MRI to CT Translation for Ultrasound Computed Tomography

Group 25

Department of Bioengineering

Imperial College London

A Project Report Submitted in
Partial Fulfilment of the

MEng Bioengineering Degree

Supervisor:
Professor Meng-Xing Tang

Department of Bioengineering
mengxing.tang@imperial.ac.uk

April, 2023

MRI to CT Translation for Ultrasound Computed Tomography

Alexia Badea, Ella Stanbury, Ioana Lazar, Fjona Lutaj, Soranna Bacanu

April 13, 2023

Abstract

MRI and CT images can be used to obtain the speed of sound in the brain and skull respectively, which would enable Full-Waveform Inversion for Ultrasound Computed Tomography. Same-patient MRI and CT are rarely available, so it would be beneficial to be able to translate from one to another. CycleGANs have been chosen for this because they implement unsupervised learning that can be used on unpaired data. In this project, we aim to first apply CycleGANs for MRI to sound speed translation and test the effect of different hyperparameters, including learning rate, betas, batch size, depth and activation function. We then apply our findings to MRI to CT translation.

Results of our hyperparameter tuning largely confirm that the hyperparameters stated in the original CycleGAN paper[1] provide optimal training for MRI to sound speed translation. Even so, valuable relationships between hyperparameters and the performance of the network were established. For example, we found that increasing β_1 in the discriminator's Adam can slow down its training, depth can be decreased to reduce computational effort without compromising on performance, and downsampling detetrmines the feature detection level. Although the optimal hyperparameters found are not guaranteed to be exactly the same in MRI and CT translation, our MRI to sound speed findings enable us to anticipate and make informed decisions about how changing the hyperparameters will affect the network. This has resulted in improvements in the MRI to CT translation generator loss, however further work needs to be done to obtain successful results.

Acknowledgements

We would like to sincerely thank Oscar Bates for his helpful guidance and support throughout this project and Clara Rodrigo Gonzales for her insightful recommendations. We would also like to thank Professor Meng-Xing Tang for allowing us to undertake this project and facilitating our progress.

1 Introduction

1.1 Motivation

The current standard brain imaging techniques are MR and CT. MR has the best spatial resolution, but both techniques present disadvantages related to limited patient accessibility, causing high symptom-to-result time and exposure to ionizing radiation in the case of CT. Ultrasound Tomography (UT) is a low-cost, safe, and portable alternative. However, conventional UT methods cannot be applied to the brain as the acoustic waves are reflected by the skull, creating distorted images.

Full-waveform inversion (FWI), an ultrasound com-

puted tomography technique, accounts for these interactions by inverting the full-wave equation.[2] The applications of FWI to medical imaging would enable fast diagnosis in many pathologies that require rapid intervention, such as stroke. To converge to accurate results, a large database of speed-of-sound (SoS) through skull and brain is needed to use as starting models for FWI.

Speed-of-sound through the brain can be derived from MR, and through skull from CT, but patients rarely undergo both scans. Therefore, the motivation behind this project is to create a synthetic CT from MR using image-to-image translation, so the two images can be used to create a starting FWI model.

1.2 Relevant Previous Work

Deep neural networks are used in medical imaging for tasks like segmentation, denoising, and cross-modality image translation. A key distinction between deep learning environments is determined by whether the input data has been labelled with a corresponding output in the target domain, thus being paired. If the input data is unpaired, the network learns to recognise patterns in the structure of the datasets. Paired MRI and CT data is limited, therefore, an important aspect of this project is investigating the most efficient method in an unsupervised learning environment.

Previous studies have achieved cross-modality image translation for unpaired datasets using CNN, U-Net, and CycleGANs. In general, CycleGAN has been proven to outperform other unsupervised methods and shows similar results to its supervised counterparts like pix2pix.[1] As training neural networks requires large datasets, optimisation-based methods like Bayesian Mixture Models have also been used for cross-modality Image translation,[3] however, less accurate results were obtained.

The adaptability of the CycleGAN method to the purpose of this project was initially demonstrated by a previous study that applied the framework to a horse-to-zebra[4] image translation task. In an additional study on deep MR to CT translation, CycleGAN showed lower errors and produced more realistic reconstructed images compared to the one-directional supervised GAN, however, paired datasets were used.[5] We, therefore, choose to investigate the performance of CycleGAN on MR to CT translation using unpaired datasets.

1.3 Background Info

1.3.1 GANs

A Generative Adversarial Network (GAN) is a deep learning framework comprised of two neural networks: a generator and a discriminator[6].[7] After being exposed to a certain dataset, the generative model produces new data to approximate the given dataset. The discriminative model then tries to distinguish between the real data samples and the fake data produced by the generator.

The aspect that has proven to make this framework highly successful[6] in image generation tasks is the principle of adversarial loss. During training, a set of fake data produced by the generator is fed into the discriminative model in addition to a set of real data.

While GANs provide an effective methodology in the presence of a paired dataset where the generated images are compared to target images, MRI to CT translation calls for an equally efficient methodology in the context of an unpaired dataset.

1.3.2 CycleGANs

CycleGANs present the addition of cycle-consistency loss to the basic GAN framework, allowing the network to continuously compare input images that have been translated between the two domains through the processes of forward and backward cycle consistency. This framework comprises two generators and two discriminators that work in conjunction. Thus, the need for a paired dataset is eliminated as feedback is provided by comparing the input and reconstructed images.

In addition, the concept of identity loss is added to stop the network from making changes to output images that already match the target domain. These operations can be summarized in Figure 1 and further information behind their functionality is provided in Section 2.2.

1.4 Main Project Objective

Our project has two key aims. First, to replicate and improve results on a previous project on MRI to speed-of-sound translation, by tuning the parameters used in the architecture and training of the networks. The second aim is to take the set of parameters found to perform the best, apply them to MRI-to-CT translation and then make adjustments based on the results and findings from MRI to speed-of-sound. Below, we will present the implementation and findings for each of

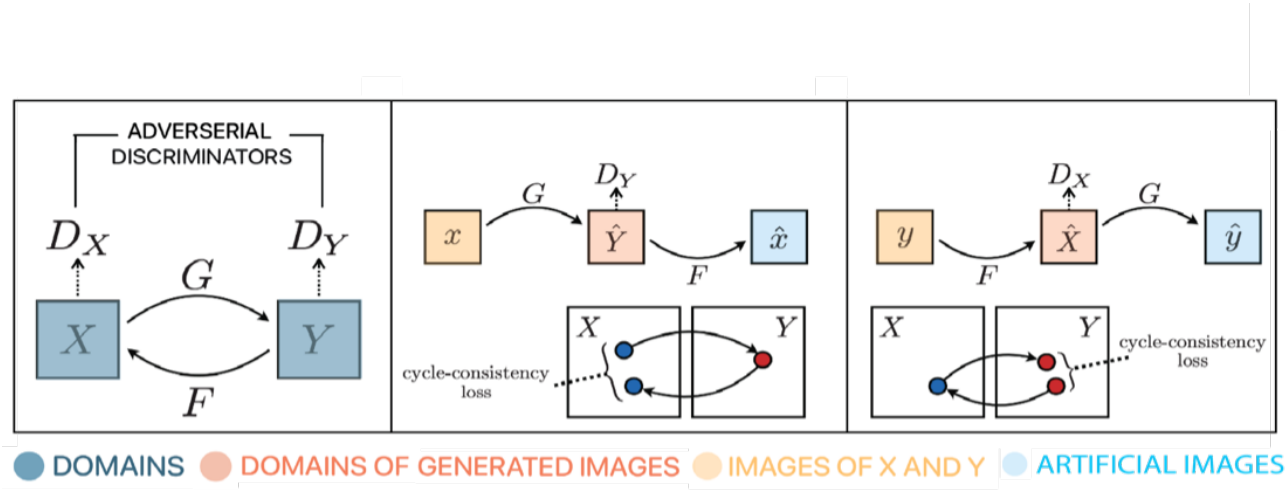

Figure 1: a) Adversarial Loss given the two mapping functions F and G. b) Forward cycle-consistency loss.c) Backward cycle-consistency loss.

these steps along with potential future improvements.

2 Methodology

Considering the eventual aim of this project is to perform the translation of MRI to CT images, CycleGANs pose a suitable solution as they have previously been proven successful with unpaired data[4] and implementing unsupervised learning. To understand how different hyperparameters affect performance, we first implemented the CycleGAN for MRI to SoS translation on a paired dataset. This section elaborates on the application of CycleGANs for this.

2.1 Dataset

2.1.1 Regulations

The MRI dataset was obtained from the Young Adult database of the Human Connectome Project[8] and was used following the WU-Minn HCP Consortium Open Access Data-Use Terms.[9] An unpublished method denoted the 'analytical method' was then used to make paired SoS volumes with the T1-MRI and T2-MRIs from the HCP dataset. The CT images are from the CQ500 dataset,[10] licensed under a Creative Commons Attribution-NonCommercial-ShareAlive 4.0 International License,[11] which our project complies with.

