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

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

28-07-2021 / 29-07-2021

CITATION

Ikuta, Masaki (2021): A Deep Recurrent Neural Network with Gated Momentum Unit for CT Image Reconstruction. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.15066138.v1>

DOI

[10.36227/techrxiv.15066138.v1](https://doi.org/10.36227/techrxiv.15066138.v1)

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

Masaki Ikuta and Jun Zhang

Abstract—Many algorithms and methods have been proposed for Computed Tomography (CT) image reconstruction, particularly with the recent surge of interest in machine learning and deep learning methods. The majority of recently proposed methods are, however, limited to the image domain processing where deep learning is used to learn the mapping from a noisy image data set to a true image data set. While deep learning-based methods can produce higher quality images than conventional model-based post-processing algorithms, these methods have limitations. Deep learning-based methods used in the image domain are not sufficient for compensating for lost information during a forward and a backward projection in CT image reconstruction especially with a presence of high noise. In this paper, we propose a new Recurrent Neural Network (RNN) architecture for CT image reconstruction. We propose the Gated Momentum Unit (GMU) that has been extended from the Gated Recurrent Unit (GRU) but it is specifically designed for image processing inverse problems. This new RNN cell performs an iterative optimization with an accelerated convergence. The GMU has a few gates to regulate information flow where the gates decide to keep important long-term information and discard insignificant short-term detail. Besides, the GMU has a likelihood term and a prior term analogous to the Iterative Reconstruction (IR). This helps ensure estimated images are consistent with observation data while the prior term makes sure the likelihood term does not overfit each individual observation data. We conducted a synthetic image study along with a real CT image study to demonstrate this proposed method achieved the highest level of Peak Signal to Noise Ratio (PSNR) and Structure Similarity (SSIM). Also, we showed this algorithm converged faster than other well-known methods.

Index Terms—Computed Tomography, image reconstruction, Recurrent Neural Network, Gated Recurrent Unit, momentum, Backpropagation Through Time

I. INTRODUCTION

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's body [1], [2]. For example, CT has become a useful screening tool to detect a potential tumor or a lesion in the 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 [3], [4], [5]. In CT, a narrow beam of X-ray is exposed to a patient and an X-ray tube and a CT detector are quickly rotated around the patient's body. This scanning process produces X-ray projection data that are often called view data [1], [6]. 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 patient anatomies while other beams can pass through the air region and these intensities are not weakened. By applying a negative log step per the Beer-Lambert law [1], view data are converted into sinogram data that show X-ray attenuation [1], [2]. The sinogram data are processed by many algorithms to produce a cross-sectional image or a slice of a patient's body. These signal and image processing steps are called CT image reconstruction. CT image reconstruction is considered one of the inverse problems. There are many processing steps to process scanning data in CT image reconstruction. The key step among them to create CT images out of sinogram data is called backprojection where scanning data in the sinogram domain are converted or projected into two or three-dimensional pixel data in the image domain. The sinogram domain and the image domain are considered two different domains. One is in the scanning coordinate (the equipment coordinate) and the other is in the patient coordinate [1]. One of the most frequently used backprojection steps is the filtered backprojection or the FBP in short. The FBP is popular for commercial CT systems because of the simplicity and 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.

Deep learning-based post-processing methods are a straightforward extension of computer vision problems [7] and the research community has successfully demonstrated that these methods can greatly improve results for inverse problems such as image de-noising, super-resolution and CT image reconstruction [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18]. Among many variations of neural network structures, Convolutional Neural Network (CNN) has been proven effective in inverse problems [11], [12], [13], [14], [15], [16], [17], [18]. CNN usually creates superior results compared with conventional model-based post-processing algorithms [19]. There are also many types of CNN structures, but the most frequently used CNN type for inverse problems is the U-Net [20]. The U-Net has down-sampling and up-sampling operations along with skip connections within the network so that it can learn overall structures of images as well as minor details localized in some areas of images [12], [21].

Masaki Ikuta and Jun Zhang are with the Department of Electrical Engineering and Computer Science, University of Wisconsin - Milwaukee, Milwaukee, WI 53211, USA. e-mail: mikuta, junzhang@uwm.edu

Masaki Ikuta is also with the Computed Tomography Engineering, Image Reconstruction at GE Healthcare, Waukesha, WI 53188, USA.

Manuscript received Month Date, Year; revised Month Date, Year.

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

Skip connections provide the 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) [22]. 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 [8], [9], [10]. Many research projects have been conducted with CNN and the U-Net; however, the majority of them has been limited in image space processing where CNN was used to learn the mapping from a noisy image data set to a true image data set in the image domain [11], [12], [21], [23]. A problem in this approach for inverse problems is that conventional inverse operators such as bicubic interpolation in super-resolution and the filtered backprojection in CT image reconstruction are used to bring data from one domain to the other and important information can be lost within these operations [24], [25], [26], [27]. This does not seem to be a major obstacle in super-resolution [26], [28]. But there are many non-linear operations within CT reconstruction and it is not clear how much CNN-based methods can compensate for the lost information within the FBP operation as well as in non-linear operations before and after the FBP. If scanning data contain high noise or even a metal object, the performance of the FBP operator deteriorates significantly. As a result, CNN's performance degrades greatly.

