Master's program:

Informatics, Image Synthesis and Computer Graphics Design

Adversarial Examples on Accelerated MRI Reconstruction using Recurrent Inference Machines

Master's Thesis

By

Gerasimos Matidis, BSc

October 2019

In the context of research internship in:

University of West Attica – University of Limoges

Master's Program: Informatics, Image Synthesis and Computer Graphics Design

Master's Thesis: Adversarial Examples on Accelerated Magnetic Resonance Image Reconstruction using Recurrent Inference Machines

Author: Gerasimos Matidis, BSc, University of West Attica/University of Limoges

Internship Location: Amsterdam University Medical Center (UMC), Biomedical Engineering & Physics Department (BMEP)

Internship Period: March – September 2019

Abstract

Although deep learning models have achieved human-level performance in a wide variety of image processing tasks, they are found to be susceptible to tiny, carefully calculated, perturbations on the images. The inputs that incorporate this imperceptible noise which intends to downgrade the performance of the networks, are widely known as adversarial examples.

The goal of this thesis was to find adversarial examples for the Recurrent Inference Machine (RIM), a learning framework for solving inverse problems, in the case of accelerated MRI reconstruction. In this way, we checked the robustness and the stability of the network against adversarial attacks.

After running several experiments, we could not find a critical parametrization which the RIM is vulnerable to, under the presence of perturbations in the images. The reason why the RIM is so robust is thought to be its recurrent architecture, with the hidden states it maintains and the way it shares the parameters.

Acknowledgments

First of all, I want to thank my supervisors Kai Lønning and Matthan Caan for their support and the endless time we spent together during this internship.

I also thank:
My colleague Dimitris Karkalousos, PhD candidate in Amsterdam UMC
My professor Thanos Voulodimos
My professors Anastasios Kesidis and Lazaros Grammatikopoulos
My colleagues from BMEP Department of Amsterdam UMC
My colleagues from Spinoza Centre for Neuroimaging

And of course, I thank my family

George, Paula and Pat,

In your case, I feel like “thank you” is not enough.
I am so blessed that I have you!

Contents

Chapter 1

Introduction

1.1 Adversarial examples

In the early 00's, the evolution of Graphics Processing Units (GPUs), with their parallel architecture and their use in the implementation of the calculations within the neural networks [1], allowed for handling enormous amounts of data in way lower times. This fact, in combination with older concepts, such as the Convolutional Neural Networks (CNNs) [2] which drastically reduced the parameters of the traditional ANNs, led the field of Deep Learning to its golden era.

Over time, the interest was focused on computer vision and image processing tasks and especially on image classification. State-of-the-art models with deep architecture, such as GoogleNet [3] and VGG-19 [4] have achieved human-level performance in classification tasks.

However, the deep learning models are very vulnerable under the presence of tiny, invisible by the human eye and carefully calculated perturbations in the images. These malicious inputs, a.k.a. adversarial examples, focus on fooling the networks and cause them to misclassify the images, also with a very high confidence.

In the last years, deep learning models have emerged as a new tool in image reconstruction [5]. In what can be framed as a regression problem, the effects of the adversarial examples can be thought to be similar to classification. In this case, the adversarial examples can be described as "malicious inputs to the models that have been intentionally designed to increase the reconstruction error as much as possible".

1.2 Goal of this thesis

In this work, we deal with the task of reconstruction and in particular with accelerated MRI reconstruction. The term "accelerated" stands for the process of subsampling the raw MR-data, which allows us to accelerate the calculations.

The goal of this thesis is to find adversarial examples for the Recurrent Inference Machine (RIM), a learning framework for solving inverse problems, which is used for accelerated MRI reconstruction. Apart from checking the robustness of the network, we might be able to use the extracted information of the results in order to enhance its performance.

1.3 Overview

Chapter 1 shortly introduces the problem of Adversarial Examples for both classification and reconstruction tasks, as well as it defines the goal of this thesis.

Chapter 2 includes the theoretical background related to this work. It starts with a brief reference to the MRI basics and introduces the k-space, a term that is used for the frequency domain where the MR data is acquired. Thereinafter, it presents the 2 main classes of neural networks, the Convolutional (CNNs) and the Recurrent (RNNs) neural networks (in particular the GRUs, a subclass of RNNs), which are used by the Recurrent Inference machines (RIMs). Finally, it illustrates the problem of accelerated MRI reconstruction and describes the architecture as well as the function of the RIMs.

Chapter 3 starts with the mathematical definition of the problem of finding adversarial examples for the reconstruction task. Then, it includes a step-by-step illustration of the adversarial examples on accelerated MRI reconstruction. Then, it shows the steps of the algorithm for creating the perturbations and the metrics that we use to evaluate the reconstructions.

Chapter 4 includes the results of the experiments of this thesis. In particular, there are plots that depict the effect of the perturbations in different iterations of the algorithm and plots with the values of the metrics for every iteration.

Chapter 5 has the conclusion and a proposal about how this work could be continue.

Chapter 2

Theoretical Background

2.1 The Principles of Magnetic Resonance Imaging (MRI)

Magnetic Resonance Imaging is a medical imaging technique that is used for depicting parts of human anatomy and, especially, the soft tissues of the body. It is based on the Nuclear Magnetic Resonance (NMR) Phenomenon [5]–[7] , which uses the magnetic properties of atomic nuclei in order to extract chemical information about them.

2.1.1 NMR Phenomenon

In their natural state, the spins (protons) are spinning around their axes in random directions (fig. 2.1.1, left). All spins cancel each other and result to a net magnetization $\mathbf{M} = 0$. But once they underlie a strong constant magnetic field $\mathbf{B_0}$, they align with it. Around the half of them align in the same (low-energy) direction and the other half in the opposite (high-energy) one (fig. 2.1.1, right). Spins in low-energy state are slightly more than those in high-energy state (around 4 per 10^6). This difference yields the maximum net magnetization $\mathbf{M} = \mathbf{M}_0$. At this stage, the components of $\mathbf{M}$ are $\mathbf{M}_z = \mathbf{M}_0$ and $\mathbf{M}_{xy} = \mathbf{0}$, where $\mathbf{M}_z$ is the longitudinal component ($\vec{\mathbf{M}}_\mathbf{z} \mathbin{/\!/} \vec{\mathbf{B}}_\mathbf{0}$) and $\mathbf{M}_{xy}$ is the transverse component ($\vec{\mathbf{M}}_\mathbf{xy} \perp \vec{\mathbf{B}}_\mathbf{0}$).

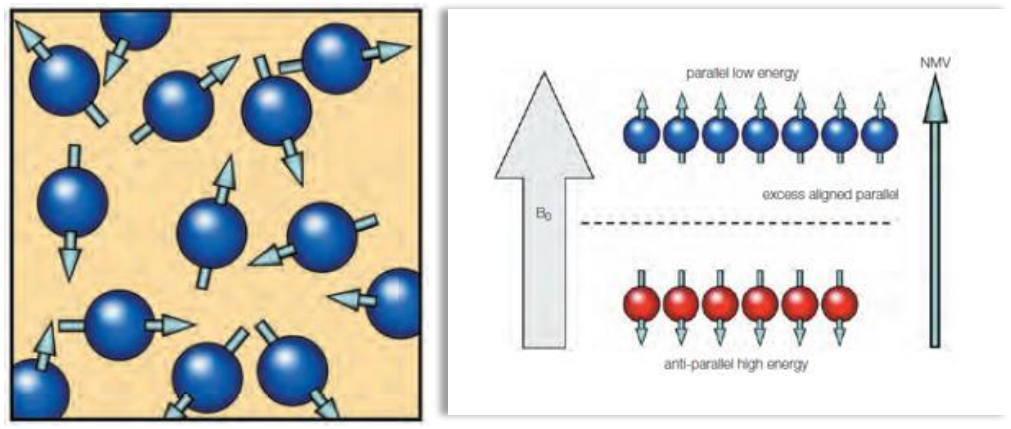


Figure 2.1.1: The natural state of the protons (left) and the net magnetization vector $\mathbf{M}_0$(right) [8]

The idea of NMR is to perturb the protons in the magnetic field, by transmitting an appropriate RF-pulse perpendicularly to $\vec{\mathbf{B}}_\mathbf{0}$ (fig.2.1.2), and acquire the inductive signal, as they return to their initial (relaxation) state. In particular, when the frequency of RF-pulse is chosen to be close to the Larmor Frequency [9] of the protons of interest (usually hydrogens), they absorb energy from the RF-pulse [10] and start precessing around $\mathbf{B_0}$ in a 90° angle. At this stage, the spins are in phase and the components of $\mathbf{M}$ are $\mathbf{M}_z = 0$ and $\mathbf{M}_{xy} = \mathbf{M}_0$.

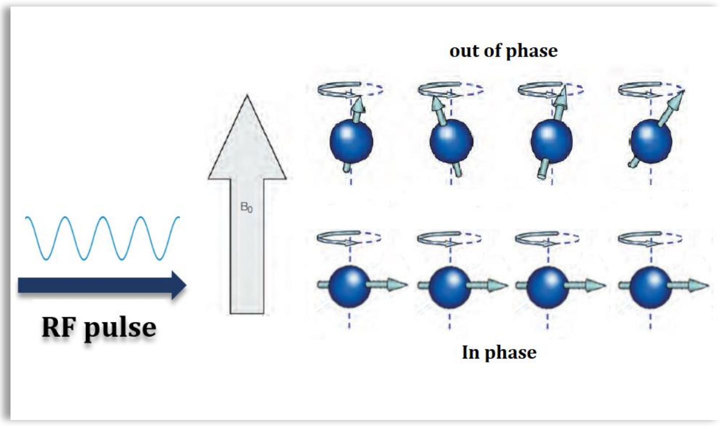


Figure 2.1.2: The RF-pulse transmission (adapted from [8])

Once the RF-Pulse is no longer transmitted, the protons start to return to their initial state, parallel to $\mathbf{B_0}$. At this stage, $\mathbf{M_z}$ starts recovering and $\mathbf{M_{xy}}$ starts decaying (or dephasing). For different type of protons, both components need different times to retrieve about 63% (**T1** relaxation time) or lose about 37% (**T2** relaxation time) of their magnetization, respectively. By measuring these times, we can define the type of the proton and hence create the MR image.

2.1.2 k-space

k-space is a term that is used in MRI for referring to the frequency domain where the data are acquired. It is expressed as an array that stores the raw data obtained from the MR scanner. In particular, the frequency of the signal is encoded along x-direction (k_x) and the phase along y-direction (k_y) (fig 2.1.3, right). Therefore, k-space represents spatial frequencies along its axes, which means that every point (k_x, k_y) does not refer to an individual point in the image, but holds information for every point in it. On the other hand, each point in the image maps to every point in k-space. The transition from k-space to the spatial domain and back, can occur by applying the Inverse Fast Fourier Transform (IFFT) and the Fast Fourier Transform (FFT), respectively.

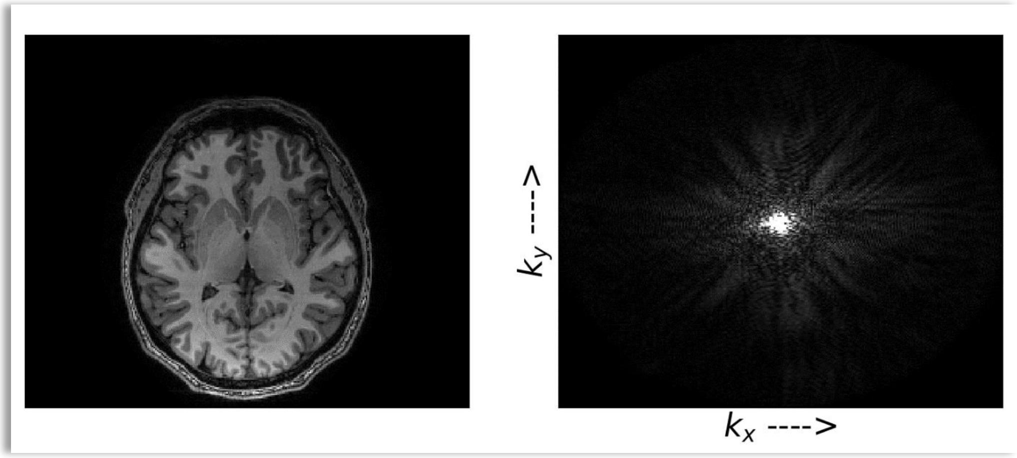


Figure 2.1.3: A brain slice in image space and its representation in k-space.

2.2 The necessity for faster imaging and the role of Deep Learning

The MR signal is directly being quantized once it is captured by the scanner. As the Nyquist – Shannon theorem [11] demands, in order to be able to entirely reconstruct the signal, we have to choose a sampling frequency $f_s \geq 2f_{max}$, where f_{max} is the maximum frequency within the signal.

However, the acquisition of MR-samples is a sequential, time-consuming procedure, with physical and technical limitations in the way of decreasing the scan time. What we do is to acquire less samples than necessary. In this way, we can lower the scan time, but this comes with a trade-off. When we apply the inverse Fourier transform to the sparsely-sampled k-space, we get a corrupted image. The goal, then, is to find the inverse transformation that takes the corrupted image back to its fully-sampled image, a process that is known as "accelerated MRI reconstruction" (see a detailed illustration in figure 2.5.1).

Deep Learning techniques can be an effective tool for addressing this issue. We can build models with deep architecture, which have the ability to detect and extract general patterns within the data, so that they reconstruct the images using less raw data.

In this work, for the subsampling process, we created subsampling masks with 2D Gaussian distribution (fig. 2.2.1). The center of the Gaussian is chosen to be at the center of k-space, as the low-frequencies contain more information about the shape of the content within the image [12]. Note that the subsampling frequency is inversely proportional to the acceleration factor, i.e. the lower the first, the higher the second.