2.1.2 Processing

Preparing the datasets involved removing unusual images and splitting the dataset into training, validation, and test subsections. Since our datasets are small, only 10% of the dataset was withheld from training for MRI to SOS, which was considered sufficient to represent the population in testing. The MRI to CT translation dataset is even smaller, so the decision was made not to remove any of the training data.

2.1.3 Type of Images

The MRI to SOS translation was performed on brain and skull tissue patches. For the networks to properly distinguish between the 2 types of tissue, spatial information is needed, so aligned images corresponding to the same brain area were used. For MRI to CT translation, we used images containing both mixed patches and skull tissue. The skull-only data was obtained by plotting the histogram of one normalized image in each domain, finding the threshold value corresponding to brain tissue, and setting the pixels found to have intensities in that range to 0.

2.2 Network Training

As observed in Figure 1, given two domains, X and Y, and their respective training samples, x and y, a model can be defined by containing two mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$. Two discriminators, D_X and D_Y, work to distinguish between real and generated samples. The adversarial loss[1] can therefore be represented mathematically as follows:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)} [log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)} [log(1 - D_Y(G(x))] \quad (1)$$

where $D_Y(G(X))$ represents the probability that the data is real.

A key feature of CycleGANs is the implementation of cycle consistency loss to reduce the number of possible mappings, which can be represented by the following equation:

$$\begin{aligned}\mathcal{L}_{cyc}(G,F) = \mathbb{E}_{x\sim p_{data}(x)}[|F(G(x)) - x|_1] \\ + \mathbb{E}_{y\sim p_{data}(y)}[|G(F(y)) - y|_1]\end{aligned} \tag{2}$$

Identity loss is also included to prevent the network from making changes to output images that already match the target domain, as given in the following equation:

$$\begin{aligned}\mathcal{L}_{id}(G,F) = \mathbb{E}_{x\sim p_{data}(x)}[|F(x) - x|_1] \\ + \mathbb{E}_{y\sim p_{data}(y)}[|G(y) - y|_1]\end{aligned} \tag{3}$$

The full objective of the network is summarized as follows:

$$\begin{aligned}\mathcal{L}_{gen}(G,F,D_X,D_Y) = \mathcal{L}_{GAN}(F,D_X,X,Y) \\ + \mathcal{L}_{GAN}(G,D_Y,X,Y) \\ + \lambda_1\mathcal{L}_{cyc}(G,F) \\ + \lambda_2\mathcal{L}_{id}(G,F)\end{aligned} \tag{4}$$

λ_1 and λ_2 are constants that control the relative importance of each loss.

Training the network consists of finding parameters that minimize the losses. The networks start with randomly initialized weights, and they are applied to batches of images from the training dataset. The generator and discriminator losses are then calculated and then backpropagated through the network to obtain their gradient or derivative, the sign of which gives information about the direction of loss decrease. An optimization algorithm based on stochastic gradient descent uses this information to find the weights that give the minimum loss. The default optimizer used in Deep Neural Networks is Adam[12] , which uses adaptive learning rate based on the statistics of the previous gradients.

2.3 Network architecture

The starting point of the networks architecture was the CycleGAN architecture proposed by Zhu et al.[4] The generator is composed of an encoder, multiple residual networks, and a decoder, whereas the discriminator is in the form of a Patch-GAN. The initially implemented architecture is showed in Figure 2. Several aspects of the model such as the activation functions, the number of residual blocks, and downsampling layers have been subjects of research to determine the most efficient CycleGAN architecture as further elaborated in Section 2.4.

2.4 Parameter Tuning

Neural networks are made up of parameters and hyperparameters. The model updates the parameters during training, whereas hyperparameters are set before learning begins. If the hyperparameters are not set correctly, the network's performance will be significantly reduced. For MRI to sound speed translation, we investigate the effects of changing the hyperparameters outlined below.

2.4.1 Batch Size

Batch size determines the number of samples used in each iteration. The use of small batch sizes has been shown to improve generalization performance and optimization convergence[13][14] as well as requiring less GPU memory. However, there are also instances of achieving the same generalization with larger batch sizes and longer training.[15] As further explained in Section 2.4.6, we only train each hyperparameter for five epochs, so smaller batch sizes were mostly tested. One common recommendation is that the batch size should be a power of two to take advantage of GPU processing.[16] As a result, batch sizes of 1,2,4,8,64 and 128 were chosen for testing.

2.4.2 Adam Optimiser Hyperparameters

In a CycleGAN, the discriminator can become too good at accurately distinguishing between real and fake images. To compensate, the generator can attempt to produce more complex images to fool the discriminator. Sometimes, these complex images are not the desired output as they might be low quality or unrealistic. To prevent this from happening, our approach is to analyze different hyperparameters of the Adam optimizer.

- **Betas**:
  The Adam algorithm updates exponential moving averages of the gradient and the squared gradient, where the hyper-parameters β_1 and β_2 control the exponential decay rates of these moving averages. β_1 controls the first-moment estimate.[17] Choosing a higher value for this parameter would result in a smoother estimate of the first moment, which would help to stabilize the optimization process. However, high values can lead to a slower training speed. The default values recommended by the

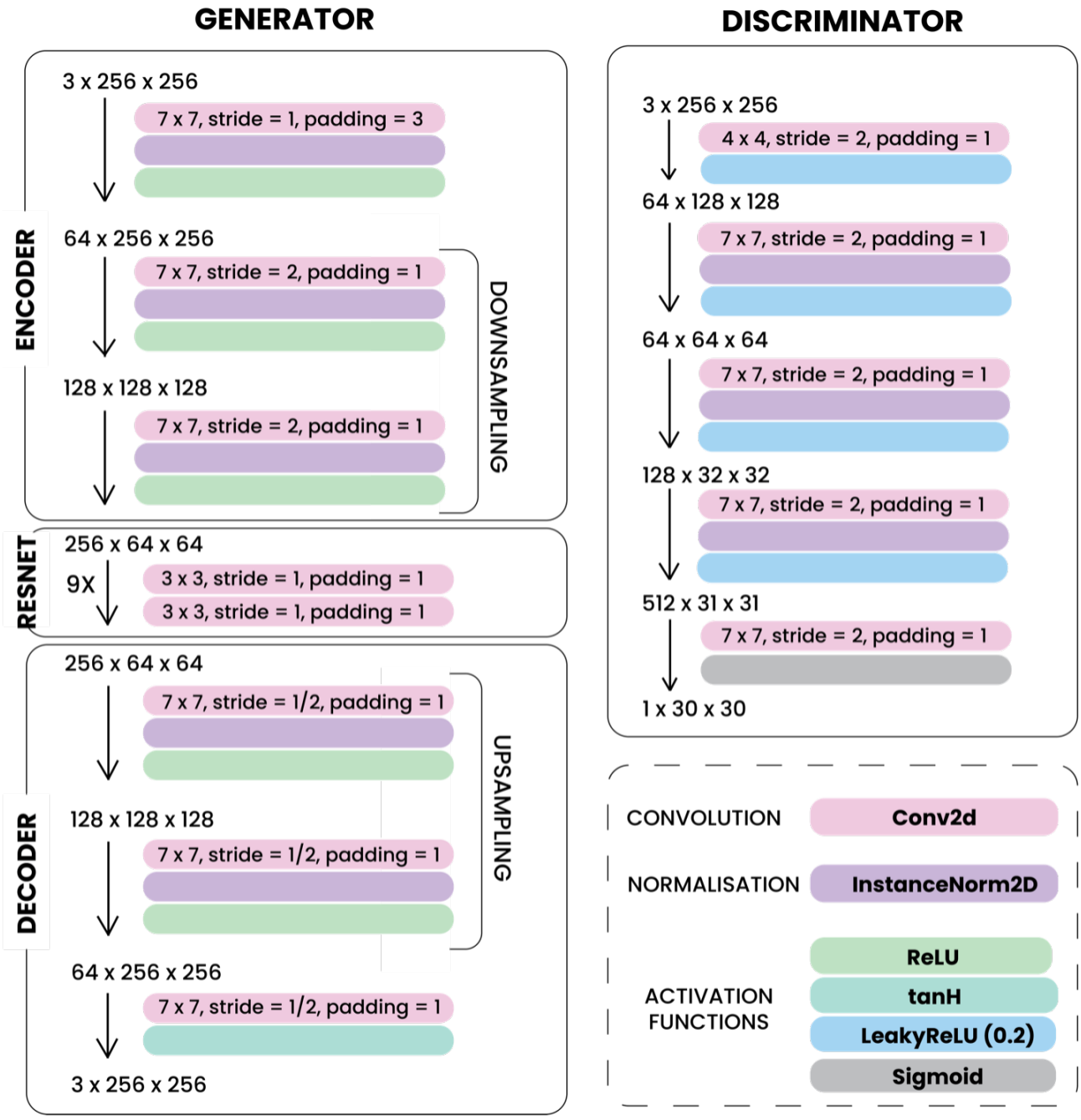

Figure 2: Architecture of the initial implementation of the Generator and Discriminator

authors are 0.9/0.999 for both generator and discriminator. Our approach is to find the optimal for our CycleGAN by testing different pairs for the generator and discriminator.

