University of Brasilia at Gama – FGA/UnB
Software Engineering

Generative Adversarial Network Prior Information for Improved Compressed Sensing Magnetic Resonance Image Reconstruction

Gabriel Gomes Ziegler

Advisor: Cristiano Jacques Miosso, PhD
Co-advisor: Davi Benevides Gusmão, MSc

UnB – University of Brasilia

FGA – Gama Faculty

Software Engineering

Gabriel Gomes Ziegler

Advisor: Cristiano Jacques Miosso, PhD

Co-advisor: Davi Benevides Gusmão, MSc

Bachelor Thesis

Software Engineering

Brasilia/DF, December 2020

UnB – University of Brasilia
FGA – Gama Faculty
Software Engineering

Gabriel Gomes Ziegler

Submitted in partial fulfillment for the Bachelor Degree in Software Engineering (Honor's Degree) to the department of Engineering of University of Brasilia

Approved by:

Cristiano Jacques Miosso, PhD

(Advisor)

Prof. José da Silva, PhD

Prof. João da Silva, PhD

Brasilia/DF, December 2020

Ficha Catalográfica

Ziegler, Gabriel
Título para a ficha,
[Distrito Federal], 2018.
xliip., 210 × 297 mm (FGA/UnB Gama, Bachelor of Engineering in Software Engineering, 2020).
Bachelor Thesis, UnB Gama Faculty, Software Engineering
1. Deep Learning 2. Magnetic Resonance Image Reconstruction
3. *Compressive Sensing* 4. Generative Adversarial Network
I. FGA UnB/UnB. II. Título (série)

Reference

Ziegler, Gabriel (2020). Título para a ficha (parte 2). Bachelor Thesis, Software Engineering, UnB Gama Faculty, University of Brasilia, Brasilia, DF, xliip.

Cessão de Direitos

Author: Gabriel Gomes Ziegler

Title: Título para a ficha (parte 2)

Degree: Bachelor of Engineering in Software Engineering

Year: 2020

É concedida à Universidade de Brasília permissão para reproduzir cópias desta monografia de conclusão de curso e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desta monografia pode ser reproduzida sem a autorização por escrito do autor.

---

gabrielziegler3@gmail.com

Brasília, DF – Brasil

Contents

List of Tables

List of Figures

Nomenclature and Abbreviations

MRI Magnetic Resonance Imaging xii

MR Magnetic Resonance xii

ANN Artificial Neural Networks viii

DL Deep Learning xiii

ML Machine Learning xii

GAN Generative Adversarial Network viii

CNN Convolutional Neural Network xiii

GPU Graphics Processing Units xiii

CV Computer Vision xiii

NLP Natural Language Processing xiii

CS Compressed Sensing xii

MLP Multilayer Perceptron xix

DFN Deep Feedforward Networks xix

SNR Signal-to-Noise Ratio ix

PSNR Peak Signal-to-Noise Ratio xxxiv

DCGAN Deep Convolutional GAN xx

ReLU Rectified linear units viii

SGD Stochastic Gradient Descent xxiv

1 Introduction

In this thesis, we propose a GAN approach for prior information extraction to feed a Compressed Sensing (CS) algorithm, aiming to reconstruct images with both reduced signal-to-noise error and less acquisition time compared to conventional CS. Achieving higher quality with a reduced number of samples allows faster exam procedures, making Magnetic Resonance Imaging (MRI) cheaper, faster and more convenient for both patients and clinics.

1.1 Context

MRI is a widely used imaging modality in medical practice because of its great tissue contrast capabilities, it has evolved into the richest and most versatile biomedical imaging technique today [10], making MRI the best option for medical imaging whenever it is possible to use.

However, like everything in life, there is a trade-off to consider when using MRI. Typically, reconstructing an MRI is an ill-posed linear inverse task (a problem that has either none or infinite solutions in the desired class). Problems of this nature impose a trade-off between *accuracy* and *speed* [11]. The information obtained from Magnetic Resonance (MR) is commonly represented by individual samples in the k-space, which translates to the Fourier transform of the image to be reconstructed [12]. This MR sampling sparse nature makes CS a liable technique to use when reconstructing MRI, hence we here propose a novel CS prior information approach for better results.

CS has been for years the state-of-art technique in MRI reconstruction and has been improved later by the use of sparsifying pre-filtering techniques and prior information [13, 14]. CS uses the premise that given a signal with a sparse representation in some known domain, it is possible to reconstruct the signal using limited linear measurements taken from a non-sparse representation.

Machine Learning (ML) methods have been utterly developed and improved recently

with the use of higher computing power derived from the invention of Graphics Processing Units (GPU) and other hardware improvements, allowing ANN to come to practicality. These ANN models, often referenced as Deep Learning (DL), have become the state-of-art in various areas, such as Computer Vision (CV), Natural Language Processing (NLP), Recommendation Systems, amongst other fields [15, 16, 17]. These fast-paced developments led to improvements in medical data processing using DL as well. ML techniques can be used in several different manners to improve medical analysis, here we focus on applying GAN in the process of attaining improved prior information to feed the CS algorithm obtaining higher signal-to-noise ratios and faster computation procedures.

1.2 Scientific Problem Definition and Proposal

MRI is great for high-quality tissue images, but there are some drawbacks: MRI exams are often very long and require the patient to be in a static position throughout the whole process, this makes the exam challenging for patients that have difficulties in keeping a still position for several minutes. Another intrinsic complication in MRI procedures is that it is nearly impossible to get images from moving tissues like a beating heart or flowing blood veins as that would require an enormous amount of samples, which with current technologies used in clinics is not viable. Algorithms that reconstruct MRI try to tackle this sampling issue by producing the best possible quality images for the least amount of samples collected, making the exams faster and less sample-dependent.

CS algorithms have been the state-of-art in MRI reconstruction for the past few years, and now with the advances of DL, new techniques are being produced taking advantages of how ANN are powerful in imaging processing, especially Convolutional Neural Network (CNN) and, more recently, GAN networks are becoming the new state-of-art techniques in several computer vision areas. Most CS contributions without the use of machine learning cite the usage of sparsifying pre-filtering techniques and prior information that have been proven to improve efficiency and achieve better reconstructions [14, 13]. More recently, the contributions in MRI reconstruction has been more focused in deep learning approaches, with some architectures using CS along ANNs [18, 19, 20, 21, 22, 23].

Prior information for CS can go from previous MRI frames and exams to even medical records. Prior information is normally generated by simplistic mathematical approaches like filtering and thresholding on the images. Besides the simpler technique applied, these information extraction procedures oftentimes is restricted to few frames and does not take into account the nature of organs and tissues structures, a feature that DL should be able to identify and use to generate better quality information. This means that there is a lot

of room for improvement towards prior information engineering techniques, as DL models have been proven superior in tasks of this nature.

Within this context, we propose a modern prior information engineering system with the usage of GAN, aiming for higher quality prior information to feed the CS and reducing the number of samples dependability. This will reduce the number of samples needed, making the MRI exams faster and, consequently, cheaper.

1.3 Objectives

1.3.1 General Objective

This thesis' goal is to develop an MR prior information system retriever based on GAN architecture to analyse if the quality of the prior information fed to CS algorithms can be improved, hence improving quality in reconstructed MRI and decreased necessity for larger sampling.

1.3.2 Specific Objective

In order to achieve the general objective described above, we have set the following specific goals:

- Implement direct and indirect CS MRI reconstruction algorithm and apply to *k-space* measurements.
- Evaluate CS MRI reconstructions with real image data.
- Implement a GAN for regular image generation with a known CV dataset.
- Implement a GAN architecture for prior information retrieval and train it against *k-space* measurements.
- Evaluate the use of GAN architecture for prior information retrieval against state-of-art prior information techniques.

2 Theory Foundation and State-of-Art

2.1 Magnetic Resonance Imagery

MRI is an indirect process that produces cross-sectional images with high spatial resolution from nuclear magnetic resonances, gradient fields and hydrogen atoms of the subject's anatomy [24]. The acquisition of these signals is performed by a measuring instrument called *receiver coil* and it can be done by using one receiver coil or in some cases with multiple coils [1, 2]. These receiver coils are placed in proximity to a specific region in the subject to be imaged. During the imaging process, the MRI machine generates a sequence of spatially and temporally-varying magnetic fields which induce the body to emit resonant electromagnetic response fields which are then measured by the receiver coil [1, 2].

2.1.1 K-space

The k-space is the output generated by the MRI machine scan after extracting measurements from a given subject tissue. The k-space is represented in the spatial frequency in two or three dimensions of a subject and may also be referred to as the Fourier space. This k-space representation contains an implicit sparsity that is exploited when performing undersampling [25] and reinforce the usage of algorithms like CS for MRI reconstruction as CS depends on signals that have a sparse representation in an orthonormal basis.

