# A DEEP RECURRENT NEURAL NETWORK WITH ITERATIVE OPTIMIZATION FOR INVERSE IMAGE PROCESSING APPLICATIONS

by

Masaki Ikuta

A Dissertation Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in Engineering

at

The University of Wisconsin-Milwaukee

December 2021

# ABSTRACT

A DEEP RECURRENT NEURAL NETWORK
WITH ITERATIVE OPTIMIZATION
FOR INVERSE IMAGE PROCESSING APPLICATIONS

by

Masaki Ikuta

The University of Wisconsin - Milwaukee, 2021
Under the Supervision of Professor Jun Zhang

Many algorithms and methods have been proposed for inverse image processing applications, such as super-resolution, image de-noising, and image reconstruction, particularly with the recent surge of interest in machine learning and deep learning methods.

As for Computed Tomography (CT) image reconstruction, the most recently proposed methods are limited to image domain processing, where deep learning is used to learn the mapping between a true image data set and a noisy image data set in the image domain. While deep learning-based methods can produce higher quality images than conventional model-based algorithms, these methods have a limitation. Deep learning-based methods used in the image domain are insufficient to compensate for lost information during a forward and backward projection in CT image reconstruction, especially with high noise. This dissertation proposes new iterative reconstruction algorithms implemented by the Recurrent Neural Network (RNN). The RNN is usually used to process sequential data, such as a stock price prediction or natural language processing. In this dissertation, we use the RNN to implement the iterative reconstruction (IR), where the RNN performs an iterative optimization for CT image reconstruction. Besides, we propose new RNN

memory cells called Gated Momentum Unit (GMU) and Recurrent FISTA Unit (RFU) to keep the RNN cell preserve a long-term memory. The GMU and GFU are similar to the Long-Short Term Memory (LSTM) and the Gated Recurrent Unit (GMU), in which the RNN cells alleviate a banishing and an exploding gradient problem. The GMU and GFU have simpler network structures than the LSTM and the GRU, and they are particularly designed to accelerate the convergence of the training optimization process. We conducted a simulation study and a real CT image study to demonstrate that these proposed methods achieved the highest Peak Signal to Noise Ratio (PSNR) and Structure Similarity (SSIM). The GMU was evaluated in CT image reconstruction, and the GFU was evaluated in CT Metal Artifact Reduction (CT MAR). Also, we showed these algorithms converged faster than other well-known methods.

Furthermore, in the fourth chapter of this dissertation, we discuss how vital image texture is in inverse image processing problems. Many methods have been proposed for these problems; however, the most popular methods, the convolutional neural network (CNN) based methods with a Mean Squared Error (MSE) are known to over-smooth images due to the nature of the MSE. The MSE-based methods minimize Euclidean distance for all pixels between a baseline image and a CNN-generated image and ignore the pixels' spatial information, such as image texture. The chapter of this dissertation proposes a new method based on Wasserstein GAN (WGAN) for inverse problems. We showed that the WGAN-based method was effective in preserving image texture. It also used a maximum likelihood estimation (MLE) regularizer to preserve pixel fidelity. Maintaining image texture and pixel fidelity is an essential requirement in medical imaging. We used PSNR and SSIM to evaluate the proposed method quantitatively. We also conducted first-order and second-order statistical image texture analysis to assess image texture.

To those who supported my Ph.D. journey

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

First of all, I would like to thank Professor Jun Zhang for being helpful, friendly, educational, visionary, humble, and patient in my Ph.D. journey. He has accommodated my busy working schedule for the past eight years. Without his understanding and unconditional support, I would not be where I am today.

I also appreciate all of the suggestions and the comments from my Ph.D. dissertation committee members: Dr. Susan McRoy, Dr. Lijing Sun, Dr. Zeyun Yu, Dr. Yi Hu, and the list goes on. It has been my honor to work with my fellow lab students in the past and present: Dr. Yingying Gu, Dr. Hongquan Zuo, Mr. Yu Lu, Mr. Carlos Ivan Jerez Gonzalez, and Ms. Mehri Bagherimohamadipour. Thank you for all of your support.

Last but not least, I would like to thank my colleagues in the Computed Tomography Image Reconstruction team at GE Healthcare. They always supported my learning journey and encouraged me to finish my Ph.D. I also appreciate how the Central Data Science team at GE Healthcare AI welcomed me as a new team member. I am genuinely honored to be part of the team to take Artificial Intelligence to the next level in the healthcare industry.

Eight years ago, I started my Ph.D. journey by taking one evening class in the university taught by a chief scientist at GE Healthcare. I thought my Ph.D. endeavor was easier and quicker to finish, given that I already had a master's degree from Japan. However, my Ph.D. journey was much more challenging and exceptionally longer than I thought. My eldest daughter started the first grade on the same day as I started my Ph.D. voyage. It has been easy for me to count how many years I have been on the program because she is now in the ninth grade in high school. I continue to be devastated by how much family time I sacrificed to finish my academic challenge every time I compare my daughters' pictures from day one to the present. But given that I have always loved

scientific subjects, I truly believe that the time I sacrificed was not wasted. I came across many sayings and proverbs during the past eight years, which encouraged me to continue. The ones I like the most are the statements given by Admiral McRaven in the U.S. Navy. General McRaven is a four-star admiral and was a Navy SEAL for 37 years.

He explained how he learned never to give up. The following is one of his statements:

> We used to have a saying in Navy SEAL training, "Take it one evolution at a time." It means that you do not look six months down the road. You do not ask yourself, or you do not look and say, "My gosh, I have got more swims and more runs and more PT's." If you do that, that event horizon becomes a little too far. And I think it can be frightening. If all you do is try to do the very best you can at that very moment, you take it one step at a time, and then six months goes by, you took it one evolution at a time, and you made it. (Admiral McRaven 2017)

He also said in a Business Insider interview about a secret to going through Navy SEAL training:

> It is easy to quit Navy SEAL training. All you have to do is ring a bell three times. A brass bell hangs in the center of the compound for all the students to see. All you have to do is ring the bell, and you are out. You don't have to talk to anybody. You don't have to do anything. You ring the bell, you take your helmet off, you put it down, and that is it. And you find that in tough times, there is always kind of a way out. And that is quitting. That's just deciding you're not going to tackle this problem. You're going to let the problem or the situation win. And so the one thing I'm always asked is, "How do you get through SEAL training?" I had a young man who was going off to SEAL training about a year ago. And he was a phenomenal

> athlete. I had lunch with him, and he said, "Well, do I need to run more?" I said, "No, I don't think so." He said, "Do I need to swim more?" I said, "Nope." "Do I need to lift more?" and he said, "What is the key to going through SEAL training?" I said, "It's simple."
> "YOU JUST DON'T QUIT." (Admiral McRaven 2017)

While I cannot simply compare the Navy SEAL training with my Ph.D. journey, there are some similarities. As a part-time student, I had more reasons to quit. The only reason I did not quit is my passion for science and mathematics. I am always excited when I see a new science problem or a complex mathematical problem. Of course, I could have "rung the bell" for an uncountable number of reasons. But it seems that my passion and grit matter in the end.

I cannot be grateful enough for all the people who supported my academic challenge. I would like to thank my wonderful wife, Rina, my beautiful daughters, Yulia and Nanoka, my friends, and my colleagues for letting me finish this difficult journey.

October 2021

Masaki Ikuta

# Chapter 1

# Introduction

Image processing has a long history. The first digital image processing application was developed for the newspaper industry in the early 1920s [1]. An image was coded and transmitted across the Atlantic Ocean, and the image was then reconstructed at the receiving end on a telegraph printer. As the world was drawn into the global war in the 1940s, medical imaging technologies emerged into practical usage. They were beneficial to diagnose war injuries. In the 1960s, NASA began developing computer algorithms to improve moon images. In the 1970s, Computed Tomography (CT) was invented, and Sir Hounsfield and Professor Cormack shared the Nobel Prize for the innovation. In the 1980s, the use of digital image processing technologies exploded. Many new medical imaging systems have been developed around that time to detect diseases in the early stages. Image processing technologies have played a critical role in many areas of our society, and it is still one of the most crucial and active research areas today.

There are many image processing applications. One of the most popular applications nowadays is object detection. A famous object detection application is autonomous driving systems where computers with multiple cameras detect other vehicles, traffic signals, and pedestrians. In addition, many researchers and engineers have been working on image classification applications recently. Examples are to detect cancer on an X-ray image,

classify a handwritten number on a gray-scale image, or assign a person's name to a picture of a face. Image denoising is another type of image processing application that is essential in the medical imaging industry. X-ray images have many kinds of noises. There are typically two types of noises. One is an equipment noise that typically follows Gaussian distribution. The other is an X-ray scanning noise that generally follows the Poisson distribution. There are different kinds of noises on images, too [2]. These multi types of noises can be reduced by increasing an X-ray dosage. However, it comes at the expense of patient body damage. Thus, it makes sense to use an image denoising application to lower these types of noises. Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) are more advanced medical imaging systems, and they have built-in image denoising applications.

Ever since image processing applications emerged into practical usage, researchers have worked on challenging problems and came up with ideas to overcome these issues. One of the early image processing methods was template matching for multi-class classification, such as classifying a handwritten digit. Many templates were created for each handwritten digit, and images were compared against these templates. Typically, a mathematical distance between a template and a handwritten input image was computed to classify a handwritten digit. Although this method is intuitive and easy to understand, there are many drawbacks to this approach. A handwritten digit often varies in size, and it is also frequently rotated. Therefore, it is not straightforward to compare a handwritten input image with a template. Also, using templates sometimes creates a computational dilemma. The less number of templates is always preferable to keep computational time manageable, but the less number of templates would degrade the classification performance. Finding an optimal number of templates is not a trivial task. Moreover, many image denoising methods have been proposed in the past decades. Performing a low pass filtering on an image is usually the most straightforward way to reduce imaging noise. But in turn, it reduces the sharpness of images as a result. Non-local mean filter (NLM

filter) and block-matching and 3D filtering (BM3D filter) are relatively recent and great image filtering methods. They try to find similar patches within an image or the peripherals and use them to reduce imaging noise. However, the performance of these filtering methods is limited by the number of images for patches because these methods only try to find similar patches within a certain number of images. Of course, a large number of images for patches would increase the computational time. Thus, there is always a trade-off between the processing time and the filtering performance. Furthermore, image reconstruction in medical imaging attracted many researchers for their studies. Bringing scanning raw data into images involves many complex non-linear calculation steps. Thus, much important information gets lost within these steps. Model-based methods have been favored to tackle these problems. These methods were developed based on scanning models, X-ray physics, and noise models for X-ray scanners and CT scanners. Magnetic fields and radio waves need to be considered for MRI scanners, which drives even more complex models to handle MRI-specific problems. Although these "model-based" methods were popular in the past, a hand-made solution needs to be developed for each one of the problems. A solution for one problem would not be scalable for another area in image processing.

Many people think that the neural network was only a recent development within this ten to twenty years. However, it also has a long history. It was first developed in the 1940s by Warren McCulloch and Walter Pitts. They wrote a paper about how neurons in the human brain would work, and they modeled a simple neural network with electrical circuits. Since then, the neural network has been researched by many people for many years. However, it is only recently that many researchers started using neural networks in image processing applications. AlexNet was developed in 2012 by a research group at the University of Toronto, and they won the ImageNet Large Scale Visual Recognition Challenge with a large margin [3]. AlexNet was one of the largest neural networks at the time of the work in 2012. Their work was significant because it showed the potential

to quickly train an extensive neural network on a large dataset using widely available gaming computer hardware called Graphical Processing Unit (GPU). Before that, neural networks were primarily trained by Central Processing Unit (CPU). The contributions of AlexNet were unique in terms of using novel ReLU activation, data augmentation, dropout, and local response normalization. All of these made it possible to achieve cutting-edge performance in the object recognition competition in 2012. Since AlexNet, the research community has been dominated by the usage of neural networks. The best part of neural network-based methods is that we do not need to design a specific solution for each problem. The neural network is already designed such that it is smart enough to handle many kinds of problems. If a particular neural network design works well in image denoising in natural imaging, it would likely work well for medical imaging. If a neural network-based method works well for CT image reconstruction, the same network would most likely work well for MRI image reconstruction without having a depth of knowledge about the specific subject. Therefore, it has been shown by many research papers that the neural network-based methods, often called the "data-driven" methods, produced superior results than the conventional "model-based" methods.

Image processing applications with deep learning and machine learning have been attracting much attention recently. Many research papers have been published, but most of them are still in active research. As for medical imaging, denoising methods with CNN became one of the first and most successful applications in academia and the commercial industry. The CNN-based techniques have been proven effective compared to the model-based algorithms. However, the majority of the methods are limited to image domain processing. As explained above, observation data for CT and MRI have to be converted into the image domain so that clinicians can use images for diagnosis. However, CT and MRI image reconstruction are only approximations. Therefore, much information gets lost during the image reconstruction steps. CNN-based methods are, so far, used in the image domain in most research papers and how to take advantage of deep

learning capability on observation data still needs to be explored. Furthermore, these CNN-based methods achieved cutting-edge performance in image denoising. However, it often creates over-smoothed images. So far, the research community only pursues improving Peak Signal-to-Noise Ratio (PSNR) and Structure Similarity (SSIM), and producing over-smoothed images by data-driven methods.

In this dissertation, we try to recover information lost from medical image reconstruction processing. We use CT data to prove our method is effective. Using a CNN in the image domain cannot recover much of the lost information during reconstruction steps. We need to take into account CT image reconstruction steps. This means that we use both image data and observation data to compensate for the lost information during reconstruction steps. This research is significant in such a way that we can potentially lower X-ray doses in CT scanning. X-ray dose is invasive to patient bodies, and it is even more problematic for pediatrics. If we can recover more information from observation data, this means we would not need to expose much of the X-ray into patient bodies. Furthermore, we also try to reduce image texture change in data-driven methods. Frequently, image texture is damaged after using data-driven techniques. However, image texture is essential for medical diagnosis. An image texture change can potentially lead to a diagnosis error. Keeping the PSNR and SSIM high with a data-driven method while preserving image texture would also help clinicians for their diagnosis tasks.

The organization of this thesis is the following. Chapter two shows how we use a deep recurrent neural network as an iterative optimizer. This neural network uses both images and observation data of CT, and it tries to recover lost information in CT image reconstruction steps. In chapter 3, we introduce Recurrent FISTA UNIT (RFU) for CT metal artifact reduction. The RFU is an RNN cell and is designed specifically for CT image reconstruction. Chapter four explains how we use Wasserstein GANs to keep the PSNR and the SSIM high while preserving image texture. The WGAN framework is

compared against a CNN-based method to prove its effectiveness. Chapter five concludes this thesis.

# Chapter 2

# A Deep Recurrent Neural Network with Gated Momentum Unit for CT Image Reconstruction

## 2.1 Introduction and Related Work

Computed Tomography (CT) is one of the most important medical imaging modalities in use today. CT scans can be used to diagnose diseases and detect injuries in various regions of a patient body [2, 4]. For example, CT has become a useful screening tool to detect a potential tumor or a lesion in abdomen. Also, a CT scan may be ordered by a physician when some type of heart disease is suspected. Because of the fast scanning and image reconstruction process, CT has become a popular choice of diagnostic tool in many clinical scenarios including diagnosis in emergency rooms [5, 6, 7]. In CT, a narrow beam of X-ray is exposed to a patient and a X-ray tube and a CT detector are quickly rotated around the patient body. This scanning process produces X-ray projection data which are often called view data [2, 8]. The view data show X-ray intensities as to how much the CT detector catches X-ray photons. Some X-ray beams are reduced by the patient anatomies while other beams can pass through the air region and these intensities are

not weakened. By applying a negative log step, view data are converted into sinogram data which show X-ray attenuation [2]. The sinogram data are processed by a number of algorithms to produce a cross-sectional image or a slice of the patient body. These signal and image processing steps are called CT image reconstruction. Although there are a number of processing steps to process scanning data, the key step to create CT images out of sinogram data is called backprojection where scanning data in sinogram domain are converted or projected into two or three dimensional pixel data in image domain. The sinogram domain and the image domain are considered to be two different domains. One is in the scanning coordinate and the other is in the patient coordinate [2]. One of the most frequently used backprojection step is the filtered backprojection or the FBP in short. The FBP is very popular for commercial CT systems because of the simplicity and the computational efficiency. However, the X-ray scanning process is inherently stochastic, with the received X-ray energy fluctuating at the CT detector. But the FBP treats it essentially as deterministic. As a result, "outliers" or more extreme projection values can produce significant artifacts.

To solve these problems, a stochastic approach known as iterative reconstruction (IR) has attracted much attention in recent years [9, 10, 11, 12, 13]. This method reconstructs cross-sectional images by iteratively maximizing the posterior probability distribution of images. This method is particularly useful in reducing metal artifacts of reconstructed images where projection data are partially blocked by metals in the patient body [14]. This method is also effective in low dose CT because the reconstruction process becomes more stochastic because of the low dosage data acquisition process [7, 10, 15]. Although the IR has become commercially available in clinical CT systems [7], this method is computationally very expensive [2, 4]. In the FBP, images are typically reconstructed and available for clinicians for review as fast as several seconds. In the IR, image reconstruction takes about 20 to 30 minutes depending upon scanning and reconstruction techniques. In addition, the IR typically needs a priori term because the posterior probability distribution

of images is converted to a likelihood term and a priori term by Bayesian framework [9]. The likelihood term is usually determined by the given sinogram data and the current estimate of images. However, the priori term is more complex and it needs to account for priori information about reconstructed images. Total Variation (TV) and MRFs (Markov Random Fields) [2, 4, 9], are often used in the priori term but they are too simple to represent the truly complex nature of the reconstructed images in which a more complex model needs to be built to achieve higher image quality of the reconstructed images.

CT image reconstruction problem is considered one of inverse problems and they have been an active research area using deep learning since the inception of AlexNet [16] in 2012. Convolutional Neural Network (CNN) has been proven effective in image de-noising, super resolution and image reconstruction problems [17, 18, 19, 20, 3, 21, 22, 23]. The CNN usually creates superior results compared to conventional model based algorithms [14]. Among many variations of the CNN network structures, the most popular choice of neural network is the U-Net [24] for inverse problems. The U-Net has down sampling, up sampling operations along with skipped connections within the network, so that it can learn overall structures of images as well as minor details localized in some areas of images [18, 25]. Skipped connections provide an ability of residual learning where the network can focus on learning noise characteristics and they usually improve Peak Signal to Noise Ratio (PSNR) and Structure Similarity (SSIM) [26]. Although the U-Net was invented to solve biomedical image segmentation problems, this network structure is now widely used in many image processing problems because of the effective learning capability [27, 28, 29]. Many research projects have been conducted with the CNN and the U-Net, however, the majorities of them have been limited in image space processing where the CNN was used to learn a mapping between a true image data set and a noisy image data set [17, 18, 25, 30]. A problem in this approach for inverse problems is that conventional inverse operators such as bicubic interpolation and the filtered backprojection are used to bring data from one domain to the other and important information can be lost within

these operations [31, 32, 33, 34]. This does not seem to be a major obstacle in super resolution[33, 35]. But there are many non-linear operations within CT reconstruction and it is not clear how much the CNN can compensate the lost information in the FBP operation as well as in non-linear operations before and after the FBP. In addition, the CNN is a useful supplement to be used with the FBP as a post processing de-noising operation. But the results are not as good as the ones from the IR, particularly with low dose CT data and with presence of metal artifacts [10, 11, 12, 13]. The performance of the FBP operator deteriorates with these challenging circumstances and the CNN performance degrades as the FBP operator's performance deteriorates. The CNN is still one time operation meaning that it is a mapping from noisy image data set to true image data set and it does not iteratively reach optimized results. CT reconstruction has many complex non-linear operations and iterative optimization has been proven the effectiveness to obtain the best image quality of reconstructed images [9, 10, 11, 12, 13].

The conventional iterative reconstruction is a minimization problem. There are two terms in the problem. One is the likelihood term and the other is the priori term as described above. Typically, the likelihood term uses sinogram and images and the priori term only uses images. The priori term is also called a regularization term and it is usually used with a regularization parameter to control the strength of the term. Primal-Dual Hybrid Gradient algorithm (PDHG) introduced by Chambolle and Pock in 2011 [36] is distinct from the conventional iterative reconstruction. The PDHG forms the iterative reconstruction problem as a min-max problem where the likelihood term is maximized while the priori term is minimized. Rather than finding a minimum point of a minimization problem, the PDHG tries to find a saddle point of a min-max problem. Images are called primal variables and sinograms are called dual variables in CT reconstruction problem in the PDHG although this method is used in wide variety of inverse problems. In addition, the PDHG uses a proximal operator, so that it can handle a non-differentiable prior term. Learned Primal-dual Reconstruction introduced by Adler et al. [31] is an extension of the

PDHG where the proximal operator is replaced by a CNN. While the Learned Primal-dual Reconstruction was demonstrated to archive the best PSNR and SSIM among the U-Net and the Total Variation, the limitation of the method is that it takes many epochs to reach an optimal point. It sometimes needs about 700 to 800 epochs for convergence according to our in-house evaluation of this method. This is likely caused by the bi-level optimization where it needs to optimize both primal and dual variables.

The CNN is not the only popular type of neural network in use for artificial intelligence community. One of the limitations of the CNN is that the input needs to be fixed-size and it cannot handle arbitrary length of input data. The Recurrent Neural Network (RNN) is an alternative choice for time series data and sequential data such as stock price predictions, autonomous driving systems and natural language processing tasks [3, 37]. In the RNN, a memory cell or a group of neurons is repeatedly used. The memory cell preserves some state across time steps. The conventional RNN is called basic RNN where there is only one simple memory cell repeatedly used. If there are multiple memory cells used in each time step, the RNN is called deep RNN [3, 37]. The backpropagation step for the RNN is distinct from the CNN. In the CNN, the output of the final layer of the network is used to compute the error which is back-propagated to the network. In the RNN, not only the final output but also the output of each time step is used to compute the error and the error in each time step is back-propagated to the network. This technique is called "Backpropagation Through Time" or BPTT [38, 39, 40, 41]. Furthermore, the RNN faces unique challenges with time series and sequential data. If a sequence is too long, the RNN will have a problem carrying information from earlier time steps to later ones. In this case, the RNN may omit important information from the beginning. Also, a vanishing or an exploding gradient problem may occur with a long sequence. The CNN has a similar problem when the layer of the network becomes too deep. In order to reduce these problems, the Long Short-Term Memory (LSTM) was proposed in 1997 [42]. In this approach, the memory cell is replaced by a LSTM cell which has more complex

sets of neurons in addition to gates that are internal mechanisms to regulate the flow of information. The basic RNN has been criticized because it tends to keep only short term information and it cannot preserve long term memory which happened many time steps ago. The LSTM cell has a forget gate which discards minor details. Thus, the RNN can preserve only important long-term information [43, 44]. Gated Recurrent Unit (GRU) was proposed by Cho et al. in 2014 [45] to simplify the LSTM but it is also as effective as the LSTM. The LSTM keeps the short-term state and the long term state. But, the GRU has only one hidden state to serve for both the short term and the long term memory. It has the same number of gates within the memory cell but it has less number of neurons. Thus, the GRU is generally simpler and faster than the LSTM [45]. Almost all state of the art results based on the RNN are archived with these two types of RNN networks [37].

The gradient descent (GD) algorithm is the simplest and frequently used optimization algorithm for deep learning [3]. Stochastic Gradient Descent (SGD) and Mini-batch Gradient Descent (Mini-batch GD) are under the same umbrella but they use different number of training data in each optimization [46]. Typically, either SGD or Mini-batch GD is used because the size of data in each optimization is small enough, so that the computational memory usage can be manageable [3]. However, these methods are often trouble navigating ravines in an optimization space, especially areas where surface curves are much sharper in one dimension than the others, which are common in the vicinity of local optima [47]. In such scenarios, these methods oscillate across the slopes of the ravine, which only makes hesitant movements toward the local optimum [47].

Momentum [47] is a useful method to accelerate GD-based algorithms in the right direction and reduces oscillation. The momentum method is essentially described as rolling a ball downhill. The ball accumulates momentum as it rolls over the hill, getting faster and faster along the way. The momentum increases toward the dimensions where the gradients continue to point to the same directions, and it decreases toward the dimensions

where the gradients frequently change their directions. Subsequently, it converges faster and reduces oscillation. RMSprop (Root Mean Square prop) [48] and ADAM (ADAptive Moment estimation) [49] are extensions of the momentum method and they are widely used in deep learning community [3]. These optimization algorithms are all first order optimization methods and they give a linear convergence. In contrast, Newton method is a second order optimization method [50, 51] and it gives a quadratic convergence. This method is slightly complex to understand, but it can converge with a much fewer iterations. Also, it does not need a step size parameter. However, one of the downsides of this algorithm is that it requires to compute a Hessian matrix whose size is $O(d^2)$ where $d$ is the dimension of the images [3, 50]. The image size of typical CT images is usually 512x512, therefore a Hessian matrix would consume 256GB of GPU memory which is not practical to allocate. GD-based algorithms only require $O(d)$ size of matrix which would be just 1MB of GPU memory. Another downside of Newton method is that the Hessian matrix needs to be inverted to compute an optimization step. If the matrix size is too big, it would not be practical either. These are the primary reasons why this method is not frequently used for image processing tasks. Quasi-newton method is known as 1.5 order optimization algorithm and it approximates an inverted Hessian matrix. Because the approximated matrix is already inverted, the method does not need to compute the inversion of the matrix. However, the matrix size is still $O(d^2)$. There is a method called Limited-memory BFGS which is a memory efficient version of quasi-newton method [50]. This method only requires $O(d)$ matrix to compute an optimization step and it is feasible to use in image processing tasks.

In this chapter, we propose a new deep RNN to implement the iterative reconstruction (IR) for CT image reconstruction. In analogous to the IR, this deep RNN has a likelihood term and a priori term as RNN cells. The likelihood cell makes sure reconstructed images are consistent to the input sinogram. The priori cell is a regularization term to prevent the likelihood cell overfit the input data. While the conventional IR has a simple loop

to repeat the optimization, the proposed deep RNN keeps a hidden state inside and this state is passed from the current iteration to the next. Also, the deep RNN has a CT forward projection and a CT backprojection inside in each iteration and it learns inaccuracies of these projections as well. Unlike the CNN based methods, each cell or each group of neurons in the deep RNN is shared across iterations. Therefore, the number of weights stay low even if there are many iterations in the algorithm. This helps the algorithm not overfit the input data because the number of training data going through each cell is effectively higher due to the repetition of the RNN cells. In addition, this chapter introduces a new RNN cell called Gated Momentum Unit (GMU). The deep RNN has the GMU inside to implement the momentum method. We still use ADAM as our optimizer and this should not be confused with the GMU. We use the GMU in the design of the RNN network to accelerate the convergence and ADAM is used to optimize the network. In addition, the momentum method is implemented with the GMU along with a forget gate to preserve long term memory. Without having a forget gate, the RNN would be more short term focus and would leave out important long term information. To make the momentum more efficiently calculated, a forget gate is used to regulate the flow of information.

The organization of this chapter is the following. The detailed method of the deep RNN and the GMU will be discussed in the following section. In the third section, qualitative and quantitative evaluation results will be shown. In the fourth section, we will discuss the conclusion of this chapter.

## 2.2 Methods

### 2.2.1 Iterative Reconstruction by deep RNN

For the sake of simplicity, we consider only the two-dimensional CT image reconstruction but our approach can be extended relatively straightforward to the three-dimensional reconstruction.

In CT image reconstruction problem, reconstructed images $x$ and signogram $y$ are stochastic processes and a Bayesian or MAP (Maximum A Posteriori) approach have been frequently used to model this problem as follows [9].

$$\begin{aligned} x^* &= \arg\max_x \big\{ \log p(x|y) \big\} = \arg\max_x \big\{ \log p(y|x) + \log p(x) \big\}, \\ &= \arg\min_x \big\{ J(y,x) \big\}, \quad \text{s.t.} \quad J(y,x) = -\log p(y|x) - \log p(x), \end{aligned} \tag{2.1}$$

where $x \in X$ and $y \in Y$ are sampled from image space $X$ and sinogram space $Y$ which are usually Hilbert spaces. $J(y,x)$ is the objective function to minimize to obtain $x^*$. In the objective function, the first term is the likelihood term and the second one is the priori term. The X-ray scan acquisition process including CT scanning follows the Poisson process and it is often approximated by a Gaussian model [2, 4] as follows.

$$p(y|x) = Z \exp^{-\frac{1}{2}(y-Ax)^T D(y-Ax)}, \tag{2.2}$$

where $Z$ is a normalization constant and $A$ is a linear or possibly a non-linear projection operator that captures the imaging geometry. $A$ is also called forward projection in contrasting to the backprojection process. $D$ is a diagonal positive definite matrix related to the Poisson noise in $y$. Each element $d_i$ in $D$ is the inverse of the variance of the projection measurement, such that $d_i = 1/\sigma_{y_i}^2$ where $\sigma_{y_i}^2$ is the variance of the projection

measurement at location $i$. The $d_i$ shows the credibility of the projection measurement. If a particular measurement $y_i$ is photon-starved by some metal object in patient body, the corresponding $d_i$ becomes reduced [9] and the contribution of the particular measurement to the image estimate would be lower.