- **Learning Rate**:
  Learning rate is also an important Adam hyperparameter because it controls the step size taken during optimization. Using too small a value may result in very slow convergence or training stopping altogether as the minimum cannot be reached, whereas using a learning rate that is too big may result in gradient descent explosion[18] and overshooting the optimal solution. Since the recommended default learning rate for Adam is 0.001,[17] our approach to finding the optimal learning rate for the CycleGAN involved testing values ranging between 10^{-8} and 1000. .

2.4.3 Depth

Adding depth to a network can increase the number of feature levels that can be detected in an image, however training by finding direct mappings in large stacks of convolutional layers is slow and can lead to an increased training error compared to shallower models. Instead, in Residual Networks a residual mapping is learned by having shortcut connections that skip layers, facilitating the optimization process.

An increased number of residual blocks[19] still requires higher computational effort. The original CycleGAN implementation uses 6 residual blocks for 128 × 128 training images, and 9 residual blocks for 256 × 256 or higher-resolution training images. As we train the networks on 64x64 images, we investigated the performance for 0 to 9 residual blocks.

2.4.4 Downsampling

Downsampling results in a decrease in spatial resolution of the encoded image. This helps the model detect high-level features in the image that it needs to map in the other domain. Too much downsampling can result in the network missing details in the input images, resulting in blurred output images. Low downsampling means the network will be good at mapping details but might fail in detecting image features. Therefore, a balance must be found. The original CycleGAN uses two downsampling layers so we tested for 0 to 4 downsampling layers to account for the difference in image size.

2.4.5 Activation Function

Activation functions introduce non-linearity and transform the summed weighted input into an output that is transferred to the next layer. Understanding which activation functions should be used is important, especially in deep networks with large numbers of hidden layers.

Conducted research[2021] proposes that ReLU and LeakyReLU functions tend to be more successful in deep learning models and they are found in the implemented CycleGAN architecture as illustrated in Figure 2.[22] However, as the variety of activation functions analyzed experimentally previously is rather small, the functions in Figure 3 have been investigated to confirm the best performing option:

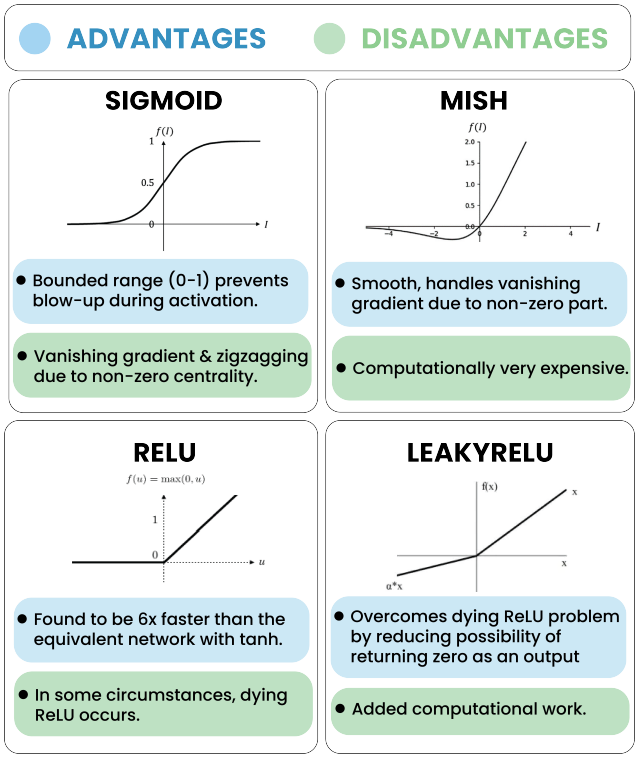

Figure 3: Function Types

2.4.6 Tuning Implementation

Our approach started with dividing the parameters and forming multiple short experiments, each focused on changing a given parameter while keeping the rest constant. It was also decided that the shorter runs would be trained for 5 epochs each, since this was found to be long enough to produce satisfactory images without being too time-consuming.

Additionally, the interdependence between hyperparameters should be recognized. For example, there is a correlation between learning rate and batch size:

Aspect			Tuning Details	Original Values	Relevance of tuning
Training details	1. Batch size		1,2,4,8,64,128	1	Training speed, steady-state loss
	2. Adam	β_G/β_D	0.6/0.9, 0.5/0.5 0.9/0.9, 0.99/0.9, 0.99/0.5, 0.99/0.99	0.9/0.9	
		Learning Rate	10^{-8}, 10^{-5}, 5×10^{-4} 2×10^{-4}, 10^{-4}, 10^{-3} 2×10^{-3}	10^{-4}	
Network Architecture	3. Resnet Depth		0 to 9 Layers	6 and 9	
	4. Downsampling Layers		0 to 4 layers	2	Image quality
	5. Activation functions		Mish, ReLU, LeakyReLU, Sigmoid	Generator: ReLU Discriminator: LeakyReLU	

Table 1: Summary of the tuning process regarding the relevance of each aspect and respective tuning details. Original values correspond to the implementation proposed by Zhu et al[4]

smaller learning rates are recommended in conjunction with smaller batch sizes.[20] There are many methods to test different combinations of hyperparameters, such as grid search, random search, and Bayesian optimization. We implemented Optuna to briefly investigate optimal combination of batch size and learning rate.

2.5 Evaluation of hyperparameters

The effect of changing hyperparameters can be evaluated from two main points of view. The first involves the average and variance of both training and validation losses. WandB[23]was implemented to log the losses during training, examples of which can be seen in Figure 21. Although looking at the loss alone cannot fully quantify how well the network can translate between image domains, it is a good indicator of whether the network is converging or diverging. We therefore cautiously define a parameter to be optimal if it results in the lowest generator loss after training for five epochs. Since part of the dataset is excluded from training, validation loss can be used to identify any overfitting.

The second type of evaluation refers to image quality, through subjective and objective assessment. After visually inspecting the images, we quantified their quality by computing the error in pixel intensity and with the Structure Similarity index[24],[25] which reflects the similarity in luminance, contrast, and structure between the original image and the synthesized or reconstructed one. As the MRI-to-SOS data is paired, these were calculated for both original to fake images and original to reconstructed images.

2.6 MRI to CT Translation

We used a set of parameters that were found to be successful for MRI to sound speed on mixed-tissue patches of MRI and CT. After checking the loss behaviour and visually assessing the images, we adjusted the parameters to improve results, based on the knowledge we gained from MRI to speed-of-sound hyperparameter tuning. After sub-optimal results for this, we proceeded to train on skull-tissue only to try and remove any issues with image alignment.

3 Results

What follows is an elaborate representation of the results obtained by the training implementation explained in section 2.4. The quality and visual assessment of images is represented in Figures 16 and 17 respectively.

3.1 Batch Size

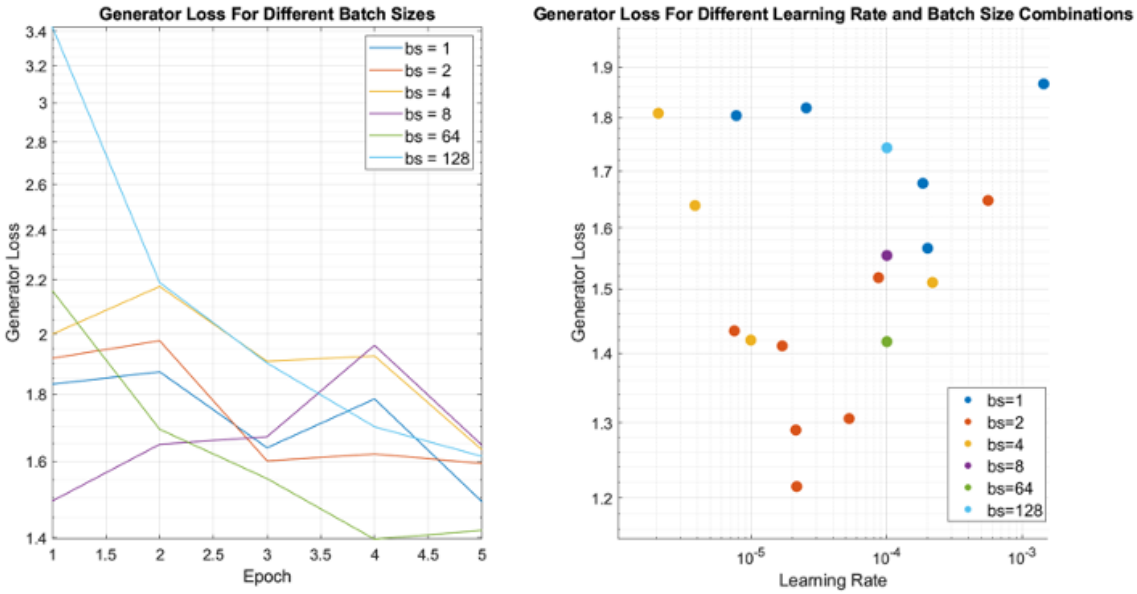

Figure 4: Generator loss throughout training. Left: Different batch sizes with the learning rate fixed at 1×10^{-4}. Right: Different combinations of batch size and learning rate

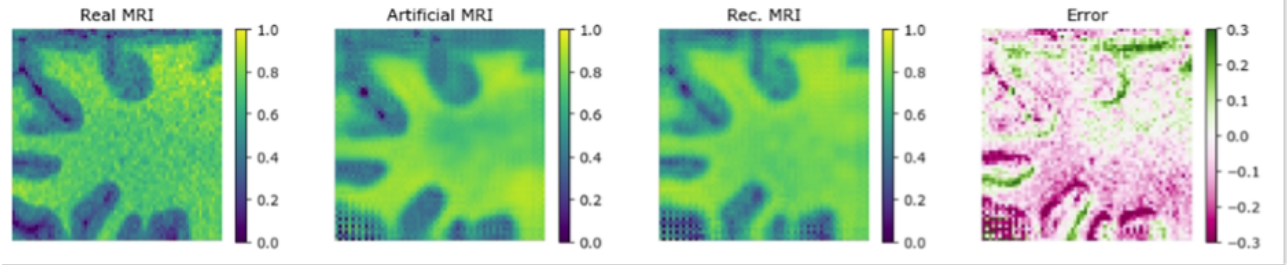

Figure 5: Checkerboarding artefact in the images obtained from training for 5 epochs with a batch size of 1 and learning rate of 1×10^{-4}.

