RESEARCH AND IMPLEMENTATION OF COMPUTER-SENSED BRAIN MRI RECONSTRUCTION

A thesis submitted to the Indian Institute of Technology Kharagpur for Master term Project By

Sayantan Kirtaniya (18EC33002)

Under the guidance of

Professor Debashis Sen

(Electronics and Electrical Communication Engg. Department)

Department of Electrical and Electronics Engineering
Indian Institute of Technology, Kharagpur

Department of Electrical and Electronics Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India- 721302

CERTIFICATE

This is to certify that the work presented in this thesis entitled to **research and implementation of computer sensed brain MRI reconstruction** has been carried out by Sayantan Kirtaniya under my supervision. This is his bonafide work and is worthy of consideration for a master-term thesis of the project of the Institute.

Prof. Debashis Sen
Department of Electronics and Electrical
Communication Engg.
Indian Institute of Technology Kharagpur
India - 721302.

DECLARATION

I hereby confirm that:

- The work presented in this thesis is the result of my original research conducted with the guidance and support of my supervisor.
- I have not submitted this work to any other institution for any degree or diploma.
- I have complied with all ethical norms and guidelines while preparing this thesis.
- I have strictly followed the guidelines provided by the Institute to complete this thesis.
- Whenever necessary, I have obtained the required permission from the copyright owners of the sources used.
- I have given appropriate credit to the authors and papers from which I have used materials, such as data, models, figures, and text, by citing them correctly. The references section includes all the details of the citations.

Sayantan Kirtaniya

ACKNOWLEDGEMENT

I would like to express my gratitude to several people who have played an instrumental role in the successful completion of this project. First and foremost, I extend my heartfelt appreciation to my supervisor, Professor Debashis Sen, whose constant guidance and valuable suggestions have been invaluable in shaping my ideas and presenting my work. I would also like to extend my special thanks to Professor Saumik Bhattacharya and Professor Prabir Kumar Biswas, who have provided me with their unwavering support and resources throughout the project. Their assistance has helped me gain a better understanding of the problem statement and implement my ideas effectively.

Moreover, I would like to mention that this project is an extension of the paper titled "Structure Preserving Compressive Sensing MRI Reconstruction using Generative Adversarial Networks." The paper has been highly influential in shaping our research and implementation strategies.

I would also like to express my appreciation to Mr Puneesh Deora, who has provided an open-source code repository for the paper. We have taken help from his repository in our implementation.

Overall, I am indebted to all the people who have contributed to this project in various ways and made it a success.

ABSTRACT

Our project delved into the fascinating world of compressed sensing, which has significant implications in both the medical and deep learning domains. Through extensive research, we gained a comprehensive understanding of the various models implemented over the years and the different metrics and loss functions used for reconstruction. We focused primarily on brain MRI data from the widely used MICCAI dataset and designed a model that would improve the quality of reconstruction. Our experiments involved testing various optimizers and activation functions to identify the optimal combination that would achieve the best results. The insights and knowledge gained from our project will prove invaluable in advancing the use of compressed sensing technology in the medical field and beyond.

Keywords: Wasserstein loss, Compressed Sensing, mSSIM, Patch-based discriminator, GAN, Transformer-based neural network, CNN, PSNR, wavelet-based loss

CONTENTS

Chapter 1

Introduction

In this project, we have worked with **magnetic resource images**, a non-invasive imaging technology used as a diagnostic tool in modern medicine. There are other uses of this technology for example in CT (computed tomography), US (ultrasound) and PAI (photoacoustic resource imaging).

1.1. The idea of reconstruction for MRIs

In the project, we will work with the **GANs** model for the reconstruction of Brain MRIs. Before starting the project, we need to understand the need to implement the models of reconstruction and betterment of them, so we will start with the drawbacks, the most significant drawback of magnetic resource image reconstruction is the long acquisition time of data, so in the process of accusation of data, it is acquired sequentially in the **K-space**, in which the spatial-frequency information gets contained [1].

Later on, a different approach has been used for solving this problem with optimization for reconstruction as well as denoising the image using a **block matching 3D model** [2], which used non-local means methodology for noise reduction, but these frameworks suffer a long computation time during the process.

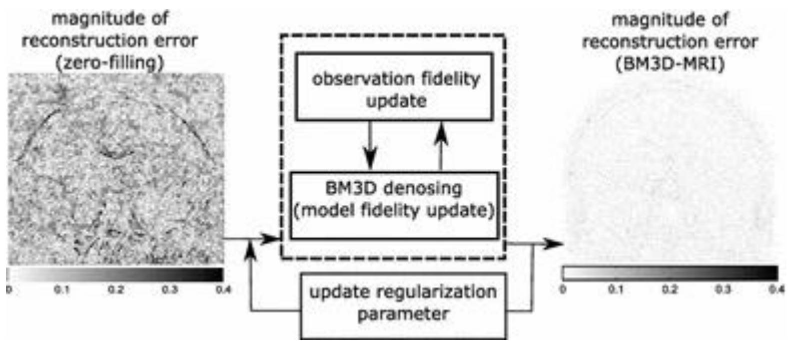
magnitude of reconstruction error (zero-filling)
0 0.1 0.2 0.3 0.4
observation fidelity update
BM3D denosing (model fidelity update)
update regularization parameter
magnitude of reconstruction error (BM3D-MRI)
0 0.1 0.2 0.3 0.4

Fig 1.1: Architecture of block matching 3D model [2]

1.2. How deep learning is helping this domain?

The GAN model helps to extract the prior information which will be required for the reconstruction. Refinement learning is also used in this idea, for example, **conditional GAN framework (DAGAN)** [3], but the paper we have worked on gave better results than all the older methods that had been used.