This likelihood term described in equation (2.2) represents how close the estimates of reconstructed images are to the input sinogram. While the likelihood term is relatively simple to calculate given the acquisition geometry shown above, the priori term is more complex to calculate. The priori term incorporates various characteristics of the ideal reconstructed images to help produce good results while preventing the likelihood term overfit the training data. In the IR, TV and MRF priors are often used [9]. Recently, neural network based priors were also proposed [32].

### 2.2.2 The RNN and Gated Momentum Unit (GMU)

The equation (2.1) can be solved iteratively and one way to do this is through the gradient descent as follows [52].

$$x^{(i+1)} \leftarrow x^{(i)} - \epsilon \nabla_x \big\{ J(y, x) \big\} \big|_{x=x^{(i)}}, \tag{2.3}$$

where $\epsilon > 0$ is a step size of the gradient $\nabla_x \big\{ J(\cdot) \big\}$ and $i$ is the iteration index.

The gradient descent algorithm, however, is slow to converge and many techniques have been proposed to provide an acceleration. A relatively simple and successful technique is the momentum algorithm [3, 47] which can be expressed as follows.

$$\begin{aligned} x^{(i+1)} &\leftarrow x^{(i)} - m^{(i)}, \\ m^{(i)} &= \eta m^{(i-1)} + \epsilon \nabla_x \big\{ J(y, x) \big\} \big|_{x=x^{(i)}}, \end{aligned} \tag{2.4}$$

where $m^{(i)}$ is the momentum term at iteration $i$. $\eta < 1.0$ is a control parameter as to how much we want to use the past momentum to calculate the current momentum. This $\eta$ is usually set to 0.9 or a similar value [46]. If we compare the equation (2.3) and (2.4), the term $\eta m^{(i-1)}$ is new and this is a weighted past momentum which is used for an acceleration and avoiding an oscillation.

The momentum method certainly helps accelerate the convergence, however, this is an exponential decay of the past momentum and the rate of decay is deterministic and also a short term focus. The momentum method rapidly "forget" the past momentum and weighs on recent momentum values. In addition, when we use the momentum method in the design of a neural network, a vanishing gradient problem may occur especially if the number of iterations is high. In this case, later iterations of the RNN would learn more than the earlier iterations. In order to solve this problem, we introduce Gated Momentum Unit (GMU) in the RNN. The GMU is similar to the LSTM [42] and the GRU [45] but the GMU uses a CNN inside instead of a fully connected layer. The LSTM and the GRU have multiple fully connected layers along with multiple gates but what we handle is a spatial signal and using a CNN block instead of a fully connected layer is more appropriate. In fact, the GMU is relatively simple and it has a concatenation, a CNN block, and a forget gate. The forget gate helps keep the important long term information and discard or "forget" short term minor detail. The GMU can be described as follows.

$$
\begin{aligned}
x^{(i+1)} &\leftarrow x^{(i)} - m^{(i)}, \\
m^{(i)} &= C_1\Big(h^{(i-1)}, \epsilon\nabla_x\big\{J(y,x)\big\}\big|_{x=x^{(i)}}\Big), \\
h^{(i-1)} &= \gamma^{(i-1)}m^{(i-1)} + (1-\gamma^{(i-1)})h^{(i-2)}, \\
\gamma^{(i-1)} &= tanh(|m^{(i-1)}|),
\end{aligned}
\tag{2.5}
$$

where $C_1(\cdot)$ is a parametrized vector-valued function which is implemented by a CNN block in this research. $h^{(i)}$ is the momentum hidden state at iteration i. $\gamma^{(i)}$ is the forget

gate to regulate the follow of information. If the current momentum $m^{(i)}$ is small, the gate is closed to 0. When it is high, the gate is open to 1. A momentum is a derivative of the posterior distribution and it can be positive or negative. But we take the absolute value of $m^{(i)}$, so that the network can capture the important information while discarding insignificant detail. As we can see in the equation (2.5), when the gate is open at iteration $(i-1)$, the momentum $m^{(i-1)}$ is passed to the next iteration as the momentum hidden state $h^{(i-1)}$. When the gate is closed, the past momentum hidden state $h^{(i-2)}$ is passed to the current iteration as the momentum hidden state $h^{(i-1)}$ although the gate is not a step function but it is a hyperbolic tangent so that the transition is smooth from 0 to 1. Usually, sigmoid function is used for a gate in the LSTM and the GRU [42, 43, 44, 45]. However, our input range to the gate is $[0, +\infty)$. Thus, we chose the hyperbolic tangent instead for the forget gate.

When we compare the equation (2.4) and (2.5), there are roughly two differences. The first difference is that the GMU uses the momentum hidden state at (i-1)-th iteration instead of the momentum at the same iteration in the second line of the equation (2.4) and (2.5). In addition, the control parameter $\eta$ is removed from the GMU. Alternatively, it uses a concatenation and a CNN block. Our internal research shows that a concatenation and a CNN block would work better in most cases than a weighting average of multiple images in deep learning. This is consistent to the best practices found in deep learning community [3].

Finally, to implement the equation (2.5) in the deep RNN, the gradient term of the objective function is replaced by other CNN blocks. The likelihood term and priori term are replaced by CNN block $C_2(\cdot)$ and $C_3(\cdot)$ respectively.

$$\begin{aligned} \epsilon \nabla_x \big\{ J(y,x) \big\} \big|_{x=x^{(i)}} &\cong C_2(Err(y, x^{(i)})) + C_3(x^{(i)}), \\ \text{s.t.} \quad Err(y, x^{(i)}) &= A^+(y - Ax^{(i)}), \end{aligned} \tag{2.6}$$

TABLE 2.1: Configuration of three CNN blocks in each RNN cell.

| Name | Input channels | CNN layers | CNN channels | Activations | Output channels |
|---|---|---|---|---|---|
| $C_1(\cdot)$ | 65 | 5 | 64 | PReLU | 1 |
| $C_2(\cdot)$ | 1 | 5 | 32 | PReLU | 32 |
| $C_3(\cdot)$ | 1 | 5 | 32 | PReLU | 32 |

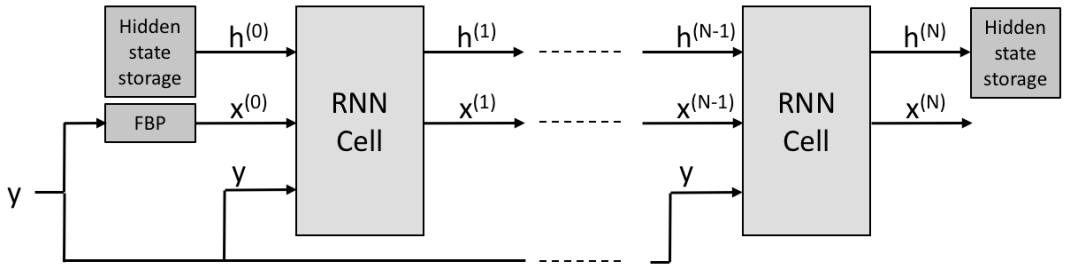



FIGURE 2.1: The network structure of the RNN GMU for CT image reconstruction. The RNN cell is shown in figure 2.2 in detail. The hidden state storage is a memory storage to store a momentum hidden state. The FBP is the filtered backprojection operator. $y$ is an input sinogram, $x^{(i)}$ is the current estimate of images at iteration $i$. $h^{(i)}$ is the momentum hidden state at iteration $i$ which is passed from the current iteration to the next.

where $Err(\cdot)$ is an error image generator. $A$ is the forward projection and $A^+$ is the backprojection. The error image generator shows how close the current estimate of an image $x^{(i)}$ is to the input sinogram $y$ in image domain. If we plug in equation (2.6) into equation (2.5), we obtain the following expression and this is how the deep RNN is designed for CT image reconstruction.

$$x^{(i+1)} \leftarrow x^{(i)} - C_1\Big(h^{(i-1)}, C_2(Err(y, x^{(i)})), C_3(x^{(i)})\Big). \tag{2.7}$$

Table 2.1 shows the network configurations of each CNN block $C_1, C_2$, and $C_3$ in the RNN. We use PReLU as our activation function in this research. ReLU has been one of the keys to the recent successes in deep learning [3] and PReLU is one of the ReLU family. It is parametrized in both the negative and the positive input values. The parameter of PReLU is trained as part of the network optimization process.

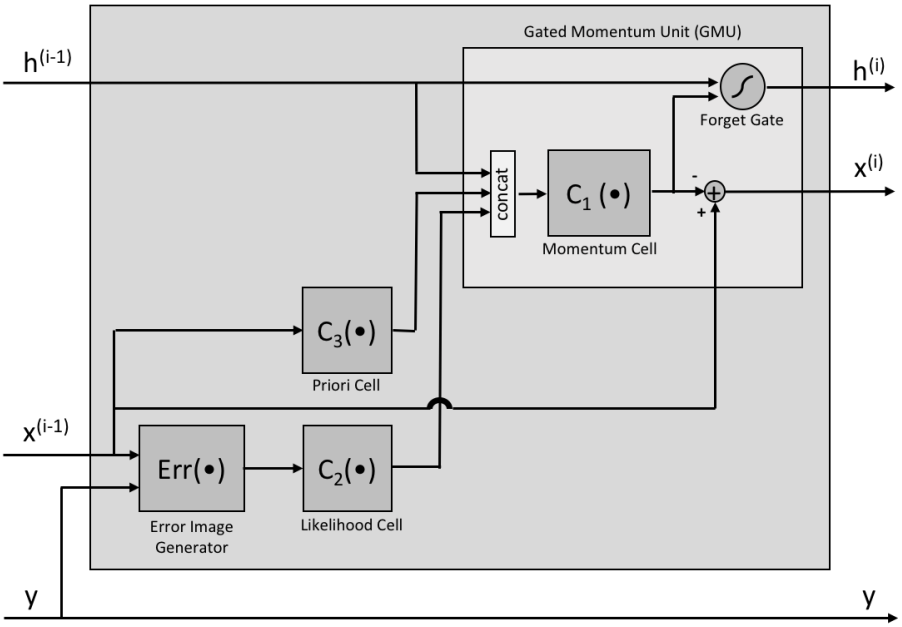



FIGURE 2.2: The network structure of each RNN cell. The likelihood cell, the priori cell and the momentum cell are implemented by the CNN. The GMU consists of a concatenation, the momentum cell and a forget gate. The forget gate is used to keep an important long term memory while it discards a minor short term detail.

Figure 2.1 shows the overall design of the deep RNN. The input to the RNN cell are sinogram $y$, the initial estimate of image $x^{(0)}$ and the initial momentum hidden state $h^{(0)}$. The FBP is used to create $x^{(0)}$ and $h^{(0)}$ is restored from the hidden state storage. $x^{(N)}$ is the final estimate of image and $h^{(N)}$ is the final hidden state which is stored into the storage for the next reconstruction job. Figure 2.2 shows the network structure of each RNN cell. It has input data of sinogram $y$, the intermediate estimate of image $x^{(i-1)}$ and the intermediate momentum hidden state $h^{(i-1)}$ as its input data. The output data are again the same data but at the i-th iteration. The GMU has the concatenation, the CNN block as the momentum generator and the forget gate.

### 2.2.3 Backpropagation

Lastly, this sub-section discusses how to backpropagate errors to the RNN in training. Before discussing the backpropagation, let us show the RNN in a mathematical form to

help explain this sub-section. In the RNN, $y$ is the input, and image $x$ and momentum hidden state $h$ are the output and they are sequences such that $x = \{x^{(0)}, x^{(1)}, \ldots, x^{(I)}\}$ and $h = \{h^{(0)}, h^{(1)}, \ldots, h^{(I)}\}$, where $I$ is the total number of iterations of the RNN. Each RNN cell can be expressed as a parameterized vector-valued function $f_\theta$ as follows.

$$\begin{aligned} x^{(i)}, h^{(i)} &= f_\theta(x^{(i-1)}, h^{(i-1)}; y), \\ x^{(I)}, h^{(I)} &= f_\theta \circ f_\theta \circ \ldots f_\theta(x^{(0)}, h^{(0)}; y), \end{aligned} \tag{2.8}$$

where $x^{(0)}, h^{(0)}$ are the initial estimate of image $x$ and the initialized hidden state of the RNN respectively. $x^{(i)}, h^{(i)}$ are the intermediate results and $x^{(N)}, h^{(N)}$ are the final results of the RNN. $\theta \in \Theta$ is some parameter space to define the behavior of the function $f_\theta$.

The CNN does not have a sequence of output. Therefore, the backpropagations in the CNN and in the usual RNN are different. In the CNN, only the final output is used to compute the error which is backpropagated. In contrast, in a usual RNN, the error at each time step is computed and backpropagated to the network. This type of backpropagation in a RNN is called Backpropagation Through Time (BPTT) as explained in the previous section [38, 39, 40] and the following equations show how to optimize the RNN with the BPTT. As we discussed in the earlier part of this section, we would like to maximize the posterior distribution of image $x$. The posterior distribution is

$$p(x|y) = \prod_{i=1}^{I} p(x^{(i)}|x^{(i-1)}, x^{(i-2)}, \ldots, x^{(0)}, y; \theta). \tag{2.9}$$

Here is how to optimize the parameter $\theta$ in the RNN by the BPTT.

$$\begin{aligned} \theta_{MAP} &= \arg\max_\theta \Big\{ \log p(x|y) \Big\}, \\ &= \arg\min_\theta \Big\{ -\sum_{i=1}^{I} \log p(x^{(i)}|x^{(i-1)}, x^{(i-2)}, \ldots, x^{(0)}, y; \theta) \Big\}, \end{aligned} \tag{2.10}$$

In order to implement equation (2.10), we assume the optimal reconstructed image $x^{(i)}$ has the shortest mean squared distance to the true image. With this assumption, we usually use a loss function to maximize the probability distribution or minimize the negative log of the distribution. The loss function at the i-th iteration can be defined as follows.

$$L^{(i)}(\theta) = \mathbb{E}_{(\bar{x}^{(i)}, x^{(i)}) \sim p_j^{(i)}}[\|\bar{x}^{(i)} - x^{(i)}\|_2^2], \tag{2.11}$$

where $p_j^{(i)}$ is a joint probability distribution of the training data set (true image data set) and the generated image data set by the RNN at the i-th iteration and it is in a compact metric space such as Hilbert space. $\bar{x}^{(i)}$ is a training sample and $x^{(i)}$ is a generated sample by the RNN at the i-th iteration respectively. In practice, however, the joint probability distribution is often inaccessible [53]. Instead, we just have a finite set of samples. Therefore, we usually replace the distribution with its empirical counter-part. Also, the expectation can be implemented as simple as Mean Square Error (MSE). Thus, the loss function is replaced by the empirical loss as follows.

$$\begin{aligned} \hat{L}^{(i)}(\theta) &= \frac{1}{N} \sum_{n=1}^{N} [\|\bar{x}_n^{(i)} - x_n^{(i)}\|_2^2], \\ \text{s.t.} \quad & x_n^{(i)} = f_\theta(x_n^{(i-1)}, h^{(i-1)}; y_n), \end{aligned} \tag{2.12}$$

where, $\hat{L}^{(i)}(\theta)$ is the loss at the i-th iteration parametrized by $\theta$. $N$ is the number of samples and $n$ is the sample index. $\bar{x}_n^{(i)}$ is the intermediate baseline image at the i-th iteration if we have. $x_n^{(i)}$ is the intermediate result from the RNN.

While the BPTT makes sense for usual RNN applications such as natural language processing tasks, there are two issues with this type of backpropagation in the context of CT image reconstruction. The first issue is that we don't have an intermediate baseline image $\bar{x}^{(i)}$ available for training. We just have the final baseline image $\bar{x}^{(I)}$ for comparison although we could create intermediate baseline images by interpolating the FBP estimate

$\bar{x}^{(0)}$ and the final $\bar{x}^{(I)}$ to provide the RNN more detailed optimization guideline. However, the second issue, which is more serious, is that it would make the backpropagation executed I times instead of once for each training. In this case, the BPTT would significantly slow down the training process. Thus, in this research, we chose not to use BPTT. Rather we chose to run the backpropagation in the CNN fashion. Here is how we choose to run backpropagation in our RNN.

$$\begin{aligned} \theta_{MAP} &= \arg\min_{\theta} \Big\{ -p(x^{(I)}|x^{(I-1)}, x^{(I-2)}, \ldots, x^{(0)}, y; \theta) \Big\}, \\ \hat{L}(\theta) &= \frac{1}{N} \sum_{n=1}^{N} [\|\bar{x}_n^{(I)} - x_n^{(I)}\|_2^2], \\ &\text{s.t.} \quad x_n^{(I)} = f_\theta \circ f_\theta \circ \ldots f_\theta(x_n^{(0)}, h^{(0)}; y_n), \end{aligned} \tag{2.13}$$

A pseudo-code of the RNN optimization is shown in Algorithm 1.

**Algorithm 1** Deep RNN with Gated Momentum Unit for CT image reconstruction

**Require:** Input sinogram $y$, the initial estimate of images $x^{(0)}$, the forward projector $A$, pseudo inverse $A^+$, the initial momentum hidden state $h^{(0)}$, the RNN network paramater $\theta$, the number of iteration I, the number of mini-batch B.

**for all** $i \in 1, ..., I$ **do**

- $e^{(i)} \leftarrow A^+(y - Ax^{(i-1)})$ ▷ Error image generation
- $x_1^{(i)} \leftarrow C_2(e^{(i)})$ ▷ Likelihood cell
- $x_2^{(i)} \leftarrow C_3(x^{(i-1)})$ ▷ Priori cell
- $m^{(i)} \leftarrow C_1\Big(concat(h^{(i-1)}, x_1^{(i)}, x_2^{(i)})\Big)$ ▷ Calculate the momentum at the iteration i
- $x^{(i)} \leftarrow x^{(i-1)} - m^{(i)}$ ▷ Calculate the estimate of images at iteration i
- $\gamma^{(i)} \leftarrow tanh(|m^{(i)}|)$ ▷ Forget gate
- $h^{(i)} \leftarrow \gamma^{(i)} * m^{(i)} + (1 - \gamma^{(i)}) * h^{(i-1)}$ ▷ Calculate the hidden state at iteration i

**end for**

$\theta \leftarrow AdamOptimizer(\nabla_\theta \sum_{b=1}^{B} Loss(b))$ ▷ network parameters of the RNN

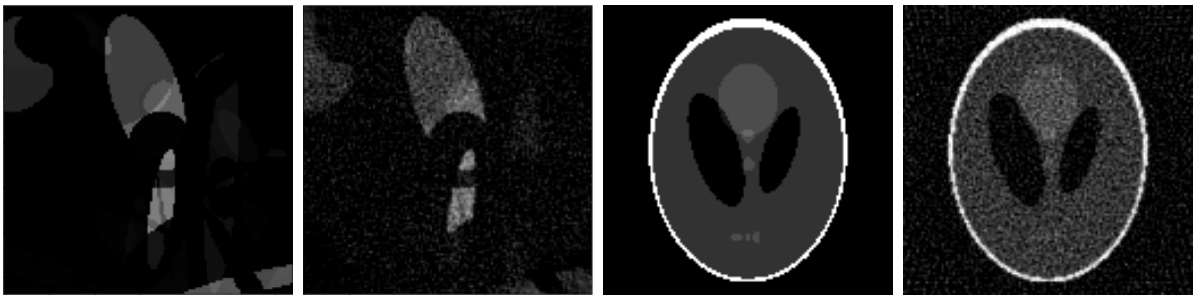
## 2.3 Experiments

In this chapter, we conducted two experiments to evaluate the proposed method. First, we evaluated the RNN along with other well-known methods with a simulation data.

Second, we conducted our evaluation with real CT image data. We used PSNR and SSIM [26] along with a visual inspection of images to understand the effectiveness of our proposed method.

In these experiments, we used the exact same training schemes for all algorithms, so that we could evaluate them equivalently. The image matrix size is 128x128 for the simulation study and 512x512 for the real CT image study. The number of iterations of the RNN was 10 in simulation and 5 in real CT data respectively. We took 1,000 epochs for all algorithms in the simulation study and 200 epochs in the real CT image study. We used ADAM optimizer as the optimization algorithm for all deep learning based methods in this chapter. We used 0.001 as the learning rate along with a cosine decay. We did not use Batch Normalization [54] in any of the evaluated neural networks in this chapter. Instead, we normalized input images to [0, 1]. The mini-batch size was set to five in simulation and one in real CT data. We used a Nvidia GeForce RTX 2080 GPU and the tensorflow python package to implement the RNN.

For the comparison, we used Filtered Back-Projection (FBP), Non-Local Mean Filter (NLMF) [55], Total Variation (TV) [56], U-Net [24], Learned Primal-dual Reconstruction (Learned PDR) [31] and Learned PDHG methods [31]. The FBP images are used as the input of our RNN implementation which is denoted as "RNN GMU" in this section. We used the ODL python package [56] to perform the forward projection and the back-projection for CT image reconstruction. The NLMF is an image space post processing filter, which is to find image patches across the entire image and averages these patches to reduce noise [55]. We used the suggested hyper-parameters for this filter described in the original paper [55]. We used a TV implementation from the ODL python package [56], which is an iterative reconstruction with a smoothness regularization term. We also used the U-Net [24] as our image space post processing algorithm counter-part to compare against. The Learned PDR is a state of the art CT image reconstruction algorithm proposed by Adler et al. [31]. It is a combination of model driven and data driven method



(A) (B) (C) (D)

FIGURE 2.3: Training and validation data for simulation study. (a) Random ellipses as training data, (b) Training data after Gaussian noise added, (c) Shepp-Logan phantom, (d) Noised Shepp-Logan phantom

and it uses a forward and backward projection inside. Learned PDHG is also proposed by Adler et al. [31] and it is quite similar to the Learned PDR. The only differences are the primal and dual variables have higher dimensions in the Learned PDR. Also, one proximal operator is used in each iteration in Learned PDHG. But Learned PDR uses different proximal operators at each iteration where the operators are allowed to differ. All of the evaluation methods in this chapter used Mean Square Error (MSE) as their losses.

### 2.3.1 Simulation study

For simulation study, we used images with random ellipses for training [31, 32]. An example training data is shown in figure 2.3. Each training image has a handful random sized and random pixel valued ellipses within the image. The number of training images used in this research were 500. We added 10% of Gaussian noise on all input images. For the validation, we only used one image for validation and testing, which contains the Shepp-Logan phantom [57]. The validation image example is shown in figure 2.3.

The PSNR and SSIM results are shown in table 2.2. Non-local mean filter did not improve PSNR and SSIM much over the FBP and a further experiment result showed that these two numbers could be slightly improved by further tweaking hyper parameters for the

algorithm. However, we chose to use the default hyper parameters stated in the original paper for a gray-scale image [55]. The TV did not improve the numbers much either while the U-Net produced higher PSNR and SSIM than the TV. This was expected because, as stated above, the TV prior is too simple and does not capture the complex nature of CT images. The RNN GMU results are better than Learned PDR by 2.25db and Learned PDHG by 7.93db.

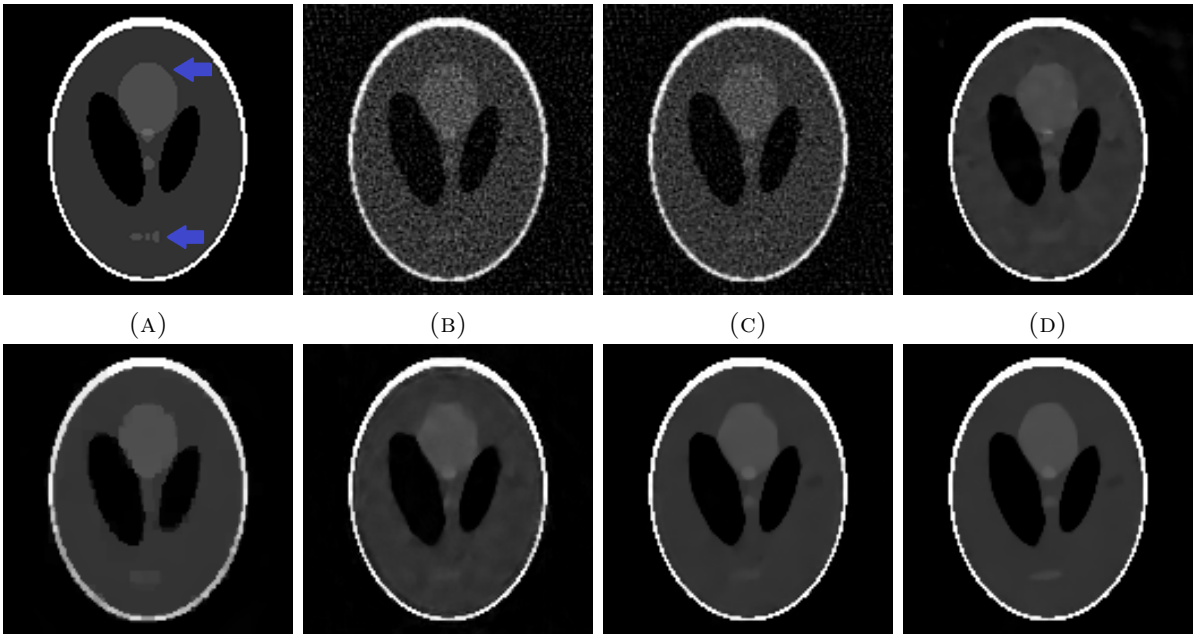
Figure 2.4 showed validation image results. The blue arrows indicates interesting areas where each algorithm performed differently. As you can see from the FBP result, this simulation data are quite challenging. Although the PSNR and the SSIM are high in the U-Net, the validation result show that this image space post processing algorithm is not sufficient to fully compensate information loss in CT image reconstruction. The TV result, on the other hand, is too smooth which is because of the regularization term. The result of Learned PDHG is not satisfactory and it had a similar level of improvement as the U-Net had. The results of Learned PDR and the RNN GMU are analogous but they have some differences. Particularly, the gray circle indicated by the upper blue arrow in Learned PDR result has a dent but the RNN GMU result does not have a dent. In addition, the three gray circles indicated by the lower blue arrow are merged into one circle in all of the methods because of too much noise added in the simulation data. The merged circle is almost fading out in the learned PDR result. But the intensity of the circle in the RNN GMU result is relatively maintained.

### 2.3.2 Real CT image study

For real CT image evaluation, we used LIDC/IDRI data set [58]. We used 1041 images for training, one image for validation, and 278 images for testing. The test is denoted as quantitative evaluation in the following. The pixel values of all images used in this study are limited to [-1000, +1000] HU (Hunsfield Unit). The pixel values out of this range are

TABLE 2.2: Quantitative evaluation on Shepp-Logan phantom data

| **Method** | **PSNR (db)** | **SSIM** |
|---|---|---|
| FBP | 21.68 | 0.49 |
| Non-local mean filter | 22.25 | 0.54 |
| Total variation | 24.17 | 0.95 |
| U-Net | 32.68 | 0.95 |
| Learned PDHG | 30.98 | 0.95 |
| Learned Primal-dual | 36.66 | 0.99 |
| RNN GMU | 38.91 | 0.99 |



(A) (B) (C) (D)

(E) (F) (G) (H)

FIGURE 2.4: CT image reconstruction results on Shepp-Logan phantom. (a) True image with two blue arrows indicating areas where each algorithm performed differently, (b) Filtered backprojection (FBP) after Gaussian noise added, (c) Non-local mean filter (d) U-Net, (e) Total Variation (TV), (f) Learned PDHG, (g) Learned Primal-dual Reconstruction (Learned PDR), (h) RNN GMU.

TABLE 2.3: Quantitative evaluation on LIDC/IDRI data set (278 test images)

| **Method** | **PSNR (db)** | **SSIM** |
|---|---|---|
| FBP | 24.58 | 0.60 |
| Non-local mean filter | 24.59 | 0.60 |
| Total variation | 27.61 | 0.85 |
| U-Net | 31.58 | 0.90 |
| Learned PDHG | 28.01 | 0.86 |
| Learned Primal-dual | 33.58 | 0.92 |
| RNN GMU | 35.36 | 0.94 |

capped by the range. This HU value range is further normalized to [0, 1] and we did not use Batch Normalization as we did neither in the simulation study. Each training step chose a mini-batch of images randomly from the training data set. Input noisy images were created by adding a 0.02% Gaussian noise to simulate CT equipment noise. This level of Gaussian noise was chosen empirically. On top of it, a Poisson noise was added as well to simulate the X-ray photon acquisition process. These two sets of noise were added not in the sinogram space but in the view data space, thus we needed to perform a negative log step to bring data back into the sinogram space. The number of views was chosen to 64 to simulate a sparse view Computed Tomography where the forward and backward projection processes become noticeably imperfect.

Table 2.3 shows the PSNR and SSIM results. As stated above, the number of quantitative evaluation data set was 278. Thus, this table shows the average PSNR and SSIM numbers for 278 images. All results are similar to the simulation results. The improvement of the RNN GMU over Learned PDR is negligible. This implies both algorithms may have converged to the same optimization point.

