

OPTIMIZING DEEP UNFOLDING NETWORKS FOR COMPRESSIVE SENSING

BY

RAWWAD SULTAN ALHEJAILI

A Thesis Presented to the
COLLEGE OF GRADUATE AND INTERDISCIPLINARY STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
ELECTRICAL ENGINEERING

OPTIMIZING DEEP UNFOLDING NETWORKS FOR COMPRESSIVE SENSING

BY

RAWWAD SULTAN ALHEJAILI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

In Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

KING FAHD UNIVERSITY
OF PETROLEUM & MINERALS

Dhahran, Saudi Arabia

MAY, 2024

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

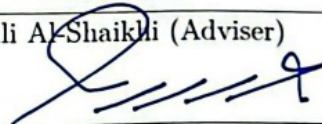
DEANSHIP OF GRADUATE STUDIES

This thesis, written by **RAWWAD SULTAN ALHEJAILI** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN ELECTRICAL ENGINEERING**.

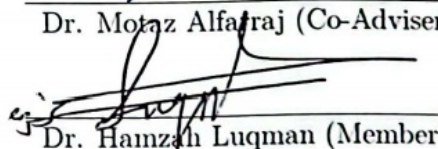
Thesis Committee



Dr. Ali Al-Shaikhi (Adviser)



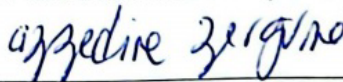
Dr. Motaz Alfarrag (Co-Adviser)



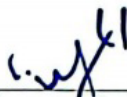
Dr. Hamzah Luqman (Member)



Dr. Mudassir Masood (Member)



Dr. Azzedine Zerguine (Member)



Dr. Mohammed AlMuhaini
Department Chairman



Dr. Suliman Al-Homidan
Dean of Graduate Studies



Date

©Rawwad Sultan Alhejaili
2024

Dedication

I dedicate the work behind this thesis to my ever-loving family. My father Sultan, my mother Aisha, my older brother, Aseel, and my sisters, Nasab, Nardeen, and Nuran. I would not have made it this far without your support, so thanks for being there with me throughout this adventure. Finally, I would like to also dedicate this thesis to my cousin, Mohammed Hussein Alhejaili. His memories will forever live on in our hearts. And we ask God the Almighty to grant him Al-Firdaus... Amen.

ACKNOWLEDGMENTS

First, I want to extend my acknowledgment to my advisor, Dr. Ali Al-Shaikhi, and my co-advisor, Dr. Motaz Alfarraj. They both have helped me tremendously throughout the years through both my highs and lows and for that, I couldn't be more grateful. The guidance of Dr. Motaz in particular has been insurmountable in shaping my MSc experience as a whole. He has been with me every single step along the way, and despite my many setbacks at the start of my MSc journey, he still firmly believed in me against all odds. And thanks to his unwavering support, I went from having no research experience to someone who sets his sights on the top research venues in the field, which is absolutely wild given how I started out this journey. My writing cannot describe how indebted I am to him, but what I can say at least is that I am eternally grateful for all the effort he has poured into my development, and I could never have been prouder and more blessed to be his student. Thus, I deeply thank him from the bottom of my heart for being my co-advisor!

I also extend my acknowledgment to all the other members of my thesis committee: Dr. Hamzah Lugman, Dr. Mudassir Masood, and Prof. Azzedine Zerguine. They have been extremely helpful in sharpening my research skills and further improving

my writing skills.

Next, I would like to acknowledge the SDAIA-KFUPM Joint Research Center. They have been incredibly helpful and played a pivotal role throughout my MSc journey. They provided me with high-end workstations, and without them, I would not have been able to run all the experiments you see in this thesis. Not only this, but they also fostered a very friendly work environment that was extremely welcoming to anyone who visited. As such, I was able to make mistakes and learn from them, particularly when it came to presentation skills, and for this, I am extremely grateful to everyone working in the research center.

And of course, I cannot forget all the professors who have been my instructors on the courses I have undertaken. They are in chronological order: Dr. Motaz Alfarraj, Dr. Suhail Al-Dharrab, Prof. Wail Mousa, Prof. Azzedine Zerguine, Dr. Hamzah Luqman, Dr. Abdulazeez Abdulraheem, and Prof. Mohamed Mohandes. They have all played a role in shaping who I am academically, so thank you. Prof. Wail in particular has played a big role in both my academic and research development, and I will always be grateful for this. I also would like to extend my gratitude to both Dr. Naveed Iqbal and Dr. Mudassir Masood for expanding my research horizons to fields I am not as familiar with.

In addition, I would like to acknowledge my undergraduate professors from Taibah University for their support. In particular, I would like to thank Dr. Raghied

Atta, Dr. Yassin Mahmoud, and Dr. Adel Hafiz. They have been an inspiration to me and they are in fact the ones responsible for sparking my curiosity, and thus, without them, I may have never started pursuing higher education to begin with, so I will forever be indebted to you! I would also like to extend my thanks to Dr. Hattan Abutarboush, Dr. Badr Lami, Dr. Omar Siddiqui, Dr. Mohammed Alzahrani, Dr. Abdullah Zaman, Dr. Fahd Ghonaim, Dr. Ahmad Alferidi, and Dr. Ahmed Alahmadi.

I would also like to extend my thanks to all the friends and colleagues I made here at KFUPM. I had the pleasure and honor of meeting Ziyad Alswaidan, Khalil Chikhaoui, Mahmoud Yassin, and Ahmed Alqulayti. You have all helped me get acquainted with living away from home, so you have my thanks. And of course, I cannot forget to extend my gratitude to my squad; my band of friends from Almadinah, some of whom I knew before seeing the light of day! They are Faris Salah Alharbi, Yasser Salah Alharbi, Abdullah Abdulaziz Al-Dgran, and Mshari Mohammed Alahmadi. Even though I'm living away from all of you now, you have made me feel as if I have never left home through your continued online interactions, and for this, you have my thanks.

And last but not least, I would like to offer my deepest gratitude to every member of my family. My father, Sultan, my mother, Aisha, my older brother, Aseel, my sisters, Nasab, Nardeen, and Nuran. Without you, I could have never been able to go this far. You have all acted as pillars of support to me during the hardest of times, and for this, I can never hope to thank you enough.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	v
LIST OF TABLES	xi
LIST OF FIGURES	xii
ABSTRACT (ENGLISH)	xiii
ABSTRACT (ARABIC)	xv
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement	5
1.2 Contribution	6
CHAPTER 2 LITERATURE SURVEY	8
2.1 Compressed Sensing Background	8
2.2 Sampling Matrices	12
2.3 Initialization	13
2.4 Iterative Optimization Based Methods	15
2.5 Deep Compressed Sensing Networks	17
2.6 Deep Unfolding Networks	19
2.7 Recursive Networks	22
2.8 Gradient Descent Module	23
CHAPTER 3 RECURSIVE DEEP UNFOLDING NETWORKS	25
3.1 Introduction	25

3.2	Methodology	27
3.2.1	COAST	27
3.2.2	Recursions	29
3.2.3	Recursion-based Feature Modulation Unit	30
3.3	Experimental Results	31
3.3.1	COAST vs. Recursive COAST	32
3.3.2	ISTA-Net+ vs Recursive ISTA-Net+	33
3.3.3	CASNet vs Recursive CASNet	34
3.4	Ablation Study	35
3.4.1	RFMU Enhances the Efficiency of Recursions	35
3.4.2	Design of the RFMU	37
3.5	Conclusion	38
CHAPTER 4	GRADIENT-INFORMED NETWORK	40
4.1	Introduction	40
4.2	Methodology	44
4.2.1	Sampling and Initialization	44
4.2.2	GDM and Deep Reconstruction Network	45
4.2.3	Gradient-Informed Network	45
4.3	Experimental Results	48
4.3.1	GIN vs. GDM	49
4.4	Ablation Study	50
4.4.1	Integrating Gradient-Informed Networks to the Recursive Framework	50
4.4.2	The Impact of Removing GDM	52
4.4.3	Why Does the GDM Matter to DUNs?	53
4.5	Conclusion	55
CHAPTER 5	CONCLUSION	57
5.1	Summary	57
5.2	Future Work	59

REFERENCES	61
VITAE	70

LIST OF TABLES

Table 2.1	Summary of the Literature Review	24
Table 3.1	PSNR/SSIM scores of the models on the BSD68 [1] dataset. .	32
Table 3.2	Training time and number of parameters of the models in Table 3.1.	33
Table 3.3	PSNR/SSIM performance comparison of the networks before and after the addition of the RFMU unit on the Set11 dataset [2].	36
Table 3.4	PSNR/SSIM scores on the Set11 dataset [2] showing the impact of each step done to the RFMU.	38
Table 4.1	PSNR/SSIM performance comparison on the Set11 and BSD68 datasets [2], [1].	49
Table 4.2	PSNR/SSIM scores comparing recursive GIN to recursive GDM networks on the Set11 [2] dataset.	51
Table 4.3	PSNR/SSIM scores comparing recursive GIN to recursive GDM networks on the BSD68 [1] dataset.	51
Table 4.4	PSNR/SSIM performance comparison on the BSD68 dataset [1].	52

LIST OF FIGURES

Figure 1.1	General schematic of most deep unfolding networks.	4
Figure 2.1	An example from the Set11 dataset [2]. (a) is the ground truth image, (b) is the recovered image using the DCT basis, and (c) is the recovered image using the SVD basis.	9
Figure 2.2	General structure of many deep unfolding networks.	20
Figure 3.1	A comparison between conventional deep unfolding networks and our recursive framework.	26
Figure 3.2	Illustration of the design of the recovery block of COAST [3] and also our proposed RFMU unit.	27
Figure 3.3	Recovery block of ISTA-Net+ [4] after the addition of the RFMU unit.	31
Figure 4.1	The learning curves for two COAST networks [3] on the validation dataset. One uses the GDM while the other does not. .	43
Figure 4.2	Recovery block schematic before and after using GIN.	48
Figure 4.3	Shows an image (a) and its respective initial solution (b) when the CS ratio is 10%	53
Figure 4.4	Intermediate recovery results from COAST [3] once without GDM, once when GDM is used, and once with our proposed GIN.	55

THESIS ABSTRACT

NAME: Rawwad Sultan Alhejaili

TITLE OF STUDY: Optimizing Deep Unfolding Networks for Compressive Sensing

MAJOR FIELD: Electrical Engineering

DATE OF DEGREE: May, 2024

Compressed sensing (CS) theory is an appealing approach to data acquisition due to its low sampling rate requirement. Deep unfolding networks have been an attractive choice for solving CS problems due to their simplicity and interpretability. With that said, their general framework suffers from two problems. First, the iterative design of such networks can suffer from redundancies. This increases the model's complexity necessitating a larger and more diverse training dataset. Second, many deep unfolding networks minimize the loss incurred in the measurements domain by using a simple gradient descent module (GDM). Since GDM only consists of a single learnable parameter, its effectiveness is limited. Furthermore, the GDM is isolated from the reconstruction network meaning that the network cannot directly minimize the loss in the measurements domain, which impacts its performance. In this work,

we propose a recursive framework to both reduce the redundancies in deep unfolding networks and enhance their performance. First, a recursive connection is added to each recovery block to reduce the overall redundancies in the framework. Furthermore, a learnable unit is introduced to modulate the features of the network based on recursion statistics, which aims to utilize the recursive connection more effectively. Second, we propose the gradient-informed network (GIN) as a replacement for GDM. Unlike GDM, GIN allows the reconstruction network to directly minimize the loss in the measurements domain, which leads to an overall uplift in performance compared to GDM. The results show that our recursive framework can reduce as much as 75% of the network parameters while enhancing its performance thanks to GIN.

ملخص الرسالة

الاسم: رُوَاد سلطان الحجيلي

عنوان الدراسة: تحسين شبكات النشر العميقة في التحسس المضغوط

التخصص: الهندسة الكهربائية

تاريخ الدرجة العلمية: مايو ٢٠٢٤

تعتبر نظرية التحسس المضغوط نهجاً جذاباً في الحصول على البيانات بسبب انخفاض متطلباتها من حيث معدل أخذ العينات. وقد كانت شبكات النشر العميقة خياراً جذاباً لحل مشاكل التحسس المضغوط نظراً لبساطتها بالإضافة إلى كونها أسهل للتفسير. ومع ذلك ، فإن إطارها العام يعاني من مشكلتين. أولاً ، يمكن للتصميم المتكرر لمثل هذه الشبكات أن يعاني من الحشو. وهذا يزيد من تعقيد الشبكة مما يستلزم مجموعة بيانات تدريبية أكبر وأكثر تنوعاً. وثانياً ، العديد من شبكات النشر العميقة تقوم بالتقليل من الخسارة المتكبدة في مجال القياسات باستخدام وحدة انحدار تدريجي بسيطة. وبما أن وحدة الانحدار التدريجي تتألف فقط من متغير واحد قابل للتعليم ، فإن فعاليتها تعد محدودة. وعلاوة على ذلك ، وحدة الانحدار التدريجي معزولة عن شبكة الإعمار بمعنى أن الشبكة لا تستطيع أن تقلل من الخسارة في مجال القياسات بشكل مباشر ، مما يؤثر على أدائها. في هذه الرسالة ، نقترح إطاراً متكرراً للحد من الحشو في شبكات النشر العميقة وتعزيز أدائها. أولاً ، يتم إضافة إتصال متكرر إلى كل كتلة استرداد للحد من الحشو الموجود بمثل هذه الشبكات. وعلاوة على ذلك ، يجري إدخال وحدة للتعليم من أجل تعديل سمات الشبكة بناءً إلى إحصائيات التكرار ، والتي تهدف إلى استخدام الإتصال المتكرر على نحو أكثر فعالية. ثانياً ، نقترح دعم الشبكة بمعلومات المنحدر بدلاً من استخدام وحدة الانحدار التدريجي. هذا يسمح للشبكة المعززة بمعلومات المنحدر بالتقليل من الخسارة في مجال القياسات بشكل مباشر ، مما يؤدي إلى ارتفاع عام في الأداء مقارنة باستخدام وحدة الانحدار التدريجي. تبين النتائج أن إطارنا التكراري يمكن أن يخفض ما يصل إلى ٧٥٪ من عدد المتغيرات بالشبكة بالإضافة إلى تحسين ادائها بفضل تعزيز الشبكة بمعلومات المنحدر.

