Human Cranium, Brain Ventricle and Blood Detection Using Machine Learning on Ultrasound Data

William Thomas

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science In Computer Science & Software Engineering

University of Washington

2022

Committee:

Erika Parsons

Pierre Mourad

Michael Stiber

Yang Peng

Program Authorized to Offer Degree:
Computer Science & Software Engineering

University of Washington

Abstract

Human Cranium, Brain Ventricle and Blood Detection
Using Machine Learning on Ultrasound Data

William Otto Thomas

Chair of the Supervisory Committee:
Dr. Erika Parsons
Computer Science & Software Engineering

Any head related injury can be very serious and may be classified as a traumatic brain injury (TBI), which can be a result of intracranial hemorrhaging. TBI is one of the most common injuries in or around a battlefield, which can be caused by both direct and indirect impacts. While assessing a brain injury in a well-equipped hospital is typically a trivial task, the same cannot be said about a TBI assessment in a non-hospital environment. Typically, a computer tomography (CT) machine is used to diagnose TBI. However, this project demonstrates the use of ultrasound and how it can be used to predict where skull, ventricles, and bleeding occur. The Pulsatility Research Group at the University of Washington has conducted three years of data collection and research to create a procedure that diagnoses TBI in a field situation. In this paper, machine learning (ML) methodologies will be used to predict these CT derived features. The result of this research shows that with adequate data and collection methods skull, ventricles, and potentially blood can be detected while applying ML to ultrasound obtained data.

Chapter 1. Introduction ..... **Error! Bookmark not defined.**

1.1 Background ..... **Error! Bookmark not defined.**

1.2 Common Terminology ..... **Error! Bookmark not defined.**

1.3 Previous Work ..... **Error! Bookmark not defined.**

1.4 Problem Statement and Scope ..... **Error! Bookmark not defined.**

1.4.1 Data Processing ..... **Error! Bookmark not defined.**

1.4.2 Skull Detection ..... **Error! Bookmark not defined.**

1.4.3 Ventricle Detection ..... **Error! Bookmark not defined.**

1.4.4 Blood Detection ..... **Error! Bookmark not defined.**

Chapter 2. Related Works ..... **Error! Bookmark not defined.**

2.1 Overview ..... **Error! Bookmark not defined.**

2.2 Existing Data ..... **Error! Bookmark not defined.**

2.2.1 Data Collection Tools ..... **Error! Bookmark not defined.**

2.2.2 Collection Methodology ..... **Error! Bookmark not defined.**

2.2.2.1 Ultrasound ..... **Error! Bookmark not defined.**

2.2.2.2 Computed Tomography Scan ..... **Error! Bookmark not defined.**

2.2.3 Data in Depth ..... **Error! Bookmark not defined.**

2.3 Related Experiments ..... **Error! Bookmark not defined.**

2.3.1 Deep Learning Methods to Identify Intercranial Hemorrhage using Tissue Pulsatility Ultrasound Imaging ..... **Error! Bookmark not defined.**