Figure 2.5 shows the validation results. The FBP results shows many streak artifacts because of the added noise and the limited number of views chosen for the study. The U-Net result shows the limitation of image space processing. The U-Net is well known for the learning capacity but it is used only in the image space post processing which

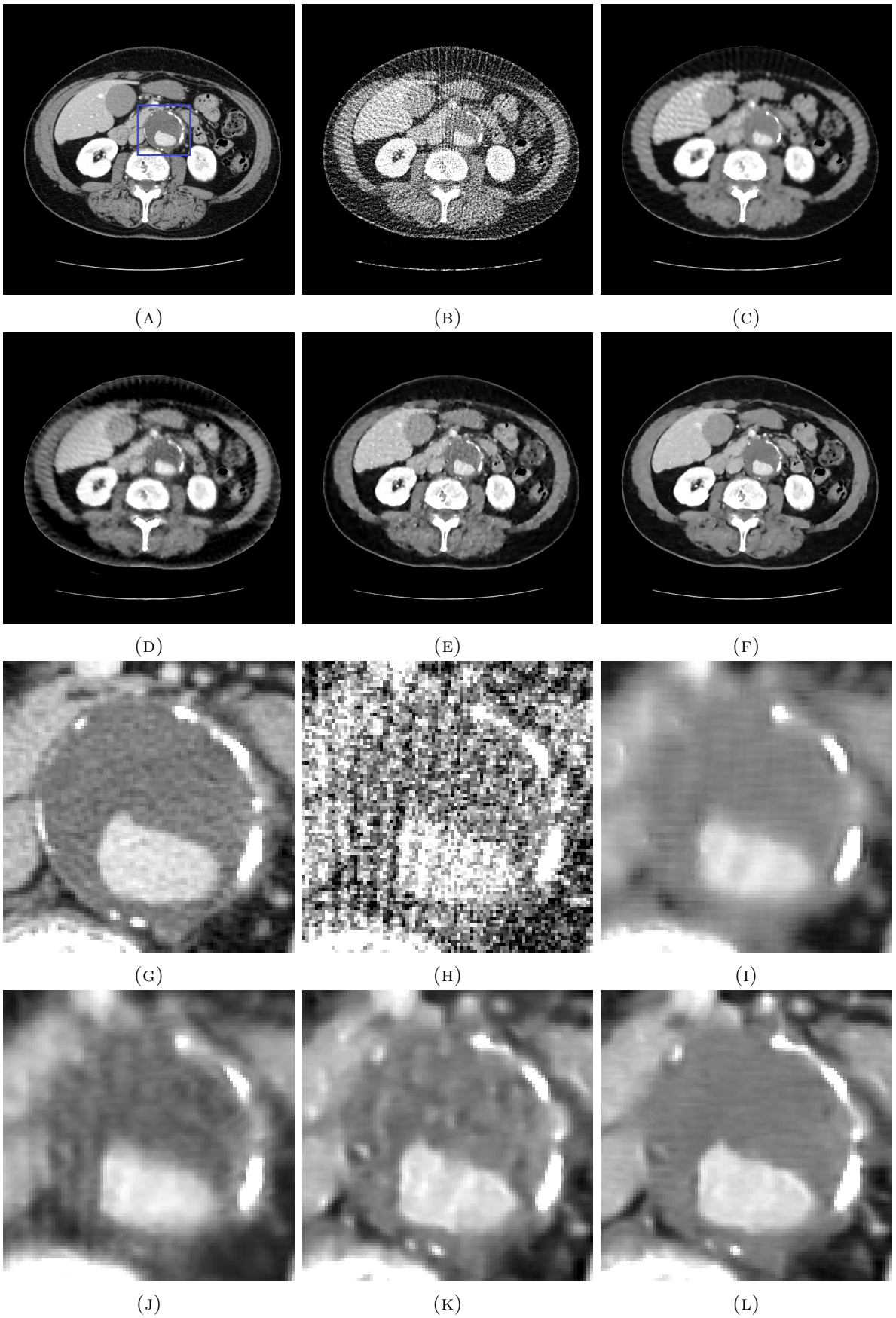
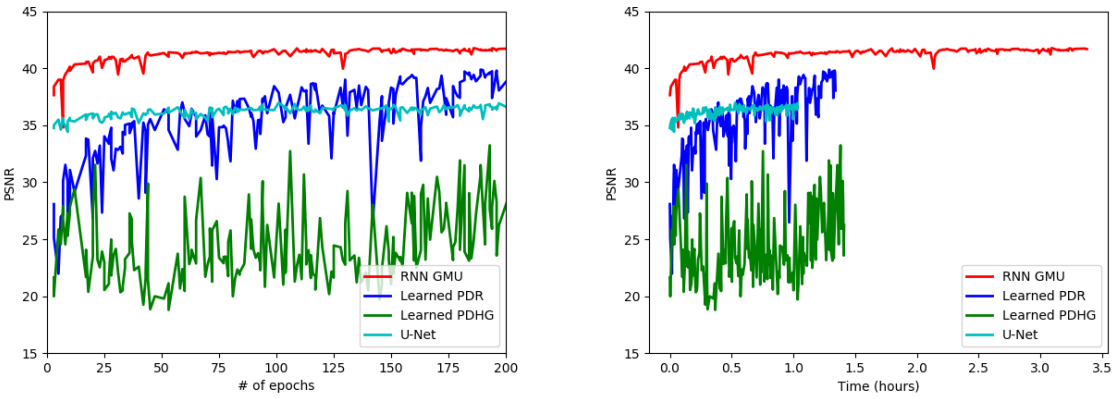



FIGURE 2.5: CT image reconstruction results on LIDC/IDRI data set. (a) True image, (b) Filtered backprojection (FBP) after Poisson and Gaussian noise added, (c) U-Net, (d) Learned PDHG, (e) Learned PDR, (f) RNN GMU, (g) True image (zoom), (h) FBP (zoom), (i) U-Net (zoom), (j) Learned PDHG (zoom), (k) Learned PDR (zoom), (l) RNN GMU (zoom)

is not sufficient to compensate the information loss in imperfect forward and backward projection. The results of the learned PDR and the RNN GMU are both good and they look very similar. We did not find any noticeable difference on these results. The last six images in the figure are zoomed versions of the same image set highlighted as a blue square in (a).

Figure 2.6 shows the PSNR improvement trends in the validation. Both figures show the improvement trends but (a) is over epochs and (b) is over time. As we can see, the learned PDR did not converge within 200 epochs while the U-Net and the RNN GMU took about 100 epochs for the convergence. This result shows a Batch Normalization (BN) [54] like improvement on the RNN GMU where BN helps speed up the convergence of a deep learning optimization process. While the RNN GMU seems to perform better than the the learned PDR, we would like to mention that the RNN GMU consumes more memory than the learned PDR. The RNN GMU consumes approximately 30% more GPU memory than the learned PDR. We believe this is due to the fact that tensorflow allocates many intermediate variables in the RNN even if the same set of weights are used in each RNN cell. These intermediate variables for the RNN are invisible to the user. Although the number of weights are much less in the RNN GMU because of the RNN cell repetition, the weight repetition would not help reduce the training time. The RNN GMU training time took almost twice as much as the learned PDR. The RNN GMU took about 3.5 hours to complete 200 epochs training, whereas the learned PDR took about 1.5 hours to complete the same number of epochs. However, the learned PDR did not converge within 1.5 hours, while the U-Net and the RNN GMU took about one hour to converge as shown in (b) of the figure. This time-wise figure shows how quickly the RNN GMU can converge.

There is one more finding in figure 2.6. The learned PDR was somewhat volatile in the training. Occasionally, the PSNR was significantly degraded during the training process and it recovered quickly. The same behavior is seen in Learned PDHG. We did not see




(A) (B)

FIGURE 2.6: Validation results with 200 epochs on LIDC/IDRI data set. (a) PSNR improvement trends over epoch, (b) PSNR improvement trends over time. The time for each method to complete 200 epochs varied.

the same level of volatility in the U-Net and the RNN GMU. While we are not sure the reason of the volatility in the learned PDR, we realized that it is very important for the learned PDR to use a learning rate decay such as cosine decay to converge well.

## 2.4 Summary

We have proposed the new algorithm for CT image reconstruction. The proposed method used the deep Recurrent Neural Network (RNN) to implement the iterative reconstruction (IR). The RNN framework was used to conduct the iterative optimization. The deep RNN has the likelihood term and the priori term in analogous to the IR as memory cells. Also, we introduced a new RNN cell called Gated Momentum Unit (GMU) as a memory cell which helps the network preserve important long term information while it discards insignificant short term detail. Unlike the Convolutional Neural Network (CNN), the RNN cell is repeatedly used in every iteration. This helps the RNN cell trained with more number of data sets and it prevents the algorithm overfit the input training data. The proposed method was evaluated in simulation data along with real CT image

data. The evaluation results showed that the new algorithm was effective to converge faster than any of the other well known methods. In the simulation study, the algorithm archived the best PSNR. In the real CT data study, the proposed method achieved the same level of Learned Primal-dual Reconstruction (Learned PDR), but it converged much faster than the learned PDR. In the study, we also found that the proposed method consumed more computational memory than other well known methods. This is caused by tensorflow python package we use to implement the RNN. As our future study, we plan to investigate to reduce intermediate variables allocated by the tensorflow to see how to reduce the memory footprint in the method. In addition, although the proposed method converged faster than the learned PDR, we found that training time for each epoch is longer to finish. This is likely caused by the deeper layers of each CNN block used in the RNN cell. We plan to review the CNN blocks to see if we can reduce the number of layers while maintaining the highest level of PSNR and SSIM. We hope this method will inspire other researchers and it will be further improved by the research community. Also, this method can be simply applied to other modalities such as Magnetic Resonance Imaging (MRI) by just replacing the forward and backward operator. We plan to evaluate this method with MRI data and other imaging modality data.

# Appendix

## Further qualitative results

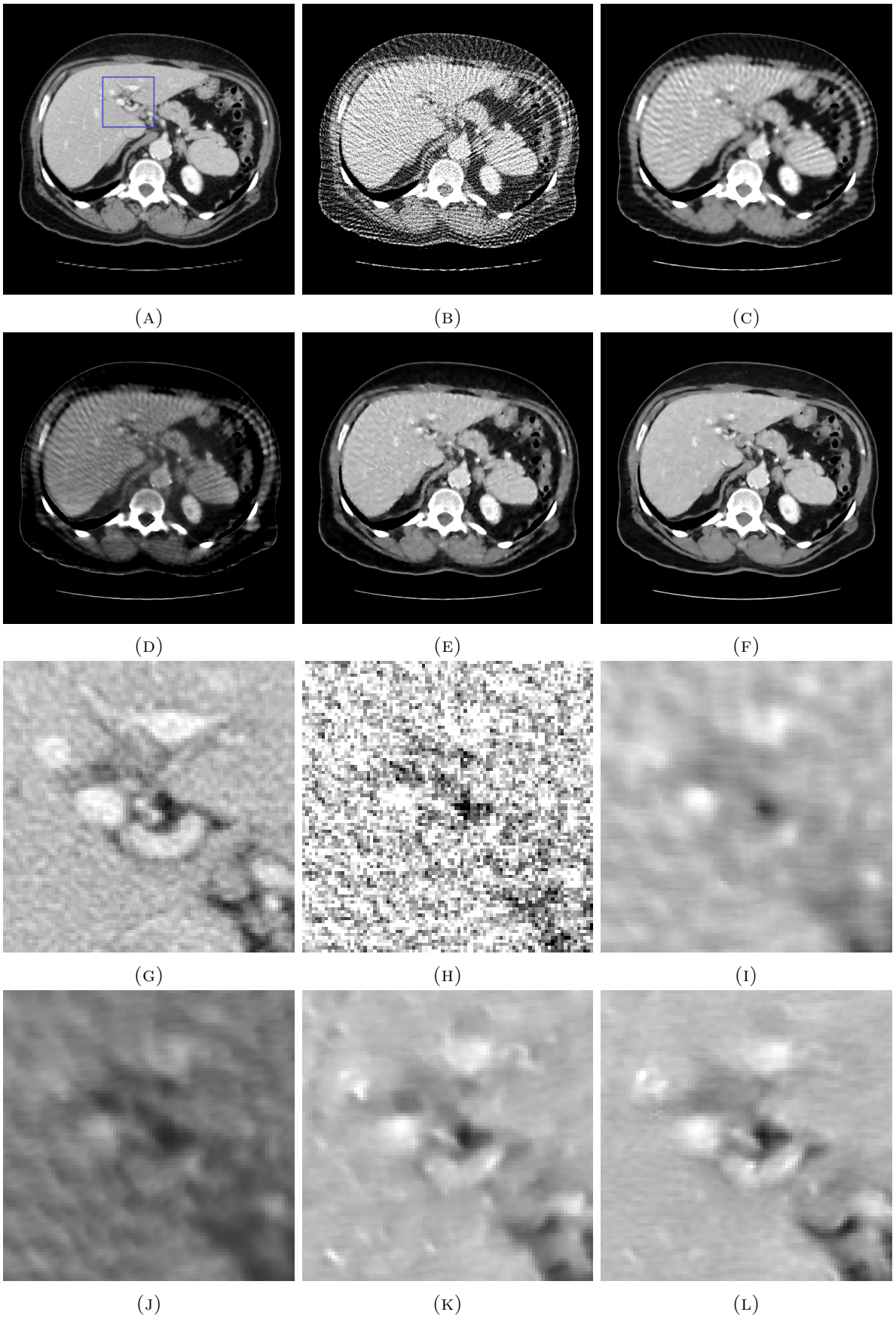



FIGURE 2.7: Another set of CT image reconstruction results on LIDC/IDRI data set. (a) True image, (b) Filtered backprojection (FBP) after Poisson and Gaussian noise added, (c) U-Net, (d) Learned PDHG, (e) Learned PDR, (f) RNN GMU, (g) True image (zoom), (h) FBP (zoom), (i) U-Net (zoom), (j) Learned PDHG (zoom), (k) Learned PDR (zoom), (l) RNN GMU (zoom)

# Chapter 3

# A Deep Recurrent Neural Network with FISTA Optimization for CT Metal Artifact Reduction

## 3.1 Introduction and Related Work

Computed Tomography (CT) imaging technology turned 50 years old in 2021. It is still evolving and is one of the most important medical imaging modalities in use today. CT scans have been used to diagnose many types of illnesses and detect injuries. Over the years, the measurement accuracy of medical and industrial CT systems has been greatly improved. These systems can measure patient bodies accurately up to $\pm 1$ Hounsfield Unit (HU) [2, 4, 72]. However, the existence of dental fillings, hip prostheses, and other metallic materials inside patient bodies create artifacts in CT images, and the measurement accuracy can deteriorate several hundred HUs [2, 4]. This measurement degradation not only impacts an X-ray dose calculation but also leads to misdiagnosis of illnesses and injuries [2, 4]. There are multiple causes of CT metal artifacts. First, the beam hardening effect creates metal artifacts in CT images [2, 4]. Beam hardening is a physical phenomenon that occurs when a polychromatic X-ray beam passes through a high-density object [2, 4].

Lower energy photons are more susceptible to X-ray attenuation when they pass through metallic objects [2, 4]. This results in reduced X-ray energy received at a CT detector [2, 4]. Second, X-ray photons generally follow Poisson statistics [2, 4]. When photons pass through high-density objects, not only the average number of photons is reduced but also the variance becomes changed because of the property of Poisson statistics [73]. Finally, metallic objects carried in human bodies often have sharp edges [2]. These edges create sharp boundaries between metallic objects and human anatomies in both CT sinogram data and CT images. These boundary pixels in images and sinogram data can create secondary artifacts when these pixels are modified by an image processing algorithm [74].

Many methods have been proposed for CT Metal Artifact Reduction (MAR). Kalender et al. proposed Linear Interpolation (LI) method [75]. This method assumes metal impacted regions in the sinogram domain are entirely unreliable, and it uses linear interpolation to replace these regions with neighboring unaffected regions in the sinogram domain. Therefore, this is a sinogram domain inpainting method. This method removes metal objects from images but introduces new secondary artifacts by interpolation. A part of the problem is the binary mask the method uses. It uses a binary mask in the sinogram domain called a metal trace to indicate where metallic objects are located in the sinogram domain. The binary values have sharp edges in boundary pixels, and these edges contribute to undesirable secondary artifacts. In addition, metal impacted regions in the sinogram domain are damaged by metal objects. However, there is some helpful information in the regions [2, 4]. Completely throwing away these pixel values seems wasteful. Meyer et al. proposed Normalized Metal Artifact Reduction (NMAR) [76]. This method creates a prior image and uses the image to improve the corresponding sinogram. This method again assumes the MAR is a sinogram domain inpainting problem, and it replaces metal impacted regions in the sinogram domain with the prior information. This method also uses a metal trace (binary mask). Thibault et al. proposed a recursive filter with Iterative Reconstruction (IR) to reduce metal artifacts [77]. This method is

an extension to the IR [9], and it runs low-pass filtering in metal impacted regions in the sinogram domain to improve the credibility of observation data (sinogram). Zhang et al. tackled the CT MAR problems by a constraint optimization [78]. A model-based method was used to build prior knowledge, and the method seems limited by the model-based prior information. A Deep Learning (DL) [79] based prior knowledge should work better for this method due to its powerful representation capability. Chang et al. proposed a method that combines an iterative reconstruction with the NMAR [80]. This method speeds up the iterative optimization process with prior information when a metal object is present in the sinogram domain. This method seems again limited by the model-based prior information.

Recently, many DL-based methods have been proposed for CT MAR. Zhang et al. proposed a Convolutional Neural Network (CNN) for CT MAR (CNN-MAR) [81]. This is one of the first well-known data-driven CT MAR methods and is now widely used as a baseline method for the recent CT MAR research [74, 82]. In this method, a CNN block is used to create prior images. These prior images are used to replace metal impacted regions in the sinogram domain. Thus, this method is again a sinogram domain inpainting method. Although prior information is built well by the data-driven method, the data-driven method is limited to the image domain. An additional data-driven method can be introduced in the sinogram domain to improve image quality further. Park et al. [83] employed the U-Net [84] to repair metal corrupted sinogram. While the single-domain deep learning method can reduce some metal artifacts, it is limited to reducing streak artifacts coming out of high-density metal objects and the relevant secondary artifacts. Gjesteby et al. [85] proposed a data-driven method for improving NMAR results. Although the method uses a CNN to improve an output image of the model-based method, the final performance is highly dependent upon the NMAR results. Thus, the performance relies on the model-based prior information in the NMAR. Wang et al. [86] proposed using a conditional Generative Adversarial Network (cGAN) [87] to reduce metal artifacts in

CT images. This method seems effective if a metal artifact is present in a localized area. However, it does not seem to work well for a significant metal artifact caused by a heavy metal object [88]. Huang et al. [89] introduced RL-ARCNN to reduce metal artifacts in cervical CT images. The deep learning network is introduced in the image domain. Ghani et al. [90] used cGAN [87, 22] to reduce metal artifacts. Their deep learning network is placed in the sinogram domain. Both methods are single-domain deep learning methods, and the abilities for the metal artifact reduction are limited because of the nature of the CT MAR problems. Liao et al. [91] proposed an unsupervised neural network to swap normal tissues with metal artifacts to tackle the CT MAR problems. Although unsupervised deep learning is more practical for industry usages, the method may not adequately replace normal tissues with metals due to the problematic nature of the CT MAR problems. Peng et al. used an image domain neural network [92] to try to reduce metal artifacts.

While the DL has been successfully applied to CT MAR problems, many previous data-driven methods have been limited to one domain. They either use a neural network in the image domain or the sinogram domain. Because CT MAR problems are complex, it is beneficial to tackle the problems in both domains. Recently, Peng et al. proposed a dual-domain method [88] for CT MAR. The method uses the U-Net [84] in the image domain and a specific neural network in the sinogram domain. Because CT MAR problems are primarily sinogram domain problems, having one neural network in the sinogram domain makes sense. However, the neural network in the sinogram domain is not sufficient to fix all types of metal artifacts. Usually, a secondary artifact is introduced by a sinogram domain correction. Thus, adding the additional network in the image domain is helpful. The method crops metal-impacted regions in the sinogram domain which seems wasteful as we discussed in the earlier section of this chapter. In addition, the method uses the pre-trained VGG-16 network [67] in the sinogram domain. The nature of sinograms is different from natural images because they are governed by X-ray physics and each

CT system's scanning geometry. Simply using the neural network trained by natural images seems in need of more extensive explanations. Although using such a network in the image domain in CT imaging can be justified, CT raw data in the sinogram domain have different characteristics from natural images [2]. Lin et al. introduced DuDoNet [82], which uses dual-domain CNNs to reduce metal artifacts. This method has two U-Nets [84]. One is in the sinogram domain, and the other is in the image domain. In order to utilize multiple neural networks, they introduced Radon Inversion Layer (RIL), which performs fan-para rebinning, filtering, and backprojection. This RIL seems a limiting factor of the method because the interpolations in the fan-para rebinning introduce additional numerical inaccuracies, and the neural networks would suffer from the interpolation inaccuracies. Most recently, Yu et al. improved DuDoNet by using differentiable Fan-beam Filtered Backprojection (FBP) and the corresponding forward projection [74]. Therefore, this method does not need a Radon Inversion Layer. Also, this method has two U-Nets in both domains as well. This method is another deep-learning-based inpainting method by a metal trace (binary mask). A metal trace without sharp boundaries should be conducive to alleviate a secondary artifact. In addition, the loss function of these deep learning networks is a simple addition of a prior loss, a sinogram loss, and a final image loss with regularization parameter adjustments. Each loss has a different unit scale, and some loss function is often more reliable than others. Therefore, simply adding multiple losses from different domains is always challenging. One universal loss function for all neural networks should be helpful to improve image quality further.

In this chapter, we propose a new deep learning method for CT MAR. The method is based upon an IR formulation, and it adopts one of the convex optimization methods [93] called Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [94]. The FISTA is used to design a Recurrent Neural Network (RNN) architecture. We name this method "RNN-MAR." The main contributions of this work are:

1. We propose the Recurrent FISTA Unit (RFU) as an RNN cell for CT MAR. The

RFU is influenced by the Gated Recurrent Unit (GRU) [45], but the RFU is specifically designed for CT imaging.

2. We show that the RNN-MAR conducts dual-domain learning. Because CT MAR problems need to be tackled in both the sinogram and the image domain, this dual-domain learning method is effective. Unlike the previous dual-domain learning methods, this proposed method uses only one objective loss function.

3. The RNN-MAR does not use a metal trace. Instead, we propose a sinogram confidence map called a C matrix for CT MAR. The C matrix is extended from Bouman and Sauer's D matrix [73], but it is tuned up for CT MAR applications.

We conducted extensive qualitative and quantitative analyses with synthetic images and real CT images. The method is compared against previous CT MAR methods to show the effectiveness.

In the following section, we show how we came up with the RNN-MAR neural network architectural design. The organization of this chapter is the following. We discuss our methods in detail in the next section. In the third section, we show qualitative and quantitative evaluation results. In the fourth section, we discuss the summary and the conclusion of this chapter.

## 3.2 Methods

The proposed RNN is specifically designed for CT imaging, particularly for CT MAR problems. The intent is to outperform popular DL methods from Computer Vision such as the U-Net on CT MAR problems. The RNN network structure design is made based upon an IR formulation with the FISTA [94] optimization. This should not be confused with the DL network optimization such as RMSprop and ADAM optimizer. This RNN

still uses ADAM to optimize the network weights in training. However, the IR formulation and the FISTA optimization are used to design the network structure of the RNN. Figure 3.1 shows a diagram of the RNN.

We begin our explanation of the RNN network design by reviewing the IR formulation. Suppose $x$ is a primal variable to optimize in the image domain, and $y$ is observation data in the sinogram domain. The conventional IR maximizes the following posterior probability distribution to obtain an optimized image $x^*$ [9].

$$\begin{aligned} x^* &= \arg\max_x \big\{p(x|y)\big\} \\ &= \arg\min_x \big\{ -\log p(y|x) - \log p(x)\big\} \\ &\cong \arg\min_x \big\{J(x) = F(Ax) + G(x)\big\}, \\ \text{s.t. } F(x) &= \frac{1}{2}(y-Ax)^T D(y-Ax), \end{aligned} \tag{3.1}$$

where $J(x)$ is the objective function to minimize. $F(\cdot)$ is called a data fidelity term to ensure an optimized image result is consistent with the corresponding observation data. $G(\cdot)$ is called a prior term, and this works as a regularization term. A is a system matrix representing a CT scanning system that performs a forward projection [9]. $D$ is a symmetric matrix that is proportional to the variance of observation data [9]. This $D$ matrix indicates the credibility of observation data [9]. For $G(\cdot)$, many forms of a prior term have been proposed [9]. It can be as simple as Total Variation prior, but many DL-based prior terms have been also proposed [31].

Here, we assume $F(\cdot)$ is convex and $\beta$-smooth because of the quadratic form. However, it is reasonable to assume the prior term $G(\cdot)$ is convex but not necessarily smooth [31]. A good example is a TV prior with L1-norm. If the objective function $J(x)$ was smooth and convex, Gradient Descent (GD) optimization would be the way to optimize the objective function in the following manner [9].

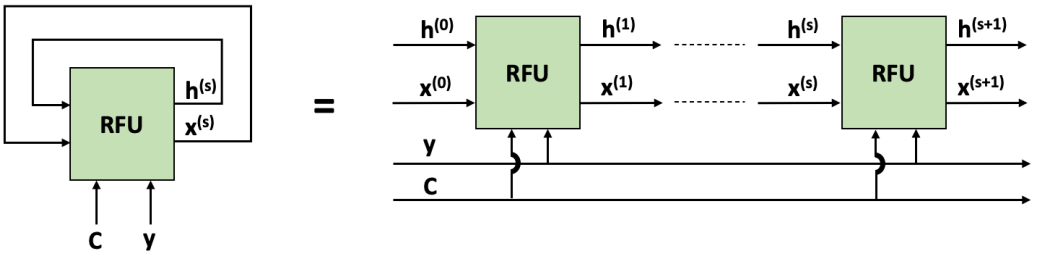



FIGURE 3.1: The overall network structure of the RNN with Recurrent FISTA Unit (RFU) for CT MAR. The RFU is shown in detail in Figure 3.2. The same network weights in the RNN cell are repeatedly used in the RNN. In the figure, $s$ is the number of stages of the RNN, $h^{(s)}$ is a hidden state of the network at stage $s$, $x^{(s)}$ is an optimized image at stage $s$, $y$ is an input sinogram, $C$ is a sinogram confidence map as known as $C$ matrix. $h^{(s)}$ and $x^{(s)}$ are changing as the number of stages is in progress. In contrast, $y$ and $C$ are constant.

$$x^{(s+1)} = x^{(s)} - \lambda \nabla_x J(x), \tag{3.2}$$

where $s$ is the number of stages (iterations). To avoid any confusion with a DL training iteration (epoch) later in this chapter, we do not use the term "iteration" in this equation, although this is the iteration for the "iterative" reconstruction. $x^{(s+1)}$ is a step further optimized result of $x^{(s)}$, and $\lambda$ is a step size of the GD. Equation (3.2) would be executed repeatedly to reach an optimal point of $x$.

As is described above, the objective function $J(x)$ is not usually smooth in the IR formulation. One way to conduct such non-smooth convex optimization is to use a proximal operator [31, 95].

$$prox_{\eta,G}(x) = \arg\min_u \{\frac{1}{2\eta}\|u - x\|_2^2 + G(u)\}, \tag{3.3}$$

where the first term is a regularization term to encourage $x$ and $u$ to be close to each other. $\eta$ is a regularization parameter to control the term.

### 3.2.1 Iterative Shrinkage-Thresholding Algorithm (ISTA)

Because the objective function $J(x)$ of the IR is a composite function of a smooth function and a non-smooth function, one way to optimize such an objective function is to use one of the proximal gradient descent methods called the ISTA [94]. The ISTA conducts a GD on one function, but it uses a proximal operator for the other function.

$$\begin{aligned} w^{(s+1)} &= x^{(s)} - \frac{1}{\beta}\nabla_x F(x^{(s)}), \\ &= x^{(s)} - \frac{2}{\beta}A^{+}D\big(Ax^{(s)} - y\big), \\ x^{(s+1)} &= prox_{\frac{1}{\beta},G}\big(w^{(s+1)}\big), \end{aligned} \tag{3.4}$$

where the first term is a GD optimization for $F(Ax)$. The second term is a proximal mapping of $G(x)$. $A^{+}$ is the adjoint of $A$. Also, $1/\beta$ is a step size of the GD, and this is determined by the smoothness $\beta$ of $F(\cdot)$.

### 3.2.2 Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

The convergence rate of the ISTA is known as $O(1/s)$ [94] and there is a faster version which is called the FISTA [94]. The convergence rate of the FISTA is known as $O(1/s^2)$ [94]. Here is the FISTA optimization formulation [94].

$$\begin{aligned} h^{(s+1)} &= prox_{\frac{1}{\beta},G}\Big(x^{(s)} - \frac{2}{\beta}A^{+}D\big(Ax^{(s)} - y\big)\Big), \\ x^{(s+1)} &= (1 - r^{(s)})h^{(s+1)} + r^{(s)}h^{(s)}, \end{aligned} \tag{3.5}$$

, where $r^{(s)} = \frac{1-\mu^{(s)}}{\mu^{(s+1)}}, \mu^{(s)} = \frac{1+\sqrt{1+4(\mu^{(s)})^2}}{2}, \mu^{(0)} = 0$. Here, the first term is equivalent to those two terms in equation (3.4). The second term is called momentum term, and this is to accelerate the convergence. The range of $r^{(s)}$ is $r^{(s)} \in [-1, 0]$ in this model-based method.