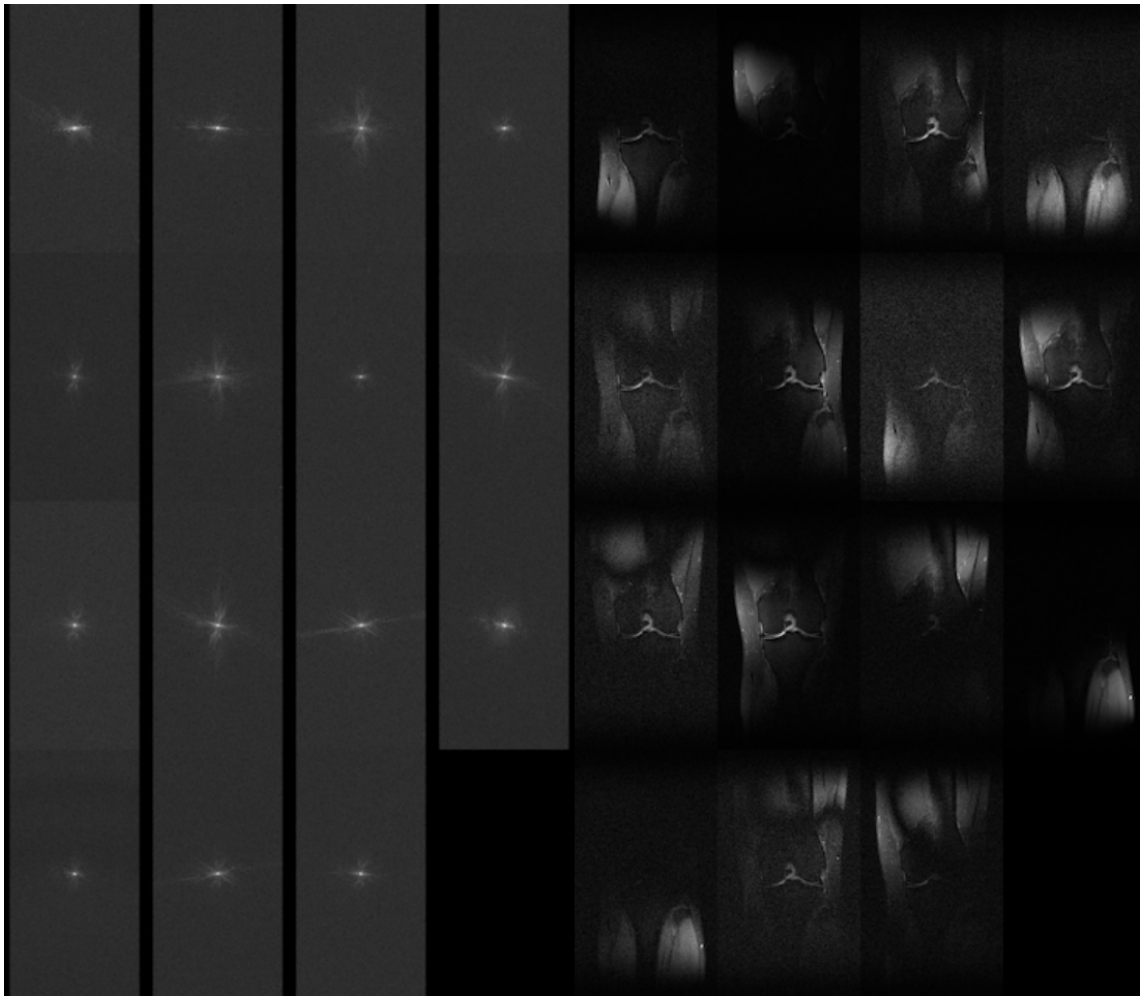

Figure 2.1. (a) FastMRI K-space data from 15 coils (b) FastMRI individual fully sampled coil spatial images [1, 2].

2.1.2 Sampling

The time required to acquire all the measurements responses from every single atom in a subject would be extremely high and problematic to everyone involved (patients, doctors and clinics). The way machines can do faster MRI is by performing *undersampling*, also referred as subsampling and sampling, when scanning the subject.

Undersampling is performed by giving the machine a known prescribed path in which it will extract measurements from the multidimensional k-space representation. This allows machines to collect only a fraction of data measurements needed for image reconstruction hence speeding up the data acquisition process without critical quality loss.

There are some undersampling patterns to use and each has its benefits depending

on several parameters, such as: the subject's region extraction, the algorithm used for reconstruction, acquisition time.

In the figure below we can see some of the most used patterns. In this research, we will focus mostly on the cartesian undersampling method, as that is the one used in the FastMRI dataset [1, 2], which we will use for our experiments.

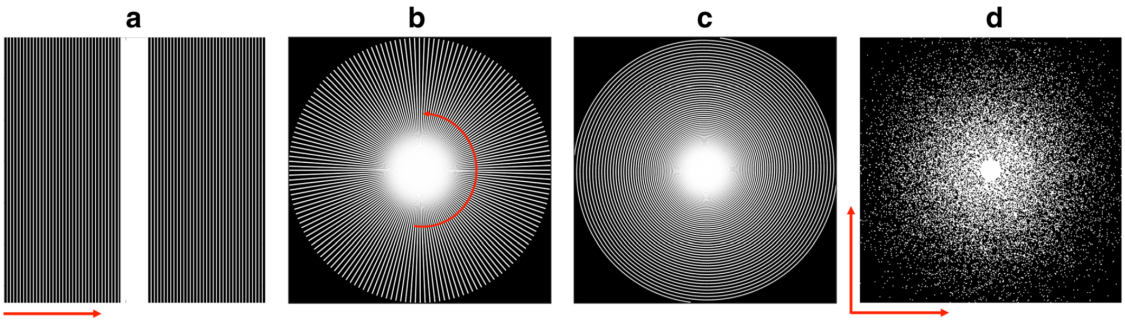

Figure 2.2. Under-sampling patterns. **(a) Cartesian undersampling**, **(b) radial undersampling**, **(c) spiral undersampling**, **(d) isolated samples in the k-space, according to the realisation of a random process** [3].

2.2 Compressed Sensing

CS is an extremely powerful algorithm that was introduced in 2004 proposing a novel technique for the acquisition of signals of sparse or compressible nature. CS has disrupted the signal processing field as it has broken the *Shannon's theorem*: the sampling signal rate must be at least twice the maximum frequency present in the signal (Nyquist rate). CS has been proven to sample the signal at a much lower rate than the Nyquist sampling rate. In MRI, when *k-space* is undersampled, the Nyquist criterion is violated [25].

The idea was inspired from questioning the necessity of extracting large portions of samples when much of these samples are discarded, exposing the inefficiency of trying to gather all signal.

> *"Why go to so much effort to acquire all the data when most of what we get will be thrown away? Can we not just directly measure the part that will not end up being thrown away?" [26]*

CS parts from the principle that if given x, a digital image or signal has a sparse representation in an orthonormal basis (e.g.wavelet, Fourier), then the N most important coefficients in that expasion allow reconstruction with l_2 error $O(N^{1/2-1/p})$ [26].

2.3 Artificial Neural Networks

2.3.1 Biological Inspirations

ANNs, as the name suggests, have been (loosely) inspired by biological neural networks (brains) from animals. The concept of using many layers of vector-valued representation is drawn from neuroscience. The choice of the functions $f^{(i)}(x)$ used to compute these representations is also loosely guided by neuroscientific observations about the functions that biological neurons compute [27]. Another trait they share is that just like the human brain can be trained to pass forward only meaningful signals to achieve larger goals of the brain, the neurons on a neural network can be trained to pass along only useful signal [5].

2.3.2 Neuron

The most basic unit in ANNs is the *artificial neuron*. Neurons act as feature detectors and this is one of the advantages of deep learning techniques in contrast to classical machine learning as the ANN is responsible for doing feature engineering and selection, and often outperform humans in this task.

These artificial neurons that are modelled mirroring the behaviour of the biological neuron as both of them are stimulated by inputs and carry some information they receive to other neurons. Artificial neurons take in inputs $x_1, x_2, \ldots, x_n$, each and multiply them by their respective weights $w_1, w_2, \ldots, w_n$. Then these weighted inputs are summed together producing the *logit* of the artificial neuron, $z = \sum_{i=0}^{n} w_i x_i + b$, with b being a constant number added called *bias*. After this, the logit is passed to a function f in order to generate the value $y = f(z)$.