Iterative Reconstruction (IR) as a stochastic approach has attracted much attention for many years [29], [30], [31], [32], [33] to handle the stochastic nature of CT image reconstruction problems. The essence of the IR is its Bayesian formulation where the method reconstructs cross-sectional images by iteratively maximizing the posterior probability distribution of images. The Bayesian formulation decomposes the posterior probability distribution into a likelihood term and a prior term and the IR iteratively maximizes both terms to get an optimal result. Thus, the IR solves a maximization problem. The likelihood term is used to make sure estimated images are consistent with the corresponding observation data (sinogram). The prior term helps the likelihood term not overfit each individual observation data. This method is particularly useful in low dose CT because the reconstruction process becomes more stochastic due to the low dosage data acquisition process [5], [30], [34]. This method is also effective in reducing metal artifacts of reconstructed images where projection data are partially blocked by metals in a patient's body [19]. Although the IR has become commercially available in clinical CT systems [5], this method is computationally expensive [1], [2]. In the FBP, images are typically reconstructed and available for clinicians for review as fast as several seconds. In the IR, image reconstruction takes at least 20 to 30 minutes depending upon scanning and reconstruction techniques. Also, the IR is often described as a linear recursive equation if we think output images are sequential data. However, the linear nature of the optimization may not be accurate enough nor fast enough in processing CT data. Furthermore, the IR needs a prior term that accounts for prior information of reconstructed images. Total Variation (TV) and Markov Random Fields (MRFs) [1], [2], [29], are often used as the prior 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 reconstructed images.

A. Summary of Contributions

In this paper, we propose a novel neural network-based approach for the IR that uses sinogram data. This approach consists of two parts. First, we use the momentum method [35] to replace the gradient descent in the IR. This can accelerate the optimization process and get better solutions for iterative optimization problems. Second, we implement the momentum method as an RNN cell. We still use ADAM (ADaptive Moment estimation) [36] to train the neural network. However, the momentum method is implemented as part of the neural network architecture to accelerate the training process. In summary, the main contributions of this work are:

- 1) We created the Gated Momentum Unit (GMU) as an RNN cell. The GMU is extended from the GRU but it is intended to be used for inverse problems such as CT image reconstruction.
- 2) We will show the GMU improved the PSNR and the SSIM numbers on output images. There are two versions of the RNN GMU. One is called RNN GMU image-domain, which only works in the image domain. The other is called RNN GMU sinogram/image-domain, which takes advantage of sinogram data in addition to image data. We will show the former produces better PSNR and SSIM than other post-processing based deep learning methods. We will also show that the latter produces better results than the state of the art CT image reconstruction methods such as Learned PDHG and Learned Primal-dual Reconstruction [24].
- 3) We also created Backpropagation Through Iteration (BPTI), which is more or less Backpropagation Through Time (BPTT) but it is used in iterative optimization. We will show how the BPTI improves PSNR and SSIM.
- 4) We conducted qualitative and quantitative analyses in a synthetic image study and a real CT image study. We will show that our proposed methods are effective in both studies.

B. Related Work and Topics

In this subsection, we discuss related work and relevant topics to our research including Primal-Dual Hybrid Gradient (PDHG) algorithm, Recurrent Neural Network (RNN), and optimization algorithms for neural network.

Primal-Dual Hybrid Gradient (PDHG) algorithm introduced by Chambolle and Pock in 2011 [37], [38] is distinct from the conventional iterative reconstruction. The PDHG transforms the iterative reconstruction problem into a primal-dual problem mathematically, and the primal problem is minimized while the dual problem is maximized concurrently. The PDHG uses the duality of convex optimization [39] because a dual problem of a primal problem is often simpler to solve. There were many research projects in the past to solve dual problems instead of trying to solve primal problems in iterative reconstruction [1], [2]. However, the PDHG is distinct in the way that it tries to solve a primal problem and the corresponding

dual problem at the same time. The PDHG uses Accelerated Proximal Gradient Method [38], [40] but each step tries to optimize both the primal and the dual variable. This is why the PDHG was named as a "Hybrid Gradient" algorithm. The PDHG can also handle a non-differentiable surface by using a proximal operator [38]. While the idea of the PDHG appears to be promising, there are a few problems with this method. The first problem is that multiple gradient descent step sizes need to be determined because of the nature of multiple variable optimization. The conventional GD algorithm is often criticized because it is not trivial to determine the step size. Therefore, determining multiple step sizes is even more difficult. There is no easy solution to this problem. The second problem is that the PDHG uses a model-based prior such as Total Variation (TV) prior and it is too simple to represent the truly complex nature of CT images as we discussed in the earlier part of this paper. Adler et al. proposed two algorithms in their paper [24]. One was "Learned PDHG" algorithm and the other was "Learned Primal-dual Reconstruction." The former replaced the proximal operators with CNN blocks. Since it used CNN, the weakness of the simple prior problem was eliminated as a result. However, it still used the multiple step sizes. The latter eliminated the multiple step sizes by adding more dimensions into the primal (images) and the dual variable (sinogram). Also, it used a concatenation with the primal and dual variables. While the Learned Primal-dual Reconstruction produced significantly higher PSNR and SSIM than 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.

