Application of Deep Learning Neural Networks for Rapid Reconstruction of Spiral K-Space Data for Diagnostic Magnetic Resonance Imaging

Keerthi Sravan Ravi[1]

[1]Department of Electrical Engineering, New York University, New York

December 22, 2017

Abstract

The speed of image reconstruction is a crucial factor in the application of any imaging modality for diagnostic medical imaging. In magnetic resonance imaging, the imaging sensors performing data acquisition encode the data in an intermediate domain, called the k-space domain. The task of image reconstruction involves performing mathematical operations on this k-space data. These mathematical operations are computationally expensive, and depending on the resolution of the desired image, take a long time to execute. This work proposes a solution to reduce the image reconstruction times by implementing a neural network. The neural network effectively learns the mapping between the sampled data values on the spiral-trajectory k-space to the Cartesian image space.

1 Introduction

1.1 Magnetic Resonance Imaging

Medical Resonance Imagine (MRI) is a non-invasive diagnostic medical imaging modality that produces detailed three-dimensional anatomical images without the use of damaging radiation. It utilizes powerful magnetic fields, radio-frequency signals and field gradients to image various anatomies. Depending on the anatomy to be imaged, different imaging protocols are used, and each protocol entails a specific image reconstruction method.

1.2 Image reconstruction

A typical scan involves producing and applying strong magnetic fields to the subject and sensing how the subject's protons react to this field. The sensors encode this data in an intermediate domain called the k-space, which is the Fourier Transform (FT) representation of the Cartesian image space. The k-space represents the spatial frequency encodings of the image space. The mathematical operation of the Inverse Fourier Transform (IFT) is performed to reconstruct a Cartesian image from k-space data.

There exist a variety of Cartesian and non-Cartesian data acquisition methods. In particular, the non-Cartesian methods have advantages such as speed and signal-to-noise ratio (SNR) efficiency, making them favorable. However, these methods involve reconstructing data that consequently do not fall on a Cartesian grid in the spatial frequency domain. Therefore, the k-space data has to be re-sampled onto a 2D Cartesian grid. One of the faster approaches for accomplishing this is 'data-driven' interpolation (known as 'gridding'), where the contribution from each data point is added to the adjacent grid points. Implementing a gridding reconstruction algorithm involves addressing three main concerns: choice of a convolution kernel, the density of the reconstruction grid, and the estimation of the sample density. The shape and size of the convolution kernel directly impacts the aliasing artefacts present in the final image.

1.3 Motivation

As noted above, image reconstruction is a complex task that can be performed in a variety of ways involving multiple design variables. Implementing specific reconstruction algorithms tailored to each data acquisition method further adds to the complexity. The high computation cost of non-uniform Fast Fourier Transform outweighs the modest reduction in reconstruction error [8]. Hence, the standard approach with appropriate parameters remains the practical method of choice for gridding reconstruction in MRI. This work attempts to present a unified approach for reconstructing images using a neural network (NN) based on the work reported by Zhu et al [9]. The NN would aid in reducing the reconstruction times by entirely overcoming gridding. Also, the NN would continuously learn from radiological analysis.

The high-level neural networks API used is Keras [1]. Keras is a high-level neural networks API, based on TensorFlow [2], written in Python [3]. The NN is trained on a VM instance available on Google Cloud Platform's Compute Engine [5], designed to reconstruct images from the intermediate k-space domain. It is important to note that the goal is for the NN to learn the feature mapping without introducing any related mathematical operations at any stage.

2 Methods

2.1 Data acquisition and preprocessing

The dataset of 5992 generic grayscale images was assembled from ImageNet [4]. A Python script was developed to download images from the "Animal", "Plant", "Flora", "Fauna", "Geo" and "Sports" categories. The images were cropped to the central 256×256 pixels and subsequently subsampled to 128×128 using the OpenCV library [6]. The ImageNet dataset was split into training and testing datasets in the ratio $95 : 5$.

K-space (sensor-domain) representations of the grayscale images were generated by a script programmed in Matlab. GPI [7], an open-source Python-based graphical tool for scientific computing, enabled the user to develop, organize and visualize scientific algorithms on a single platform. An algorithm was developed in GPI to perform the task of gridding. Each spiral-trajectory k-space sample of complex data-type consisted of 280 points over 58 shots. Figure 1 shows the GPI algorithm used for generating spiral-trajectory k-space data.