2.3.2 Generative Adversarial Networks ..... **Error! Bookmark not defined.**

Chapter 3. Methodology .....**Error! Bookmark not defined.**

3.1 Project Architecture .....**Error! Bookmark not defined.**

3.1.1 Hardware System Diagram .....**Error! Bookmark not defined.**

3.1.2 Data Collection and Processing System Diagram .....**Error! Bookmark not defined.**

3.1.3 Machine Learning and Data Processing System Diagram..... **Error! Bookmark not defined.**

3.1.4 System Architecture as a Whole .....**Error! Bookmark not defined.**

3.2 Data Processing.....**Error! Bookmark not defined.**

3.2.1 bMode and Displacement Processing .....**Error! Bookmark not defined.**

3.2.2 Skull and Ventricle Processing .....**Error! Bookmark not defined.**

3.2.3 Blood Processing .....**Error! Bookmark not defined.**

3.2.4 Preparing Data for Training .....**Error! Bookmark not defined.**

3.3 Synthetic Data Generation .....**Error! Bookmark not defined.**

3.4 Detection .....**Error! Bookmark not defined.**

3.4.1 Skull Detection.....**Error! Bookmark not defined.**

3.4.2 Ventricle Detection .....**Error! Bookmark not defined.**

3.4.3 Blood Detection .....**Error! Bookmark not defined.**

3.5 Result Analysis .....**Error! Bookmark not defined.**

Chapter 4. Experiments and Results .....**Error! Bookmark not defined.**

4.1 Data Generation .....**Error! Bookmark not defined.**

4.1.1 Experiment Setup.....**Error! Bookmark not defined.**

4.1.2 Training.....**Error! Bookmark not defined.**

4.1.3 Results ..... **Error! Bookmark not defined.**

4.1.3.1 bMode StyleGan2-ada ..... **Error! Bookmark not defined.**

4.1.3.2 bMode Bleed StyleGan2-ada ..... **Error! Bookmark not defined.**

4.2 Skull and Ventricle Detection ..... **Error! Bookmark not defined.**

4.2.1 Experiment Setup ..... **Error! Bookmark not defined.**

4.2.2 Training ..... **Error! Bookmark not defined.**

4.2.3 Results ..... **Error! Bookmark not defined.**

4.2.3.1 Skull Detection ..... **Error! Bookmark not defined.**

4.2.3.2 Ventricle Detection ..... **Error! Bookmark not defined.**

4.3 Blood Detection ..... **Error! Bookmark not defined.**

4.3.1 Experiment Setup ..... **Error! Bookmark not defined.**

4.3.2 Training ..... **Error! Bookmark not defined.**

4.3.3 Results ..... **Error! Bookmark not defined.**

4.3.3.1 bMode Blood Detection ..... **Error! Bookmark not defined.**

4.3.3.2 Displacement Blood Detection ..... **Error! Bookmark not defined.**

Chapter 5. Discussion and Conclusion ..... **Error! Bookmark not defined.**

5.1 Conclusion ..... **Error! Bookmark not defined.**

5.1.1 Synthetic Data Generation ..... **Error! Bookmark not defined.**

5.1.2 Skull and Ventricle Detection ..... **Error! Bookmark not defined.**

5.1.3 Blood Detection ..... **Error! Bookmark not defined.**

5.2 Limitations ..... **Error! Bookmark not defined.**

5.2.1 Data ..... **Error! Bookmark not defined.**

5.2.1.1 Collection Method ..... **Error! Bookmark not defined.**

5.2.1.2 Sample Size.....**Error! Bookmark not defined.**

5.2.1.3 Complexity.....**Error! Bookmark not defined.**

5.3 Future Work.....**Error! Bookmark not defined.**

APPENDIX A:.....**Error! Bookmark not defined.**

1 LIST OF FIGURES

Figure 2.1: The Terason Tablet Device[5] 7
Figure 2.2: Transducer Locations When Collecting Data with the Ultrasound Device 8
Figure 2.3: Transducer Rolling to Capture Data 9
Figure 2.4: Example of a Conditional GANs Discriminator Seeing Input [7] 13
Figure 2.5: Pix2Pix Aerial photo to outline and vice versa [7] 14
Figure 3.1: High Level Architecture 15
Figure 3.2: CT Workflow 17
Figure 3.3 Ultrasound Hardware Diagram 18
Figure 3.4 Data Processing Architecture by John Kucewicz 20
Figure 3.5 Patient Data Collection Flow 20
Figure 3.6 Machine Learning and Data System Diagram 22
Figure 3.7 System Architecture High Level 23
Figure 3.8: Blank Appearing Data Due to Not Normalizing 25
Figure 3.9: Normalized CT Data 26
Figure 3.10: Demonstrates a post processing image of bMode, where color indicates an object inside the head 27
Figure 3.11: Demonstrates a post processing image of displacement 27
Figure 3.12: skullMaskThick data following skull data processing 28
Figure 3.13: ventricleMaskThick data following data processing 29
Figure 3.14: Contains 2 examples of skullMask and bMode training pairs 32
Figure 3.15: Contains 2 examples of ventricle bMode training pairs 33
Figure 3.16: Contains 2 example of ventricles and displacement training pairs 33
Figure 3.17: Before and After BloodMaskThick was layered on bMode 34
Figure 3.18: Paired bModeBlood and bMode Image 34
Figure 3.19: Paired bleed and displacement data 34
Figure 3.20: Nvidia Stylegan2-ada architecture and Randomness Scale (p)[8] 36
Figure 3.21: Stylegan2-ada Generator architecture for 256x256 images [8] 36
Figure 3.22: Stylegan2-ada Discriminator architecture for 256x256 images 37