CHAPTER 1

INTRODUCTION

Nowadays, most of the media we consume are in digital format. From TV shows to podcasts to music, they all live in the digital domain. However, all those signals do in fact originate from the analog domain. Due to the advantages digital signals have over their analog counterpart such as noise suppression, analog-to-digital converters (ADCs) are often employed. They work by capturing a snapshot of the analog signal at a specified sampling rate. If the sampling rate is high enough, then the reconstructed digital signal will be identical to its analog counterpart. Therefore, the choice of the sampling rate plays a critical role in the quality of the captured signal. If the sampling rate is too low, aliasing effects occur and the original analog signal cannot be recovered. Increasing the sampling rate too high unnecessarily increases the energy and cost of the ADC. Thus, choosing the minimum possible sampling rate is often advised.

The Nyquist–Shannon sampling theorem [5] defines the minimum sampling rate required to perfectly capture any signal. It states that if the highest frequency present in the signal is B , then any sampling rate greater than $2B$ can perfectly capture that signal.

However, sampling the signal at the Nyquist rate [5] can be unappealing in many cases. For example, in data storage, fulfilling the Nyquist rate can lead to excess storage leading to insufficient utilization of the data drives. In seismic data acquisition, economic or geological constraints can make it difficult to sample the data at the Nyquist–Shannon rate [6]. In medical imaging, requiring a high sampling rate is especially undesirable [7]. For example, X-ray computed tomography (CT) imaging employs ionizing radiation, and excessive exposure to this radiation can increase the patient’s risk of developing certain types of cancer [8]. Therefore, the minimum sampling rate governed by the Nyquist–Shannon sampling theorem [5] can still be unappealing in some applications.

Compressed sensing (CS) challenges this conventional golden standard as it promises that under certain conditions, signals sampled below the Nyquist rate can *still* be recovered [9] [10]. CS achieves this by utilizing the fact that most signals of interest follow a structure. Thus, if a signal is represented in a sparsifying basis that adheres to that structure, then the signal becomes sparse and only a few coefficients are necessary to describe it [11]. CS then exploits this by finding the fewest possible coefficients that agree with the compressed measurements [12] [13] [14]. This is represented by

$$\mathbf{y} = \Phi \mathbf{x} = \Phi \Psi \mathbf{s}, \quad (1.1)$$

$$\min \|\mathbf{s}\|_0 \quad \text{subject to} \quad \mathbf{y} = \Phi \Psi \mathbf{s}. \quad (1.2)$$

where $\mathbf{y} \in \mathbb{R}^M$ is the sampled measurements, $\mathbf{x} \in \mathbb{R}^N$ is the uncompressed signal, $\Phi \in \mathbb{R}^{M \times N}$ is the sampling matrix, $\Psi \in \mathbb{R}^{N \times N}$ is the sparsifying basis, and $\mathbf{s} \in \mathbb{R}^N$ is

the signal $\mathbf{x}$ after being transformed to the sparse domain. The CS ratio is defined by $sr = M/N$ where $M \ll N$. The CS ratio is also frequently called the sampling ratio.

While the use of compressed sensing can be advantageous, it does suffer from some downsides. First, the performance of CS is highly dependent on the choice of the sparsifying basis [15], which exhibits different sparsity levels depending on the data. Secondly, finding the sparsest possible solution to Equation (1.2) involves iterative calculations that are computationally expensive to run as they require many iterations to converge [13] [14] [16].

Such problems can be circumvented by adapting the general CS problem to a deep compressed sensing network (DCSN). Those networks can find a better basis using a data-driven approach [17] while utilizing the representative power of deep networks. This results in DCSNs having better reconstruction quality than purely conventional methods and at a lower computational complexity [18]. However, many DCSNs suffer from a lack of interpretability due to their black-box design.

To address the interpretability concerns, deep unfolding networks (DUNs) have been proposed for CS. Such networks are designed after a specific conventional CS method. This allows DUNs to have higher interpretability than regular DCSNs while still achieving state-of-the-art results. Most DUNs are designed around ISTA due to its use of the proximal gradient descent (PGD) [3], [19], [20]. Such networks are often called proximal gradient descent deep unfolding networks (PGD-DUNs). However, there have been DUNs that are designed after different conventional CS techniques such as the alternating direction method of multipliers (ADMM) [21], [22] and approximate message passing (AMP) [23], [24].

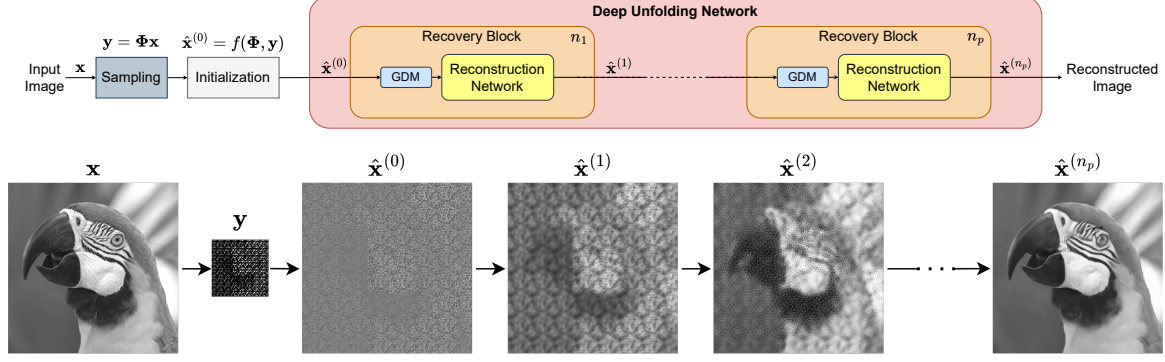


Figure 1.1: General schematic of most deep unfolding networks.

Although the design of DUNs can vary, the most common one is depicted in Figure 1.1, which is used by PGD-DUNs (e.g., [4], [3], [25], [20], [26]). First, the image is compressed in the sampling block using a sampling matrix. Then, an initial solution is constructed, which restores the dimensions of the image. Afterward, this initial solution is fed to the deep unfolding network, which consists of multiple sequential recovery blocks. Each recovery block performs two steps. The first one is called the gradient descent module (GDM), which as the name suggests, performs gradient descent on the input image. It is done by the following:

$$\mathbf{x}^{(k)} = \hat{\mathbf{x}}^{(k-1)} - \rho \Phi^T (\Phi \hat{\mathbf{x}}^{(k-1)} - \mathbf{y}), \quad (1.3)$$

where k is the iteration index, $\mathbf{x}$ is the uncompressed image, $\hat{\mathbf{x}}$ is the reconstructed image, and ρ is a learnable scalar. GDM helps to align the reconstructed image with its measurements, which is essential to CS theory. In the second step, the resulting image is fed to a reconstruction network to recover the image. Notably, despite all recovery blocks sharing the same structure, they all use distinct learnable parameters.

1.1 Problem Statement

In this work, we argue that despite the robustness of the general DUN framework in Figure 1.1, it can still be optimized further to provide higher efficiency and performance. Optimizing such a widely used framework means that many DUNs can stand to benefit from such optimization. Therefore, we propose two points of optimization to this framework. The first point tackles improving the efficiency of the framework, while the second one is concerned with enhancing its performance.

First, deep unfolding networks usually consist of multiple sequential recovery blocks, and increasing the number of recovery blocks can often improve their performance [4], [19], [20]. However, we argue that this practice can be inefficient. Specifically, we argue that the enhanced performance is not only attributed to the increase in recovery blocks but also to the increased number of *iterations*. As such, increasing the number of recovery blocks introduces unnecessary complexity to the networks, which makes them more prone to overfitting. Consequently, networks will require a larger and more diverse training dataset as a result [27], and this can also limit their deployment options [28]. Second, despite the importance of the GDM step, it only consists of a single learnable parameter, and thus, it could hardly utilize the representative power of deep learning (DL) networks. It also isolates the GDM step from the reconstruction step. As such, the reconstruction network could produce an output that is not consistent with the measurements, which is a necessary condition in CS theory.

1.2 Contribution

First, we aim to address the problem of the increasing size of deep unfolding networks. Specifically, we aim to introduce an approach that decreases the number of learnable parameters in said networks without compromising their performance. To that end, we propose using a recursive framework to tackle this problem. Its main idea is that the performance of deep unfolding networks can be improved without increasing their size by utilizing recursions. By performing several iterations over each recovery block, the number of learnable parameters can be decreased while maintaining the performance of the networks.

For the second point of optimization, we draw attention to the GDM step in Equation (1.3). We propose infusing the GDM step into the reconstruction network. This approach allows the reconstruction network to produce an output that is consistent with the measurements, and thus, make it aligned with CS theory.

In this thesis, our contributions can be summarized in the following:

- We improve the efficiency of DUNs for CS in terms of learnable parameters by introducing a recursive framework. By using recursions, the framework allows recovery blocks to be used multiple times before passing to the next block. This decreases the total number of *distinct* blocks being used, thus, decreasing the total number of learnable parameters.
- The recursive framework is further utilized by introducing a novel training strategy, which decreases the training time.

- To further enhance the efficiency of the recursive framework, a recursion-based feature modulation unit (RFMU) is proposed. The RFMU can modulate the features of the network based on recursion statistics, which improves the performance of the recursive framework.
- Finally, we propose a novel yet simple solution that allows reconstruction networks in most DUNs to produce reconstructions that are also consistent with the measurements. This is done by integrating the gradients of the GDM into the reconstruction networks, which we dub gradient-informed networks (GINs). The results demonstrate that despite the simplicity of GIN, it improves the performance of multiple DUNs.

The thesis will be structured as follows. First, a literature review is presented in Chapter 2, which discusses the basics of CS theory, sampling matrices, initialization, iterative optimization-based CS methods, DCSN, and DUNs. Then, in Chapter 3 we discuss our proposed recursive framework including the RFMU. Afterward, in Chapter 4, we discuss how gradients can be integrated into the reconstruction network using GIN. Finally, we conclude the findings of this thesis in Chapter 5.

CHAPTER 2

LITERATURE SURVEY

2.1 Compressed Sensing Background

Many signals such as natural images, audio, or seismic data have an inherent structure to them. Such signals are called structured signals. This structure makes them occupy a very small size in the image space. This can be seen by demonstrating the vastness of image space. Assume that we have a moderately small image of size 256×256 , and for simplicity, let us assume it is a binary image. Although such an image is limited in what it can represent visually, its image space contains $2^{256 \times 256}$ distinct images, which is over 10^{19728} images. For reference, by 2010, Google announced that their search engine had indexed over 10 billion images [29], which is negligible in the vastness of image space. In other words, structured signals are exceptionally rare when considering the vastness of image space. Therefore, if our interest only lies in natural images for example, then this implies that such images are highly compressible when represented in the proper basis, which is often called a sparsifying basis.

In general, the performance of sparsifying bases is influenced by the data. How-



Figure 2.1: An example from the Set11 dataset [2]. (a) is the ground truth image, (b) is the recovered image using the DCT basis, and (c) is the recovered image using the SVD basis.

ever, universal bases do exist such as Fourier modes or wavelets [15]. For improved compression, it is possible to construct a basis tailored to the data. This can be done using singular value decomposition (SVD) [30] for example [15]. However, this requires the sparsifying basis to be transmitted along with the measurements, which is undesirable.

Figure 2.1 demonstrates the compressibility of images and the effect of the sparsifying basis. In this example, we used an image from the Set11 dataset [2]. First, the image was transformed using the sparsifying basis of choice. In Figure 2.1(b), the discrete cosine transformed (DCT) was used, while for Figure 2.1(c), the SVD was used to construct a sparsifying basis tailored specifically to the ground truth image. Then, the highest 20% of the coefficients in terms of magnitude were kept, and the rest were set to zero. Finally, the image was transformed back to the spatial domain.

From Figure 2.1, it can be observed that even though only 20% of the coefficients were kept, the image mostly retains its quality in both bases. This demonstrates that

images can be compressed with minimal degradations. However, it can be observed that using a tailored basis significantly improves the compression quality of the image in terms of both the peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM). Since the basis was tailored specifically to the ground truth image, this is a perfect and unrealistic case for the SVD-constructed basis. However, this still demonstrates how the choice of the sparsifying basis can play an essential role in data compression.

While the above compression paradigm is certainly effective, it still requires the data to be sampled at the Nyquist-Shannon rate to determine which samples should be kept. In other words, the data acquisition stage will acquire samples that will ultimately be dropped. Consequently, this will unnecessarily increase the sampling rate and acquisition costs.

For this purpose, compressive sensing is proposed to perform the compression directly during the acquisition stage. Instead of sampling data points that will eventually be dropped, CS aims to directly sample the most important points of the data. Although the Nyquist-Shannon rate sets the minimum sampling rate for exact recovery, CS shows that by adopting a non-uniform sampling scheme, it can use a lower sampling rate and still recover the signal [12], [31]. Mathematically, the CS recovery problem can be represented by Equation (1.1)

Problems that require recovering $\mathbf{x}$ from $\mathbf{y}$ such as Equation (1.1) are widely used in other fields such as in super-resolution (e.g., [32], [33], [34], [35], [36]). In super-resolution, both terms on the right-hand side of Equation (1.1) are unknown, which makes it challenging to solve. While CS is challenging as well, the sampling matrix is

at least known, therefore, its solution set can be restricted by using Equation (1.1). If the reconstructed measurements obtained from the reconstructed signal do not match the original measurements, then the reconstructed signal cannot be correct. With this, algorithms can be designed to find a signal that returns the same measurements as the original signal. However, finding a solution that is consistent with the measurements is not enough to find the true solution. Since $M \ll N$, an infinite number of solutions still exist. Thus, an assumption about $\mathbf{x}$ must be made. In compressive sensing, it is stated that if the signal $\mathbf{x}$ is sparse in some domain, then it can be recovered [15], [37], [12]. Representing the sparse signal is done by:

$$\mathbf{x} = \Psi \mathbf{s}, \quad (2.1)$$

$$\mathbf{y} = \Phi \Psi \mathbf{s} = \Theta \mathbf{s}, \quad (2.2)$$

where $\Theta \in \mathbb{R}^{M \times N}$ is the sensing matrix.

In compressed sensing, $\mathbf{s}$ is assumed to be a sparse vector. Therefore, this problem can be solved by finding the vector $\mathbf{s}$ with the least non-zero entries that also match the measurements. Ideally, this can be achieved by minimizing the l_0 -norm of $\mathbf{s}$ [15], which is seen in Equation (1.2). However, l_0 -norm minimization is prone to combinatorial complexity [37], and thus, has a high computational cost. This can be overcome by using l_1 -norm minimization instead. In the literature, it has been shown that l_1 -norm minimization problems can converge to the sparsest solution under the following conditions [15]:

- The rows of the sampling matrix Φ must be incoherent to the columns of the

sparsifying basis Ψ .

- The number of measurements must be sufficient.