3.2 Adam Optimiser Hyperparameters

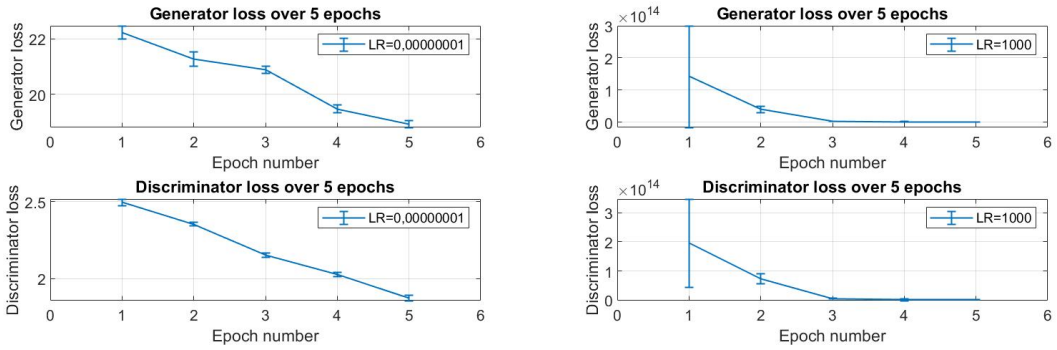

Figure 6: Training Generator and Discriminator loss over 5 epochs for learning rates of 10^{-8} and 1000

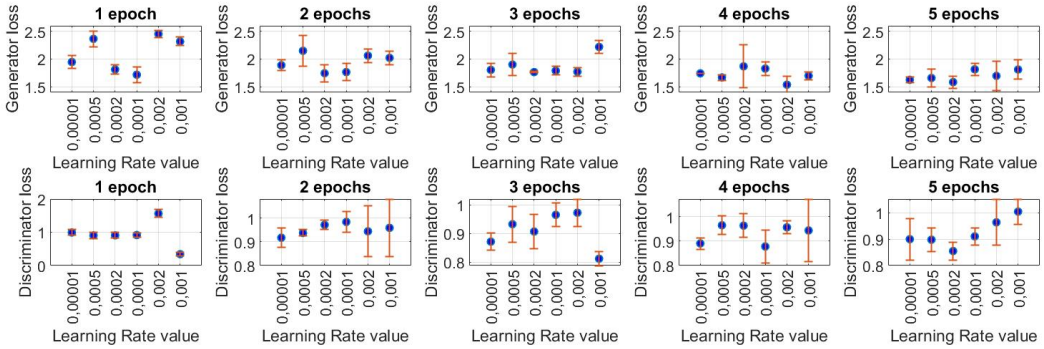

Figure 7: Comparison of Generator and Discriminator loss for networks with different learning rates for each epoch

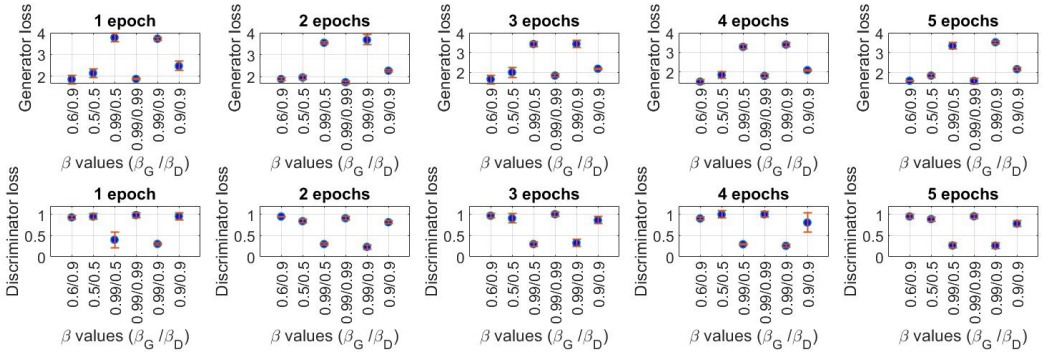

Figure 8: Comparison of Generator and Discriminator loss for networks with different β_G/β_D combinations for each epoch

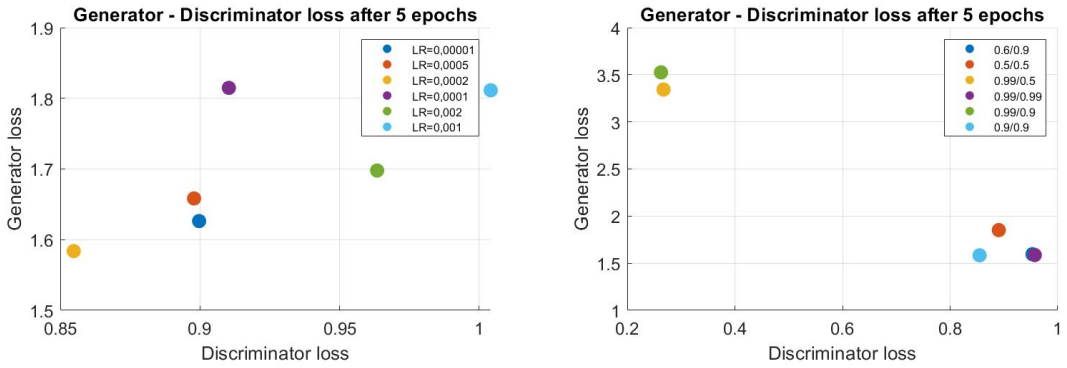

Figure 9: Generator-Discriminator losses after 5 epochs for networks with different learning rates and betas

3.3 Depth

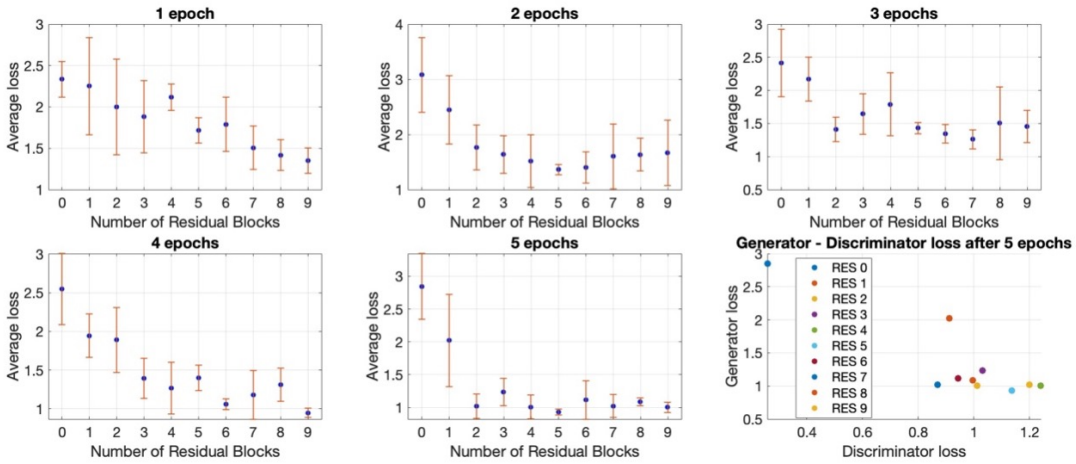

Figure 10: Comparison of Generator Loss for networks with 0 to 9 residual blocks after each of the first 5 epochs. Generator-discriminator losses relation after 5 epochs

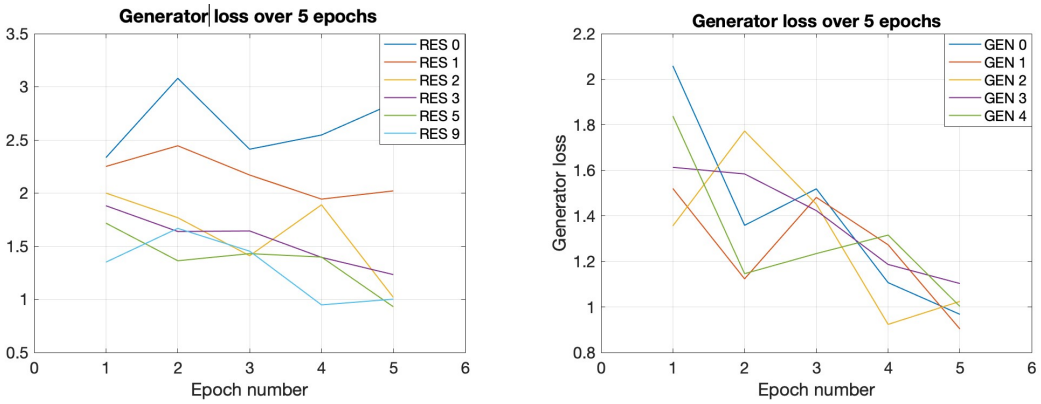

Figure 11: Training generator loss over the first 5 epochs. Left: Models with 0,1,2,3,5,9 residual blocks. Right: Models with 0 to 4 downsampling layers

Models were then tested on unseen data.