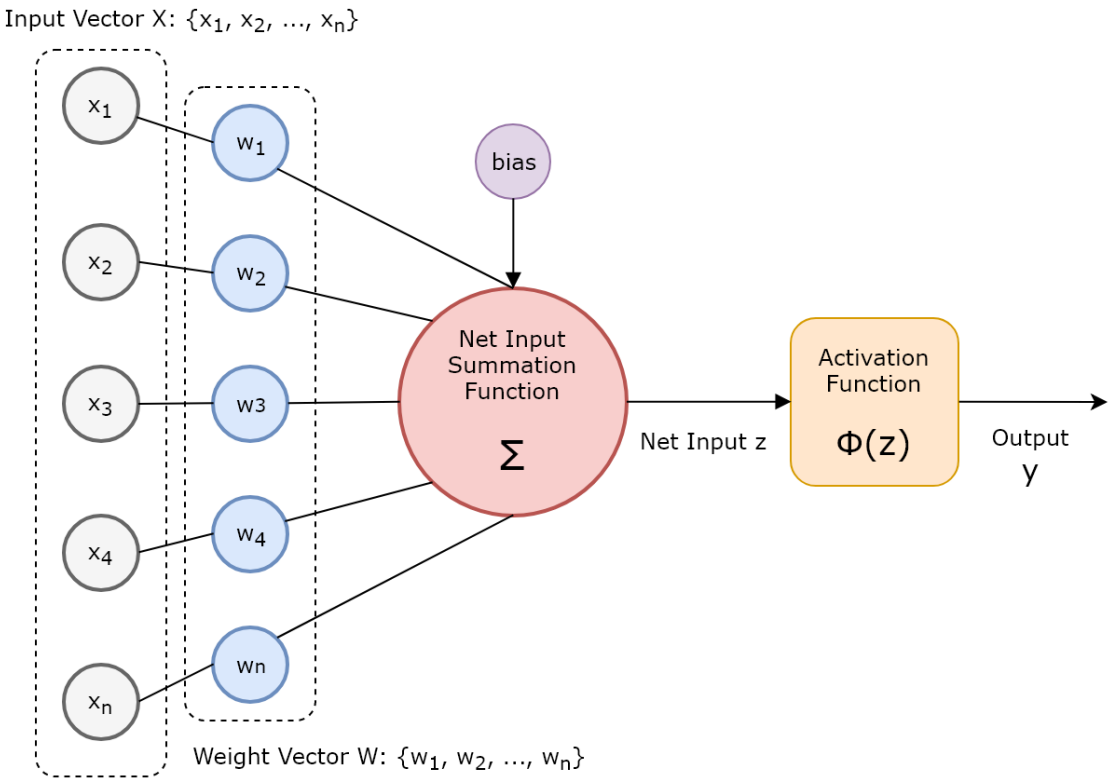

Figure 2.3. Schematic of an Artificial Neuron. Source: [4]

2.3.3 Multilayer Perceptron

Deep Feedforward Networks (DFN) or Multilayer Perceptron (MLP)s are a type of ANN very commonly used. It is the foundation of many famous architectures like CNNs. DFNs have an input layer followed by one or many hidden layers and a single output layer. Each layer is fully connected to the adjacent layer.

MLPs are computational models that flow information through the function f that evaluates x. The goal is to approximate some function $f*$. For instance, a classifier $y = f*(x)$ maps an input x to a category y. The feedforward defines a mapping $y = f(x;\theta)$ and learns the value of the parameters θ that result in the best function approximation [27].

The behaviour of an ANN is shaped by its architecture, which describes the number of units it should have and how these units connect to each other and how complex the model is. Often adding too much complexity to the network will lead to overfitting the training set, which occurs when the model shapes the training data too precisely and cannot generalise new data fed.

Most ANNs are organized into rows of neurons called layers. These layers are arranged

in a chain-like structure, with each layer being a function of the layer before it. These layers' goal is to extract *representations* out of the data fed and generalize what is meaningful towards minimizing the error rate. This architecture scheme is represented by the following equation, where i is the layer index:

$$h^{(i)} = g^{(i)}(W^{(i)T}x + b^{(i)})$$

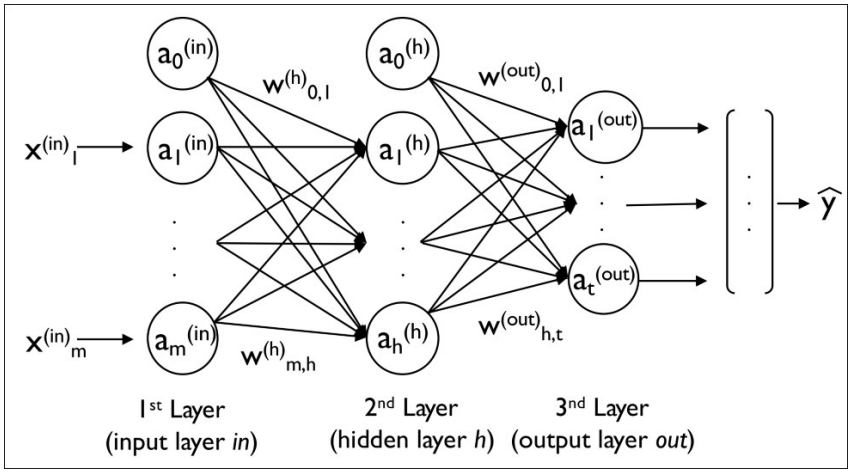

Figure 2.4. ANN Architecture Sample.

2.3.4 Activation Functions

Activation functions are a scalar-to-scalar function used to propagate the output of one layer's neurons forward to the next layer. There are several types of activation functions for different purposes and network architectures.

The Deep Convolutional GAN (DCGAN) architecture used in my MNIST experiment is built using ReLU activations in the generator network and a *tanh* in the output layer. The discriminator network uses LeakyReLU activations for all layers and a *sigmoid* function for the output layer. LeakyReLU activation functions [28, 29] have been proven to work well for higher resolution modelling [8] in contrast to the usage of maxout activation functions that were first proposed in the original GAN paper [30].

Name	Function	Derivative
Sigmoid	$\phi(x) = \frac{1}{1+e^{-x}}$	$\phi'(x) = \phi(x)(1-\phi(x))$
TanH	$\phi(x) = \frac{2}{1+e^{-2x}} - 1$	$\phi'(x) = 1 - \phi(x)^2$
ReLU	$\phi(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$	$\phi'(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$
Leaky ReLU	$\phi(x) = \begin{cases} \alpha x & x \leq 0 \\ x & x > 0 \end{cases}$	$\phi'(x) = \begin{cases} \alpha & x \leq 0 \\ 1 & x > 0 \end{cases}$

Table 2.1. Activation functions and respective derivatives.

Some of the most used and also required activation functions in the use of GANs and other widely used neural networks are described below.

2.3.4.1 ReLU

The ReLU [31] transform activates a node only if the input is above a certain threshold having a linear relationship with the dependent variable and outputs zero for every input below zero.

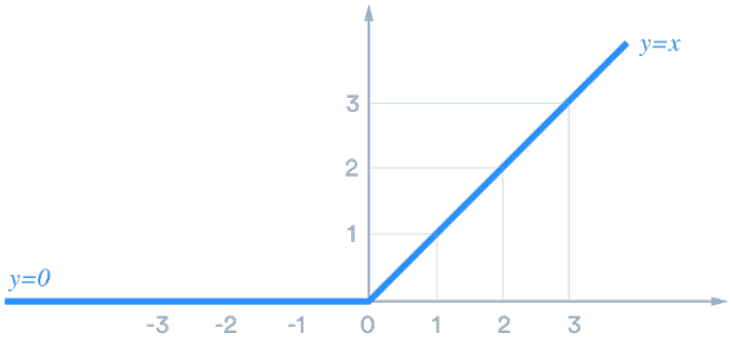

Figure 2.5. ReLU activation function

2.3.4.2 Leaky ReLU

ReLU activation functions have the "dying ReLU" problem, where a ReLU neuron is stuck in the negative side and always outputs 0 [32, 33]. This happens when the slope of ReLU in the negative range is also 0, once a neuron gets negative, it is unlikely for it to recover. These "dead" neurons are not playing any role in discriminating the input and

are essentially useless.

To mitigate this issue within ReLUs, LeakyReLUs are a strategy that opposed to having the function being zero when $x < 0$, it has instead a small negative slope (most times with $\alpha = 0.01$).

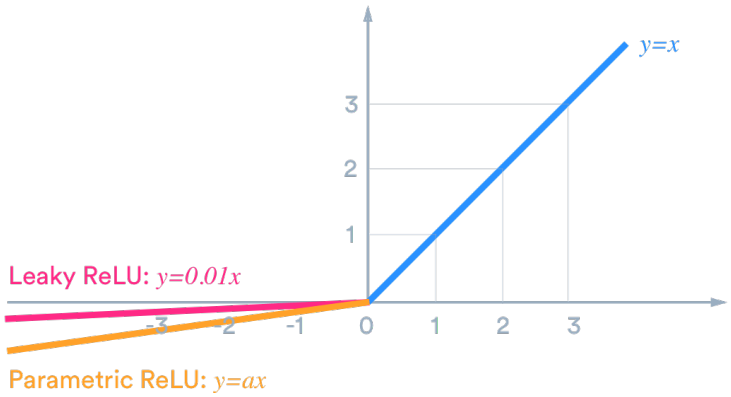

Figure 2.6. Leaky ReLU activation function

2.3.4.3 Tanh

Tanh is a hyperbolic trigonometric function that deals more easily with negative numbers [5]. Unlike the Sigmoid function, tanh ranges from -1 to 1.

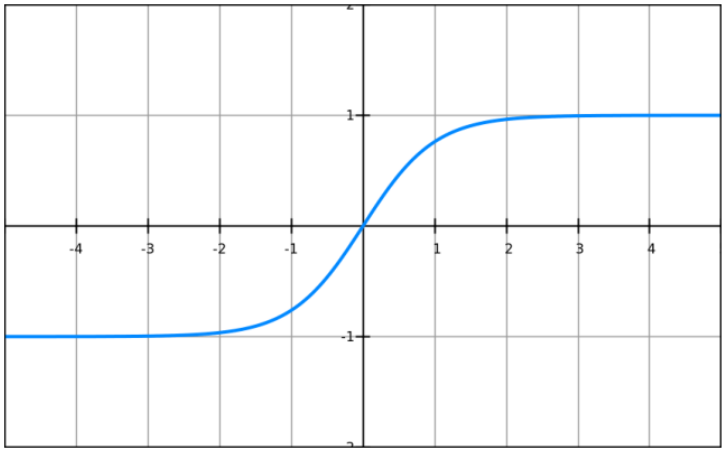

Figure 2.7. *Tanh* activation function. Source: [5]

2.3.4.4 Sigmoid

Sigmoids can reduce extreme values or outliers in data without removing them, framing the input from 0 to 1 and most outputs will be close to either 0 or 1.