The coherence μ can be computed using:

$$\mu = \sqrt{N} \max(|\Phi \Psi|), \quad (2.3)$$

and its range is defined to be [15]:

$$1 \leq \mu \leq \sqrt{N}, \quad (2.4)$$

while the minimum number of measurements for recovery is approximated by:

$$M_{\min} \approx cK \ln \left(\frac{N}{K} \right), \quad (2.5)$$

where K is the number of non-zero coefficients in the signal $\mathbf{s}$, and c is a constant that depends on the incoherence between the sampling matrix and the sparsifying basis.

As such, a lower coherence coefficient leads to a smaller $M_{\min}$, which is desirable.

2.2 Sampling Matrices

Compressed sensing can reconstruct signals sampled below the Nyquist-Shannon rate by adopting a non-uniform sampling scheme [12]. This is reflected in the sampling matrix Φ . Such non-uniformity can easily be incorporated into the sampling matrix, however, the challenging part lies in ensuring that the rows of the sampling matrix are

incoherent to the columns of the sparsifying basis as this will decrease the minimum number of measurements needed for recovery. Fortunately, many families of conventional sampling matrices have been constructed that have sufficient incoherence to most sparsifying bases [15]. This includes random single pixel, Bernoulli, sparse random, and Random Gaussian matrices [15]. The latter in particular is a commonly used conventional sampling matrix. It also sees use in DCSNs and DUNs alike. Examples include ReconNet [2], ISTA-Net [4], and MADUN [19].

Some works in the literature have focused on the compression aspect of compressed sensing. Specifically, they argued that taking higher quality measurements $\mathbf{y}$ can lead to easier and better reconstruction quality. For that purpose, they proposed that a sampling matrix can be learned to enhance the quality of the measurements (see [17], [38], [25], [20]). Their results showcased superior performance when compared to conventional sampling matrices when applied to either conventional CS methods or DCSNs.

2.3 Initialization

Solving problems similar to Equation (1.1) is time-intensive [38]. However, the process of finding a solution can be accelerated by providing an initial solution. In DCSNs, the most popular method for initialization is to use the least-squares (LS) method, which can be done in two approaches. If the initialization is done independently of

the data, then the LS initialization is computed by:

$$\arg \min_{\mathbf{x}} \frac{1}{2} \|\Phi \mathbf{x} - \mathbf{y}\|_2^2, \quad (2.6)$$

$$\hat{\mathbf{x}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}, \quad (2.7)$$

where $\hat{\mathbf{x}}^{(0)} \in \mathbb{R}^N$ is the initial solution. When the rows of Φ are orthonormal, the LS becomes simpler to compute [3] and can be obtained by:

$$\hat{\mathbf{x}} = \Phi^T \mathbf{y}, \quad (2.8)$$

This approach can be used when there is no access to a training dataset. If such access is available, then the LS solution can be initialized by the following [4]:

$$\arg \min_{\mathbf{Q}} \frac{1}{2} \|\mathbf{X} - \mathbf{Q}\mathbf{Y}\|_2^2, \quad (2.9)$$

$$\mathbf{Q} = \mathbf{X}\mathbf{Y}^T(\mathbf{Y}\mathbf{Y}^T)^{-1}, \quad (2.10)$$

$$\hat{\mathbf{x}} = \mathbf{Q}\mathbf{y}, \quad (2.11)$$

where $\mathbf{Q} \in \mathbb{R}^{N \times M}$ is the linear estimator, $\mathbf{X} \in \mathbb{R}^{N \times k}$ is the horizontal concatenation of the uncompressed signals, $\mathbf{Y} \in \mathbb{R}^{M \times k}$ is the horizontal concatenation of their respective *measurements*, and k is the number of samples in the dataset. Note that the samples in both $\mathbf{X}$ and its respective $\mathbf{Y}$ are limited to the training dataset. Also, both Φ and $\mathbf{Q}$ can be learned for better performance [17].

Although Equation (2.8) should only be used when Φ has orthonormal rows, it

has been shown that it can still be used with learned sampling matrices. OPINE-Net [25] shows that learned sampling matrices can have orthonormal rows by adding an orthogonality penalty to the cost function. Interestingly, CASNet [20] shows that no such penalties are needed and that learned sampling matrices will inherently have orthonormal rows when Equation (2.8) is used as the initial solution and GDM is used throughout the network.

While initialization, in general, is widely used, in DCSNs, some networks forego the use of an initial solution altogether such as CSformer [18]. It reconstructs the image directly from the measurements without the need for an initial solution.

2.4 Iterative Optimization Based Methods

In compressed sensing, the recovered signal should ideally have the minimum l_0 -norm in the sparse domain. However, due to the difficulty of minimizing the l_0 -norm, others have proposed minimizing the l_1 -norm instead, which can converge to the minimum l_0 -norm solution [12]. Some works that use l_1 -norm minimization include orthogonal matching pursuit (OMP) [14], least absolute shrinkage and selector operator (LASSO) [16], and iterative shrinkage-thresholding algorithm (ISTA) [13]. ISTA in particular is noteworthy when it comes to the discussion of deep unfolding networks as it has a significant influence on the design of DUNs (including this work). ISTA adapts the CS problem to a LASSO optimization problem, which is defined as follows:

$$\min_{\mathbf{s}} \|\Phi\Psi\mathbf{s} - \mathbf{y}\|_2^2 + \|\mathbf{s}\|_1. \quad (2.12)$$

Then, in each iteration, ISTA performs the following two steps to solve the LASSO optimization problem:

$$\mathbf{s}^{(k)} = \text{sign}(\mathbf{s}^{(k-1)}) \text{ReLU}(|\mathbf{s}^{(k-1)}| - \lambda), \quad (2.13)$$

$$\mathbf{x}^{(k)} = \hat{\mathbf{x}}^{(k-1)} - \rho \Phi^T (\Phi \hat{\mathbf{x}}^{(k-1)} - \mathbf{y}), \quad (2.14)$$

where k is the iteration index, $\hat{\mathbf{x}}$ is the reconstructed image, ρ is the step size, and λ is a shrinkage threshold parameter. Equation (2.13) performs shrinkage and thresholding to $\mathbf{s}$, which is done to sparsify it. However, this sparsification step will let $\hat{\mathbf{x}}$ deviate away from the measurements $\mathbf{y}$. In other words, it will increase the loss of the first term in Equation (2.12), which will be dubbed L_y . To counteract this, Equation (2.14) performs gradient descent, and is appropriately called gradient descent module (GDM). This GDM is performed to align $\hat{\mathbf{x}}$ with the measurements $\mathbf{y}$, which decreases the loss of the first term in Equation (2.12). When both steps are performed in each iteration, ISTA can converge to a sparse solution that also agrees with the measurements.

Other works have proposed minimizing the total variation (TV) instead [39], [40]. Rather than assuming that images can be sparse in some domains, TV utilizes the fact that the *gradient* of images is already sparse. Compared to l_1 -norm minimization, images reconstructed using TV minimization are often sharper [40]. However, it is more computationally expensive than l_1 -norm minimization. The TV minimization

problem is modeled as follows:

$$\min_{\hat{\mathbf{x}}} \sum_i \|\mathbf{D}_i \hat{\mathbf{x}}\|_p \quad \text{subject to} \quad \mathbf{y} = \Phi \hat{\mathbf{x}}, \quad (2.15)$$

where $\mathbf{D}_i \hat{\mathbf{x}}$ is the discrete gradient of the image at pixel i , and $p = 1$ or 2 . TV minimization methods include TVAL3 [40] and denoising-based approximate message passing (D-AMP) [39].

In general, conventional approaches have made significant progress to enhance their computational efficiency. However, their computational complexity still remains high as a result of their iterative calculations.

2.5 Deep Compressed Sensing Networks

Due to the success of deep learning (DL) networks in general image restoration tasks (e.g., [41], [42], [43], [44], [45]), they were eventually proposed for compressed sensing problems. Mousavi *et al.* [46] propose a stacked denoising autoencoder (SDA) for both sampling and reconstruction using multi-layer perceptrons (MLPs). ReconNet [2] on the other hand samples the images using a random Gaussian matrix. For reconstruction, however, it utilizes convolutional neural networks (CNNs), which are then followed by a conventional denoiser (BM3D [47]). CSNet [17] is a model that can learn the sampling matrix. Notably, it simplified the sampling operation by modeling it using convolutional layers. Furthermore, Shi *et al.* [38] demonstrated that learned sampling matrices can also enhance the performance of iterative optimization-based CS

methods. The authors also proposed limiting the learned sampling matrices to binary or bipolar numbers for easy storage and hardware implementation [38]. Uformer-ICS [48] proposes an encoder-decoder transformer for the reconstruction network. CS-former [18] combines CNNs and transformers in the reconstruction network and it also reconstructs the image directly from its measurements.

Despite the advances deep learning networks have made in the sampling stage of CS, many of them can only reconstruct images from a single CS ratio (see [17], [25], [38], [2], [4]). Such networks need to be retrained for each specific CS ratio. However, recent works have proposed flexible networks that can reconstruct images from multiple CS ratios using a single network. For example, SCSNet [49] progressively samples the image. Specifically, the network will sample the image in the first layer to obtain the base measurements. Then, each subsequent layer will sample more unseen measurements until it achieves the specified CS ratio. A similar approach was also proposed by [50] where the network will progressively sample and reconstruct the image until it reaches the desired CS ratio. CASNet [20] tackles this problem by learning a single but large sampling matrix, and then, it truncates its rows based on the currently chosen CS ratio. While all the previous methods allow the network to reconstruct images from multiple CS ratios, they all require an initial reconstruction. To the best of our knowledge, no network has been proposed that can both reconstruct the image directly from its compressed measurements and have a variable CS ratio. Fortunately, the work in this thesis is not concerned with the sampling stage as the current progress is sufficient for our work.

2.6 Deep Unfolding Networks

Deep unfolding networks also called deep unrolling networks are generally designed after iterative conventional methods and are not limited to CS. They map a single iteration of some conventional method to a recovery block. Then, they unfold multiple iterations of said method to the network by using multiple recovery blocks sequentially where each block corresponds to a single iteration. Since the recovery blocks of DUNs are designed after conventional methods, DUNs enjoy higher interpretability than other deep learning networks. While DUNs have been proposed for various image-related tasks such as image super-resolution [51], [52], low-light image enhancement [53], and image restoration [54], they have also been proposed for CS [4], [3]. Since this work purely tackles CS, it can be safely assumed that when DUNs are discussed, we are referring to DUNs in CS.

Deep unfolding networks for CS acknowledge the rich CS literature and its associated conventional methods and aim to amplify them by utilizing deep learning networks. They take an iterative traditional method (such as ISTA [13]) and then *unfold* a fixed number of its iterations into several deep learning blocks, where each iteration is mapped to a single recovery block. This allows DUNs to preserve some of the interpretability of the conventional methods while boosting their performance. Compared to DCSNs, DUNs have better interpretability while having similar or even higher performance, however, their iterative design usually makes them slower than DCSNs.

ISTA-Net [4] aimed to adapt ISTA [13] to a deep unfolding network. It did this

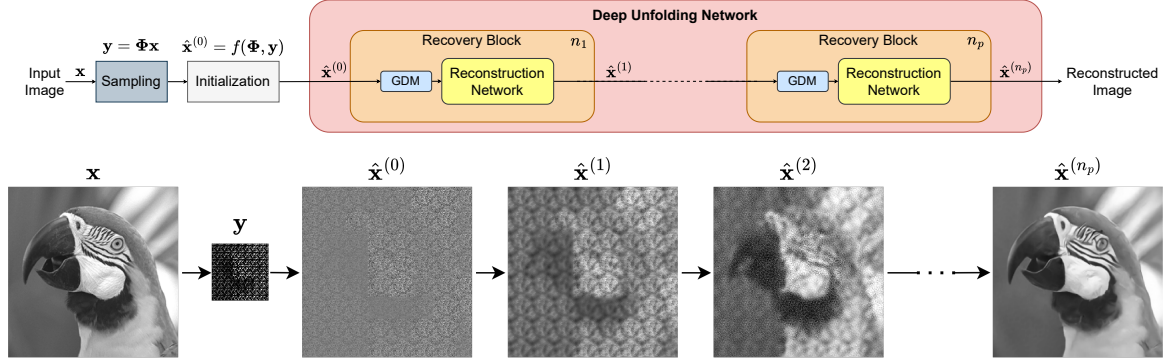


Figure 2.2: General structure of many deep unfolding networks.

by adapting both Equation (2.13) and Equation (2.14) to a deep network. First, the GDM was modified by making the step size ρ in Equation (2.14) a learnable parameter, and it was initialized to have a value of 0.50. As for sparsification, ISTA-Net uses a convolutional neural network to learn an advanced basis that sparsifies the image in the feature space. Then, it performs the shrinkage and thresholding step in the feature space rather than the spatial domain. Finally, ISTA-Net invokes certain conditions on the convolutional neural network so that it finds a proper sparsifying basis such that it can be reversed. By adapting both Equation (2.13) and Equation (2.14), ISTA-Net achieved state-of-the-art results at the time of its publication.

Following the success of ISTA-Net [4], many DUNs have been designed using a similar framework. A general schematic of the framework is seen in Figure 2.2. OPINE-Net [25] proposed the use of a learned sampling matrix. They also incorporated constraints to the sampling matrix to ensure that it can be both orthogonal and binary. MADUN [19] proposed memory augmentation that allows the network to share features between recovery blocks. COAST [3] proposed the random projection augmentation (RPA) training strategy that allows it to reconstruct data from arbitrary CS ratios.

This was achieved by using a least-squares initialization stage to fix the dimensions of the input, and also, by using multiple sampling matrices at different CS ratios to generalize COAST to arbitrary CS ratios.

CASNet [20] is another notable DUN that follows the framework set by ISTA-Net [4]. CASNet proposed the use of adaptive sampling for recovery. It argued that some image blocks are easier to recover than others. Therefore, less number of measurements should be allocated by them, while more measurements should be allocated to challenging image blocks. The CS ratio allocation strategy is determined by a saliency map, which is produced by a lightweight network. Furthermore, while CASNet uses a single learned sampling matrix, it is still able to reconstruct images from arbitrary CS ratios. This is achieved by learning a single sampling matrix with dimensions $N \times N$ and then truncating its rows depending on the chosen CS ratio. Finally, CASNet uses an encoder-decoder network for the reconstruction network, and it also concatenates the saliency map depthwise with the input image for enhanced recovery.