Recurrent Neural Network (RNN) is an alternative choice of neural network for inverse problems. In CNN, the input data need to be fixed-size and it cannot handle arbitrary lengths of input data. On the other hand, RNN can handle time-series data and sequential data such as stock price predictions, autonomous driving systems, and natural language processing tasks [15], [41]. In 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 a deep RNN [15], [41]. The backpropagation step in the RNN is distinct from the CNN. In the CNN, the output of the final layer of the network is used to compute an error that 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 an error and the error in each time step is back-propagated to the network. This technique is called "Backpropagation Through Time" or BPTT [42], [43], [44], [45]. 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. To reduce these problems, the Long Short-Term Memory (LSTM) was proposed in 1997 [46]. In this approach, the memory cell is replaced with an LSTM cell that 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 that happened many time steps ago. The LSTM cell has a forget gate that discards minor short-term details. Thus, the RNN can preserve only important long-term information [47], [48]. Gated Recurrent Unit (GRU) was proposed by Cho et al. in 2014 [49] 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 as its hidden states. 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 fewer neurons. Thus, the GRU is generally simpler and faster than the LSTM [49]. Almost all state of the art results based on the RNN were archived with these two types of RNN cells [41].

Optimization algorithms are the backbone of neural network and machine learning. The gradient descent (GD) algorithm is the simplest and the most frequently used optimization algorithm in machine learning [15]. Stochastic Gradient Descent (SGD) and Mini-batch Gradient Descent (Mini-batch GD) are under the same umbrella but they use a different number of training data in each optimization [50]. 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 [15]. However, these methods often have 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 [35]. In such scenarios, these methods oscillate across the slopes of the ravine, which only makes hesitant movements toward the local optimum [35]. Momentum [35] 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 downhill, getting faster and faster along the way [35]. 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) [51] and ADAM (ADaptive Moment estimation) [36] are extensions of the momentum method and they are now widely used in the deep learning community [15].

C. Outline of the Paper

In the following sections, we show how the weaknesses of previous methods are addressed by our deep RNN and the GMU. The organization of this paper is that we discuss our methods of the RNN and the GMU in detail in the next section. In the third section, we show qualitative and quantitative evaluation results. In the fourth section, we discuss the conclusion of this paper.

II. METHODS

A. Iterative Reconstruction and the Optimization

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

In CT image reconstruction problems, reconstructed images x and sinogram y are stochastic processes and a Bayesian or MAP (Maximum A Posteriori) approach has been frequently used to model this problem as follows [29].

$$\begin{aligned} x^* &= \arg \max_x \{ \log p(x|y) \} \\ &= \arg \max_x \{ \log p(y|x) + \log p(x) \} \\ &= \arg \min_x \{ J(y, x) \}, \\ \text{s.t. } J(y, x) &= -\log p(y|x) - \log p(x) = k(y, x) + l(x), \end{aligned} \quad (1)$$

where $x \in X$ and $y \in Y$ are sampled from image space X and sinogram space Y , which are usually Hilbert spaces. x^* is an optimized image. $J(y, x)$ is the objective function to minimize to obtain x^* . In the objective function, the posterior probability distribution $p(x|y)$ is decomposed into the likelihood term $p(y|x)$ and the prior term $p(x)$ by Bayesian formulation [29]. Thus, the objective function includes both terms. Here, the objective function is also denoted as $J(y, x) = k(y, x) + l(x)$ for a sake of simplicity. The X-ray scan acquisition process, including CT scanning, follows the Poisson process and the likelihood term is often approximated by a Gaussian model [1], [2] as follows.

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

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 contrast 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 detector cell location i [29]. The d_i shows the credibility of the projection measurement [29]. If a particular measurement y_i is photon-starved by some metal object in the patient's body, the corresponding d_i becomes reduced and the contribution of the particular measurement to the image estimate would be lower [29].

This likelihood term described in equation (2) represents how close the estimates of reconstructed images are to the corresponding input sinogram. While the likelihood term is relatively straightforward to calculate given the acquisition geometry shown above, the prior term is more complex to calculate. The prior term incorporates various characteristics of the ideal reconstructed images to help produce good results while preventing the likelihood term from overfitting the training data. In the IR, TV and MRF priors are often used [29]. Recently, neural network-based priors were also proposed [25].