Figure 3.23: An example of a patient's skull as shown from CT to bMode. In this example, the patient skull is clearly visible in the bMode data. 38
Figure 3.24: An example of a patient's skull as shown from CT to bMode. In this example, the patient skull is no visible in the bMode data. 39
Figure 3.25: Non-visible Ventricles in bMode 41
Figure 3.26: Potentially Visible Ventricle in Displacement 41
Figure 3.27: bMode Bleed and bMode Pair 42
Figure 3.28: bloodMaskThick and Displacement Pairing 42
Figure 3.29: FID and Disturbance Level[17] 46
Figure 4.1: Initial StyleGan2-ada Generated bMode Images. 50
Figure 4.2: Final StyleGan2-ada Generated Fakes. 51
Figure 4.3: Original bMode Color Histogram 52
Figure 4.4: Generated bMode Color Histogram. 53
Figure 4.5: StyleGan2-ada bMode FID. 54
Figure 4.6: StyleGan2-ada bMode Discriminator Loss 55
Figure 4.7: StyleGan2-ada bMode Generator Loss 55
Figure 4.8 Augmentation Probability while Training. 56
Figure 4.9: StyleGan2-ada bModeBleed 57
Figure 4.10: StyleGan2-ada bModeBleed Final Produced Images 58
Figure 4.11: StyleGan2-ada bModeBleed FID Score 59
Figure 4.12: StyleGan2-ada bModeBleed Discriminator Loss 59
Figure 4.13: StyleGan2-ada bModeBleed Generator Loss 60
Figure 4.14: StyleGan2-ada bModeBleed Augmentation Probability 60
Figure 4.15: Skull Detection MSE 63
Figure 4.16: Skull Detection RMSE 64
Figure 4.17: Skull Detection MS-SSIM 65
Figure 4.18: Skull Detection FID (y) at each Epoch (x) 65
Figure 4.19: Skull Detection Fake Image Loss 66
Figure 4.20: Skull Detection Real Image Loss 66

Figure 4.21: Skull Detection where left is bMode, middle is target/label, and right prediction 67
Figure 4.22: Skull Detection where left is Synthetically generated bMode by StyleGan2-ada, and the right is Pix2Pix model prediction..... 68
Figure 4.23: Ventricle Detection MSE 69
Figure 4.24: Ventricle Detection RMSE..... 70
Figure 4.25: Ventricle Detection MS-SSIM 71
Figure 4.26: Ventricle Detection FID (y) at each Epoch (x) 71
Figure 4.27: Ventricle Detection Fake Image Loss 72
Figure 4.28: Ventricle Detection Real Image Loss..... 72
Figure 4.29: Ventricle Detection Input, Target, Prediction 73
Figure 4.30: bModeBlood MSE..... 77
Figure 4.31: bModeBlood RMSE..... 78
Figure 4.32: bModeBlood MS-SSIM 79
Figure 4.33: bModeBlood FID(y) at Epoch (x)..... 79
Figure 4.34: bModeBlood Training MSE..... 80
Figure 4.35 bModeBlood Training Generator Loss..... 80
Figure 4.36 bModeBlood Training Discriminator Loss 81
Figure 4.37: bModeBlood Detection for input, target, generated..... 82
Figure 4.38: Displacement Blood Detection FID(y) for Epoch (x)..... 84
Figure 4.39 Displacement Blood Detection Training MSE..... 84
Figure 4.40 Displacement Blood Detection Training Generator Loss 85
Figure 4.41 Displacement Blood Detection Training Discriminator Loss 85
Figure 4.42 Displacement Blood Detection MSE..... 86
Figure 4.43 Displacement Blood Detection RMSE..... 86
Figure 4.44 Displacement Blood Detection MS-SSIM 87
Figure 4.45: Displacement Blood Detection for Input, target, generated..... 88
Figure 5.1: bMode Generated Skull with Blurring..... 90
Figure 5.2: Displacement Ventricle Detection..... 91
Figure 5.3 Poor Displacement Blood Predictions..... 92