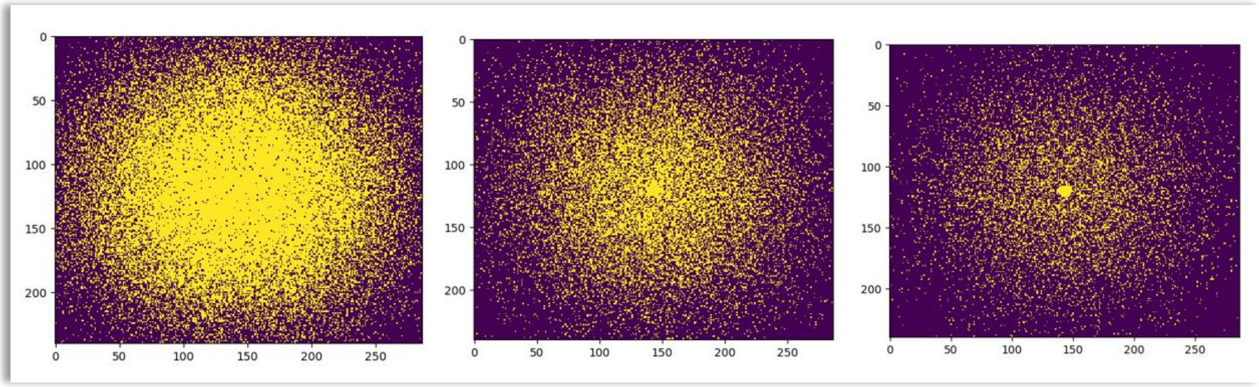


Figure 2.2.1: Sub-sampling masks with acceleration factors (from left to right) 2x, 4x and 8x

2.3 Convolutional Neural Networks (CNNs)

The class of Convolutional Neural Networks [2] is the most representative model in Deep Learning, commonly used for tasks that deal with images. Its name comes from the operation of convolution that takes place between specific layers of the network. In contrast with earlier concepts of neural networks which include fully-connected layers, CNNs use small, usually squared regions (a.k.a. kernels) around a neuron in order to calculate their outputs to the next layer, leading thus to a drastic reduction of the needed operations needed. This idea relies on the locally-sensitive cortical neurons of living organisms, which respond to stimuli only from small regions of the visual field, called receptive fields. The receptive fields correspond to the kernels. In the case of the CNNs, the kernels are expected to capture elementary features (e.g. corners, edges) in the first convolutional layers and combine them in order to capture higher-order features (e.g. polygons, objects) in the next layers. Figure 2.3.1 illustrates how the convolution process works. This example also shows the parameters of the convolutions that we used in this work, which are: kernel size = 3x3 and padding = 1 (i.e. we expand each side of the input image with a row of zero-valued pixels in order to keep the same size in the output image, a.k.a. feature map).

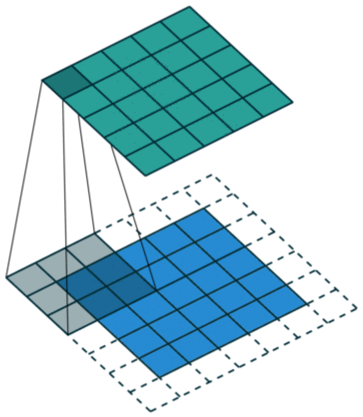


Figure 2.3.1: The convolution process. Each pixel of the output image (green) is calculated as a weighted sum of a region of the input image (blue). This region is defined by the kernel (gray), which also stores the weights. Source: http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html

2.4 Recurrent Neural Networks (RNNs)

The Recurrent Neural Networks form a class of neural networks that is used for processing variable-length sequential data [13]. The term *Recurrent* comes from the fact that the network updates its hidden state for every element of the sequence that passes through it, so that keeps information (memory) about the previous time steps. The update equation that RNNs use is:

$$h_t = f(Ux_t + Wh_{t-1} + b_h) \tag{2.1}$$

and the output is calculated by:

$$y_t = g(Vx_t + b_v) \tag{2.2}$$

where f , g are sigmoid functions (usually *tanh)*, x_t the input, h_{t-1} the hidden state of the previous step U, W, V are weight vectors and b_h, b_v are training biases.

In practice, classic RNNs cannot learn long sequences of data due to the *vanishing gradient problem* [14]. During backpropagation, the network calculates the partial derivatives of the error in order to update the weight of each neuron. In some cases, in which the gradient is extremely small, the weights do slightly change their values and in the worst case, they completely stop being updated. As the partial derivatives are calculated by the chain rule (which means that the gradient is being smaller and smaller while it is backpropagating), one can infer that the first layers tend to be less trained than the last ones.

The two most popular classes of RNNs that address the vanishing gradient problem are the LSTMs (Long Short-Term Memory) and the GRUs (Gated Recurrent Units). These units, by using a gating mechanism, have the ability to decide whether the current time-step information is important and forward it to the next step or not important and discard it. Recurrent Inference Machines use GRUs.

2.4.1 Gated Recurrent Units (GRUs)

Gated Recurrent Units [15] are similar to LSTM but they have fewer parameters. A typical GRU cell (unit) is composed of a memory cell and two gates (also called regulators) (fig. 2.4.1). The update gate determines "how much" of the input must be updated and the reset gate determines "how much" of the input is not important and must be discarded. GRU architecture, as well as, its detailed operations are the following.

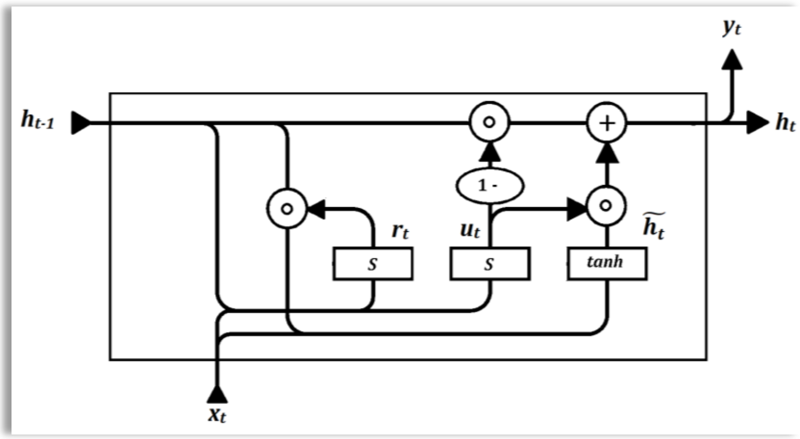


Figure 2.4.1: The architecture of GRU. The operation ⊙ denotes the Hadamard product of the vectors.

The update gate r_t, as well as, the reset gate u_t make use of a logistic sigmoid function in order to calculate the following vectors:

$$u_t = \sigma(U_{u_t}x_t + W_{u_t}h_{t-1} + b_{u_t}) \tag{2.3}$$
$$r_t = \sigma(U_{r_t}x_t + W_{r_t}h_{t-1} + b_{r_t}) \tag{2.4}$$

where x_t the input vector, h_{t-1} the previous hidden state, U_{u_t}, U_{r_t}, W_{u_t}, W_{r_t} the weight vectors and b_{u_t}, b_{r_t} the training biases. Since the two gates use a sigmoid, the range of u_t, r_t is (0, 1).
A candidate vector $\tilde{h}_t$, is calculating by:

$$\tilde{h}_t = tanh(U_{\tilde{h}_t}x_t + r_t \odot (W_{\tilde{h}_t}h_{t-1}) + b_{\tilde{h}_t}) \tag{2.5}$$

Here, the reset gate r_t determines how relevant is the previous hidden layer h_{t-1} to computing the candidate $\tilde{h}_t$. The symbol $\odot$ denotes the Hadamard product of the vectors. The final cell state h_t is calculated by:

$$h_t = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1} \tag{2.6}$$

As one can notice, when $u_t \rightarrow 0$ (no update) the new cell state is $h_t \simeq h_{t-1}$ and when $u_t \rightarrow 1$ (update) the new cell state is $h_t \simeq \tilde{h}_t$.

2.5 The Recurrent Inference Machine (RIM)

The Recurrent Inference Machine [16] is a learning framework with deep architecture, which is a solver for general inverse problems. In this thesis, the RIM is used for accelerated (sub-sampled) MRI reconstruction [17].

2.5.1 Accelerated MRI reconstruction

In the case of the accelerated MRI reconstruction, the forward model can be formulated as:

$$y_i = PfS_i x + n_i, \quad i = 1, 2, \dots, c \tag{2.7}$$

where y_i is the sparse-sampled k-space, P is the sub-sampling mask, S_i the sensitivity map that weights every pixel of the true image according to its distance from the ith coil, x the true image and n_i a matrix that holds an additive noise value for each pixel of the ith coil and follows a Gaussian distribution centered around 0. The symbol f denotes the Fourier transform of the image. y_i then, represents the sparse signal of each coil in k-space.

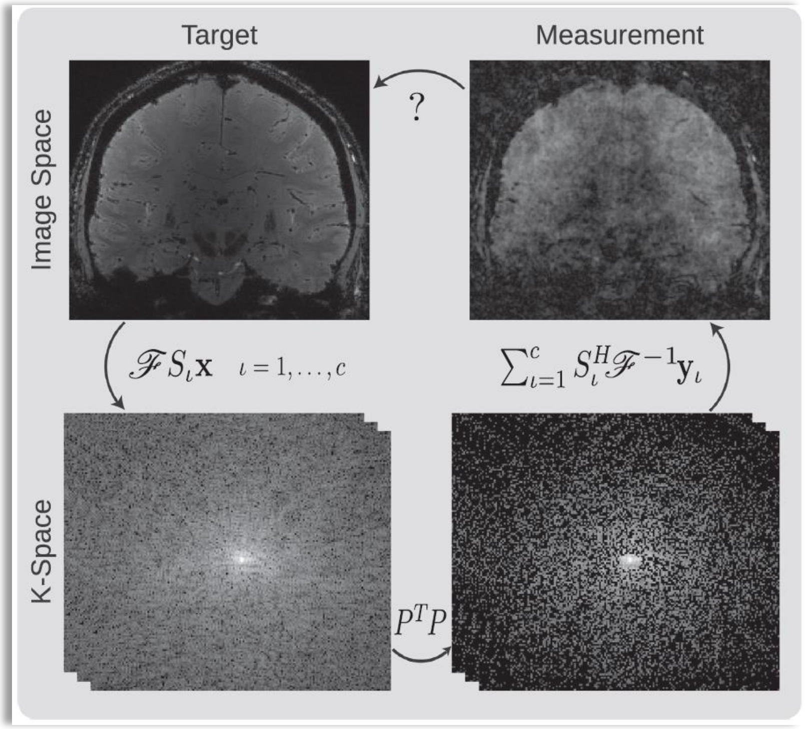

Target
Measurement
?
Image Space
$\mathscr{F} S_\iota \mathbf{x} \quad \iota = 1, \dots, c$
$\sum_{\iota=1}^{c} S_\iota^H \mathscr{F}^{-1} \mathbf{y}_\iota$
K-Space
$P^T P$

Figure 2.5.1: Illustration of the accelerated MRI reconstruction process.

The equation that takes the undersampled k-space back to the corrupted image, is given by:

$$x_0 = \sum_{i=1}^{c} S_i^H f^{-1} P^T y_i \tag{2.8}$$

The RIM takes as input the corrupted image and aims to adapt its weights in order to lead to the target image. The optimization strategy that is used is to maximize the posterior probability $P(\hat{x}|y)$, which is the probability of the reconstruction $\hat{x}$ being correct given the measurements y. According to the Bayes' theorem, it holds:

$$P(\hat{x}|y) = \frac{P(y|\hat{x})P(\hat{x})}{P(y)} \tag{2.9}$$

Since the normalization factor $P(y)$ does not depend on $\hat{x}$, and thus we can ignore it, maximizing 2.9 can be written as:

$$\underset{x}{\operatorname{argmax}}\left(\log P\,(\hat{x}|y)\right) = \underset{x}{\operatorname{argmax}}\left(\log P\,(y|\hat{x}) \;+ \log P\,(\hat{x})\right) \tag{2.10}$$

where $\log P\,(y|\hat{x})$ is the log-likelihood of y given the reconstruction $\hat{x}$ in a certain time-step and $\log P\,(\hat{x})$ is the log-prior distribution, which incorporates the knowledge about what, in general, a MRI image should look like.

By assuming that the noise in the observed values follows an identical, normal distribution with $(0, \sigma)$, we are allowed to express the log-likelihood in 2.10 as:

$$\log P\,(y|x) = \frac{1}{\sigma^2} \sum_{i=1}^{c} \|PfS_i x - \; y_i\|_2^2 \tag{2.11}$$

However, the log-prior distribution $\log P\,(\hat{x})$ is hard to pin down and comes down to the model design. Deep learning algorithms have the ability to learn the log-prior distribution from the data. Hence, in the case of RIM, the current reconstruction x_τ is passed as an input to the network.

2.5.2 The RIM architecture

The RIM that is used in this work is a unit that includes 3 CNN layers and 2 GRU cells, as shown in figure 2.5.2. It maintains 2 hidden states s_t^1 and s_t^2. In each time-step it takes as inputs the current estimation x_τ and the log-likelihood gradient $\nabla_{\mathrm{y}|\mathrm{x_t}} \coloneqq (Pf\mathrm{S_i}\mathrm{x_t} \; - \mathrm{y})$ and outputs the update vector Δx.

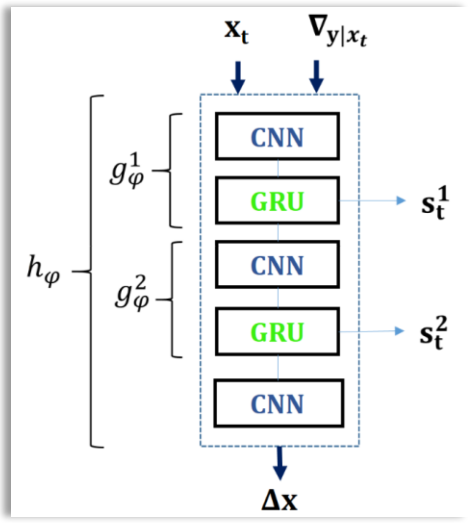


Figure 2.5.2: The RIM unit

The initialized parameters are:

$$s_0^1 = s_0^2 = 0$$

$$\mathrm{x}_0 = \sum_{i=1}^{c} S_i^H f^{-1} P^T y_i\,, \quad i = 1,\ 2, \dots, c \text{ (the corrupted image)}$$

The update equations are:

$$\mathrm{s}_{\mathrm{t}+1} = g_\varphi\,(\nabla_{\mathrm{y}|\mathrm{x_t}},\ \mathrm{x_t},\ \mathrm{s_t})$$

$$\mathrm{x}_{\mathrm{t}+1} = \mathrm{x_t} + \underbrace{h_\varphi\,(\nabla_{\mathrm{y}|\mathrm{x_t}},\ \mathrm{x_t},\ \mathrm{s}_{\mathrm{t}+1})}_{\Delta x}$$

Chapter 3

Implementation Environment

3.1 Introduction

As it has already been referred, the purpose of this work is to examine the performance of the RIM in MRI reconstruction under the presence of small, almost invisible by the human eye, perturbations on the images. Saying "performance" we mean the instabilities that may occur due to the perturbations and lead the network to way different reconstructions with unacceptable errors.

The procedures to this 2-fold goal are based on the work of Antun et al. [5]. According to the authors, the main idea behind the experiment is the following: "*Given an image and a neural network, designed for image reconstruction from samples provided by a specific sampling modality, the algorithm searches for a perturbation of the image that makes the most severe change in the output of the network while still keeping the perturbation small.*"

The aforementioned goal can be mathematically expressed as:

$$\max_{r} \|\mathcal{R}(y + Pr) - \mathcal{R}(y)\| \qquad \text{subject to } \|r\| \leq \theta \tag{3.1}$$

where $\mathcal{R}$ is the RIM, y the forward model as expressed in 2.7, P the subsampling mask, r the perturbations and θ a threshold. As we will see in §4.2, it is not always possible to choose an appropriate value for θ, because of the distribution of the calculated perturbations on the image. In this work, we approach the problem using a ridge regression model, such as to maximize the following loss function:

$$Q_y(r) = \|\mathcal{R}(y + Pr) - \mathcal{R}(y)\|_2^2 - \lambda\|r\|_2^2 \tag{3.2}$$

3.2 Adversarial Examples on accelerated MRI reconstruction

The step-by-step process of how we create adversarial examples in the case of the accelerated MRI reconstruction, is illustrated in figure 3.1.1. Once the raw data (top left) has been subsampled (top center), we add the subsampled perturbations to them. By taking the weighted sum of the inverse Fourier transform of each coil, we can create the perturbed corrupted image (bottom left). Respectively to the process in §2.5.1, the goal is to go for the perturbed corrupted image back to the target image, in which, this time, we have added the perturbations in image space (bottom right).