Equation (1) can be solved iteratively and one way to do this is through the gradient descent as follows [52].

$$x^{(i)} = x^{(i-1)} - \epsilon \nabla_x \{ J(y, x) \} \big|_{x=x^{(i-1)}}, \quad (3)$$

where $\epsilon > 0$ is a step size of the gradient $\nabla_x \{ J(\cdot) \}$ and i is the iteration index. Here, we assume $J(y, x)$ is continuous and differentiable everywhere. It is reasonable to assume the likelihood term $k(y, x)$ is differentiable everywhere because it is often modeled as a quadratic form. However, it is more accurate to assume that the prior term $l(x)$ is continuous but not necessarily differentiable everywhere. A good example is a TV prior with L1-norm. One way to handle such a function $l(x)$ is to use a proximal operator [40].

$$\text{prox}_{l,t}(x) = \arg \min_z \left\{ \frac{1}{2t} \|z - x\|_2^2 + l(z) \right\}, \quad (4)$$

where the first term is a regularization term to encourage x and z are close to each other. t is a regularization parameter to control the term. By using equation (4), equation (3) can be rewritten as [40]

$$\begin{aligned} x^{(i)} &= x^{(i-1)} - t_i K_{t_i}(x^{(i-1)}), \\ \text{s.t. } K_t(x) &= \frac{x - \text{prox}_{l,t}(x - t \nabla_x k(y, x))}{t}, \end{aligned} \quad (5)$$

where t_i is a step size of $K_{t_i}(\cdot)$. A gradient descent algorithm with the proximal operator is often denoted as $x^{(i)} = \text{prox}_{l,t_i}(x^{(i-1)} - t_i \nabla_x k(y, x) \big|_{x=x^{(i-1)}})$. But equation (5) is a more familiar form and we will use this form to explain the following sections.

B. Iterative Optimization with the RNN

If we assume $x^{(i)}$ is a sequence, equation (5) is a recursive equation. There are many ways to implement equation (5) as a model-driven method [1], [2]. However, data-driven methods have attracted much attention recently and they have been proven effective [11], [12], [13], [14], [15] to outperform many model-driven methods. In this research, we chose to use a Recurrent Neural Network (RNN) to implement it. The RNN is a suitable choice due to the recursive nature of the neural network. Since the RNN is highly non-linear, equation (5) can be implemented as a non-linear recursive equation that should in principle work better than a linear recursive equation.

If we use the RNN for iterative optimization such as the IR in Computed Tomography, the RNN will have two inputs. One is an observation (sinogram) and the other is the initial estimate of an image. The RNN also has a hidden state to carry from one iteration to the next. If we think the estimate of an image and the hidden state are sequences, then the image x and the hidden state h can be described as sequences such that $x = \{x^{(0)}, x^{(1)}, \dots, x^{(I)}\}$ and $h = \{h^{(0)}, h^{(1)}, \dots, h^{(I)}\}$, where I is the total number of iterations of the RNN. Also, each RNN cell can be expressed as a parameterized vector-valued function f_θ as follows.

$$\begin{aligned} (x^{(i)}, h^{(i)}) &= f_\theta(x^{(i-1)}, h^{(i-1)}; y), \\ (x^{(I)}, h^{(I)}) &= f_\theta \circ f_\theta \circ \dots \circ f_\theta(x^{(0)}, h^{(0)}; y), \end{aligned} \quad (6)$$

Fig. 1: The overall network structure of the RNN GMU for CT image reconstruction. The RNN cell is shown in Figure 2 in detail. The same network weights in the RNN cell are repeatedly used in the RNN. The FBP is the filtered backprojection operator, y is an input sinogram, and $x^{(i)}$ is the current estimate of images at iteration i . $m^{(i)}$ is the momentum hidden state at iteration i that is passed from the current iteration to the next.

where y is an observation and $x^{(0)}, h^{(0)}$ 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 $x^{(I)}, h^{(I)}$ are the final results of the RNN. $\theta \in \Theta$ is some parameter space to define the behavior of the function f_θ .

C. The Gated Momentum Unit (GMU)

Now, we would like to discuss the internal design of the RNN. Since we want to use the RNN as an iterative optimizer, the usual RNN cells would not work for the purpose. We need to come up with a new RNN cell that is specifically designed for iterative optimization that implements equation (5) as an RNN cell. That would be an RNN cell to compute $t_i K_{t_i}(x^{(i-1)})$ with a skip connection. The skip connection would take care of the subtraction of equation (5). This means that each RNN cell will be trained to compute the gradient term $t_i K_{t_i}(x^{(i-1)})$ of equation (5) at each step, and then we subtract the gradient from the previous estimate of image $x^{(i-1)}$ at each iteration. However, this gradient descent approach would be too slow to converge and we would like to accelerate the convergence. Many techniques have been proposed to provide a learning acceleration. A relatively simple and successful technique is the momentum method [15], [35] which can be expressed as follows.

$$\begin{aligned} x^{(i)} &= x^{(i-1)} - m^{(i)}, \\ m^{(i)} &= \eta m^{(i-1)} + t_i K_{t_i}(x^{(i-1)}), \end{aligned} \quad (7)$$

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 η is usually set to 0.9 or a similar value [50]. If we compare equation (5) and (7), the term $\eta m^{(i-1)}$ is new and this is a weighted past momentum that is used for an acceleration and avoids an oscillation.

The momentum method certainly helps to 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 "forgets" the past momentum and weighs on recent momentum values. Also, when we use the momentum method as a design of the RNN, a vanishing gradient problem may occur especially if the number of iterations is large. In this case, later iterations of

the RNN would learn more than the earlier iterations. To solve this problem, we introduce the Gated Momentum Unit (GMU) for the RNN. The GMU extends from the Gated Recurrent Unit (GRU) to implement the momentum method.