Figure 5.4 Poor bMode Blood Predictions 92
Figure 5.5: CT Derived Blood Mask Variable..... 100
Figure 5.6: Blood Mask in the Form of Ultrasound View 100
Figure 5.7 bloodMaskThick CT Data 101
Figure 5.8: bloodMaskThick in the Form of Ultrasound View 101
Figure 5.9: Ultrasound Collected bMode data 102
Figure 5.10: Ultrasound bMode Data as it would appear from the technician 102
Figure 5.11: Ultrasound collected data that has been normalized 103
Figure 5.12: Ultrasound collected data that is viewed as it may be seen by the technician103
Figure 5.13: CT collected BrainMask data 104
Figure 5.14: BrainMask data as it would appear from ultrasound..... 104
Figure 5.15: Full CT data of a patients head..... 105
Figure 5.16: Full CT Data as seen through ultrasound view 105
Figure 5.17: Displacement Data Collected by the Ultrasound Device 106
Figure 5.18: Displacement data as would be viewed from the ultrasound device..... 106
Figure 5.19: Normalized Displacement Data..... 107
Figure 5.20: Normalize Displacement in Form of Ultrasound View..... 107
Figure 5.21: CT Derived Skull Mask..... 108
Figure 5.22: Ultrasound Point of View for Skull Mask..... 108
Figure 5.23: Skull Mask Thick data..... 109
Figure 5.24: Skull Mask Thick viewed from Ultrasound 109
Figure 5.25: CT derived Ventricle Mask 110
Figure 5.26: Ventricle Mask depicted from Ultrasound view 110
Figure 5.27: ThickVentricleMask derived from CT data 111
Figure 5.28: ThickVentricleMask simulated by ultrasound view..... 111

LIST OF TABLES

Table 1.1: Glasgow Coma Score an Objective TBI assessment metric [4] 2
Table 3.1: TBI Data Usage 24
Table 3.2: Model and Datatypes Used 30
Table 3.3: Metrics for Model Analysis 44
Table 4.1: Machines used for training the Stylegan2-ada model 47
Table 4.2: StyleGan2-ada bMode Initial and Final Scores 54
Table 4.3: StyleGan2-ada bModeBleed Initial and Final Scores 57
Table 4.4: System Hardware used in Skull and Ventricle Detection 61
Table 4.5: Initial Skull and Ventricle Training Parameters 62
Table 4.6 Skull Detection Final Metrics 63
Table 4.7: Ventricle Detection Final Metrics 69
Table 4.8: Blood Detection Hardware 74
Table 4.9: Blood Detection Initial Training Parameters 75
Table 4.10: bMode Blood Detection Model Results 76
Table 4.11: Displacement Blood Detection Final Results 83
Table 5.1: Potential Issues with Data Collection 93

1 INTRODUCTION

1.1 BACKGROUND

Modern warfare has led to several new types of injures in comparison to previous wars, the most prevalent neurological injure being traumatic brain injures (TBI). Improvised explosive devices, artillery, and other explosion-based weapons have led to a higher rate of TBI in soldiers. The increased percentage of neurological injuries from the war in Iraq are closed head injuries caused by blasts [1]. Explosive-based neurological injuries require immediate medical attention. However, when left untreated or undiagnosed, the patient may suffer from long-term medical conditions [1]. Globally, 50 million people (about twice the population of Texas) will sustain a traumatic brain injury per year [2]. Improving the rapid diagnosis of neurological injuries may reduce the severity and likelihood of lifelong impacts of TBI. However, diagnosing neurological injuries or any sort of internal bleeding is incredibly difficult in a rapid, portable manner outside of a medical facility.

Traumatic brain injuries are certainly not new, but how we diagnose them has not changed much. Currently, to diagnose TBI, a patient is brought to a hospital where they are placed inside of a computed tomography (CT) or a magnetic resonance imaging (MRI) machine where imagining is done to analyze the brain for bleeding [2], [3]. Both a CT and MRI machine are incredibly large and non-portable, making them impossible to transport to and use in a remote setting like a battlefield or a car accident [4]. While imaging provides a more statistical approach to detecting bleeding, the Glasgow coma scale offers another means to give a shorthand style summary of the status of someone that may have TBI [5]. Table 1.1 outlines the three distinct types of responses necessary for detecting TBI and the score associated with the varying response scale. The GCS score is typically used to aid early assessment of an individual suspected of TBI.

The Department of Defense Pulsatility Project[1] at the University of Washington, aims to enable first responders to assess better and detect neurological injuries and create new means to see skull and ventricles. They are allowing first responders to utilize a portable ultrasound device to make better decisions on treating an individual's injury that may be suffering from TBI.