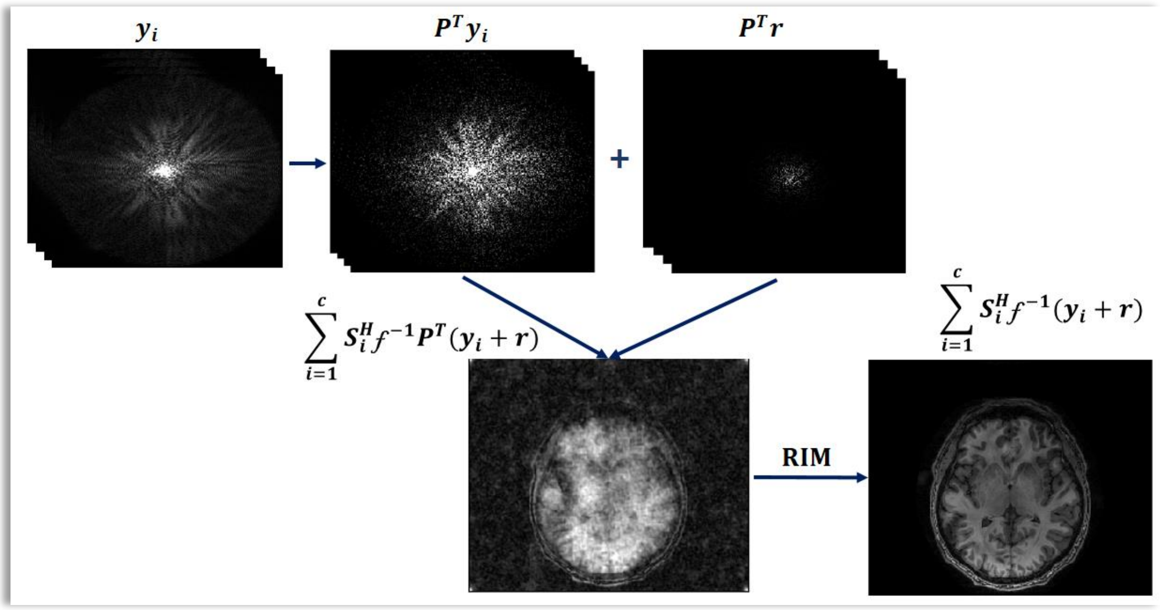

y_i
$P^T y_i$
+
$P^T r$
$\sum_{i=1}^{c} S_i^H f^{-1} P^T (y_i + r)$
$\sum_{i=1}^{c} S_i^H f^{-1} (y_i + r)$
RIM

Figure 3.1.1: The process of creating adversarial examples on accelerated MRI reconstruction

3.2 The Network Parameters

For the procedures of this work, we trained 3 models that only differ in the number of the feature maps they extract. In particular:

- In the first 2 convolutional layers, the 1st model extracts 32 feature maps, the 2nd 64 feature maps and the 3rd 128 feature maps respectively.
- In the third convolutional layer, all the models extract 2 feature maps.
- The kernel size for all the convolutional layers is 3x3.
- In both of the GRU layers, the 1st model extracts 32 feature maps, the 2nd 64 feature maps and the 3rd 128 feature maps respectively.

The other parameters of the training process are the same for all of the models and they are the following:

Dataset: 256 T1-weighted slices of a brain in transverse direction with size 240 x 287, acquired by a 32-channel head coil, *Batch size:* 8, *time-steps:* 8, *Optimizer:* ADAM, *learning rate:* 0.001, *momentum:* 0.8, *number of epochs:* 40, *Activation function:* ReLU, *loss function:* L1.

For the reconstruction process, as well as, for creating perturbations, we used as target image a slice from the middle of the dataset volume. By using the algorithm in §3.3, we created 18 sets of reconstructions with all the possible combinations between *time-steps* = {2, 8}, λ = {1.0, 1.5, 2.0} and *feature maps* = {32, 64, 128}. For corrupting the target image in k-space, we used a sub-sampling mask with a Gaussian distribution centered at the center of the image and acceleration factor 8x.

3.3 The algorithm

The optimization algorithm that we use for approaching appropriate local maxima for 3.2 is Gradient Ascent with Momentum. The parameters, as well as, the steps of the algorithm are the following:

- Input Parameters: the true image x, the network h, the sub-sampling mask P, the sensitivity map S and the number of the iterations i.
- Output parameter: The perturbations r.
- Initialized parameters:
  - $y = P\mathcal{F}Sx$, the corrupted image in k-space.
  - $\lambda = \{1.0, 1.5, 2.0\}$, the regularization factor.
  - $\eta = 0.001$, the learning rate.
  - $\gamma = 0.9$, the momentum.
  - $\tau = 0.05$, a scaling factor that its value depends on the image.
  - r_0, with size equal to the image, is randomly initialized in the interval [0, 1] and then multiplied by the scaling factor τ.
  - $i = 2000$, the number of iterations.

Algorithm's procedure:

```
While $i \leq 2000$:
    $v_{i+1} = \gamma * v_i + \eta * \nabla_r Q(r_i)$
    $r_i = r_i + v_{i+1}$
```

3.4 The evaluation metrics

The metrics that are used for examining the difference (or the similarity) between the reconstruction of the corrupted image and the reconstruction of the perturbed (corrupted image plus the perturbations) image, are the following:

- <u>RMSE</u> (Root Mean Square Error):

$$RMSE = \sqrt{\frac{1}{i * j} \sum_{i=1}^{h} \sum_{j=1}^{w} (x_{i,j} - \hat{x}_{i,j})^2} \tag{3.3}$$

  where $x_{i,j}$ and $\hat{x}_{i,j}$, the value of the pixel (i, j) of the perturbed and the corrupted image, respectively and h, w, the size of the height and the width of the images, respectively.
- <u>SSIM</u> (Structural Similarity):

$$SSIM(x, \hat{x}) = \frac{(2\mu_x \mu_{\hat{x}} + c_1)(2\sigma_{x\hat{x}} + c_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + c_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + c_2)} \tag{3.4}$$

where $\mu_x, \mu_{\hat{x}}$ the average of x and $\hat{x}$ respectively, $\sigma_x, \sigma_{\hat{x}}$ their respective variance and $\sigma_{x\hat{x}}$ their covariance. The variables c_1 and c_2 are used to stabilize the division with a potentially weak denominator. Their values are $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$, where $k_1 = 0.01$ and $k_2 = 0.03$ by default and L is the dynamic range of the pixel-values.

- PSNR (Peak Signal-to-Noise Ratio)

$$PSNR = 10 \log_{10}\left(\frac{maxI^2}{MSE}\right) \tag{3.5}$$

where $maxI$ is the maximum possible pixel value and MSE the mean square error.

Chapter 4

Implementation and results

4.1 Introduction

In the previous chapter, we referred that we created 18 sets of reconstructions of perturbed images, by alternating the number of the feature maps, the time-steps and the regularization factor λ. For every one of these sets we calculated the 3 metrics that were referred in §3.4.

In order to visualize the procedures and their results, we created 3 different types of plots. The one plot depicts the effects of algorithm's execution w.r.t. the iterations (fig. 4.1.1). In the first row, it shows the target image (left), the target image plus the perturbations (center) and just the perturbations (right). In the second row, it shows the reconstructions of the corrupted image (left), the corrupted image after adding the perturbations (center) and their absolute difference (right). The reason why we choose to plot the target and the perturbed target image instead of the corrupted and the perturbed corrupted image is because they are used as reference for calculating the metrics. The other two plots show the variation of the errors on the reconstructions. In particular, the one shows, side-by-side, the errors of reconstructions for different number of feature maps (fig. 4.1.2) and same all the other parameters and the other shows, respectively, the errors of reconstructions for different time-steps (fig. 4.1.3). The gray areas on the plots define a range of iterations where the perturbations are clearly visible on the corrupted image, so we have to ignore the errors in there. The left edge of the areas define a boundary where the perturbations start to be visible.

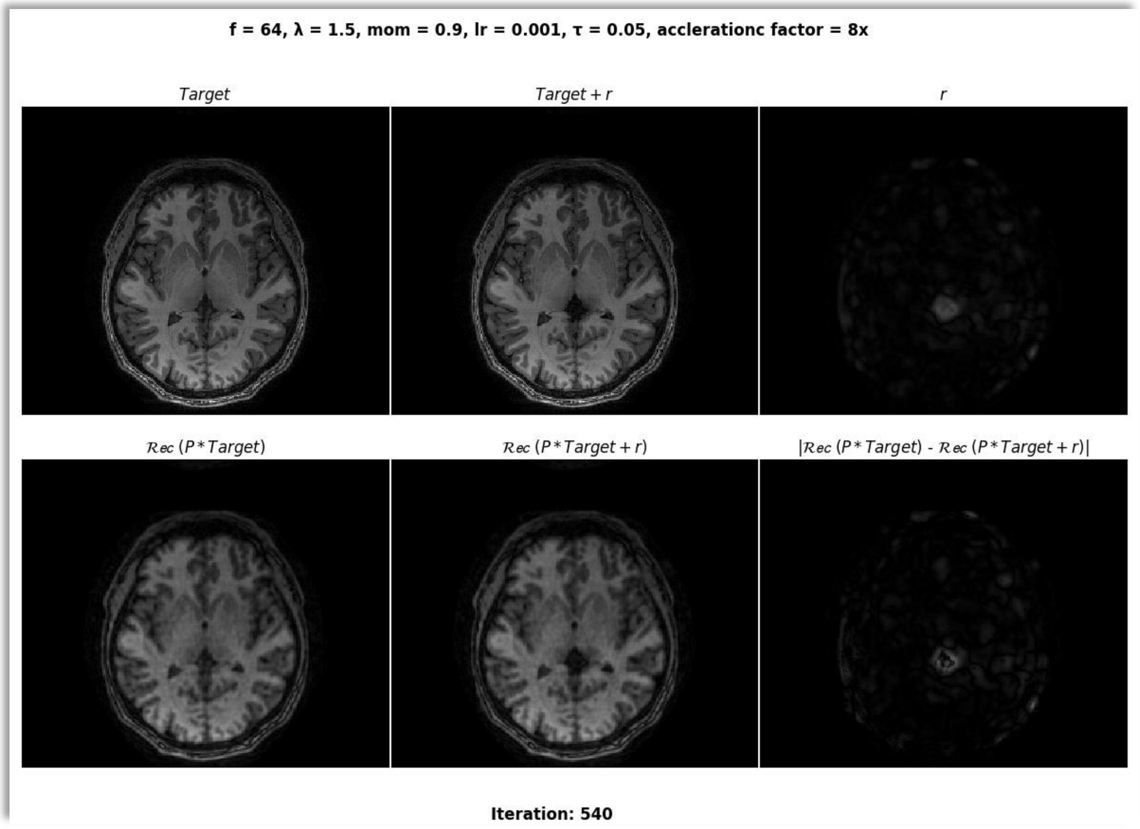

f = 64, λ = 1.5, mom = 0.9, lr = 0.001, τ = 0.05, acclerationc factor = 8x
Target
Target + r
r
Rec (P*Target)
Rec (P*Target + r)
|Rec (P*Target) - Rec (P*Target + r)|
Iteration: 540

Figure 4.1.1: Example of a plot of the algorithm's procedures w.r.t. the iterations (all the images are in the image space).

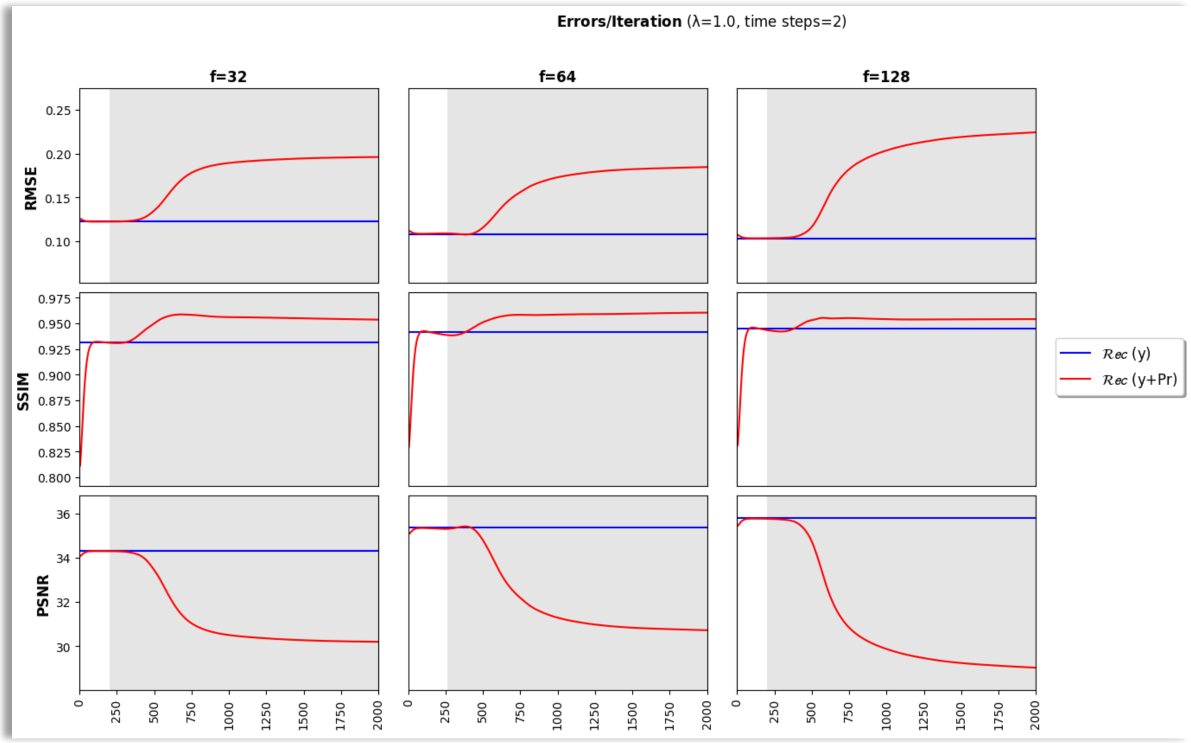


Figure 4.1.2: Example of a plot of the errors' values w.r.t. the iterations, for different number of feature maps.