Figure 1 shows the overall design of the deep RNN. The input to the RNN cell is sinogram y , the initial estimate of image $x^{(0)}$, and the initial momentum hidden state $m^{(0)}$. An RNN hidden state is usually denoted as $h^{(i)}$ in many RNN applications. But since our hidden state is supposed to carry momentum information, we denote it as $m^{(i)}$. The FBP is used to create $x^{(0)}$ and $m^{(0)}$ is set to zeros in this research. I is the total number of iterations of the deep RNN. $x^{(I)}$ is the final estimate of the image and $m^{(I)}$ is the final hidden state.

In summary, equation (6) can be re-written as the following.

$$\begin{aligned} (x^{(i)}, m^{(i)}) &= \text{GMU}(x^{(i-1)}, m^{(i-1)}; y), \\ (x^{(I)}, m^{(I)}) &= \text{GMU} \circ \text{GMU} \circ \dots \text{GMU}(x^{(0)}, m^{(0)}; y), \end{aligned} \quad (8)$$

where the GMU implements a parametrized vector-valued function f_θ in equation (6).

Figure 2 shows the network structure of each RNN cell. In the figure, (a) is the GRU [49] and (b) is the GMU. The GRU is only shown there to highlight the differences between two RNN cells. The hidden state in the GRU is named as $h^{(i)}$ following the convention of the original paper [49]. Other than this hidden state naming difference between $h^{(i)}$ and $m^{(i)}$, there are actually four differences between these two RNN cells. The GRU has three fully connected layers. We replaced them with three different CNN blocks with concatenations because the GMU is used for image processing applications. Also, the tanh activation function in the GRU has been removed from the GMU because the GMU is intended to perform a regression task. Furthermore, a skip connection is added in the GMU to perform the momentum optimization. A skip connection is usually an addition of two channels. However, we use a subtraction of two channels for the skip connection following the momentum equation (7) in this research. Finally, an optional CNN block (CNN4) along with the forward (FP) and backward projection (FBP) has been added in the GMU to take advantage of sinogram data y . If CNN4 is not enabled, the GMU will be optimized only in the image-domain. When the CNN4 is enabled, the GMU needs to be re-trained from scratch. In this paper, we call the GMU "RNN GMU (sinogram/image-domain)" if we enable the optional CNN block. If not, we call it "RNN GMU (image-domain)" that is trained only in image-domain. In summary, RNN GMU (sinogram/image-domain) works as an IR algorithm and RNN GMU (image-domain) works as a post-processing de-noising algorithm.

As the inventors of the GRU stated in their paper [49], the gates are important to get a meaningful result from the RNN cell. The forget gate and the input gate work together and they are in fact a convex combination such that $\text{ForgetGate} + \text{InputGate} = 1$. If the cell decides to take short-term information, it discards long-term information first by forgetting the saved long-term memory before overriding the particular memory location by the short-term information [49]. The reset gate decides if the long-term information needs

(a) Gated Recurrent Unit (GRU)

(b) Gated Momentum Unit (GMU)

Fig. 2: The network structures of two different RNN cells. (a) the Gated Recurrent Unit (GRU) Cell [49], (b) the Gated Momentum Unit (GMU) Cell. There are four differences between the GRU and the GMU. The GRU has three fully connected layers. We replaced them with three different CNN blocks with concatenations. Also, the tanh activation function in the GRU has been removed from the GMU because the GMU is intended to perform a regression task. Furthermore, a skip connection is added in the GMU to perform the momentum optimization [35]. Finally, an optional CNN block (CNN4) along with the forward (FP) and backward projection (BP) has been added in the GMU to take advantage of sinogram data y . If CNN4 is not enabled, the GMU will be optimized only in the image domain. The blue notations in (b) are added to explain the connection between the momentum (equation (7)) and the GMU optimization (equation (9)).

to be used to calculate the output information. In the GMU, we use all three gates from the GRU.

The following equation describes the GMU mathematically. The main difference between equation (7) and equation (9) is that the scaling factor η is replaced by the forget and the input gate.

$$\begin{aligned} x^{(i)} &= x^{(i-1)} - m^{(i)}, \\ m^{(i)} &= \text{ForgetGate} \otimes m^{(i-1)} \\ &\quad + \text{InputGate} \otimes t_i K_{t_i}(x^{(i-1)}) \end{aligned} \quad (9)$$

where $\otimes$ is the element-wise multiplication.

The "CNN3" in the GMU is a CNN block and it is intended to calculate a derivative or a step of the objective function that is $t_i K_{t_i}(x^{(i-1)})$ in equation (5). The optional CNN block "CNN4" along with the FP and the FBP can be enabled if we want to use sinogram data y . Here is the mathematical description of how to calculate the gradient at each iteration i .

$$\begin{aligned} g^{(i)} &= t_i K_{t_i}(x^{(i-1)}) \\ &\cong C_3(\text{ResetGate} \otimes m^{(i-1)}, l^{(i)}, x^{(i-1)}), \quad (10) \\ \text{s.t. } l^{(i)} &= A^+(C_4(y, Ax^{(i-1)})), \end{aligned}$$

where A is the forward projection and A^+ is the backprojection. $C_3(\cdot)$ and $C_4(\cdot)$ are "CNN3" and "CNN4" in the RNN GMU, respectively. $r^{(i)}$ is the output of "CNN1" after the sigmoid in the GMU. If the image matrix size is small such as the one in the synthetic image study, it is beneficial to have the likelihood cell in the image-domain. In this case, the likelihood term can be described as $l^{(i)} = C_4(A^+(y - Ax^{(i-1)}))$.