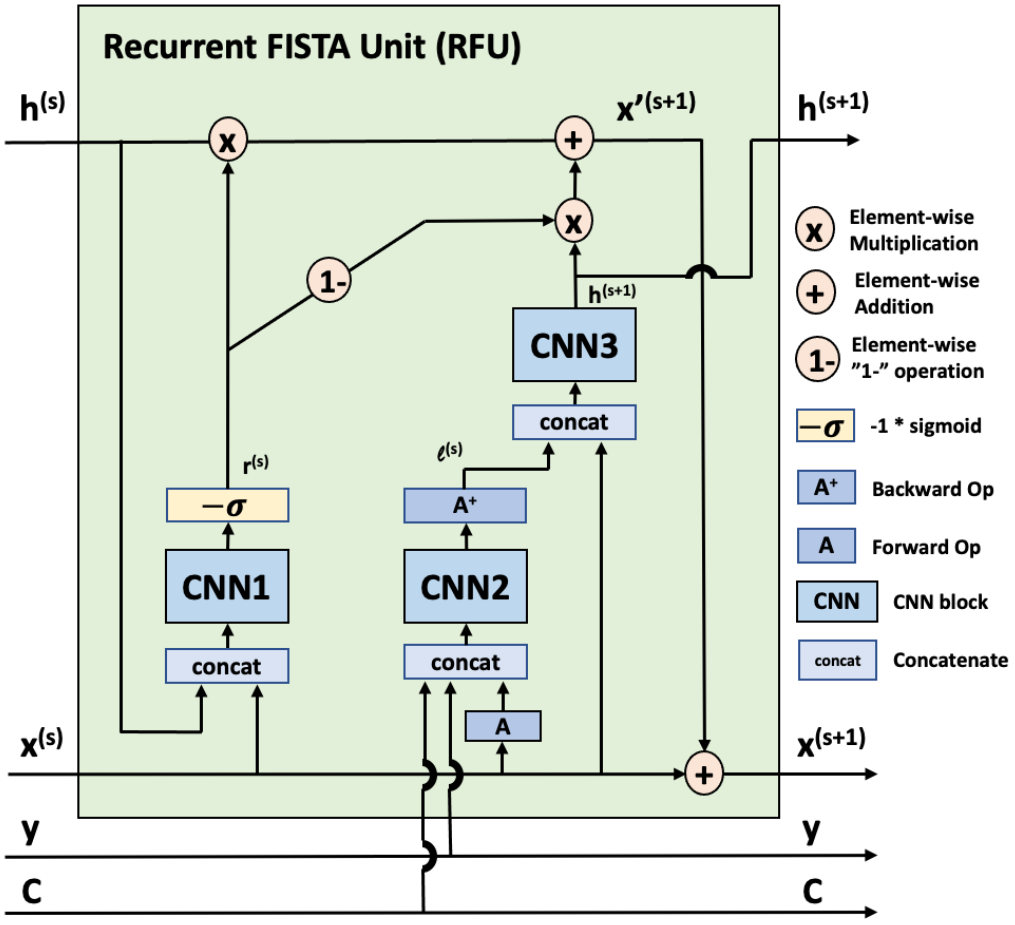



FIGURE 3.2: The RNN cell structure of the Recurrent FISTA Unit (RFU). The design of $r^{(s)}$ calculation is influenced by the forget gate in the Gated Recurrent Unit (GRU) [45], as it regulates how much a hidden state should be used at each stage. Multiple fully connected layers are used in the GRU, but the RFU uses multiple CNN blocks with concatenations instead. In the RFU, $r^{(s)}$ is learned, but the value range is preserved from the model-based method such that $r^{(s)} \in [-1, 0]$. The RFU conducts residual learning. $A$ is the forward operator, and $A^+$ is its adjoint. $CNN1$ and $CNN3$ are placed in the image domain, but $CNN2$ is allocated in the sinogram domain. Thus, this conducts dual-domain learning.

### 3.2.3 Recurrent FISTA Unit (RFU)

We could use the model-based method described in equation (3.5) to obtain an optimized image $x^*$. However, we chose to use the DL to implement equation (3.5) in this research to take advantage of its powerful representation capability [79]. In particular, we implement the FISTA by the RNN. The RNN comes into the picture because of the "recurrent"

nature of the neural network. Equation (3.5) is an iterative optimization of the IR form, and the "iterative" can be implemented as a "recurrent" neural network.

To begin with, some past research projects show that a proximal operator can be implemented by a CNN block [31, 96]. In fact, this data-driven proximal operator is known to be more effective than the conventional model-based proximal operator [31]. We transform equation (3.5) into the following form by using multiple CNN blocks.

$$\begin{aligned} h^{(s+1)} &= CNN3\big(\big[x^{(s)}, l^{(s)}\big]\big), \\ x'^{(s+1)} &= (1 - r^{(s)})h^{(s+1)} + r^{(s)}h^{(s)}, \\ x^{(s+1)} &= x^{(s)} + x'^{(s+1)}, \end{aligned} \tag{3.6}$$

where $l^{(s)} = A^{+}\big(CNN2\big([C, Ax^{(s)}, y]\big)\big)$. $h^{(s+1)}$ becomes a hidden state of the RNN. The proximal operator in equation (3.5) is replaced by $CNN3$. The +/- operators in the model-based method in equation (3.5) are replaced with concatenations in the DL equation (3.6). We could use +/- operators in the network, but a concatenation is more effective. This is, for example, often described as the difference between $f(x_1 + x_2)$ and $f(x_1, x_2)$ [79] where $f$ is a function and $x_1, x_2$ are some scalar variables. The former is a one-dimensional function, and the latter is a two-dimensional function. Using a concatenation is similar to the latter. Thus, using a concatenation gives the DL framework more freedom and is more powerful, although concatenations consume more GPU memory. In equation (3.6), the $D$ matrix is replaced by a sinogram confidence map called a $C$ matrix. This will be explained later in this section in detail. $CNN2$ is placed between $A^{+}$ and $A$ operators. This is because the forward operator and its adjoint do not completely match. Therefore, this CNN block helps compensate for the mismatch. $CNN2$ is also essential for CT MAR. This CNN block is valuable for repairing sinogram data if a heavy metal object corrupts the data. We discuss more about this later in this section. In addition, $r^{(s)}$ is now learned and $r^{(s)} = -\sigma\big(CNN1\big(\big[h^{(s)}, x^{(s)}\big]\big)\big)$. The model-based approach to calculate $r^{(s)}$ would not work for this RNN because the number of stages is limited by

the amount of GPU memory in DL training. Thus, it needs to be learned to adjust the number of stages the RNN can accommodate. However, we chose to preserve the range of $r^{(s)} \in [-1, 0]$. This is why $CNN1$ has $-\sigma$ ($-$sigmoid) operation. $CNN1$ takes $h^{(s)}, x^{(s)}$ as its input. This is influenced by the forget gate in the Gated Recurrent Unit (GRU) [45] as $r^{(s)}$ regulates how much the hidden state needs to be "forgotten." Furthermore, the second line of equation (3.6) is the momentum term that matches the model-based counterpart. The third line is a skip connection, and it conducts residual learning [79]. This does not exist in the model-based method, but we chose to add it to take advantage of one of the best practices in the DL [79]. Figure 3.2 shows a diagram of the RFU, and the figure matches equation (3.6).

As described in the earlier section of this chapter, our RNN does not use a metal trace (a binary mask). Instead, it uses a $C$ matrix (a floating-point mask). A sinogram confidence map $C$ matrix is related to Bouman and Sauer's $D$ matrix [73, 9] in the following way.

$$
\begin{aligned}
d_{ij} \propto I_{ij} &= I_0 \exp^{-p_{ij}} \cong \frac{1}{\sigma^2_{p_{ij}}}, \\
c_{ij} &= \begin{cases} 1 & \text{if } d_{ij} > \alpha_2, \\ 0 & \text{if } d_{ij} < \alpha_1, \\ \frac{d_{ij}-\alpha_1}{\alpha_2-\alpha_1} & \text{otherwise,} \end{cases}
\end{aligned}
\tag{3.7}
$$

where $c_{ij}$ and $d_{ij}$ are the elements of $C$ and $D$ matrices at $i, j$ location, respectively. $I_{ij}$ is measured X-ray photon counts at CT detector $i, j$ location. $I_0$ is an incident X-ray. $p_{ij}$ is the projection at location $i, j$. $\sigma^2_{p_{ij}}$ is the variance of projection $p_{ij}$. $\alpha_1$ and $\alpha_2$ are hyper-parameters. Bouman and Sauer's $D$ matrix shows the credibility of observation data [73, 9]. However, this $D$ matrix cannot be directly used for feeding into the neural network for CT MAR because the element values of a $D$ matrix for air regions and metal-impacted regions are not different enough. Also, the element values are not normalized to $[0, 1]$. These issues could lead to a DL optimization performance degradation. $D$ matrix

was designed for model-based methods, and it needs to be adjusted for DL applications. We would like to set 0 (no confidence) in heavily metal impacted regions in the sinogram domain and set 1 (the highest confidence) in air regions. We set the hyper-parameters empirically so that $c_{ij} = 0$ if two or more metal objects block an incident X-ray. In air regions, $c_{ij} = 1$. Other regions are set to $c_{ij} \in [0, 1]$ depending on how much an incident X-ray is attenuated. This $C$ matrix has smoother edge pixels because the element values are floating-point values instead of binary numbers. This is helpful in reducing secondary artifacts.

As Figure 3.2 shows, the proposed method uses three CNN blocks. $CNN1$ and $CNN3$ are placed in the image domain. $CNN2$ is placed in the sinogram domain. Thus, this accomplishes dual-domain learning. $CNN2$ is related to the data fidelity term in equation (3.1), and it helps "repair" sinogram in the CT MAR problems. $CNN1$ is intended to help the convergence acceleration. $CNN3$ would help fix a secondary artifact that $CNN2$ might introduce.

In addition, this network utilizes only one objective function because each domain network is executed alternatively and iteratively. Previous work [74, 82] used multiple losses from two different domains. In their research, the addition was adjusted by regularization parameters such as $\mathcal{L}_{total} = \lambda_1 * \mathcal{L}_{sinogram} + \lambda_2 * \mathcal{L}_{image}$, where $\mathcal{L}$ is a loss in a domain and $\lambda_1, \lambda_2$ are the regularization parameters, respectively. However, these regularization parameters directly impact the amount of the total loss. Also, a loss in the image domain has a different unit scale from a loss in the sinogram domain. Furthermore, a loss in one domain may be more reliable than another loss in the other domain. This regularization parameter adjustments are more complicated than they seem. Using a universal loss for both image and sinogram domain should work better. Here is how we tackle this problem. Because the RNN utilizes a differentiable FP and its adjoint inside, the network does not need to calculate losses for each CNN block. Instead, the RNN computes the final loss against the corresponding ground truth image $x_{gt}$ at the last stage $s = S$ such that

$\mathcal{L}_{total} = \mathcal{L}_{image} = MSE(x^{(S)}, x_{gt})$. The calculated loss is back-propagated to all networks in both domains by a DL software (TensorFlow or PyTorch) because the forward and the backward operators are differentiable and visible to the DL software. If a loss is originated from the image domain, such as an image texture problem, it gets fixed in the image domain. If a loss comes from the sinogram domain such as a CT detector cell malfunction, for example, it gets adjusted in the sinogram domain. The backpropagation process takes care of distributing an error into each domain appropriately. This is the power of differentiable operators. This idea comes from Learned Primal-dual Reconstruction by Adler et al. [31, 32], although this advantage is not clearly articulated in the paper.

Table 3.1 shows the network configurations of each CNN block in the RFU. We use PReLU as our activation function in the middle layers in this research, as it was done by Adler [31]. ReLU has been one of the key factors to the recent successes of deep learning [79], and PReLU is a member of the ReLU family. It is parametrized in both the negative and the positive input values. The parameter of PReLU is trained as part of the network optimization process. In addition, $CNN1$ uses $-\sigma$ (a negative sigmoid) activation function for its last layer.

There is one more comment about the CNN configurations in Table 3.1 and the FISTA optimization. We assume $F(\cdot)$ is convex and smooth, and $G(\cdot)$ is convex but not smooth in equation (3.1). This assumption is essential for this data-driven method. If we assumed both terms were not smooth, it would lead to using two proximal operators in equation (3.4). Therefore, the optimization algorithm would not be the ISTA or the FISTA. Instead, it would be the Alternating Direction Method of Multipliers (ADMM) [93] or the Primal-dual optimization [36]. Our assumption of $F(\cdot)$ being smooth helps simplify $F(\cdot)$ optimization and enables to move more GPU resources from $F(\cdot)$ to $G(\cdot)$ optimization, which uses a proximal operator. This is why $CNN3$ has twice as many channels as other CNN blocks. Our further experiments show that this simplification improves the final image quality of the RNN-MAR. This is why we chose the FISTA optimization in this

TABLE 3.1: The configurations of three CNN blocks in the Recurrent FISTA Unit (RFU). From left to right, the table shows the name of CNN block (Name), the number of input channels (Inputs), the number of CNN layers in each block (Layers), the number of CNN channels (Chans), the type of activation function (Activation), the number of output channels (Outputs). CNN1 uses PReLU as its activation function in the middle layers and the negative sigmoid $-\sigma$ in the final layer. CNN2 and CNN3 use PReLU in the middle layers but do not use an activation function in the final layer.

| **Name** | **Inputs** | **Layers** | **Chans** | **Activation** | **Outputs** |
|---|---|---|---|---|---|
| CNN1 | 2 | 6 | 32 | PReLU/$-\sigma$ | 1 |
| CNN2 | 3 | 6 | 32 | PReLU/None | 1 |
| CNN3 | 2 | 6 | 64 | PReLU/None | 1 |

research project. Thus, the smoothness assumption in the model-based method (FISTA) is vital for more efficient GPU memory usage in the data-driven method (RFU).

A pseudo-code of the RFU is shown in Algorithm 2. The RFU initializes the hidden state $h^{(0)}$ with zeros and $x^{(0)}$ with a LI MAR image [75].

**Algorithm 2** Recurrent FISTA Unit (RFU)

**Require:** Input sinogram $y$, the initial estimate of images $x^{(0)} \leftarrow$ LI image [75], the forward projector $A$, its adjoint $A^+$, the initial hidden state $h^{(0)} \leftarrow \vec{0}$, the sigmoid function $\sigma$, element-wise multiplication $\otimes$, the RNN network parameters $\theta$, the number of RNN stages S, the number of mini-batch B, a concatenation $[,]$. The loss is calculated by the Mean Squared Error (MSE) in the image domain.

**for all** $s \in 1, ..., S$ **do**

$r^{(s)} \leftarrow -\sigma\Big(CNN1\big([x^{(s)}, h^{(s)}]\big)\Big)$

$l^{(s)} \leftarrow A^+\Big(CNN2([y, C, Ax^{(s)}])\Big)$

$h^{(s+1)} \leftarrow CNN3\Big([l^{(s)}, x^{(s)}]\Big)$

$x'^{(s+1)} \leftarrow r^{(s)} \otimes h^{(s)} + (1 - r^{(s)}) \otimes h^{(s+1)}$

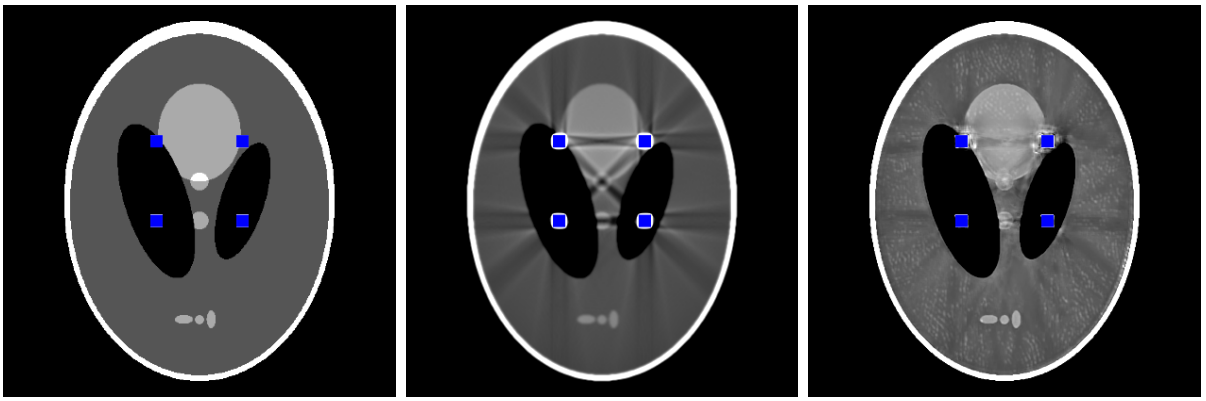
$x^{(s+1)} \leftarrow x^{(s)} + x'^{(s+1)}$

**end for**

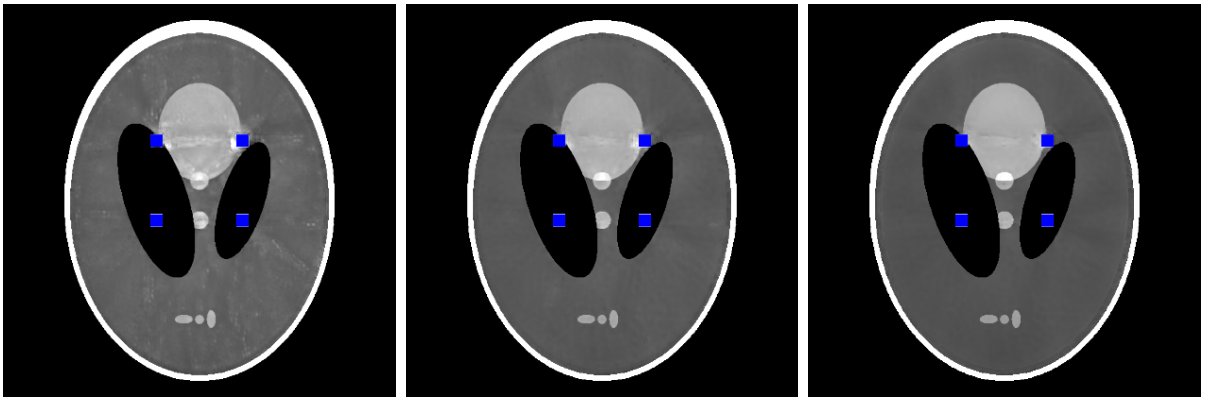
$\theta \leftarrow AdamOptimizer(\nabla_\theta \sum_{b=1}^{B} Loss^{(b)})$

## 3.3 Experimental Results

The proposed method was evaluated by comparing against FBP, LI [75], NMAR [76], CNN-MAR [81], DuDoNet [82], and Deep Sinogram Completion [74]. The evaluation was done with synthetic image data and real CT image data. We followed previous CT MAR literature to simulate metal artifacts [81, 97], where beam hardening and Poisson noise were simulated. We used titanium and iron in the metal artifact simulation. These materials were randomly picked when training and testing data were generated. We assumed that a polychromatic X-ray source was used, and the incident X-rays had $1.0\times10^7$ photons. A fan-beam geometry was adopted. We used a simple thresholding method of 2,000 HU to extract a metal mask in the image domain. $\alpha_1, \alpha_2$ for a C matrix were set to $9.42 \times 10^6$ and $1.0 \times 10^7$ photon counts, respectively. The image matrix size was 416x416, and the sinogram size was 1025 detector channels and 984 views. The number of RNN stages in RNN-MAR was four in both the training and the testing. The ADAM optimizer was used in the DL training to optimize DL weights. We used 0.001 as the learning rate with a cosine decay [31] in the proposed method. The TensorFlow Python package was used to implement RNN-MAR. Mean Square Error (MSE) was used for the RNN's loss. In quantitative analyses, we calculated Root Mean Squared Error (RMSE), Peak Signal-to-Noise Ratio (PSNR), and Structure Similarity (SSIM) to conduct quantitative analyses. RMSE calculations excluded metallic objects, following previous research papers [74, 81]. On the other hand, PSNR and SSIM calculations used all pixels, including metallic objects, again following another research paper [82]. Each training step randomly chose images from the training data set. We used 800 epochs in the training for each DL-based algorithm we tested. We used the pre-trained CNN-MAR network for the evaluation [81]. Since the implementation for DuDoNet [82] and Deep Sinogram Completion [74] are not available in public, we implemented them ourselves. The mini-batch size was 8 for these algorithms. We used mini-batch size 2 for the proposed method



(A) Ground truth (B) 0th stage (FBP) (C) 1st stage



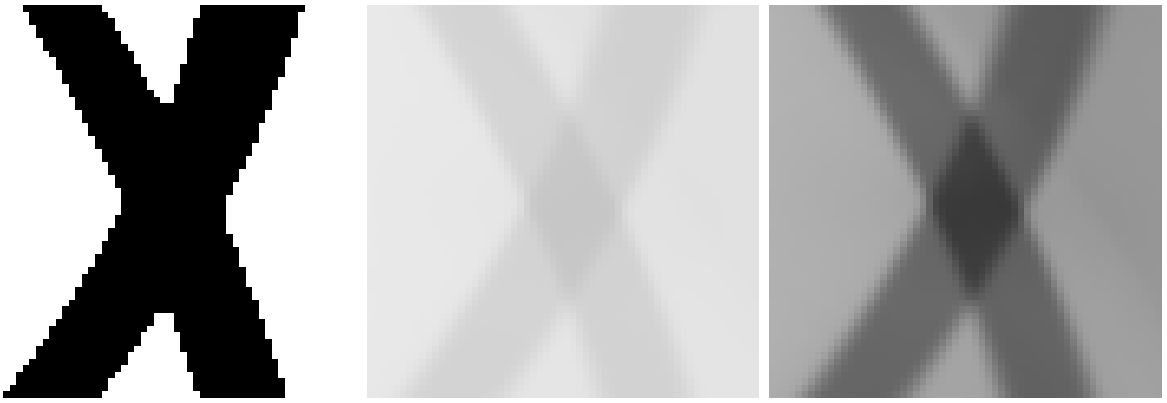
(D) 2nd stage (E) 3rd stage (F) 4th stage (Final)

FIGURE 3.3: Synthetic image study results on Shepp-Logan phantom. Blue boxes indicate metal objects. (a) The ground truth image, (b) 0th stage (FBP), (c) 1st stage, (d) 2nd stage, ....., (f) 4th stage (Final). The RNN is configured with four stages in training and testing.

because of a GPU memory limitation. We used Nvidia V100 GPU in the training and the testing. We used TensorFlow to implement DuDoNet [82], Deep Sinogram Completion [74], and RNN-MAR (ours). The ODL Python package [56] was used to perform the forward projection and its adjoint for CT image reconstruction.

### 3.3.1 Synthetic Image Study

We used 1,000 images in the training and the Shepp-Logan phantom in the testing in the synthetic image study. The Shepp-Logan phantom is shown in Figure 3.3. Each training image has a random number of randomly shaped ellipses with a random number of metals



(A) Metal trace (B) D matrix (C) C matrix

FIGURE 3.4: An example of a metal trace, a D matrix, and a C matrix. In this figure, we use the pixel value range of $[0, 1]$ to visualize (a) and (c). For (b), we use $[8.0 \times 10^6, 1.0 \times 10^7]$ for the visualization.

TABLE 3.2: Quantitative evaluation of the synthetic image study test results with the various number of stages of the RNN. As the RNN runs more stages, the results improve. The training is done with four stages.

| **Method** | **RMSE (HU)** | **PSNR (dB)** | **SSIM (%)** |
|---|---|---|---|
| 0th (FBP) | 521.58 | 23.48 | 66.62 |
| 1st stage | 343.90 | 27.54 | 81.71 |
| 2nd stage | 195.01 | 32.49 | 94.77 |
| 3rd stage | 94.96 | 38.72 | 97.79 |
| 4th stage (Final) | 50.39 | 44.20 | 98.74 |

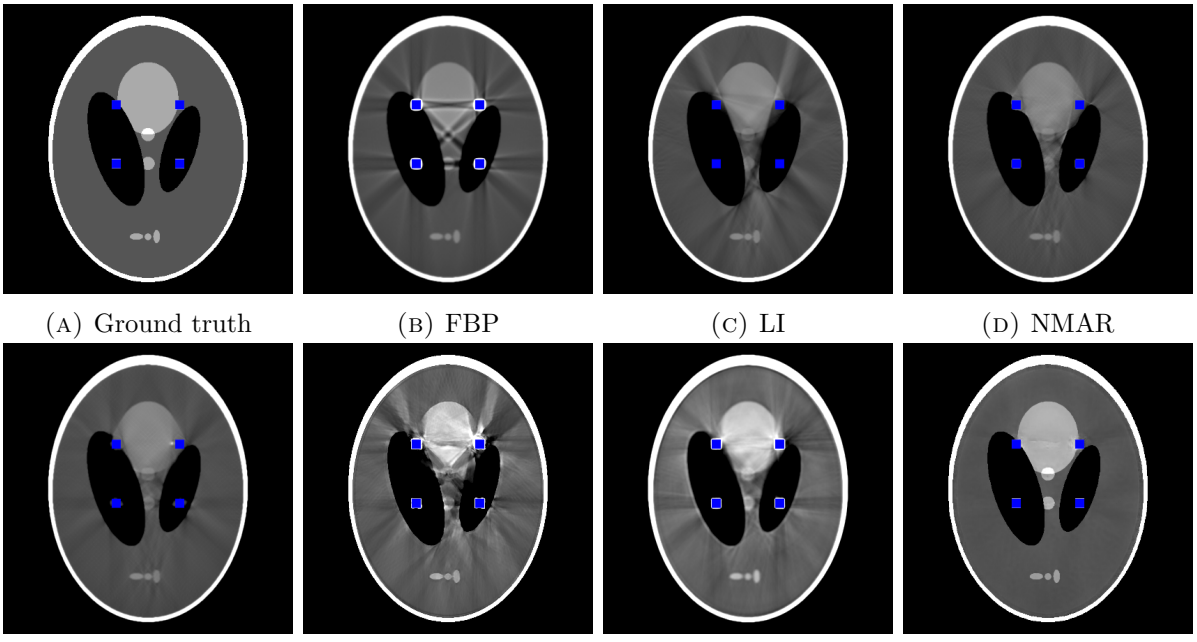
[98]. The number of metals in each training image is in the range of $[1, 4]$. The number of metals in the testing image is four.

Figure 3.3 shows test results on each RNN stage. As the number of stages increases, the metal artifacts are reduced more significantly. Table 3.2 shows quantitative analyses on each RNN stage. As the number of RNN stages increases, all metrics improve.

Figure 3.4 shows an example of a metal trace (a), a D matrix (b), and a C matrix (c) from the synthetic data shown in Figure 3.5. The metal trace is a binary mask $\in \{0, 1\}$. It has sharp edges on the boundary pixels of metal regions and non-metal regions. On the other hand, the D and C matrices have floating-point numbers. Thus, edges are not sharp. Both of these matrices show the credibility of sinogram data. However, the $C$ matrix is

TABLE 3.3: Quantitative evaluation of the synthetic image study test results. Bold numbers indicate the best results.

| Method | RMSE (HU) | PSNR (dB) | SSIM (%) |
|---|---|---|---|
| FBP | 521.58 | 23.48 | 66.62 |
| LI [75] (1987) | 477.75 | 21.66 | 75.80 |
| NMAR [76] (2010) | 485.36 | 21.99 | 74.71 |
| CNN-MAR [81] (2018) | 485.63 | 21.81 | 74.73 |
| DuDoNet [82] (2019) | 168.85 | 28.66 | 88.30 |
| Deep Sino Comp [74] (2021) | 285.63 | 28.62 | 61.08 |
| RNN-MAR (ours) | **50.39** | **44.20** | **98.74** |



(A) Ground truth (B) FBP (C) LI (D) NMAR

(E) CNN-MAR (F) DuDoNet (G) Deep Sino Complete (H) RNN-MAR (ours)

FIGURE 3.5: Synthetic image study test results. (a) the ground truth image, (b) FBP, (c) LI [75], (d) NMAR [76], (e) CNN-MAR [81], (f) DuDoNet [82], (g) Deep Sino Complete [74], (h) RNN-MAR (ours).

re-scaled so that the element values are in the range of $[0, 1]$. We adjusted $\alpha_1, \alpha_2$ for the C matrix in equation (3.7) accordingly so that air regions have the highest confidence ($element = 1$). A region where multiple metals block the corresponding incident X-ray has the lowest confidence ($element = 0$). Other regions are in the range of $[0, 1]$. Thus, boundary pixels are smooth.

Figure 3.5 shows the testing results of synthetic image study with various methods. The

TABLE 3.4: Quantitative evaluation of CT MAR results on the DeepLesion dataset with 227 test images. Bold numbers indicate the best results.

