. Next, we combine the END score and the shifting transform to define the SEND score,

$$s_{\text{SEND}}(\boldsymbol{x}) \triangleq -\tfrac{1}{2}\|\boldsymbol{r}^-(\boldsymbol{x})\|^2, \tag{4.42}$$

with

$$\boldsymbol{r}(\boldsymbol{x}) = \begin{bmatrix} q_{1,1}(s_1(R_1(\boldsymbol{x}))) \\ \vdots \\ q_{1,n_r}(s_1(R_{n_r}(\boldsymbol{x}))) \\ q_{2,1}(s_2(R_1(\boldsymbol{x}))) \\ \vdots \\ q_{n_s,n_r}(s_{n_s}(R_{n_r}(\boldsymbol{x}))) \end{bmatrix}, \tag{4.43}$$

and $\boldsymbol{r}^-(\boldsymbol{x})$ defined as the negative component of $\boldsymbol{r}(\boldsymbol{x})$, as in (4.28). We report several experiments with SEND in the section below.

4.4.1 Inlier Scores

For our experiments, we choose several scores to ensemble with END and SEND. These scores will be utilized as the base scores for our ensembling strategies. Our first two experiments will utilize inlier scores that can be computed with the CSI [7] model. CSI uses three scores, the "norm" score, "cos" score, and shift-classifier score. The norm score is given by the norm of the features output by the SimCLR head of the CSI network,

$$s_{\text{norm}}(\boldsymbol{x}) \triangleq \|\boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{x})\|. \tag{4.44}$$

Next, the "cos" score is given by the nearest-neighbor cosine distance of the SimCLR features to the training data,

$$s_{\cos}(\boldsymbol{x}, j) \triangleq \min_{\boldsymbol{x}' \in X} \mathsf{sim}(\boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{R}_j(\boldsymbol{x}'))), \tag{4.45}$$

where $\mathsf{sim}(\cdot, \cdot)$ is the cosine similarity, given by $\mathsf{sim}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|}$, and j is the shift index used to evaluate the training samples. The third score used in CSI is the shift-classifier score, given by

$$s_{\text{shift}}(\boldsymbol{x}, j) \triangleq \ell_{\boldsymbol{\theta}}(\boldsymbol{x})_j, \tag{4.46}$$

where $\ell_{\boldsymbol{\theta}}(\boldsymbol{x})_j$ denotes the jth logit of the shift-classifier head of the CSI model. The shift index, j, is an argument of s_{shift} because we intend to use the jth logit when we apply the jth shifting transform $\boldsymbol{R}_j(\cdot)$ in SEND. The CSI score ensembles the norm,

cos, and shift-classifier scores as follows:

$$s_{\text{con}}(\boldsymbol{x}, j) \triangleq s_{\text{norm}}(\boldsymbol{x}) s_{\cos}(\boldsymbol{x}, j) \tag{4.47}$$

$$s_{\text{CSI}}(\boldsymbol{x}) \triangleq \sum_{j=1}^{n_r} \left[\lambda_j^{\text{con}} s_{\text{con}}(\boldsymbol{R}_j(\boldsymbol{x}), j) + \lambda_j^{\text{shift}} s_{\text{shift}}(\boldsymbol{R}_j(\boldsymbol{x}), j) \right] \tag{4.48}$$

where λ_js are weights chosen to balance the scores,

$$\lambda_j^{\text{con}} = \frac{n}{\sum_{i=1}^{n} \| \boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{R}_j(\boldsymbol{x}_i)) \|} \tag{4.49}$$

$$\lambda_j^{\text{shift}} = \frac{n}{\sum_{i=1}^{n} \ell_{\boldsymbol{\theta}}(\boldsymbol{R}_j(\boldsymbol{x}_i))_j}. \tag{4.50}$$

We note that the CSI ensembling strategy does not extend to other scores.

As an additional score, we propose to utilize a class dependent OC-SVM [111] trained on the backbone features of the CSI network. The backbone features are the common layer before the contrastive and shift-classifier head networks. First, we train an OC-SVM to perform one-class classification of each class of the inlier dataset. At evaluation time, we score each test sample using the minimum distance to the separating hyperplanes of the various OC-SVMs. We use the scikit-learn [114] implementation of OC-SVM using the default model parameters.

For our large-ensemble experiments, we utilize additional inlier scores that are now described. The first of these extra scores utilize the Sup-CSI network from [7]. The final layer of the Sup-CSI network is a linear classifier that attempts to jointly predict the class label and shift index of the test sample. The logits of this network will be denoted as $\boldsymbol{\ell}_{SupCSI}(\boldsymbol{x}) \in \mathbb{R}^{K \times n_s}$. The (k, j)th index of these logits can be interpreted as the confidence that the sample came from class k and shift j. We define the Sup-CSI softmax-thresholding score as

$$s_{\text{supCSI-soft}}(\boldsymbol{x}, j) = \max_{k \in \{1 \ldots K\}} \text{softmax}(\boldsymbol{\ell}_{SupCSI}(\boldsymbol{x})_j)_k, \tag{4.51}$$

where $\boldsymbol{\ell}_{SupCSI}(\boldsymbol{x})_j \in \mathbb{R}^K$ is the column of logits associated with shift j and the softmax function is given by

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}. \tag{4.52}$$

We can interpret this score as thresholding the confidence in the most likely class, under the assumption that shift j was used. This score is an adapted version of the original Sup-CSI score, from [7], that is compatible with SEND.

We also propose another score that utilizes a classification network trained to jointly predict the class label, k, and shift label, j. In particular, we define a joint Mahalanobis distance score,

$$s_{\text{joint-Mah}}(\boldsymbol{x}) = -\min_{(k,j)} (\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widehat{\boldsymbol{\mu}}_{kj}) \widehat{\boldsymbol{\Sigma}}_{kj}^{-1} (\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widehat{\boldsymbol{\mu}}_{kj}), \tag{4.53}$$

where $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ is the penultimate layer of the class-shift classification network, and $\widehat{\boldsymbol{\mu}}_{kj}$ and $\widehat{\boldsymbol{\Sigma}}_{kj}$ are the sample mean and sample covariance of joint label (k, j). This score can be interpreted as a joint class-shift version of the class-dependent Mahalanobis distance detector proposed in [117].

4.4.2 Ensembles

Now we define three combinations of these base scores, which we call END-4, SEND-16, and SEND-24. The number will designate the number of base scores being ensembled by the method. END-4 uses the norm, cos, shift-classifier, and OC-SVM base scores using the END ensembling technique. This experiment does not consider shifts of these base scores. For SEND-16, we will utilize the same $n_s = 4$ base scores used in the END-4 experiment, but we also consider $n_r = 4$ shifts of each, using the image-rotation shifting-transform. This gives us a total ensemble size of $n_s n_r = 16$.

Finally, our SEND-24 ensemble uses the norm, cos, shift, and OC-SVM base scores from END-4 and SEND-16 plus the joint Mahalanobis base score (4.53) and the Sup-CSI softmax-thresholding base score (4.51), giving us $n_s = 6$. Like SEND-16, SEND-24 also utilizes the image rotation shifting transform, for $n_r = 4$. This gives us a total ensemble size of $n_s n_r = 24$.

4.5 Experiments

4.5.1 Setup

Our experiments will use standard image datasets like CIFAR-10, CIFAR-100 [6], SVHN [118], ImageNet [119], and LSUN [120]. We use the versions of ImageNet and LSUN that have been downsampled to 32×32 by the authors of [121]. For each experiment, we will treat one dataset as the inliers. We train the feature encoder networks, fit base score parameters (such as the $\boldsymbol{\mu}_{j,k}$s and $\widehat{\boldsymbol{\Sigma}}_{j,k}$s for $s_{\text{joint-Mah}}(\cdot)$), and fit quantile transforms with the training partition of the inlier dataset. Then, we evaluate inlier scores with the test partition of the inlier dataset and the test partition of another, novelty dataset. Performance is evaluated by the Area Under the receiver operating Curve (AUC).

4.5.2 Model Training

We now describe the training of the CSI, Sup-CSI, standard classification network, and joint class-shift classification network. All models will utilize the ResNet-18 [102] architecture as the feature encoder. CSI and Sup-CSI are trained using the code from the associated paper [7] following the parameter settings from the paper. The standard classification network is trained using cross-entropy loss with inlier class labels, and without shifting transforms. The joint classifier network is trained using

Inlier Dataset	CIFAR-10				SVHN				
Novelty Dataset	SVHN	LSUN	ImageNet	CIFAR-100	LSUN	ImageNet	CIFAR-10	CIFAR-100	Avg.
Norm	0.9769	0.9247	0.9017	0.8339	0.5263	0.6064	0.6036	0.6467	0.7525
Cos	0.9750	0.9690	0.9571	**0.8592**	0.7781	0.7819	0.8120	0.7911	0.8654
Shift-Classifier	0.9791	0.9789	**0.9843**	0.8480	0.9422	0.9485	**0.9815**	**0.9729**	**0.9544**
OC-SVM	0.9013	0.8552	0.8581	0.6642	**0.9629**	**0.9579**	0.9807	0.9706	0.8939
END-4	**0.9892**	**0.9817**	0.9838	0.8479	0.9441	0.9493	0.9748	0.9643	**0.9544**

Table 4.1: Results from the simple ensembling experiments.

cross-entropy loss with label smoothing [122] and joint class-shift labels. We train two versions of each model, one using CIFAR-10 and one using SVHN. We train the networks for 100 epochs using the LARS [123] optimizer. The learning rate is scheduled using cosine annealing [124] with initial learning rate of 0.1, a max learning rate of 1.0, one epoch of warmup, and a final learning rate of 10^{-6}.