There are several advantages to the GMU over conventional deep RNN-based methods. First of all, the GRU has been used by many researchers and it has been proven effective. Therefore, extending from the GRU is a wise choice to implement the iterative optimization. Second, the GMU uses the momentum method inside to accelerate the optimization, so that the GMU can learn an optimization problem with a limited number of epochs. Thus, it converges faster. Besides, η in the momentum method has been replaced by the forget gate and the input gate. This scaling factor η is too simple for the truly complex nature of optimization problems such as CT image reconstruction or medical image reconstruction in general. These two gates are, in fact, non-linear and learned gates. They regulate the flows of information and decide if short-term information needs to be discarded or kept. If it decides to keep the current information, the forget gate gets opened and the input gate gets closed. These two gates have been also proven effective to alleviate a vanishing and an exploding gradient problem in RNN. Rather than using a scaling factor like 0.9 in the momentum method, we want to use a deep learning-based RNN cell to determine how much we should use long-term or short-term information to calculate the momentum at each iteration. Third, the gradient generator "CNN3" works as the prior term. Since we use a CNN block as the prior term, this is a learned prior term and it should work better than the TV prior and the MRF prior. We also considered placing an additional CNN block in the GMU to have an explicit prior term. However, CNN3 is already a CNN block and we decided not to have it explicitly to minimize the GPU memory usage. Finally, we added an optional CNN block as "CNN4" along

with the FP and the FBP. This likelihood cell helps the current estimate of images to be consistent with sinogram data y . If we disable this cell, the GMU works only in the image domain and the pixel fidelity of the resulting images would be deteriorated because lost information during backprojection is not compensated well. The likelihood cell is important and it should be always used for inverse problems, especially when the backward operator is complex, such as filtered backprojection in CT. We evaluated the image domain version of the GMU along with other methods just for comparison's sake in the following section.

A pseudo-code of the GRU and the GMU are shown in Algorithm 1 and Algorithm 2, respectively. The GRU algorithm is shown here to highlight the differences between these two RNN cells.

Algorithm 1 Gated Recurrent Unit (GRU) [49]

Require: The initial input data $x^{(0)}$, the initial hidden state $h^{(0)}$, the sigmoid function σ , hyperbolic tangent $\tanh$, element-wise multiplication $\otimes$, the RNN network weights $W_r, W_z, W_g, b_r, b_z, b_g$, the RNN network parameter θ , the number of iteration I, the number of mini-batch B.

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

$$r^{(i)} \leftarrow \sigma(W_r[h^{(i-1)}, x^{(i-1)}] + b_r)$$

$$z^{(i)} \leftarrow \sigma(W_z[h^{(i-1)}, x^{(i-1)}] + b_z)$$

$$g^{(i)} \leftarrow \tanh(W_g[h^{(i-1)} \otimes r^{(i)}, x^{(i-1)}] + b_g)$$

$$x^{(i)}, h^{(i)} \leftarrow h^{(i-1)} \otimes (1 - z^{(i)}) + g^{(i)} \otimes z^{(i)}$$

end for

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

Algorithm 2 Gated Momentum Unit (GMU)

Require: Input sinogram y , the initial estimate of images $x^{(0)}$, the forward projector A , pseudo inverse A^+ , the initial hidden state $m^{(0)}$, the sigmoid function σ , CNN blocks C_1, C_2, C_3, C_4 , element-wise multiplication $\otimes$, the RNN network parameter θ , the number of iteration I, the number of mini-batch B, a concatenation "[,]".

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

$$r^{(i)} \leftarrow \sigma(C_1([m^{(i-1)}, x^{(i-1)}]))$$

$$z^{(i)} \leftarrow \sigma(C_2([m^{(i-1)}, x^{(i-1)}]))$$

$$l^{(i)} \leftarrow A^+(C_4([y, Ax^{(i-1)}]))$$

$$g^{(i)} \leftarrow C_3([m^{(i-1)} \otimes r^{(i)}, l^{(i)}, x^{(i-1)}])$$

$$m^{(i)} \leftarrow m^{(i-1)} \otimes (1 - z^{(i)}) + g^{(i)} \otimes z^{(i)}$$

$$x^{(i)} \leftarrow x^{(i-1)} - m^{(i)}$$

end for

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

Table I shows the network configurations of each CNN block in the GMU. We use PReLU as our activation function in this research as it was done by Adler [24]. ReLU has been one of the keys to the recent successes in deep learning [15] and PReLU is one of the ReLU families. 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.

TABLE I: The configuration of four CNN blocks in each RNN cell. From the left to the 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).