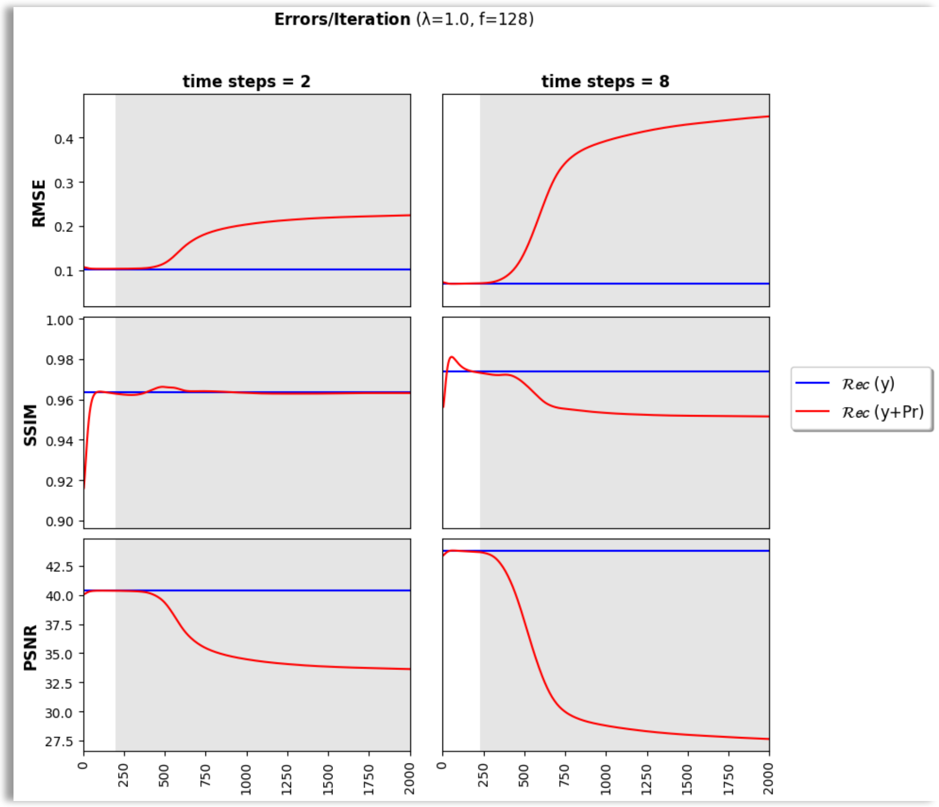


Figure 4.1.3: Example of a plot of the errors' values w.r.t. the iterations, for different number of time-steps.

4.2 Results

This section includes images that illustrate the effects of the algorithm for different number of iterations, as well as, all the plots of the errors per iteration. In particular, subsection 4.2.1 includes plots that show the images as the iterations (hence, the perturbations) are being increased. Since the number of these plots for all the 18 combinations between feature maps, time-steps and λ is too big, we are only going to present the plots for $f = 32, t = \{2, 8\}$ and $\lambda = 1.5$. An extra reason for that is that for all the combinations these plots are similar. Subsection 4.2.2 includes the 6 plots of the errors for different number of feature maps and subsection 4.2.3 the 9 plots of the errors for different number of time-steps.