For all experiments, the batch size was set to $128/n_r$, which makes the batch augmented with the shifting transforms have a size of 128. We note that the training time scales with n_r, in that training a model with $n_r = 4$ takes four times as long as training a model with $n_r = 1$.

4.5.3 Simple Ensemble Experiments

Our first experiments compare the END-4 ensemble to individual use of the norm, cos, shift-classifier, and OC-SVM base scores. These experiments do not utilize a shifting transform ($n_r = 1$). Results can be found in Table 4.1. On average, END and the shift-classifier tied for highest AUC. For certain experiments, like CIFAR-10/SVHN, END-4 performed better than all the individual base scores. However, there are a few experiments where individual base scores perform much better than END-4, such as SVHN/LSUN. The gap between the best base score and the END-4 performance seems to get larger when there is a poorly performing base score in

Inlier Dataset	CIFAR-10				SVHN				
Novelty Dataset	SVHN	LSUN	ImageNet	CIFAR-100	LSUN	ImageNet	CIFAR-10	CIFAR-100	Avg.
CSI [7]	**0.9950**	0.9750	0.9783	**0.8628**	0.9571	0.9631	0.9638	0.9556	0.9563
GLRT ($\boldsymbol{\Sigma} = \widehat{\boldsymbol{\Sigma}}$)	0.9712	0.9608	0.9683	0.8239	**0.9922**	**0.9917**	0.9692	0.9618	0.9549
SEND-16	**0.9950**	**0.9768**	**0.9787**	0.8584	0.9888	0.9890	**0.9801**	**0.9726**	**0.9674**

Table 4.2: Results from the shift ensembling experiments.

the ensemble. This occurs in the SVHN/LSUN experiment, where the norm score performed very poorly at an AUC of 0.5263, which is just better than guessing (i.e., AUC=0.5). We also saw an example of this behavior in our two-score synthetic experiments in Fig. 4.7. As we will see below, our ensembling methods work better as we increase the number of scores.

4.5.4 Shift Ensemble Experiments

Our next experiments will utilize our SEND ensembling technique with the image rotation shifting transforms. We will compare the SEND-16 ensemble, CSI [7], and the GLRT ensembling method (4.20) with $\boldsymbol{\Sigma} = \widehat{\boldsymbol{\Sigma}}$. The GLRT ensemble will use the same base scores as SEND-16 (norm, cos, shift-classifier, and OC-SVM). Results from these experiments can be found in Table 4.2. From the table, we can see that SEND-16 gave the best performance on average. Also, SEND-16 matched or outperformed CSI in all but one of the experiments. The SEND ensemble performed better than the GLRT ensemble in all but two experiments, which further justifies our choice to use $\boldsymbol{\Sigma} = \mathbf{I}$ rather than $\boldsymbol{\Sigma} = \widehat{\boldsymbol{\Sigma}}$. These results suggest that, when we utilize shifting transforms to create a larger ensemble, SEND can leverage correlation among scores, as discussed in Section 4.3.2.

Inlier Dataset	CIFAR-10				SVHN				
Novelty Dataset	SVHN	LSUN	ImageNet	CIFAR-100	LSUN	ImageNet	CIFAR-10	CIFAR-100	Avg.
Mah [117]	0.9041	0.7175	0.6209	0.7077	0.9069	0.9122	0.7809	0.7692	0.7899
OC-SVM [125]	0.9813	0.7581	0.6998	0.6891	0.9929	0.9921	0.9862	0.9774	0.8846
CSI [7]	0.9950	0.9750	0.9783	0.8628	0.9571	0.9631	0.9638	0.9556	0.9563
Sup-CSI [7]	0.9844	0.9787	0.9788	**0.9191**	0.9882	0.9904	0.9909	**0.9860**	0.9771
SEND-24	**0.9968**	**0.9834**	**0.9841**	0.9044	**0.9945**	**0.9952**	**0.9926**	0.9838	**0.9793**

Table 4.3: Results from the large ensembling experiments.

4.5.5 Large Ensemble Experiments

Finally, we perform these experiments with the SEND-24, large ensemble. We will compare our method with CSI [7], Sup-CSI [7], OC-SVM [125], and Mahalanobis [117]. We note that the Mahalanobis and OC-SVM methods in this experiment utilize a standard classification network trained using cross-entropy, as in the original work. The results of these experiments can be seen in Table 4.3. The table shows that SEND-24 performed best on average. Furthermore, it outperformed all baseline methods in six out of the eight experiments and performed second best in the remaining two experiments. The only method to outperform SEND-24 on any experiment was Sup-CSI [7], which performed better on the CIFAR-10/CIFAR-100 experiment and the SVHN/CIFAR-100 experiment. Overall, these last experiments show state-of-the-art novelty detection performance for the proposed SEND-24 method.

4.6 Summary

In this work, we developed SEND, a score ensembling technique for novelty detection. SEND has two primary motivations: i) the large number of inlier scores that have been developed by the research community, and ii) the performance gains achievable by expanding the set of base scores using shifting transforms [7]. We set

out to develop a principled method for ensembling scores for novelty detection using the framework of hypothesis testing [113]. We then performed synthetic and practical experiments to show the potential benefits and limitations of SEND. Our practical experiments show that SEND achieved state-of-the-art novelty detection performance with common image datasets. We note that SEND could be easily applied to other data types and utilize other inlier scores. In the next chapter, for example, we apply SEND to novelty detection of radar and communication waveforms.

Chapter 5: Novelty Detection for RF Waveforms with Ensembled Contrastive Learning

5.1 Introduction

Radio-Frequency (RF) waveform identification is a difficult engineering problem with many open research areas standing between the current state-of-the-art and the deployment of real-world systems [126]. These open areas of research include detecting dynamic signals, detecting signals in crowded spectrum, and detecting signal classes that were not included in the training set. In this work, we focus on the last case, known as novelty detection.

If we have a simple mathematical model for the approved modulation types, it may be possible to design a closed-form detector [113]. However, these techniques are not able to capture the complexity of modern radar and communication systems. For this reason, there has been significant research into utilizing data-driven approaches for waveform identification, such as deep neural networks (DNNs) [104,127].

One possible application of novelty detection in RF waveforms would be to monitor spectrum for non-approved modulation types [128]. In this application, we would have a list of approved types of modulations with any other type being illegal. It would be near-impossible to train a binary DNN classifier for this problem because

we would need to construct a dataset with every non-approved type of modulation, which would be prohibitively large and ever changing as new modulation types are developed. In this example, we want to be able to train a DNN-based detector to flag illegal modulation types, but with a training set made-up exclusively of approved waveforms.

Following this motivation, assume we have a dataset of approved waveforms $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ are waveform samples, $y_i \in \{1 \dots K\}$ the corresponding label, and K is the number of known classes. If $K > 1$, our problem is an example of multi-class novelty detection [106], and if $K = 1$, we call the problem one-class-classification [105].

Recently, Tack et al. [7] introduced Contrastive Shifting Instances (CSI). CSI is a novelty detection technique, developed for image datasets, that utilizes a feature encoder trained on the self-supervised contrastive loss. CSI also utilizes "shifting transforms," which are transforms designed to semantically change a sample, which they use to augment training and to ensemble their technique over multiple transformations of the test sample. CSI will be discussed in more detail in the background section below, but for now, it suffices to say that, first, CSI as originally proposed is not compatible with RF waveform data and, second, CSI uses multiple scores but combines them in a heuristic manner.

In this chapter, we develop a technique for RF waveform novelty detection. First, we adapt the CSI [7] framework to work with RF waveforms. An intermediate result of this effort is self-supervised learning for RF waveforms. We then apply the SEND framework, developed in chapter 4, to RF waveforms. In our experiments, we show that SEND significantly outperforms other state-of-the-art techniques.

5.2 Background

5.2.1 Low-Dimensional Novelty Detection

Low-dimension novelty detection is mostly a solved problem. There are many popular methods such as Mahalanobis distance [109], nearest neighbor distance [110], kernel density estimation [64], one-class support vector machine (OC-SVM) [111], and support vector data description [112]. Later, we will utilize the Mahalanobis distance, defined as

$$s_{\text{Mah}}(\boldsymbol{x}) = (\boldsymbol{x} - \widehat{\boldsymbol{\mu}})^T \widehat{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{x} - \widehat{\boldsymbol{\mu}}), \tag{5.1}$$

where $\widehat{\boldsymbol{\mu}}$ is the sample mean and $\widehat{\boldsymbol{\Sigma}}$ is the sample covariance matrix of the training set X. Mahalanobis distance can be interpreted as the negative log-likelihood of a Gaussian distribution fit to the training data. We will also use the nearest-neighbor distance

$$s_{\text{NN}}(\boldsymbol{x}) = \min_{\boldsymbol{x}' \in X} d(\boldsymbol{x}, \boldsymbol{x}'), \tag{5.2}$$

where $d(\cdot, \cdot)$ is an arbitrary distance function such as ℓ_2 distance or cosine distance.