Previous methods in compressed sensing have utilized loss functions such as $\ell 2$ or $\ell 1/\ell 2$ norm of **pixel-wise difference minimization**, but these approaches suffer from a higher peak-to-signal ratio in the reconstructed image. One of the main drawbacks of these methods is their inability to ensure good reconstruction in the details of the pixel structure in the K-space.

To address this issue, some researchers have proposed the use of the $\ell 1$ norm of the p**atch-to-patchwise difference** as a **loss function**. This approach has shown promise in reconstructing data in high-frequency cases and improving the quality of the reconstruction.

Also, we have observed the implementation of **SSIM (structural similarity-based loss function)** [4] for the preservation of the structural details in the generator of the GAN model, as well as used **wavelet-based loss** with it to increase the performance.

Chapter 2

Literature Review

2.1. Self-supervised Models

The effectiveness of a model for reconstructing MRI images can be highly dependent on the specific domain, as demonstrated by the research of M. Darestani in 2022 [5]. The same model may work well for knee MRI but not for brain MRI, and vice versa.

To overcome this limitation, a domain adaptation-based method was proposed. This approach relies on self-supervision during training and testing inference to adapt the model to the specific domain and achieve better reconstruction results. The research highlights the importance of developing models that are tailored to the specific characteristics and requirements of each domain, rather than relying on a one-size-fits-all approach.

2.2. Transformer-based Neural Networks

The works of M. Lorenzana in 2022 proposed a transformer-based compressed sensing method, which involves breaking images into patch-based tokens using ViT transformers [6].

In this approach, the model's convolutional components were replaced by Transformer-based Neural Networks (TNN) to enable better image reconstruction. This method demonstrates the potential of using transformer-based architectures in compressed sensing and highlights the importance of continually exploring and integrating new technologies to improve image reconstruction techniques.

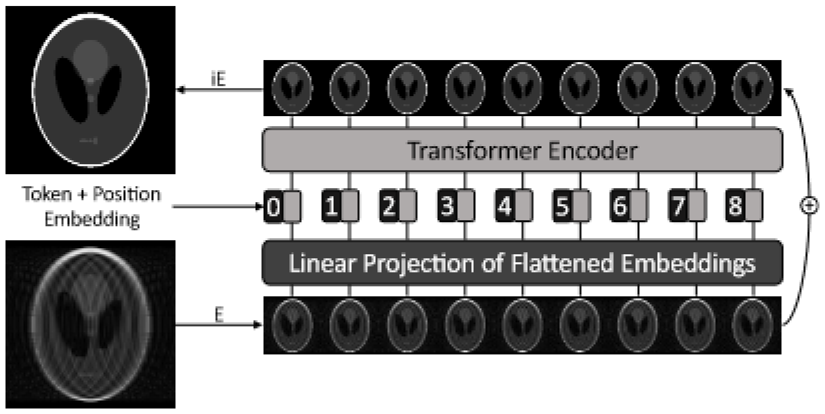
iE
Transformer Encoder
Token + Position Embedding
0
1
2
3
4
5
6
7
8
Linear Projection of Flattened Embeddings
E

Fig 2.1: Architecture of DcTNN model [6]

2.3. Feature-Space Optimization-Inspired Network

W. Chen proposed a novel method called FSOINet (Feature-Space Optimization-Inspired Network) for image compressive sensing in 2022 [7].

The experiments conducted on this method demonstrated superior performance compared to existing state-of-the-art (SOTA) models. FSOINet leverages feature space optimization to improve the accuracy and speed of image reconstruction, highlighting the potential of new optimization techniques in improving the quality of compressed sensing.

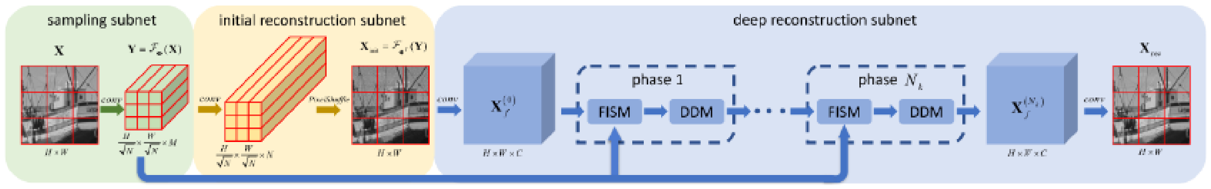
sampling subnet
initial reconstruction subnet
deep reconstruction subnet
X
$Y = \mathcal{F}_\Phi(X)$
$X_{init} = \mathcal{F}_{\Psi}(Y)$
conv
PixelShuffle
$H \times W$
$\frac{H}{\sqrt{N}} \times \frac{W}{\sqrt{N}} \times M$
$\frac{H}{\sqrt{N}} \times \frac{W}{\sqrt{N}} \times N$
$X_f^{(0)}$
$H \times W \times C$
phase 1
phase N_k
FISM
DDM
$X_f^{(N_k)}$
X_{rec}

Fig 2.2: Architecture of Feature-Space Optimization-Inspired Network model [7]

2.4. Reconstruction using the GAN model

For research understanding, we have understood the works of P. Deora [8], so we are going to analyze the process step by step below.

2.4.1. Acquisition model for the CS-MRI reconstruction

$$\boxed{u = Gy + \eta}$$

u = *Observation vector*

η = *noise vector*

G = *random sampling in the K-Space*

Which is a product of two matrices/functions:

- F which computes the FT.
- U which is for upsampling.