Despite the success of those DUNs, the main general framework depicted in Figure 2.2 has rarely seen any changes. The main change that occurred to it is that recent networks no longer add constraints to ensure that they find a reversible sparsifying basis [3], [19], [20]. As such, it can be seen that recent DUNs replace the thresholding and shrinkage step with a reconstruction step instead. Other than this, the framework is used as is.

2.7 Recursive Networks

There have been attempts to utilize recursions to decrease the overall size of networks. Denoising prior driven deep neural network (DPDNN) [55] is one such network. Its recovery block consists of an encoder-decoder network, and by default, it stacks 6 recovery blocks sequentially to reconstruct the image. Unlike COAST [3], all recovery blocks share the same learnable parameters. In other words, the network can be described as having a single recovery block with 6 iterations per recovery block. Other networks that have experimented with recursions include ISTA-Net [4] and CASNet [20]. The authors of those networks demonstrated how they can share the learnable parameters over all recovery blocks and still maintain competent performance. Those findings show that recursions can successfully be applied to decrease the total number of learnable parameters.

However, the implementation of recursions in the previous methods was lacking. Mainly, the approach limited the networks to a single distinct recovery block. This restricted the performance of recursions due to diminishing returns. In addition, the previous approaches did not utilize the flexibility of recursions with respect to computational complexity. Specifically, they always used the maximum number of recursions during training, which can unnecessarily increase the training time. Finally, the features of the previous methods were fixed regardless of recursion statistics such as the total number of iterations and the current iteration index. Since different features should be emphasized depending on the recursions statistics, this fixed structure prevents efficient use of recursions.

2.8 Gradient Descent Module

Although GDM has rarely been discussed directly in the literature, there have been other works that have contributed to enhancing it or minimizing the measurements loss L_y . MR-CCSnet [56] has proposed a measurements-reuse block that feeds the measurements to the network at regular intervals and different scales. Although MR-CCSNet does not use GDM, its measurements-reuse block theoretically allows it to minimize L_y . As such, the network can minimize both terms of the optimization objective in Equation (2.12), and they have shown how this leads to better performance. However, their method suffers from two problems. First, the measurements reuse block is designed specifically for their network. Thus, it cannot be easily applied to other networks. Secondly, its reliance on the measurements means a single network cannot reconstruct data from multiple CS ratios. DUN-CSNet [26] also acknowledges the literature’s lack of attention to the GDM. They argue that restricting the step size of ρ in Equation (2.14) to a single scalar is not ideal. As such, they proposed a content-adaptive gradient descent network that finds the value of ρ for each pixel in the image instead. Although this work does indeed focus directly on the GDM, it still isolates it from the reconstruction network. Therefore, the reconstruction network can produce results that are not consistent with the measurements, which contradicts Equation (2.12) and can lead to lower performance.

Finally, to conclude the literature survey section, we summarize the recent work on CS in Table 2.1. It includes a list of CS algorithms, their sampling method, category, number of recovery blocks, and their major contributions. It should be noted that

Algorithm	Sampling	Category	Recovery Blocks	Major Contribution
OMP [14]	Conventional	Iterative	N/A	Fast L_1 -norm minimization
TVAL3 [40]	Conventional	Iterative	N/A	TV minimization
D-AMP [39]	Conventional	Iterative	N/A	Integrating denoisers into CS
SDA [46]	Conventional or Non-linear	DCSN	N/A	First use of DL in CS
ReconNet [2]	Conventional	DCSN	N/A	Non-iterative recovery
ISTA-Net [2]	Conventional	DUN	9	Adapting ISTA to DL
SCSNet [49]	Learned	DCSN	N/A	Recovery from multiple CS ratios
CSNet [38]	Learned	DCSN	N/A	Hardware implementation of a learned sampling matrix
OPINE-Net [25]	Learned	DUN	9	Learned sampling matrix with orthonormal rows
MADUN [19]	Conventional	DUN	25	Transmitting features between recovery blocks
COAST [3]	Conventional	DUN	20	Feature modulation based on CS ratio and noise level
MR-CCSNet [56]	Learned	DCSN	N/A	Measurements-reuse block
CASNet [20]	Learned	DUN	13	Augmenting recovery blocks with saliency maps
Uformer-ICS [48]	Learned	DCSN	N/A	Adaptive sampling
CSformer [18]	Learned	DCSN	N/A	Direct recovery from the measurements
DUN-CSNet [26]	Learned	DUN	15	Gradient descent network

Table 2.1: Summary of the Literature Review

conventional sampling refers to the sampling matrices discussed in section 2.2 such as random Gaussian matrices.

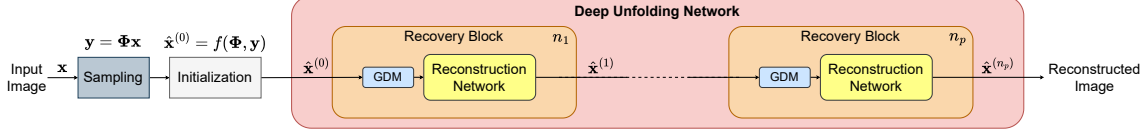
CHAPTER 3

RECURSIVE DEEP UNFOLDING NETWORKS

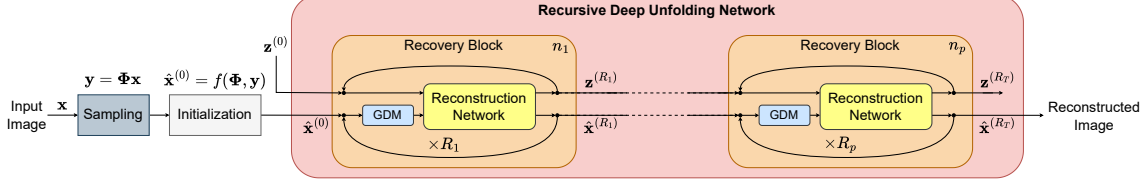
3.1 Introduction

Recently, DUNs have been used to solve CS problems. DUNs in CS are designed after a conventional CS algorithm such as ISTA [13]. They map a single iteration of one such algorithm to a recovery block. Then, they unfold a fixed number of iterations of that algorithm by using multiple sequential recovery blocks, which form the deep unfolding network. All recovery blocks share the same structure, yet they consist of distinct learnable parameters. The results in the literature show that generally, increasing the number of recovery blocks leads to improved performance [13], [20].

In this chapter, we argue that the reason for this increase in performance is not only due to the increase in the number of recovery blocks but also due to the overall increase in the total number of iterations. As such, we propose the recursive framework to enhance the efficiency of DUNs in CS. The framework performs multiple



(a) General structure of most deep unfolding networks.



(b) The proposed recursive framework where $z^{(\cdot)}$ carries parameters to the proposed RFMU unit that resides within the reconstruction network.

Figure 3.1: A comparison between conventional deep unfolding networks and our recursive framework.

iterations over each recovery block as seen in Figure 3.1b. This allows the network to use less number of recovery blocks while maintaining the same total number of iterations. Thus, decreasing the number of learnable parameters. However, a naive implementation of recursions is not optimal. Recursive networks should extract different features depending on their current iteration index. To that end, we propose the recursion-based feature modulation unit (RFMU). It modulates the features of the network based on recursion statistics, which are the current iteration index and the total number of iterations. This inclusion of the RFMU allows networks to compensate for their lower number of learnable parameters and achieve similar performance to their non-recursive counterpart. Finally, we randomize the number of recursions only during training to decrease the training time.

This chapter is structured as follows. First is the methodology section, which discusses how to apply the recursive framework with its associated RFMU unit to COAST [3], ISTA-Net+ [4], and CASNet [20]. This is followed by the results section where we analyze the results of all networks both when they have access to the entire

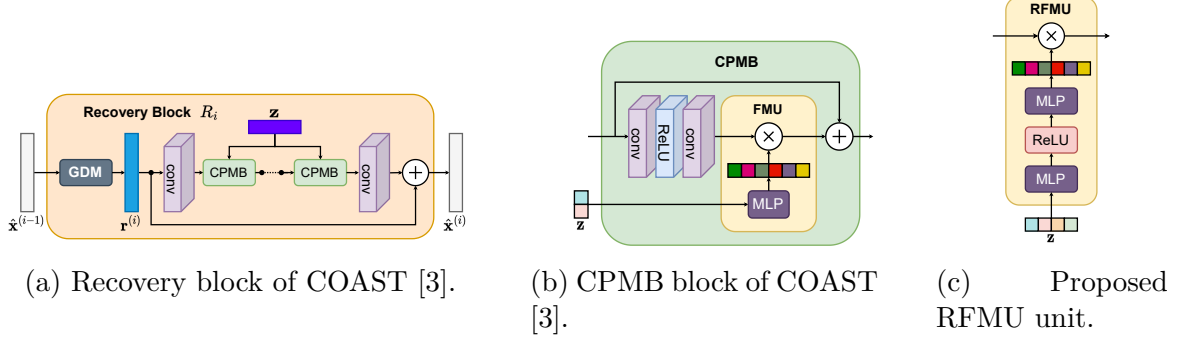


Figure 3.2: Illustration of the design of the recovery block of COAST [3] and also our proposed RFMU unit.

training dataset, and when their access is limited. This is done to test their resilience to overfitting. Then we move on to the ablation study where we show the results of the recursive framework before and after the addition of the RFMU. Finally, we conclude our findings in the conclusion section.

3.2 Methodology

To illustrate how a deep unfolding network can be adapted to our recursive framework, we will apply it to COAST [3]. With that said, the general process can be applied to other DUNs as well. The overall structure of COAST [3] will be discussed while the finer details can be found in the original work [3].

3.2.1 COAST

COAST [3] adheres to the general structure illustrated in Figure 2.2. First, the sampling is performed by random Gaussian matrices where the rows of said matrix are

orthogonal to each other. Next, an initial solution is obtained by:

$$\hat{\mathbf{x}}^{(0)} = \Phi^T \mathbf{y} \quad (3.1)$$

Then, the image is passed to the recovery network.

The fundamental recovery block of COAST is illustrated in Figure 3.2a. First, the image is fed to a gradient descent module (GDM) to align the prediction $\hat{\mathbf{x}}^{(k)}$ with the measurements $\mathbf{y}$ such that the difference between $\hat{\mathbf{x}}^{(k)}$ and $\mathbf{y}$ remains minimal. It is computed by:

$$\mathbf{r}^{(k)} = \hat{\mathbf{x}}^{(k-1)} - \rho^{(k)T} (\Phi \hat{\mathbf{x}}^{(k-1)} - \mathbf{y}) \quad (3.2)$$

where $\rho^{(k)}$ is a learnable step size, and i is the index of the recovery block. Next, the image is fed to a convolutional layer to project it to the feature space where the number of channels is equal to N_f . Then, N_c residual blocks are used to extract further features at a deeper level. Those residual blocks are called controllable proximal mapping blocks (CPMBs) by their authors [3] and are depicted in Figure 3.2b. The controllability of the CPMBs stems from their ability to modulate the magnitudes of each channel based on both the CS ratio and the noise level. This is carried out by the feature modulation unit (FMU) using an MLP layer. Finally, a convolutional layer is used to project the features from the feature space back to the spatial domain.

3.2.2 Recursions

Generally, applying the recursive framework to deep unfolding networks such as COAST [3] is straightforward. For each recovery block, a recursive connection is added that feeds the output back to the input for R iterations per recovery block (IPRB). After this addition, COAST will now resemble the structure found in Figure 3.1b.

Furthermore, we utilize the flexibility of recursions to decrease the training time. Specifically, the number of iterations performed in a recursive network can be decreased as needed. This is utilized to decrease the training time. Instead of using a fixed number of iterations, we randomize it only during training, which decreases the training time and allows the network to adapt to varying IPRB. For example, when ignoring the RFMU, a network with 5 recovery blocks and 3 IPRB will have the same FLOPS as a network that has 15 recovery blocks but 1 IPRB during test time. However, during training, the number of iterations of the former network will vary from 5 to 15 due to randomized recursions, which will average out at 10 iterations. On the other hand, the number of iterations of the latter network will be fixed to 15. As such, even though both networks will perform a maximum of 15 iterations, the recursive network will use around 33% less FLOPs during training because it randomizes the number of recursions it performs.

3.2.3 Recursion-based Feature Modulation Unit

Although relying only on recursions showed promise in improving the framework’s efficiency, it was not the most effective strategy. This is due to the inability of each recovery block to predict the quality level of its input, thus preventing optimal enhancement of its features. Building on the concept of feature modulation presented in COAST [3], we expand upon the feature modulation unit introduced in the CPMB by integrating recursion statistics into the vector $\mathbf{z}$. Specifically, our proposed RFMU unit adjusts the network’s features based on two additional parameters, which are the current iteration index k and the total number of iterations denoted as K . To achieve this, we employ two linear layers, separated by ReLU [57], which take the input vector $\mathbf{z}$ along with our extra two parameters to derive N_f modulation coefficients (referred to as σ_c). Therefore, each feature in the output of the CPMB block is modulated by its corresponding modulation coefficient σ_c . Figure 3.2c shows an illustration of the RFMU unit.

Originally, ISTA-Net+ [4] does not incorporate a feature modulation unit. However, integrating the RFMU into it is relatively straightforward, which can be seen in Figure 3.3. The sole alteration made to the network involves inserting the RFMU after the initial convolutional layer and another RFMU preceding the final convolutional layer. Apart from these adjustments and the incorporation of recursions, the recursive ISTA-Net+ model remains unchanged compared to its reference design in [4].

As for CASNet [20], it is an encoder-decoder network that makes heavy use of

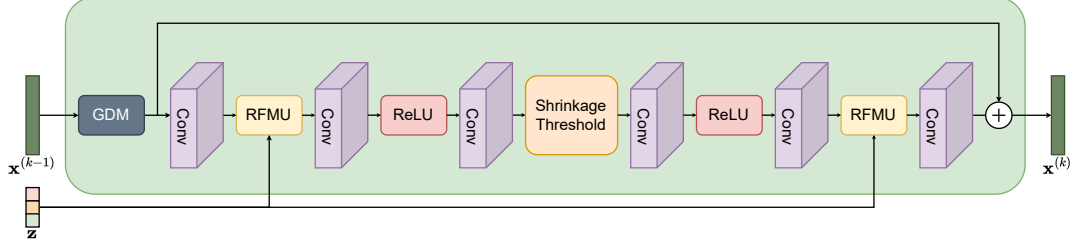


Figure 3.3: Recovery block of ISTA-Net+ [4] after the addition of the RFMU unit.

residual blocks. The design of said residual blocks bears similarity to the CPMB block used by COAST [3] shown in Figure 3.2b without the FMU. It uses two convolutional layers separated by ReLU [57] with a residual connection that carries the input to the output. In this work, the RFMU was added to the residual blocks of CASNet just before the addition of the residual connection. Its structure becomes similar to Figure 3.2b, but it uses the RFMU instead of the FMU.