These low-dimensional methods assume a notion of similarity based on a distance function, such as the ℓ_2-norm. In high-dimensional signals, such as radar waveforms, this notion of similarity does not hold.

5.2.2 High-Dimensional Novelty Detection

In high-dimensional novelty detection it is common to use a DNN to encode the signal into a lower-dimensional space, where a classic novelty detection technique, like those discussed above, can be used [108]. We will call this feature encoder $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$, parameterized by weights $\boldsymbol{\theta}$, and the score function $s(\cdot)$.

When class-labels are available, a common solution is to utilize a DNN classification network as the feature extractor $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$. A popular inlier score is the maximum of the softmax outputs of that DNN classifier. This technique is called softmax-thresholding [129]. It is popular due to its simplicity, but often performs poorly due to DNN classifiers being overconfident on samples far from the training set [130]. More sophisticated novelty detection techniques utilize the penultimate features of a classification DNN as $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$. Andrew et al. [125] introduce a technique where a OC-SVM is trained on the penultimate features of a DNN classifier to provide the inlier score. A similar technique is applied to radar waveform data in [131]. In Lee et al. [117], the penultimate layer of a DNN classifier is used as the features and scored by the class-dependent Mahalanobis distance

$$s_{\text{Mah}}(\boldsymbol{x}) = -\min_{j=\{1\ldots K\}} (\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widehat{\boldsymbol{\mu}}_j)^T \widehat{\boldsymbol{\Sigma}}_j^{-1} (\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widehat{\boldsymbol{\mu}}_j), \tag{5.3}$$

where $\widehat{\boldsymbol{\mu}}_j$ is the sample mean of $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ for class j and $\widehat{\boldsymbol{\Sigma}}_j$ is the sample covariance matrix for class j, both evaluated on the training set X.

Label smoothing [122] is a technique used to improve calibration and generalization of DNN classifiers. It is shown in [132] that label smoothing results in penultimate features being more compact. More recently, label smoothing has been shown to assist with novelty detection performance [133]. During training, the target label distribution is smoothed by taking a weighted average of the true label, y, and a uniform distribution

$$\boldsymbol{p}^{LS}(y) = (1-\alpha)\boldsymbol{e}_y + \frac{\alpha}{K}, \tag{5.4}$$

where $\boldsymbol{e}_i$ is the unit vector with a 1 in the ith element and all other elements being zero. The label smoothing loss is the cross-entropy between this target distribution

and the distribution defined by the output of the DNN classifier

$$\mathcal{L}_{LS}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} -p^{LS}(y_i)_k \log(\widehat{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)_k). \tag{5.5}$$

Note that if $\alpha = 0$, this loss simplifies to the standard cross-entropy loss.

These classifier-based methods for novelty detection are popular, but there are several downsides to using features from a classification network for novelty detection. The first is that we need class-labels to train the classifier. This means this technique cannot be used for one-class classification problems. The second downside is that a DNN classifier will learn features that are good for classification, but may learn to ignore features important to novelty detection [13]. Thus a better option for learning features is self-supervised learning.

5.2.3 Self-Supervised Learning

Self-supervised learning [134] is where DNN feature encoders are trained without using class-labels. The first way to do this is to train the DNN to perform an auxiliary task on the data. One example is to train the DNN to classify the rotation of images that have been rotated 0, 90, 180, or 270 degrees [135]. Since natural images usually have some notion of "up," it is possible to train a DNN to classify these rotations. In learning the ability to classify rotations, the DNN learns to encode semantically important features.

Another approach to self-supervised learning is contrastive learning [136]. In contrastive learning, DNNs are trained so that the encoding of semantically similar images are "close" in some metric, like cosine or ℓ_2 distance. This technique requires us to have some notion of similarity. For supervised problems, a notion of similarity could be if two samples come from the same class [137]. However, in self-supervised

learning we do not have this notion of similarity built into the data. A recent implementation of contrastive learning, Simple framework for Contrastive Learning of visual Representations (SimCLR) [12], solves this problem.

In SimCLR, the authors take any batch of samples as dissimilar to one another and use random augmentations of the same sample to create pairs of similar samples. We will refer to these random augmentations as "similarity" transforms because they produce samples that are semantically similar to the input. We will denote similarity transforms with $\boldsymbol{T}_j(\cdot)$, where the index j captures different possible random augmentations. When working with images, the similarity transforms would be some combination of random crops, gray-scaling, color jitter, etc. We will develop similarity transformations for use with RF waveform data below. SimCLR utilizes the normalized temperature cross-entropy (NT-Xent) loss [138]. Given a sample $\boldsymbol{x}$, a sample $\boldsymbol{x}_+$ that is "similar" to $\boldsymbol{x}$, and a batch of images X_- dissimilar to $\boldsymbol{x}$, the NT-Xent loss is given by

$$\begin{aligned}&\mathcal{L}_{NT-Xent}(\boldsymbol{x}, \boldsymbol{x}_+, X_-) \\ &= -\log \frac{\exp(\text{sim}(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_+))/\tau)}{\sum_{\boldsymbol{x}_- \in X_-} \exp(\text{sim}(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_-)/\tau) + \exp(\text{sim}(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_+))/\tau)},\end{aligned} \tag{5.6}$$

where $\text{sim}(\cdot, \cdot)$ is some similarity metric, such as cosine similarity, and $\tau > 0$ is a tunable normalizing temperature. This loss rewards features that encode similar samples to be close in terms of $\text{sim}(\cdot, \cdot)$, while punishing dissimilar samples from being close in terms of $\text{sim}(\cdot, \cdot)$.

The next contribution of SimCLR is the addition of a "head" network that takes the output of the encoder $\boldsymbol{f}_{\boldsymbol{\theta}}$ as its input. The head network is made up of two fully-connected layers with a ReLU activation in between. We will denote the SimCLR head evaluated on $\boldsymbol{x}$ as $\boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{x})$. SimCLR shows that by encouraging "like" images

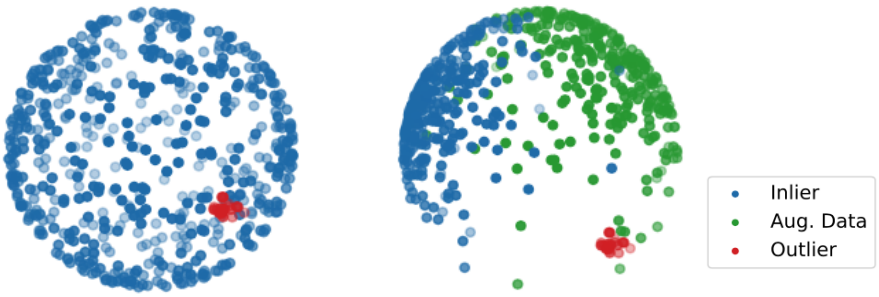


Figure 5.1: Distribution of features where encoders are trained with (left) normal SimCLR and (right) SimCLR augmented with shifting transforms. We can see that by utilizing shifting transforms, novelty detection becomes easier. Figure from [8]

.

to be close in cosine similarity, given by $\mathsf{sim}(\boldsymbol{z}, \boldsymbol{z}') = \frac{\boldsymbol{z}^T \boldsymbol{z}'}{\|\boldsymbol{z}\|\|\boldsymbol{z}'\|}$, the encoding of $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ becomes better for classification.

For a batch $B = \{\boldsymbol{x}_i\}_{i=1}^{n_b}$, SimCLR augments the batch with two transforms, $\widetilde{\boldsymbol{x}}_{i,1} = \boldsymbol{T}_1(\boldsymbol{x}_i)$ and $\widetilde{\boldsymbol{x}}_{i,2} = \boldsymbol{T}_2(\boldsymbol{x}_i)$ for all $i \in \{1 \dots n_b\}$. We will denote the augmented batch as $\widetilde{B} = \{\widetilde{\boldsymbol{x}}_{i,1}\}_{i=1}^{n_b} \cup \{\widetilde{\boldsymbol{x}}_{i,2}\}_{i=1}^{n_b}$. The SimCLR loss can be written as

$$\mathcal{L}_{SimCLR}(B) = \frac{1}{2n_b} \sum_{i=1}^{n_b} \sum_{j=1}^{2} -\log \frac{\exp(\mathsf{sim}(\boldsymbol{z}_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_{i,j}), \boldsymbol{z}_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_{i,3-j}))/\tau)}{\sum_{(i',j') \neq (i,j)} \exp(\mathrm{sim}(\boldsymbol{z}_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_{i,j}), \boldsymbol{z}_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_{i',j'})/\tau)}. \quad (5.7)$$

For our experiments, we set $\tau = 0.5$. Each sample is randomly transformed twice and its matching pair is used as the similar sample for the contrastive loss, while all other samples in the batch are the negative, or unlike, samples.