Here η is considered a nonzero vector, for a given input vector we need to solve the **reconstruction problem** for y, and we solve it with a **GAN model.**

2.4.2. The model

In used conditional GAN (CGAN) [9] there is a Discriminator: D and a Generator: G tries to fool D, it transforms the input vector to the distribution of true data And D tries to distinguish samples of true data from the generated samples of G.

Training loss: In place of Binary cross entropy, **Wasserstein loss is** used. For the **stabilization process of training** in standard GAN, it helps.

In a distribution of scores of real images and generated or fake images, Wasserstein loss tries to increase the gap, which can be summarized from the paper:
critic loss = [average of critic score on real images] - [average of critic source on generated images].

2.4.3. The Generator

Architecture is based on **UNET** [10] (having an encoder and decoder). Decoders decrease the size and increase the number of feature maps and the decoder increases the size of feature maps due to transposed convolution layers. Skip connections are used for transferring features from the encoder to the decoder.

Instead of obtaining lower-size feature maps, **RRDBs** (Residual in residual dense block) [11], have been used. The output of each dense Block is scaled by β to help **identity mapping** in the layers.

Residual blocks help to make the connection such that they reduce the vanishing of the gradient and create deeper connections in shallow layers.
It helps both were lesser parameters than CNNs and important information is also passed through the layers. Throughout network BN, L-ReLU and in the output TanH activation function is used.

2.4.4. The discriminator

It's a patch-based discriminator [12], and its **loss function** is ℓ_1 **norm of the pixel-wise difference** mainly focusing on low-frequency components, but we are trying to preserve high-frequency components, It focuses on every patch and checks whether it is real or fake and gives an avg score in the final output.

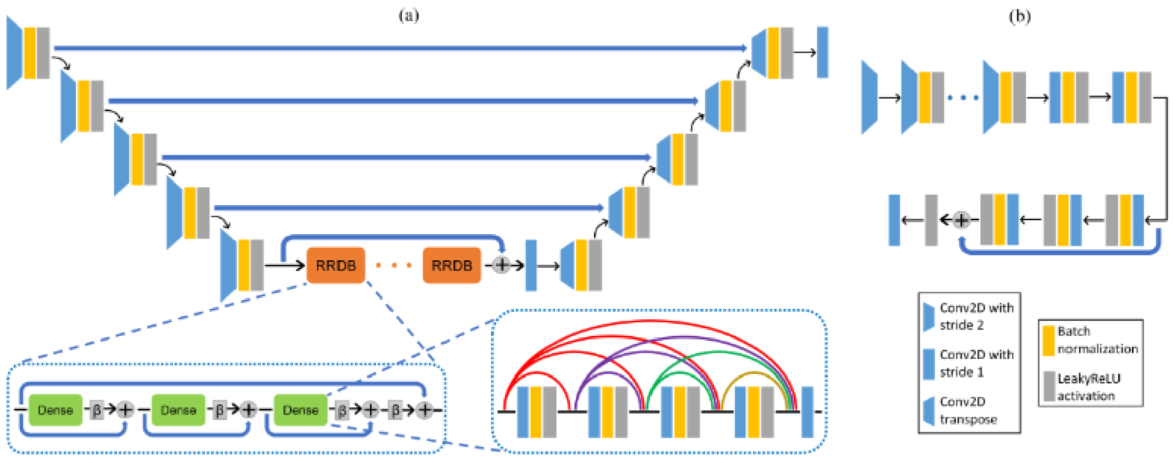

Fig 2.3: (a) generator model architecture and (b) discriminator model architecture [8]

Reducing the pixel-wise difference between generated image and the real image in the generator means an absolute error-based loss in use.
$L_{MAE} = E(\,||G(x) - y\,||_1)$, $||.||_1$ Denotes the ℓ_1 **norm**.

2.4.5. Loss function

Super-resolution and the loss of the reconstruction:
A **mean-SSIM** (mSSIM) [13] based loss is introduced for the reconstruction of the fine textural details in the images.

$$L_{mSSIM} = 1 - E\left(\frac{1}{K}\sum_{j=1}^{K} SSIM(G_j(x), y_j)\right) \tag{1}$$

K is the number of patches, x as input and y is true data

SSIM:

$$SSIM(u,v) = \frac{2\mu_u\mu_v + c_1}{\mu_u^2 + \mu_v^2 + c_1}\frac{2\sigma_{uv} + c_2}{\sigma_u^2 + \sigma_v^2 + c_2} \tag{2}$$

u and v are two patches from real and fake respectively. μ and σ are the mean and variance. c_1 and c_2 are small non-zero constants to avoid zeros.

Overall generator loss:

$$L_{GEN} = \alpha_1 L_{MAE} + \alpha_2 L_{mSSIM} - \alpha_3 E(D(G(x))) \tag{3}$$

Where, α_1, α_2 and α_3 are weighting factors.

It calculates the **structural similarity index** between -1 to +1, for two images, basically, if +1 is coming then both of the images are very similar and if -1 is coming then both of the images are very different from each other.

For that one factor is been chosen which is luminance, which we get by averaging over all the pixels. And is denoted by μ and for that, we use the formula:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4}$$

And for luminance comparison we use the function $l(x,y)$ for μ_x and μ_y . Another factor is there which is contrast and it is denoted by σ, and for the usage of that, here is the formula:

$$\sigma_x = (\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2)^{\frac{1}{2}} \tag{5}$$

For contrast comparisons, we use $c(x,y)$ for σ_x and σ_y .
Now the **luminance comparison function** can be defined as,

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{6}$$

And the **contrast comparison function** can be defined as,

$$c(x,y) = \frac{2\sigma_{xy} + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{7}$$

Here c_1 and c_2 can be written as: $c_1 = (k_1L)^2$ and $c_2 = (k_2L)^2$, where c_1 and c_2 are normal constants, L is the dynamic range of pixel values, for example, (0,255) normally.