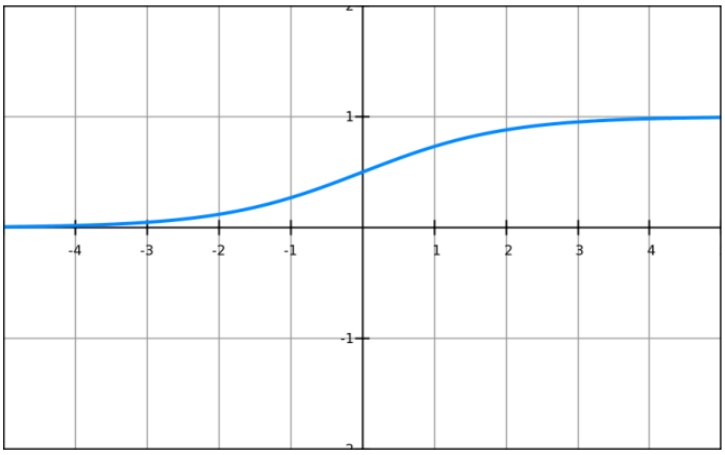

Figure 2.8. Sigmoid activation function. Source: [5]

2.3.5 Loss Functions

Loss functions are used to determine how a neural network is performing on the given data. A metric is calculated based on the error observed in the network's predictions and the model then tries to minimize this error in an optimization problem fashion.

Some of the most commonly used functions are described in the table below.

Mean squared error	MSE	=	$\frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	RMSE	=	$\sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	MAE	=	$\frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	MAPE	=	$\frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Table 2.2. Loss functions and formulas.

In the case of generative networks, the original GAN paper presents a loss function called *minmax* [30], that is described as.

$$min_G max_D V(D,G) = E_{x \sim P_{data}(x)}[logD(x)] + E_{z \sim pz(z)}[log(1 - D(G(z)))] \tag{2.1}$$

Where x is the input data representing an image, $D(x)$ is the discriminator network,

$G(x)$ is the generator function and z represents the latent vector that is mapped to data-space by G. Hence, the scalar probability that the output of the generator G is a real image is given by $D(G(z))$ [30].

2.3.6 Backpropagation

Backpropagation is a technique used to implement gradient descent in weight space for an MLP [34, 35]. In essence, backpropagation computes the error partial derivatives of an approximating function $F(w, x)$ computed by the ANN with respect to the weight and input vector for each training example [36].

The development of the backpropagation algorithm is a milestone in neural networks development and research as it made computationally efficient to train MLPs, thus confirming that ANNs research field was filled with potential in the mid-1980s.

In order to evaluate the derivatives of the function $F(w, x)$ with respect to all the elements in the weight vector w for an input vector $x = [x_1, x_2, \ldots, x_{m0}]^T$ for an MLP with layer $l = 2$, we have the following equation where φ is the activation function, w is the ordered weight vector and x is the input vector fed into the MLP [36]:

$$F(\mathbf{w}, \mathbf{x}) = \sum_{j=0}^{m_1} w_{oj} \varphi \left(\sum_{i=0}^{m_0} w_{ji} x_i \right) \tag{2.2}$$

2.3.7 Gradient Descent

Gradient descent is an optimization algorithm frequently used in ANNs to find the values of coefficients of a function that minimizes a cost function. Gradient descent can be very time consuming on large datasets due to the necessity of having a prediction for each instance in the training set. In scenarios where there is a large number of data instances, a variation of gradient descent called Stochastic Gradient Descent (SGD) can be used. SGD updates the coefficients for each training instance or batch instead of at the end after running through all the training set instances.

Most deep learning models are powered by the SGD and it can be visualized as the figure below demonstrates: the function starts in a random point in the loss function and after each iteration, the SGD calculates how it should adjust parameters in order to reach the minimal point in the loss function, hence moving towards the valley as illustrated.

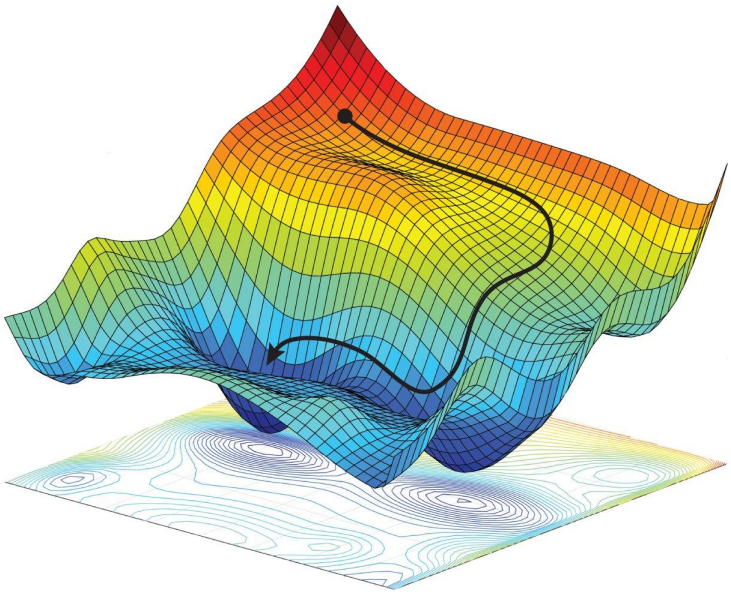

Figure 2.9. Gradient descent example visualization. Source: [6]

2.4 Generative Adversarial Networks

GANs are a machine learning strategy proposed in 2014 by Ian Goodfellow [30] that consists of two simultaneously trained models: the *Generator* $G(x)$ and the *Discriminator* $D(x)$. The generator has the role to generate fake data whilst the discriminator is trained to discern whether the given input is real or fake.

In essence, the generator takes a vector of random numbers (z) as input and outputs a fake example that strives to look as close as possible to the training data pattern. The discriminator takes an image (x) as input from two sources: real examples from the training set and fake examples generated by the generator network, then the discriminator outputs a scalar probability that the image is real [7].

GANs play a minimax two-player game in which D tries to maximize the probability to correctly classify real and fake samples ($\log D(x)$), whilst G tries to minimize the chance that D will correctly predict its generated outputs are fake ($\log(1 - D(G(x)))$).

Ideally, this minimax game would resolve to a solution with $pg = p_{data}$, where the discriminator is incapable of distinguishing real from fake inputs. However, GANs are still a novel neural network approach with its convergence theory is still being highly researched and hardly reaching this point in reality [30].

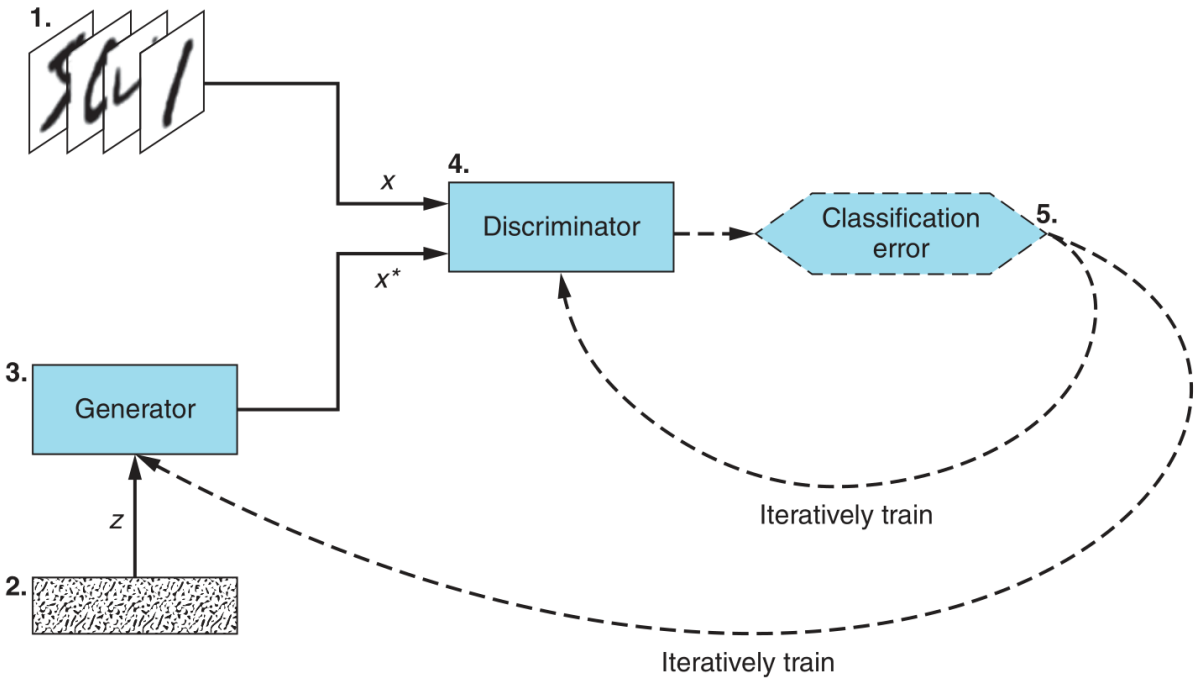

Figure 2.10. GAN training diagram. Source: [7]

3 Methodology

3.1 1-D Direct vs Indirect L1-Minimization

L1-minimization admits both direct and indirect approaches, in which there is the accuracy x resources trade-off. The direct method often produces a higher quality reconstruction but is very memory consuming, whilst the indirect method loses a little bit of quality, but requires much less memory to compute the equations system.