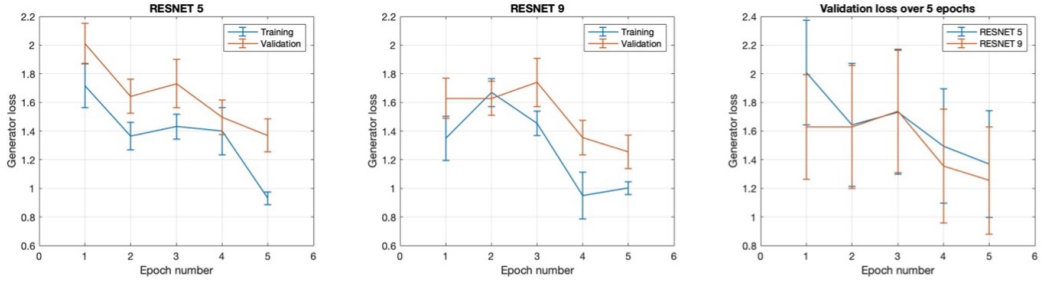

Figure 12: Generator validation loss over the first 5 epochs. From left to right: Training and validation generator loss comparison for network with 5 and 9 residual blocks. Comparison of the validation loss of the 2 models. Error bars correspond to std of the mean obtained in 4 different runs

Validation loss	Mean	STD (within the group)
Residual blocks: 9	1.2547	0.3588
Residual blocks:	1.3691	0.3739

Table 2: Validation loss for 5 and 9 residual blocks

3.4 Downsampling

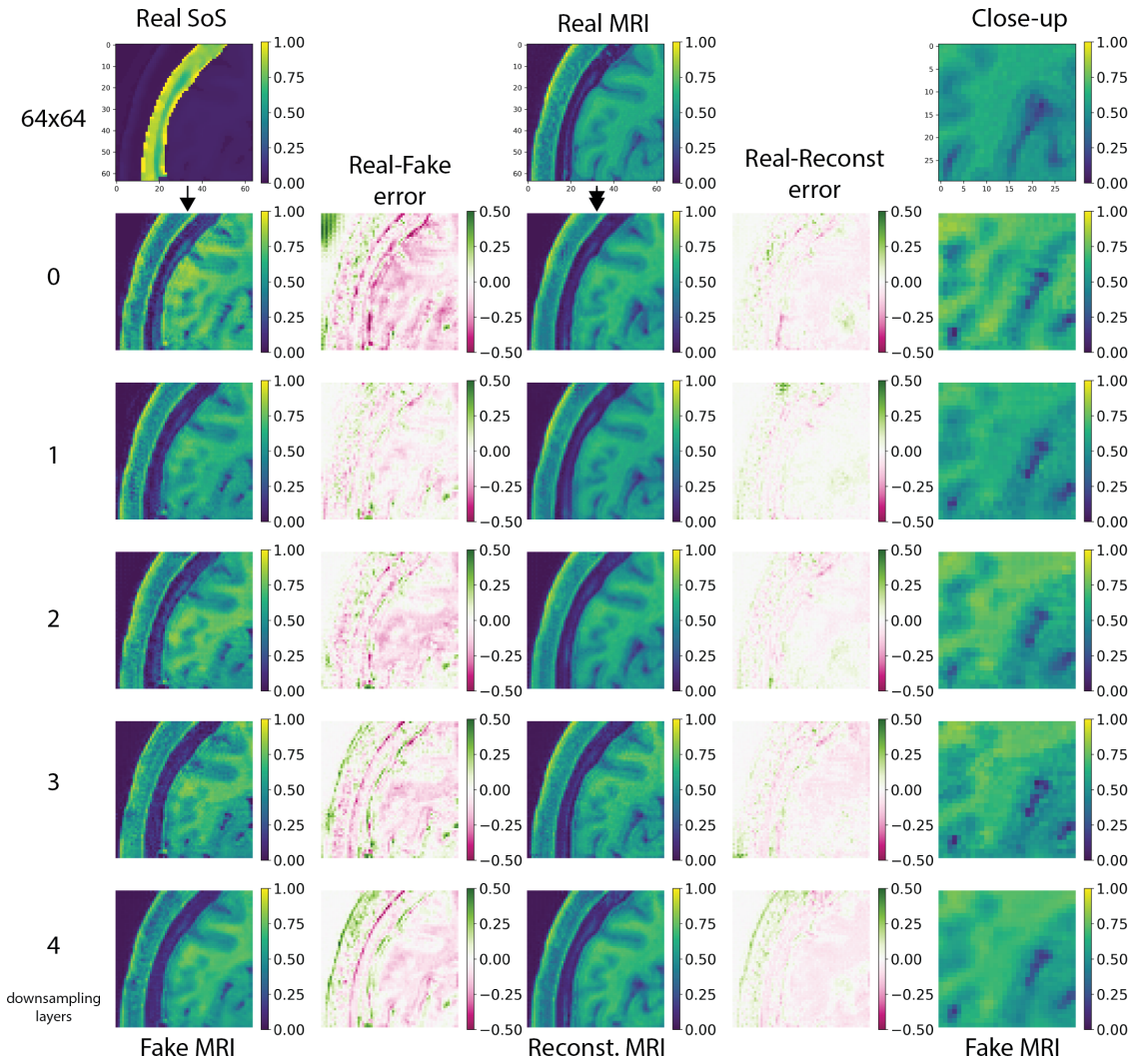

Figure 13: Visual assessment of images in the MRI domain, after undergoing one translation (Fake Images) and 2 translations (Reconstructed Images). Each row displays results of a different number of downsampling layers (see Appendix 6.3 for Sos domain)

3.5 Activation function

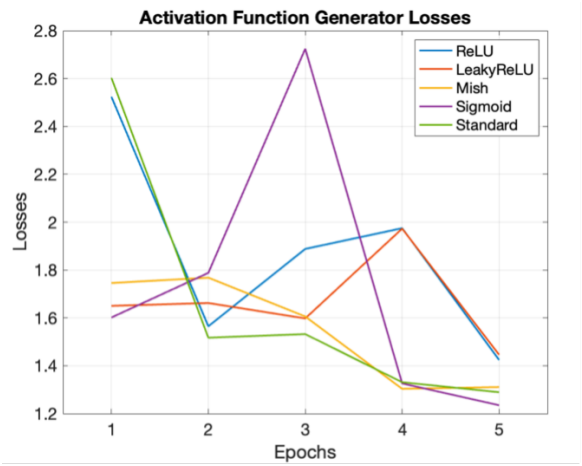

Figure 14: Generator losses for different activation functions over 5 epochs

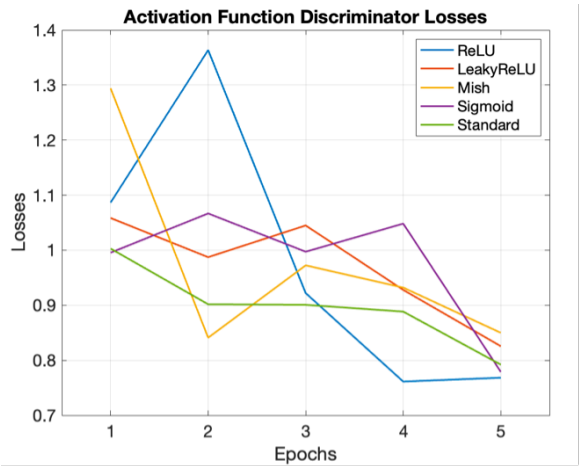

Figure 15: Discriminator losses for different activation functions over 5 epochs

3.6 Evaluation

MEAN | STD		SSIM							
		MRI				SOS			
		Reconstructed		Fake		Reconstructed		Fake	
DOWNSAMPLING LAYERS (encoded image size)	0 (64x64)	0.8712	0.09	0.6610	0.10	0.9133	0.08	0.7957	0.09
	1 (32x32)	0.8876	0.08	0.7108	0.08	0.9157	0.08	0.8026	0.08
	2 (16x16)	0.8974	0.08	0.7239	0.08	0.9360	0.06	0.8162	0.08
	3 (8x8)	0.8858	0.08	0.7186	0.08	0.9242	0.07	0.8121	0.08
	4 (4x4)	0.8625	0.09	0.6876	0.09	0.9277	0.06	0.8016	0.08
ACTIVATION FUNCTION	Original	0.9056	0.91	0.5755	0.58	0.9378	0.94	0.7936	0.79
	ReLU	0.8821	0.88	0.6350	0.63	0.9129	0.91	0.7919	0.79
	LeakyReLU	0.9266	0.93	0.6243	0.62	0.9176	0.92	0.7558	0.76
	Sigmoid	0.9620	0.96	0.6439	0.64	0.9622	0.96	0.7435	0.74
	Mish	0.9220	0.92	0.6352	0.64	0.9346	0.93	0.7622	0.76