Name	Inputs	Layers	Chans	Activation	Outputs
CNN1	2	5	32	PReLU	1
CNN2	2	5	32	PReLU	1
CNN3	3	5	64	PReLU	1
CNN4	2	5	32	PReLU	1

D. Backpropagation Through Iteration (BPTI)

A CNN does not have a sequence of outputs. Therefore, the backpropagations in a CNN and a usual RNN are different. In CNN, only the final output is used to compute the error that 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 an RNN is called Backpropagation Through Time (BPTT) as explained in the introduction section [42], [43], [44]. Because our RNN works over iteration rather than time, we call this framework Backpropagation Through Iteration (BPTI) to avoid confusion in this paper. The following equations show how to optimize an RNN with the BPTI. 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

$$p(x|y) = \prod_{i=1}^I p(x^{(i)} | x^{(i-1)}, \dots, x^{(0)}, y; \theta). \quad (11)$$

Here is how to optimize the parameter θ in an RNN by the BPTI.

$$\theta_{MAP} = \arg \max_{\theta} \{ \log p(x|y) \},$$

$$= \arg \min_{\theta} \left\{ - \sum_{i=1}^I \log p(x^{(i)} | x^{(i-1)}, \dots, x^{(0)}, y; \theta) \right\}, \quad (12)$$

To implement equation (12), we assume the conditional probability follows Gaussian distribution and the optimal reconstructed image $x^{(i)}$ has the shortest mean squared distance to the true image. With these assumptions, 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.

$$L^{(i)}(\theta) = \mathbb{E}_{(\bar{x}^{(i)}, x^{(i)}) \sim p_j^{(i)}} [\|\bar{x}^{(i)} - x^{(i)}\|_2^2], \quad (13)$$

where $p_j^{(i)}$ is the joint probability distribution of a training data set (true image data set) and a 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 counterpart. 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.

$$\begin{aligned} \hat{L}^{(i)}(\theta) &= \frac{1}{N} \sum_{n=1}^N [\|\bar{x}_n^{(i)} - x_n^{(i)}\|_2^2], \\ \text{s.t. } x_n^{(i)} &= f_\theta(x_n^{(i-1)}, h^{(i-1)}; y_n), \end{aligned} \quad (14)$$

where, $\hat{L}^{(i)}(\theta)$ is the loss at the i -th iteration parametrized by θ . 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 it. $x_n^{(i)}$ is the intermediate result from the RNN. Each loss $\hat{L}^{(i)}(\theta)$ is backpropagated in each iteration i in the BPTI.

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 do not have an intermediate baseline image $\bar{x}^{(i)}$ available for training. We just have the final baseline images $\bar{x}^{(I)}$ 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 guidance. However, the second issue, which is more serious, is that it would make the backpropagation executed I times instead of once in each training. In this case, the BPTI would significantly slow down the training process. In this research, we evaluated BPTI in a synthetic image study and the result is shown in the next section. For the sake of completing this subsection, let us show how to run the backpropagation in CNN fashion in our RNN.

$$\begin{aligned} \theta_{MAP} &= \arg \min_{\theta} \left\{ -p(x^{(I)} | x^{(I-1)}, \dots, x^{(0)}, y; \theta) \right\}, \\ \hat{L}(\theta) &= \frac{1}{N} \sum_{n=1}^N [\|\bar{x}_n^{(I)} - x_n^{(I)}\|_2^2], \\ \text{s.t. } x_n^{(I)} &= f_\theta \circ f_\theta \circ \dots \circ f_\theta(x_n^{(0)}, h^{(0)}; y_n), \end{aligned} \quad (15)$$

Therefore, the loss $\hat{L}(\theta)$ is backpropagated in the final iteration (I -th iteration) only once in this case.

III. EXPERIMENTS

This section is not available at this time.

IV. CONCLUSION

We have proposed the new Recurrent Neural Network (RNN) architecture for CT image reconstruction. We introduced that the Gated Momentum Unit (GMU) was extended from the Gated Recurrent Unit (GRU) [49] to perform iterative optimization tasks in CT image reconstruction. The RNN GMU has the likelihood term and the prior term analogous to the IR. Unlike the conventional IR as a form of a linear recursive equation, the RNN GMU formulates a non-linear recursive equation. Also, the deep learning-based prior term in the RNN GMU is more sophisticated than the conventional TV prior or the MRF prior. Also, we showed the RNN GMU accelerated the convergence by the momentum method. It helps preserve important long-term information while discarding insignificant short-term detail in the optimization process. The gates in the

RNN GMU are more refined than the simple scaling factors in the momentum method. Furthermore, unlike the Convolutional Neural Network (CNN), the same RNN cell is repeatedly used in every RNN iteration. This helps the RNN cell trained with more number of data sets and it prevents the network from overfitting the input training data. Finally, we showed Backpropagation Through Iteration (BPTI) is effective to obtain higher image quality. The proposed method was evaluated in the synthetic image data. The evaluation results showed that the new RNN architecture was effective to converge faster than any of the other well-known methods. In the synthetic image study, the RNN GMU sinogram/image-domain achieved the best PSNR overall. It got 2.32dB and 7.06dB higher PSNR in the synthetic image study than Learned PDR and U-Net archived. In the real CT data study, the improvements of the RNN GMU were 1.99dB and 3.99dB over Learned PDR and U-Net with 200 epochs. With 1,000 epochs, the improvements were 0.26dB and 4.04dB, respectively. In the evaluation, we also found that the proposed method consumed more computational memory than other well-known methods. We plan to work on how to reduce GPU memory consumption in the near future 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 medical imaging modalities such as Magnetic Resonance Imaging (MRI) by replacing the forward and backward operator. We plan to evaluate this method with MRI data and other imaging modality data.