[1] For brevity purposes, going forward we will refer to this project as the TBI Project.

Table 1.1: Glasgow Coma Score, an Objective TBI assessment metric [5]

Response	**Scale**	**Points**
Eye Opening	Eyes do not open	1
	Eyes open to pain	2
	Eyes open to sound	3
	Eyes open spontaneously	4
Verbal	No verbal response	1
	Non coherent responses	2
	Incorrect words	3
	Confused	4
	Orientated	5
Motor Skill	No motor skills	1
	Abnormal extension to pain	2
	Abnormal flexion to pain	3
	Withdrawal from pain	4
	Localizing pain	5
	Obeys commands	6

1.2 Previous Work

This research is part of an ongoing project for which there is preexisting work but little success when it comes to detection of brain bleeding. In the past couple of years, efforts to detect blood have focused on an algorithmic approach with the use of common machine learning models.

There has also been considerable focus on understanding the data, e.g., how to reduce the number of dimensions in data, what data is important, and algorithms to change the data into something more meaningful. Using past models and current data, it may be possible to start working on new models that will have greater accuracy.

ML algorithms efforts to predict bleeds revolves around the use of time series data that was synced up with patient heart pulsations. In theory, utilizing the patient heartrate could allow for a higher percentage of blood at a given peak, but the dataset had limitations that might prevent such an approach from being effective. This problem can be approached from a computer vision perspective where algorithms such as ResNet can be used to predict these bleeds[6]. An initial study on this strategy identified that such an approach may not be suitable for detecting intracranial hemorrhaging. However, it is worth noting that these models may not have been the main issue but rather the data that was being used could have lacked information necessary to predict bleeding.

1.3 Problem Statement and Scope

As discussed in 1.1 there is an imperative need for a way of detecting head and brain related injuries that do not require a CT machine, or any large machinery. Creating a solution that can accurately predict, or lead to the prediction of TBI, could quite possibly save lives, which highlights the need for a solution. This research proposes a machine learning approach to detecting ultrasound[7] derived human head features, that is skull, ventricles, and blood. Section 1.3.1 through section 1.3.4 give a further breakdown of each goal in this study. This research refers to detection as both the ability to find the desired feature, but also to show its location and an approximation to what it looks like. That is to say, the output of our detection models should be an image that resembles the CT derived data.

1.3.1 *Data Processing*

The project has hundreds of Gigabytes of data associated with it. Work has been done over the last 4 years to collect data on 120 patients that either have or had TBI. This document will include a section detailing the existing data collection methodology, data descriptions, and an analysis of the data. The emphasis being which data was used in the project.

1.3.2 *Skull Detection*

Skull detection can be useful for additional or future work; hence, this task is formalized as an intermediate research goal. During the course of this research, a machine learning model was created to determine where skull is in an ultrasound (B-mode) image. The resulting image is a CT-like depiction of a skull mask.

1.3.3 *Ventricle Detection*

Similar to skull detection, ventricle detection can be useful for evaluating patients' health. Brain ventricles are a network of cavities that fill with spinal fluid. To identify ventricles given an ultrasound-based image, a machine learning model was created to identify or predict where they are in a patient. Computer vision plays a large part in determining where the ventricles are located.

1.3.4 *Blood Detection*

The foremost and motivating goal of this research has been to detect blood inside the skull, with a focus on injuries related to/or caused by combat situations. The brain, much like any other part of the body, can be injured, can start hemorrhaging and blood can start accumulating and/or clotting. Much like the other work being proposed in this study, computer vision and generative adversarial networks will be used to determine if and where a patient may be bleeding internally in their head.

1.4 RESEARCH CONTRIBUTION

There are three main contributions relevant to the success of this work. The following points scope these areas of contribution, which also fit within the context of the Pulsatility Research group's research goals:

Data Abstractions: Determine which datatypes are the most suitable for each detection method in section 1.3. This required investigation into the pre-existing datatypes. Extract and process the pre-existing data so that it may be used to explore new avenues of research.

Data Generation: Determine how data can be created to increase current dataset sizes. Research methods in generating synthetic data or models that could be used for generation and apply to the research project.

"Feature" Detection: Investigate different approaches to computer vision detection models and other classification models and apply them in the three aforementioned feature detection scenarios (skull, ventricles, blood). We will explore ideas on how current data and researched models could be used to detect and predict such features.