4.2.1 Plots of the algorithm's effects for different iterations

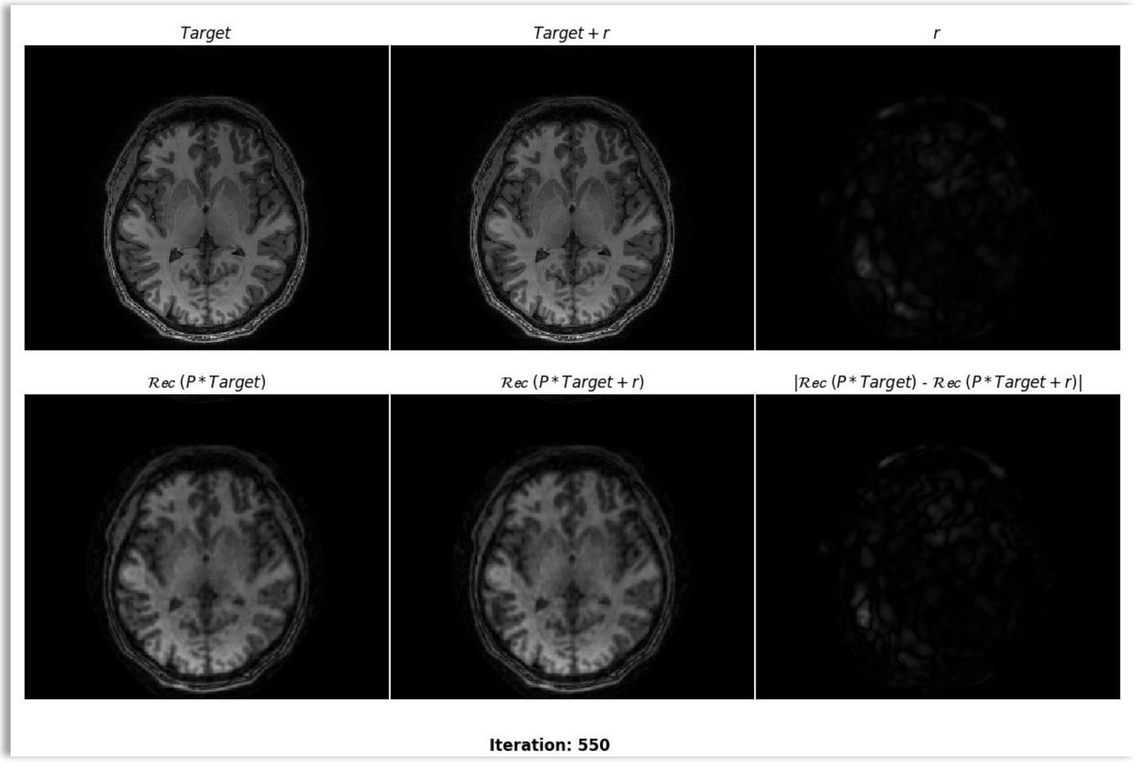


Figure 4.2.1: f = 32, t = 2, λ = 1.5

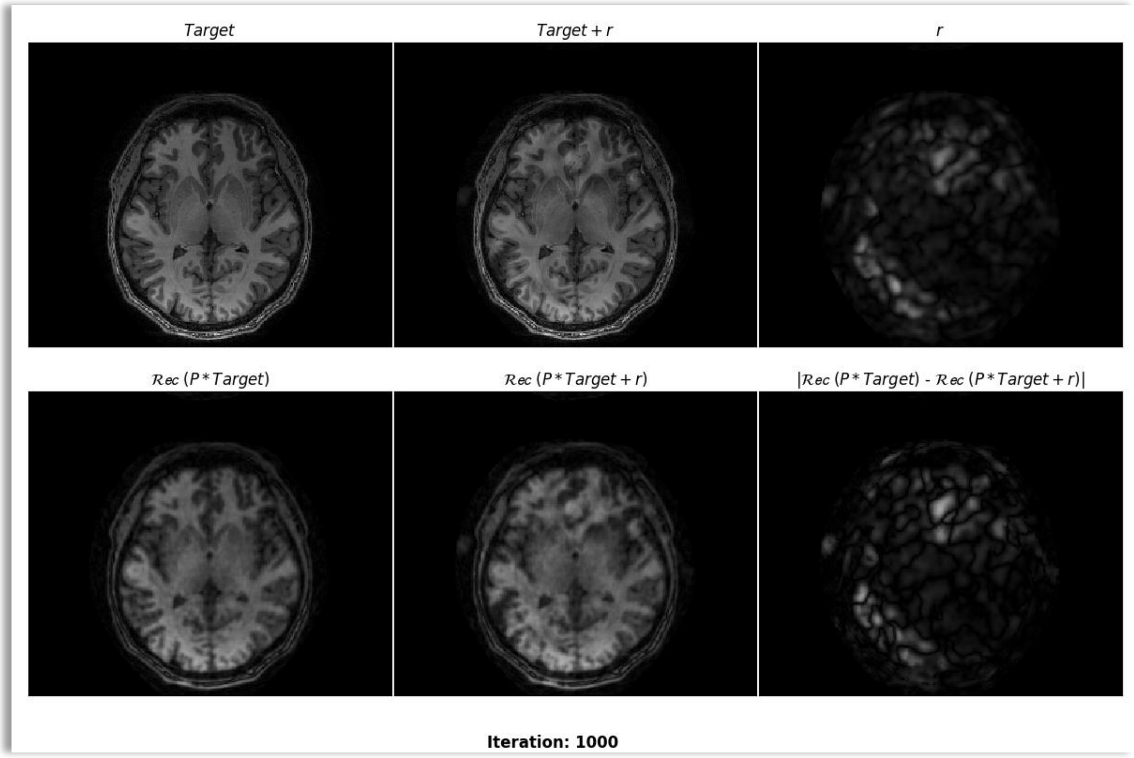


Figure 4.2.2: f = 32, t = 2, λ = 1.5

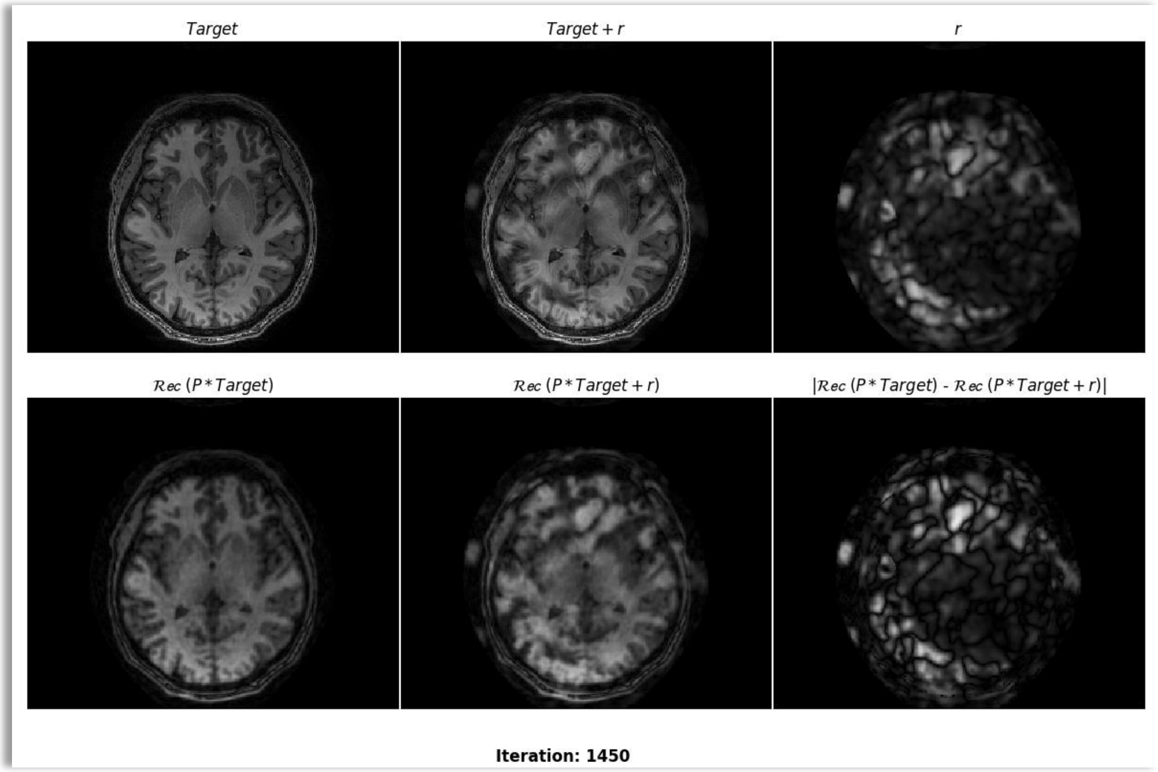


Figure 4.2.3: f = 32, t = 2, λ = 1.5

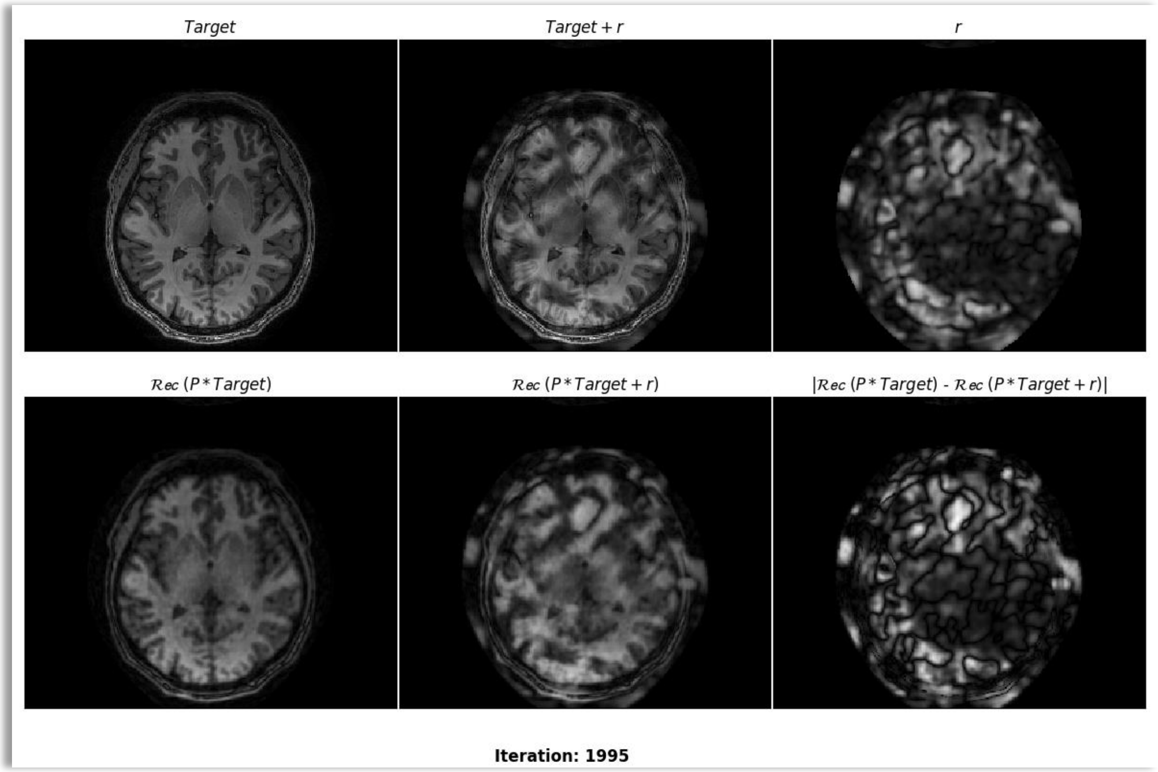
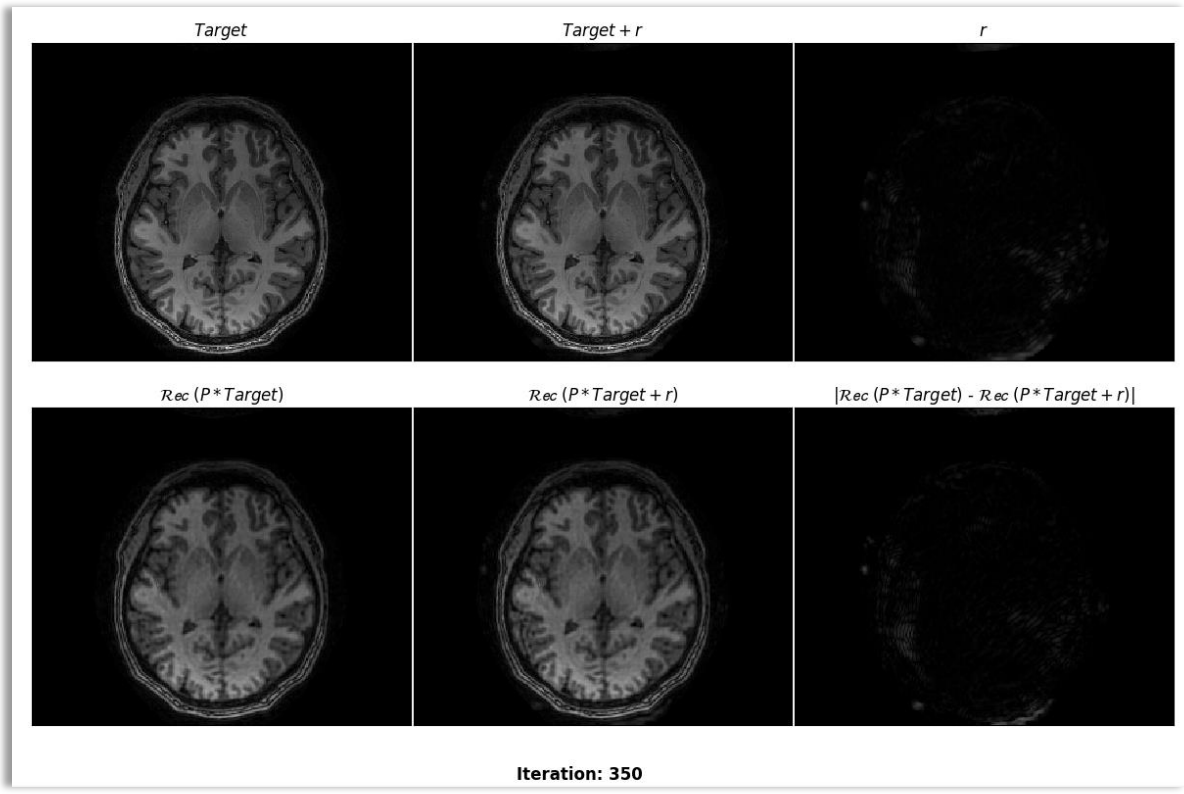
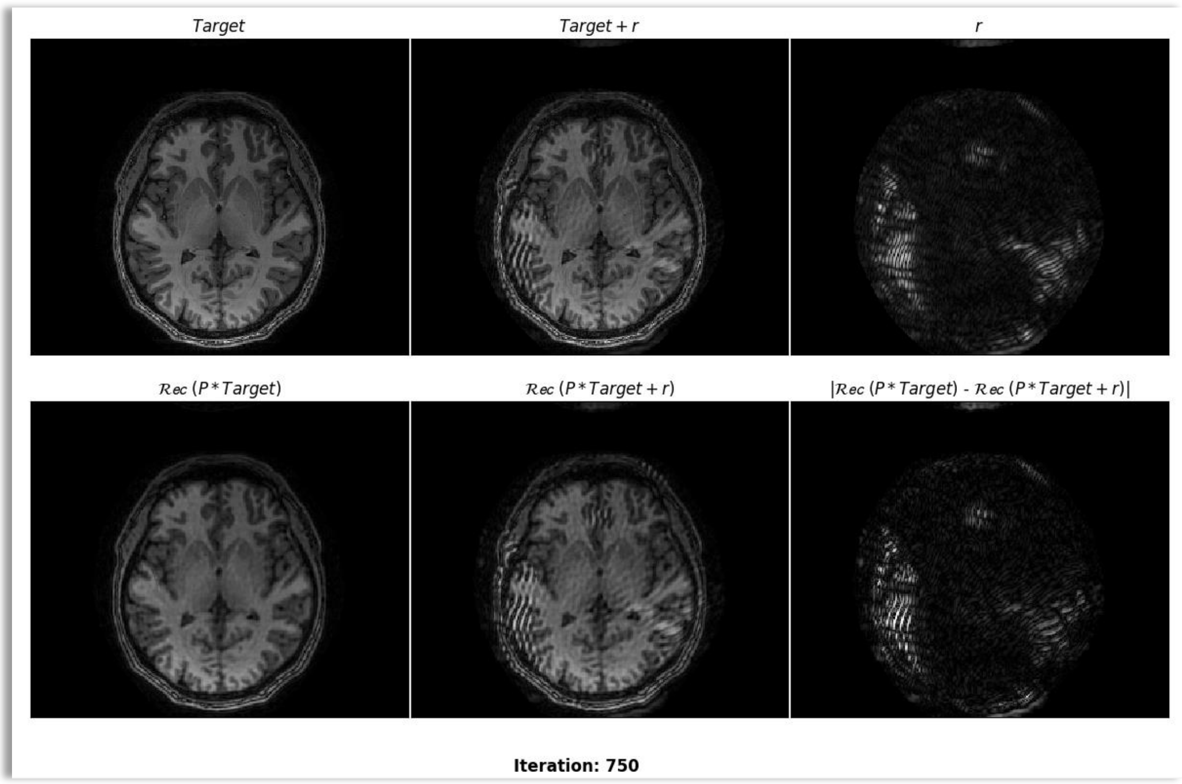


Figure 4.2.4: f = 32, t = 2, λ = 1.5



$Target$
$Target + r$
r
$\mathcal{R}ec\,(P * Target)$
$\mathcal{R}ec\,(P * Target + r)$
$|\mathcal{R}ec\,(P * Target)$ - $\mathcal{R}ec\,(P * Target + r)|$
Iteration: 350

Figure 4.2.5: f = 32, t = 8, λ = 1.5



$Target$
$Target + r$
r
$\mathcal{R}ec\,(P * Target)$
$\mathcal{R}ec\,(P * Target + r)$
$|\mathcal{R}ec\,(P * Target)$ - $\mathcal{R}ec\,(P * Target + r)|$
Iteration: 750