5.2.4 Self-Supervised Learning for Novelty Detection

One problem with using SimCLR features for novelty detection is that the loss will try to spread feature encodings of the inliers uniformly throughout the feature space. This problem is illustrated in the left plot of Fig. 5.1. Sohn et al. [8] discuss that this problem can be alleviated by reducing the batch size and utilizing "shifting"

augmentation transforms. They augment the training with these shifting augmentations and use them as dissimilar samples during SimCLR training. This has the effect of not filling the space with only inliers. The effects of this technique can be seen in right plot of Fig. 5.1. A shifting transform should significantly change the data so it is semantically different from the unshifted version. This is in contrast to the similarity transforms that attempt to not change a sample semantically. For example, [8] use 90, 180, and 270 degree rotations for their experiments using images. A 90 degree rotation of a natural image will be semantically different from an image that is right side up.

In parallel work, Tack et al. [7] introduce Contrastive Shifting Instances (CSI), which is another self-supervised novelty detection technique that utilizes shifting transforms. We define a set of shifting transforms $\mathcal{R} = \{\boldsymbol{R}_j\}_{j=1}^{n_r}$ and a shifted version of batch B as $B_{\boldsymbol{R}} = \{\boldsymbol{R}(\boldsymbol{x}_t)\}_{t=1}^{n_b}$. Now we can define the contrastive-CSI loss,

$$\mathcal{L}_{con-CSI} = \mathcal{L}_{SimCLR}\left(\bigcup_{j=1}^{n_r} B_{\boldsymbol{R}_j}\right). \tag{5.8}$$

Samples evaluated on different shifting transforms are treated as negative, or unlike, samples within the contrastive loss. To use images as an example, a 90 degree rotated image of a dog would be a negative sample to the upright version of that same dog. The shifting transformations also allow us to evaluate a test point multiple times. We can augment a test point $\boldsymbol{x}$ with $\mathcal{R}$, giving us $\{\boldsymbol{R}_1(\boldsymbol{x}), \boldsymbol{R}_2(\boldsymbol{x}), \ldots \boldsymbol{R}_{n_r}(\boldsymbol{x})\}$. This augmentation will allow us to evaluate a single novelty score n_r times.

The second contribution of CSI is the addition of a shift-classification head. CSI also uses an additional network head that acts on the features $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$. This layer has an output dimension equal to the number of shifts n_r. We will denote the logits of

this layer as $\boldsymbol{\ell}_{\boldsymbol{\theta}}(\cdot)$. This shift-classification head is trained with cross-entropy loss

$$\mathcal{L}_{con-shi} = \frac{1}{2n_b n_r} \sum_{j=1}^{n_r} \sum_{t=1}^{n_b} \sum_{l=1}^{2} -\log \text{softmax}(\boldsymbol{\ell}_{\boldsymbol{\theta}}(\boldsymbol{R}_j(\widetilde{\boldsymbol{x}}_{t,l})))_j \tag{5.9}$$

$$\text{softmax}(\boldsymbol{\ell})_j = \frac{\exp(\ell_j)}{\sum_{i=1}^{n_r} \exp(\ell_i)}. \tag{5.10}$$

These losses are combined to make the CSI loss

$$\mathcal{L}_{CSI} = \mathcal{L}_{con-CSI} + \lambda_{shi} \mathcal{L}_{con-shi}, \tag{5.11}$$

where $\lambda_{shi} > 0$ trades off between the two losses. In [7], the authors found that setting $\lambda_{shi} = 1$ worked well. We will also use this setting in our experiments.

Finally, CSI introduces several inlier scores that work well with SimCLR/CSI trained encoders. They develop a feature norm score,

$$s_{\text{norm}}(\boldsymbol{x}) = \|\boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{x})\|. \tag{5.12}$$

This score is based on the observation that features of inlier samples have a larger norm than features of novelties.

They also develop a nearest neighbor cosine similarity score. This score utilizes the features from the training set as a dictionary. The nearest neighbor cosine similarity score can be written as

$$s_{\cos}(\boldsymbol{x}, j) = \max_{\boldsymbol{x}' \in X} \mathsf{sim}(\boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{R}_j(\boldsymbol{x}'))), \tag{5.13}$$

where X is the training set. SimCLR and CSI are training the feature encodings of semantically similar samples to be close in terms of cosine similarity. Therefore, we assume inliers, which should be semantically similar to something in the training set, should have an encoding that is close, in terms of cosine similarity, to a sample from the training set.

In CSI, these scores are combined in a product to form what they call the contrastive score.

$$s_{\text{con}}(\boldsymbol{x}, j) = s_{\text{norm}}(\boldsymbol{x}) s_{\cos}(\boldsymbol{x}, j). \tag{5.14}$$

This product is one way the CSI authors combine scores, but there is no justification for it, other than empirical success. CSI also utilizes the shift classifier head logits $\boldsymbol{\ell}_{\boldsymbol{\theta}}(\cdot)$ with the shift score

$$s_{\text{shift}}(\boldsymbol{x}, j) = \ell_{\boldsymbol{\theta}}(\boldsymbol{x})_j \tag{5.15}$$

where $\ell_{\boldsymbol{\theta}}(\cdot)_j$ is the logit corresponding to the jth shift. Inliers are more likely to have the shift correctly identified by the shift-classifier than novelties.

CSI combines s_{con} and s_{shift} by performing a weighted sum over the n_r shifts

$$s_{\text{CSI}}(\boldsymbol{x}) = \sum_{j=1}^{n_r} \left[\lambda_j^{\text{con}} s_{\text{con}}(\boldsymbol{R}_j(\boldsymbol{x}), j) + \lambda_j^{\text{shift}} s_{\text{shift}}(\boldsymbol{R}_j(\boldsymbol{x}), j) \right] \tag{5.16}$$

where λ_js are weights chosen to balance the scores,

$$\lambda_j^{\text{con}} = \frac{n}{\sum_{i=1}^{n} \left\| \boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{R}_j(\boldsymbol{x}_i)) \right\|} \tag{5.17}$$

$$\lambda_j^{\text{shift}} = \frac{n}{\sum_{i=1}^{n} \ell_{\boldsymbol{\theta}}(\boldsymbol{R}_j(\boldsymbol{x}_i))_j}. \tag{5.18}$$

Again, the authors of CSI give no justification for using this weighted sum to combine scores other than empirical success. We are motivated by the empirical success of CSI's ensembling to develop a technique that is more grounded in detection theory. CSI is summarized in Fig. 5.2. In our work, we use the CSI loss to train our one-class models and we use the CSI score as a state-of-the-art ensembling benchmark.

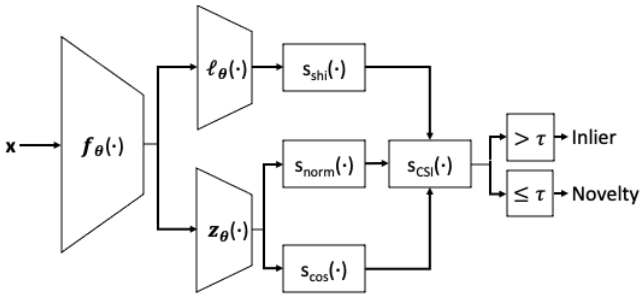


Figure 5.2: CSI uses two head networks, a SimCLR head, $\boldsymbol{z_\theta}(\cdot)$, and a shift-classifier head, $\boldsymbol{\ell_\theta}(\cdot)$, which both utilize a common feature encoder network, $\boldsymbol{f_\theta}(\cdot)$. These networks are jointly trained to minimize $\mathcal{L}_{CSI}$. The final CSI score performs ensembling over the shift score, norm score, and cos score.

5.3 Contrastive Learning for RF Waveform Data

5.3.1 Similarity Transforms

Based on the success of self-supervised novelty detection techniques on images, we want to develop self-supervised learning for RF waveform data. In this section, we will discuss our efforts to adapt SimCLR to work with radar and communication waveform data. First, we must develop new similarity transforms (denoted as $\boldsymbol{T}(\cdot)$ above). The goal of these transformations is to retain the semantic information of the signal, such as modulation type, while inserting some of the variability present in actual measurements. For image data, SimCLR utilizes random crops and color variation that caputres the inherent spatialvariability from camera pointing and variability in lighting. We want our SimCLR model to take two transformations of the same sample and output two encodings that are close with respect to cosine similarity. In order to design similarity transforms for RF waveforms, we started with the SIDLE radar pulse measurement model from [139]. These waveforms are modeled as a pulse train

given by

$$y(t) = \sum_{p=1}^{P} A_p x(t/\alpha - t_p) \exp(j\omega_0 t) + w(t) \tag{5.19}$$

where $x(t)$ is the complex-valued, base radar pulse, $j = \sqrt{-1}$, A_p is the pulse amplitude, α is a time scaling parameter, t_p is a pulse-dependent time-shift, ω_0 is the carrier frequency, and $w(t)$ is additive noise. These signals are sampled at a constant sampling rate well above the Nyquist rate. With this knowledge of the measurement model, we can design several similarity transformations. We will try using the following transformations: adding noise, random crop (in time), and random phase rotation.