For comparison, the standard deviation is the factor which is looked into, so in the **structure comparison function,** $s(x, y)$ the **standard deviation** is σ . and can be written as

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{8}$$

So after multiplying all the functions and putting all the factors as 1 (α, β, γ) we get the formula:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{9}$$

But once the computation has been done over the image, simply **mean of all the local SSIM** values are calculated we get:

$$mSSIM(x,y) = \frac{1}{m}\sum_{j=1}^{m} SSIM(x_j, y_j) \tag{10}$$

Chapter 3

Research Gap

Based on our literature review we can try to find out the research gap in this domain. Using a ViT transformer for MRI reconstruction has some drawbacks. The transformer architecture is primarily suited for natural images, and it may not be optimal for medical images due to their unique characteristics. In addition, transformers require substantial training data, which can be scarce in medical imaging due to privacy concerns. This may lead to overfitting and inadequate generalization of new data. Finally, transformers can be computationally demanding, making them impractical for real-time or low-resource scenarios.

Mean-SSIM (M-SSIM) has a few disadvantages when used in brain MRI reconstruction. It may not fully capture the perceptual quality of reconstructed images, especially in medical images where certain features or regions of interest are important. Additionally, M-SSIM can be computationally expensive and difficult to use as real-time feedback during training. Lastly, M-SSIM's evaluation of model performance can be inconsistent and inaccurate due to its sensitivity to parameter choices and variations in images being compared, making cross-dataset or cross-modality comparisons challenging.

Chapter 4

Dataset

4.1 Adding noise

The dataset used in this study is the MICCAI 2013 grand challenge dataset [14], which consists of medical images with a size of 256x256. The training set of the dataset contains 19,797 images, out of which 6,335 images were intentionally added with 10% and 20% noise, while the remaining images were kept without any noise.

4.2 Splitting the dataset

Additionally, 990 images were selected from the rest of the images and were added with noise, making the total number of images for training 20,787. The study also mentions the use of MRNet [15], but the focus was on the MICCAI dataset.

Chapter 5

The proposed idea

As we have already done some research work on this domain we are going to observe the implementation of the model done by P. Deora [8] and Implement the full code and reconstruct the results of the model.MICCAI data taken by submitting this form: 2013 MICCAI Challenge Workshop on Segmentation: Algorithms, Theory and Applications ("SATA") [14]. MRNET data from the kaggle dataset: **MRNet v1** and downloaded to colab using this documentation [15].

We are going to implement a loss function for the generator and combine it with other loss functions. Our target will be the implementation of **wavelet loss**, **mean absolute error** and **mSSIM**, and also observe their change in the graph with respect to each other.

5.1. Wavelet-based loss

We are using a generator model to generate information, which will be reconstructed from the information that we have given [16].
Let's say, X, y = original and reconstructed information respectively.
And $W(X), W(y)$ = coefficient of original and reconstructed information respectively. We are going to calculate,

$$L(W) = ||W(X) - W(y)||_1 \tag{11}$$

We are going to calculate this for all the original and reconstructional images. And then the mSSIM, Wavelet-based loss and MAE going to be used in the generator training with some weightage.
For our experimentation **pyWavelets library** has been used and there are a few parameters are there that can be tuned to get better results.

5.1.1. Wavelet parameter

The choice of wavelet type is crucial for achieving high-quality reconstructed images in compressed sensing MRI reconstruction. Sparsity-promoting wavelets, such as Daubechies wavelets, can effectively represent signals with sparse structures, leading to reduced computational costs, improved reconstruction quality, and faster processing times.

The choice of wavelet type for CS MRI reconstruction depends on factors such as imaging modality, acquisition parameters, reconstruction quality, and computational constraints. It may require experimentation with multiple wavelet types to determine the best results.

5.1.2. Mode parameter

The choice of mode type for the wavelet transform in compressed sensing (CS) MRI reconstruction can have a significant impact on the quality of the reconstructed image. Common modes for wavelet transformation in CS MRI reconstruction include periodization, symmetric, and reflection, each suited for different types of signals.

The choice of mode type depends on the characteristics of the MRI data, such as its periodicity and smoothness, and the specific requirements of the CS reconstruction algorithm. Experimentation with multiple modes may be necessary to determine the best results.

5.1.3. Level parameter

The level type in compressed sensing (CS) MRI reconstruction controls the amount of detail in the wavelet transform decomposition and has a significant impact on the quality of the reconstructed image. A higher number of levels can improve reconstruction quality but also leads to increased computational cost and longer processing times.

The appropriate number of levels for a specific CS MRI reconstruction problem depends on the imaging modality, acquisition parameters, desired reconstruction quality, and computational and time constraints, and experimentation with different levels may be necessary.

5.2. Implementing different optimizers

To improve the training of machine learning models, different optimizers such as **RMSprop**, **Adamax**, **Adagrad**, and **Adam** can be utilized. It is important to analyze the mean absolute error plot of these different optimizer setups to understand their respective performances and determine the most appropriate optimizer for a given problem. These optimization algorithms are utilized in machine learning to enhance the efficiency and convergence rate of models [17].