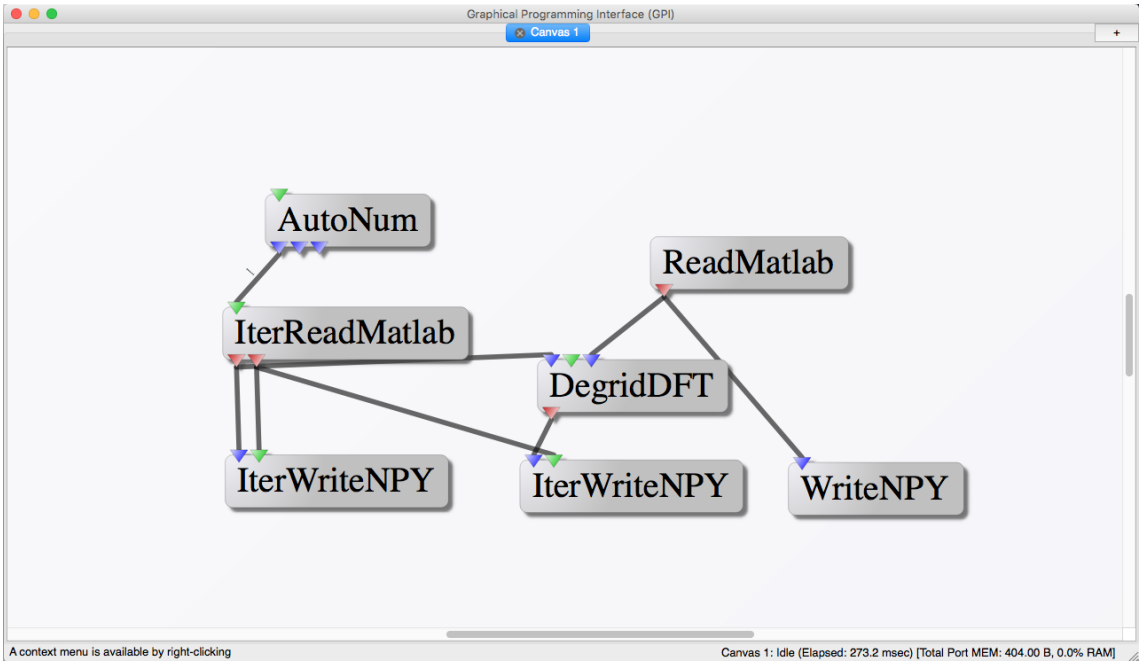

Figure 1: GPI algorithm for generating spiral k-space data from grayscale images.

Architecture	Layers	Total number of trainable parameters
Original	$3FC \rightarrow 3CV \rightarrow 1DCV$	166,813,379
Variation	$3FC \rightarrow 4CV \rightarrow 1DCV$	166,915,971

Table 1: Fully Connected layers, Convolution layers and Deconvolution layers are depicted as FC, CV and DCV respectively. The outputs of the FC layers were reshaped as necessary for convolutional processing. Every CV layer had a subsequent batch-normalization layer.

K-space data per sample was normalized to compress the range of k-space values from 0 to 127. The k-space sample's was then converted into *int16* data-type to mitigate out of memory (resource exhausted) errors. Equation 1 details the normalization procedure.

$$\text{k-space sample } k = \frac{k * 127}{\max |k|} \tag{1}$$

2.2 Designing the neural network

The NN (referred to as *original model* hereon) was designed based on the architecture reported by Zhu et al. One variation of the original model was tried out (referred to as *modified model*).

The original model comprised of three fully connected layers, three 2D convolution layers and one 2D deconvolution layer. Each fully connected layer had 4096 units. Each convolution layer and the one deconvolution layer had one subsequent batch-normalization layer [10] for normalizing data to avoid internal covariate shift. The fully connected, convolution and deconvolution layers were activated by ReLU functions[11], and the initial weights were set to 0.1. Each convolution layer implemented 64 5×5 size filters, defined by a Glorot-uniform initializer [12]. Reshape layers were implemented as needed, for example, to prepare the outputs of the fully connected for convolutional processing. The input was flattened before it was fed to the model. The flattened output was reshaped to a 64×64 image. The original model was compiled with an Adam optimizer with an initial learning rate of $5e^{-3}$, and a custom loss function was defined which normalized the Mean Squared Error (MSE) losses. Batch-size of 32 was used for training purposes. The modified model was configured exactly as above, except that it had an additional 2D convolution layer (and its subsequent batch-normalization layer). Table 1 outlines the two major architectural variations and their respective configurations, and the total number of trainable parameters in each of the models.