Figure 4.2.6: f = 32, t = 8, λ = 1.5

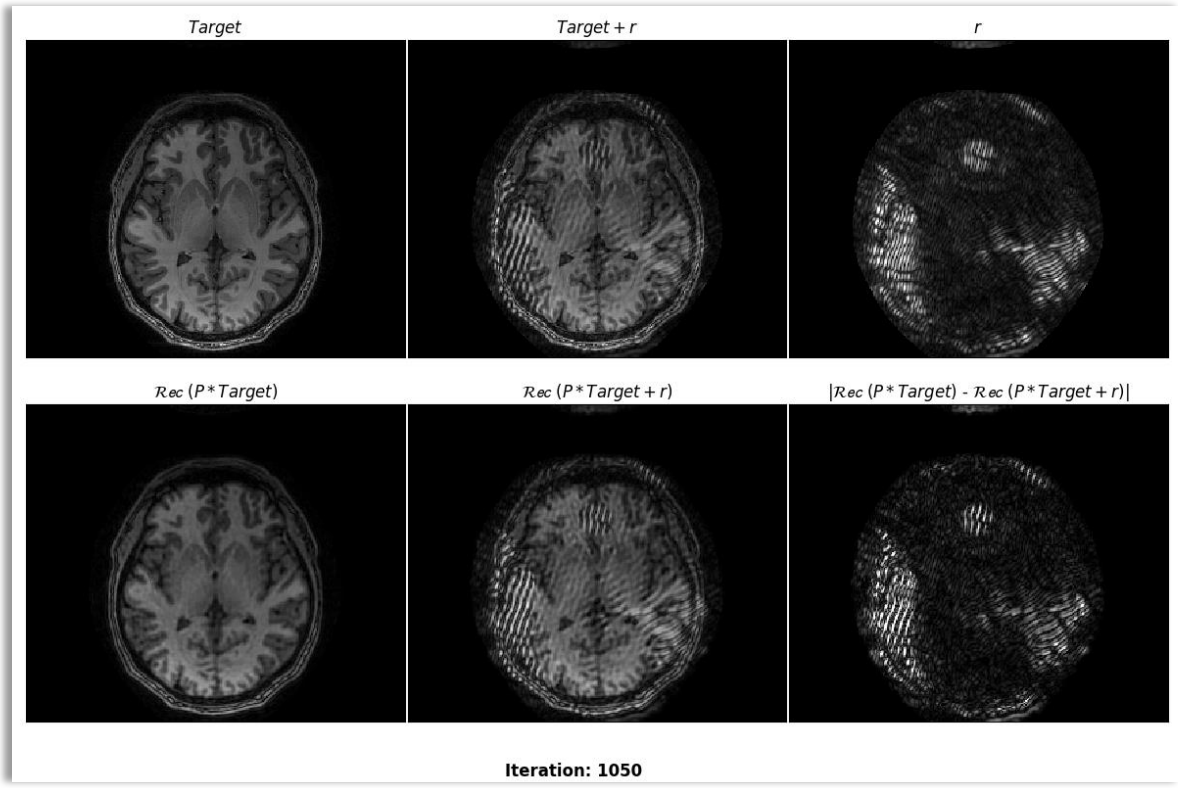


Figure 4.2.7: f = 32, t = 8, λ = 1.5

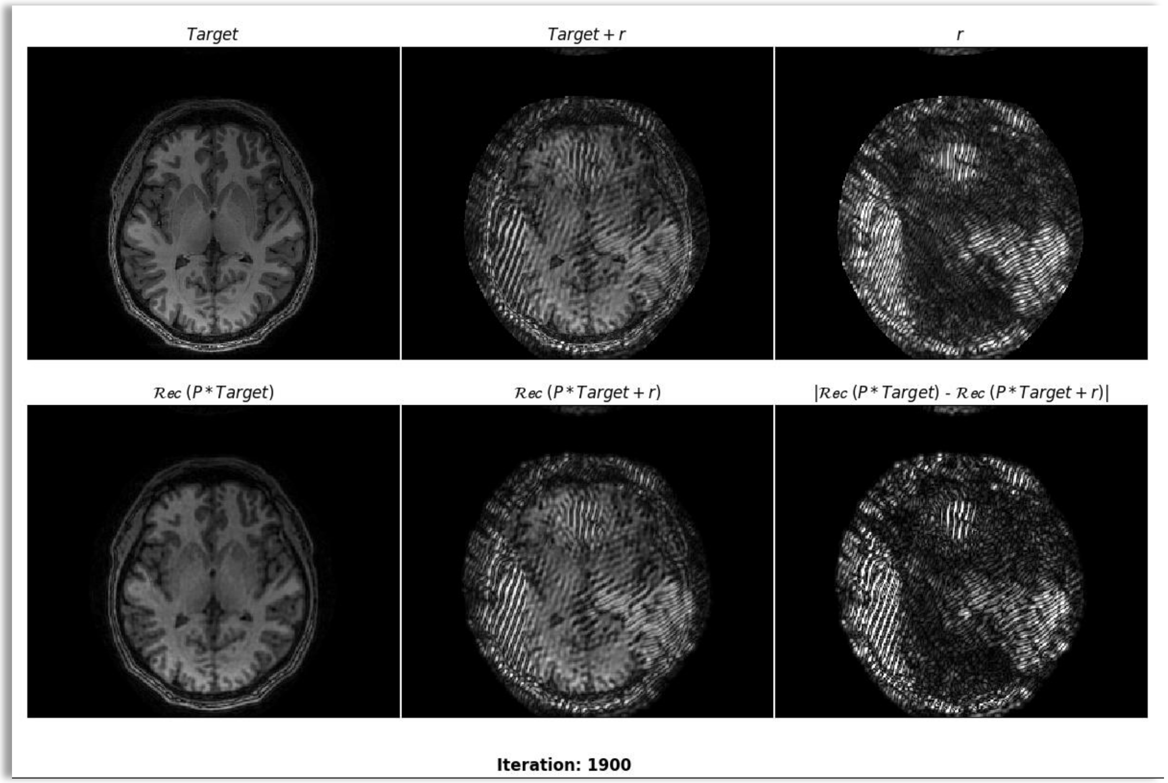


Figure 4.2.8: f = 32, t = 8, λ = 1.5

4.2.2 Plots of the errors w.r.t. the iterations for different number of feature maps

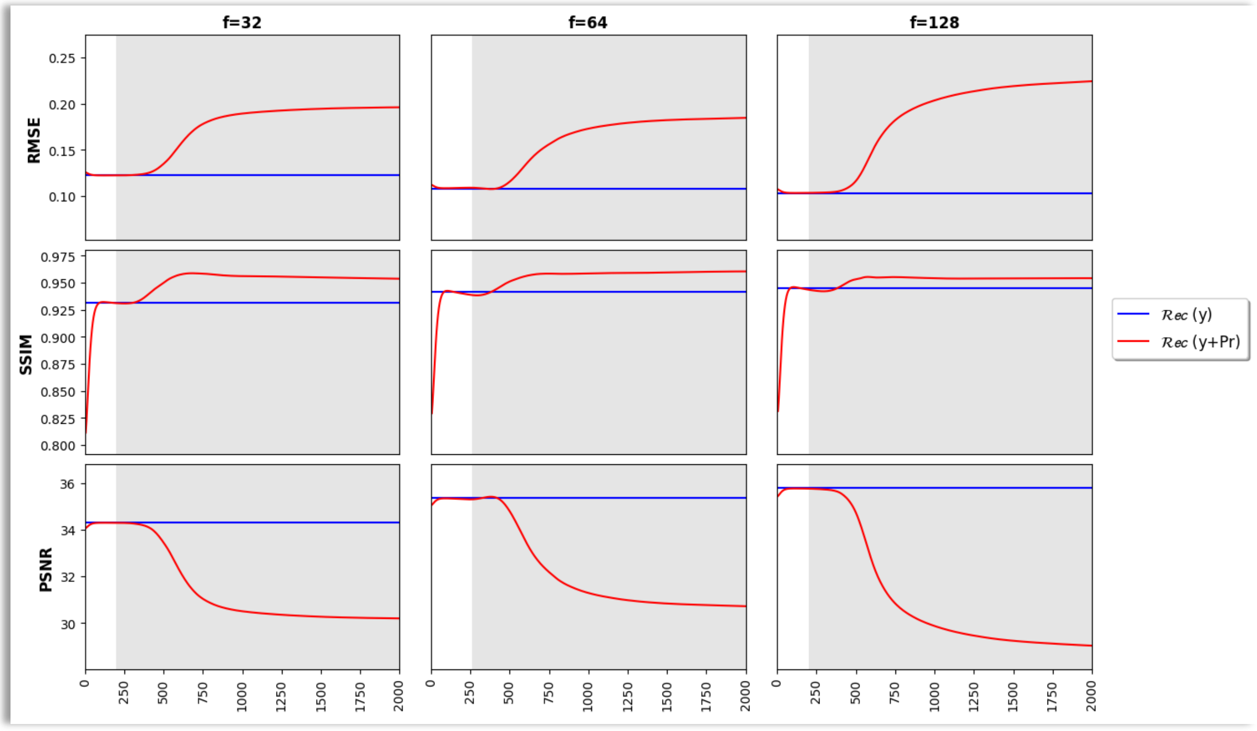


Figure 4.2.9: t = 2, λ = 1.0

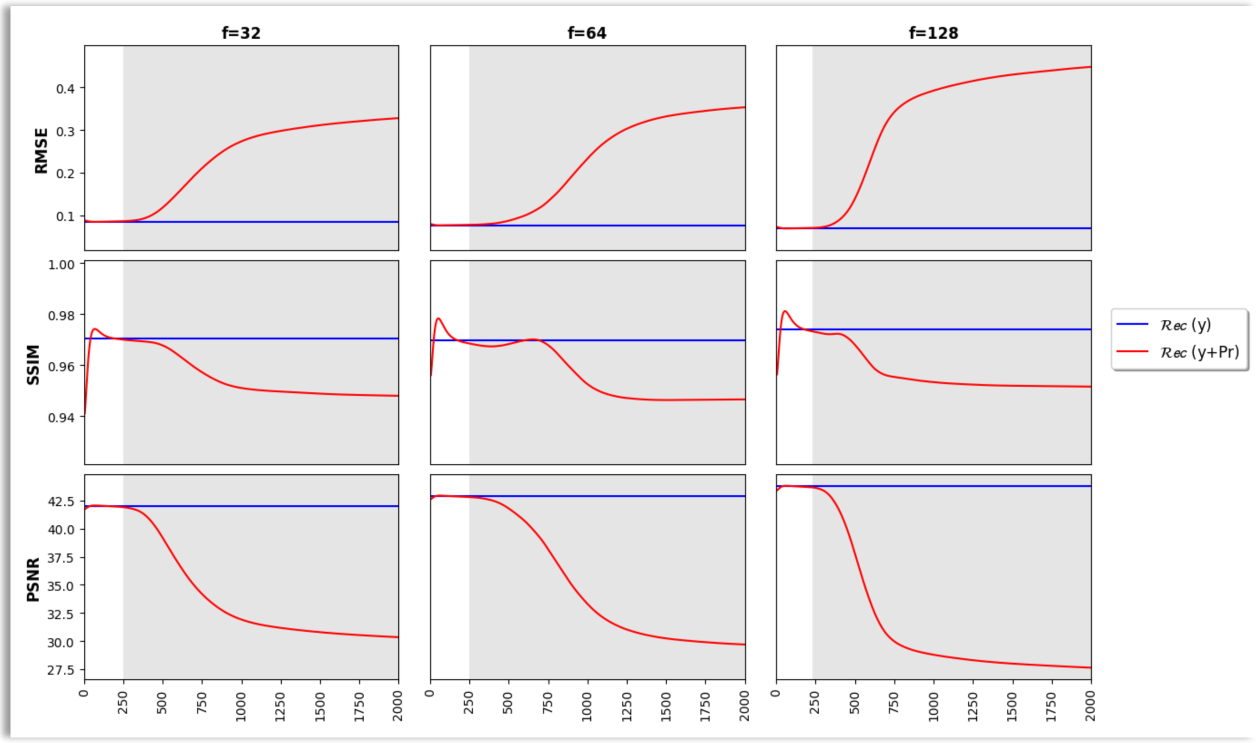


Figure 4.2.10: t = 8, λ = 1.0

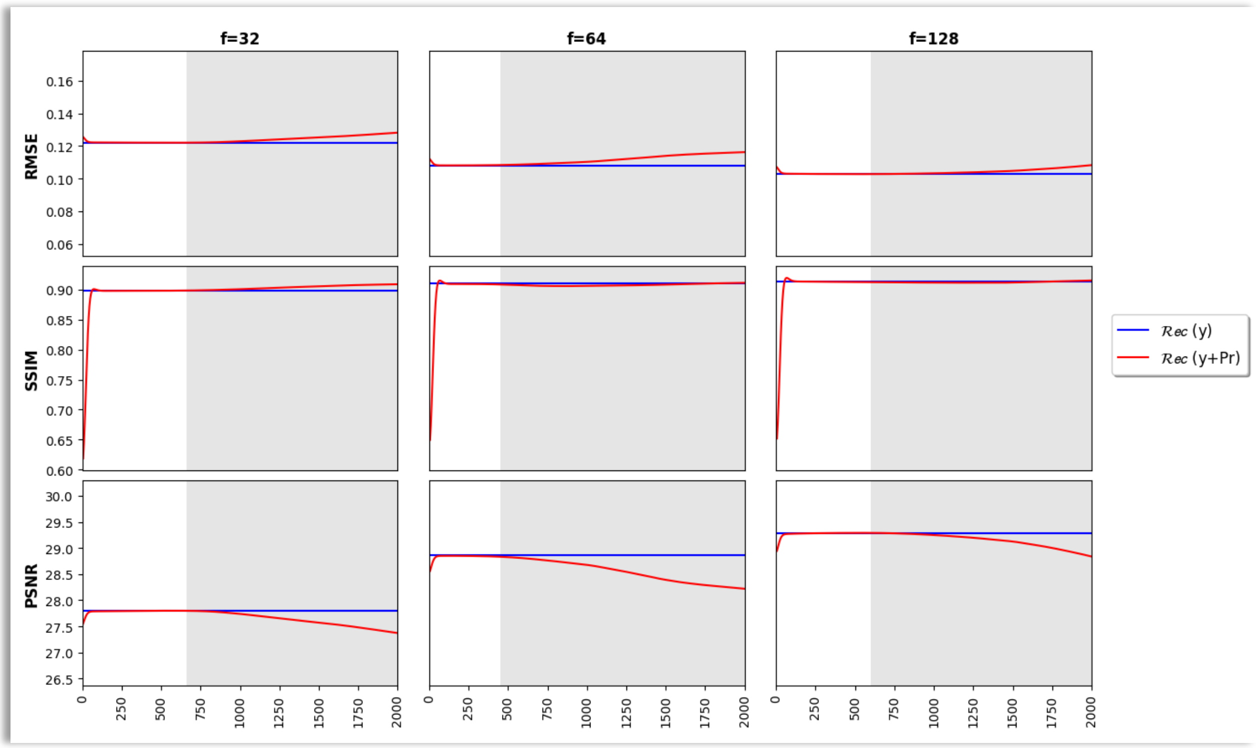


Figure 4.2.11: t = 2, λ = 1.5

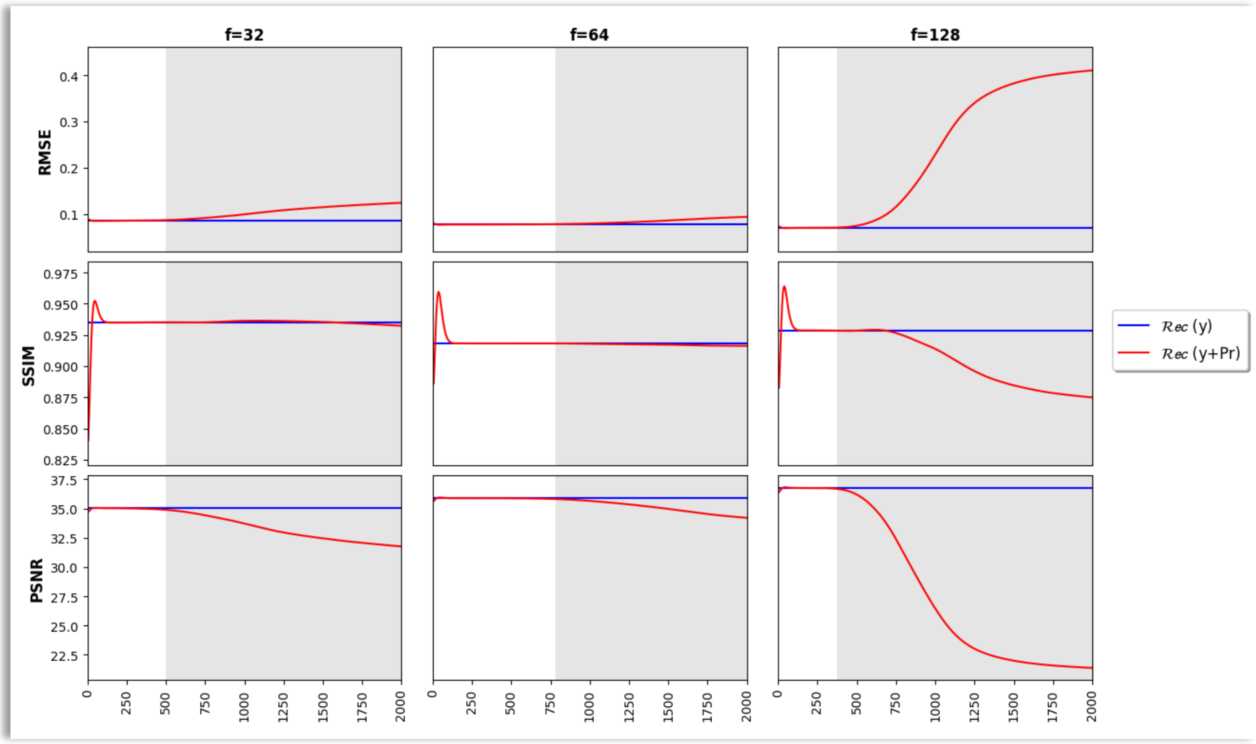


Figure 4.2.12: t = 8, λ = 1.5

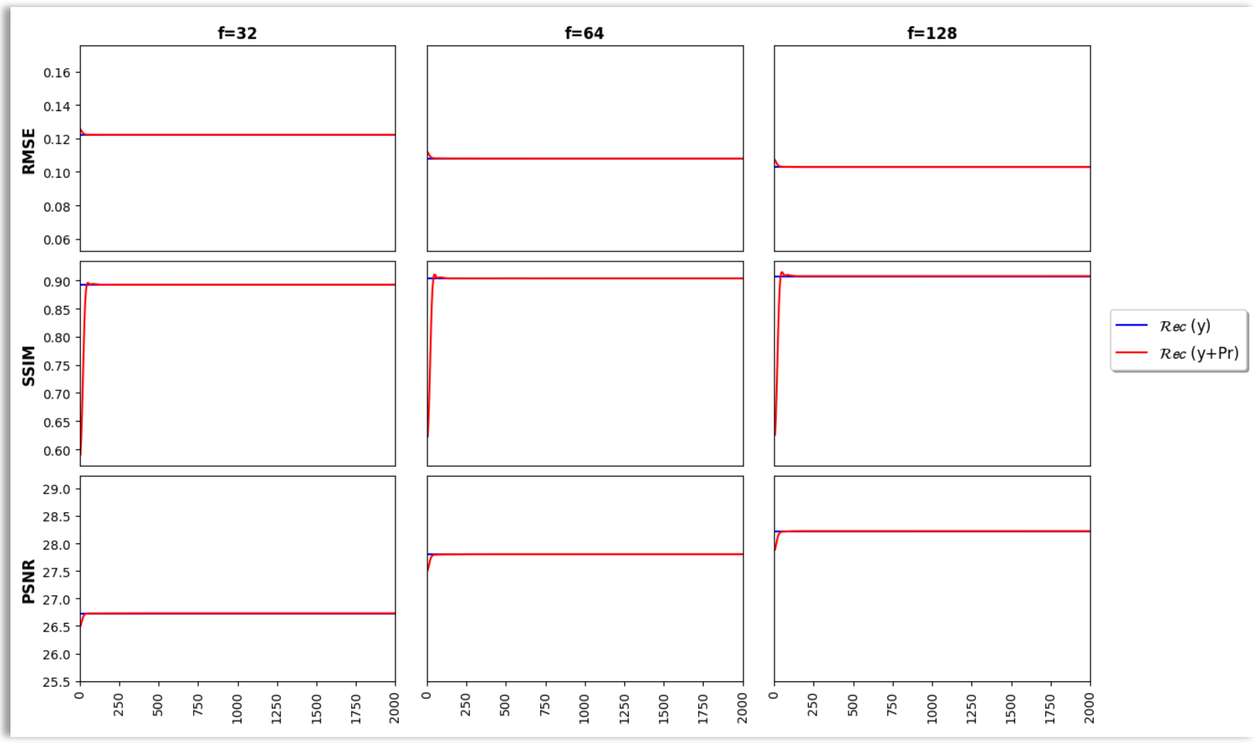


Figure 4.2.13: t = 2, λ = 2.0

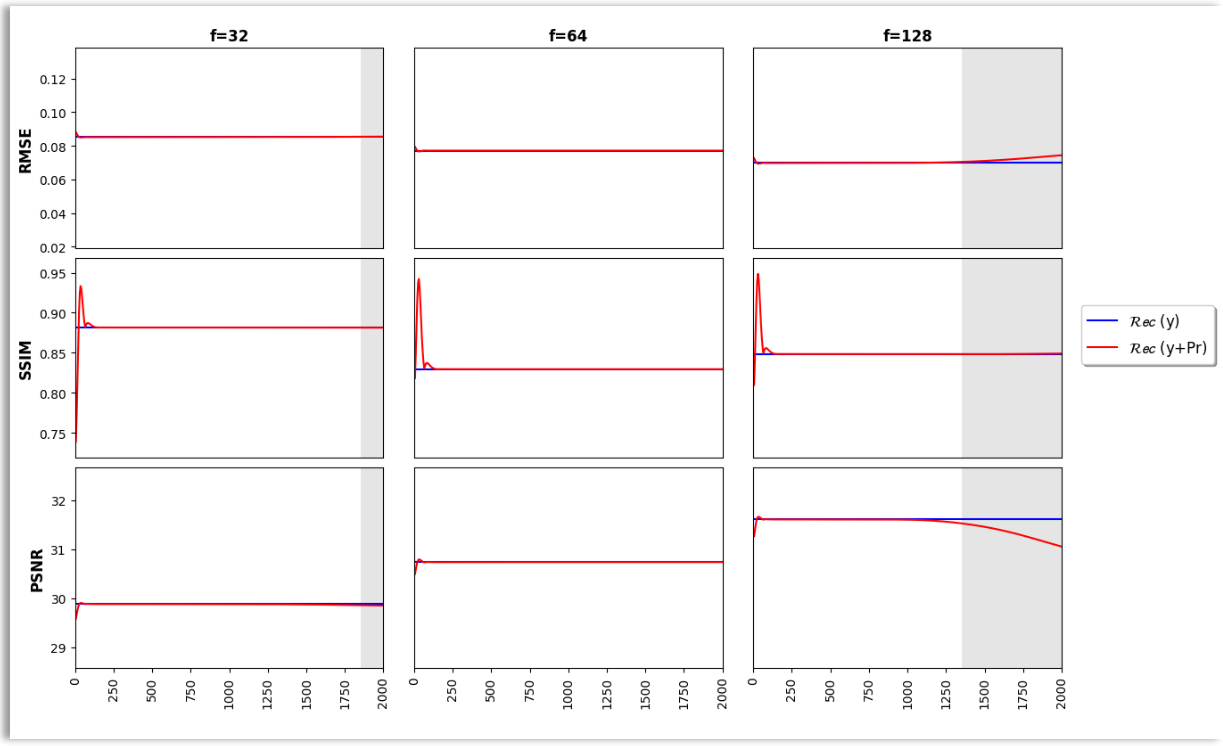


Figure 4.2.14: t = 8, λ = 2.0

4.2.3 Plots of the errors w.r.t. the iterations for different number of time-steps

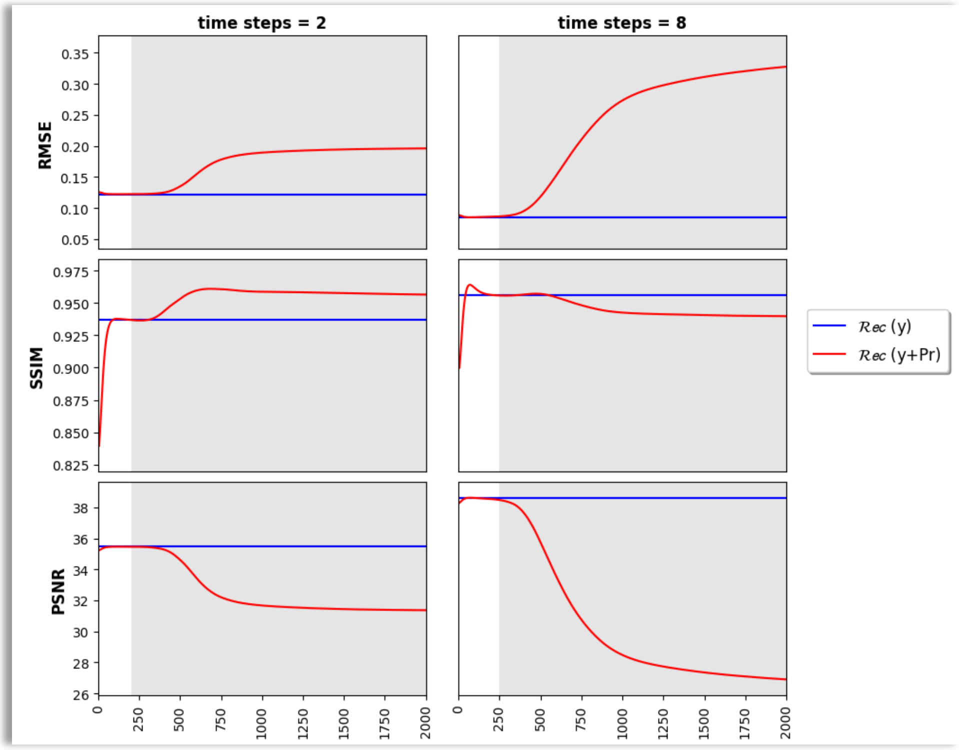


Figure 4.2.15: f = 32, λ = 1.0

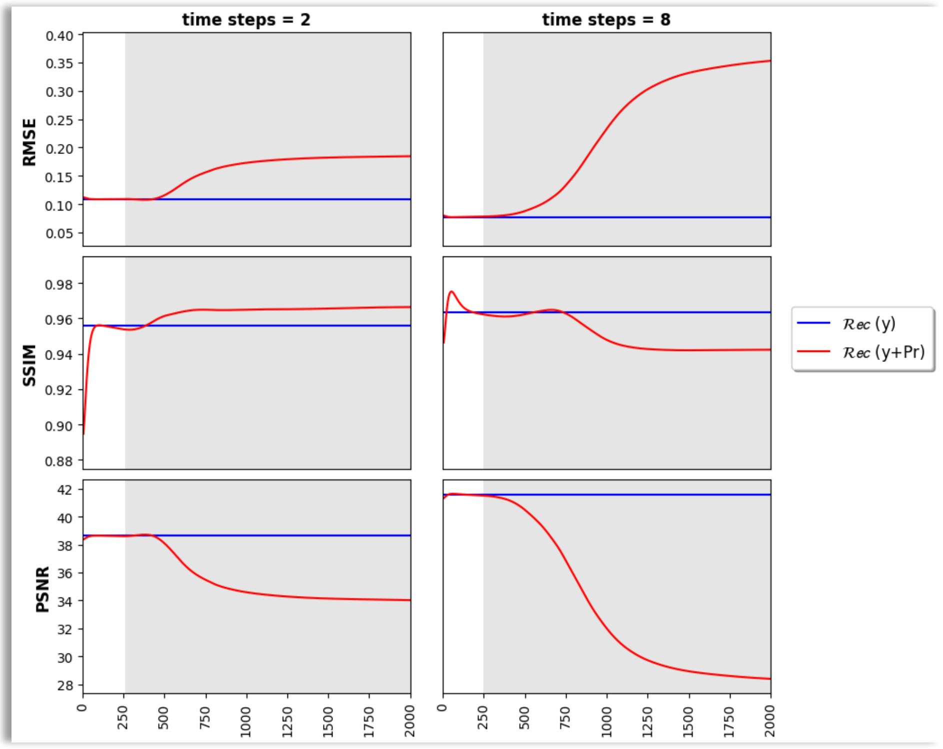


Figure 4.2.16: f = 64, λ = 1.0

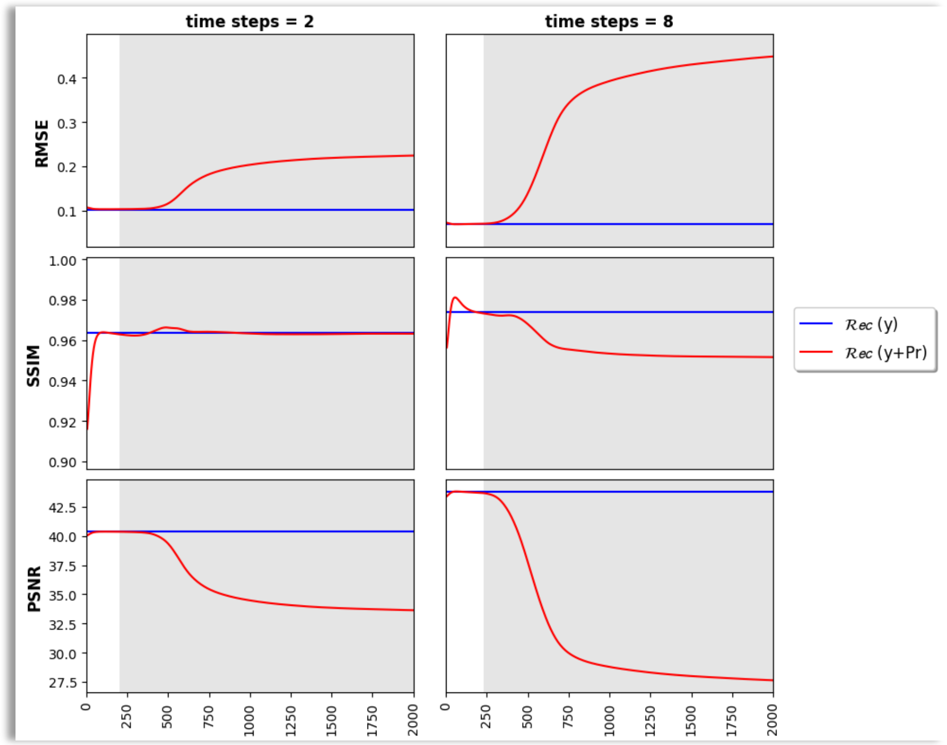


Figure 4.2.17: f = 128, λ = 1.0

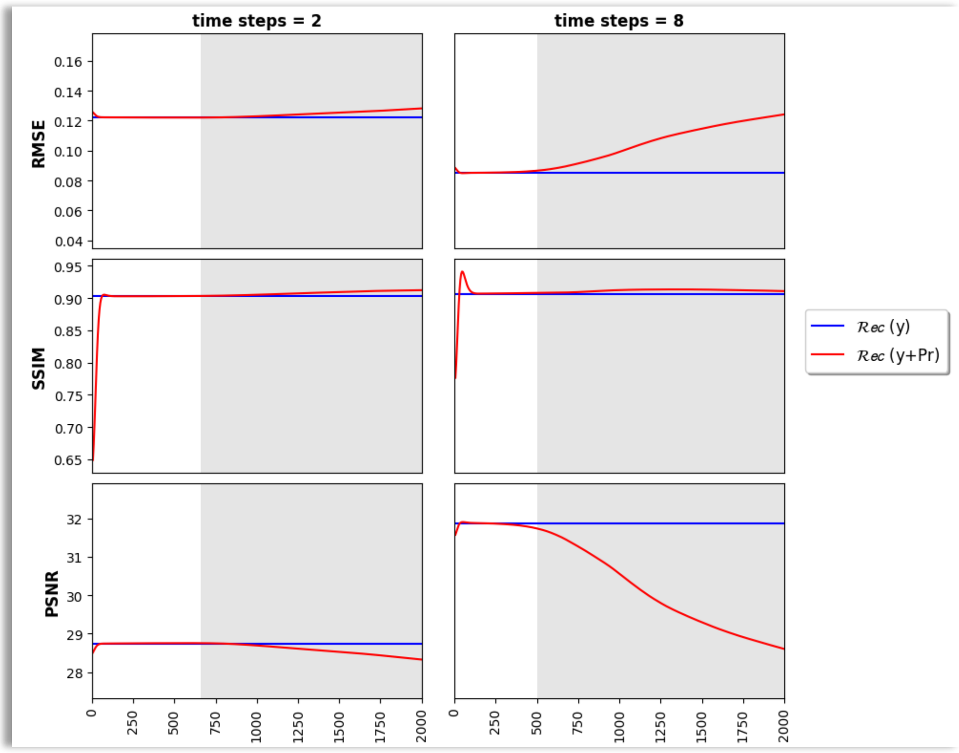


Figure 4.2.18: f = 32, λ = 1.5

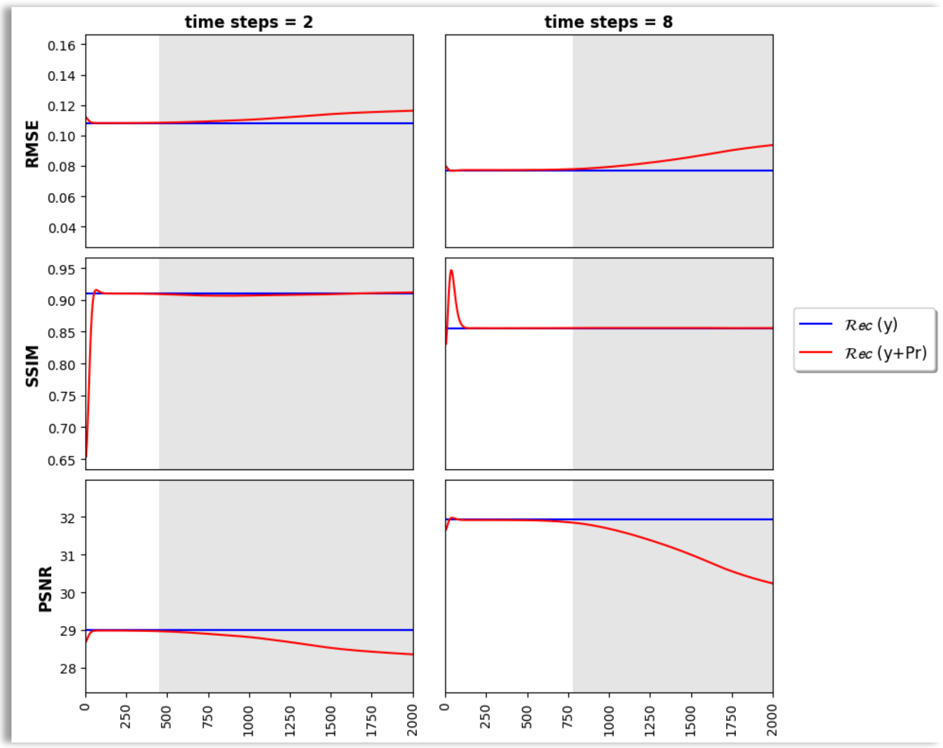


Figure 4.2.19: f = 64, λ = 1.5

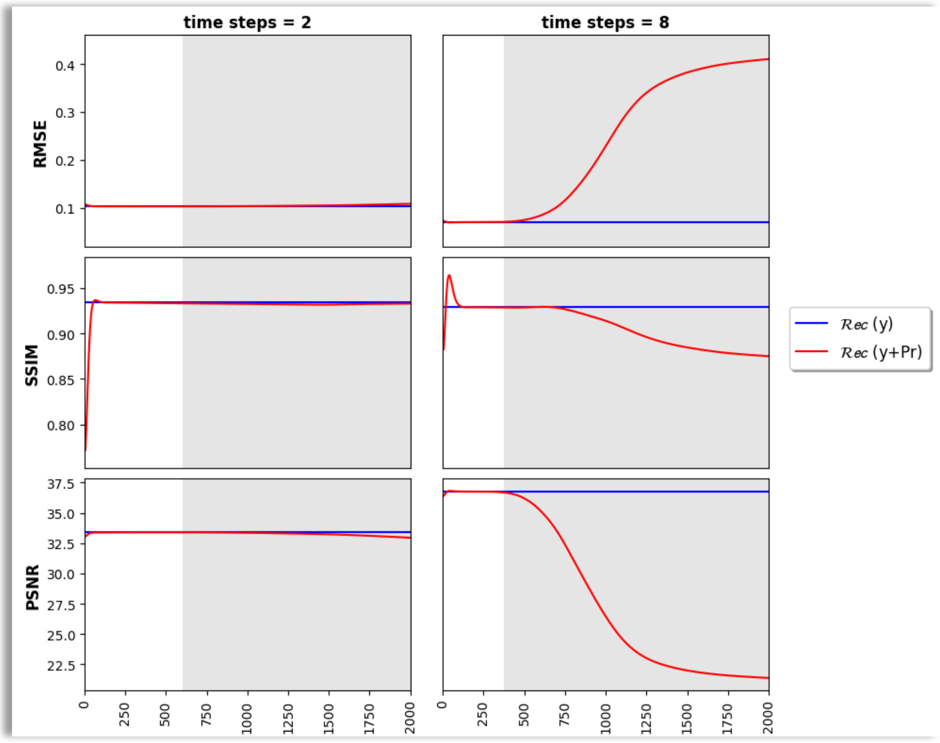


Figure 4.2.20: f = 128, λ = 1.5

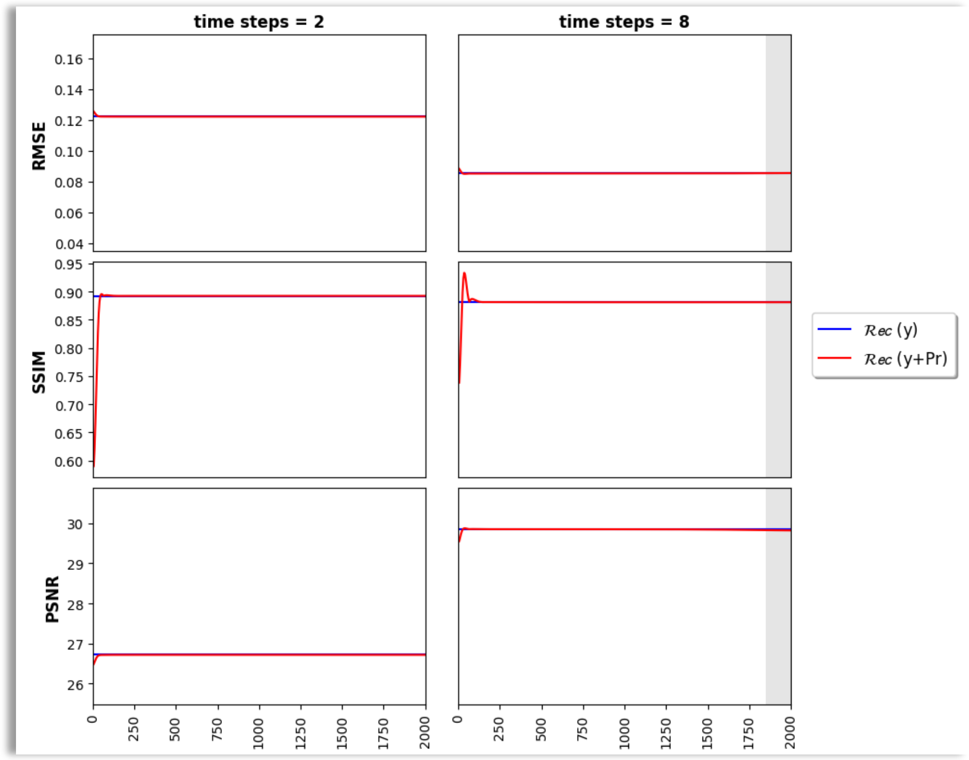


Figure 4.2.21: f = 32, λ = 2.0

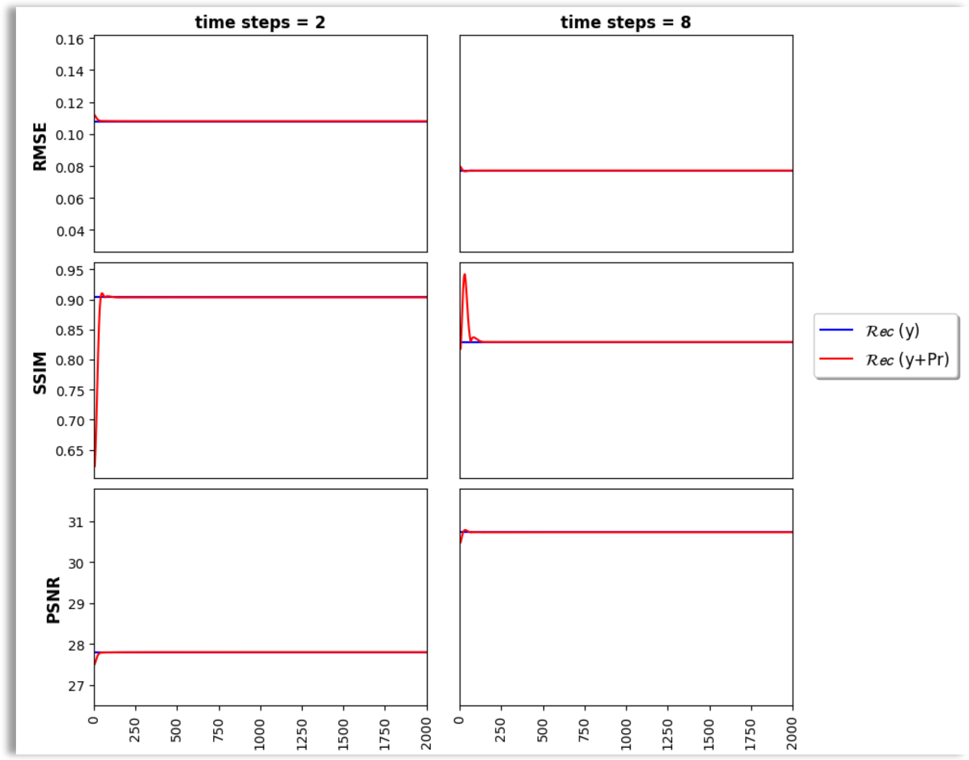


Figure 4.2.22: f = 64, λ = 2.0

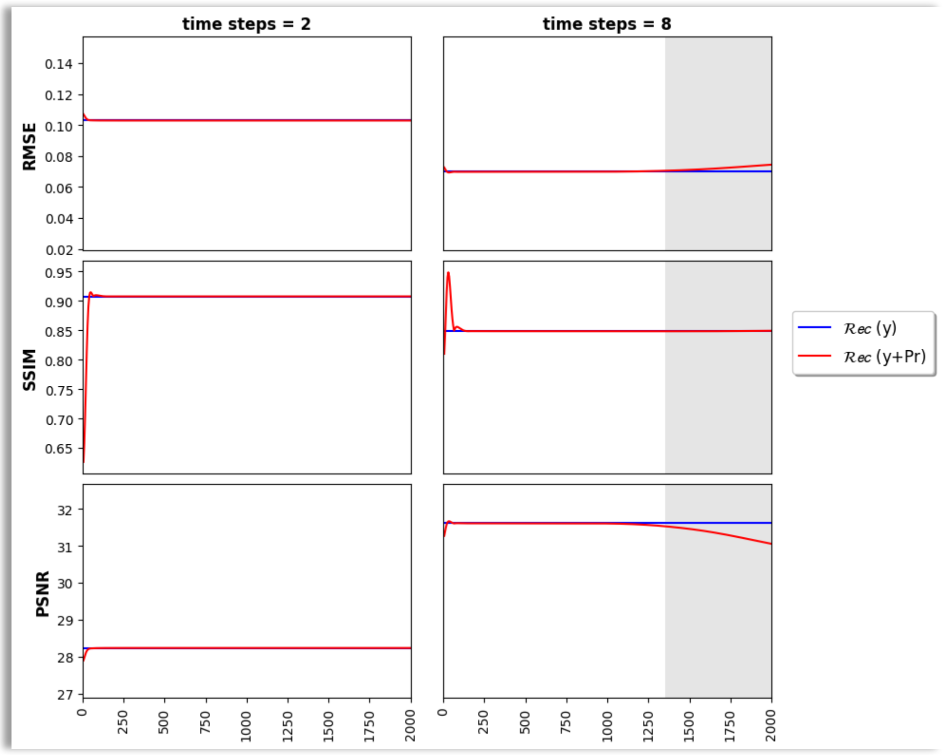


Figure 4.2.23: f = 128, λ = 2.0

Chapter 5

Discussion and future work

5.1 General summary

In this work, we tried to find adversarial examples for the RIM on the MRI reconstruction task, for different setups of the network's architecture and procedures (see §3.2). The purpose was to find a critical parameterization, where the network is unstable and provides a "poor" reconstruction process, under the presence of perturbations in the image. We were unable to find such a parameterization in any of the experiments, which means that our network is robust for all the parameter combinations tested.

We believe that the main reason why the RIM is so stable is its recurrent architecture, with the two internal states it maintains. The MRI-VN, which is used in [5] is not robust to the tiny perturbations. Although it is iterative and uses as input the log-likelihood $\nabla_{y|x_t}$, like the RIM does, it is not recurrent. Furthermore, the GRU cells in the RIM allow us to increase the depth of the network, while having fewer trained parameters than other networks which use only CNN layers.

In the subsection 4.2.1 we illustrate an example of what happens in different steps of the algorithm (see figures 4.2.1-4.2.8). It is shown that the image with the perturbations (top right), as well as the image with the difference between the reconstructions (bottom right) seem to be very close to each other, for all the iterations. This example is representative since the same fact occurs for every one of the 18 datasets we created. The difference between the reconstruction of the