To visualize this trade-off, I have created 200 random 1-D arrays ranging values from the standard normal distribution and have taken 10% of data points randomly to reconstruct the whole signal using different L sizes.

The L variable denotes the The figure below displays the first 10 signals created.

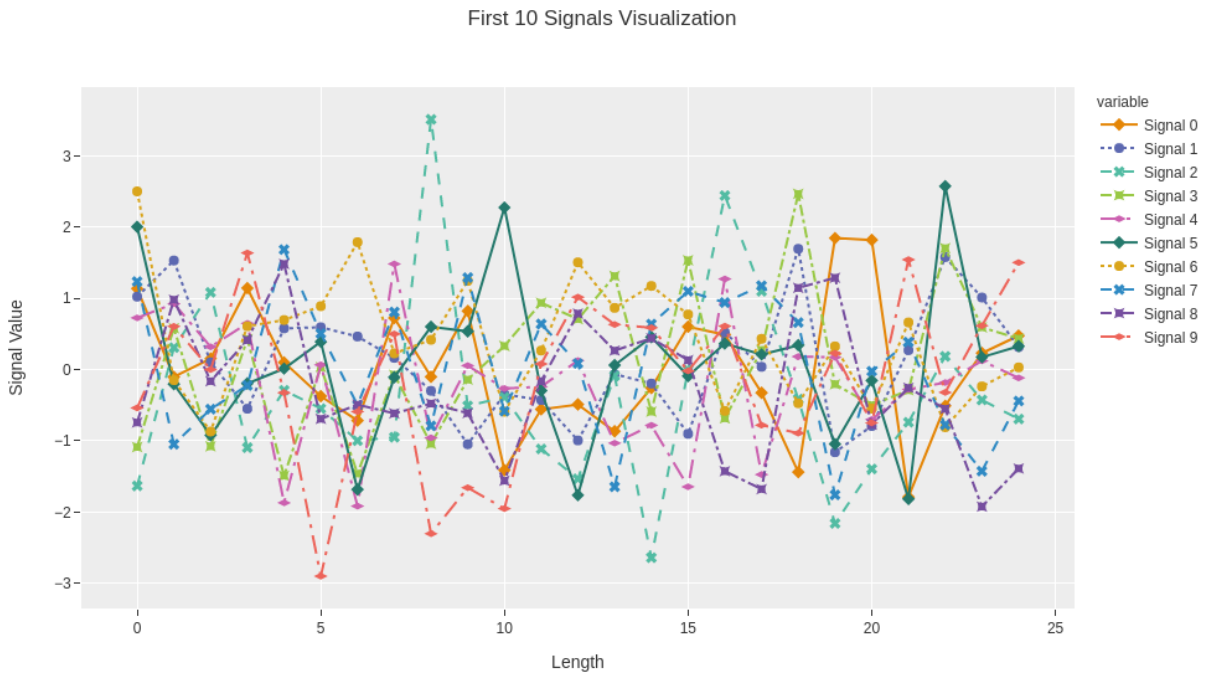

Figure 3.1. First 10 random 1-D signals

3.2 Phantom Compressed Sensing Reconstruction with Pre Filtered Signal

To experiment the sparsifying method of applying pre-filtering to the input signals in the k-space, I have conducted an experiment with the well-known Shepp-Logan phantom [37] to evaluate how the usage of sparsifying filters impact the total-variation minimization as it is known to improve the reconstruction in several scenarios [12, 13].

3.2.1 Subsampling

I then created a phantom image with dimension of 256×256, hence 65536 data points, using the phantominator python module. Then, I simulated an undersampled phantom image by applying the spiral undersampling pattern achieving approximately 30.95% of data points from the fourier space which resulted in a matrix with 20285 non zero elements.

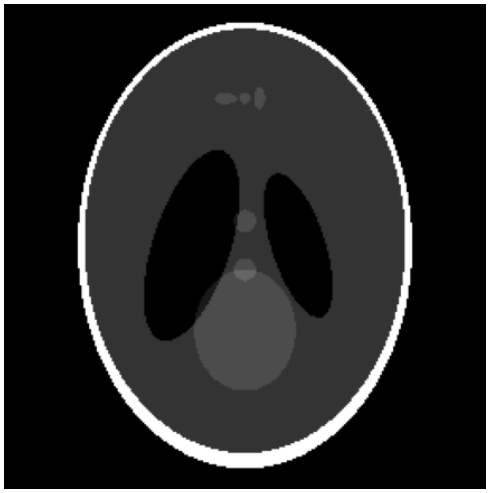

Figure 3.2. Shepp-Logan phantom reference image.

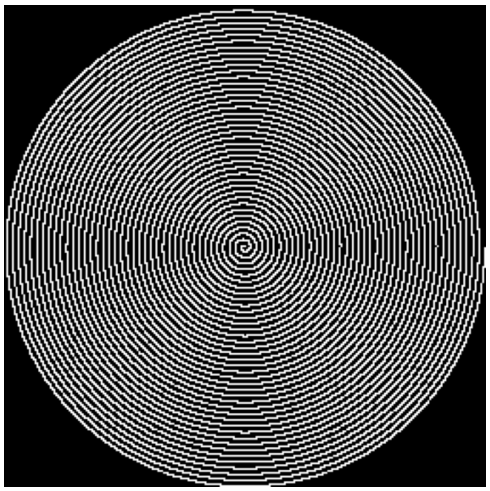

Figure 3.3. Spiral undersampling method with 30.95% data points.

3.2.2 Pre-filtering sparsifying transform

For the pre-filtering step, $f = 3$ is used where f is the number of filters applied to increase sparsity in the signal to be reconstructed. The filters are all 2×2 matrices and increase the sparsity in the signal from different perspectives, using more filters leverages the ability to sparsify the signal. The 3 filtered images are then composed into one single image containing the highest gain each filter could provide given a single pixel in the

image [13]. The different filters used can be better seeing in the figure below.

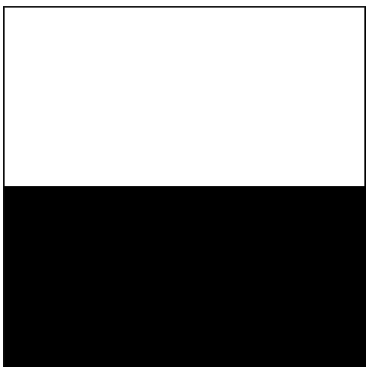

Figure 3.4. 2-D High pass horizontal filter.

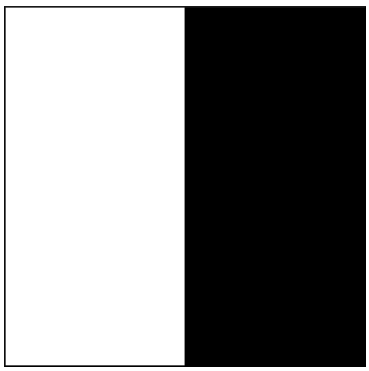

Figure 3.5. 2-D High pass vertical filter.

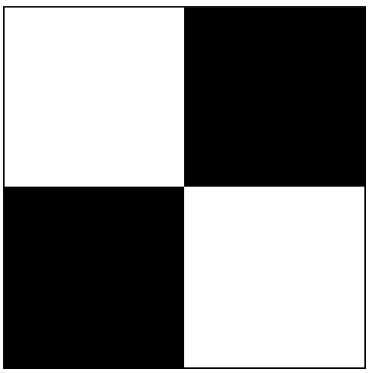

Figure 3.6. 2-D High pass diagonal filter.

The pre-filtering method is evaluated against the zero-fill reconstruction method (used as a dummy baseline) and an L1-minimization method without pre-filtering with the very same parameters used in the pre-filtering L1-minimization.

3.3 Preliminary Tests with Generative Adversarial Networks

In order to test the usage of GANs for data generation and in the future use it along with *prior information* for CS systems, I have developed a GAN capable of generating handwritten digits from 0 to 9 using the notable MNIST dataset. The MNIST dataset contains 60,000 examples for training and 10,000 examples for testing. The digits have been size-normalized and centred in a fixed-size image (28×28 pixels) with values from 0 to 9. For simplicity, each image has been flattened and converted into a 1-dimensional numpy array of 784 features (28×28).

The idea is to test if the neural network can output liable digits that look both readable (to the extent in which the MNIST dataset is) and also like it has been made by a human, just like the dataset itself.