5.2.1. Adagrad optimizer

Adagrad is an optimization algorithm that modifies the learning rate based on the frequency of parameters. It performs larger updates for infrequent and smaller updates for frequent parameters. Adagrad is highly effective in sparse data settings where the frequency of the features can vary widely.

5.2.2. RMSprop optimizer

RMSprop is a gradient-based optimization algorithm that adjusts the learning rate adaptively to update the model weights. It maintains a moving average of the squared gradient to scale the learning rate for each weight, which can improve convergence even in the presence of noisy gradients.

5.2.3. Adam optimizer

Adam is a frequently used optimization algorithm in machine learning that combines the benefits of Adagrad and RMSprop. Like RMSprop, it maintains a moving average of past gradients and squared gradients but also includes a momentum term to enhance convergence when dealing with sparse gradients. The learning rate is adjusted for each weight based on the gradient and past squared gradients. Adam is a popular choice for deep learning applications due to its effectiveness and computational efficiency, although it is essential to experiment with various algorithms and hyperparameters to determine the optimal approach for a particular problem.

5.2.4. Adamax optimizer

Adamax is an extension of the Adam optimization algorithm, developed specifically for settings with large, sparse gradients. It utilizes the maximum of the past gradients rather than their root-mean-square, which can be more stable for very large gradients. It also includes an L-infinity weight decay regularization term.

In general, these optimization algorithms are employed to improve the efficiency and convergence rate of machine learning models, particularly in deep learning applications with large, complex datasets. Selecting the appropriate optimization algorithm relies on the specific problem and data characteristics and requires experimentation to determine the optimal approach.

5.2.5. Parameters of optimizers

Optimization algorithm parameters, such as the learning rate, momentum, decay rates, and batch size, can have a significant impact on machine learning model performance. Properly tuning these parameters can help the model converge to a good solution quickly and effectively. However, finding the optimal parameters is a challenging task that requires intuition, experimentation, and optimization techniques such as grid search or Bayesian optimization.

5.3. Implementing different activation functions

Usage of **ELU** [18] and **SELU** [19] instead of ReLU in the generator, to observe the patterns of mean absolute error of different setups for activation functions.

Activation functions are a crucial component in deep learning, as they allow neural networks to model complex non-linear relationships between input and output data. Among the various activation functions available, ELU, SELU, and ReLU are the most commonly used functions.

Trying out different activation functions can enhance the performance of a machine-learning model. Activation functions govern the output of each neuron in a neural network, and they have varying properties that can influence the model's capability to learn and generalize to new data.

By experimenting with different activation functions, we can evaluate their impact on the training and validation performance of the model, and select the one that is most effective for our specific problem. For example, while ReLU may be appropriate for some problems because of its simplicity and computational efficiency, ELU or SELU may be more suitable for other problems due to their smoothness and capability to handle negative inputs.

In some scenarios, it might be useful to employ a combination of activation functions or to develop a customized activation function tailored to the specific problem. In general, experimenting with different activation functions can help us discover the best method for a given machine learning task, resulting in increased accuracy and better generalization ability for the model.

5.3.1. ReLU

ReLU is a simple yet highly effective activation function that sets all negative values to zero, making it computationally efficient. However, ReLU has a limitation in that it can only output positive values, which can result in the so-called "dying ReLU" problem. This occurs when ReLU outputs zero for all inputs, effectively "killing" the neuron and preventing it from learning any further.

5.3.2. ELU

To overcome this limitation, ELU was introduced as a variant of ReLU that maps negative inputs to small negative values, providing continuous and smooth gradients for negative inputs. This helps to alleviate the "dying ReLU" problem and has been shown to improve performance in some deep-learning applications.

5.3.3. SELU

SELU takes this idea even further by introducing a self-normalizing variant of ELU that maintains a mean activation of zero and a standard deviation of one. This helps to improve convergence and generalization in deep neural networks, particularly in deep networks with many layers.

Choosing the right activation function depends on the specific problem and data characteristics, and is often a matter of trial and error. However, it is important to note that activation functions can greatly affect the performance and accuracy of a neural network, and therefore, it is essential to choose the appropriate activation function for the given problem to achieve the best results.

Chapter 6

Experimentation

Experimentation is essential in deep learning to find the best approach for a particular problem by testing different models, architectures, hyperparameters, and optimization techniques. It is also necessary to keep up with the latest advances and discover new methods and techniques in the rapidly evolving field of deep learning research.

6.1. Parameters of the optimizers

Here we are using different optimisers with a set of parameters for training the discriminator and generator respectively:

Name of the optimizers	Learning rate	Beta 1
Adam	0.0002,0.0001	0.5
RMSprop	0.0002,0.0001	-
Adamax	0.0002,0.0001	0.5
Adagrad	0.0002,0.0001	-

6.2. Activation functions

Here we are using different activations for training the generator respectively:

- ReLU (rectified linear activation unit)
- ELU (Exponential Linear Unit)
- SeLU (Scaled Exponential Linear Unit)

6.3. Combination of loss functions

Combining loss functions can improve model performance by capturing different aspects of its ability to fit and generalize. However, it requires careful balancing of objectives. We are going to a combination of different loss functions for training the generator respectively:

- m-SSIM (Mean Structural Similarity)
- Wavelet loss
- MAE (Mean absolute error)

Chapter 7

Results and discussion

Results and discussion in deep learning are important for communicating findings to others and interpreting results in a broader context, including strengths, weaknesses, and future research directions.