3.3 Experimental Results

To investigate the effectiveness and generalizability of the recursive framework, we apply it to three different networks. Those networks are ISTA-Net+ [4], COAST [3], and CASNet [20]. For a fair comparison, all models were trained using a similar training strategy to what was used in their respective papers, including the training datasets. ISTA-Net+ and COAST were trained using CS ratios from 10% up to 50%, but CASNet was trained using CS ratios from 1% to 100% as this was used by its reference [20]. For ISTA-Net and COAST, both used a random Gaussian sampling matrix, while CASNet used a learned sampling matrix. For COAST in particular, we faithfully followed their RPA training strategy, which trains COAST using multiple

sampling matrices. The mean-square error (MSE) was used as the cost function. ADAM [58] was used as the optimizer of choice with momentum and weight decay set to 0.9 and 0.999 respectively. Its learning rate is fixed throughout the training at 1×10^{-4} for both ISTA-Net+ and COAST, but for CASNet, it was divided by 10 for the last 20 epochs. Both ISTA-Net+ and COAST were trained for a total of 400 epochs. CASNet on the other hand was trained for 320 epochs, which is the number of epochs used in the original work. All model implementations were executed using PyTorch [59], and training was conducted on a single Nvidia RTX 3090 GPU. Validation was performed on the Set11 dataset [2], while testing utilized the BSD68 dataset [1]. Evaluation metrics employed include the PSNR and the SSIM [60].

3.3.1 COAST vs. Recursive COAST

To showcase the efficacy of the recursive framework, it was applied to COAST [3]. The first model, labeled COAST, retains the same configuration outlined in [3], which features 20 non-recursive recovery blocks. The second model, denoted as R-COAST, comprises 5 recursive recovery blocks at 4 IPRB. The results are shown in Table 3.1 and Table 3.2.

Table 3.1: PSNR/SSIM scores of the models on the BSD68 [1] dataset.

Training Data %	Model	Layers	IPRB	CS Ratio					Avg. Score	Parameters
				10%	20%	30%	40%	50%		
100%	ISTA-Net+ [4]	9	1	25.28/0.7003	28.08/0.8152	30.20/0.8769	32.06/0.9156	33.94/0.9423	29.91/0.8501	336,978
	RISTA-Net+	3	3	25.42/0.7074	28.20/0.8189	30.36/0.8802	32.26/0.9177	34.09/0.9434	30.07/0.8535	114,630
	COAST [3]	20	1	26.23/0.7379	28.90/0.8372	30.97/0.8910	32.81/0.9244	34.58/0.9475	30.70/0.8676	1,122,024
	R-COAST	5	4	26.12/0.7325	28.81/0.8339	30.89/0.8892	32.74/0.9237	34.52/0.9468	30.62/0.8652	298,250
	CASNet [20]	12	1	28.09/0.8156	30.80/0.8933	32.98/0.9322	35.00/0.9558	36.99/0.9710	32.77/0.9136	15,680,080
	R-CASNet	4	3	28.12/0.8176	30.85/0.8949	33.02/0.9333	35.05/0.9565	37.02/0.9715	32.81/0.9147	6,194,160
3%	ISTA-Net+ [4]	9	1	24.28/0.6614	27.06/0.7823	29.04/0.8498	30.87/0.8960	32.54/0.9264	28.76/0.8232	336,978
	RISTA-Net+	3	3	24.71/0.6799	27.57/0.8017	29.65/0.8659	31.49/0.9066	33.20/0.9341	29.32/0.8376	114,630
	COAST [3]	20	1	26.00/0.7294	28.64/0.8309	30.68/0.8867	32.49/0.9213	34.22/0.9446	30.40/0.8626	1,122,024
	R-COAST	5	4	26.06/0.7321	28.74/0.8334	30.79/0.8883	32.61/0.9228	34.37/0.9468	30.51/0.8647	298,250
	CASNet [20]	12	1	27.25/0.7944	29.73/0.8755	31.74/0.9179	33.60/0.9446	35.43/0.9624	31.55/0.8990	15,680,080
	R-CASNet	4	3	27.31/0.7963	29.92/0.8782	32.00/0.9205	33.91/0.9467	35.77/0.9640	31.78/0.9011	6,194,160

The PSNR/SSIM scores for the BSD68 dataset are illustrated in Table 3.1. When both models are trained on the complete training dataset, we observe that despite the recursive model utilizing only 27% of the parameters of the baseline, it still achieves competitive results in terms of both PSNR and SSIM metrics. However, upon limiting the training dataset to only 3% of its original size, we find that the recursive network outperforms the baseline thanks to its smaller size.

Table 3.2: Training time and number of parameters of the models in Table 3.1.

Model	Configuration	Layers	IPRB	Training Time (Hours)	Inference Time (FPS)	Parameters
ISTA-Net+ [4]	Baseline	9	1	4.52	46.34	336,978
	Recursive	3	3	3.60	43.32	114,630
COAST [3]	Baseline	20	1	13.71	19.12	1,122,024
	Recursive	5	4	9.00	18.89	298,250
CASNet [20]	Baseline	12	1	42.85	16.80	15,680,080
	Recursive	4	3	33.85	13.31	6,194,160

Furthermore, while the recursive COAST model experiences minimal impact on its performance, as indicated in Table 3.2, it demonstrates a 35% reduction in training time and a 73% decrease in learnable parameters, with negligible effects on inference time. Consequently, we can infer that the recursive framework enhances the efficiency of the COAST model with minimal compromise on its performance.

3.3.2 ISTA-Net+ vs Recursive ISTA-Net+

To assess the generalization ability of our recursive framework, we conduct a comparison on the ISTA-Net+ model [4] with and without recursion. These are labeled RISTA-Net+ and ISTA-Net+ respectively. The non-recursive network adheres to the

configuration outlined in [4], which features 9 recovery blocks without recursions. The recursive model comprises 3 recovery blocks and 3 IPRB. The results of this comparison are presented in Table 3.1.

After evaluating the results on the testing dataset, we discover that, on average, the recursive configuration surpasses the baseline model’s performance while employing 66% fewer parameters. Notably, when the training dataset is restricted to only 3% of its original size, the recursive configuration further solidifies its advantage over the baseline. This can be attributed to its inherently reduced parameter count, thus making it more resistant to overfitting.

Table 3.2 presents the training and inference times of the networks. Notably, the recursive configuration exhibits a 20% reduction in training time, attributed to the utilization of randomized recursions during training. However, the inference time sees a small impact due to the extra computations performed by the RFMU.

Overall, applying the recursive framework to ISTA-Net+ [4] results in a reduction in training time and parameter count while improving the performance at the same time when the training dataset is large. However, when the dataset is limited, the recursive model widens its gap over the baseline.

3.3.3 CASNet vs Recursive CASNet

The last network used to verify the generalizability of the recursive framework is CASNet [20]. The non-recursive network will use 12 recovery blocks and will be dubbed CASNet while the recursive network uses 4 recovery blocks and 3 IPRB and will be dubbed R-CASNet.

Table 3.1 shows the results of CASNet. In general, CASNet shows a similar pattern to ISTA-Net+, where the performance of the recursive network actually outperforms the results of the non-recursive network. This is the case under all CS ratios, and the gap only grows larger as the size of the training dataset decreases.

Finally, in Table 3.2, we observe that the R-CANet enjoys a reduction in training time by 21% and its number of parameters is reduced by 60% with no degradation to its performance. The only downside is that its inference time has decreased by 21%, which is due to the extra computations of the RFMU, as it uses it more frequently than the other networks. Overall, the recursive framework has proved to be effective for the CASNet network.

3.4 Ablation Study

3.4.1 RFMU Enhances the Efficiency of Recursions

One challenge in the basic implementation of recursions is the significant variation in reconstruction quality between each iteration, particularly noticeable in the early stages. Specifically, the model cannot roughly predict the quality level of its input as it lacks information about how many iterations its input has gone through already. Therefore, it cannot adjust its features accordingly. Essentially, the weights applied in the early iterations should differ from those in the last iterations. In this work, we proposed that the model’s performance can be enhanced by letting it modulate its features based on recursion statistics. To verify the validity of this approach, we conducted a comparative analysis of the RFMU’s effectiveness in ISTA-Net+ [4],

COAST [3], and CASNet [20]. The results are detailed in Table 3.3.

Table 3.3: PSNR/SSIM performance comparison of the networks before and after the addition of the RFMU unit on the Set11 dataset [2].

Network	Layers	IPRB	RFMU	CS Ratio 10%
ISTA-Net+ [4]	3	3	✗	26.92/0.8155
RISTA-Net+	3	3	✓	27.17/0.8215
COAST [3]	5	4	✗	28.37/0.8522
R-COAST	5	4	✓	28.56/0.8559
CASNet [20]	4	3	✗	30.22/0.9004
R-CASNet	4	3	✓	30.45/0.9029

The results of COAST are detailed in Table 3.3. It shows the results of two COAST networks consisting of 5 recovery blocks at 4 IPRB, with one incorporating the RFMU (labeled R-COAST) while the other does not (labeled COAST). The results indicate that the R-COAST model outperforms its counterpart, achieving a higher PSNR score of 28.56 dB, whereas the COAST model without the RFMU attained 28.38 dB. This suggests that the inclusion of the RFMU does indeed enhance the model’s performance.

The results of the RFMU implementation in ISTA-Net+ are presented in Table 3.3. We trained two models with 3 recovery blocks at 3 IPRB, with one model omitting the RFMU (labeled ISTA-Net+) while the other incorporates it (labeled RISTA-Net+). Once more, as indicated in Table 3.3, we observe that RISTA-Net+ demonstrates superior performance compared to ISTA-Net+ due to its integration of the RFMU.

A similar pattern is observed in the results of CASNet in Table 3.3. Two CASNet networks were trained where one does not use the RFMU (dubbed CASNet) while the other one does (dubbed R-CASNet). Both of them used 4 recovery blocks at

3 IPRB. Again, we see that when the RFMU is incorporated into the network, it shows enhanced performance. This is the case here as R-CASNet exhibits superior performance compared to its counterpart.

Given the results in Table 3.3, we can conclude that the addition of the RFMU allows models to more efficiently utilize the recursive connection, which leads to better results.

3.4.2 Design of the RFMU

In subsection 3.4.1, we have discussed how the RFMU effectively enhances the performance of the recursive network. Here we will discuss the design of the RFMU unit. The main points that make the RFMU differ from the FMU used by COAST [3] are:

- Adding the current iteration index k and the total number of iterations K as parameters to the input $\mathbf{z}$, which are later used for feature modulation.
- The RFMU is non-linear as it uses two linear layers separated by ReLU [57].
- Using distinct RFMU units in each recovery block rather than sharing a single RFMU throughout the entire network.

To validate the design of the RFMU, we show the results when each item from the above is applied to COAST [3], which is shown in Table 3.4.

Table 3.4 shows each step done to the RFMU. First, we observe that by adding the current iteration index k and the total number of iterations K to the RFMU, we see improved results at all CS ratios, especially at 10% CS ratio. This shows that the network benefits from having access to some information about recursion statistics.

Table 3.4: PSNR/SSIM scores on the Set11 dataset [2] showing the impact of each step done to the RFMU.

Network	k & K	Non-linearity	Distinct Parameters	10%	20%	CS Ratio		40%	50%	Avg. Score
	$\times$	$\times$	$\times$	28.37/0.8522	32.16/0.9188	34.69/0.9460	36.67/0.9609	38.50/0.9715	34.08/0.9299	
COAST [3]	$\checkmark$	$\times$	$\times$	28.48/0.8546	32.21/0.9205	34.73/0.9474	36.77/0.9623	38.51/0.9725	34.14/0.9315	
	$\checkmark$	$\checkmark$	$\times$	28.48/0.8549	32.22/0.9204	34.79/ 0.9477	36.82/0.9626	38.64/0.9728	34.19/0.9317	
	$\checkmark$	$\checkmark$	$\checkmark$	28.56/0.8559	32.28/0.9209	34.82/0.9476	36.82/0.9624	38.61/0.9727	34.22/0.9319	

Secondly, the results show the performance of the RFMU after adding to it a second linear layer preceded by a ReLU activation function [57], which can be seen in Figure 3.2c. Here, we observe that while the scores didn’t improve for low CS ratios, the average score overall has increased by 0.05 dB. Finally, when distinct RFMU units are used in every recovery block, the results do improve at low CS ratios. For example, the network manages to improve its performance by 0.08 dB when the CS ratio is at 10%. Since this configuration provides the best overall results, it was chosen to be the final design of the RFMU unit.

3.5 Conclusion

This work explored the redundancy issues inherent in many deep unfolding networks due to their iterative design. We argued that these redundancies can largely be mitigated through the adoption of our recursive framework. To validate this claim, we applied our proposed framework to both COAST and ISTA-Net+. Our results affirm that the iterative nature of such networks indeed suffers from redundancies, and employing the recursive framework effectively alleviates a significant portion of these redundancies. Furthermore, by incorporating randomized IPRB during training, the recursive framework can reduce training time as well. In addition, we showed how the

performance of the recursive framework can be enhanced by using the RFMU, which modulates the features of the network based on recursion statistics.

We believe that the most significant part of the recursive framework is its simplicity. This allows it to be generalized to other networks since the core concept of the framework revolves around the recursive connection, which is relatively straightforward to implement in iterative networks. Moreover, we demonstrate the potential for further optimization through the incorporation of simple blocks such as our RFMU, which enhances the framework’s efficiency.

For future research, we aim to delve deeper into the recursive framework, exploring more intricate implementations. One possible approach would be to extend recursion to the feature space rather than the spatial domain, as this will allow it to be applied to networks other than deep unfolding networks, which could be appealing to generative networks due to their large size. Another possible route for improving the performance is to let the network itself distribute the IPRB. For example, instead of having an IPRB of 4 over 3 recovery blocks, the network might determine that it is better to have IPRBs of 2, 4, and 6 over the first, second, and third recovery blocks respectively. Therefore, a learned distribution of IPRBs might lead to better results as well.

CHAPTER 4

GRADIENT-INFORMED NETWORK

4.1 Introduction

Recently, DUNs have emerged as an attractive type of network within DCSNs. They owe their appeal to their closer connection to the handcrafted CS methods, as their central design revolves around them, thus, making them more interpretable. The most influential network in the field of DUNs is ISTA-Net [4], which is designed after ISTA [13].