The noise transformation is simply adding Gaussian noise to the original, noiseless waveform $\boldsymbol{x}$,

$$\boldsymbol{T}_{noise}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{w}, \tag{5.20}$$

where $\boldsymbol{w}$ is a realization of Gaussian random noise. We note that the training data is noiseless. The SNR for every sample in the augmented batch is drawn independently from a continuous uniform random distribution. The bounds of this distribution are -12 to 12 dB for the radar waveforms and -20 to 20 dB for the communication waveforms. By using a range of noise levels during training, the feature encoder learns to be agnostic to noise level.

The random crop transformation is a uniformly random crop that is 100 samples smaller than the initial window. Radar and communication signals from an unsynchronized transmitter will occur at unknown timing. Therefore, the original sample from the training set and a time-shifted version of that sample are just as likely to be seen by our receiver.

The random phase angle rotation transform is given by

$$\boldsymbol{T}_{phase}(\boldsymbol{x}) = \boldsymbol{x}\exp(j\phi), \tag{5.21}$$

where ϕ is the phase angle uniformly drawn between 0 and 2π. This transform takes advantage of the phase mismatch between the transmitter and receiver. If $\boldsymbol{x}$ is a valid waveform, then $\boldsymbol{T}_{phase}(\boldsymbol{x})$ is valid for any $\phi \in [0, 2\pi)$.

In our experiments, we will use a composition of these transforms. When a batch is drawn during training, each transformation is performed on every sample with an independently drawn transform realization. For example, if we are using the noise and phase transformations, every sample will get an independent random draw of the phase rotation and the noise realization.

To evaluate these self-supervised training methods, we train a model with the SimCLR loss (5.7) using different combinations of the above transformations. We then train a linear classifier on the encoding $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ and report the error rate of this classifier on a hold out test set. Experimental settings, such as learning rate schedule and epochs, are the same as our novelty detection experiments, which are described in the experiments section below.

Results from the SimCLR experiments can be seen in Table 5.1. For the SIDLE data, noise + phase and noise + phase + crop are tied for best performance. For the GNURadio data, noise + crop is the best. We notice is that the phase + crop combination does the worst for both SIDLE and GNURadio data. This shows that adding noise is a powerful similarity transform when training a SimCLR method on RF waveform data. We also notice that the random phase rotation boosts performance on the SIDLE data, and the addition of the random crop boosts performance

Table 5.1: Performance of Linear classifier trained on SimCLR features for SIDLE and GNURadio PSK signals.

	Error Rate (%)	
Transform	SIDLE	GNURadio
Phase + Crop	23.722	37.550
Noise	0.033	32.033
Noise + Crop	0.033	**31.317**
Noise + Phase	**0.022**	32.283
Noise + Phase + Crop	**0.022**	33.100

on the GNURadio data. Therefore we use all three similarity transformations (noise + phase + crop) when training SimCLR and CSI models for the rest of this chapter.

5.3.2 Shifting Transforms

We now consider shifting transforms for RF waveforms. As discussed above, the shifting transform's goal is to semantically change the signal. We develop a transform based on (5.19), intuition about radar and communication signals, and observations of the types of transformations that worked well for images in CSI.

In CSI, a 0, 90, 180, and 270 degree rotation operator was shown to work well for images. For 1D signals, an analogous operation would be to "time-reverse" the signal. For time-reverse, we have the number of transforms $n_r = 2$, where $\boldsymbol{R}_1(\cdot)$ returns the original signal and $\boldsymbol{R}_2(\cdot)$ returns the time-reversed signal. We note that in (5.19), our signals are not at baseband, but at a carrier frequency ω_0. If the base pulse signal $x(t)$ were symmetric in time and processing were done at baseband, time-reverse would not be effective as a shifting transform, because it would not semantically change the signal. Our experiments, in the section below, show that utilizing the

time-reverse shifting transform significantly boosts the performance of the novelty detection method.

5.4 SEND for RF Waveforms

We are motivated by CSI's observation that novelty detection performance can be improved by ensembling several scores. As discussed above, CSI uses both a weighted sum and a product in order to combine inlier scores. The CSI ensembling method works well for some datasets (e.g., their results are state-of-the-art for CIFAR-10), but we find that their strategy does not perform well in general. Motivated by the partial success of CSI's ensembling technique we set out to develop an improved ensembling strategy.

We now apply the SEND ensembling technique, developed in Section 4.3 and 4.4 of the previous chapter, to RF waveforms. Recall, for a collection of score functions, $\{s_i(\cdot)\}_{i=1}^{n_s}$ and shifting transforms, $\{\boldsymbol{R}_j(\cdot)\}_{j=1}^{n_r}$ the SEND score is given by

$$s_{\text{SEND}}(\boldsymbol{x}) = \tfrac{1}{2}\|\boldsymbol{r}^-(\boldsymbol{x})\|^2 \tag{5.22}$$

$$\boldsymbol{r}(\boldsymbol{x}) = \begin{bmatrix} q_{1,1}(s_1(\boldsymbol{R}_1(\boldsymbol{x}))) \\ \vdots \\ q_{1,n_r}(s_1(\boldsymbol{R}_{n_r}(\boldsymbol{x}))) \\ q_{2,1}(s_2(\boldsymbol{R}_1(\boldsymbol{x}))) \\ \vdots \\ q_{n_s,n_r}(s_{n_s}(\boldsymbol{R}_{n_r}(\boldsymbol{x}))) \end{bmatrix}, \tag{5.23}$$

where $q_{i,j}(\cdot)$ is the quantile transform for the ith score and the jth shifting transform and $r^-(\boldsymbol{x})$ is the negative component of $\boldsymbol{r}(\boldsymbol{x})$, defined by

$$r_i^-(\boldsymbol{x}) = \begin{cases} r_i(\boldsymbol{x}) & r_i(\boldsymbol{x}) < 0 \\ 0 & r_i(\boldsymbol{x}) \geq 0 \end{cases}. \tag{5.24}$$

Next, we will discuss possible combinations of shifting transforms and score functions to be used with SEND.

1. We first introduce our one-class novelty detection strategy, One-Class SEND (OC-SEND). For this approach, we will use a DNN trained using the CSI loss given by (5.11). We will ensemble $s_{\text{norm}}(\cdot)$, $s_{\cos}(\cdot)$, and $s_{\text{shift}}(\cdot)$ with $s_{\text{SEND}}(\cdot)$. The norm and cos scores will utilize features from the SimCLR head ($\boldsymbol{z}_{\boldsymbol{\theta}}(\cdot)$) and the shift score will utilize the shift-classifier head ($\boldsymbol{\ell}_{\boldsymbol{\theta}}(\cdot)$). The shifting transform used for both training and testing will be time-reverse. This gives us $n_s = 3$ and $n_r = 2$ for a total ensemble of six scores. This implementation of SEND does not require access to the class-labels, so it is compatible with the one-class classification problem.

2. Next, we introduce our Multi-Class SEND (MC-SEND) technique. For MC-SEND we will train a classifier to learn the joint class-transform label (y, j), where y is the class label and j is the shift transform index. If we have n_r shifts, we turn a K-class problem into a $n_r K$-class problem. This means the final layer of the neural network is changed to have an output of dimension $n_r K$. This classifier will be trained using the label smoothing loss (5.5), with $\alpha = 0.1$. With this trained classifier we can evaluate the joint class-dependent Mahalanobis distance

$$s_{\text{joint-Mah}}(\boldsymbol{x}) = -\min_{(k,j)}(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widehat{\boldsymbol{\mu}}_{kj})\widehat{\boldsymbol{\Sigma}}_{kj}^{-1}(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widehat{\boldsymbol{\mu}}_{kj}), \tag{5.25}$$

where $\widehat{\boldsymbol{\mu}}_{kj}$ and $\widehat{\boldsymbol{\Sigma}}_{kj}$ are the sample mean and sample covariance of joint class (k, j).

For the MC-SEND ensemble we will utilize both this joint-label classification network and the self-supervised network used in OC-SEND. We ensemble $s_{\text{joint-Mah}}(\cdot)$, evaluated on the penultimate features of the joint-label classifier, $s_{\text{norm}}(\cdot)$ and $s_{\cos}(\cdot)$ evaluated on the features of the SimCLR head ($\boldsymbol{z}_{\boldsymbol{\theta}}(\cdot)$), and $s_{\text{shift}}(\cdot)$ evaluated on the

shift-classifier head ($\boldsymbol{\ell_\theta}(\cdot)$). Our experiments in the next section show that the addition of the joint-Mahalanobis score increases performance over the base OC-SEND technique.

5.5 Experiments

5.5.1 Datasets

For our experiments, we use the SIDLE radar waveform dataset used in [131] and a communication dataset constructed using GNURadio [140], which is an altered version of the RadioML10a dataset [127]. The SIDLE dataset contains radar waveforms of various types. The waveforms are either padded or cropped to a length of 5000 samples. This padding or cropping is done randomly, so that the location of the waveform may not be centered in the window. Noise is added at training time so that we can get different noise realizations of the same signal. SNR levels are chosen uniformly random between -12 and 12 dB. We used classes 1-10 (excluding class 6) of the SIDLE dataset as inlier classes, while classes 11-18 were treated as the novelty classes.