Each MNIST image contains a 28×28 black and white image, like the following:

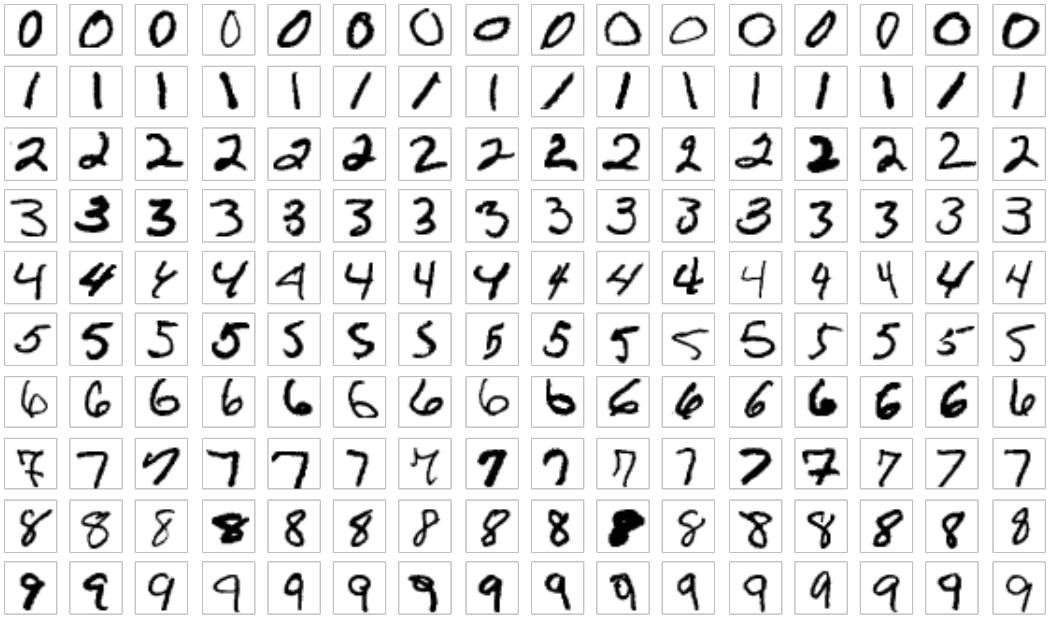

Figure 3.7. Sample of digits from MNIST

A DCGAN was used for the experiment. A DCGAN is an extension of the GAN, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator networks, respectively [8].

3.3.1 Data Transformation

Each input image used by the *dataloader* went through a computer-vision pre-processing step that includes:

- Grayscale transform: convert the image to greyscale. When loaded, the MNIST digits are in RGB format with three channels. Greyscale reduces these three to one.
- ToTensor: convert the image to a PyTorch Tensor, with dimensions (channels, height, width). This also rescales the pixel values, from integers between 0 and 255 to floats between 0.0 and 1.0.
- Normalize: scale and translate the pixel values from the range 0.0, 1.0 to -1.0, 1.0. The first argument is μ and the second argument is σ, and the function applied to each pixel is:

$$\rho \leftarrow \frac{(\rho - \mu)}{\sigma} \tag{3.1}$$

3.3.2 Generator Network Architecture

The generator network architecture is implemented using PyTorch as:

- A linear *fully-connected* module (or layer) to map the latent space to a $7 * 7 * 256 = 12544$-dimensional space that will later be undersampled several times until we reach $1 \times 28 \times 28$.
- An optional 1-dimensional batch normalization module
- A leaky ReLU module.
- A 2-dimensional convolutional layer with $padding = 2$, $stride = 1$ and 5×5 kernel (or filter).
- Two 2-dimensional transposed convolutional layers with $padding = 1$, $stride = 2$ and 4×4 kernel.
- Two optional 2-dimensional batch normalization modules after each 2-dimensional transposed convolutional layer.
- A *Tanh* activation function, rescaling the images to a $[-1, 1]$ range.

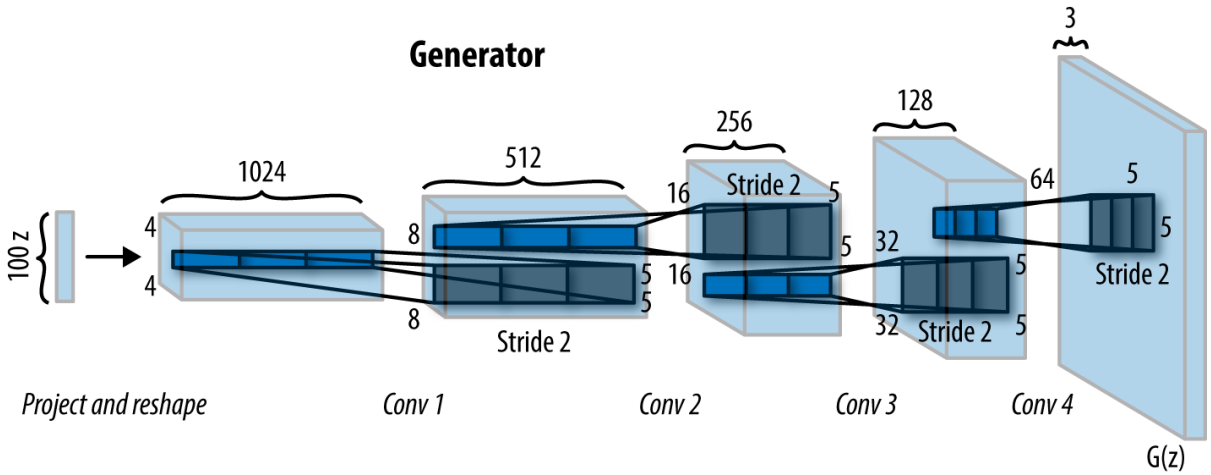

Figure 3.8. Generator network architecture of a 3-D image. Source: [8, 9]

The latent space (random signal) input goes through each layer being upscaled until it reaches the target image dimension 28×28 and then fed into the discriminator network.

3.3.3 Discriminator Network Architecture

The discriminator is a CNN-based image binary classifier network that takes an image as input and outputs a scalar probability that the given image is real or generated. The architecture is quite similar to the Generator network, except backwards. Here, the discriminator takes a $1 \times 28 \times 28$ input image, processes it through a series of convolutions, batch normalizations, and LeakyReLU layers, and outputs the final probability through a Sigmoid activation function.

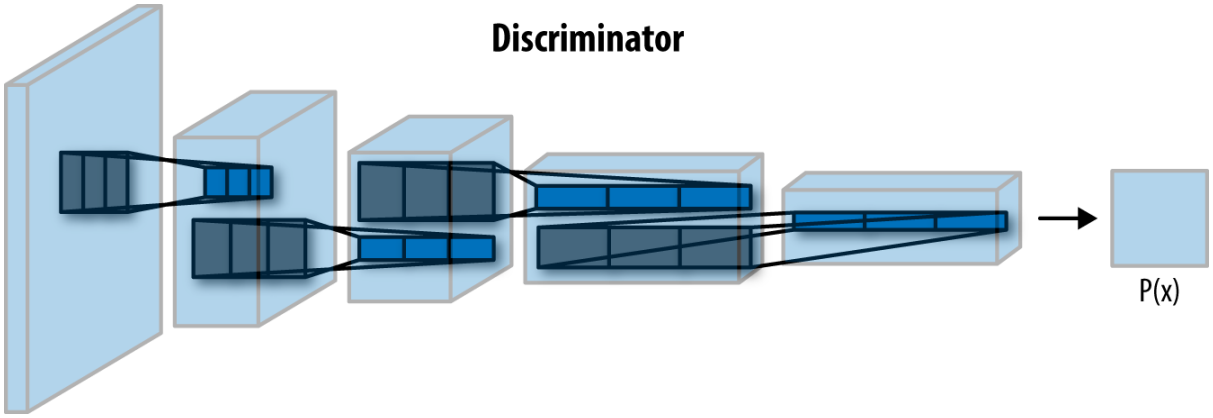

Figure 3.9. Discriminator network architecture of a 3-D image. Source: [8, 9]

4 Preliminary Results

4.1 1-D Compressed Sensing Reconstruction

The quality over resources demanded trade-off really starts making a big difference with L size around 200 samples, which is close to 80% of the data present in the signal to be reconstructed. This demonstrates that there is no big prejudice in using indirect reconstruction method for very undersampled signals (close to 10% of the data) such as the ones used in MRI