corrupted and the perturbed corrupted image starts to increase only at the point when the perturbation appears as non-negligible. The reconstructions' difference increases accordingly with the perturbation. This is more obvious in the plots of the errors (figures 4.2.9-4.2.23) where the 2 lines in the white area, after the first iterations, are either identical or very close each other. Thus, we were not able to find small perturbations that heavily affected image quality in the reconstruction.

The perturbations, as expected, grow faster as the number of the time-steps increases. By increasing the number of time-steps, we increase the chances for the error to propagate and grow into a bigger error.

The results on the number of feature maps seem to be inconclusive. In our experiments, we see that increasing the number of feature maps (fig. 4.2.9 - 4.2.14), when facing adversarial examples, does not always yield to better reconstructions (i.e. better metrics values). Also, the critical point (iteration) where the perturbations start to become visible does not seem to be related to the number of feature maps.

The range that we have empirically chosen for the regularization factor λ is considered to be appropriate in order to control the impact of the regularizer. Since we want to maximize $Q_y(r)$ we subtract instead of add the L2 penalty in the equation 3.2. When λ=1 the perturbations start to grow fast and be early visible in the image, while for λ=2 the perturbations grow either very slow or they do not grow at all.

5.2 On defining a global perturbation threshold

As referred in §4.1, the gray area on the plots defines a range of iterations where the perturbations are clearly visible on the corrupted image. Therefore, the left edge of the area is the threshold, above which, the perturbations are considered to be "too large".

The first aspect that we have to notice here is that the threshold variates for different number of feature maps and time-steps. This is reasonable, since the perturbations are developed in different rhythm and follow different distribution, according to the network parameters. The second aspect is that the magnitude of the perturbations is not a safe decision criterion for choosing an appropriate threshold θ. In figure 5.2.1, it is shown that a relatively small Frobenius norm of *r* may cause a bigger effect on the corrupted image than a bigger one. In the first case (top left), the perturbations are mostly concentrated in a small area of the image, such as they cause bigger local distortion on the corrupted image (top right). In contrast, in the second case (bottom left), the perturbations are more distributed, so that they do not cause that big local distortion on the corrupted image (bottom right), although their magnitude is bigger. Hence, we can assume that the distribution of the perturbations can play a more critical role than their magnitude.

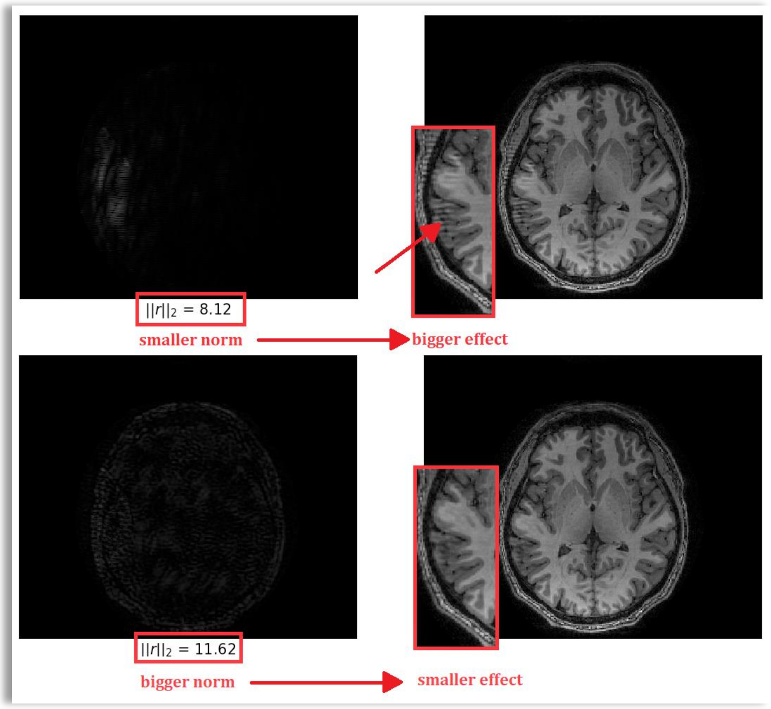


Figure 5.2.1: The inability to choose threshold for *r*, according to its magnitude.