Our GNURadio communication dataset simulates various communication modulation types. We use similar parameters to the RadioML10a dataset, except we choose to make our signals length 1 024, in order to use similar techniques to SIDLE. We use the GNURadio dynamic channel model, which gives us a fading channel with random timing, frequency, and phase offsets. Noise is added at training time with SNRs randomly chosen from between -20 and 20 dB. We use the PSK modulations (BPSK, QPSK, 8PSK) as inliers and the analog modulations (AM-DSB, AM-SSB,

WBFM) as novelties. For both datasets, train/test splits are selected by uniformly sampling 10% of each class for the test set.

5.5.2 Evaluation

We will use the area under the receiver operating curve (AUC) and detection probability (DPR) at a fixed false alarm rate to evaluate the proposed techniques. We use AUC because it is a threshold independent metric of novelty detection performance. We also report DPR at a fixed false alarm rate because it is more tangible than AUC and gives a specific point on the receiver operating curve. DPR will be evaluated at 1% false alarm rate on the SIDLE dataset and 5% on the GNURadio dataset. We report results for a higher false alarm rate for GNURadio because that detection problem is harder than the SIDLE radar waveforms problem.

5.5.3 Model Training

For the neural network architecture, we use an adapted version of ResNet-18 [102]. We use the adaptation technique specified in [104]. Waveforms are complex valued, so we use complex multiplications in the convolutional layers. Batch norm is computed separately for the real and imaginary components. The convolutional kernel size of layer 1 is set to 11, while all others are set to 9. We also convert 2D operations (for images) to 1D operations (for time series).

The self-supervised network is trained to minimize $\mathcal{L}_{SimCLR}$ or $\mathcal{L}_{CSI}$. The joint-label classifier network is trained on the label smoothing loss with joint class-shift labels. The training samples for the joint-label classifier network are augmented with the same similarity transforms as the self-supervised network. We train the networks

for 100 epochs on the SIDLE data and 300 epochs on the GNURadio data. Optimization is performed by the LARS [123] optimizer. The learning rate is scheduled using cosine annealing [124] with initial learning rate of 0.1, a max learning rate of 1.0, 1 epoch of warmup, and an ending learning rate of 10^{-6}.

For all experiments, the batch size was set to $128/n_r$, which makes the augmented batch have a size of 128. We note that the training time scales with n_r. Training a model with $n_r = 2$ takes twice as long as training a model with $n_r = 1$.

For our experiments, we split the training set into two subsets. The first is the "dictionary," which is used for the cosine nearest neighbor score and Mahalanobis distance, and the second will be used for the sample CDF (4.41) for the quantile transform. We use 10% of the training set for the dictionary set and the remaining 90% for fitting the quantile transform. We will implement the quantile transforms with the python sklearn [114] package, which approximates the sample CDF (4.41) with 10 000 bins.

5.5.4 Baseline Methods

We will compare our methods to multiple baseline novelty detection methods. For the one-class classification problem we will compare our methods to the CSI [7] score. We note that this score utilizes the same DNN model and base inlier scores (norm, cos, and shift) as OC-SEND, but combines the scores differently. We will also compare to the contrastive score evaluated on a SimCLR model.

We also use two baseline multi-class novelty detection methods that utilize the class-labels during training. Both methods use the penultimate layer of a DNN classifier, trained using cross-entropy, as the feature encoder. The first utilizes the class

Table 5.2: Novelty detection performance for various techniques on the SIDLE and GNURadio waveform datasets. Area under the receiver operating curve is labeled as AUC and detection probability is labeled DPR. DPR is evaluated at 1% false alarm rate for the SIDLE data and 5% false alarm rate for the GNURadio data.

Dataset	SIDLE		GNURadio	
Inlier Score	AUC	DPR	AUC	DPR
Con [7]	0.9563	92.51	0.2026	1.43
CSI [7]	0.9769	76.76	0.4243	1.03
OC-SEND (ours)	**0.9997**	**99.88**	**0.8456**	**56.70**
Mah. [117]	0.9944	89.33	0.7812	23.43
OC-SVM [125,131]	0.9749	22.50	0.7676	39.45
MC-SEND (ours)	**0.9998**	**99.99**	**0.8700**	**62.62**

dependent Mahalanobis distance evaluated on the penultimate layer of the DNN classifier [117]. The second uses a class-dependent OC-SVM, trained on the penultimate features of the DNN classifier [125]. We note that this is similar to the method used in [131].

5.5.5 Novelty Detector Performance

We now evaluate our developed novelty detection techniques. We provide novelty detection performance for various feature encoder/score function pairs on the SIDLE and GNURadio datasets in Table 5.2. The table lists the training loss used to train the model(s) and the inlier score function used. The tables are broken into two sections. The top section includes methods that are one-class classifiers, which means they do not use any class-labels. These methods could be used when class-labels are unavailable or for datasets with a single class. The bottom section displays methods that utilize the class-labels during training. These methods are limited to solving the multi-class novelty detection problem.

Our experiments show that our SEND framework significantly benefits novelty detection performance. For the one-class classification methods, OC-SEND is the best on both datasets in both DPR and AUC. For the multi-class novelty detection methods, the Joint-LS / MC-SEND pair performs best. These results show the strong benefit of utilizing the SEND methods to ensemble multiple inlier scores. We also see that MC-SEND performs better than OC-SEND. This result is not surprising since MC-SEND is able to use the joint classifier features as well as the self-supervised CSI features. These results also show that label smoothing boosts performance of MC-SEND. In every metric the Joint-LS version of MC-SEND beats the Joint-CE version.

5.6 Summary

In this work, we demonstrated SimCLR and CSI for waveform data and showed that several versions of SEND perform well on novelty detection for SIDLE radar waveform data and GNURadio communication waveforms. The experiments demonstrate that our ensembling method increases performance over base score functions as well as over other novelty detection ensembling strategies.

Bibliography

[1] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Trans. Comp. Imag.*, vol. 5, pp. 52–67, Mar. 2019.

[2] R. Ahmad, C. A. Bouman, G. T. Buzzard, S. Chan, S. Liu, E. T. Reehorst, and P. Schniter, "Plug and play methods for magnetic resonance imaging," *IEEE Signal Process. Mag.*, vol. 37, no. 1, pp. 105–116, 2020.

[3] S. Liu, E. Reehorst, P. Schniter, and R. Ahmad, "Free-breathing cardiovascular MRI using a Plug-and-Play method with learned denoiser," in *Proc. IEEE Int. Symp. Biomed. Imag.*, Apr. 2020.

[4] M. Wharton, E. T. Reehorst, and P. Schniter, "Compressive SAR image recovery and classification via CNNs," in *Proc. Asilomar Conf. Signals Syst. Comput.*, pp. 1081–1085, 2019.

[5] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, 2017.

[6] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.

[7] J. Tack, S. Mo, J. Jeong, and J. Shin, "CSI: Novelty detection via contrastive learning on distributionally shifted instances," in *Proc. Neural Inform. Process. Syst. Conf.*, vol. 33, pp. 11839–11852, 2020.

[8] K. Sohn, C.-L. Li, J. Yoon, M. Jin, and T. Pfister, "Learning and evaluating representations for deep one-class classification," in *Proc. Internat. Conf. on Learning Repres.*, 2021.

[9] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, vol. 2, pp. 60–65, 2005.

[10] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, 2017.

[11] A. Hyvärinen, "Estimation of non-normalized statistical models by score matching," *J. Mach. Learn. Res.*, vol. 6, pp. 695–709, 2005.

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learning*, pp. 1597–1607, 2020.

[13] J. Winkens, R. Bunel, A. G. Roy, R. Stanforth, V. Natarajan, J. R. Ledsam, P. MacWilliams, P. Kohli, A. Karthikesalingam, S. Kohl, T. Cemgil, S. M. A. Eslami, and O. Ronneberger, "Contrastive training for improved out-of-distribution detection," *arXiv:2007.05566v1*, July 2020.

[14] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Proc. IEEE Global Conf. Signal Info. Process.*, pp. 945–948, 2013.

[15] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magnetic Resonance Med.*, vol. 58, no. 6, pp. 1182–1195, 2007.

[16] J. Fessler, "Optimization methods for MR image reconstruction (long version)," *arXiv:1903.03510*, 2019.

[17] C. M. Hyun, H. P. Kim, S. M. Lee, S. Lee, and J. K. Seo, "Deep learning for undersampled MRI reconstruction," *Physics in Medicine & Biology*, vol. 63, no. 13, p. 135007, 2018.

[18] H. K. Aggarwal, M. P. Mani, and M. Jacob, "Model based image reconstruction using deep learned priors (MODL)," in *Proc. IEEE Int. Symp. Biomed. Imag.*, pp. 671–674, 2018.

[19] A. Hauptmann, S. Arridge, F. Lucka, V. Muthurangu, and J. A. Steeden, "Real-time cardiovascular MR with spatio-temporal artifact suppression using deep learning—Proof of concept in congenital heart disease," *Magnetic Resonance Med.*, vol. 81, no. 2, pp. 1143–1156, 2019.

[20] M. Akçakaya, S. Moeller, S. Weingärtner, and K. Uğurbil, "Scan-specific robust artificial-neural-networks for k-space interpolation (RAKI) reconstruction: Database-free deep learning for fast imaging," *Magnetic Resonance Med.*, vol. 81, no. 1, pp. 439–453, 2019.