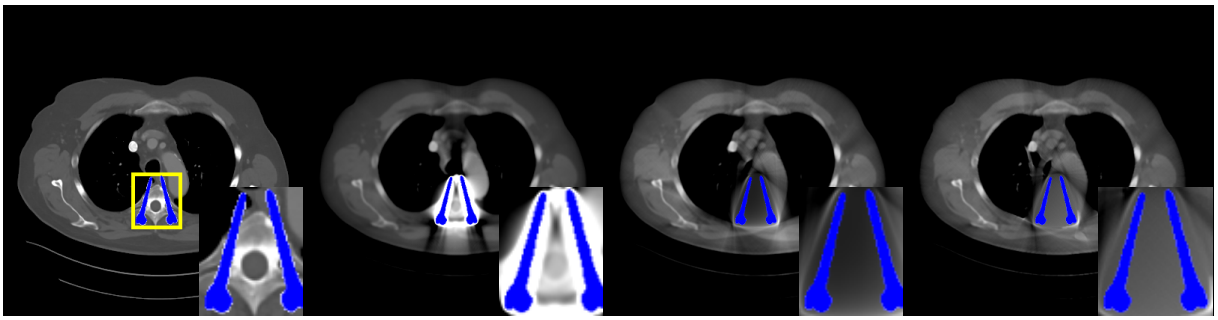
| **Method** | **RMSE (HU)** | **PSNR (dB)** | **SSIM (%)** |
|---|---|---|---|
| FBP | $137.52 \pm 31.15$ | $34.61 \pm 3.02$ | $72.18 \pm 3.68$ |
| LI [75] (1987) | $105.06 \pm 17.75$ | $28.94 \pm 5.04$ | $75.81 \pm 5.21$ |
| NMAR [76] (2010) | $104.02 \pm 18.92$ | $29.12 \pm 4.98$ | $76.53 \pm 4.58$ |
| CNN-MAR [81] (2018) | $102.66 \pm 14.00$ | $29.08 \pm 5.00$ | $77.56 \pm 3.53$ |
| DuDoNet [82] (2019) | $100.68 \pm 96.83$ | $37.05 \pm 5.81$ | $93.46 \pm 2.49$ |
| Deep Sino Complete [74] (2021) | $68.32 \pm 20.57$ | $39.13 \pm 3.16$ | $86.68 \pm 3.95$ |
| RNN-MAR (ours) | $\mathbf{21.06 \pm 9.70}$ | $\mathbf{52.80 \pm 3.85}$ | $\mathbf{98.10 \pm 1.33}$ |

model-based methods (LI and NMAR) and the CNN-MAR successfully suppress metal artifacts, but contrasts of ellipses become lower. On the other hand, the contrasts of data-driven methods (DuDoNet and Deep Sinogram Completion) seem preserved, although the level of metal artifact reduction in these methods is low because of this challenging testing case. Overall, the metal artifacts are successfully reduced, and the contrasts are well preserved in our proposed method.

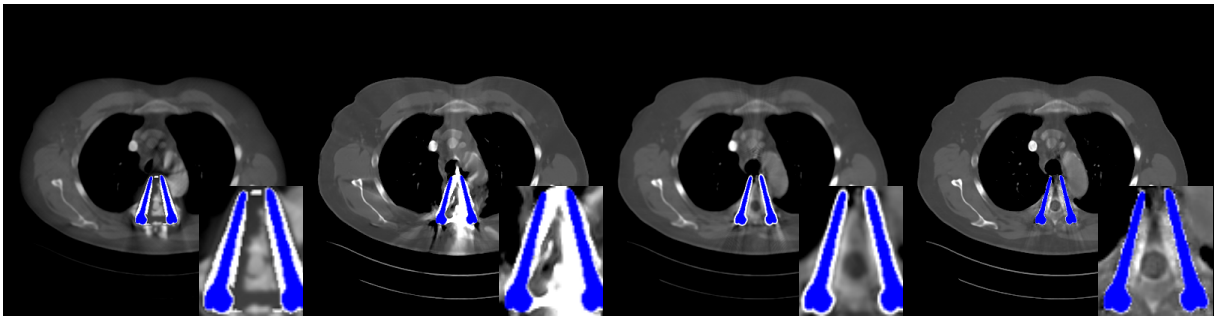
RMSE, PSNR, and SSIM results are shown in Table 3.3. The improvements of DuDoNet and Deep Sinogram Completion over other conventional methods are quite significant. The qualitative result of DuDoNet shown in Figure 3.5 does not agree with the quantitative results shown in Table 3.3. Our further analysis shows that DuDoNet recovers the air regions accurately, which drives the improvements in the quantitative results. Again, overall, RNN-MAR outperforms the conventional methods as well as Deep Sino Complete quite significantly. The proposed method outperforms the state-of-the-art method Deep Sinogram Completion by RMSE 232.08 HU, PSNR 16.11 dB, SSIM 38.79%.

### 3.3.2 Real CT Image Study

In the real CT image study, we used the DeepLesion dataset [99]. We used 1026 images in the training and 227 images in the testing. Each training step chose a mini-batch of



(A) Ground truth (B) FBP (C) LI (D) NMAR

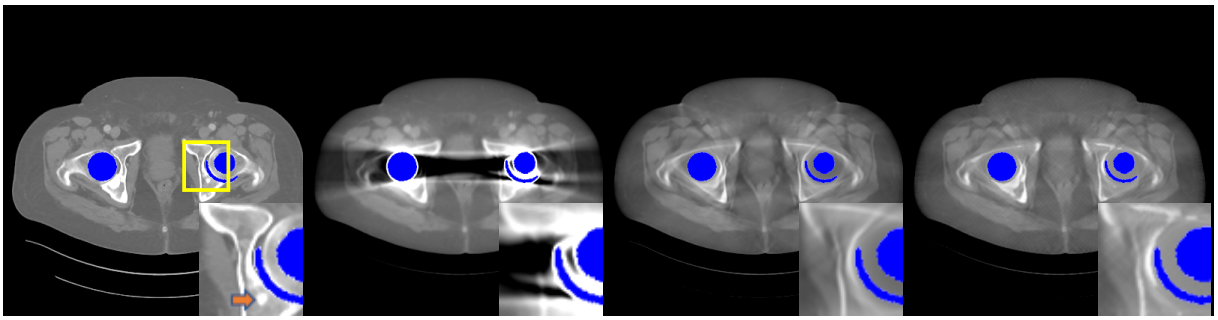


(E) CNN-MAR (F) DuDoNet (G) Deep Sino Complete (H) RNN-MAR (ours)

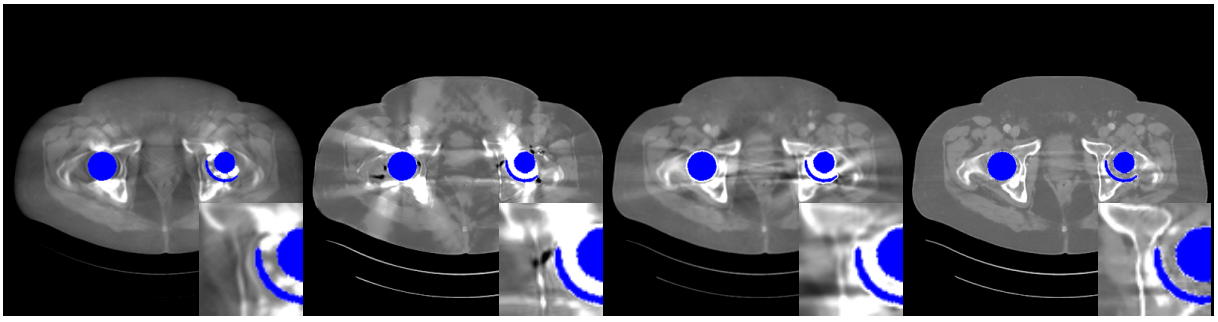
FIGURE 3.6: An example set of test results with various CT MAR methods on the DeepLesion dataset. The yellow box in (a) indicates where to zoom. Metal objects are colored blue. (a) the ground truth image, (b) FBP image, (c) LI [75], (d) NMAR [76], (e) CNN-MAR [81], (f) DuDoNet [82], (g) Deep Sinogram Completion [74], (h) RNN-MAR (ours). The window range is [-500, 1000] HU, and these metals are iron.

images randomly from the training data set. In this study, we adopted the metal mask collection from the previous research projects [81, 91]. The collection includes 100 metal implants that are manually segmented in various shapes and sizes. Specifically, 1026 real CT images and 90 metal masks were randomly paired to create the training data. The remaining ten metal masks were randomly paired with another 227 real CT images for the testing.

Table 3.4 shows quantitative evaluation results. As expected, the RMSE numbers improve from older methods to newer methods. DuDoNet has a higher standard deviation in the RMSE measurement, but Deep Sino Complete successfully lowers it although these two methods are similar. PSNR results are not usually reported in the past CT MAR literature except in DuDoNet. SSIM numbers generally go up with newer methods. Overall, RNN-MAR outperforms all of the past methods in the table, including the



(A) Ground truth (B) FBP (C) LI (D) NMAR

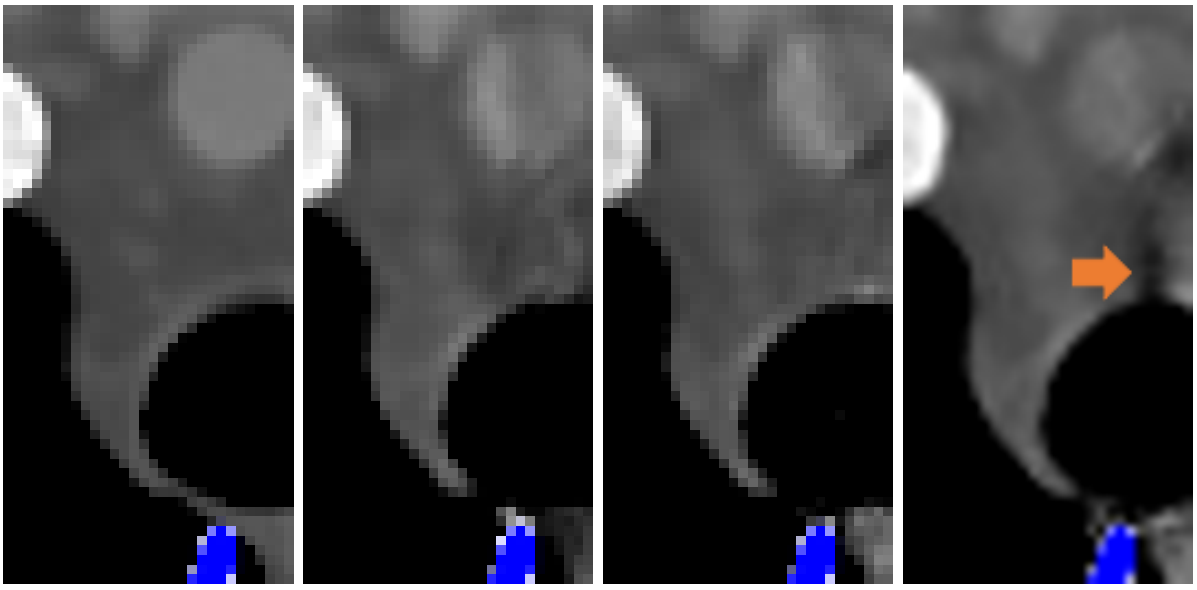


(E) CNN-MAR (F) DuDoNet (G) Deep Sino Complete (H) RNN-MAR (ours)

FIGURE 3.7: Other examples of test results with various CT MAR methods on the DeepLesion dataset. The yellow box in (a) indicates where to zoom. Metal objects are colored blue. The orange arrow in (a) indicates where to watch. (a) the ground truth image, (b) FBP image, (c) LI [75], (d) NMAR [76], (e) CNN-MAR [81], (f) DuDoNet [82], (g) Deep Sinogram Completion [74], (h) RNN-MAR (ours). The window range is [-500, 500] HU, and these metals are titanium. The white circle indicated by the orange arrow in (a) is removed from all results except RNN-MAR's (ours). Only our proposed method can keep the white circle.

state-of-the-art method Deep Sino Complete. The improvement of RNN-MAR over Deep Sino Complete is quite significant. The amounts of improvement are 47.26 HU in RMSE, 13.67 dB in PSNR, and 11.42% in SSIM on real CT images.

Figure 3.6 shows an example set of testing results. The LI and the NMAR removed metal artifacts in the results, but some parts of the images were destroyed, and anatomies can no longer be seen in these regions. CNN-MAR and DuDoNet removed metal artifacts but introduced new artifacts. Deep Sino Complete was able to remove metal artifacts nicely. The only downside we found in this method was that the anatomies between those two metal plates were over-smoothed. We believe this is because of the U-Net the method uses. The U-Net has down-sampling and up-sampling layers, and it helps capture the



(A) Ground truth (B) C matrix (C) D matrix (D) Metal trace

FIGURE 3.8: Example test results with a C matrix, a D matrix, and a Metal trace with RNN-MAR. Blue indicates a metal object. The arrow indicates where to watch. Quantitative results are the following [RMSE (HU), PSNR (dB), SSIM (%)]. C matrix = [32.66, 49.30, 98.68], D matrix = [32.90, 48.83, 98.68], Metal trace = [37.27, 48.19, 98.58].

overall image features while keeping the details by using skip connections. However, the down/up sampling layers sometimes cause resultant images to be over-smoothed. On the other hand, RNN-MAR is not only capable of removing metal artifacts but also can keep output images sharp. The best example is the anatomies between these two metal plates. RNN-MAR keeps anatomies sharp compared to the Deep Sinogram Completion.

Another set of qualitative results is shown in Figure 3.7. This example test set is quite challenging because two metal bars block incident X-ray substantially in the horizontal direction. The model-based methods can remove metal artifacts, but the resultant images become over-smoothed. CNN-MAR performs well in this particular testing image. DuDoNet and Deep Sinogram Completion remove metal artifacts but introduce new artifacts. Overall, RNN-MAR produces the best image.

Figure 3.8 shows example results on RNN-MAR with different confidence matrices. This is an example of a real CT image. In the figure, (a) is the ground truth. (b) is with

a C matrix. (c) is with a D matrix, and (d) is with a metal trace. We normalized elements of the $D$ matrix for (c) by each respective incident X-ray intensity $I_0$ in this experiment to make the DL optimization process easier to conduct. Below the figure, quantitative results are available. Overall, RNN-MAR with the C matrix produces the best quantitative results. Visually, the C and the D matrices produce the same level of image quality. The improvements from the D matrix to the C matrix result are 0.24 HU in RMSE, 0.47 dB in PSNR, and 0.0% in SSIM. As for the result (d) in the figure, the metal trace creates a shadow indicated by the orange arrow. The shadow originates from a metal object. This is because of the sharp boundaries in the metal trace. We also saw that PSNR validation results were going up and down significantly during the training phase with a metal trace. The improvements of C matrix result over metal trace result are 4.37 HU in the RMSE, 1.11 dB in the PSNR, 0.1% in the SSIM.

## 3.4 Summary

We have developed a new RNN architecture for CT MAR. The RNN-MAR is specifically designed for CT imaging and CT MAR. The network architecture is based upon the IR formulation, which maximizes the posterior distribution of image data given observation data. We assumed the likelihood term is smooth and convex, but the prior term is convex and non-smooth. This assumption led to the FISTA optimization. In this research, the FISTA was implemented as a novel RNN cell called the Recurrent FISTA Unit (RFU). The RFU conducts an iterative optimization in each stage of the RNN. The smoothness assumptions helped us allocate GPU resources efficiently. We allocated fewer DL layers in the likelihood term and more DL layers in the prior term. This improved the image quality of the results. In addition, we introduced the novel sinogram confidence map called a $C$ matrix. It was adopted from Bouman and Sauer's $D$ matrix, but it was rescaled to replace a metal trace with a C matrix. This floating-point matrix can suppress secondary

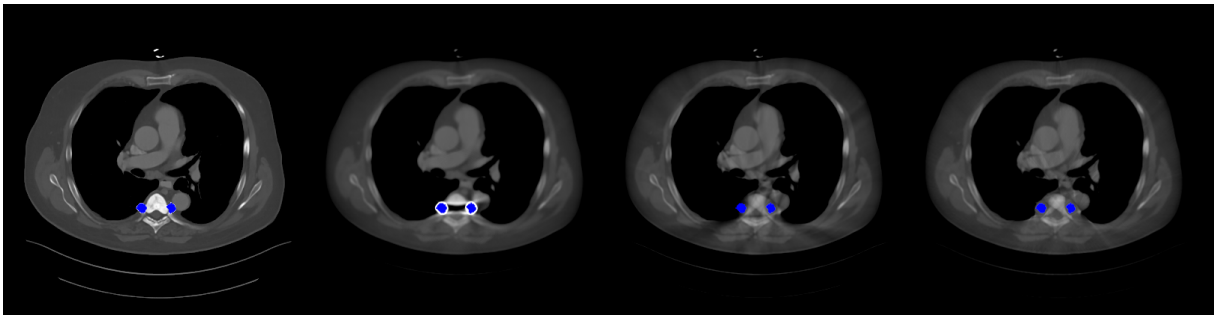
artifacts, a CT MAR method may introduce. A C matrix works better than a metal trace because of floating-point element values instead of binary numbers. Furthermore, the design of the momentum term in the RFU was influenced by the forget gate of the GRU [45] design that is one of the most successful RNN cells in the DL research community. This helped us design how much the RFU "forgets" the hidden state. Also, the RFU conducts residual learning, which is one of the best practices in the computer vision community with DL. Moreover, the RFU conducts dual-domain learning that is critical for CT MAR as the problems need to be tackled in both domains. We show that the RFU only uses one loss function for more efficient optimization in the RNN training. This was enabled by the differentiable FP and its adjoint. If these operators are differentiable and visible to a DL software such as TensorFlow or PyTorch, these software packages can distribute losses to both domains. Thus, there is no need to calculate losses in two different domains as using losses from two different domains often leads to an optimization difficulty.

The proposed method was evaluated with the synthetic image data and real CT image data. In the synthetic image study, the evaluation results showed that RNN-MAR outperforms the state-of-the-art method Deep Sinogram Completion [74] by RMSE 232.08 HU, PSNR 16.11 dB, and SSIM 38.79%. The qualitative results also agree with the quantitative results. In the real CT image study, RNN-MAR outperforms the Deep Sinogram Completion [74] by RMSE 47.26 HU, PSNR 13.67 dB, and SSIM 11.42%.

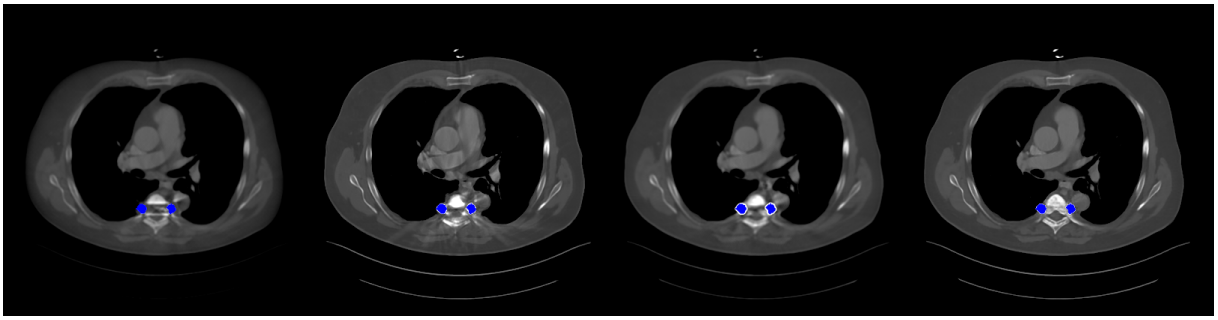
In conclusion, it is shown that RNN-MAR outperforms the state-of-the-art data-driven methods [74, 82]. Even though this research only evaluates the RNN-MAR for CT MAR problems, the proposed method has potential, and it can be applied to de-noising, semantic segmentation, or other DL problems in CT imaging. The RNN-MAR design has the potential to outperform any U-Net-based DL methods in CT imaging.

# Appendix

Further qualitative analysis results.

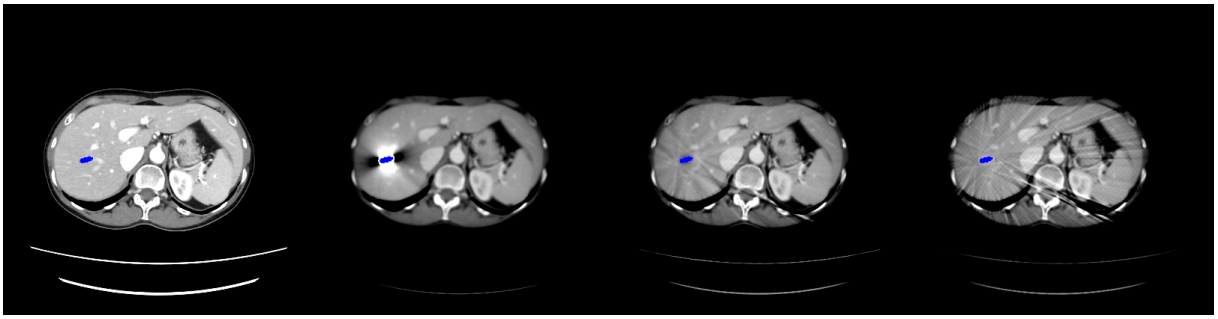


(A) Ground truth (B) FBP (C) LI (D) NMAR

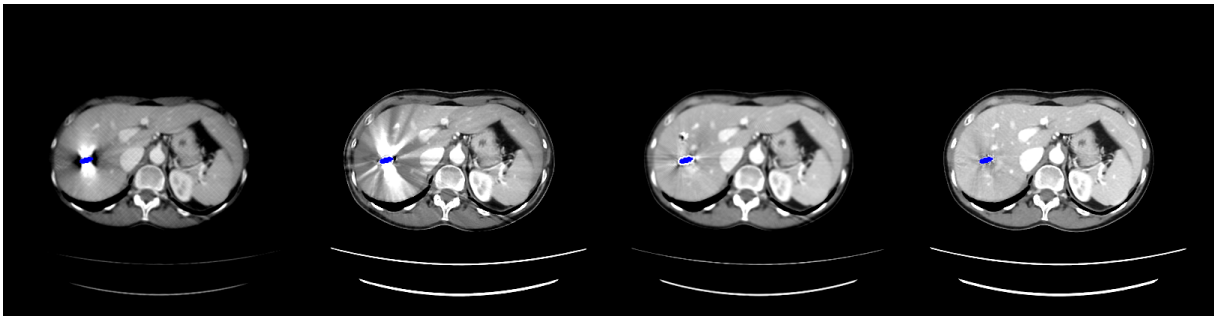


(E) CNN-MAR (F) DuDoNet (G) Deep Sino Complete (H) RNN-MAR (ours)

FIGURE 3.9: An example set of testing results with various CT MAR methods on the DeepLesion dataset. The yellow box in (a) indicates where to zoom. Metal objects are colored blue. (a) the ground truth image, (b) FBP image, (c) LI [75], (d) NMAR [76], (e) CNN-MAR [81], (f) DuDoNet [82], (g) Deep Sinogram Completion [74], (h) RNN-MAR (ours).



(A) Ground truth (B) FBP (C) LI (D) NMAR



(E) CNN-MAR (F) DuDoNet (G) Deep Sino Complete (H) RNN-MAR (ours)

FIGURE 3.10: An example set of testing results with various CT MAR methods on the DeepLesion dataset. The yellow box in (a) indicates where to zoom. Metal objects are colored blue. (a) the ground truth image, (b) FBP image, (c) LI [75], (d) NMAR [76], (e) CNN-MAR [81], (f) DuDoNet [82], (g) Deep Sinogram Completion [74], (h) RNN-MAR (ours).

# Chapter 4

# TextureWGAN: Texture Preserving WGAN with MLE Regularizer for Inverse Problems

## 4.1 Introduction and Related Work

Image de-noising, super-resolution and image reconstruction are all categorized as inverse problems [32] [31]. In image de-noising problems, the transformation from a noisy image to a de-noised image is the identical operator $I$ plus de-noising operator. These two sets of images reside in the same domain. On the other hand, the transformation from low-resolution images to high-resolution images in super-resolution is an up-sampling operator and a down-sampling operator in the other direction. These two sets of images are typically considered in two different domains, such as low-resolution domain and high-resolution domain [33]. The most popular up-sampling and down-sampling operator is bicubic interpolation because of the simplicity and the effectiveness. Regarding image reconstruction such as Computed Tomography image reconstruction, the transformation from raw data to images is called backprojection [2]. The transformation for the other direction is called forward projection [8]. Both of them have two-dimensional and three-dimensional versions depending on accounting for z-direction in the image detector of

the CT acquisition system [9]. Raw data in CT is converted into a sinogram after the negative log step [2]. Sinogram is in the "sinogram" domain. The reconstructed image is in the "image" domain. These inverse problems have been an active research area for a long time [27] [28].

The concept of Generative Adversarial Networks (GANs) was introduced by Goodfellow et al [22] in 2014. GANs have been mostly used to learn data distribution of training data set and produce new data from the learned distribution. It is often used to create human face images whose people have never existed [22]. Ledig et al [35] used GANs in super-resolution and showed GANs were helpful in inverse problems. While Variational Auto Encoder (VAE) used KL divergence [59], GANs, which are a newer version of generative models, used Jensen–Shannon divergence (JS divergence) [22]. This is because KL divergence has a weakness where the divergence becomes zero or positive infinity if two distributions are far from each other [59]. JS divergence, on the other hand, produces a finite number as divergence even if two density distributions of images are not overlapped each other [22]. However, JS divergence also has a weakness [60]. If two images are not close enough, the derivative of JS divergence becomes too small which causes a vanishing gradient problem [60]. Therefore, it prevents for deep learning optimization algorithm from converging. Arjovsky et al [60] proposed Wasserstein GAN (WGAN) to battle the vanishing gradient problem in GANs. Gulrajani et al [61] introduced WGAN Gradient Penalty (WGAN-GP) to make the loss function lie within a compact space. Under mild assumptions, the loss function of WGAN-GP is continuous everywhere and differentiable almost everywhere [61].

The training process of GANs has notable differences from the training for the usual deep learning framework. The usual deep learning is a minimization problem where a neural network produces a sample being as close as possible to the training data set. On the other hand, GANs are a min-max problem where the generator tries to minimize the distance between real data and generated data while discriminator tries to maximize the

same distance [22]. The training is completed when both networks cannot minimize or maximize the distance further. This terminal point is called the Nash equilibrium [22]. Besides, the learning process of GANs is known to be much more difficult than the one for the usual deep learning with one neural network [23]. The primary reason is that the optimization problem for the generator keeps changing as discriminator is evolving. This is not the case for the usual deep learning where the optimization problem is fixed and it does not change when the neural network is being trained. Radford et al [23] made very specific suggestions for GANs to make the training stable. They suggested replacing any pooling layers with strided convolutions in discriminator and fractional-strided convolutions in a generator. They also suggested using ReLU activation in a generator but suggested LeakyReLU activation in a discriminator. These suggestions have been made after many experiments to stabilize the training process of GANs. Despite the differences and difficulties in training, GANs are known to be more resistant to overfitting and allowed for quality enhancement with much less data than required for conventional supervised deep learning [23].

Inverse problems have been an active research area while Deep Learning is enjoying its popularity since the inception of AlexNet in 2012 [16] [20]. The most favored loss function in Deep Learning in inverse problems is Mean Square Error (MSE) [3]. In fact, MSE is the first loss function to learn when people start learning deep learning. L2 distance and Root Mean Square Error (RMSE) are in the same family. MSE measures Euclidean distance between two images. MSE loss is known to be convex, therefore, it is guaranteed to find the local minimum assuming having a sufficient number of iterations in optimization algorithms [3]. It is also known that MSE preserves pixel fidelity in inverse problems [17] [18] which is the most important requirement when it comes to medical imaging. While MSE loss provides many benefits, it is also known that resulting images are often too smooth. In medical imaging, clinicians express their concerns that de-noised images look unrealistic because de-noising algorithms make images too flat and their diagnostic ability

is degraded due to the different look and feel of post-processed images [62]. The main cause of this problem is that MSE loss measures the Euclidean distance for all pixels in each image and ignores the spatial information of the pixels such as image texture.

Changing the look and feel or image texture by an image processing algorithm is a major concern in medical imaging [62] [5] [6] [7]. Low contrast detectability in Computed Tomography is highly dependent upon image texture [7]. The visibility of low contrast objects is impacted by the noise texture pattern more significantly than that of high contrast objects with the same size [7]. The image texture of cancer is equally important as the tumor size in cancer diagnosis [7]. In fact, the image texture of cancer is highly relevant to patient survival rates [7]. Many clinicians use a blending factor to avoid image texture change in de-noising algorithms [62]. They use a de-noised image and the original noisy image along with a blending factor to create a new image. For example, a 20% de-noised image means a 20% de-noised image and an 80% original noisy image. This blended image becomes popular because image texture is known to be preserved. But this blending method comes with the expenses of degraded Peak Signal to Noise Ratio (PSNR) and Structure Similarity (SSIM).