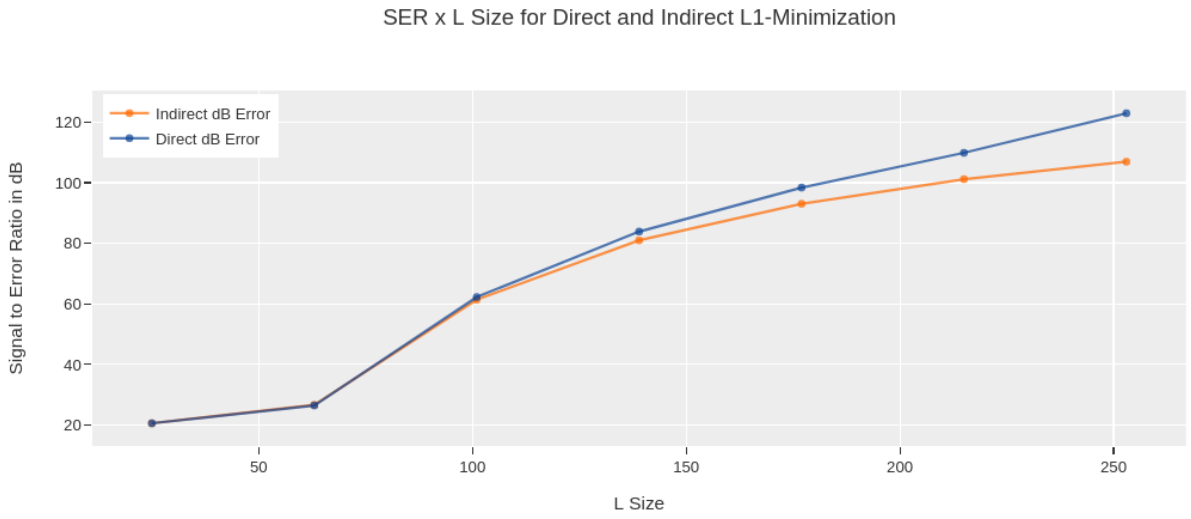

Figure 4.1. SNR x L size for direct and indirect L1-minimization for 200 random 1d signals

4.2 Compressed Sensing Reconstruction with Pre Filtered Signal

The results in the experiment show a huge gain of resolution in the pre-filtering method reconstructed image compared to the L1-minimization alone.

The zero-filled reconstruction (dummy baseline) was unable to reconstruct a high fidelity image and performed very poorly in the PSNR and SNR metrics. The L1-

minimization compressed sensing approach reconstructed the image with some noticeable noise artefacts, yet much better than the zero-filling approach. Finally, the L1-minimization along the usage of sparsifying pre-filtering delivered a great looking image without eye-catching artefacts and also increased the metrics hugely.

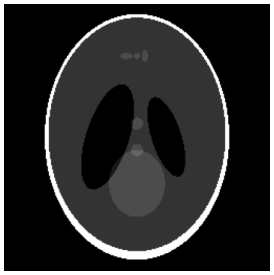

Figure 4.2. Phantom reference image

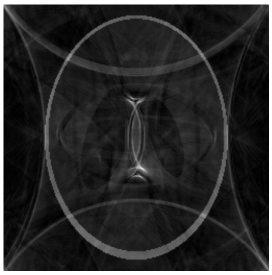

Figure 4.3. Phantom zero-filled reconstruction

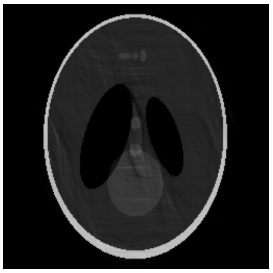

Figure 4.4. Phantom L1-minimization reconstruction

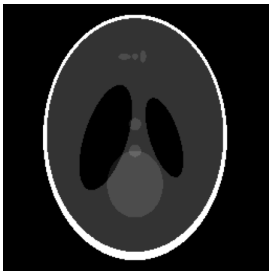

Figure 4.5. Phantom L1-minimization with pre-filtering reconstruction

The usage of L1-minimization certainly improves the MRI reconstruction, but the metrics reinforce how adding the pre-filtering step to preprocess the image achieves incredibly higher scores in Peak Signal-to-Noise Ratio (PSNR) and SNR.

	PSNR	**SSIM**	**SNR**	**MSE**
Zero-fill	22.41	0.36	4.22	0.02
L1-minimization	38.76	0.96	20.57	5.3e-5
Pre-filtering L1-minimization	91.89	0.99	73.70	2.5e-9

Table 4.1. Phantom reconstruction metrics.

4.2.1 Brain Sagittal Reconstruction

Another experiment done was using a reference sagittal head image of shape (256×256). The same spiral undersampling pattern with 30.95% data points used in the phantom experiment was used here for artificial undersampling.

This image poses a harder reconstruction challenge as it is filled with more details and more complex structures than the phantom. That said, it is clear that the undersampling pattern and amount of data points has not been sufficient to reconstruct a high fidelity image in any scenario, but the L1-minimization with pre-filtering reconstruction looks like the winner again, reinforcing the idea that pre-filtering is a good pre-processing strategy.

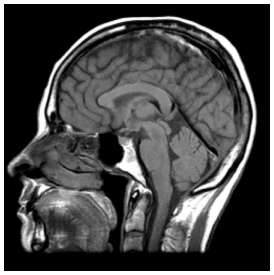

Figure 4.6. Sagittal head reference image

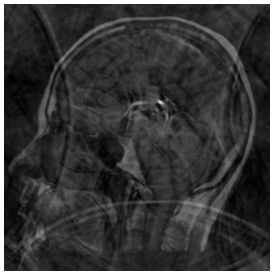

Figure 4.7. Sagittal head zero-filled reconstruction

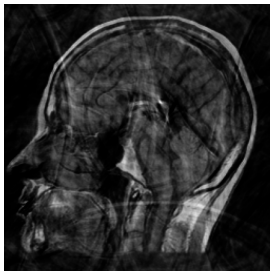

Figure 4.8. Sagittal head L1-minimization reconstruction

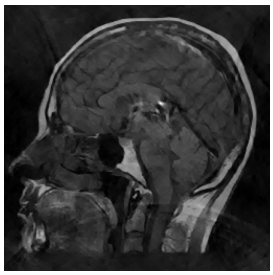

Figure 4.9. Sagittal head L1-minimization with pre-filtering reconstruction

The metrics evaluated were not affected as much as they were for the phantom experiment, but they were mostly improved by pre-filtering usage.

	PSNR	**SSIM**	**SNR**	**MSE**
Zero-fill	17.68	0.32	5.52	2392.57
L1-minimization	21.51	0.43	8.76	1134.88
Pre-filtering L1-minimization	21.03	0.45	8.97	1081.53

Table 4.2. Sagittal reconstruction metrics.

4.2.2 Knee Singlecoil Reconstruction

Figure 4.10. Singlecoil knee reference image

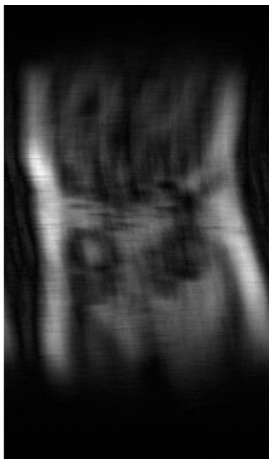

Figure 4.11. Knee zero-filled reconstruction

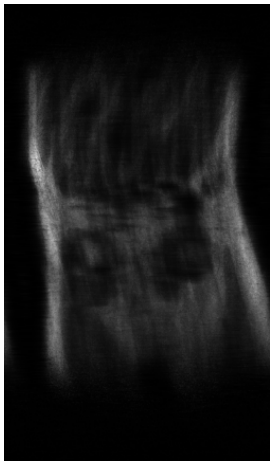

Figure 4.12. Knee L1-minimization reconstruction

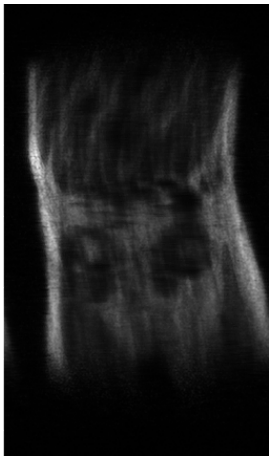

Figure 4.13. Knee L1-minimization with pre-filtering reconstruction

	PSNR	SSIM	SNR	MSE
Zero-fill				
L1-minimization				
Pre-filtering L1-minimization				

Table 4.3. Singlecoil knee reconstruction metrics.

4.3 Preliminary Tests with Generative Adversarial Networks

Both discriminator losses (fake and real) start very high and quickly decreases as the generator loss curve goes up in an invertedly correlated manner. This happens especially because the generator starts by tricking the discriminator network very easily as it is naïve to determine if an image is real or generated. Quickly the discriminator starts to detect how the data is disposed and manages to interpret the generated images are different from the training examples it is seeing.

This phenomenom exposes how bad the generator is in the first epochs and how easily the discriminator can distinguish between created and real. Then as the epochs go by and both networks get more sophisticated, the generator starts to get better at creating the

desired signal style and makes the discriminator's loss get higher again as it is observed in the loss curves below.