Both models were trained for 50 and 500 epochs. Both models were validated against the same set of images, and their normalized losses were recorded. Figure 2(a) shows the architecture of the original model, with the additional layer present in the modified model highlighted with a red box.

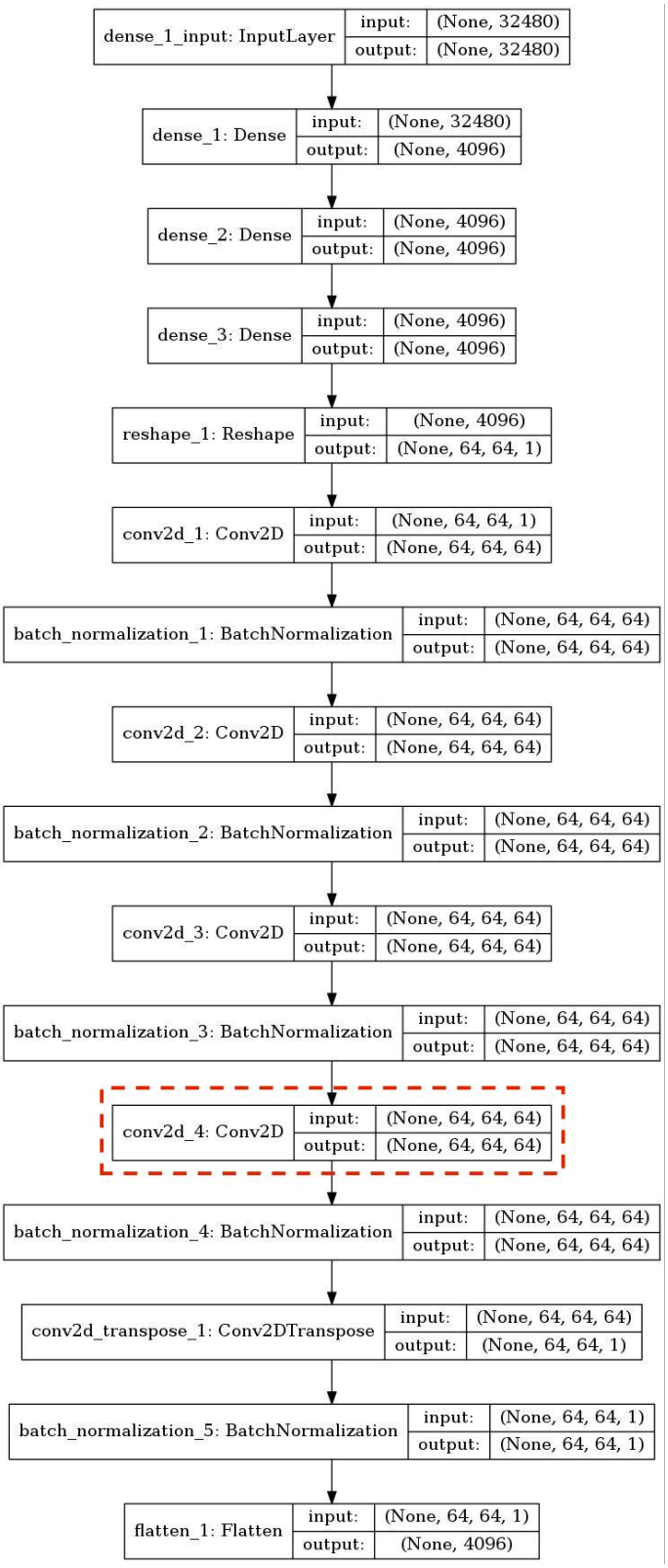

Figure 2: Architecture of the original model.