There are a few research about using multiple loss functions in inverse problems. Yang et al [17] used MSE, Wasserstein and perception loss with deep neural network in low-dose CT image de-noising problem to evaluate whether PSNR and SSIM were improved. This research was conducted only with CT images. Only mean and standard deviation comparisons were done for image texture analysis. Without having depth image quality analysis, this research seems inconclusive although the paper had positive results suggesting WGAN has a potential for image quality improvement. Besides, it is not clear how much weight we need to put in each loss function to be able to maximize the image quality performance. Liu et al [18] also attempted similar research in synchrotron-based x-ray tomography to reduce noise in projection data. However, this research was conducted only with synchrotron-based x-ray CT images and it is not clear how much the

research results are relevant to generic de-noising problem. In addition, the authors only conducted PSNR, SSIM and pixel value comparison. The method is essentially same as [17] and they did not use any other de-noising algorithms to compare against, which makes it more difficult to understand how effective their method is.

In this chapter, we propose a new method for inverse problems. The proposed method uses a discriminator and a generator of Wasserstein GAN (WGAN) [61]. WGAN minimizes the distance between the probability distribution of true image data set and generated image data set. WGAN ensures it can reduce noise and the underlining density distribution of the de-noised image is also close enough to the one of the true image data set. In addition, the proposed method uses maximum likelihood estimation (MLE) regularizer to help the algorithm to keep pixel fidelity and make the resulting images look more genuine. Multiple loss functions are used in the regularizer. To utilize multiple loss functions in the regularization, a MLE approach is used to balance multiple losses in the proposed algorithm. This is because each loss function has a different unit scale. Also, some loss function is more reliable than others. MLE regularizer is introduced to utilize multiple loss functions more effectively. Furthermore, we conduct image texture analysis whether the proposed method preserves image texture. PSNR and SSIM analysis are also performed as many other inverse problem research did.

The contribution of this chapter is two-fold. We will show this WGAN method can preserve image texture in inverse problems while PSNR and SSIM are kept high level. A convolutional neural network (CNN) with MSE loss function will be used for the comparison. In addition, we will show the effectiveness as to how the MLE regularizer helps utilize multiple loss functions in the regularizer.

The organization of this chapter is the following. The detailed methods of the WGAN algorithm and MLE regularizer will be discussed in the next section. Image texture analysis methods are also discussed in the same section. In the third section, qualitative

and quantitative evaluation results will be shown. In the fourth section, we will discuss the conclusion of this chapter.

## 4.2 Methods

### 4.2.1 Generative adversarial networks

#### 4.2.1.1 Discriminator optimization

The usual deep neural network uses one network. In contrast, GANs use two networks which are called discriminator and generator [22]. Training in the usual deep neural network is a minimization problem where deep neural network tries to generate samples just like data from the training data set. Mathematically, the distance between the training data set and samples generated by a deep neural network is minimized. On the other hand, in GANs, a generator creates fake samples just like true data set and a discriminator tries to distinguish between true data set and fake data set. This is often called a Min-Max problem where the generator minimizes the distance between true and fake data set while discriminator tries to maximize the same distance. The following equation describes the min-max problem [22].

$$\min_G \max_D V(D,G) = \min_G \max_D \left\{ \mathbb{E}_{X\sim p_r}[D(X)] + \mathbb{E}_{Z\sim p_z}[1 - D(G(Z))] \right\} \tag{4.1}$$

where G is generator, D is discriminator, $p_r$ is the training data distribution (true data set) in a compact metric space and $p_z$ is an uniform distribution. $X$ and $Z$ are samples from $p_r$ and $p_z$ respectively. In GANs, the input of generator is a vector of uniform random variables $Z$ where $Z \sim U[0,1]$ and $U$ is the uniform distribution. When GANs are used in inverse problems such as image de-noising or image reconstruction, it is a general practice

to use $X_n \sim p_n$ as the input of generator where $p_n$ is noisy data distribution [35] [17] [18]. Thus, the equation (4.1) is rewritten for inverse problems as

$$\begin{aligned} \min_G \max_D V(D,G) &= \min_G \max_D \Big\{ \mathbb{E}_{X\sim p_r}[D(X)] + \mathbb{E}_{X_n \sim p_n}[1 - D(G(X_n))] \Big\} \\ &= \min_G \max_D \Big\{ \mathbb{E}_{X\sim p_r}[D(X)] + \mathbb{E}_{\hat{X}\sim p_g}[1 - D(\hat{X})] \Big\} \end{aligned} \tag{4.2}$$

where $p_g$ is the generated distribution and $\hat{X} = G(X_n)$. In fact, $X_n$ is generated by an inverse operator $A^+$ with a noisy measurement $y$ in inverse problems, such that $X_n = A^+ y$. The inverse operator $A$ is the identical operator $I$ in a de-noising problem. $A$ is an up-sampling operator in a super resolution problem. Filtered backprojection is the most frequently used inverse operator in CT reconstruction problems. Furthermore, the noisy measurement $y$ is a noisy input image in a de-noising problem, a low-resolution input image in a super resolution problem, and an input noisy sinogram in a CT reconstruction problem. Therefore, $\hat{X} = G(A^+ y)$.

Whether using Z or $X_n$ as the input of GAN's generator is worthwhile to have more comment. GAN framework is often used to generate a human face or a bedroom which has never existed. In this setting, the GAN framework is used to learn the density of data distribution. Thus, it can generate common human faces that are extracted from the peak of the learned distribution. It can also generate rare human faces that are extracted from the further left or right bottom of the learned distribution. Therefore, we can say the GAN framework learns conditional density $p(Y|Data)$ where $Y$ is a random variable and $Data$ is training data distribution. On the other hand, in inverse problems, we do not need to learn the conditional density. Instead, we want to learn the conditional expectation $\mathbb{E}[Y|Data]$ which provides the most probable estimate of Y given training distribution. Thus, we just want to learn the mean value of the distribution. This is why we use $X_n$ as the input of the generator instead of $Z$ in inverse problems.

The optimized discriminator $D$ given any generator $G$ is to maximize V(D, G).

$$
\begin{aligned}
D_G^* &= \arg\max_D V(D,G) \\
&= \arg\max_D \left\{ \mathbb{E}_{X\sim p_r}[D(X)] + \mathbb{E}_{\hat{X}\sim p_g}[1 - D(\hat{X})] \right\} \\
&= \arg\min_D \left\{ \mathbb{E}_{X\sim p_g}[D(\hat{X})] - \mathbb{E}_{X\sim p_r}[D(X)] \right\}
\end{aligned} \tag{4.3}
$$

WGAN uses Wasserstein distance as known as Earth Mover's (EM) distance to obtain the distance of equation (4.3) [60]. In fact, the discriminator is called as critic in WGAN because it cannot really discriminate between real and fake images in WGAN. This is one difference from GANs. WGAN also requires that the critic must lie within the space of 1-Lipschitz functions to keep the continuity and differentiability of its loss function where WGAN enforces through weight clipping [60]. However, weight clipping in WGAN leads to optimization difficulties [61]. Gulrajani et al [61] proposed to use a gradient penalty to overcome this problem.

$$
D_G^* = \arg\min_D \left\{ \mathbb{E}_{\hat{X}\sim p_g}[D(\hat{X})] - \mathbb{E}_{X\sim p_r}[D(X)] + \mu \mathbb{E}_{\tilde{X}\sim p_i}[(\|\nabla_{\tilde{X}} D(\tilde{X})\| - 1)^2] \right\} \tag{4.4}
$$

where $p_i$ is the sampling distribution which is sampled uniformly along straight lines between pairs of points sampled from the true data distribution $p_r$ and the generated data distribution $p_g$ [61]. Also, $\mu$ is called penalty coefficient and it was suggested to use $\mu = 10$ [61]. In summary, the equation (4.4) can be mathematically simplified to the following expression and this is how the critic is optimized.

$$
\begin{aligned}
D_G^* &= \arg\max_D \Big\{ \text{Wass}(p_r, p_g) \Big\} \\
&= \arg\max_D \inf_{\gamma \in \Pi(p_r, p_g)} \Big\{ \mathbb{E}_{(X,\hat{X})\sim\gamma}[\|D(X) - D(\hat{X})\|] \Big\} \\
&= \arg\max_D \sup_{D \in \mathbf{D}_{1-Lip(p_r,p_g)}} \Big\{ \mathbb{E}_{X\sim p_r}[D(X)] - \mathbb{E}_{\hat{X}\sim p_g}[D(\hat{X})] \Big\} \\
&= \arg\min_D \inf_{D \in \mathbf{D}_{1-Lip(p_r,p_g)}} \Big\{ \mathbb{E}_{\hat{X}\sim p_g}[D(\hat{X})] - \mathbb{E}_{X\sim p_r}[D(X)] \Big\},
\end{aligned}
\tag{4.5}
$$

where $\mathbf{D}_{1-Lip}$ is a set of 1-Lipschitz functions for the joint probability distributions of $p_r$ and $p_g$. $\Pi(p_r, p_g)$ is the set of all transport plans for joint distributions $p_r$ and $p_g$. $\gamma$ indicates how much "mass" or "earth" needs be transported from $X$ to $\hat{X}$ to transform the distribution $p_r$ into the distrubtion $p_g$, which is why Wasserstein distance is called Earth Mover (EM) distance. $\text{Wass}(\cdot, \cdot)$ is the Wasserstein distance between two probability distributions. The second line of equation (4.5) is known highly intractable [60]. On the other hand, the third line of equation (4.5) is known to be equivalent to the second line according to the Kantorovich-Rubinstein duality [60]. In summary, the equation (4.5) shows that the optimal critic can be obtained by maximizing Wasserstein distance of these two probability distributions.

#### 4.2.1.2 Generator optimization

Similarly, the optimized generator $G$ given any critic $D$ is to minimize V(D, G).

$$
\begin{aligned}
G_D^* &= \arg\min_G V(D, G) \\
&= \arg\min_G \Big\{ \mathbb{E}_{X\sim p_r}[D(X)] + \mathbb{E}_{\hat{X}\sim p_g}[1 - D(\hat{X})] \Big\} \\
&= \arg\min_G \Big\{ - \mathbb{E}_{\hat{X}\sim p_g}[D(\hat{X})] \Big\}
\end{aligned}
\tag{4.6}
$$

In the above equation, $\mathbb{E}[D(X)]$ is dropped because the training data set is constant and the critic is not being optimized here. Thus $D(X)$ is constant.

#### 4.2.1.3 Generator regularization

Equation (4.6) is how the generator is optimized. This is sufficient to obtain the conditional expectation of data distribution. However, when it comes to inverse problems in medical imaging, the estimation of this conditional expectation often diverges from the input noisy data. It is important in medical imaging to ensure pixel fidelity is maintained. To improve pixel-level integrity, we use a regularizer in generator optimization.

$$G_D^* = \arg\min_G \Big\{ -\mathbb{E}_{\hat{X}\sim p_g}[D(\hat{X})] + \mathbb{R}(p_r, p_g) \Big\} \tag{4.7}$$

To ensure pixel-level integrity and improve image look and feel, we use MSE loss and perception loss respectively in the regularization. These two loss functions have been previously researched and they were found effective in inverse problems [17] [18].

$$\mathbb{R}(p_r, p_g) = \lambda_1 \mathbb{E}_{\substack{X\sim p_r \\ \hat{X}\sim p_g}}[\|X - \hat{X}\|^2] + \lambda_2 \mathbb{E}_{\substack{X\sim p_r \\ \hat{X}\sim p_g}}[(\Psi_\theta(X) - \Psi_\theta(\hat{X}))^2], \tag{4.8}$$

where $\lambda_1$ and $\lambda_2$ are regularization parameters. $\Psi_\theta$ is a parametrized functional in a compact metric space. The first term is MSE loss and the second term is perception loss in this chapter although this equation can be extended to other loss functions.

#### 4.2.1.4 Multiple loss likelihoods in regularization

Loss functions have significant roles in generator performance. However, tuning these regularization parameters by hand is a difficult and expensive process. In fact, the performance of a generator is highly dependent on an appropriate tuning of regularization parameters. However, using multiple loss functions poses a few problems. The first problem is that each loss function is on a different measurement scale. The magnitude of the

noise of loss function is also different. The second problem is that some loss functions are more reliable than others. We would like to use higher weights on reliable loss functions but use lower weights on unstable loss functions. A similar problem was tackled by multi-task learning [71]. As the name suggests, multi-task learning has multiple learning objectives. Thus, it has one neural network but has multiple outputs where multiple loss functions need to be calculated. Kendall et al [71] proposed a method to simultaneously learn various losses of varying quantities and units rather than manually tweaking parameters for multi-task learning. In this chapter, we were inspired by their method and applied the concept to the generator regularization.

In the following, we derive a regularization parameter optimization scheme based on maximizing the Gaussian likelihood. Suppose $X$ is a true image from data distribution and $\hat{X}$ is a fake image created by the generator. Both of them are random variables. Also, suppose $F(\cdot)$ is a parameterized function in a compact metric space such as an adversarial critic. Then the noise of the function can be modeled as follows.

$$F(X) - F(\hat{X}) = \epsilon, \tag{4.9}$$

where $\epsilon$ is a noise. We can consider this noise as an adversarial loss or a perception loss in the context of this chapter. The equation (4.9) can be written as the following.

$$F(X) = F(\hat{X}) + \epsilon. \tag{4.10}$$

Then, the conditional density distribution of $F(X)$ given $F(\hat{X})$ can be expressed as follows.

$$p(F(X)|F(\hat{X})) = \mathcal{N}(F(\hat{X}), \sigma^2), \tag{4.11}$$

Here, we assume this conditional density distribution follows Gaussian distribution with the function of generated image as the mean and $\sigma$ as the standard deviation. The logarithm of equation (4.11) is proportional to the following.

$$\log p(F(X)|F(\hat{X})) \propto -\frac{1}{2\sigma^2}\|F(X) - F(\hat{X})\|^2 - \log \sigma \tag{4.12}$$

In case of multiple functions, we need to derive the joint probability of these functions. The joint probability distribution of $F_1(X), F_2(X)$ given $F_1(\hat{X}), F_2(\hat{X})$ are the following.

$$p(F_1(X), F_2(X)|F_1(\hat{X}), F_2(\hat{X})) = p(F_1(X)|F_1(\hat{X}))\ p(F_2(X)|F_2(\hat{X})), \tag{4.13}$$

where we assume $F_1(X), F_2(X)$ are independent each other. To obtain optimized standard deviation $\sigma_1^*, \sigma_2^*$, we will need to maximize the logarithm of the joint probability distribution described in equation (4.13).

$$\begin{aligned}
\sigma_1^*, \sigma_2^* &= \arg\max_{\sigma_1,\sigma_2\in\pi} \Big\{ \log p(F_1(X), F_2(X)|F_1(\hat{X}), F_2(\hat{X})) \Big\} \\
&= \arg\min_{\sigma_1,\sigma_2\in\pi} \Big\{ \frac{1}{2\sigma_1^2} \mathbb{E}_{\substack{X\sim p_r \\ \hat{X}\sim p_g}} [\|F_1(X) - F_1(\hat{X})\|^2] + \\
&\qquad \frac{1}{2\sigma_2^2} \mathbb{E}_{\substack{X\sim p_r \\ \hat{X}\sim p_g}} [\|F_2(X) - F_2(\hat{X})\|^2] + \log \sigma_1\sigma_2 \Big\}.
\end{aligned} \tag{4.14}$$

MSE loss and perception loss are used in this chapter. Equation (4.14) can be rewritten as the following.

$$\begin{aligned}
\sigma_1^*, \sigma_2^* = \arg\min_{\sigma_1,\sigma_2\in\pi} \Big\{ &\frac{1}{2\sigma_1^2} \mathbb{E}_{\substack{X\sim p_r \\ \hat{X}\sim p_g}} [\|X - \hat{X}\|^2] + \\
&\frac{1}{2\sigma_2^2} \mathbb{E}_{\substack{X\sim p_r \\ \hat{X}\sim p_g}} [(\Psi_\theta(X) - \Psi_\theta(\hat{X}))^2] + \log \sigma_1\sigma_2 \Big\},
\end{aligned} \tag{4.15}$$

where $\Psi_\theta$ is a parameterized functional to calculate perception of the input image. Therefore, the first term is weighted MSE loss. The second term is weighted perception loss. The third term can be thought as a regularization term for $\sigma_1, \sigma_2$ to avoid them going to infinity. These weights $\frac{1}{2\sigma_1^2}$ and $\frac{1}{2\sigma_2^2}$ control each loss function to normalize the losses. In addition, these weights can increase or decrease depending upon how reliable each loss function is.

Finally, the regularization term described in equation (4.8) can be rewritten to

$$\mathbb{R}(p_r, p_g) = \frac{\lambda_1}{2\sigma_1^2}\mathbb{E}_{\substack{X\sim p_r \\ \hat{X}\sim p_g}}[\|X - \hat{X}\|^2] + \frac{\lambda_2}{2\sigma_2^2}\mathbb{E}_{\substack{X\sim p_r \\ \hat{X}\sim p_g}}[(\Psi_\theta(X) - \Psi_\theta(\hat{X}))^2]. \tag{4.16}$$

Notice that $\lambda_1, \lambda_2$ are still used in the equation. This is because each loss function is normalized by its scale and the impact of each loss function is effectively equalized after the normalization assuming both of them are equally reliable measures. In this case, we would still like to have an additional adjustment to put more weight on a favorable loss function and less weight on the other. This is why the regularization parameter $\lambda_1, \lambda_2$ remain in the equation (4.16). In case a loss function is not reliable, the regularization scaling factor $\sigma$ becomes bigger and the influence of the loss function will be minimized. Thus, the effect of the regularization parameter $\lambda$ gets reduced even if it is set high. In this chapter, the parameter $\lambda$ is called regularization parameter lambda or simply regularization lambda for a convenience. The parameter $\sigma$ is called regularization scaling factor sigma or simply regularization sigma in short.

A pseudo-code of critic and generator optimization is shown in Algorithm 3.

**Algorithm 3** Critic and generator optimization in TextureWGAN

**Require:** The number of critic optimization steps K, batch size M, inverse $A^+$, gradient penalty coefficient $\mu$, optimization hyper-parameters $\alpha_D, \alpha_G, \alpha_\sigma$, regularization lambda $\lambda_1, \lambda_2$, parametric perception functional $\Psi_\theta(\cdot)$

**while** $\eta_D$ and $\eta_G$ have not converged **do**

/*
* Critic optimization
*/

**for all** $k \in 1, ..., K$ **do**

**for all** $m \in 1, ..., M$ **do**

Sample ground truth image $x \sim p_r$ and noisy measurement $y \sim p_y$

Sample random number $\epsilon \sim U[0, 1]$

$x_n \leftarrow A^+ y$

$\hat{x} \leftarrow G_D^*(x_n)$

$\tilde{x}_m \leftarrow \epsilon x + (1 - \epsilon)\hat{x}$

$L_m^D \leftarrow D_G^*(\hat{x}) - D_G^*(x) + \mu(\|\nabla_{\tilde{x}_m} D_G^*(\tilde{x}_m)\| - 1)^2$ ▷ Loss of D

**end for**

$\eta_D \leftarrow Optimizer(\nabla_{\eta_D} \sum_{m=1}^{M} L_m^D, \alpha_D)$ ▷ network parameters of D

**end for**

/*
* Generator optimization
*/

**for all** $m \in 1, ..., M$ **do**

Sample ground truth image $x \sim p_r$ and noisy measurement $y \sim p_y$

$x_n \leftarrow A^+ y$

$\hat{x} \leftarrow G_D^*(x_n)$

$L_m^G \leftarrow -D_G^*(\hat{x}) + \frac{\lambda_1}{2\sigma_1^2}\|x - \hat{x}\|^2 + \frac{\lambda_2}{2\sigma_2^2}(\Psi_\theta(x) - \Psi_\theta(\hat{x}))^2$ ▷ Loss of G

$L_m^\sigma \leftarrow \frac{1}{2\sigma_1^2}\|x - \hat{x}\|^2 + \frac{1}{2\sigma_2^2}(\Psi_\theta(x) - \Psi_\theta(\hat{x}))^2 + \log \sigma_1 \sigma_2$ ▷ Loss of sigma

**end for**

$\eta_G \leftarrow Optimizer(\nabla_{\eta_G} \sum_{m=1}^{M} L_m^G, \alpha_G)$ ▷ network parameters of G

$\sigma_1, \sigma_2 \leftarrow Optimizer(\nabla_{\sigma_1, \sigma_2} \sum_{m=1}^{M} L_m^\sigma, \alpha_\sigma)$ ▷ regularization sigma

**end while**

### 4.2.2 Image texture analysis

The final section of the chapter describes image texture analysis. Many texture analysis methods have been proposed, but the most popular and effective texture analysis methods, especially in medical imaging, are first-order and second-order statistical texture analysis [6].

#### 4.2.2.1 First-order statistical texture analysis

First-order texture analysis measures utilize the image histogram, or pixel occurrence probability, to measure texture. The primary interest of this approach is its simplicity because it uses standard descriptions such as mean and variance to measure texture [6]. However, this approach has a certain limitation in differentiating among unique textures since the method does not account for the spacial relationship of data. In general, mean, variance and entropy are popular choices of first-order texture analysis [6]. The following is an example of how to calculate the local standard deviation of a two-dimensional image.

$$S_{i,j} = \sqrt{\frac{1}{N} \sum_{k \in \pi_{ij}} (x_k - \bar{x}_{ij})^2}, \tag{4.17}$$

where $\pi_{ij}$ is the neighborhoods of pixel location (i, j). N is the total number of neighborhoods. $\bar{x}_{ij}$ is the mean of the neighborhoods.

In this chapter, we implemented rangefilt, stdfilt and entropyfilt of Matlab functions [64] in python for the first-order statistical analysis to make the analysis more integrated into the other part of the system such as deep neural networks.

#### 4.2.2.2 Second-order statistical texture analysis

The most regularly used method for texture analysis is based on obtaining various textural features from a gray level co-occurrence matrix (GLCM) [66]. There are plenty of research papers recommending the GLCM and other texture features are significantly associated with patient survival in cancer diagnostics in medical imaging [5]. In fact, Eilaghi et al. [5] suggested the GLCM feature set is one of the best-accepted tools for texture analysis in medical imaging. Their research showed that texture analysis is more strongly associated with patient survival rate than tumor size in their medical imaging cancer research.

The GLCM of an image is an measurement of the second-order joint probability, $P_\delta(i,j)$ of the intensity values of two pixels (i and j), a distance $\delta$ apart along a given direction $\theta$ [66].

If an image does not contain any texture (i.e. the intensity is flat), the obtained GLCM would be completely diagonal [66]. As the image texture becomes rich (i.e. as the local pixel intensity variations increase), the off-diagonal values in the GLCM become larger [66].

Haralick et al. [63] suggested using GLCMs calculated from four displacement vectors with $\delta$=1, or 2 pixels and $\theta = 0°, 45°, 90°, 135°$.

Haralick et al. [63] also proposed a quantitative analysis of the GLCM through 14 textural descriptors computed from $P_\delta$, although generally only a few of these are commonly used [66]. In this chapter, four of the most commonly used descriptors (contrast, correlation, energy and homogeneity) are used to extract texture features from the 35 GLCMs of the image dataset [65] [66].

$$
\begin{aligned}
Contrast &= \sum_{i,j} |i-j|^2 P_\delta(i,j), \\
Correlation &= \sum_{i,j} \frac{(i-u_i)(j-u_j)P_\delta(i,j)}{\sigma_i \sigma_j}, \\
Energy &= \sum_{i,j} P_\delta(i,j)^2, \\
Homogeneity &= \sum_{i,j} \frac{P_\delta(i,j)}{1+|i-j|},
\end{aligned}
\tag{4.18}
$$

where i and j are column and row index of a given GLCM. $u_i, u_j, \sigma_i, \sigma_j$ are the means and standard deviations of column and row direction of a given GLCM.

Contrast measures the amount of local variations in an image. Correlation represents a measure of pixel-wise linear dependencies within the image. A higher correlation value

means a more steady texture value along certain lines [66]. Energy as known as Angular Second Moment provides the sum of squared elements of GLCM and it gives a level of uniformity of an image. Homogeneity measures the closeness of the distribution of elements in GLCM to the GLCM diagonal [66]. In other words, it measures how much each element of GLCM are close to each other. If GLCM is diagonal, then Homogeneity is one [62].

In this chapter, the GLCM offset was set to 1 pixel for the spatial relationship between adjacent pixels ($\delta = 1$). We also used four angles ($\theta = 0°, 45°, 90°, 135°$) in the calculation of GLCM. In addition, we implemented the GLCM and texture analysis tools of Matlab [65] in python for the second-order statistical analysis as well with the same reason from the first-order statistical analysis.

## 4.3 Experiments

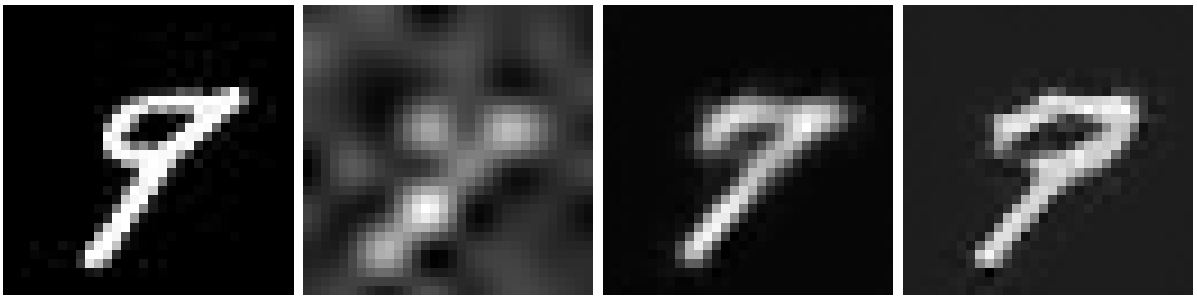
In this chapter, we conducted three experiments to evaluate the proposed method. First of all, we evaluated our method along with other well-known methods in super-resolution to understand the effect at a high level. Since we used the MNIST data set and those images are quite simple, we did not perform any quantitative analysis. Second, we conducted our evaluation in image de-noising with natural images. We used PSNR, SSIM, first-order and second-order statistical texture analysis in de-noising to understand the effectiveness of our proposed method. Finally, we also assessed it in Computed Tomography image reconstruction.

In these three experiments, we took 65,000 training steps to train the generator network. As the previous research papers suggested [23], the critic was trained five times per each one generator training step to stabilize the critic training. In addition, we trained the critic 5000 times before starting generator training. We used UNet architecture [24]

for the generator. It has four down-sampling and four up-sampling convolutional layers respectively with skip-connections. Leaky ReLU activators were used. For the critic, we used a simple eight-layer CNN with a total of four strided convolution layers with stride 2, Leaky ReLU (alpha=0.1) activations and two top dense layers [32]. We used RMSProp for the generator and critic optimization. We did not use Batch Normalization [54] in the neural networks. Instead, we normalized input images to [0, 1] per prior research suggestions [32] [31]. We used the VGG19 pre-trained network [67] from Keras [68] for perception loss. We did not use the top three fully connected layers of VGG19. Therefore, the total number of pre-trained network layers was 16. The mini-batch size was chosen to be 16. Lambda for MSE was set to 0.9 to maintain pixel fidelity on the output results. Lambda for perception loss was set to 0.001 to keep the high level of pixel fidelity. Our experimental results showed that image texture improved as the lambda of perception loss increased while PSNR and SSIM were deteriorated. This is in fact consistent to the previous research [17]. Since this method aims at medical imaging problems, we chose such low lambda of perception loss to keep pixel fidelity high on resulting images.

For the comparison, we used a convolutional neural network (CNN) with the UNet architecture. The same architecture was used for the generator of TextureWGAN. The difference is that the CNN uses only MSE as its loss function. This method is labeled as "MSE 100%". In addition, we used non-local mean filter as a comparison, which is to find image patches across the entire image and averages these patches to reduce noise [55]. We used the suggested hyper-parameters for this filter described in the original paper [55]. This method is labeled as "NLM Filter".