Figure 16: Image quality assessment through SSIM for different number of downsampling layers in the generator and different activation functions. Note: Comparison to be effectuated within the category and not across, as different data was used for training

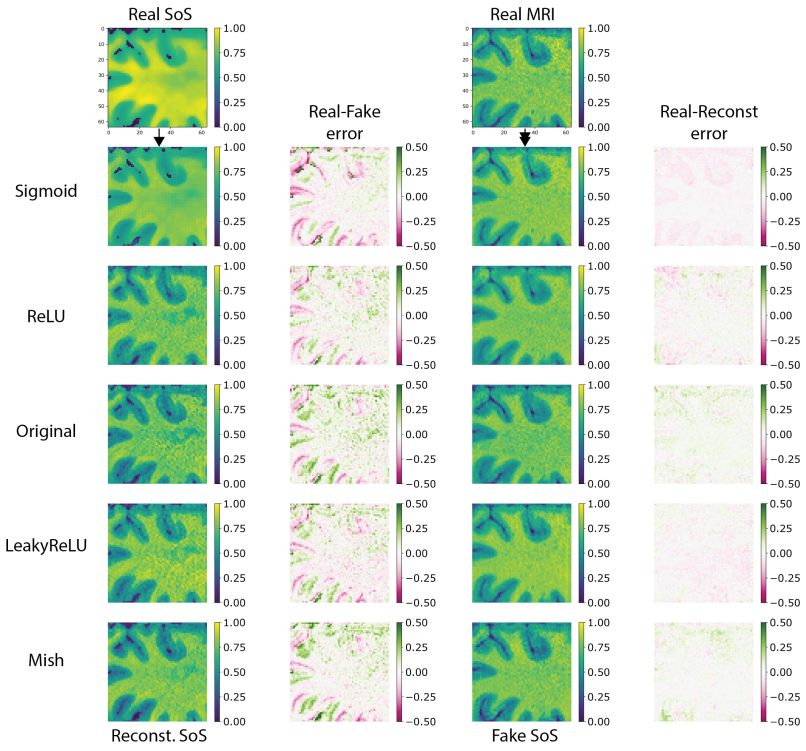

Figure 17: Visual assessment of images in the MRI domain, after undergoing one translation (Fake Images) and 2 translations (Reconstructed Images). Each row displays results from using different activation functions

3.7 MRI to CT

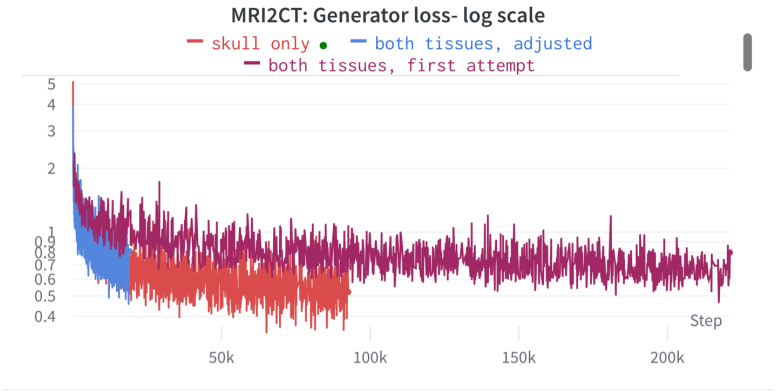

Figure 18: Comparison of training loss over time for models applied on mixed-tissue and skull-only images

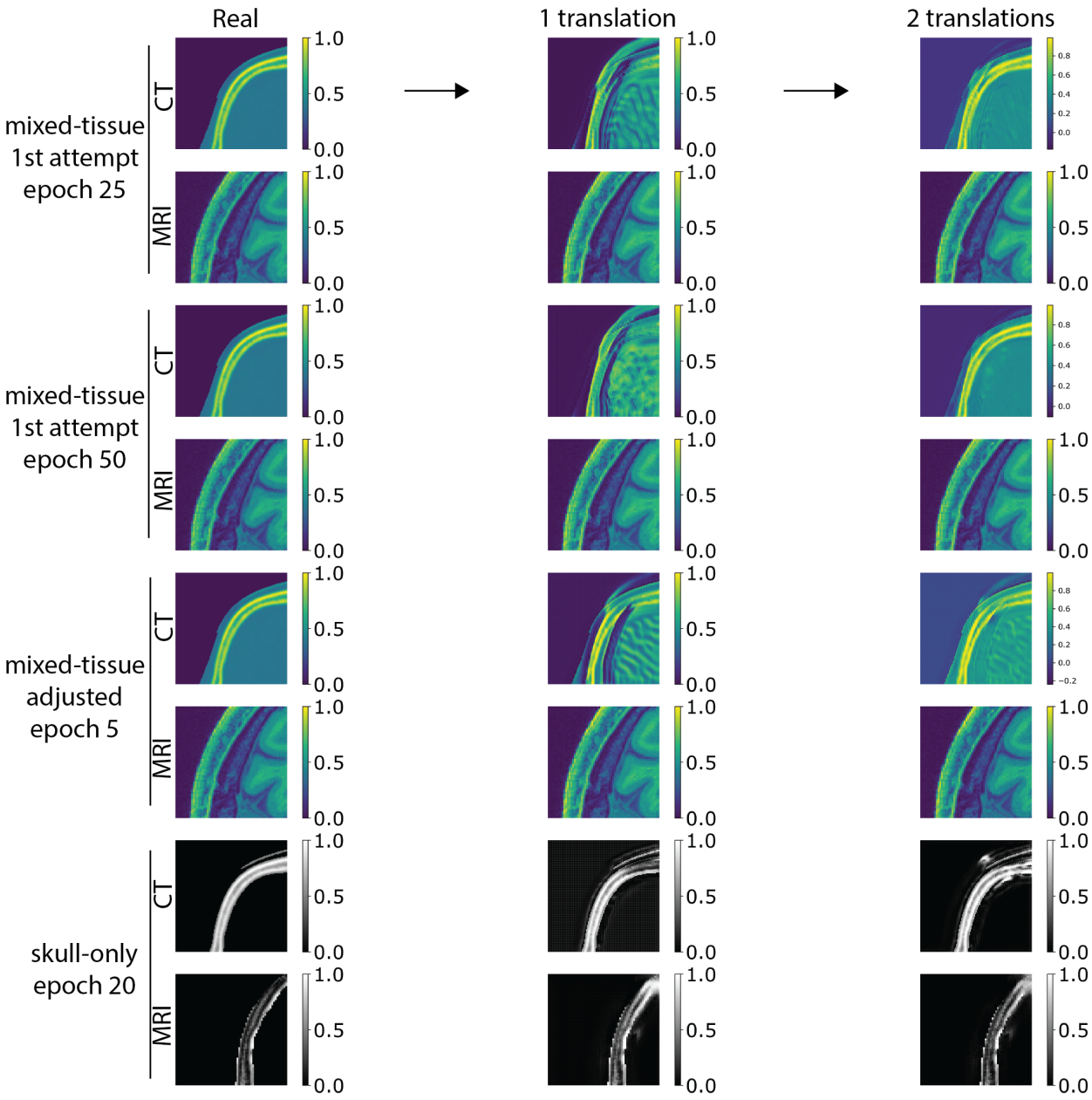

Figure 19: Visual assessment of MRI to CT translation results at different stages of the iterative process

Model number	batch size	downsampling layers	residual blocks	learning rate	β generator	β discriminator
1	2	1	5	2×10^{-5}	0.9	0.9
2	4	4	5	2×10^{-4}	0.9	0.9

Table 3: Hyperparameters used in the 2 models trained on mixed-tissue

4 Discussion

4.1 Batch size and learning rate

- **Artefacts at large batch sizes:**
  The generator implements transposed convolution to upsample input images, which can have uneven overlaps, appearing as checkerboard artifacts in the output images seen in Figure 5 It is possible that overfitting is exacerbating this issue, meaning that for large batch sizes the model is learning the specific patterns of the training data. This decreased ability of the trained network to generalize on new data could also be explained by looking at the type of minimizers that are converged to. Numerical evidence suggests that large batch sizes tend to converge to sharp minimizers of the training function, meaning that the function increases rapidly in a small neighborhood.[26] This would explain why the training losses for the large batch sizes are at a similar level as the smaller ones at epoch 5 in Figure 4 but the images for large batch sizes contain artifacts.