3 Results

The additional 2D convolution layer in the modified model did not aid in image reconstruction as expected, but instead resulted in higher losses. This anomaly could be attributed to the small size

Model	Epochs	Normalized Loss	Execution times
Original	100	26.8%	5800s @ 58s/epoch
Original	500	3.1%	29000s @ 58s/epoch
Modified	100	27.3%	6800s @ 68s/epoch
Modified	500	3.7%	34000s @ 68s/epoch

Table 2: Normalized error rates and execution times listed per model per run.

of the dataset, and possibly to the lower resolution of the images. Table 2 lists the normalized losses of both models for 100 and 500 epoch runs, along with execution times. Figure 3 shows both model's reconstructions for a standard set of testing images. Figure 3 shows the reconstructions from both models for the same set of k-space data, along with the ground truths.

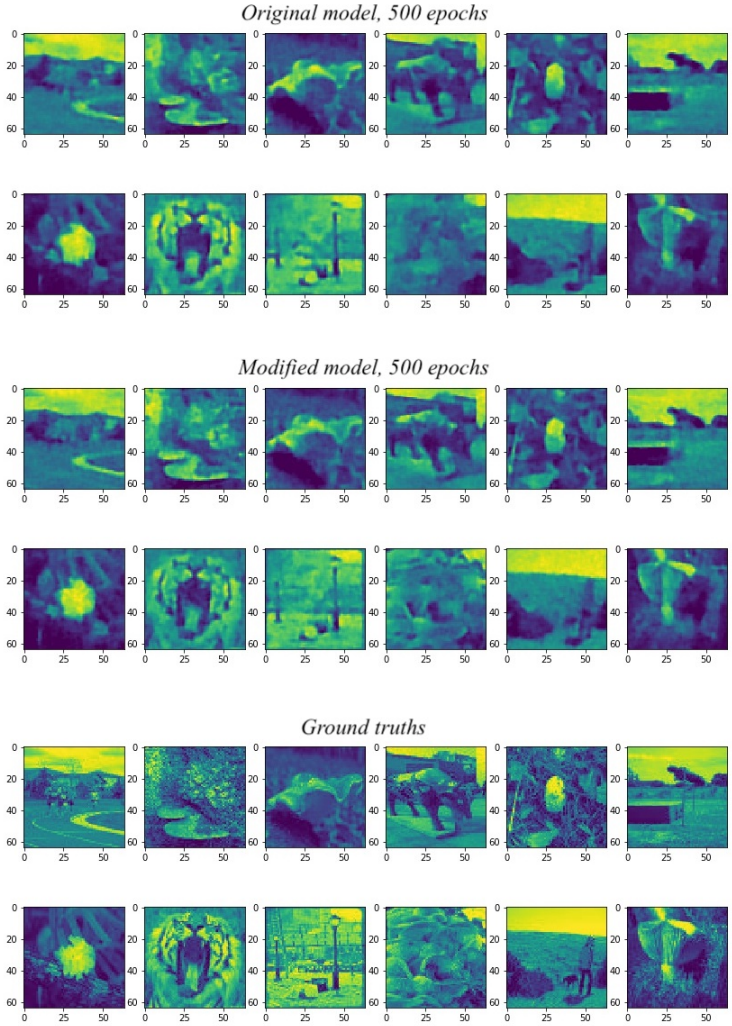

Figure 3: Plots of 12 predictions from the original and modified models, and the ground truths.

Future work includes modifying the model as needed to improve the accuracy, so that Cartesian images can be successfully reconstructed from spiral-trajectory k-space data.

References

[1] Keras available at https://keras.io/.

[2] TensorFlow available at https://www.tensorflow.org/.

[3] Python available at https://www.python.org/.

[4] ImageNet available at http://www.image-net.org/.

[5] Google Cloud Platform available at https://cloud.google.com/.

[6] OpenCV available at https://opencv.org/.

[7] Zwart, Nicholas R., and James G. Pipe. "Graphical programming interface: a development environment for MRI methods." Magnetic resonance in medicine 74.5 (2015): 1449-1460.

[8] Fessler, Jeffrey A. "On NUFFT-based gridding for non-Cartesian MRI." Journal of Magnetic Resonance 188.2 (2007): 191-195.

[9] Zhu, Bo, et al. "Image reconstruction by domain transform manifold learning." arXiv preprint arXiv:1704.08841 (2017).

[10] Understanding the gradient flow through the batch-normalization layer available at https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html.

[11] Hahnloser, Richard HR, et al. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit." Nature 405.6789 (2000): 947-951.

[12] Glorot-Uniform Initializer available at https://keras.io/initializers/glorot-uniform.