Furthermore, as it was described in the introduction section of this chapter, many clinicians use a blending factor to avoid image texture change with de-noised images. To compare this image blending scheme against other methods, we create an MSE 50% image by averaging MSE 100% image and the noisy input image. The reason why we call



(A) (B) (C) (D)

FIGURE 4.1: MNIST super-resolution results. (a) the true image, (b) bicubic interpolation result of input noisy image, (c) MSE 100% and (d) TextureWGAN

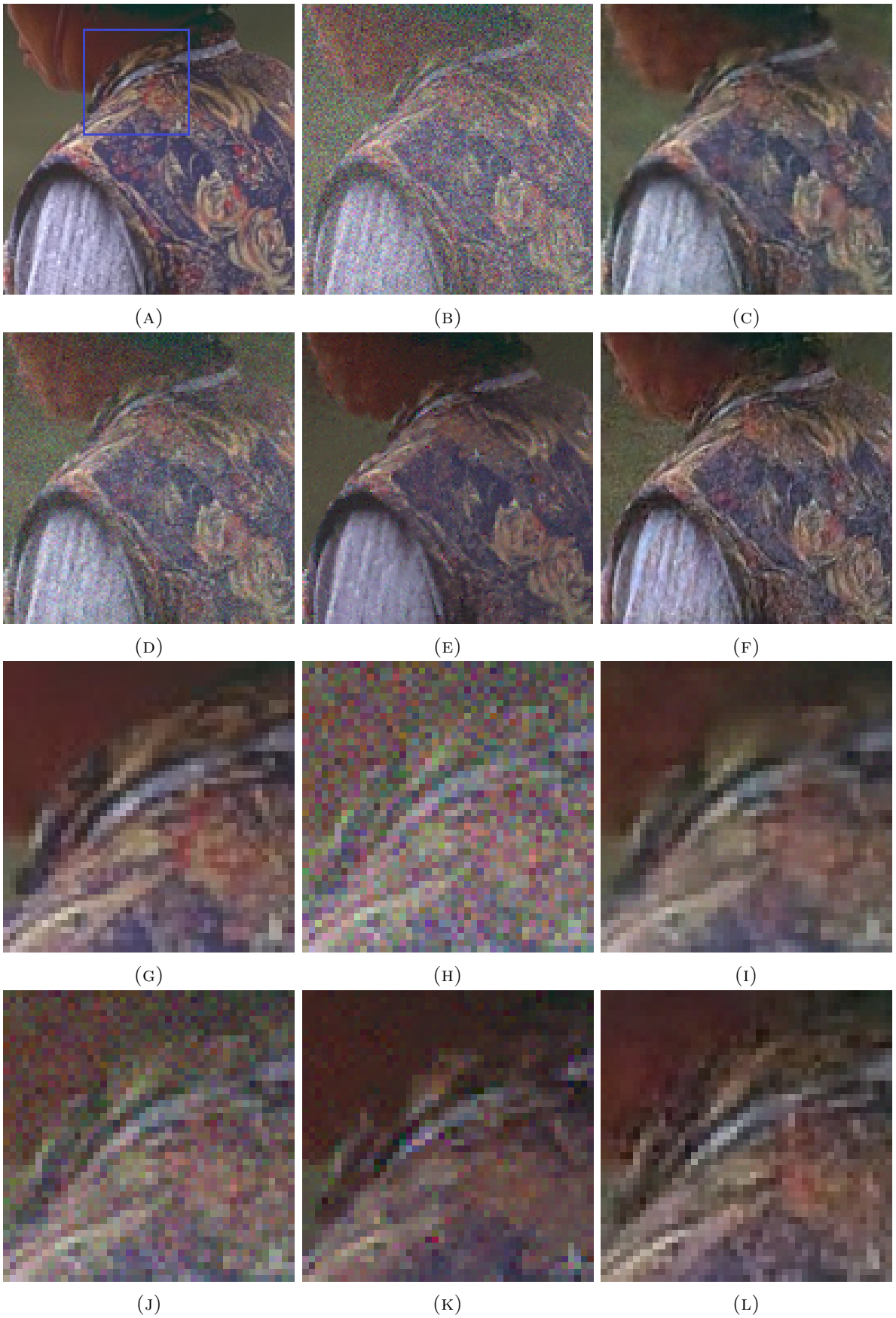
this image MSE 50% is that 50% of the pixel value of each image comes from MSE 100% and the other 50% comes from the noisy input image.

### 4.3.1 Super-resolution

For super-resolution, we used the MNIST data set [69]. We used 60,000 images for training and 10,000 images for evaluation. Each training step randomly chose images from the training data set. Each image was down-sampled 4x, thus the images were transformed from 28x28 pixels to 7x7 pixels. Input noisy images were created by adding 10% Gaussian noise on down-sampled images. Fig.4.1 was one example result. MSE 100% result created a reasonable result considering this very challenging test case. However, the output image became too smooth and it does look quite different from the input image. On the other hand, the result of TextureWGAN looks as good as the one by MSE 100%. However, the output image looks similar to the input image and it does not look over-smoothed.

### 4.3.2 Image de-noising

For image de-noising, we used the BSDS500 data set [70]. We have used 400 images for training and 100 images for evaluation. Each training step randomly chose images from the training data set. All images were down-sampled to 128x128 pixels for training



(A) (B) (C)

(D) (E) (F)

(G) (H) (I)

(J) (K) (L)

FIGURE 4.2: Image de-noising on the BSDS data set. (a) the true image, (b) noise added, (c) MSE 100% (d) MSE 50% (e) Non-local mean filter, (f) TextureWGAN, and (g) the true image (zoom), (h) noise added (zoom), (i) MSE 100% (zoom) (j) MSE 50% (zoom) (k) Non-local mean filter (zoom) and (l) TextureWGAN (zoom)

TABLE 4.1: Quantitative evaluation on natural images (BSDS)

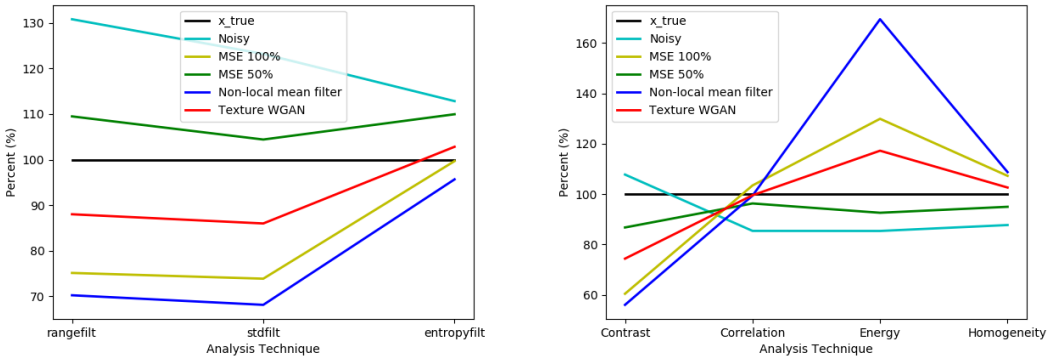
| **Method** | **PSNR (db)** | **SSIM** | **rangefilt (%)** | **stdfilt (%)** | **entropyfilt (%)** |
|---|---|---|---|---|---|
| Original | N/A | N/A | 100.00 | 100.00 | 100.00 |
| +10% Noise | 20.37 | 0.57 | 130.81 | 123.2 | 112.84 |
| MSE 100% | 28.09 | 0.91 | 75.13 | 73.87 | 99.68 |
| MSE 50% | 24.73 | 0.76 | 109.49 | 104.41 | 109.96 |
| NLM Filter | 27.08 | 0.88 | 70.23 | 68.13 | 95.67 |
| TextureWGAN | 27.62 | 0.90 | 88.01 | 85.99 | 102.82 |

| **Method** | **Contrast (%)** | **Correlation (%)** | **Energy (%)** | **Homogeneity (%)** |
|---|---|---|---|---|
| Original | 100.00 | 100.00 | 100.00 | 100.00 |
| +10% Noise | 107.76 | 85.37 | 85.34 | 87.70 |
| MSE 100% | 60.39 | 103.43 | 129.92 | 107.25 |
| MSE 50% | 86.73 | 96.28 | 92.57 | 94.95 |
| NLM Filter | 55.98 | 99.37 | 169.49 | 108.73 |
| TextureWGAN | 74.34 | 99.54 | 117.24 | 102.60 |

efficiency. Input noisy images were created by adding 10% Gaussian noise on input images. Fig.4.2 was one example result. Adding Gaussian noise 10% made this evaluation very challenging. Again, MSE 100% result was reasonable. But the output image again became over-smoothed. The output image appears different from the input image. On the other hand, the result of TextureWGAN looks almost as good as the one by MSE 100%. However, the output image looks visually similar to the input image and it does not look over-smoothed. This over-smoothed effects in MSE 100% and TextureWGAN can be relatively easier to be seen in the zoomed images in Fig.4.2.

We calculated PSNR, Structure Similarity (SSIM), first-order statistical texture analysis (rangefilt, stdfilt and entropyfilt) as well as second-order analysis (Contrast, Correlation, Energy and Homogeneity) to conduct quantitative analysis shown in Table 4.1. All texture analyses were conducted by normalizing all resulting numbers by the results of the original images. This is why all numbers of texture analysis results of original images were 100%. Other numbers were relative to the numbers of original images.

As we can see the results in Table 4.1, PSNR and SSIM of MSE 100% and TextureWGAN

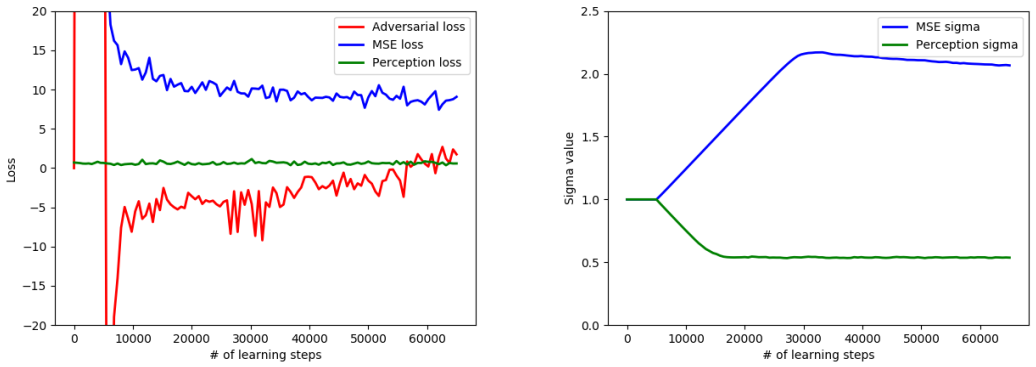



(A) (B)

FIGURE 4.3: Statistical texture analysis of natural images (BSDS). (a) 1st-order analysis, (b) 2nd-order analysis

are on the same level. To make texture analysis results easily visualized, we provided results in graph format in Fig.4.3. Noisy images and MSE 100% were degraded on both first-order and second-order texture results. Non-local mean filter results are similar to the ones for MSE100%. On the other hand, MSE 50% maintained image textures, which is consistent with the previous research conducted by other researchers [62]. As a downside, PSNR and SSIM of MSE 50% were much worse than the ones of MSE 100%. Regarding the results from TextureWGAN, although PSNR and SSIM are satisfactory, both first-order and second-order texture were relatively maintained. The textures of MSE 50% and TextureWGAN are relatively close to each other. These quantitative results show TextureWGAN is capable of keeping high pixel fidelity while preserving image texture.

Finally, Fig.4.4 shows MSE loss, adversarial loss and perception loss along with regularization scaling factor sigma for MSE and perception losses. MSE loss was quite stable after some number of iterations. MSE scaling factor sigma or MSE sigma seems to have converged although it was slightly decreasing after 30,000 iterations. The reason of the decrease after converging is that MSE loss was gradually decreasing overtime. Thus, MSE sigma was trying to adjust the amount of change as it works as a normalization factor. Adversarial loss is known to be volatile because of the difficult nature of GAN/WGAN training [60]. The magnitude of the perception loss was quite small compared to the other




(A) (B)

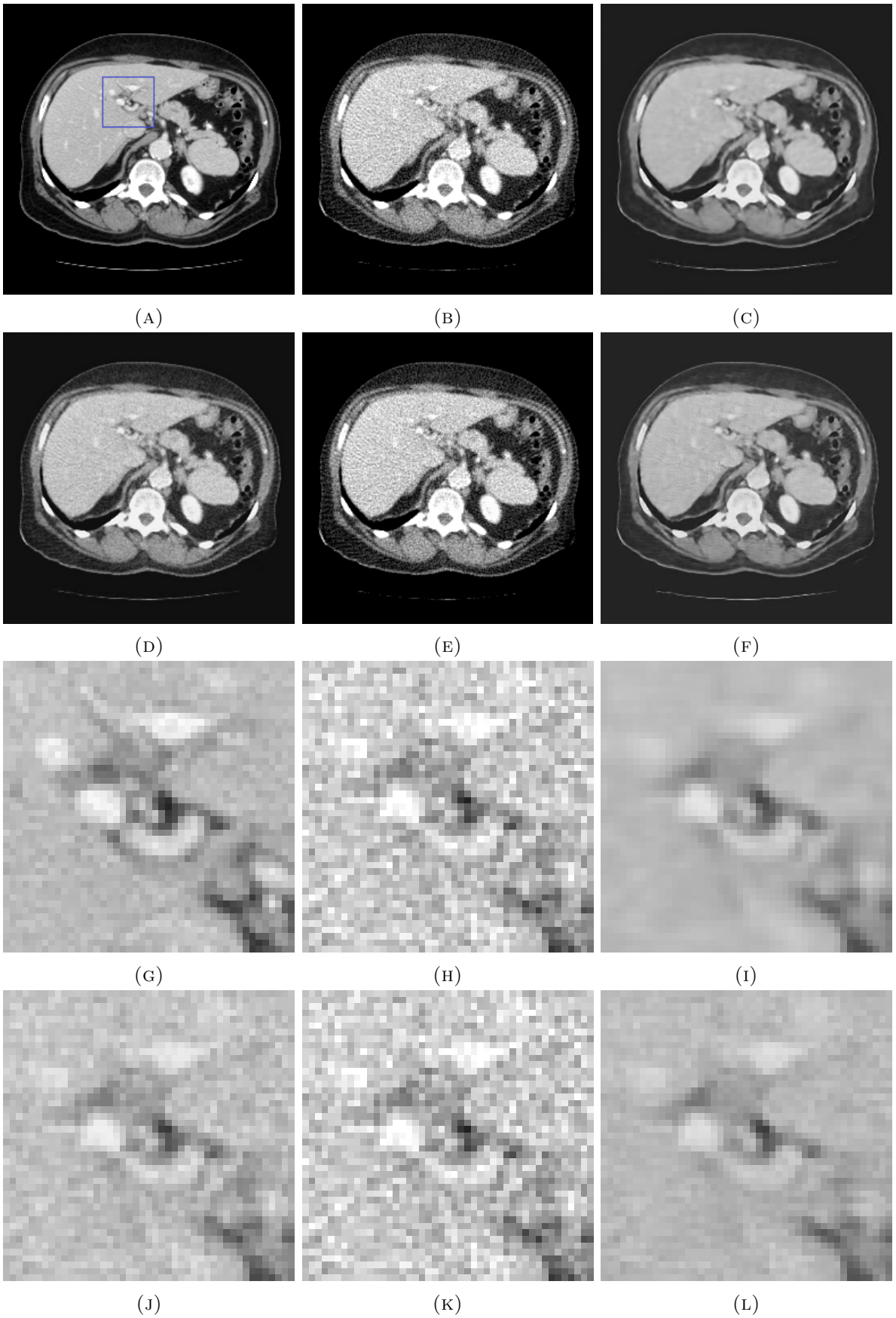
FIGURE 4.4: (a) MSE loss, adversarial loss and perception loss on the BSDS data set, (b) MSE sigma and perception sigma on the BSDS data set

losses and it was very stable. Therefore, the perception sigma converged quickly and it was very stable after a number of iterations.

### 4.3.3 Computed Tomography (CT) image reconstruction

For Computed Tomography image reconstruction, we used LIDC/IDRI data set [58]. We used 1041 images for training and 98 soft tissue-rich images for evaluation. All images were down-scaled from 512x512 to 256x256 pixels for training efficiency. Other training schemes were the same as the ones for image de-noising evaluation. We used the ODL python package [56] to perform the forward projection and the backprojection for CT reconstruction.

To evaluate TextureWGAN, we created low dose CT transmission simulation data by adding statistically independent Poisson noise distribution plus statistically independent Gaussian noise distribution [2]. Poisson noise was needed to simulate a low-dose CT scan because X-ray photon flux obeys Poisson statistics [2]. Poisson noise was added in raw data space after forward projecting CT images of high dose CT scans. In addition, Gaussian noise was added on top of Poisson noise to simulate CT system equipment noise such as CT detector and other electronic system background noise. These two sets



(A) (B) (C)

(D) (E) (F)

(G) (H) (I)

(J) (K) (L)

FIGURE 4.5: CT reconstruction results on LIDC/IDRI data set. (a) true image, (b) filtered backprojection after Poisson and Gaussian noise added, (c) MSE 100% (d) MSE 50%, (e) Non-local mean filter, (f) TextureWGAN, (g) true image (zoom), (h) filtered backprojection (zoom), (i) MSE 100% (zoom), (j) MSE 50% (zoom), (k) Non-local mean filter (zoom), (l) TextureWGAN (zoom)

TABLE 4.2: Quantitative evaluation of CT reconstruction (LIDC/IDRI)

| **Method** | **PSNR (db)** | **SSIM (%)** | **rangefilt (%)** | **stdfilt (%)** | **entropyfilt (%)** |
|---|---|---|---|---|---|
| Original | N/A | N/A | 100.00 | 100.00 | 100.00 |
| FBP | 23.11 | 0.83 | 136.55 | 129.69 | 103.08 |
| MSE100% | 28.42 | 0.94 | 74.34 | 71.36 | 119.17 |
| MSE 50% | 26.33 | 0.90 | 98.10 | 92.78 | 121.93 |
| NLM Filter | 23.10 | 0.83 | 135.56 | 128.75 | 103.70 |
| TextureWGAN | 28.06 | 0.93 | 83.69 | 80.08 | 121.49 |

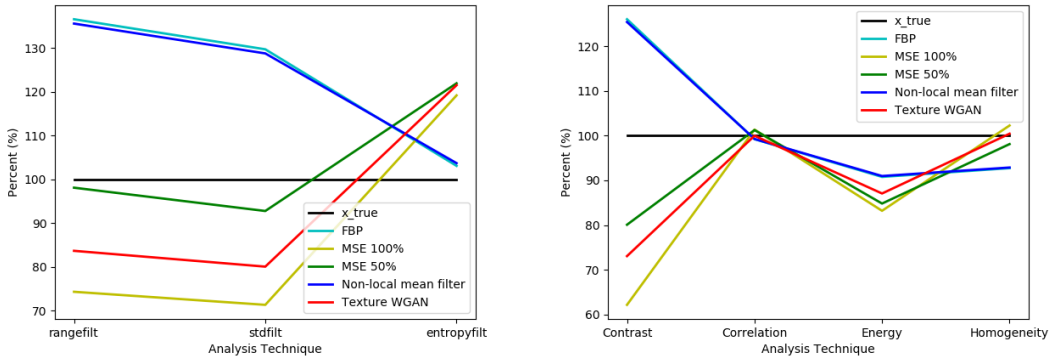
| **Method** | **Contrast (%)** | **Correlation (%)** | **Energy (%)** | **Homogeneity (%)** |
|---|---|---|---|---|
| Original | 100.00 | 100.00 | 100.00 | 100.00 |
| FBP | 126.04 | 99.22 | 90.77 | 92.73 |
| MSE 100% | 62.15 | 101.35 | 83.19 | 102.27 |
| MSE 50% | 80.07 | 101.25 | 84.81 | 98.11 |
| NLM Filter | 125.42 | 99.25 | 90.97 | 92.87 |
| TextureWGAN | 73.05 | 99.98 | 87.06 | 100.43 |

of noise were added not in the sinogram space but in the raw data space, thus we needed to perform the negative log step to bring data back into the sinogram space. After that, we used the ODL backprojector to create input noisy CT images.

The results in CT image reconstruction are consistent with the ones for image de-noising. Even though forward projection and backprojection in CT are not trivial operators, TextureWGAN produced better results. Fig.4.5 showed one of the results. We chose this image since it has plenty of soft tissue pixels. Soft tissue pixels are generally very texture sensitive. As you can see, MSE 100% over-smoothed the resulting image while TextureWGAN did not. Zoomed image of MSE 100% looks clearly different.

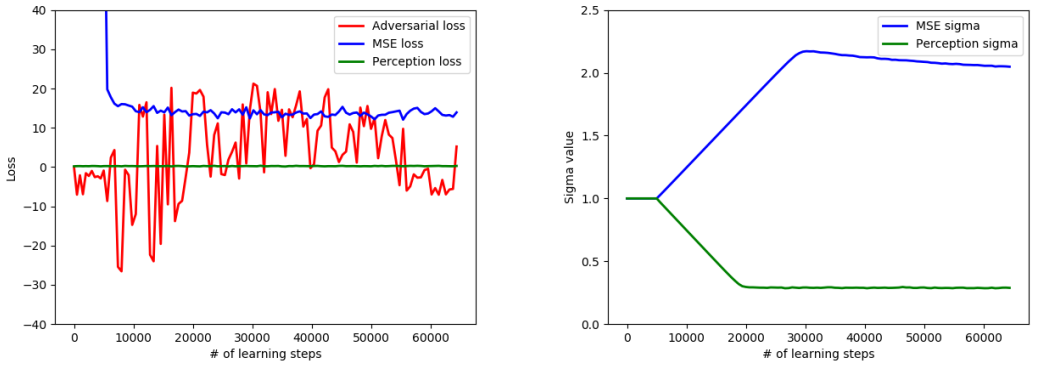
As far as PSNR and SSIM go, again the result of MSE 100% and TextureWGAN are on the same high level. However, the texture results of MSE 100% are worse than those of TextureWGAN and MSE 50%.

As it was stated previously, TextureWGAN can produce better texture images which can be close to MSE 50% by increasing the lambda value of perception loss. However, this




(A) (B)

FIGURE 4.6: Statistical texture analysis on CT reconstruction (LIDC/IDRI). (a) 1st-order analysis, (b) 2nd-order analysis




(A) (B)

FIGURE 4.7: (a) MSE loss, adversarial loss and perception loss on the LIDC/IDRI data set, (b) MSE sigma and perception sigma on the LIDC/IDRI data set

comes with the expenses of PSNR and SSIM deterioration. This is also true for non-local mean filter by adjusting its hyper-parameters. Thus, we used the default hyper-parameters for non-local mean filter as the original paper suggested [55] to keep high PSNR and SSIM.

MSE, adversarial and perception losses provided similar results as ones for image de-noising. The MSE and perception sigma results are also very similar to the results of de-noising.

## 4.4 Summary

We have proposed the new algorithm for inverse problems. The proposed method used the WGAN framework along with the MLE regularizer. The algorithm was evaluated in super-resolution, image de-noising and Computed Tomography image reconstruction. This research showed that the WGAN framework was effective to preserve image texture in inverse problems as other inverse problems often destroy image texture. We also showed that MLE regularizer automatically adjusted regularization parameters based on the scale and reliability of each loss function. The regularization was helpful to ensure high level of pixel fidelity with the proposed method. The proposed method produced the same level of PSNR and SSIM with those of the popular CNN based architecture with MSE loss function.

In addition, we have performed through image texture analysis with various methods. CNN based architecture with MSE loss produced the high PSNR and SSIM, however, the image texture was damaged. We also showed that the blending factor method, which used CNN based architecture and input noisy image, preserved image texture which was consistent with other researchers' findings. This method, however, came with the expenses of degraded PSNR and SSIM. Finally, we have shown that the proposed method preserved image texture while it produced high PSNR and SSIM. Image texture analysis was conducted by visual inspection along with first-order and second-order statistical texture analysis with gray-level co-occurrence matrix.

The proposed method is particularly suitable for medical imaging where pixel fidelity and image texture need to be retained for clinical diagnostics. We plan to use more loss functions to try to maximize the performance of the proposed method.

# Appendix

## Further qualitative results

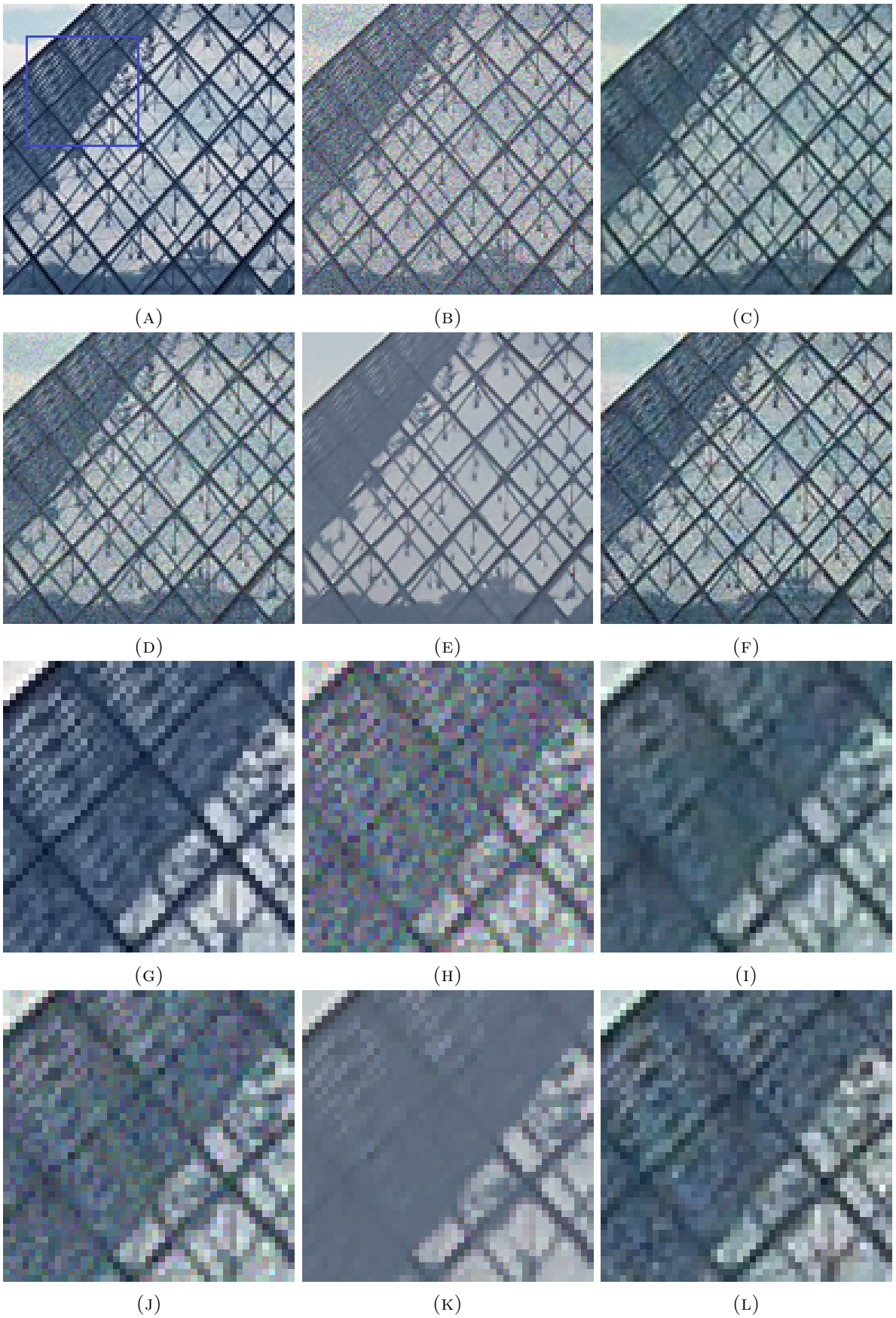
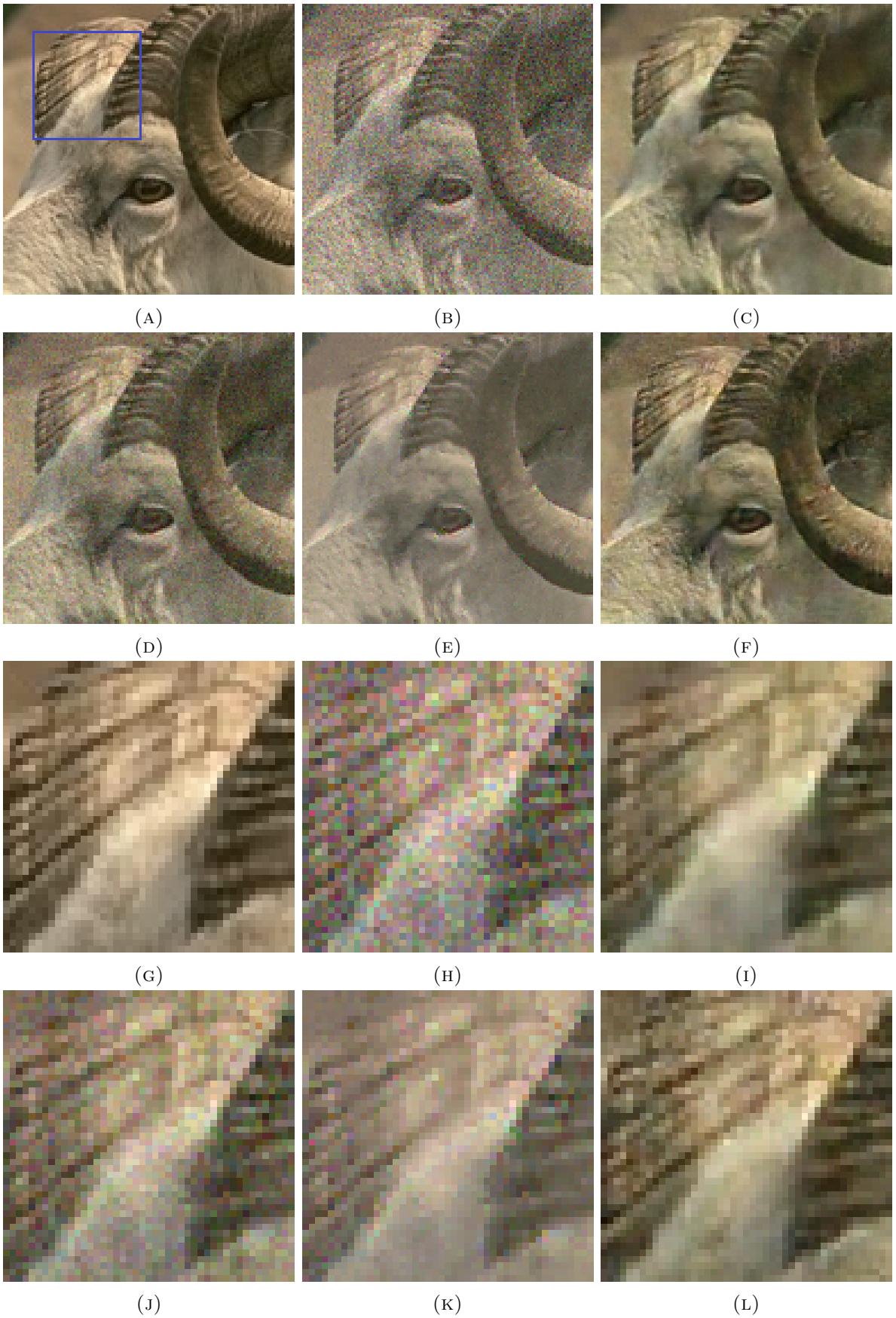