ISTA formulates the CS problem as a LASSO cost function, which is represented by:

$$\min_{\hat{\mathbf{s}}} \frac{1}{2} \|\Phi \Psi \hat{\mathbf{s}} - \mathbf{y}\|_2^2 + \gamma \|\hat{\mathbf{s}}\|_1, \quad (4.1)$$

where $\hat{\mathbf{s}}$ is the reconstructed signal in the sparse domain, and γ is the weight of the sparsity constraint. The first term in Equation (4.1) only reaches zero when

the measurements obtained from the reconstructed signal $\hat{\mathbf{x}}$ exactly match the true measurements $\mathbf{y}$. This will be referred to as the measurement loss, or the loss in the measurements domain. By using Equation (2.1), it can be redefined as:

$$L_y = \frac{1}{2} \|\Phi \hat{\mathbf{x}} - \mathbf{y}\|_2^2, \quad (4.2)$$

where L_y is the measurements loss and $\hat{\mathbf{x}}$ is the reconstructed signal. The second term of Equation (4.1) reaches zero as $\hat{\mathbf{s}}$ approaches the zero vector in the sparse domain. As such, the loss in Equation (4.1) is minimized by finding the sparsest $\hat{\mathbf{s}}$ that agrees with the measurements, which directly follows the essential principles of CS theory.

ISTA solves this minimization problem by cycling between the next two update steps to $\hat{\mathbf{x}}$ and $\hat{\mathbf{s}}$.

$$\hat{\mathbf{x}}^{(k)} = \hat{\mathbf{x}}^{(k-1)} - \rho \frac{\partial L_y}{\partial \hat{\mathbf{x}}^{(k-1)}}, \quad (4.3)$$

$$\mathbf{s}^{(k)} = \text{sign}(\mathbf{s}^{(k-1)}) \text{ReLU}(|\mathbf{s}^{(k-1)}| - \lambda), \quad (4.4)$$

where k is the iteration index, ρ is the step size, λ is the shrinkage threshold parameter, and $\frac{\partial L_y}{\partial \hat{\mathbf{x}}^{(k-1)}}$ can be found by:

$$\frac{\partial L_y}{\partial \hat{\mathbf{x}}^{(k-1)}} = \Phi^T (\Phi \hat{\mathbf{x}}^{(k-1)} - \mathbf{y}). \quad (4.5)$$

Equation (4.3) uses gradient descent to align $\mathbf{x}^{(k)}$ with the measurements $\mathbf{y}$, which minimizes L_y , and this equation is also gradient descent module (GDM). However, this step could make $\hat{\mathbf{s}}$ less sparse. To counteract this, Equation (4.4) applies thresholding

to $\mathbf{s}$ to sparsify it and minimize the second term of Equation (4.1). By cycling between those two update steps, ISTA [13] can find a sparse solution to Equation (2.2) that is also consistent with the measurements.

ISTA-Net [4] adapted these two update steps of the ISTA algorithm [13] to a deep unfolding network. Specifically, they mapped Equation (4.4) to a neural network block by first learning a sparse representation of $\mathbf{x}$ in the feature space. Then, the thresholding step of Equation (4.4) was applied in the feature space as well. As for Equation (4.3), it only received a minor modification by making ρ^k a learnable parameter. A general schematic of its design is found in Figure 2.2.

ISTA-Net [4] shows that not only do they significantly outperform traditional handcrafted CS methods, but they also do so at a decreased computational complexity. Due to its impressive performance, ISTA-Net [4] has inspired many networks to follow its general ISTA structure, which helped them to achieve state-of-the-art results [25], [3], [19], [20]. However, most of the innovation of said networks lies strictly in improving Equation (4.4), while the GDM remains largely unchanged. Although one might be encouraged to argue that since the GDM in Equation (4.3) only consists of one learnable parameter, its impact on the network could be respectively small. In practice, however, this is not the case. Figure 4.1 shows the learning curves of two variants of COAST [3], which is a deep unfolding network. The only difference between the two variants is that one uses the GDM while the other one does not. As can be seen from Figure 4.1, the gap between the two networks is significant. This clearly demonstrates the essential role of the GDM in deep unfolding networks despite its negligible footprint in terms of learnable parameters. This raises the following ques-

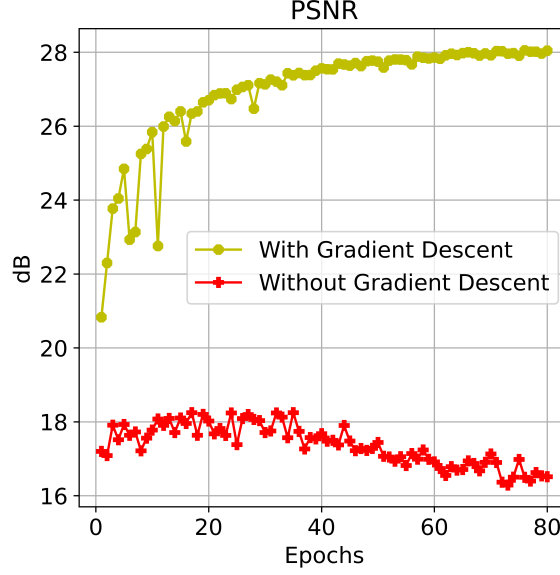


Figure 4.1: The learning curves for two COAST networks [3] on the validation dataset. One uses the GDM while the other does not.

tions:

- First, why does the GDM have such a large impact on the reconstruction performance?
- Secondly, can the GDM be redesigned to be better adapted to deep unfolding networks?

In this chapter, we aim to provide answers to the above questions. Specifically, the contributions of our proposed work are three-fold:

- First, we provide examples to show why the GDM is essential to DUNs in CS.
- Secondly, we propose gradient-informed networks (GINs), which is a simple yet effective method to adapt the GDM to DUNs. Specifically, we propose feeding the gradients in Equation (4.3) directly to the reconstruction network. This allows the network to perform Equation (4.3) and Equation (4.4) jointly. Thus,

satisfying both terms of the minimization objective of Equation (4.1) in only a single step.

- Finally, we demonstrate the generalizability of our proposed solution by implementing it in three networks, which are ISTA-Net+ [4], COAST [3], and CASNet [20].

The experimental results show that when the GDM is replaced with our proposed work, it manages to improve the results on all tested networks.

4.2 Methodology

In this section, we will discuss the motives behind our work. First, we will discuss the general aspects of DUNs such as the sampling and initialization stages. Then, we will discuss the general recovery block design in most DUNs. Finally, we will discuss the design behind GIN.

4.2.1 Sampling and Initialization

The general structure of deep unfolding networks is seen in Figure 2.2. First, the image is sampled using a sampling matrix. That sampling matrix can take many forms such as a random Gaussian sampling matrix, binary matrix, or it can even be learned for enhanced performance. Regardless of the form, the image will be sampled according to Equation (1.1). Then the image is usually initialized using the least-squares (LS). When the rows of the sampling matrix are orthonormal, then the LS

solution is obtained by:

$$\hat{\mathbf{x}}^{(0)} = \Phi^T \mathbf{y}. \quad (4.6)$$

Afterward, the image is fed to the recovery network.

4.2.2 GDM and Deep Reconstruction Network

Usually, deep unfolding networks in CS consist of multiple sequential recovery blocks. Those recovery blocks are made of two parts. The first is the GDM, which is simply the gradient descent step in Equation (4.3). It aims to minimize the discrepancy between the measurements obtained by $\hat{\mathbf{x}}^{(k)}$ and the true measurements $\mathbf{y}$. The second part is the deep reconstruction network, which is used to minimize the loss of the second term in Equation (4.1). Those two steps will be performed only once in each recovery block. However, since deep unfolding networks use those recovery blocks sequentially, the network will cycle between both update steps, thus, allowing it to minimize both terms in Equation (4.1).

4.2.3 Gradient-Informed Network

One problem with the current design of many deep unfolding networks is that minimizing L_y is completely isolated from the reconstruction network and is offloaded to the GDM instead. As such, the output of the reconstruction network can only minimize the second term of Equation (4.1), while it has to rely on the GDM to minimize L_y . We argue that such a design is bound to be suboptimal. Ideally, the reconstruction network should be able to jointly optimize both parts of Equation (4.1) to more

optimally realize the network’s capacity and become more aligned with CS theory.

To effectively minimize both terms of Equation (4.1), the network must be provided with two pieces of information. The first one is the current reconstructed output $\hat{\mathbf{x}}^{(k)}$, which is already done in most DUNs (e.g., [4], [61], [25], [62]). The second one is to provide the network with a path to minimize L_y . One approach to achieve this is to use $\hat{\mathbf{x}}$, Φ , and $\mathbf{y}$ as inputs to the network. The first two inputs $\hat{\mathbf{x}}$ and Φ will allow the network to estimate $\hat{\mathbf{y}}$ internally, while $\mathbf{y}$ allows it to determine how far does $\hat{\mathbf{y}}$ deviates from the true measurements $\mathbf{y}$. With all three inputs, the network should now be able to find a mapping to minimize L_y , and thus, jointly minimize both parts of Equation (4.1).

While the above is theoretically sound, in practice, it is difficult to implement. The network will have three different inputs with three different dimensions. Not only that but the dimensions of both Φ and $\mathbf{y}$ will change depending on the chosen CS ratio. As such, this will make it immensely difficult for a single network to reconstruct data from arbitrary CS ratios.

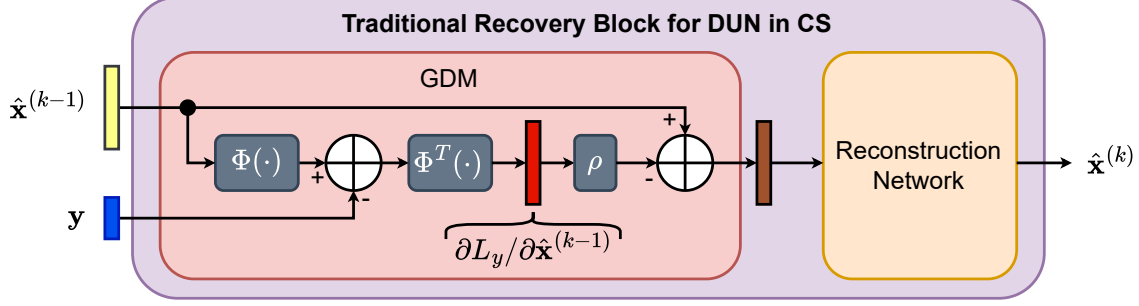
To overcome the aforementioned dimensionality problems, we propose replacing both Φ and $\mathbf{y}$ with the gradient of the loss instead, which is defined in Equation (4.5). Using the gradient in place of both Φ and $\mathbf{y}$ is superior due to its versatility. For one, its dimensions will always match the dimensions of $\hat{\mathbf{x}}^{(k)}$, thus, enabling a single network to reconstruct data from arbitrary CS ratios. Secondly, the gradient directly provides the network with both information about L_y and *how* to minimize it (done by following the gradient). Therefore, allowing the network to minimize L_y .

With this, our proposed work becomes quite simple. In each recovery block, we

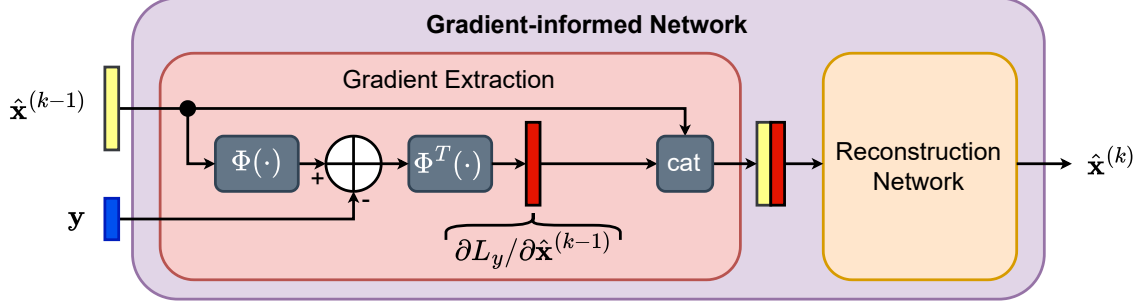
propose feeding the reconstruction network with both $\hat{\mathbf{x}}^{(k)}$ and $\partial L_y / \partial \hat{\mathbf{x}}^{(k)}$, where both are concatenated depth-wise. This allows the network to jointly minimize both terms of Equation (4.1) in a single step. We call this Gradient-informed Network (GIN). A general schematic comparing the recovery block before and after using GIN is shown in Figure 4.2, where cat is the concatenation operation. Furthermore, it can be proven that GDM is a special case of GIN. This can be shown by modeling Equation (4.3) using a convolution operation.

$$\mathbf{r}^{(k)} = \text{conv} \left[\text{cat} \left(\hat{\mathbf{x}}^{(k-1)}, \frac{\partial L_y}{\partial \hat{\mathbf{x}}^{(k-1)}} \right), \rho \right], \quad (4.7)$$

where $\mathbf{r}^{(k)}$ is the input to the reconstruction network, conv is the 2D-convolution operation, cat is the depth-wise concatenation operation, and ρ is the kernel. When the number of kernels is one, and its size is 1×1 , then Equation (4.7) can be reduced to the traditional GDM operation. Our GIN on the other hand uses a regular kernel size such as 3×3 , but more crucially, it uses multiple kernels to extract features from both the image and the gradient. This lets GIN extract features from both the input image and its gradient, and then propagate said features throughout the network. By propagating features extracted from the gradient, the network can now jointly satisfy both terms in Equation (4.1) in a singular step. Finally, we note that instead of adding an extra convolutional layer, GIN uses the first convolutional layer of the network. This is done purely to limit the increase in the number of parameters.



(a) Recovery block of DUN when GDM is used.



(b) Recovery block of DUN when our proposed GIN is used where cat denotes the concatenation operation.

Figure 4.2: Recovery block schematic before and after using GIN.

4.3 Experimental Results

To investigate the effectiveness of our proposed GIN. We apply it to three networks. Those networks are ISTA-Net+ [4], COAST [3], and CASNet [20]. For all three networks, GIN will be applied to them while their GDM will be removed. This will test if GIN can truly replace the GDM operation.

All networks underwent training utilizing the same training dataset used in their initial publications. Moreover, the training process for all networks incorporated the mean-squared error (MSE) as their designated cost function. The training encompassed compressive sensing (CS) ratios ranging from 10% to 50%. ADAM optimizer [58] was employed across all networks, with momentum at 0.9, weight decay at 0.999, and the learning rate set to 1×10^{-4} . While both ISTA-Net+ and COAST under-

Table 4.1: PSNR/SSIM performance comparison on the Set11 and BSD68 datasets [2], [1].