- **Batch Size and Learning Rate Combinations**
  In the graph in Figure 4, there appears to be a small cluster forming of batch size 2 and learning rate at approximately 2×10^{-5} . These combinations have the lowest generator loss after five epochs, including a lower result than the original values (batch size =1, learning rate 10^{-4}. Initially, this may lead to concluding that bs=2, lr=10^{-5} is the optimal combination, however, the nature of stochastic gradient descent (SGD) should be considered. All the results are within a small range of 1.21-1.86. SGD involves frequent updates with a high variance that causes the loss function to fluctuate heavily,[27] as can be seen by the noisy WandB graph in Appendix 6.3. This fluctuation is highest for batch size 1. Also, the results for batch sizes 1,2 and 4 are only from 16 Optuna repeats. Now that batch size 2 with a learning rate of 2×10^{-5} has been identified as a potential improvement, more trials should be carried out at this learning rate for batch size of 1 to confirm whether the differences in the final loss value are just due to the variability.

4.2 Adam Optimiser Hyperparameters

Analysing the effects of the learning rate started with comparing the generator and discriminator losses for an extreme high: 1000 and low: 10^{-8}. The learning rate determines the step size of the weight updates during the gradient descent. This justifies the rapid drop loss in Figure 6. This also increases the variability of the loss, as the weights oscillate around the optimal value, unable to reach it because of the big steps. In opposition, low step sizes may lead to the networks never converging because of the slow training and vanishing gradient. Further, the training losses for learning rates in the common range were compared at each of the first 5 epochs (Figure 7). The smallest learning rate, 10^{-5} led to the smallest decrease in generator loss over the 5 epochs. Which confirms the theory. An anomaly can be observed in the general trend of the discriminator loss of models in the high-end range, 10^{-3} and $2{*}10^{-3}$. A possible cause is that even though the losses were averaged across runs, the losses might have been in a point of oscillation when they were saved, considering higher learning rates produce high variability. To conclude which of the tested learning rates is the most optimal, a scatter plot of the generator loss over discriminator loss for the 5^{th} epoch was plotted in fig. It can be clearly seen that a learning rate of 0.0002 leads to both the lowest generator and discriminator loss after 5 epochs. A similar analysis was conducted on the beta values (Figure 8). Comparing the training losses at each of the first 5 epoch, a strong conclusion can be drawn that when β_1 of the generator is larger than the one of the discriminator, the generator loss is higher and the discriminator loss is lower compared to the rest of the models. As increasing beta slows down training, this result shows that when the higher generator β, the discriminator trains too fast, and it is too good at detecting fake images. Trying to fool it, the generator starts creating artefacts in the images causing high losses. The scatter plot of the generator over discriminator loss after epoch 5 (Figure 9) helps visualise this result.

4.3 Depth

One first aspect highlighted by the depth effect investigation was that a minimum of 2 blocks are needed for the loss to converge (Figure 11). A second aspect was that the correlation between depth and loss found after the first epoch did not remain consistent, as the losses of the converging models evened out over the following epochs(Figure 10). This suggests that depth cannot be increased indefinitely to improve performance. To reduce computational effort, we are looking for the minimum number of blocks that gives satisfactory results. The model with 5 residual blocks resulted in the lowest mean and standard deviation across 4 runs after 5 epochs, so it was further compared with the deepest model, of 9 blocks. When tested on unseen data, the deeper model had a lower mean loss and a 4% lower standard deviation across the validation dataset (Table 2).This suggests that more residual blocks can make the networks less specific to the training data. Nevertheless, (Figure 12 showed that overfitting does

not occur for the 2 models over the first 5 epochs.

4.4 Downsampling

After checking that the losses converge (Figure 13), the fake and reconstructed images created with 0 to 4 downsampling layers were visually assessed. Figure (Figure 16) confirmed the expected effect: images created with 4 downsampling layers captured the overall structure of the brain tissue better, however, the image appeared blurred. The best-performing models had 1 and 2 downsampling layers, corresponding to encoded images of sizes 32x32 and 16x16 for an 64x64 input. In comparison, the original CycleGAN used 2 layers for an input of size 256x256. Using 1 layer on the medical data led to a closer approximation of the actual pixel values. The model with 2 layers appears to overestimate the values through the brain and have less sharp edges, but generate a more accurate shape. Therefore, the optimal downsampling depends on the application. Generally, the reconstructed images were superior in replicating the original image, which is expected due to the cycle-consistency loss from training, while the adversarial loss is less powerful in unsupervised training. Further, the image quality was objectively assessed with SSIM. Unlike the visual assessment and simple difference error calculation, higher SSIM values were mainly obtained for the model with 2 layers. This can be explained by the presence of small artefacts in the networks with less downsampling, which can be seen in the close-up images.

4.5 Activation functions

As can be observed in Figures 14 and 15, losses pertaining to the standard CycleGAN model converge quicker and with greater stability than the other models. This combination indicates a better computational performance as ReLU mitigates the vanishing gradient, and LeakyReLU introduces a small slope α which negates the dying ReLU problem as explained above. Another notable performance is that of the Sigmoid function model, whose losses converge slower but eventually have a lower value than the standard model in the fifth epoch. Due to its complexity, involving multiple division and exponentiation operations, the Sigmoid function becomes computationally expensive, resulting in its slow convergence. The SSIM assessment (Figure 16) indicates that images generated through the Sigmoid model perform best. As for the subjective visual assessment, the Sigmoid model seem to miss the salt and pepper noise effect noticeable in the real MRI images. As the losses in the sigmoid discriminator model show a quicker convergence than in the generator, potentially changing the activation function to sigmoid in the final MRI to CT model would be of interest to inspect how it impacts the synthesised image quality. Nevertheless, standard architecture is considered the optimal model in the analysis of activation functions as it combines computational efficiency and good image quality.

4.6 MRI to CT

The networks were first trained with the parameters showed in Table 3 The generator loss converged and had similar values with the successful MRI to speed-of-sound model. Looking at the images, the MRI generator was found to recognize and synthesize the brain area, and longer training proved to improve results. However, the CT generator did not make visible changes to the input image. As the network failed to detect high-level features, we increased the downsampling layers to 4. Besides, the batch size was increased to 4 to reduce variability in the loss. These adjustments lowered the generator loss, and the MRI generator started producing results earlier. However, the CT generator did not show changes. One major limitation that can cause network failure is the reduced dataset size and the misalignment of the MRI and CT images. Because the network cannot use spatial information to learn to differentiate between tissues, we agreed that mixed-tissue translation is not suitable with the available dataset, so we trained the network on skull-only images. Changes in the intensity ranges were noticed between the 2 domains. It should be noted that the test images were not unseen, as no data was withheld from training.

5 Conclusion

5.1 Summary

The bulk of our results focus on the hyperparameter tuning of MRI to sound speed translation. By splitting the hyperparameters into categories, we have been able to observe the effects of each in isolation. In many cases this began with testing extreme values, such as 0 Resnet layers, 0 downsampling layers and learning rates of 10^{-8} and 1000, which gave clear examples of what a failing network looks like. We have been able to verify that some of the hyperparameters in the original CycleGAN implementation are also optimal for MRI to sound speed translation, including using a learning rate of 2×10^{-4} (at a batch size of 1). We also have some alternatives which could help for MRI to CT, such as using a larger batch size to reduce variability or changing the activation function to Sigmoid in the downsampling and upsampling layers.

5.2 Future Work

The network's performance will be further analyzed for longer training of up to 100 epochs and adjusted based on the finding of the research. Once patch-based training proves successful, we will move on to MRI to CT translation of full slices. To improve results, data augmentation will be considered to artificially increase the dataset size, through image transformation techniques applied to the available data, such as contrast flip, rotations, or deep learning methods.[28] Regarding the reliability of results, statistical significance tests could be used to prove the effect of each hyperparameter change, and a more thorough analysis of the variability of the losses across the dataset (within the group) and across the different runs (between groups) should be conducted.

6 Appendix

6.1 Project Management Assessment

Our initial project plan is presented in the Gantt Chart from Figure 19. Conducting the proper research on CycleGAN and grasping how it works by replicating the horse to zebra example was indeed the lengthiest part of our project timeline, taking up to the end of January to be completed.

Replicating the MRI to SoS and altering parameters tasks were delayed until the end of March. Having one computer with the appropriate GPU we could run and test the models on, only one team member could use and run different models at a time, adding much unaccounted time to our initial plan. In addition, each model that was tested was trained several times (approximately 3-5 runs each) for accurate metrics, which also took much time we did not account for in the Gantt Chart.

Due to these delays, all the following tasks were also moved to a later date. To still be able to meet our deadlines, we interlaced altering the parameters with writing the report and running the MRI to CT.

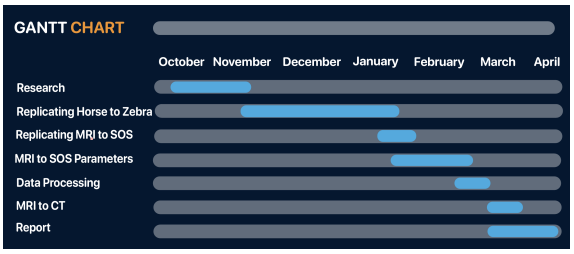

Figure 20: Project progress plan

6.2 Key Project Management Lessons

1. Flexibility

   At the start of the project, none of us were familiar with machine learning, so even though we planned and set deadlines, we were unable to anticipate the specifics of what each task entailed. Various unexpected obstacles slowed down our progress.