7.1. Optimizers

Table 7.1: Results of different optimizers in model training till 5^{th} step

	Adam	**RMSprop**	**Adamax**	**Adagrad**
Discriminator loss	-0.799	-1.144	-0.628	-0.043
Accuracy	0.750	0.969	0.750	0.000
Wasserstein loss	0.002	0.019	0.004	0.001
Mean Absolute error	0.456	0.310	1.238	1.115
m-SSIM	0.963	0.850	1.000	1.000
Generator loss	10.091	7.047	25.779	23.306

Table 7.2: Results of different optimizers in model training till 10^{th} step

	Adam	**RMSprop**	**Adamax**	**Adagrad**
Discriminator loss	-1.578	-2.016	-0.967	-0.056
Accuracy	1.000	1.000	0.953	0.000
Wasserstein loss	0.002	0.031	0.013	0.001
Mean Absolute error	0.307	0.217	1.030	1.094
m-SSIM	0.667	0.640	1.000	1.000
Generator loss	6.812	4.980	21.602	22.883

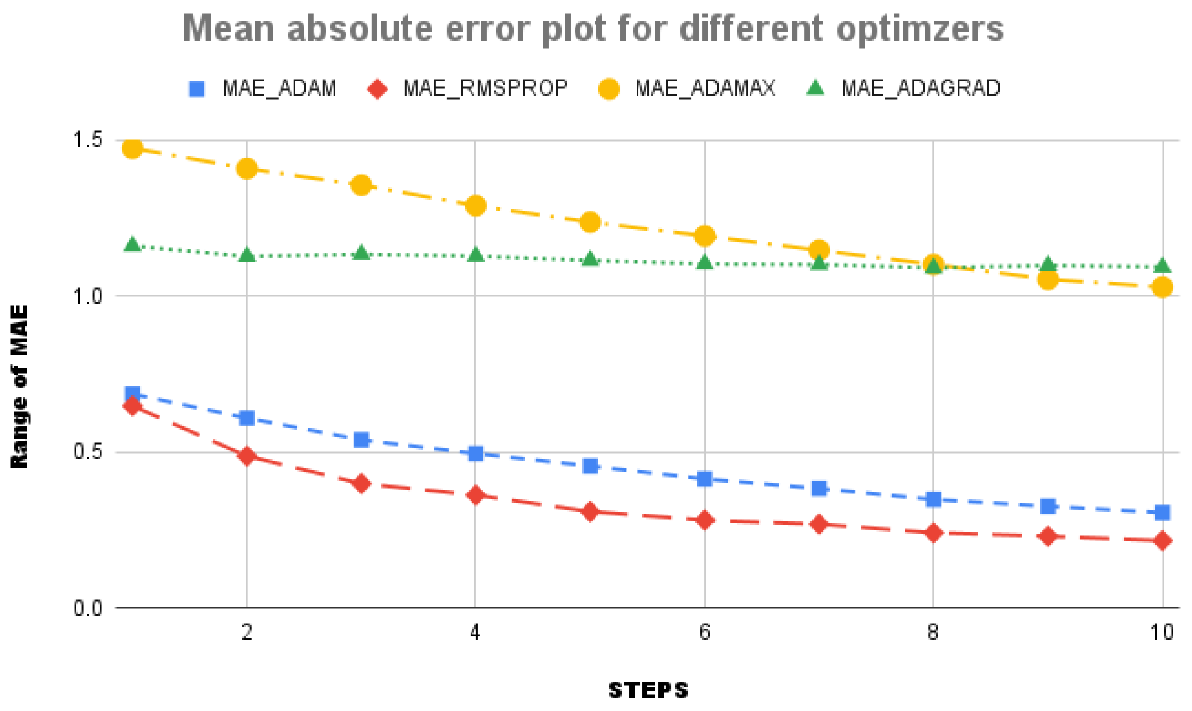

Fig 7.1: Plot of different optimizers in model training

Based on experiment 6.1 we have observed the trends from Table 7.1 and 7.2 of the different optimizers for the 5th and 10th steps, and observed how RMSprop works better than other optimization techniques, also Adam worked quite well, which we can understand from the graph.

7.2. Activation functions

Table 7.3: Results of different activation functions in model training till 5^{th} step

	ReLU	**ELU**	**SELU**
Discriminator loss	-0.799	-0.836	-0.800
Accuracy	0.750	0.812	0.770
Wasserstein loss	0.002	0.015	-0.004
Mean Absolute error	0.456	0.664	0.674
m-SSIM	0.963	1.000	0.992
Generator loss	10.091	14.286	14.471

Table 7.4: Results of different activation functions in model training till *10th* step

	ReLU	ELU	SELU
Discriminator loss	-1.578	-1.563	-1.495
Accuracy	1.000	1.000	1.000
Wasserstein loss	0.002	0.045	-0.011
Mean Absolute error	0.307	0.521	0.494
m-SSIM	0.667	0.995	0.921
Generator loss	6.812	11.409	10.804