Dataset	Network	Conf.	CS Ratio					Avg. Score	Parameters
			10%	20%	30%	40%	50%		
Set11	ISTA-Net+ [4]	GDM	26.75/0.8108	30.93/0.9020	33.84/0.9394	36.04/0.9578	38.09/0.9707	33.13/0.9161	336,978
		GIN	26.92/0.8152	30.97/0.9040	33.78/0.9391	36.05/0.9582	38.13/0.9711	33.17/0.9175	339,561
	COAST [3]	GDM	28.64/0.8583	32.34/0.9222	34.91/0.9486	36.92/0.9631	38.75/0.9733	34.31/0.9331	1,122,024
		GIN	28.82/0.8627	32.49/0.9242	35.00/0.9498	37.05/0.9640	38.85/0.9739	34.44/0.9349	1,127,764
	CASNet [20]	GDM	30.38/0.8983	34.17/0.9470	36.68/0.9644	38.76/0.9746	40.59/0.9816	36.11/0.9532	15,680,080
		GIN	30.62/0.9035	34.45/0.9493	36.95/0.9662	38.98/0.9760	40.84/0.9825	36.37/0.9555	15,681,796
BSD68	ISTA-Net+ [4]	GDM	25.28/0.7003	28.08/0.8152	30.20/0.8769	32.06/0.9156	33.94/0.9423	29.91/0.8501	336,978
		GIN	25.36/0.7050	28.09/0.8171	30.24/0.8778	32.09/0.9157	33.97/0.9425	29.95/0.8516	339,561
	COAST [3]	GDM	26.23/0.7379	28.90/0.8372	30.97/0.8910	32.81/0.9244	34.58/0.9475	30.70/0.8676	1,122,024
		GIN	26.33/0.7422	29.00/0.8393	31.05/0.8923	32.89/0.9258	34.65/0.9486	30.78/0.8696	1,127,764
	CASNet [20]	GDM	28.08/0.8112	30.74/0.8894	32.85/0.9285	34.80/0.9525	36.71/0.9681	32.63/0.9099	15,680,080
		GIN	28.13/0.8152	30.81/0.8925	32.95/0.9313	34.94/0.9550	36.87/0.9703	32.74/0.9129	15,681,796

went training for 400 epochs, CASNet’s training spanned 320 epochs. For validation purposes, the Set11 dataset [2] was utilized, while the BSD68 dataset [1] was used for testing. The evaluation metrics of choice were the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) [60].

4.3.1 GIN vs. GDM

To assess the performance of our proposed GIN and whether it can replace GDM, we trained ISTA-Net+, COAST, and CASNet once using their default configuration, which uses GDM, and once using GIN. The results can be viewed in Table 4.1. The results show that on all networks and all datasets, GIN manages to outperform GDM. In fact, GIN was able to outperform it on all possible settings except for one. Furthermore, the gap between them generally increases as the CS ratio decreases. This indicates that GIN is more effective at lower CS ratios compared to GDM.

We also observe that the gap between GIN and GDM increases as the size of the network increases. From Table 4.1, we observe that ISTA-Net+ is the smallest network and we see that it exhibits the smallest performance gain using GIN, and

vice-versa for CASNet. This does suggest that GIN scales better with size, and thus, is more suitable for larger networks.

In general, the results clearly show that GIN successfully replaces GDM and allows the network to achieve higher performance, especially when the network size is large.

4.4 Ablation Study

4.4.1 Integrating Gradient-Informed Networks to the Recursive Framework

So far, we have discussed how GIN enhances the performance of deep unfolding networks. In this section, we aim to discuss if GINs can be integrated into the recursive framework. While GIN clearly outperforms GDM, this may not necessarily be the case for recursive networks. The concerning point is that recursive networks use fewer learnable parameters. Since the results in Table 4.1 and Table 4.1 show that GIN tends to scale better with larger networks, they may not show the same performance improvement when applied to recursive networks. To investigate this, GIN will be incorporated into the recursive framework and its results will be compared to its GDM-based counterpart. In addition, those results will be compared to the GDM-based non-recursive networks to serve as a general baseline.

Table 4.2 and Table 4.3 show the results of the recursive framework before and after using GIN on both the Set11 [2] and BSD68 [1] datasets respectively. The results in Table 4.2 indicate that integrating GIN into our recursive framework boosts

Table 4.2: PSNR/SSIM scores comparing recursive GIN to recursive GDM networks on the Set11 [2] dataset.

Model	Conf.	Layers	IPRB	CS Ratio					Avg. Score
				10%	20%	30%	40%	50%	
ISTA-Net+ [4]	Baseline	9	1	26.75/0.8108	30.93/0.9020	33.84/0.9394	36.04/0.9578	38.09/0.9707	33.13/0.9161
	GDM	3	3	27.05/0.8181	31.18/0.9056	34.02/0.9411	36.23 /0.9591	38.23/0.9714	33.34/0.9191
	GIN	3	3	27.17/0.8215	31.19/0.9069	34.05/0.9415	36.23/0.9592	38.24/0.9715	33.38/0.9201
COAST [3]	Baseline	20	1	28.64/0.8583	32.34/0.9222	34.91 /0.9486	36.92/0.9631	38.75/0.9733	34.31 /0.9331
	GDM	5	4	28.56/0.8559	32.28/0.9209	34.82/0.9476	36.82/0.9624	38.61/0.9727	34.22/0.9319
	GIN	5	4	28.75/0.8603	32.38/0.9231	34.91/0.9488	36.84/0.9630	38.64/0.9729	34.30/0.9336
CASNet [20]	Baseline	12	1	30.39/0.9019	34.33/0.9493	36.95/0.9665	39.08/0.9765	40.95/0.9829	36.34/0.9554
	GDM	4	3	30.45/0.9029	34.39/ 0.9499	37.02/0.9670	39.19/0.9770	41.05/0.9833	36.42/0.9560
	GIN	4	3	30.66/0.9057	34.44/0.9499	37.09/0.9674	39.22/0.9772	41.11/0.9834	36.50/0.9567

Table 4.3: PSNR/SSIM scores comparing recursive GIN to recursive GDM networks on the BSD68 [1] dataset.

Model	Conf.	Layers	IPRB	CS Ratio					Avg. Score
				10%	20%	30%	40%	50%	
ISTA-Net+ [4]	Baseline	9	1	25.28/0.7003	28.08/0.8152	30.20/0.8769	32.06/0.9156	33.94/0.9423	29.91/0.8501
	GDM	3	3	25.42/0.7074	28.20 /0.8189	30.36/0.8802	32.26/0.9177	34.09/0.9434	30.07 /0.8535
	GIN	3	3	25.47/0.7089	28.20/0.8202	30.34/0.8791	32.21/0.9172	34.06/0.9433	30.06/ 0.8537
COAST [3]	Baseline	20	1	26.23/0.7379	28.90 /0.8372	30.97 /0.8910	32.81 /0.9244	34.58 /0.9475	30.70 /0.8676
	GDM	5	4	26.12/0.7325	28.81/0.8339	30.89/0.8892	32.74/0.9237	34.52/0.9468	30.62/0.8652
	GIN	5	4	26.26/0.7395	28.90/0.8382	30.94/0.8914	32.77/0.9250	34.51/0.9476	30.67/0.8683
CASNet [20]	Baseline	12	1	28.09/0.8156	30.80/0.8933	32.98/0.9322	35.00/0.9558	36.99/0.9710	32.77/0.9136
	GDM	4	3	28.12 /0.8176	30.85 /0.8949	33.02 /0.9333	35.05 /0.9565	37.02 /0.9715	32.81 /0.9147
	GIN	4	3	28.11/ 0.8178	30.84/ 0.8953	33.02/0.9337	35.03/0.9568	37.00/0.9718	32.80/0.9151

its performance, which is consistent across all three networks. Notably, while the recursive COAST network originally was unable to outperform its baseline when using GDM, GIN on the other hand allows it to outperform the baseline at CS ratios lower than 30%.

Compared to the results on the Set11, the results on the BSD68 show smaller gains when GIN is used. This can be attributed to the Set11 dataset being used for validation, and thus, the results will not be as consistent on other datasets. With that said, we observe that GIN still achieves the highest SSIM for all networks. Furthermore, we also see again that the performance advantage of GIN usually increases as the CS ratio decreases, which again indicates that its effectiveness increases on lower CS ratios.

Overall, when compared to the baseline, combining GIN with the recursive framework manages to enhance both its performance and efficiency in terms of the learnable parameters and training time.

4.4.2 The Impact of Removing GDM

In this work, we have discussed the importance of the GDM, and we showed how better results can be obtained by improving its adaptation to neural networks. However, we would like to further highlight how important the GDM is by completely removing it. That is when neither GDM nor GIN is used by the network.

Table 4.4: PSNR/SSIM performance comparison on the BSD68 dataset [1].

Network	GDM	CS Ratio 10%
ISTA-Net+ [4]	X	24.40/0.6608
	✓	25.28/0.7003
COAST [3]	X	19.59/0.4237
	✓	26.23/0.7379
CASNet [20]	X	27.44/0.8025
	✓	28.08/0.8112

Table 4.4 shows the results of ISTA-Net+ [4], COAST [3], and CASNet [20] before and after the GDM was removed from them. It can be observed in Table 4.4 that using GDM improves their results compared to their non-GDM counterpart. However, one can notice that the gap is inconsistent between the two configurations. This discrepancy is *mainly* due to initialization. CASNet has the best initialization due to its use of a learned sampling matrix, and thus, it sees the least gain from GDM, while COAST is vice-versa.

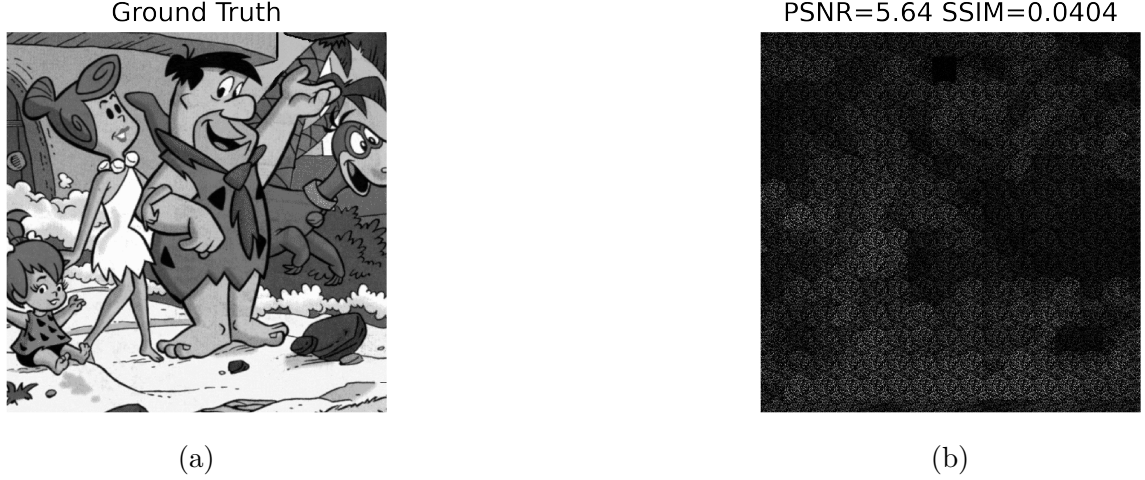


Figure 4.3: Shows an image (a) and its respective initial solution (b) when the CS ratio is 10%

Those results do bring further insight, however. We observe that as the size of the network increases, the performance gain from GDM decreases, while from Table 4.1 and Table 4.1, we observe that as the network increases in size, the gap between GIN and GDM increases. This could indicate that as the network becomes larger and has better initialization, it becomes challenging for GDM to minimize L_y , while GIN on the other hand scales better in such cases.

4.4.3 Why Does the GDM Matter to DUNs?

In section 4.1, we demonstrated the importance of the GDM in Figure 4.1. It has the learning curves of two COAST [3] networks. One was trained with the GDM, and the other was not. Although the difference between the two networks in terms of learnable parameters is negligible, their performance gap is significant. It is important to discuss why using GDM led to such a large gap in the first place.

First, to see why COAST [3] performed poorly without GDM, we have to look at

the initial solution $\hat{\mathbf{x}}^{(0)}$ that was fed to the network. Figure 4.3a shows an example from the Set11 dataset [2] and its respective initial solution Figure 4.3b. It can be observed that the initial solution fed to the network is very noisy as it only scores 5.64 dB in terms of PSNR. With such a noisy input, it becomes very challenging for the network to reconstruct the image. Its reconstruction results can be seen in Figure 4.4, which show poor reconstruction performance. However, this challenge can be alleviated if we enforce some conditions the output should satisfy. This is exactly where compressed sensing theory and ISTA [13] can be used. They state that such problems can be solved if they are formulated as in Equation (4.1). Since COAST in Figure 4.4 does not use GDM, then it is completely unable to minimize L_y , thus, preventing it from using the minimization function set by ISTA [13], which is why it demonstrates poor performance. This is why minimizing L_y plays a major role in the recovery of deep unfolding networks.

On the other hand, when GDM is used, the network sees a significant uplift in performance. Figure 4.4 shows the intermediate recovery results of COAST [3] when GDM is used normally. It can be observed that the network generally improves its results with each subsequent recovery block thanks to its use of GDM.

Finally, we also show the intermediate results of COAST [3] when it uses GIN in Figure 4.4. The results demonstrate that not only does GIN allow COAST to achieve higher performance at the last recovery block, but even during the first four intermediate results. This indicates that GIN allows the network to more fully realize each recovery block, and thus, could potentially achieve similar performance to GDM using fewer recovery blocks. It also hints at GIN being more effective at minimizing

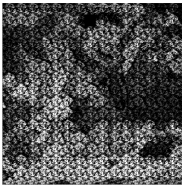
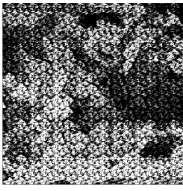
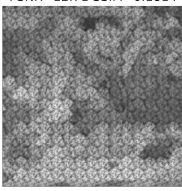
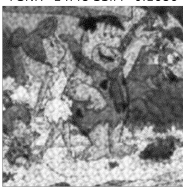
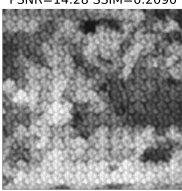
Iteration	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(20)}$
No GDM	PSNR=7.81 SSIM=0.0936 	PSNR=7.86 SSIM=0.0911 	PSNR=7.61 SSIM=0.0875 	PSNR=7.78 SSIM=0.0882 	PSNR=14.94 SSIM=0.3194 
GDM	PSNR=12.71 SSIM=0.1814 	PSNR=14.48 SSIM=0.1686 	PSNR=12.61 SSIM=0.1297 	PSNR=18.80 SSIM=0.4762 	PSNR=26.36 SSIM=0.8288 
GIN	PSNR=14.28 SSIM=0.2090 	PSNR=17.37 SSIM=0.4153 	PSNR=19.53 SSIM=0.5377 	PSNR=20.15 SSIM=0.5551 	PSNR=26.60 SSIM=0.8323 

Figure 4.4: Intermediate recovery results from COAST [3] once without GDM, once when GDM is used, and once with our proposed GIN.