[21] F. Knoll, K. Hammernik, E. Kobler, T. Pock, M. P. Recht, and D. K. Sodickson, "Assessment of the generalization of learned image reconstruction and the potential for transfer learning," *Magnetic Resonance Med.*, vol. 81, no. 1, pp. 116–128, 2019.

[22] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen, "Image reconstruction by domain-transform manifold learning," *Nature*, vol. 555, no. 7697, p. 487, 2018.

[23] K. Kunisch and T. Pock, "A bilevel optimization approach for parameter learning in variational models," *SIAM J. Imag. Sci.*, vol. 6, no. 2, pp. 938–983, 2013.

[24] S. Lunz, O. Öktem, and C.-B. Schönlieb, "Adversarial regularizers in inverse problems," in *Proc. Neural Inform. Process. Syst. Conf.*, pp. 8507–8516, 2018.

[25] A. Dave, A. K. Vadathya, R. Subramanyam, R. Baburajan, and K. Mitra, "Solving inverse computational imaging problems using deep pixel-level prior," *IEEE Trans. Comp. Imag.*, vol. 5, no. 1, pp. 37–51, 2018.

[26] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, 2007.

[27] J. A. Fessler, "Model-based image reconstruction for MRI," *IEEE Signal Process. Mag.*, vol. 27, no. 4, pp. 81–89, 2010.

[28] M. S. Hansen and P. Kellman, "Image reconstruction: An overview for clinicians," *Journal of Magnetic Resonance Imaging*, vol. 41, no. 3, pp. 573–585, 2015.

[29] A. Macovski, "Noise in MRI," *Magnetic Resonance Med.*, vol. 36, no. 3, pp. 494–497, 1996.

[30] B. Adcock, A. Bastounis, and A. C. Hansen, "From global to local: Getting more from compressed sensing," *SIAM News*, October 2017.

[31] P. J. Shin, P. E. Larson, M. A. Ohliger, M. Elad, J. M. Pauly, D. B. Vigneron, and M. Lustig, "Calibrationless parallel imaging reconstruction based on structured low-rank matrix completion," *Magnetic Resonance Med.*, vol. 72, no. 4, pp. 959–970, 2014.

[32] L. Ying and J. Sheng, "Joint image reconstruction and sensitivity estimation in SENSE (JSENSE)," *Magnetic Resonance Med.*, vol. 57, no. 6, pp. 1196–1202, 2007.

[33] Y. Liu, Z. Zhan, J.-F. Cai, D. Guo, Z. Chen, and X. Qu, "Projected iterative soft-thresholding algorithm for tight frames in compressed sensing magnetic resonance imaging," *IEEE Trans. Med. Imag.*, vol. 35, no. 9, pp. 2130–2140, 2016.

[34] C. Bilen, I. W. Selesnick, Y. Wang, R. Otazo, D. Kim, L. Axel, and D. K. Sodickson, "On compressed sensing in parallel MRI of cardiac perfusion using temporal wavelet and TV regularization," in *Proc. IEEE Int. Conf. Acoust. Speech & Signal Process.*, pp. 630–633, 2010.

[35] R. Ahmad and P. Schniter, "Iteratively reweighted ℓ_1 approaches to sparse composite regularization," *IEEE Trans. Comp. Imag.*, vol. 10, pp. 220–235, Dec. 2015.

[36] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[37] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 3, no. 1, pp. 123–231, 2013.

[38] S. Sreehari, S. V. Venkatakrishnan, B. Wohlberg, G. T. Buzzard, L. F. Drummy, J. P. Simmons, and C. A. Bouman, "Plug-and-play priors for bright field electron tomography and sparse interpolation," *IEEE Trans. Comp. Imag.*, vol. 2, pp. 408–423, 2016.

[39] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Trans. Comp. Imag.*, vol. 3, no. 1, pp. 84–98, 2017.

[40] A. Pour Yazdanpanah, O. Afacan, and S. Warfield, "Deep plug-and-play prior for parallel MRI reconstruction," *Proc. IEEE Intl. Conf. Comp. Vision*, 2019.

[41] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[42] E. Esser, X. Zhang, and T. F. Chan, "A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science," *SIAM J. Imag. Sci.*, vol. 3, no. 4, pp. 1015–1046, 2010.

[43] U. Kamilov, H. Mansour, and B. Wohlberg, "A plug-and-play priors approach for solving nonlinear imaging inverse problems," *IEEE Signal Process. Lett.*, vol. 24, pp. 1872–1876, May 2017.

[44] S. Ono, "Primal-dual plug-and-play image restoration," *IEEE Signal Process. Lett.*, vol. 24, no. 8, pp. 1108–1112, 2017.

[45] Y. Sun, B. Wohlberg, and U. S. Kamilov, "An online plug-and-play algorithm for regularized image reconstruction," *IEEE Trans. Comp. Imag.*, vol. 5, pp. 395–408, 2019.

[46] S. T. Ting, R. Ahmad, N. Jin, J. Craft, J. Serafim da Silverira, H. Xue, and O. P. Simonetti, "Fast implementation for compressive recovery of highly accelerated cardiac cine MRI using the balanced sparse model," *Magnetic Resonance Med.*, vol. 77, pp. 1505–1515, Apr. 2017.

[47] Z. Shen, K.-C. Toh, and S. Yun, "An accelerated proximal gradient algorithm for frame-based image restoration via the balanced approach," *SIAM J. Imag. Sci.*, vol. 4, no. 2, pp. 573–596, 2011.

[48] G. T. Buzzard, S. H. Chan, S. Sreehari, and C. A. Bouman, "Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium," *SIAM J. Imag. Sci.*, vol. 11, no. 3, pp. 2001–2020, 2018.

[49] S. H. Chan, "Performance analysis of plug-and-play ADMM: A graph signal processing perspective," *IEEE Trans. Comp. Imag.*, vol. 5, no. 2, pp. 274–286, 2019.

[50] A. M. Teodoro, J. M. Bioucas-Dias, and M. A. T. Figueiredo, "Scene-adapted plug-and-play algorithm with convergence guarantees," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, pp. 1–6, Sep. 2017.

[51] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "Nonlocal transform-domain filter for volumetric data denoising and reconstruction," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 119–133, 2013.

[52] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig, "ESPIRiT—An eigenvalue approach to autocalibrating parallel MRI: Where SENSE meets GRAPPA," *Magnetic Resonance Med.*, vol. 71, no. 3, pp. 990–1001, 2014.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Internat. Conf. on Learning Repres.*, 2015.

[54] R. Ahmad, H. Xue, S. Giri, Y. Ding, J. Craft, and O. P. Simonetti, "Variable density incoherent spatiotemporal acquisition (VISTA) for highly accelerated cardiac MRI," *Magnetic Resonance Med.*, vol. 74, no. 5, pp. 1266–1278, 2015.

[55] M. Buehrer, K. P. Pruessmann, P. Boesiger, and S. Kozerke, "Array compression for MRI with large coil arrays," *Magnetic Resonance Med.*, vol. 57, no. 6, pp. 1131–1139, 2007.

[56] R. Otazo, E. Candès, and D. K. Sodickson, "Low-rank plus sparse matrix decomposition for accelerated dynamic MRI with separation of background and dynamic components," *Magnetic Resonance Med.*, vol. 73, no. 3, pp. 1125–1136, 2015.

[57] B. Wen, S. Ravishankar, L. Pfister, and Y. Bresler, "Transform learning for magnetic resonance image reconstruction: From model-based learning to building neural networks," *arXiv:1903.11431*, 2019.

[58] S. Ravishankar, B. E. Moore, R. R. Nadakuditi, and J. A. Fessler, "Low-rank and adaptive sparse signal (LASSI) models for highly accelerated dynamic imaging," *IEEE Trans. Med. Imag.*, vol. 36, no. 5, pp. 1116–1128, 2017.

[59] Z. Tan, Y. C. Eldar, A. Beck, and A. Nehorai, "Smoothing and decomposition for analysis sparse recovery," *IEEE Trans. Signal Process.*, vol. 62, pp. 1762–1774, April 2014.

[60] J. Zbontar, F. Knoll, A. Sriram, M. J. Muckley, M. Bruno, A. Defazio, M. Parente, K. J. Geras, J. Katsnelson, H. Chandarana, *et al.*, "fastMRI: An open dataset and benchmarks for accelerated MRI," *arXiv:1811.08839*, 2018.

[61] A. Buades, B. Coll, and J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Sim.*, vol. 4, no. 2, pp. 490–530, 2005.

[62] P. Milanfar, "A tour of modern image filtering: New insights and methods, both practical and theoretical," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 106–128, 2013.

[63] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, 2017.

[64] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, 1962.

[65] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2007.

[66] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, pp. 18914–18919, Nov. 2009.

[67] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inform. Theory*, vol. 57, pp. 764–785, Feb. 2011.

[68] S. Som and P. Schniter, "Compressive imaging using approximate message passing and a Markov-tree prior," *IEEE Trans. Signal Process.*, vol. 60, pp. 3439–3448, July 2012.

[69] D. L. Donoho, I. M. Johnstone, and A. Montanari, "Accurate prediction of phase transitions in compressed sensing via a connection to minimax denoising," *IEEE Trans. Inform. Theory*, vol. 59, June 2013.