The work towards these contributions is explored in this thesis. We will present and discuss how various ideas, implementations and experiments lead us to determine that there are models that can successfully detect skull and ventricle features, while blood detection was not achieved through any of our strategies. In addition, we show how synthetic data with a high degree of realism can be generated for future research involving ML models.

This document is organized as follows: Chapter 2 contains related work information, Chapter 3 presents the methodologies used to investigate and implement algorithms, Chapter 4 concerns experiments and results discussions and Chapter 5 will present conclusions and future work.

2 RELATED WORKS

2.1 OVERVIEW

As an ongoing project, the TBI Project has related research, experiments and other works. In previous years, approximately 130 patients worth of data has been collected, processed, and analyzed by the TBI team. The collected data has been experimented with to help identify and highlight specific features, build visual aids, and much more. In the past, various experiments have been conducted to identify Intra-Axial (IA) hemorrhaging, Extra-Axial (EA) hemorrhaging, skull, ventricles, and midline shift[2].

2.2 EXISTING DATA

The TBI Project started approximately 4 years ago and data from patients that were suffering from suspected IA or IE started to be collected then.

2.2.1 *Data Collection Tools*

Data has been collected using various tools to ensure a wide spectrum of data points for projects. These tools include a Terason Ultrasound Device, a CT machine, and MATLAB.

- **Terason Ultrasound Device:** Depicted in Figure 2.1, transmits and receives short bursts of high frequency sound between 1 and 10 Megahertz. The sound frequencies propagate through the patient's skull and brain mass. Along the way, frequencies are scattered and lost, but some make their way back to the transducer after hitting various objects (skull, tissue, ventricles). Some parts of the brain slow the response of the frequency traveling to and back from the object it hits[8]. Dr. John Kucewicz explains the way tissue interacts with sound waves as the

> *"speed of sound in tissue is relatively constant, two-dimensional images of tissue structure can be created based on the amplitude of the received*

[2] Midline shift is defined as "Midline shift refers to a shift (displacement) of brain tissue across the centre line of the brain." [40]

ultrasound and the time between a sound burst's transmission and its reception."[9], [10]

Which means that images of blood and or tissue velocity can be interpreted by transmitting a minimum of 2 burst of 1-10 MHz sound and measuring the temporal shits in the received ultrasound.

- B-Mode: uses 2 transmissions to create a 2-dimensional image. In Figure 2.1, The image pixels represent where these reflections or waves bounced off, giving a basic image of the internal structure that is being scanned.

Figure 2.1: The Terason Tablet Device[7]

- **Computed Tomography (CT)**: is a machine that scans an individual that assists in creating images of areas inside the body being analyzed. The machine takes images at different angles which creates a 3-dimensional image of what is being scanned. In this research, a CT machine is used to analyze the patients head to determine where they are bleeding from.
- **MATLAB**[11]: is a powerful data science programming platform that is used by engineers, data scientists, and researchers all around the world. MATLAB supplies many frameworks and tools that allow individuals to build machine learning models, collect data, analyze data, and much more. In the case of TBI, this programming platform was used to stream

data directly from the Terason 3200T utilizing their built in API (Application Programing Interface) calls. Post stream and input, the data was changed and analyzed using various MATLAB packages.

2.2.2 *Data Collection Methodology*

Data collection for TBI revolved around the patient's needs, meaning that if someone needed immediate treatment, they were first administered aid prior to collecting ultrasound data. Meaning that some of the data may be skewed. For instance, in every case that there is suspected bleeding, the patient is sent to get an CT, however the ultrasound would not occur until the patient, or their family has given consent. There were 3 primary ways data was collected, ultrasound, CT, and an assessed Glasgow Coma Score. Each method supplies information that allows for either a baseline or future data and experimentation work. Next, we describe the basic manner in which ultrasound and CT are used to collect data:

Ultrasound: This method is meant to be used as exploratory work to develop a substantial and accurate way to assess head trauma. Ultrasound was collected in a manner which allowed for consistency across all patients. Scans for each patient are broken into their own MATLAB file for further investigating and modification The standard operating procedure (SOP) for collecting ultrasound images of a patient are as follows:

1. The transducer is placed above the ear as seen in Figure 2.2
2. The transducer is rolled as seen in Figure 2.3
3. Transducer gets rolled from angels between 1 and 179 degrees
4. Repeat steps 1-3 on the other side of the head