   Therefore, flexibility is one of the key lessons from this project, as we had to adjust our approach to take these difficulties into account. For example, after individually trying to implement previous examples of CycleGANs, it became apparent that we first needed to spend time with our supervisor learning about the fundamental architecture of the CycleGAN and how to build the different layers from scratch.

2. Delegation

   Knowing when it is best to work as a group and when it is more efficient to divide the work is a distinction that this project has made apparent. Always working as a group is slow but it is important that everyone has a good understanding of the whole project, which is hard to get if everyone is working on something separately. As a result, we learned to identify whether delegation or group work is more appropriate for a given task. For example, when learning about CycleGANs we had group coding sessions to ensure we all understood the fundamentals, whereas when we made the mind map for our assessment, we assigned research areas individually before coming together to share our findings.

3. Strategic Approach

   The iterative approach was very useful for our team, as none of us was familiar with machine learning. Starting our project by replicating the results of previous projects was helpful, as we gained a better understanding of CycleGANs, without wasting time and resources.

6.3 Additional Results

Some additional results can be found in Figures 21 and 22 below.

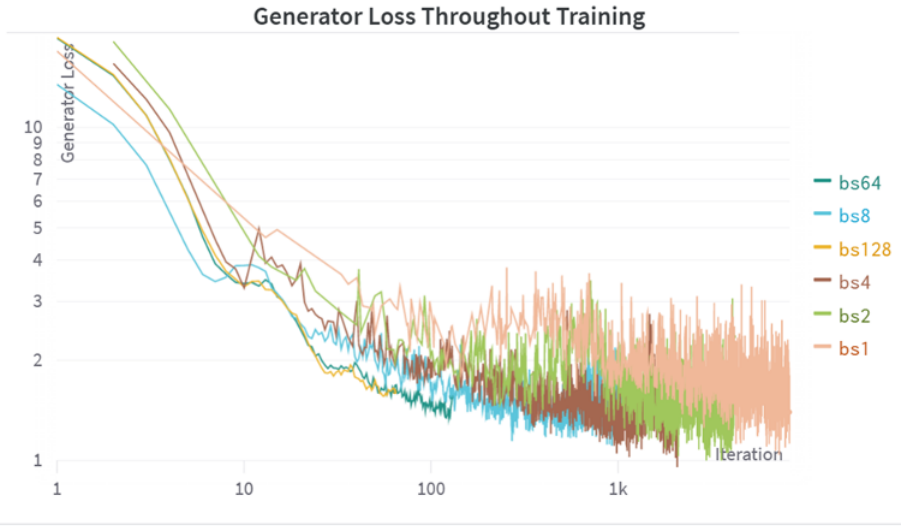

Figure 21: WandB generator losses for different batch sizes

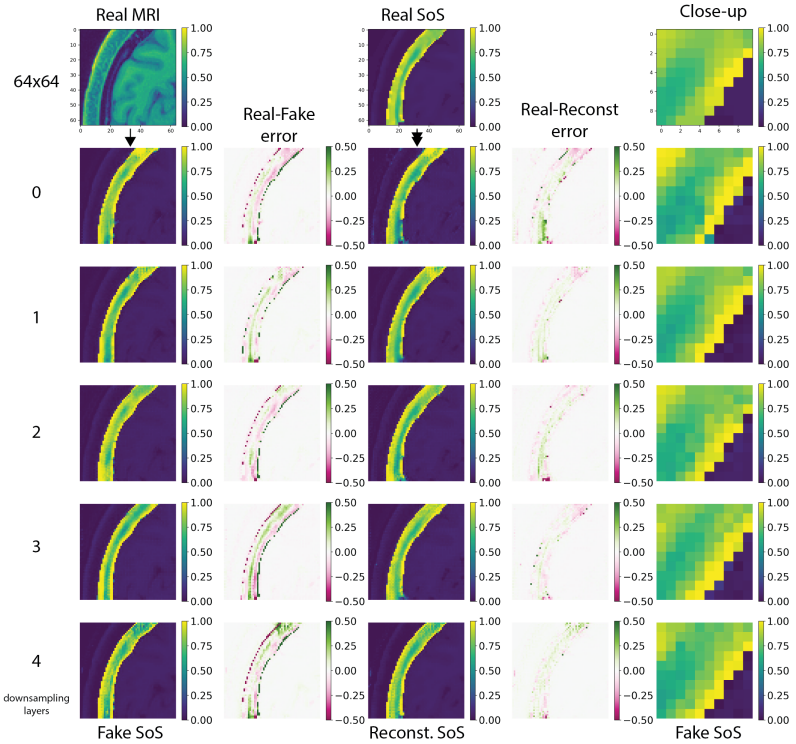

Figure 22: Visual assessment of images in the SoS domain, after undergoing one translation (Fake Images) and 2 translations (Reconstructed Images)

Additional information about the CycleGAN code applied in each stage can be found in the form of a collection of Jupyter Notebook files on the Github repository of this project which can be accessed by clicking on GitHub.

References

[1] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2242–2251.

[2] C. A. O. T. M. Guasch, L., "Full-waveform inversion imaging of the human brain." *Digit. Med.*, (2020). [Online]. Available: https://doi.org/10.1038/s41746-020-0240-8

[3] M. Brudfors, J. Ashburner, P. Nachev, and Y. Balbastre, "Empirical bayesian mixture models for medical image translation," in *Simulation and Synthesis in Medical Imaging*. Springer International Publishing, 2019, pp. 1–12. [Online]. Available: https://doi.org/10.1007%2F978-3-030-32778-1_1

[4] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2020.

[5] J. M. Wolterink, A. M. Dinkla, M. H. F. Savenije, P. R. Seevinck, C. A. T. van den Berg, and I. Isgum, "Deep mr to ct synthesis using unpaired data," 2017.

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[8] D. C. V. Essen and M. F. Glasser, "The human connectome project: Progress and prospects," *Cerebrum*, 2016. [Online]. Available: ttps://www.humanconnectome.org/

[9] Human Connectome Project, "Data use terms: Hcp open access," https://www.humanconnectome.org/storage/app/media/documentation/s1200/HCPDataUseTerms-OpenAccess.pdf, 2013, accessed on April 13, 2023.

[10] S. T. M. B. N. G. C. V. K. V. V. M. P. R. Sasank Chilamkurthy, Rohit Ghosh and P. Warier, "Development and validation of deep learning algorithms for detection of critical findings in head ct scan." [Online]. Available: http://headctstudy.qure.ai/#dataset

[11] Creative Commons, "Creative commons — attribution-noncommercial-sharealike 4.0 international," https://creativecommons.org/licenses/by-nc-sa/4.0/, 2013, accessed on April 13, 2023.

[12] A. Karpathy, J. Johnson, and L. Fei-Fei, "Cs231n convolutional neural networks for visual recognition," *Neural Networks*, vol. 1, no. 1, 2016. [Online]. Available: https://cs231n.github.io/

[13] D. R. Wilson and T. R. Martinez, "The general inefficiency of batch training for gradient descent learning." *Neural Networks*, vol. 16, no. 10, pp. 1429–1451, 2003. [Online]. Available: http://dblp.uni-trier.de/db/journals/nn/nn16.html#WilsonM03

[14] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Muller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. Berlin: Springer, 2012, pp. 9–48. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-35289-8_2

[15] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," 2018.

[16] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," 2020.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[18] E. Nielsen, N. O. Hansen, and A. Tierney, *Automated Deep Learning Using Neural Network Intelligence: Develop and Design PyTorch and TensorFlow Models Using Python*. Springer, 2020. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4842-5956-5

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[20] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222008426

[21] A. K. Bhoi, P. K. Mallick, C.-M. Liu, and V. E. Balas, Eds., *Bio-inspired Neurocomputing*. Springer Singapore, 2021. [Online]. Available: https://doi.org/10.1007%2F978-981-15-5495-7

[22] D. Misra, "Mish: A self regularized non-monotonic activation function," 2020.

[23] Weights & Biases, "Weights & biases - developer tools for ml," https://wandb.ai/, 2023, accessed on April 13, 2023.

[24] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.

[25] Y. Pang, J. Lin, T. Qin, and Z. Chen, "Image-to-image translation: Methods and applications," 2021.

[26] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," 2017.

[27] L. Bottou, "Stochastic gradient descent tricks," *Neural Networks: Tricks of the Trade*, pp. 421–436, 2012, accessed on April 13, 2023. [Online]. Available: https://bottou.org/papers/Bottou-tricks-2012.pdf

[28] V. Sandfort, K. Yan, P. J. Pickhardt, R. M. Summers, and A. K. Hara, "Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks," *Sci Rep*, vol. 9, no. 1, p. 16884, 2019. [Online]. Available: https://doi.org/10.1038/s41598-019-52737-x