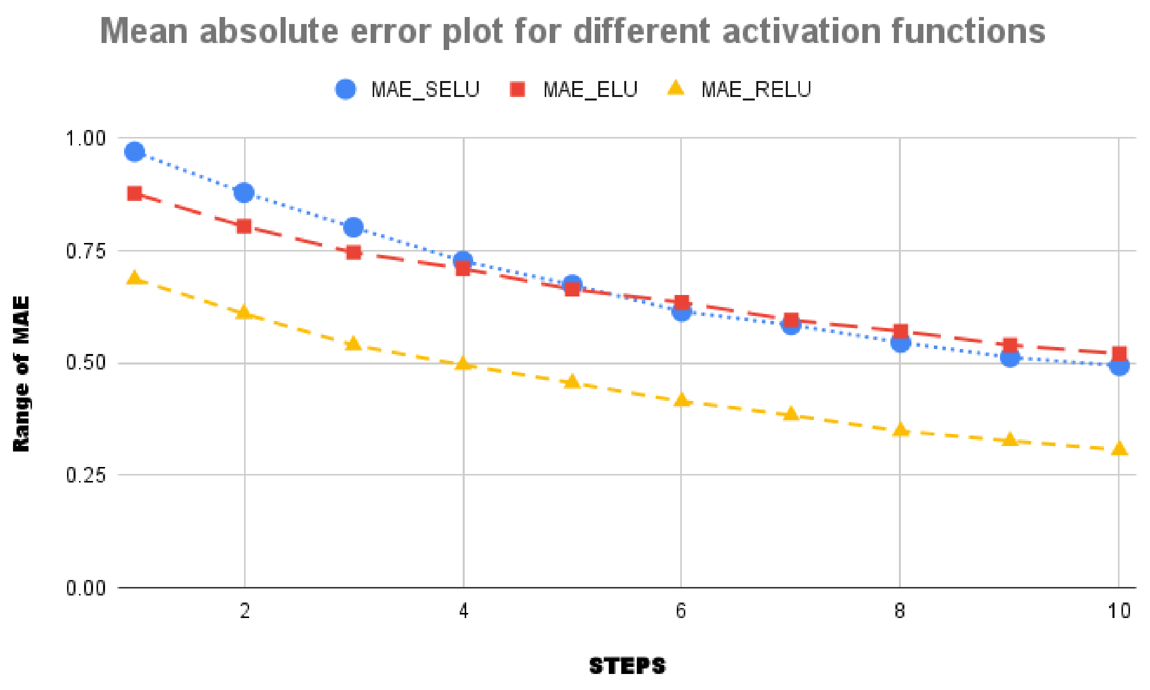

Fig 7.2: Plot of different activation function in model training

Based on experiment 6.2 we have observed the trends from Table 7.3 and 7.4 of the different optimizers for the 5th and 10th steps, and observed how ReLU worked better than other optimization techniques, though ELU and SELU didn't work well in our set of observation.

7.3. Loss functions

Table 7.5: Results of combination for loss functions in model training till *10th* step

Step	Mean Absolute Error	Wavelet based loss	m-SSIM
1st step	0.717	0.8	0.994
5th step	0.491	0.704	0.944
8th step	0.407	0.545	0.831
10th step	0.367	0.504	0.717

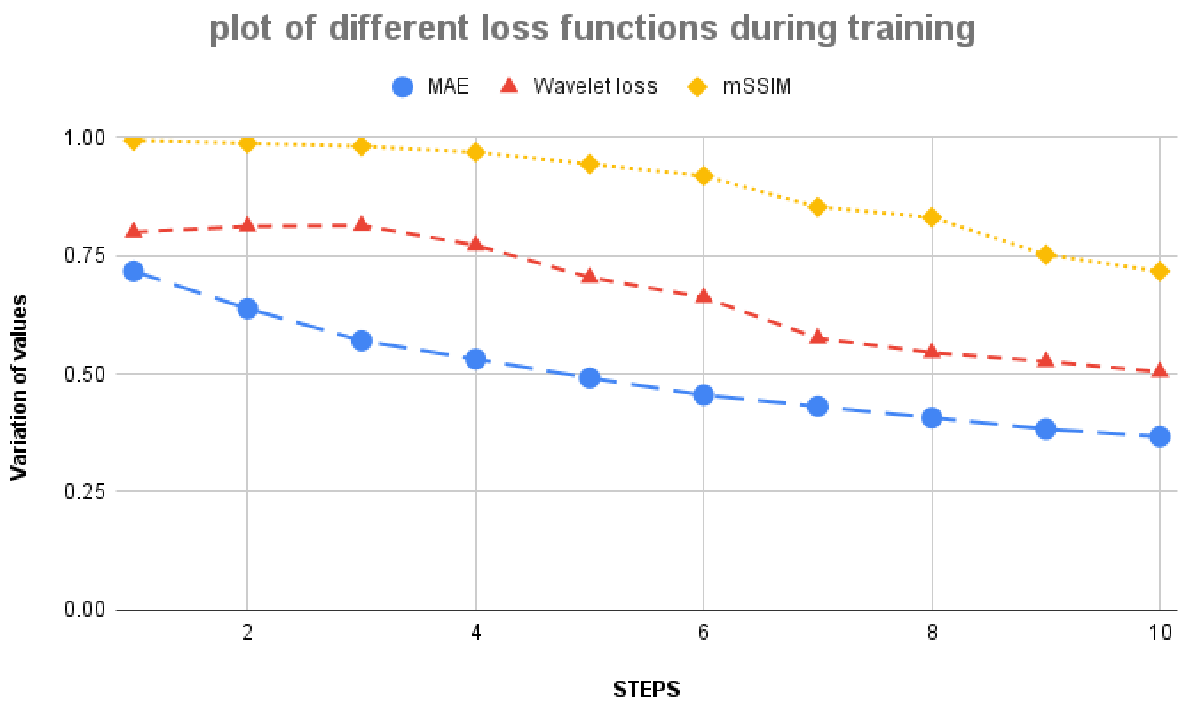

Fig 7.3: Plot of different activation function in model training

Based on experiment 6.3 we have observed the trends from Table 7.5 of the loss functions for all steps, and it can be distinguished the decreasing values with the increasing of the steps.