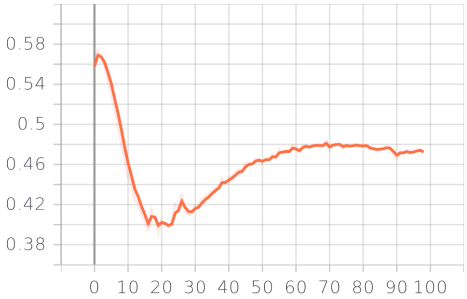

Figure 4.14. Discriminator fake loss over epochs

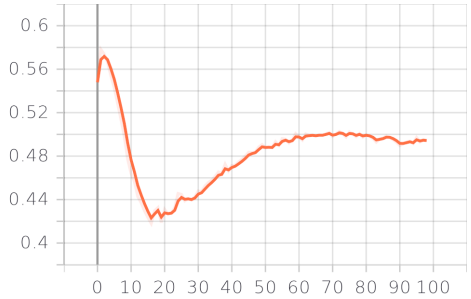

Figure 4.15. Discriminator real loss over epochs

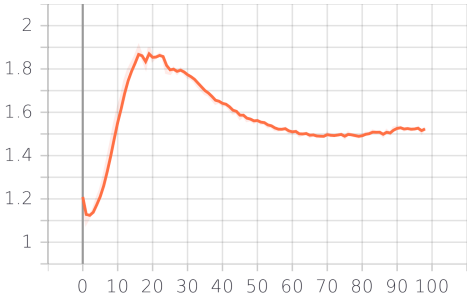

Figure 4.16. Generator loss over epochs

After 100 epochs, the DCGAN for MNIST number generation had an exceptionally good performance when the generated images are displayed. It is hard to tell if these are generated images or if they are part of the training set. The generated images sometimes have a bit more blur to them, but certainly with more epochs and more training samples this could be minimized.

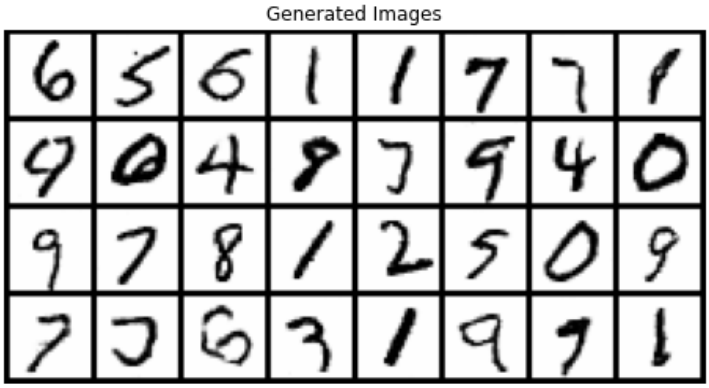

Figure 4.17. GAN generated MNIST digits

5 Conclusion

List of References

[1] Jure Zbontar, Florian Knoll, Anuroop Sriram, Tullie Murrell, Zhengnan Huang, Matthew J. Muckley, Aaron Defazio, Ruben Stern, Patricia Johnson, Mary Bruno, Marc Parente, Krzysztof J. Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdzal, Adriana Romero, Michael Rabbat, Pascal Vincent, Nafissa Yakubova, James Pinkerton, Duo Wang, Erich Owens, C. Lawrence Zitnick, Michael P. Recht, Daniel K. Sodickson, e Yvonne W. Lui. fastMRI: An Open Dataset and Benchmarks for Accelerated MRI. *arXiv:1811.08839 [physics, stat]*, December 2019. arXiv: 1811.08839.

[2] Florian Knoll, J. Zbontar, Anuroop Sriram, M. Muckley, M. Bruno, Aaron Defazio, Marc Parente, Krzysztof J. Geras, Joe Katsnelson, H. Chandarana, Zi-zhao Zhang, Michal Drozdzalv, A. Romero, M. Rabbat, Pascal Vincent, James T. Pinkerton, D. Wang, N. Yakubova, E. Owens, C. L. Zitnick, M. Recht, D. Sodickson, e Y. Lui. fastMRI: A Publicly Available Raw k-Space and DICOM Dataset of Knee Images for Accelerated MR Image Reconstruction Using Machine Learning. *Radiology. Artificial intelligence*, 2 1:e190007, 2020.

[3] Jong Chul Ye. Compressed sensing MRI: a review from signal processing perspective. *BMC Biomedical Engineering*, 1(1):8, December 2019.

[4] J. Quddus. *Machine Learning with Apache Spark Quick Start Guide: Uncover patterns, derive actionable insights, and learn from big data using MLlib.* Packt Publishing, 2018.

[5] Josh Patterson e Adam Gibson. *Deep Learning: A Practitioner's Approach.* O'Reilly, Beijing, 2017.

[6] Daniela Rus. Alexander Amini. Gradient descent relies on trial and error to optimize an algorithm, aiming for minima in a 3d landscape. adapted by m. atarod/science, 2020. Available at: https://www.sciencemag.org/news/2018/05/ai-researchers-allege-machine-learning-alchemy. Last access on November 30th, 2020.

[7] Jakub Langr e Vladimir Bok. *GANs in action: deep learning with generative adversarial networks*. Manning Publications, Shelter Island, New York, 2019. OCLC: on1050335878.

[8] Alec Radford, Luke Metz, e Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434 [cs]*, January 2016. arXiv: 1511.06434.

[9] Dominic Monn. Deep convolutional generative adversarial networks with tensorflow, 2017. Available at: https://www.oreilly.com/content/deep-convolutional-generative-adversarial-networks-with-tensorflow/. Last access on November 18th, 2020.

[10] R Nick Bryan. *Introduction to the Science of Medical Imaging*. Cambridge University Press, Cambridge, 2009.

[11] S. I. Kabanikhin. Definitions and examples of inverse and ill-posed problems. *Journal of Inverse and Ill-posed Problems*, 16(4), January 2008.

[12] Cristiano Jacques Miosso. *Compressive Sensing with Prior Information Applied to Magnetic Resonance Imaging*. PhD Thesis, Department of Electrical and Computer Engineering, University of Texas at El Paso (UTEP).

[13] C. J. Miosso, R. von Borries, e J. H. Pierluissi. Compressive sensing method for improved reconstruction of gradient-sparse magnetic resonance images. In *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, pages 799–806, Pacific Grove, CA, USA, 2009. IEEE.

[14] C.J. Miosso, R. von Borries, M. Argaez, L. Velazquez, C. Quintero, e C.M. Potes. Compressive Sensing Reconstruction With Prior Information by Iteratively Reweighted Least-Squares. *IEEE Transactions on Signal Processing*, 57(6):2424–2431, June 2009.

[15] Li Wan, Matthew Zeiler, Sixin Zhang, e Yann LeCun. Regularization of Neural Networks using DropConnect. page 12.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, e Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[17] Daeryong Kim e Bongwon Suh. Enhancing VAEs for collaborative filtering: flexible priors & gating mechanisms. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 403–407, Copenhagen Denmark, September 2019. ACM.

[18] yan yang, Jian Sun, Huibin Li, e Zongben Xu. Deep ADMM-Net for Compressive Sensing MRI. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, e R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 10–18. Curran Associates, Inc., 2016.

[19] Guang Yang, Simiao Yu, Hao Dong, Greg Slabaugh, Pier Luigi Dragotti, Xujiong Ye, Fangde Liu, Simon Arridge, Jennifer Keegan, Yike Guo, e David Firmin. DAGAN: Deep De-Aliasing Generative Adversarial Networks for Fast Compressed Sensing MRI Reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1310–1321, June 2018.

[20] Morteza Mardani, Enhao Gong, Joseph Y. Cheng, Shreyas S. Vasanawala, Greg Zaharchuk, Lei Xing, e John M. Pauly. Deep Generative Adversarial Neural Networks for Compressive Sensing MRI. *IEEE Transactions on Medical Imaging*, 38(1):167–179, January 2019.

[21] Dong Liang, Jing Cheng, Ziwen Ke, e Leslie Ying. Deep MRI Reconstruction: Unrolled Optimization Algorithms Meet Neural Networks. *arXiv:1907.11711 [physics, stat]*, July 2019.

[22] Elizabeth K Cole, John M Pauly, Shreyas S Vasanawala, e Frank Ong. Unsupervised MRI Reconstruction with Generative Adversarial Networks. page 8, 2020.

[23] D. Liang, J. Cheng, Z. Ke, e L. Ying. Deep magnetic resonance image reconstruction: Inverse problems meet neural networks. *IEEE Signal Processing Magazine*, 37(1):141–151, 2020.

[24] P. C. Lauterbur. Image Formation by Induced Local Interactions: Examples Employing Nuclear Magnetic Resonance. *Nature*, 242(5394):190–191, March 1973.

[25] Michael Lustig, David Donoho, e John M. Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, December 2007.

[26] D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, April 2006.

[27] Ian Goodfellow, Yoshua Bengio, e Aaron Courville. *Deep Learning*. MIT Press, 2016.

[28] Andrew L Maas, Awni Y Hannun, e Andrew Y Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. page 6.

[29] Bing Xu, Naiyan Wang, Tianqi Chen, e Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, e Yoshua Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, e K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[31] Richard H. R. Hahnloser, Rahul Sarpeshkar, Misha A. Mahowald, Rodney J. Douglas, e H. Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, June 2000.

[32] L. Trottier, P. Giguere, e B. Chaib-draa. Parametric exponential linear unit for deep convolutional neural networks. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 207–214, 2017.

[33] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018.

[34] David E. Rumelhart, Geoffrey E. Hinton, e Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986.

[35] Paul John Werbos. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. Wiley-Interscience, USA, 1994.

[36] Simon S. Haykin e Simon S. Haykin. *Neural networks and learning machines*. Prentice Hall, New York, 3rd ed edition, 2009. OCLC: ocn237325326.

[37] L. A. Shepp e B. F. Logan. The Fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science*, 21(3):21–43, 1974.