[70] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Trans. Inform. Theory*, vol. 62, no. 9, pp. 5117–5144, 2016.

[71] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector approximate message passing," in *Proc. IEEE Int. Symp. Inform. Thy.*, pp. 1588–1592, 2017.

[72] P. Schniter, S. Rangan, and A. K. Fletcher, "Denoising-based vector approximate message passing," in *Proc. Intl. Biomed. Astronom. Signal Process. (BASP) Frontiers Workshop*, p. 77, 2017.

[73] R. Berthier, A. Montanari, and P.-M. Nguyen, "State evolution for approximate message passing with non-separable functions," *arXiv:1708.03950*, 2017.

[74] A. K. Fletcher, P. Pandit, S. Rangan, S. Sarkar, and P. Schniter, "Plug-in estimation in high-dimensional linear inverse problems: A rigorous analysis," in *Proc. Neural Inform. Process. Syst. Conf.*, pp. 7440–7449, 2018.

[75] T. S. Huang, G. J. Yang, and Y. T. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Trans. Acoust. Speech & Signal Process.*, vol. 27, no. 1, pp. 13–18, 1979.

[76] W. Rudin, *Principles of Mathematical Analysis*. New York: McGraw-Hill, 3rd ed., 1976.

[77] S. Kantorovitz, *Several Real Variables*. Springer, 2016.

[78] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[79] R. R. Coifman and D. L. Donoho, "Translation-invariant de-noising," in *Wavelets and Statistics* (A. Antoniadis and G. Oppenheim, eds.), pp. 125–150, Springer, 1995.

[80] F. Ong, P. Milanfar, and P. Getreurer, "Local kernels that approximate Bayesian regularization and proximal operators," *arXiv:1803.03711*, 2018.

[81] P. Milanfar, "Symmetrizing smoothing filters," *SIAM J. Imag. Sci.*, vol. 30, no. 1, pp. 263–284, 2013.

[82] P. Milanfar and H. Talebi, "A new class of image filters without normalization," in *Proc. IEEE Int. Conf. Image Process.*, pp. 3294–3298, 2016.

[83] T. Goldstein, C. Studer, and R. Baraniuk, "Forward-backward splitting with a FASTA implementation," *arXiv:1411.3406*, 2014.

[84] H. Robbins, "An empirical Bayes approach to statistics," in *Proc. Berkeley Symp. Math. Stats. Prob.*, pp. 157–163, 1956.

[85] B. Efron, "Tweedie's formula and selection bias," *J. Am. Statist. Assoc.*, vol. 106, no. 496, pp. 1602–1614, 2011.

[86] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, 2011.

[87] C. M. Stein, "Estimation of the mean of a multivariate normal distribution," *Ann. Statist.*, vol. 9, pp. 1135–1151, 1981.

[88] F. Luisier, *The SURE-LET Approach to Image Denoising*. PhD thesis, EPFL, Lausanne, Switzerland, 2010.

[89] M. Raphan and E. P. Simoncelli, "Least squares estimation without priors or supervision," *Neural Comput.*, vol. 23, pp. 374–420, Feb. 2011.

[90] T. Blu and F. Luisier, "The SURE-LET approach to image denoising," *IEEE Trans. Image Process.*, vol. 16, no. 11, pp. 2778–2786, 2007.

[91] S. A. Bigdeli and M. Zwicker, "Image restoration using autoencoding priors," *arXiv:1703.09964*, 2017.

[92] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3563–3593, 2014.

[93] M. A. Ranzato, Y.-L. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," in *Proc. Neural Inform. Process. Syst. Conf.*, pp. 1185–1192, 2008.

[94] A. Beck and M. Teboulle, "Gradient-based algorithms with applications to signal recovery," in *Convex Optimization in Signal Processing and Communications* (D. P. Palomar and Y. C. Eldar, eds.), pp. 42–88, Cambridge, 2009.

[95] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering* (H. Bauschke, R. Burachik, P. Combettes, V. Elser, D. Luke, and H. Wolkowicz, eds.), pp. 185–212, Springer, 2011.

[96] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 794–816, 2017.

[97] M. A. T. Figueiredo, J. M. Bioucas-Dias, and R. D. Nowak, "Majorization-minimization algorithms for wavelet-based image restoration," *IEEE Trans. Image Process.*, vol. 16, no. 12, pp. 2980–2991, 2007.

[98] C. A. Metzler, P. Schniter, A. Veeraraghavan, and R. G. Baraniuk, "prDeep: Robust phase retrieval with flexible deep neural networks," in *Proc. Int. Conf. Mach. Learning*, pp. 3501–3510, 2018 (see also *arXiv:1803.00212*).

[99] A. Sidi, *Vector Extrapolation Methods with Applications*. SIAM, 2017.

[100] T. Hong, Y. Romano, and M. Elad, "Acceleration of RED via vector extrapolation," *arXiv:1805:02158*, 2018.

[101] H. H. Bauschke. and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. CMS Books in Mathematics, New York: Springer, 1st ed., 2011.

[102] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 770–778, 2016.

[103] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *Proc. IEEE Int. Conf. Acoust. Speech & Signal Process.*, 2017.

[104] M. Wharton, A. Pavy, and P. Schniter, "Phase-Modulated radar waveform classification using deep networks," *arXiv:2102:07827*, Feb. 2021.

[105] P. Perera, P. Oza, and V. M. Patel, "One-class classification: A survey," *arXiv:12101.03064*, 2021.

[106] P. Perera and V. M. Patel, "Deep transfer learning for multiple class novelty detection," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 11544–11552, 2019.

[107] D. Miller, N. Sünderhauf, M. Milford, and F. Dayoub, "Class anchor clustering: A loss for distance-based open set recognition," *arXiv:2004.02434*, 2020.

[108] L. Ruff, J. R. Kauffmann, and R. A. Vandermeulen, "A unifying review of deep and shallow anomaly detection," *Proc. IEEE*, pp. 756–795, 2021.

[109] R. Gnanadesikan and J. R. Kettenring, "Robust estimates, residuals, and outlier detection with multiresponse data," *Biometrics*, pp. 81–124, 1972.

[110] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using k-nearest neighbour graph," 2004.

[111] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, pp. 1443–1471, July 2001.

[112] D. Tax and R. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, pp. 45–66, 2004.

[113] H. V. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer, 2nd ed., 1994.

[114] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[115] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 3rd ed., 1991.

[116] M. S. Anderson, J. Dahl, and L. Vandenberghe, "CVXOPT: A python package for convex optimization."

[117] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Proc. Neural Inform. Process. Syst. Conf.*, 2018.

[118] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. Neural Inform. Process. Syst. Conf.*, 2011.

[119] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *arXiv:1409.0575*, 2014.

[120] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv:1506.03365*, 2015.

[121] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution imge detection in neural networks," in *Proc. Internat. Conf. on Learning Repres.*, 2018.

[122] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, 2016.

[123] Y. You, B. Ginsburg, and I. Gitman, "Large batch training of convolutional networks," *arXiv:1708:03888v3*, 2017.

[124] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rate," in *Proc. SPIE, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, May 2019.

[125] J. Andrews, T. Tanay, E. J. Morton, and L. D. Griffin, "Transfer representation-learning for anomaly detection," in *Proc. Int. Conf. Mach. Learning*, 2016.

[126] X. Li, F. Dong, S. Zhang, and W. Guo, "A survey on deep learning techniques in wireless signal recognition," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 12, Article ID 5629572, 2019.

[127] T. J. O'Shea and N. West, "Radio machine learning dataset generation with GNU radio," in *Proc. GNU Radio Conf.*, vol. 1, 2016.

[128] S. Harrison, R. Coles, T. Robishaw, and D. D. Rizzo, "RFI novelty detection using machine learning techniques," in *Proc. IEEE RFI Workshop*, 2019.

[129] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proc. Internat. Conf. on Learning Repres.*, 2017.

[130] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, pp. 41–50, 2019.

[131] R. V. Chakravarthy, H. Liu, and A. M. Pavy, "Open-set radar waveform classification: Comparison of different features and classifiers," in *Proc. IEEE Radar Conf.*, pp. 542–547, 2020.

[132] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?," in *Proc. Neural Inform. Process. Syst. Conf.*, (Vancouver, B.C.), Dec. 2019.

[133] D. Bahri, H. Jiang, and D. Metzler, "Label smoothed embedding hypothesis for out-of-distribution detection," *arXiv:2102.05131*, 2021.

[134] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, pp. 4037–4058, 2021.

[135] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve robustness and uncertainty," in *Proc. Neural Inform. Process. Syst. Conf.*, 2019.

[136] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Conf. Comp. Vision Pattern Recog.*, 2006.

[137] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Proc. Neural Inform. Process. Syst. Conf.*, vol. 6, 1993.

[138] K. Sohn, "Improved deep metric learning with multi-class N-pair loss objective," in *Proc. Neural Inform. Process. Syst. Conf.*, 2016.

[139] B. Rigling and C. Roush, "ACF-Based classification of phase modulated waveforms," in *Proc. IEEE Radar Conf.*, pp. 287–291, 2010.

[140] GNU Radio Website, accessed January 2021. available at gnuradio.org.