ACKNOWLEDGMENT

The authors thank Dr. Jonas Adler and Dr. Sebastian Lunz for publishing their software. This research was inspired by their research projects.

The authors also acknowledge the National Cancer Institute and the Foundation for the National Institutes of Health in the creation of the free publicly available LIDC/IDRI database used in this study.

REFERENCES

- [1] Jiang Hsieh et al. Computed tomography: principles, design, artifacts, and recent advances. *SPIE Press*, 2009.
- [2] Thorsten M. Buzug. Computed tomography: from photon statistics to modern cone-beam ct. *Springer*, 2008.
- [3] 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.
- [4] William Henry Nailon. Texture analysis methods for medical image characterisation. *Biomedical Imaging*, <https://doi.org/10.5772/8912>, IntechOpen, 2010.
- [5] 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.
- [6] Bruno De Man et al. Distance-driven projection and backprojection in three dimensions. *Physics in Medicine and Biology*, 49(11):2463-2475, 2004.
- [7] Ge Wang et al. Deep learning for tomographic image reconstruction. *Nature machine intelligence*, 2020.
- [8] G. Wang. A perspective on deep imaging. *IEEE Access*, 4:8914-8924, 2016.
- [9] G. Wang et al. Machine learning will transform radiology significantly within the next 5 years. *Medical Physics*, 44(6):2041-2044, 2017.

- [10] G. Wang et al. Image reconstruction is a new frontier of machine learning. *IEEE Transactions on Medical Imaging*, 37(6):1289-1296, 2018.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] Yann LeCun et al. Deep learning. *nature*, 521(7553):436, 2015.
- [15] Ian Goodfellow et al. Deep learning. *MIT Press*, 2016.
- [16] Kaiming He et al. Deep residual learning for image recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016*, 2016.
- [17] Ian Goodfellow et al. Generative adversarial nets. *Advances in Neural Information Processing Systems (NIPS)*, 2014:2672-2680, 2014.
- [18] Alec Radford et al. Unsupervised representation learning with deep convolutional generative adversarial networks. Paper available online from <https://arxiv.org/abs/1511.06434>, 2016.
- [19] 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.
- [20] Olaf Ronneberger et al. U-Net: convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 9351, 2015.
- [21] 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.
- [22] Alain Horé et al. Image quality metrics: psnr vs. ssim. *2010 20th International Conference on Pattern Recognition*, 2366-2369, 2010.
- [23] 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.
- [24] Jonas Adler et al. Learned primal-dual reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1322-1332, 2018.
- [25] Sebastian Lunz et al. Adversarial regularizers in inverse problems. *Advances in Neural Information Processing Systems (NIPS)*, 2018:8507-8516, 2018.
- [26] W. Yang et al. Deep learning for single image super-resolution: a brief review. *IEEE Transactions on Multimedia*, 21(12):3106-3121, 2019.
- [27] 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.
- [28] 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.
- [29] 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.
- [30] 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.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] 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.
- [36] Adam: a method for stochastic optimization. Diederik P. Kingma et al. Paper available online from <https://arxiv.org/abs/1412.6980>, 2014.
- [37] Mingqiang Zhu, Tony Chan An efficient primal-dual hybrid gradient algorithm for total variation image restoration. *CAM Reports, UCLA, Center for Applied Math.* 08-34, 2008.
- [38] 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.
- [39] Stephen Boyd and Lieven Vandenbergh Convex Optimization *Cambridge University Press*, 2004.
- [40] Ryan 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.
- [41] Aurélien Géron. Hands-on machine learning with scikit-learn, keras, and tensorflow. *O'Reilly*, 2017.
- [42] 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.
- [43] 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.
- [44] Debasmitha 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.
- [45] Yoshua Bengio et al. Advances in optimizing recurrent networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8624-8628, 2013.
- [46] Sepp Hochreiter et al. Long short-term memory. *Neural Computation*, 9(8), 1735-1780, 1997.
- [47] Haşim Sak et al. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. Paper available online from <https://arxiv.org/abs/1402.1128>, 2014.
- [48] Wojciech Zaremba et al. Recurrent neural network regularization. Paper available online from <https://arxiv.org/abs/1409.2329>, 2014.
- [49] 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.
- [50] Sebastian Ruder. An overview of gradient descent optimization algorithms. Paper available online from <https://arxiv.org/abs/1609.04747>, 2016.
- [51] Brendan McMahan et al. Delay-tolerant algorithms for asynchronous distributed online learning. *Advances in Neural Information Processing Systems (NIPS)*, 2014:1-9, 2014.
- [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] Alex Krizhevsky et al. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 2012:1097-1105, 2012.