Due to the aforementioned reasons and since the distribution of the perturbations relies on the inherent network's procedures, it is obvious that choosing an appropriate threshold is a matter that cannot be addressed using an objective, global criterion. In the plots of this work, the thresholds have been defined according to the author's opinion.

5.3 Future work

For the reconstructions that took place in this work, we used one slice of a T1-weighted brain volume. We noticed that, for the 18 different experiments that we made, it was not only the magnitude of the perturbations that was varying, but also their distribution in the image. In other words, the perturbations started to appear and grow in different areas of the image, according to each parameterization, and they did not follow a uniform pattern (see §5.2). This fact could be further investigated. An idea would be to keep the parameters of the algorithm in § 3.3 stable and run for different images, so that we might distinguish how the distribution of the perturbations relies on the model and how relies on the image.

Bibliography

[1] K.-S. Oh and K. Jung, “GPU implementation of neural networks,” *Pattern Recognit.*, vol. 37, no. 6, pp. 1311–1314, Jun. 2004.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[3] C. Szegedy *et al.*, “Going Deeper with Convolutions,” *ArXiv14094842 Cs*, Sep. 2014.

[4] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *ArXiv14091556 Cs*, Sep. 2014.

[5] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen, “On instabilities of deep learning in image reconstruction - Does AI come at a cost?,” *ArXiv190205300 Cs*, Feb. 2019.

[6] I. I. Rabi, J. R. Zacharias, S. Millman, and P. Kusch, “A New Method of Measuring Nuclear Magnetic Moment,” *Phys. Rev.*, vol. 53, no. 4, pp. 318–318, Feb. 1938.

[7] F. Bloch and W. W. Hansen, “The Nuclear Induction Experiment,” vol. 70, 1946.

[8] E. M. Purcell and H. C. Torrey, “Resonance Absorption by Nuclear Magnetic Moments in a Solid,” 1946.

[9] J. Larmor, “On the theory of the magnetic influence on spectra and on the radiation from moving ions,” 1897.

[10] C. Westbrook, “MRI at a Glance, Third Edition,” p. 138.

[11] C. E. Shannon, “Communication in the Presence of Noise,” *Proc. IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[12] D. Karkalousos, “Optimizing a Recurrent Inference Machine for Accelerated MRI Reconstruction Using Deep Learning,” University of West Attica/University of Limoges, 2018.

[13] B. Y. and C. A. Goodfellow I., *Deep Learning*, vol. 24, no. 1. 2017.

[14] Josef Hochreiter, “Diplomarbeit im Fach Informatik,” 2008.

[15] K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” *ArXiv14061078 Cs Stat*, Jun. 2014.

[16] P. Putzky and M. Welling, “Recurrent Inference Machines for Solving Inverse Problems,” no. Nips, 2017.

[17] K. Lønning, P. Putzky, J.-J. Sonke, L. Reneman, M. W. A. Caan, and M. Welling, "Recurrent inference machines for reconstructing heterogeneous MRI data," *Med. Image Anal.*, vol. 53, pp. 64–78, Apr. 2019.