FIGURE 4.8: The second example of image de-noising results on the BSDS data set. (a) the true image, (b) noise added, (c) MSE 100% (d) MSE 50% (e) Non-local mean filter, (f) TextureWGAN, and (g) the true image (zoom), (h) noise added (zoom), (i) MSE 100% (zoom) (j) MSE 50% (zoom) (k) Non-local mean filter (zoom) and (l) TextureWGAN (zoom)



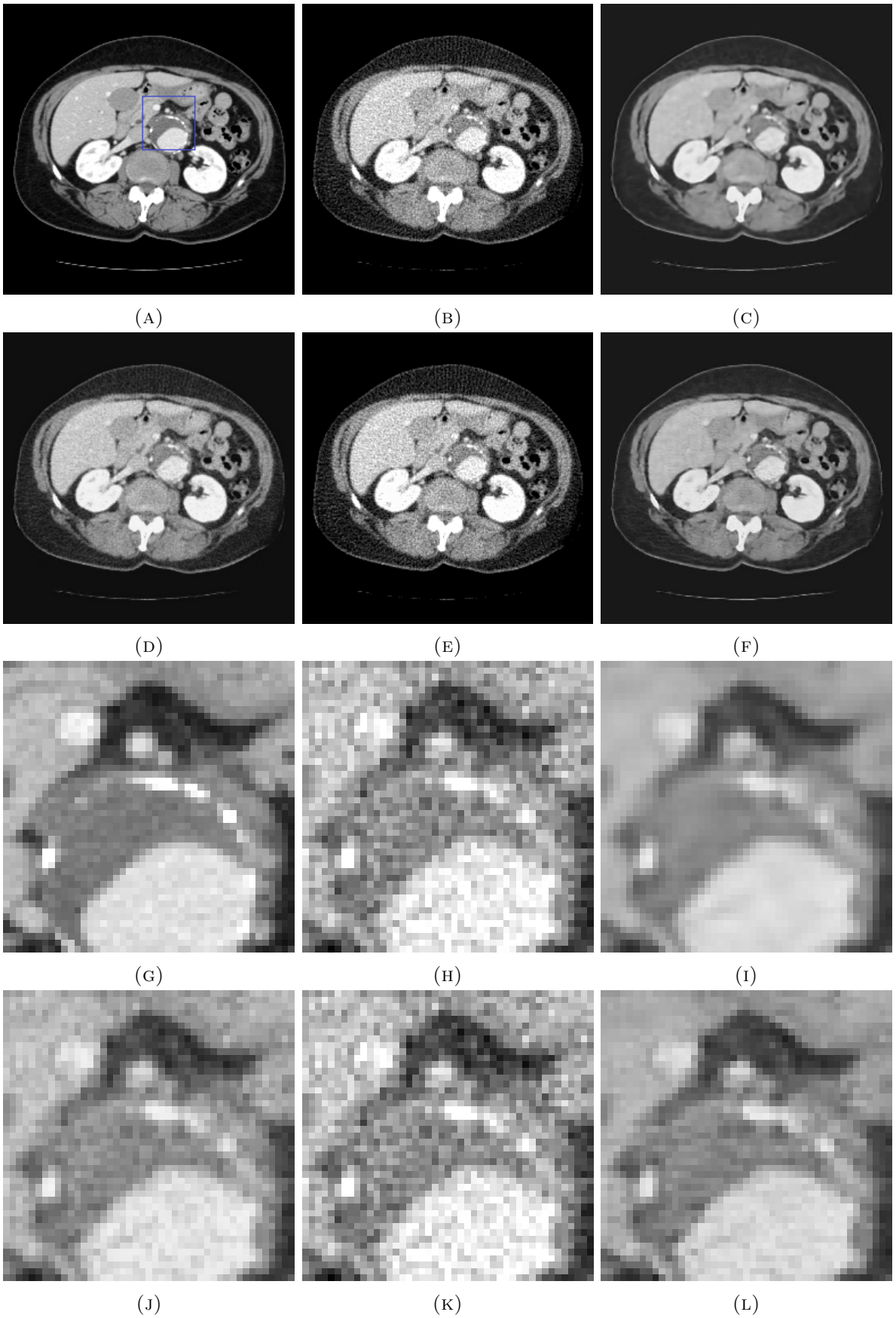
(A) (B) (C)

(D) (E) (F)

(G) (H) (I)

(J) (K) (L)

FIGURE 4.9: The third example of image de-noising results on the BSDS data set. (a) the true image, (b) noise added, (c) MSE 100% (d) MSE 50% (e) Non-local mean filter, (f) TextureWGAN, and (g) the true image (zoom), (h) noise added (zoom), (i) MSE 100% (zoom) (j) MSE 50% (zoom) (k) Non-local mean filter (zoom) and (l) TextureWGAN (zoom)



(A) (B) (C)

(D) (E) (F)

(G) (H) (I)

(J) (K) (L)

FIGURE 4.10: The second example of CT reconstruction results on LIDC/IDRI data set. (a) true image, (b) filtered backprojection after Poisson and Gaussian noise added, (c) MSE 100% (d) MSE 50%, (e) Non-local mean filter, (f) TextureWGAN, (g) true image (zoom), (h) filtered backprojection (zoom), (i) MSE 100% (zoom), (j) MSE 50% (zoom), (k) Non-local mean filter (zoom), (l) TextureWGAN (zoom)

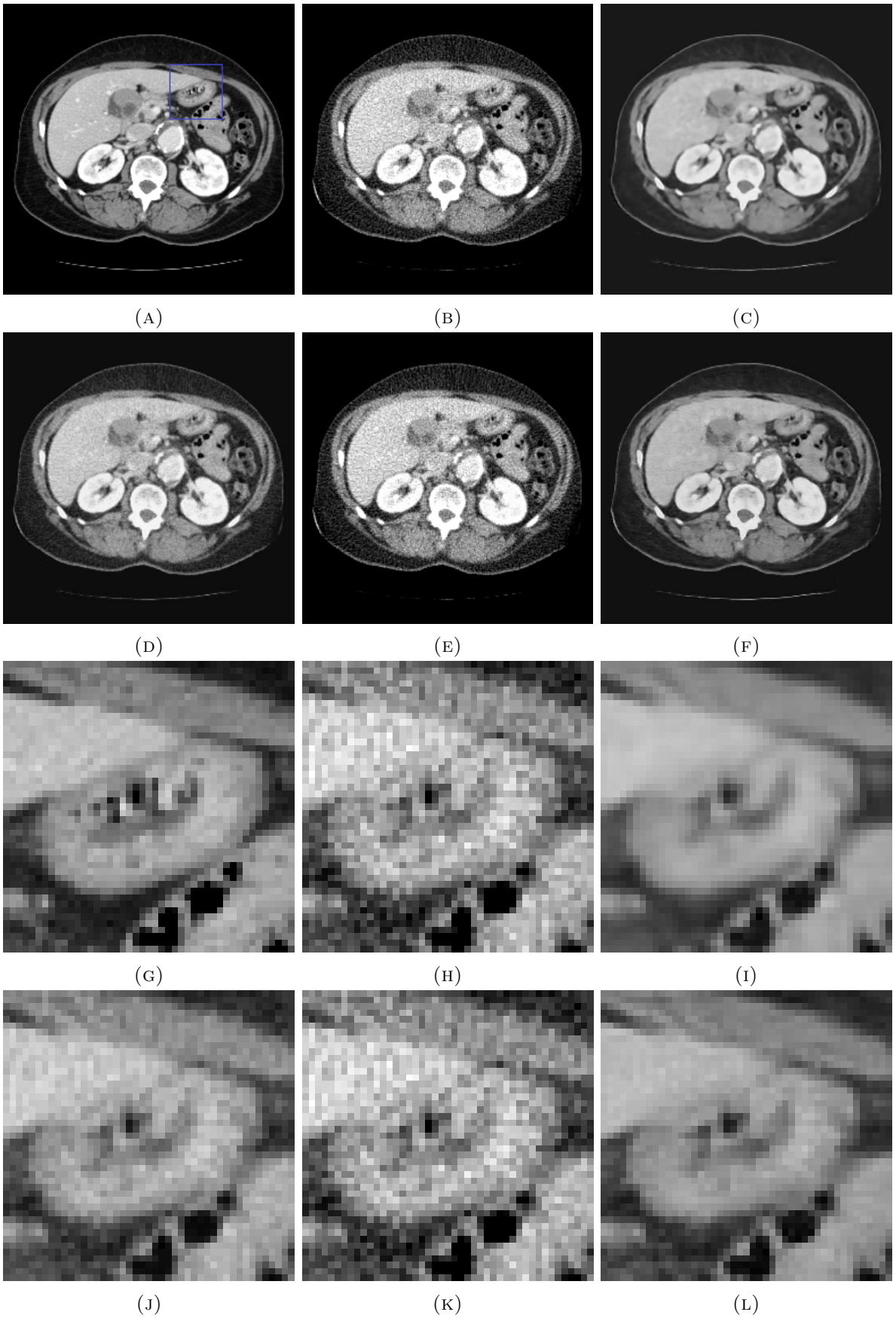



FIGURE 4.11: The third example of CT reconstruction results on LIDC/IDRI data set. (a) true image, (b) filtered backprojection after Poisson and Gaussian noise added, (c) MSE 100% (d) MSE 50%, (e) Non-local mean filter, (f) TextureWGAN, (g) true image (zoom), (h) filtered backprojection (zoom), (i) MSE 100% (zoom), (j) MSE 50% (zoom), (k) Non-local mean filter (zoom), (l) TextureWGAN (zoom)

# Chapter 5

# Conclusion and Next Steps

In chapter two of this dissertation, we proposed a new algorithm for CT image reconstruction. The proposed method used a deep Recurrent Neural Network (RNN) to implement an iterative reconstruction (IR) algorithm. Also, we introduced a new RNN cell called Gated Momentum Unit (GMU) as a memory cell to make the RNN converge faster.

Chapter three of this dissertation proposed a new algorithm for CT Metal Artifact Reduction (CT MAR). The proposed method used a deep RNN to implement an iterative reconstruction (IR) algorithm for CT MAR. Also, we introduced a new RNN cell called Recurrent FISTA Unit (RFU) as an RNN memory cell to conduct an iterative optimization as part of the Deep Learning (DL) framework.

Chapter four of this dissertation proposed a new algorithm for inverse problems. This algorithm primarily aims at CT image reconstruction applications. The proposed method used the WGAN framework along with the MLE regularizer. This research showed that the WGAN framework effectively preserves image texture in inverse problems while keeping the PSNR and the SSIM high. The proposed method is particularly suitable for medical imaging where pixel fidelity and image texture need to be retained for clinical diagnosis.

A possible next step for these research projects is integrating the RNN GMU and RFU framework with the WGAN framework for CT image reconstruction applications. The RNN GMU and RFU can build significantly better mapping functions between a true image data set and a generated image dataset by the RNN in CT image reconstruction problems. As discussed in chapters two and three, the RNN GMU and RFU are not limited to image space processing. It takes advantage of observation data too. Besides, the WGAN framework can be integrated into the RNN frameworks as a priori cell. Currently, we use a generic CNN as the priori cell in these RNNs. However, using the WGAN framework as a priori cell can improve image texture while keeping the PSNR and the SSIM high by the RNN frameworks.

# References


[1] John Jensen Introductory Digital Image Processing: A Remote Sensing Perspective, 4th Edition *Pearson*, 2015.

[2] Jiang Hsieh et al. Computed tomography: principles, design, artifacts, and recent advances. *SPIE Press*, 2009.

[3] Ian Goodfellown et al. Deep learning. *MIT Press*, 2016.

[4] Thorsten M. Buzug. Comptued tomography: from photon statistics to modern cone-beam ct. *Springer*, 2008.

[5] Eilaghi, A. et al. CT texture features are associated with overall survival in pancreatic ductal adenocarcinoma – a quantitative analysis. *BMC Med Imaging*, 17(38), 2017.

[6] William Henry Nailon. Texture analysis methods for medical image characterisation. *Biomedical Imaging*, `https://doi.org/10.5772/8912`, IntechOpen, 2010.

[7] Yoshinori Funama et al. Improving low-contrast detectability and noise texture pattern for computed tomography using iterative reconstruction accelerated with machine learning method: a phantom study. *Academic Radiology*, 7:1-8, 2020.

[8] Bruno De Man et al. Distance-driven projection and backprojection in three dimensions. *Physics in Medicine and Biology*, 49(11):2463-2475, 2004.

[9] Jean-Baptiste Thibault et al. A three-dimensional statistical approach to improved image quality for multislice helical ct. *Medical Physics*, 34(11):4526–4544, 2007.

[10] Dufan Wu et al. Iterative low-dose ct reconstruction with priors trained by artificial neural network. *IEEE Transactions on Medical Imaging*, 36(12): 2479-2486, 2017.

[11] Dong Hye Ye et al. Deep residual learning for model-based iterative ct reconstruction using plug-and-play framework. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6668-6672, 2018.

[12] Korbinian Mechlem et al. Joint statistical iterative material image reconstruction for spectral computed tomography using a semi-empirical forward model. *IEEE Transactions on Medical Imaging*, 37(1): 68-80, 2018.

[13] Camille Chapdelaine et al. Error-splitting forward model for iterative reconstruction in x-ray computed tomography and application with gauss–markov–potts prior. *IEEE Transactions on Computational Imaging*, 5(2): 317-332, 2019.

[14] Zhiqian Chang et al. Prior-guided metal artifact reduction for iterative x-ray computed tomography. *IEEE Transactions on Medical Imaging*, 38(6):1532-1542, 2019.

[15] D. Zeng et al. A simple low-dose x-ray ct simulation from high-dose scan. *IEEE Transactions on Nuclear Science*, 62(5):2226-2233, 2015.

[16] Alex Krizhevsky et al. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 2012:1097-1105, 2012.

[17] Q. Yang et al. Low-dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss. *IEEE Transactions on Medical Imaging*, 37(6):1348-1357, 2018.

[18] Zhengchun Liu et al. TomoGAN: low-dose synchrotron x-ray tomography with generative adversarial networks. *Journal of the Optical Society of America A*, 37(3):422-434, 2020.

[19] Tobias Würfl et al. Deep learning computed tomography: learning projection-domain weights from image domain in limited angle problems. *IEEE Transactions on Medical Imaging*, 37(6), 1454-1463, 2018.

[20] Yann LeCun et al. Deep learning. *nature*, 521(7553):436, 2015.

[21] Kaiming He et al. Deep residual learning for image recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016*, 2016.

[22] Ian Goodfellow et al. Generative adversarial nets. *Advances in Neural Information Processing Systems (NIPS)*, 2014:2672-2680, 2014.

[23] Alec Radford et al. Unsupervised representation learning with deep convolutional generative adversarial networks. Paper available online from *https://arxiv.org/abs/1511.06434*, 2016.

[24] Olaf Ronneberger et al. U-Net: convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 9351, 2015.

[25] Joscha Maier et al. Real-time scatter estimation for medical ct using the deep scatter estimation: method and robustness analysis with respect to different anatomies, dose levels, tube voltages, and data truncation. *Medical Physics*, 36(1):238-249, 2019.

[26] Alain Horé et al. Image quality metrics: psnr vs. ssim. *2010 20th International Conference on Pattern Recognition*, 2366-2369, 2010.

[27] G. Wang. A perspective on deep imaging. *IEEE Access*, 4:8914-8924, 2016.

[28] G. Wang et al. Machine learning will transform radiology significantly within the next 5 years. *Medical Physics*, 44(6):2041–2044, 2017.

[29] G. Wang et al. Image reconstruction is a new frontier of machine learning. *IEEE Transactions on Medical Imaging*, 37(6):1289-1296, 2018.

[30] Eunhee Kang et al. A deep convolutional neural network using directional wavelets for low-dose X-ray ct reconstruction. *Medical Physics*, 44(10):360-375, 2017.

[31] Jonas Adler et al. Learned primal-dual reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1322–1332, 2018.

[32] Sebastian Lunz et al. Adversarial regularizers in inverse problems. *Advances in Neural Information Processing Systems (NIPS)*, 2018:8507-8516, 2018.

[33] W. Yang et al. Deep learning for single image super-resolution: a brief review. *IEEE Transactions on Multimedia*, 21(12):3106-3121, 2019.

[34] Masaki Ikuta et al. TextureWGAN: texture preserving wgan with mle regularizer for inverse problems. Paper available online from *https://arxiv.org/abs/2008.04861*, 2020.

[35] Christian Ledig et al. Photo-realistic single image super-resolution using a generative adversarial network. Paper available online from *http://arxiv.org/abs/1609.04802*, 2018.

[36] Antonin Chambolle, Thomas Pock et al. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2010.

[37] Aurélien Géron. Hands-on machine learning with scikit-learn, keras, and tensorflow. *O'Reilly*, 2017.

[38] Shuai Li et al. Independently recurrent neural network (INDRNN): building a longer and deeper rnn. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5457-5466, 2018.

[39] Hao Tang et al. On training recurrent networks with truncated backpropagation through time in speech recognition. *2018 IEEE Spoken Language Technology Workshop (SLT)*, 48-55, 2018.

[40] Debasmita Mishra et al. Deep recurrent neural network (Deep-RNN) for classification of nonlinear data. *Computational Intelligence in Pattern Recognition. Advances in Intelligent Systems and Computing*, 1120, 2020.

[41] Yoshua Bengio et al. Advances in optimizing recurrent networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8624-8628, 2013.

[42] Sepp Hochreiter et al. Long short-term memory. *Neural Computation*, 9(8), 1735-1780, 1997.

[43] Haşim Sak et al. Long short-term memory based recurrent neural metwork architectures for large vocabulary speech recognition. Paper available online from *https://arxiv.org/abs/1402.1128*, 2014.

[44] Wojciech Zaremba et al. Recurrent neural network regularization. Paper available online from *https://arxiv.org/abs/1409.2329*, 2014.

[45] Kyunghyun Cho et al. Learning phrase representations using rnn encoder–decoder for statistical machine translation. Paper available online from *https://arxiv.org/abs/1406.1078*, 2014.

[46] Sebastian Ruder. An overview of gradient descent optimization algorithms. Paper available online from *https://arxiv.org/abs/1609.04747*, 2016.

[47] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks: the official journal of the International Neural Network Society*, 12(1): 145-151, 1999.

[48] Brendan McMahan et al. Delay-tolerant algorithms for asynchronous distributed online learning. *Advances in Neural Information Processing Systems (NIPS)*, 2014:1-9, 2014.

[49] Adam: a method for stochastic optimization. Diederik P. Kingma et al. Paper available online from *https://arxiv.org/abs/1412.6980*, 2014.

[50] Mark Eisen et al. A primal-dual quasi-newton method for exact consensus optimization. *IEEE Transactions on Signal Processing*, 67(23), 5983-5997, 2019.

[51] Huiming Chen et al. A stochastic quasi-newton method for large-scale nonconvex optimization with applications. *IEEE Transactions on Neural Networks and Learning Systems*, doi: 10.1109/TNNLS.2019.2957843, 2019.

[52] Jun Zhang et al. A deep rnn for ct image reconstruction. *Proceedings of SPIE Medical Imaging 2020*, 11312:113124N, 2020.

[53] Kevin Murphy. Machine learning a probabilistic perspective. *MIT Press*, 2013.

[54] Sergey Ioffe et al. Batch normalization: accelerating deep network training by reducing internal covariate shift. Paper available online from *http://arxiv.org/abs/1502.03167*, 2015.

[55] A. Buades et al. Non-local means denoising, image processing on line *http://dx.doi.org/10.5201/ipol.2011.bcm_nlm*, 2011.

[56] J. Adler et al. Operator discretization library (ODL). Software available online from *https://github.com/odlgroup/odl*, 2017.

[57] H. Michael Gach et al. 2D & 3D shepp-logan phantom standards for mri. *2008 19th International Conference on Systems Engineering*, 521-526, 2008.

[58] Samuel Armato et al. The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on ct scans. *Medical Physics* 38(2), 2011.

[59] Yunus Saatci et al. Bayesian GAN. *Advances in Neural Information Processing Systems (NIPS)*, 2017:3622-3631, 2017

[60] Martín Arjovsky et al. Wasserstein generative adversarial networks. *International Conference on Machine Learning, ICML 2017*, 2017.

[61] Ishaan Gulrajani et al. Improved training of wasserstein GANs. *Advances in Neural Information Processing Systems (NIPS)*, 2017:5767–5777, 2017.

[62] Hyun Gi Kim et al. Quantitative analysis of the effect of iterative reconstruction Using a phantom: determining the appropriate blending percentage. *Yonsei Medical Journal*, 56(1):253-261, 2015.

[63] R. M. Haralick et al. Textural Features for Image Classification *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610-621, 1973.

[64] The MathWorks, Inc. Calculate statistical measures of texture. `https://www.mathworks.com/help/images/texture-analysis.html`, Accessed March 21, 2020.

[65] The MathWorks, Inc. Properties of gray-level co-occurrence matrix. `https://www.mathworks.com/help/images/ref/graycoprops.html`, Accessed March 21, 2020.

[66] M.H. Bharati et al. Image texture analysis: methods and comparisons. *Chemometrics and Intelligent Laboratory Systems*, 72(1):57-71, 2004.

[67] K. Simonyan et al. Very deep convolutional networks for large-scale image recognition. Paper available online from *https://arxiv.org/abs/1409.1556*, 2014.

[68] Google LLC. tf.keras.applications.VGG19. `https://www.tensorflow.org/api_docs/python/tf/keras/applications/VGG19`, Accessed March 21, 2020.

[69] Yann LeCun et al. Gradient-based learning applied to document recognition *Proceedings of the IEEE*, 86(11):2278-2324, 1998.

[70] Pablo Arbelaez et al. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898-916, 2011.

[71] R. Cipolla et al. Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. *IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018*, 2018.

[72] G. Wang et al. Image reconstruction is a new frontier of machine learning. *IEEE Transactions on Medical Imaging*, 37(6):1289-1296, 2018.

[73] Bouman and Sauer et al. A unified approach to statistical tomography using coordinate descent optimization. *IEEE Transactions on Image Processing*, 5(3):480-492, 1996.

[74] Yu et al. Deep Sinogram Completion With Image Prior for Metal Artifact Reduction in CT Images. *IEEE Transactions on Medical Imaging*, 40(1):228-238, 2021

[75] Kalender et al. Reduction of CT artifacts caused by metallic implants. *Radiology*, 164(2):576-577, 1987.

[76] Meyer et al. Normalized metal artifact reduction (NMAR) in computed tomography. *Medical Physics*, 37(10):5482–5493, 2010.

[77] Thibault et al. A recursive filter for noise reduction in statistical iterative tomographic imaging. *SPIE Electronic Imaging*, 2006.

[78] Zhang et al. Metal artifact reduction in X-ray computed tomography (CT) by constrained optimization. *Medical Physics*, 38(2):701–711, 2011.

[79] Goodfellown et al. Deep learning. *MIT Press*, 2016.

[80] Chang et al. Prior-Guided Metal Artifact Reduction for Iterative X-Ray Computed Tomography. *IEEE Transactions on Medical Imaging*, 38(6):1532-1542, 2019.

[81] Zhang et al. Convolutional neural network based metal artifact reduction in X-ray computed tomography. *IEEE Transactions on Medical Imaging*, 37(6):1370–1381, 2018.

[82] Lin et al. DuDoNet: Dual domain network for CT metal artifact reduction. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019

[83] Park et al. CT sinogram–consistency learning for metal induced beam hardening correction. *Medical Physics*, 45(12):5376-5384, 2018

[84] Ronneberger et al. U-Net: convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 9351, 2015.

[85] Gjesteby et al. A dual-stream deep convolutional network for reducing metal streak artifacts in CT images. *Physics in Medicine and Biology*, 64(23), 2019.

[86] Wang et al. Conditional generative adversarial networks for metal artifact reduction in ct images of the ear. *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 3-11, 2018.

[87] Isola et al. Image-to-Image translation with conditional adversarial networks *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1125-1134, 2017.

[88] Peng et al. A Cross-Domain Metal Trace Restoring Network for Reducing X-Ray CT Metal Artifacts. *IEEE Transactions on Medical Imaging*, 39(12), 2020.

[89] Huang et al. Metal artifact reduction on cervical CT images by deep residual learning. *BioMedical Engineering OnLine*, 17(1): 175-190, 2018.

[90] Ghani et al. Fast Enhanced CT Metal Artifact Reduction Using Data Domain Deep Learning. *IEEE Transactions on Computational Imaging*, 6:181-193, 2020.

[91] Liao et al. ADN: Artifact disentan- glement network for unsupervised metal artifact reduction. *IEEE Transactions on Medical Imaging*, 39(3), 634-643, 2020.

[92] Peng et al. An irregular metal trace inpainting network for X-ray CT metal artifact reduction. *Medical Physics*, 47(9), 2020.

[93] Boyd et al. Convex Optimization *Cambridge University Press*, 2004.

[94] Beck et al. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *Society for Industrial and Applied Mathematics*, 2(1), 183-202, 2009.

[95] Tibshirani et al. Proximal gradient descent and acceleration. Lecture Notes available online from

https://www.stat.cmu.edu/~ryantibs/convexopt-S15/scribes/08-prox-grad-scribed.pdf, 2010.

[96] Meinhardt et al. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. *International Conference on Computer Vision (ICCV)*, 2017.

[97] Sakamoto et al. Bayesian Segmentation of Hip and Thigh Muscles in Metal Artifact-Contaminated CT Using Convolutional Neural Network-Enhanced Normalized Metal Artifact Reduction. *Journal of Signal Processing Systems*, 92:335–344, 2020

[98] Ikuta et al. A Deep Recurrent Neural Network with Gated Momentum Unit for CT Image Reconstruction. Paper available online from https://doi.org/10.36227/techrxiv.15066138, 2021.

[99] Yan et al. Deep lesion graphs in the wild: Relationship learning and organization of significant radiology image findings in a diverse large- scale lesion database. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018

# Curriculum Vitae

Masaki Ikuta

Place of birth: Kochi, Japan

Education:

- B.E., Hosei University, Japan, March 1999
  Major: Industrial and Systems Engineering
- M.E., Hosei University, Japan, March 2001
  Major: Systems Engineering and Computer Science

Professional Career:

- Software Engineer, Computed Tomography, GE Healthcare, Hino, Tokyo, Japan, April 2001 to March 2006
- Senior Software Engineer, Computed Tomography Image Reconstruction, GE Healthcare, Waukesha, Wisconsin, U.S.A., March 2006 to Sep 2021
- Staff Data Scientist, Central Data Science, GE Healthcare AI, San Ramon, California, U.S.A., October 2021 to the present

Dissertation Title: A DEEP RECURRENT NEURAL NETWORK WITH ITERATIVE OPTIMIZATION FOR INVERSE IMAGE PROCESSING APPLICATIONS

Publications:

- Masaki Ikuta et al., A Deep Recurrent Neural Network with Primal-dual Optimization for CT Metal Artifact Reduction (Accepted), SPIE Medical Imaging Conference 2022.

- Masaki Ikuta et al., A Deep Recurrent Neural Network with Gated Momentum Unit for CT Image Reconstruction, Paper also available online from *https://doi.org/10.36227/techrxiv.15066138.v1*, 2021.
- Masaki Ikuta et al., TextureWGAN: Texture Preserving WGAN with MLE Regularizer for Inverse Problems, SPIE Medical Imaging Conference 2021, Paper also available online from *https://doi.org/10.1117/12.2580434*, 2021.
- Masaki Kagawa Ikuta et al., A Phase Unwrapping Method Using Direct Residue Removing, Proceedings of IEEE International Geoscience and Remote Sensing Symposium, 2000.
- Masaki Kagawa Ikuta et al., An automated method to estimate the baseline parameters for deriving the accurate digital elevation model, Proceedings of SPIE European Remote Sensing Symposium (EUROPTO 2000), Vol. 4173, 2000.
- Masaki Kagawa Ikuta et al., A fast method for unwrapping InSAR raw interferogram, Proceedings of SPIE European Remote Sensing Symposium (EUROPTO 1999), Vol. 3871 pp.63-70, 1999.