Chapter 8

Future work

Here are some of the parts that can be improved or implemented based on our research work:

- We can train a classifier model from the embeddings for the gan model and use them for classifying tumour or non-tumour brain MRI, as we are creating the MRI with highly spectral information we feel it will help this medical domain a lot and make the testing procedure faster.

- We are focusing now either on Knee MRI or brain MRI, we can implement some model based on the ideas we have already or the model structures we already have based on that, and more organ MRI reconstruction can be done, like the pancreas, and intestine areas like those, this technique will extend this reconstruction technique with GAN modelling on a new level.

- This would involve collecting a large dataset of MRI scans of these organs and using GANs to generate high-quality images from the noisy and incomplete data. By improving the quality of MRI images, this technique can help doctors and medical practitioners to accurately diagnose and treat patients, leading to better patient outcomes.

Chapter 9

Conclusion

We have some amount of experimentation in the analysis of the project let's make some conclusions on those:

- Based on the outcomes obtained, it can be concluded that **RMSprop** proved to be an effective optimizer for the given dataset. RMSprop utilizes the second moment and a decay rate to enhance its performance over the Adagrad optimizer. On the other hand, the Adam optimizer is also a feasible alternative, as it takes into account both the **first** and **second moments**.

- Also usage of the **ReLU activation** gave good results in the code implementation, as that was used in the generator, which gave better results of training the generator, as ReLU avoids and rectifies vanishing gradient problems it helped the model.

- Usage of **m-SSIM** along with **wavelet loss** showed a nice pattern in the graph, as it is going closer to zero and with that, we can see how **MAE** is performing, combining these loss functions together helped us to make the training process more accurate.

References

[1] D. L. Donoho, “Compressed sensing,” *IEEE Trans Inf Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006, doi: 10.1109/TIT.2006.871582.

[2] E. M. Eksioglu, “Decoupled Algorithm for MRI Reconstruction Using Nonlocal Block Matching Model: BM3D-MRI,” *J Math Imaging Vis*, vol. 56, no. 3, pp. 430–440, Nov. 2016, doi: 10.1007/s10851-016-0647-7.

[3] M. Mardani *et al.*, “Deep Generative Adversarial Neural Networks for Compressive Sensing MRI,” *IEEE Trans Med Imaging*, vol. 38, no. 1, pp. 167–179, Jan. 2019, doi: 10.1109/TMI.2018.2858752.

[4] Jim Nilsson and Tomas Akenine-Möller, “Understanding SSIM,” Jul. 2020, Accessed: May 03, 2023. [Online]. Available: https://research.nvidia.com/publication/2020-07_understanding-ssim

[5] Mohammad Zalbagi Darestani, Jiayu Liu, and Reinhard Heckel, “Test-Time Training Can Close the Natural Distribution Shift Performance Gap in Deep Learning Based Compressed Sensing,” in *Proceedings of the 39th International Conference on Machine Learning, PMLR*, 2022, pp. 4754–4776.

[6] M. B. Lorenzana, C. Engstrom, and S. S. Chandra, “Transformer Compressed Sensing Via Global Image Tokens,” in *2022 IEEE International Conference on Image Processing (ICIP)*, IEEE, Oct. 2022, pp. 3011–3015. doi: 10.1109/ICIP46576.2022.9897630.

[7] Wenjun Chen, Chunling Yang, and Xin Yang, “FSOINet: Feature-Space Optimization-Inspired Network for Image Compressive Sensing,” Apr. 2022, doi: https://doi.org/10.48550/arXiv.2204.05503.

[8] Puneesh Deora, Bhavya Vasudeva, Saumik Bhattacharya, and Pyari Mohan Pradhan, “Structure Preserving Compressive Sensing MRI Reconstruction Using Generative Adversarial Networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020, pp. 522–523.

[9] Mehdi Mirza, “Conditional Generative Adversarial Nets,” *Simon Osindero*, doi: https://doi.org/10.48550/arXiv.1411.1784.

[10] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” 2015, pp. 234–241. doi: 10.1007/978-3-319-24574-4_28.

[11] Xintao Wang *et al.*, “ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks,” in *The European Conference on Computer Vision (ECCV) Workshops*, Sep. 2018, pp. 63–79.

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros, “Image-To-Image Translation With Conditional Adversarial Networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1125–1134.

[13] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image Quality Assessment: From Error Visibility to Structural Similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: 10.1109/TIP.2003.819861.

[14] Alireza Akhondi-asl, “NCI-MICCAI 2013 Grand Challenges in Image Segmentation,” 2013. https://wiki.cancerimagingarchive.net/display/Public/NCI-MICCAI+2013+Grand+Challenges+in+Image+Segmentation (accessed May 03, 2023).

[15] N. Bien *et al.*, “Deep-learning-assisted diagnosis for knee magnetic resonance imaging: Development and retrospective validation of MRNet,” *PLoS Med*, vol. 15, no. 11, p. e1002699, Nov. 2018, doi: 10.1371/journal.pmed.1002699.

[16] Lukas Prantl, Jan Bender, Tassilo Kugelstadt, and Nils Thuerey, “Wavelet-based Loss for High-frequency Interface Dynamics,” Sep. 06, 2022. https://arxiv.org/abs/2209.02316 (accessed May 03, 2023).

[17] Sebastian Ruder, “An overview of gradient descent optimization algorithms,” Jun. 15, 2017. https://arxiv.org/abs/1609.04747 (accessed May 03, 2023).

[18] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs),” in *4th International Conference on Learning Representations, ICLR 2016*, San Juan, Feb. 2016.

[19] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter, “Self-Normalizing Neural Networks,” in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.