L_y since the number of parameters of the network is mostly the same.

4.5 Conclusion

In this work, we have discussed the general design of deep unfolding networks, and how they focus on improving the deep reconstruction network while mostly leaving the GDM part unchanged. We then pointed out its critical role and showed how its inclusion can significantly enhance the reconstruction performance. Given the importance of the GDM, we proposed the gradient-informed network. GIN is designed to replace GDM, which is why simplicity was one of its core design pillars. This allows it to be easily applied to other deep unfolding networks with minimal changes to

their structure. To prove this, we applied GIN to three networks, which were ISTA-Net+, COAST, and CASNet. The results show that in almost all test conditions, GIN outperforms GDM. We also showed that GIN tends to scale better with larger networks. We believe the simplicity of GIN can open the doors for other people in the CS community to experiment with other methods on how to best feed the gradient to the reconstruction network.

CHAPTER 5

CONCLUSION

5.1 Summary

In this thesis, we have discussed the wide success of deep unfolding networks for compressed sensing applications. With that said, we pointed out two points of weakness that are associated with most DUNs. First, DUNs usually improve their performance by increasing their number of recovery blocks. However, each recovery block is only used once. We argued that this approach is inefficient as it unnecessarily increases the number of recovery blocks while leaving the recovery blocks underutilized. Secondly, we drew your attention to the GDM found in most DUNs. Despite its critical role in recovery, it only consists of a single learnable parameter, which limits its effectiveness. Furthermore, the GDM is isolated from the reconstruction network. This makes it difficult for the reconstruction network to produce an output that is consistent with the measurements, and thus, cannot fulfill some of the CS theory requirements by itself.

Regarding the first point of weakness, we proposed a recursive framework that

introduces a recursive connection to each block to more fully utilize its capacity and decrease the number of learnable parameters. In addition, we introduced a training strategy that makes use of the recursive connection to decrease the training time, and lastly, we introduced our RFMU that modulates the features of the recovery block in each iteration to enhance its performance. The results of the recursive framework showed that the number of learnable parameters can decrease the training time by around 28% and decrease the number of learnable parameters by more than 67% while mostly preserving the performance of the networks. Therefore, the recursive framework successfully addressed our first point of concern regarding DUNs.

As for the second point of weakness found in most DUNs, we proposed GIN as an effective solution that can completely replace GDM. By concatenating the input depthwise with the gradients, the reconstruction network can minimize the measurement loss by itself, thus, enabling it to satisfy CS theory. Our results show that in almost all test conditions across all tested networks, GIN outperforms GDM. Notably, GIN outperforms GDM while barely increasing the number of learnable parameters. This indicates that the enhancements brought by GIN are attributed to the way it adapts the CS problem to DUNs. Furthermore, the simplicity of GIN allows it to be applied to a wide range of DUNs that use GDM to enhance their performance. Lastly, we showed how GIN can easily be applied to our recursive framework and see similar performance gains.

5.2 Future Work

Finally, while we have improved the general structure of deep unfolding networks, more enhancements could still be made. In this work, we first focused on recursions and how they can enhance efficiency. However, the recursive framework was limited to deep unfolding networks and the recursions were also limited to the spatial domain. As such, we would like to delve deeper into integrating the recursive framework into other types of networks. This would be feasible if recursions were successfully applied in the feature space, and would benefit not only DCSNs but also other generative networks as their number of parameters can reach billions. Therefore, this is one of the routes we are currently investigating. Furthermore, during inference time, the number of iterations per recovery block (IPRB) is fixed for all recovery blocks. This may not be ideal since the effectiveness of recursions may depend on the recovery block. Thus, it might be more optimal to investigate methods that can find a more optimal value for the IPRB for each recovery block.

In the second part of this work, we tackled the GDM part and how the output of the network must match the measurements. As such, our next point for optimization would be directed towards sparsity, and how it can be utilized for better reconstruction. Integrating sparsity to deep unfolding networks makes intuitive sense as it aligns with CS theory. However, letting the reconstruction network utilize that sparsity is difficult. For example, assume one sparsified the image using the DCT transform and then fed that transformed image to the network. While the input is sparse, it has lost its local information. Since most deep unfolding networks are convolutional, feature

extraction would become difficult as convolutional layers rely on local information for feature extraction. Therefore, feeding the network with a sparse input directly using the DCT is not recommended, and thus, would need a more elaborate method for optimal results. Another direction for future work would be to find a better method of feeding the gradient to the network. For example, combining the gradient descent network proposed by DUN-CSNet with GIN might be a viable method for enhancing GIN.

REFERENCES

- [1] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2. IEEE, 2001, pp. 416–423.
- [2] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, “Reconnet: Non-iterative reconstruction of images from compressively sensed measurements,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 449–458.
- [3] D. You, J. Zhang, J. Xie, B. Chen, and S. Ma, “Coast: Controllable arbitrary-sampling network for compressive sensing,” *IEEE Transactions on Image Processing*, 2021.
- [4] J. Zhang and B. Ghanem, “Ista-net: Interpretable optimization-inspired deep network for image compressive sensing,” in *CVPR*, 2018, pp. 1828–1837.
- [5] C. Shannon, “Communication in the presence of noise,” *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

- [6] W. Mousa, *Advanced Digital Signal Processing of Seismic Data*. Cambridge University Press, 2020.
- [7] C. G. Graff and E. Y. Sidky, “Compressive sensing in medical imaging,” *Applied optics*, vol. 54, no. 8, pp. C23–C44, 2015.
- [8] D. J. Brenner and E. J. Hall, “Computed tomography—an increasing source of radiation exposure,” *New England journal of medicine*, vol. 357, no. 22, pp. 2277–2284, 2007.
- [9] L. Li, Y. Fang, L. Liu, H. Peng, J. Kurths, and Y. Yang, “Overview of compressed sensing: Sensing model, reconstruction algorithm, and its applications,” *Applied Sciences*, vol. 10, no. 17, p. 5909, 2020.
- [10] L. C. Potter, E. Ertin, J. T. Parker, and M. Cetin, “Sparsity and compressed sensing in radar imaging,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1006–1020, 2010.
- [11] T. Strohmer, “Measure what should be measured: progress and challenges in compressive sensing,” *IEEE Signal Processing Letters*, vol. 19, no. 12, pp. 887–893, 2012.
- [12] E. J. Candès, J. Romberg, and T. Tao, “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,” *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.
- [13] I. Daubechies, M. Defrise, and C. De Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Communications on Pure and*

- Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.
- [14] J. A. Tropp and A. C. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *IEEE Transactions on information theory*, vol. 53, no. 12, pp. 4655–4666, 2007.
 - [15] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.
 - [16] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.
 - [17] W. Shi, F. Jiang, S. Zhang, and D. Zhao, “Deep networks for compressed image sensing,” *CoRR*, vol. abs/1707.07119, 2017. [Online]. Available: <http://arxiv.org/abs/1707.07119>
 - [18] D. Ye, Z. Ni, H. Wang, J. Zhang, S. Wang, and S. Kwong, “Csformer: Bridging convolution and transformer for compressive sensing,” *IEEE Transactions on Image Processing*, 2023.
 - [19] J. Song, B. Chen, and J. Zhang, “Memory-augmented deep unfolding network for compressive sensing,” in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 4249–4258.
 - [20] B. Chen and J. Zhang, “Content-aware scalable deep compressed sensing,” *IEEE Transactions on Image Processing*, vol. 31, pp. 5412–5426, 2022.

- [21] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.
- [22] J. Sun, H. Li, Z. Xu *et al.*, “Deep admm-net for compressive sensing mri,” *Advances in neural information processing systems*, vol. 29, 2016.
- [23] D. L. Donoho, A. Maleki, and A. Montanari, “Message-passing algorithms for compressed sensing,” *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18 914–18 919, 2009.
- [24] Z. Zhang, Y. Liu, J. Liu, F. Wen, and C. Zhu, “Amp-net: Denoising-based deep unfolding for compressive image sensing,” *IEEE Transactions on Image Processing*, vol. 30, pp. 1487–1500, 2020.
- [25] J. Zhang, C. Zhao, and W. Gao, “Optimization-inspired compact deep compressive sensing,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 765–774, 2020.
- [26] W. Cui, X. Fan, J. Zhang, and D. Zhao, “Deep unfolding network for image compressed sensing by content-adaptive gradient updating and deformation-invariant non-local modeling,” *IEEE Transactions on Multimedia*, vol. 26, pp. 4012–4027, 2024.
- [27] X. Ying, “An overview of overfitting and its solutions,” in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

- [28] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “A survey of model compression and acceleration for deep neural networks,” *arXiv preprint arXiv:1710.09282*, 2017.
- [29] N. Smith, “Ooh! ahh! google images presents a nicer way to surf the visual web,” <https://googleblog.blogspot.com/2010/07/ooh-ahh-google-images-presents-nicer.html>, 2010, accessed on May 23, 2023.
- [30] G. H. Golub and C. Reinsch, “Singular value decomposition and least squares solutions,” *Linear algebra*, vol. 2, pp. 134–151, 1971.
- [31] D. Donoho, “Compressed sensing,” *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [32] S. C. Park, M. K. Park, and M. G. Kang, “Super-resolution image reconstruction: a technical overview,” *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 21–36, 2003.
- [33] K. Nasrollahi and T. B. Moeslund, “Super-resolution: a comprehensive survey,” *Machine vision and applications*, vol. 25, pp. 1423–1468, 2014.
- [34] D. Glasner, S. Bagon, and M. Irani, “Super-resolution from a single image,” in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 349–356.
- [35] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, “Fast and robust multiframe super resolution,” *IEEE transactions on image processing*, vol. 13, no. 10, pp. 1327–1344, 2004.

- [36] E. J. Candès and C. Fernandez-Granda, “Super-resolution from noisy data,” *Journal of Fourier Analysis and Applications*, vol. 19, pp. 1229–1254, 2013.
- [37] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Springer New York, 2013. [Online]. Available: <https://doi.org/10.1007/978-0-8176-4948-7>
- [38] W. Shi, F. Jiang, S. Liu, and D. Zhao, “Image compressed sensing using convolutional neural network,” *IEEE Transactions on Image Processing*, vol. 29, pp. 375–388, 2019.
- [39] C. A. Metzler, A. Maleki, and R. G. Baraniuk, “From denoising to compressed sensing,” *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5117–5144, 2016.
- [40] C. Li, W. Yin, H. Jiang, and Y. Zhang, “An efficient augmented lagrangian method with applications to total variation minimization,” *Computational Optimization and Applications*, vol. 56, no. 3, pp. 507–530, 2013.
- [41] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.
- [42] C. Dong, C. C. Loy, and X. Tang, “Accelerating the super-resolution convolutional neural network,” in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 391–407.

- [43] A. Watson, “Deep learning techniques for super-resolution in video games,” *CoRR*, vol. abs/2012.09810, 2020. [Online]. Available: <https://arxiv.org/abs/2012.09810>
- [44] Z. Wang, J. Chen, and S. C. H. Hoi, “Deep learning for image super-resolution: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3365–3387, 2021.
- [45] J. Li, Z. Pei, and T. Zeng, “From beginner to master: A survey for deep learning-based single-image super-resolution,” *ArXiv*, vol. abs/2109.14335, 2021.
- [46] A. Mousavi, A. B. Patel, and R. G. Baraniuk, “A deep learning approach to structured signal recovery,” in *2015 53rd annual allerton conference on communication, control, and computing (Allerton)*. IEEE, 2015, pp. 1336–1343.
- [47] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3-d transform-domain collaborative filtering,” *IEEE Transactions on image processing*, vol. 16, no. 8, pp. 2080–2095, 2007.
- [48] K. Zhang, Z. Hua, Y. Li, Y. Zhang, and Y. Zhou, “Uformer-ics: A u-shaped transformer for image compressive sensing service,” *IEEE Transactions on Services Computing*, 2023.
- [49] W. Shi, F. Jiang, S. Liu, and D. Zhao, “Scalable convolutional neural network for image compressed sensing,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 282–12 291.

- [50] V. Kravets and A. Stern, “Progressive compressive sensing of large images with multiscale deep learning reconstruction,” *Scientific Reports*, vol. 12, no. 1, p. 7228, 5 2022. [Online]. Available: <https://doi.org/10.1038/s41598-022-11401-7>
- [51] K. Zhang, L. V. Gool, and R. Timofte, “Deep unfolding network for image super-resolution,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [52] Q. Ma, J. Jiang, X. Liu, and J. Ma, “Deep unfolding network for spatio-spectral image super-resolution,” *IEEE Transactions on Computational Imaging*, vol. 8, pp. 28–40, 2022.
- [53] W. Wu, J. Weng, P. Zhang, X. Wang, W. Yang, and J. Jiang, “Uretinex-net: Retinex-based deep unfolding network for low-light image enhancement,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 5901–5910.
- [54] C. Mou, Q. Wang, and J. Zhang, “Deep generalized unfolding networks for image restoration,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 17 399–17 410.
- [55] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, “Denoising prior driven deep neural network for image restoration,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2305–2318, 2018.

- [56] Z.-E. Fan, F. Lian, and J.-N. Quan, “Global sensing and measurements reuse for image compressed sensing,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 8954–8963.
- [57] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
- [58] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [59] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, vol. 32, 2019.
- [60] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [61] D. You, J. Xie, and J. Zhang, “Ista-net++: flexible deep unfolding network for compressive sensing,” in *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2021, pp. 1–6.
- [62] J. Song, B. Chen, and J. Zhang, “Dynamic path-controllable deep unfolding network for compressive sensing,” *IEEE Transactions on Image Processing*, 2023.

VITAE

- Name: Rawwad Sultan Abdul Mohie Alhejaili
- Nationality: Saudi
- Date of Birth: 30/12/1996
- Email: *Rawwad.Alhejaili@Gmail.com*
- Permanent Address: Madinah, Saudi Arabia

Thesis Products:

- *R. Alhejaili, M. Alfarraj, H. Luqman, and A. Al-Shaikhi, "Recursions Are All You Need: Towards Efficient Deep Unfolding Networks," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Jun. 2023.*