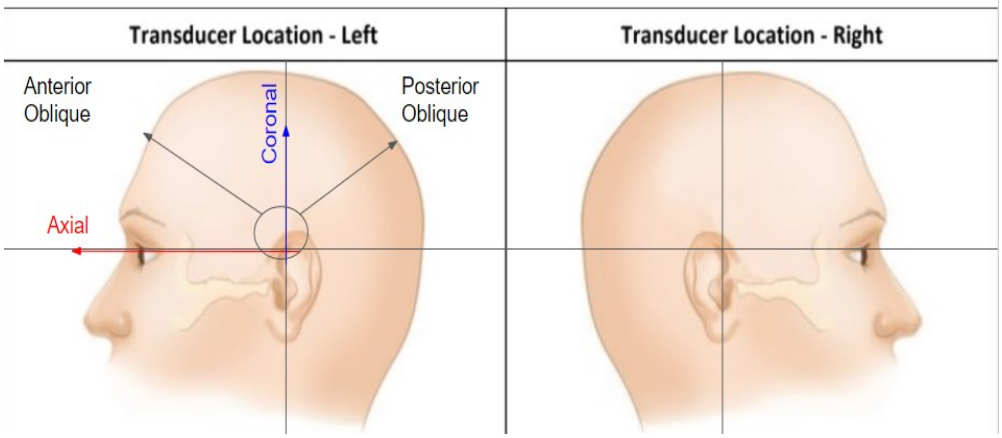

Figure 2.2: Transducer Locations When Collecting Data with the Ultrasound Device[9]

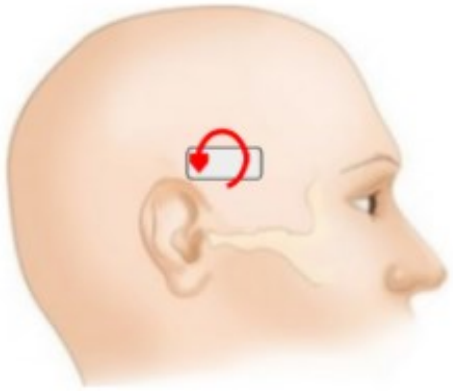

Figure 2.3: Transducer Rolling to Capture Data[9]

Computed Tomography (CT) Scan: Brain injuries often require CT machines to best evaluate the status of a patient. While not all patients who have CT images taken have TBI, some of them do, and that is why the CT data is collected. As a patient enters the hospital and is suspected of having a brain injury, they are immediately assessed and transferred to the CT machine where they can be further evaluated. The standard operating procedure for collecting magnetic resonance images can be summarized as:

1. Patient is admitted to hospital
2. Patient is transported to CT room
3. Patient is placed inside of CT for 8 minutes
4. Patients' results are read and acted upon

2.2.3 *Data in Depth*

Typically[3], a patient is scanned 30 times where 15 scans occur on their left, and another 15 scans on their right. Each side received 5 axial scans, 5 coronal scans, and 5 oblique scans. The angle that the subject was scanned at was derived from the CT data and where bleeds were visible from the ultrasound device. Files use the naming scheme based on their specific subject id, for example DoD001, where 001 would mean that this is patient 1. Associated image and MATLAB files take the naming schema DoDo###_Ter###_**#, for example, DoD001_Ter010_RO4 where the patient

[3] Note: data collection had to change during COVID-19 – with a limited number of scans taken from the right and left side of the head only.

id comes first, Ter010 refers to the scan number (in this case the 10^{th}), and RO4 refers to the image positioning, in this case RO stands for *right* axial *oblique* scan, and 4 is the instance of that scan taken in that specific position. Axial and coronal scans would receive RA4 and RC4, respectively. Each patient receives around 30 scans total (with a decreased number due to COVID-19). CT data was also collected for each patient. The CT data was registered to a slide-by-slide comparison, meaning that they are from the same location and should match up when looked at from a side-by-side comparison. Some data has been dubbed "thick" data, meaning that it is an average from a stack of the original data. Each patient data file contains:

- **bMode:** A 3-dimensional array of brightness indicating where there is tissue or some other object inside the brain. The data can be looked at as a stack of independent bMode images that are taken over a brief period of time. Figure 5.8 depicts an example of how the brightness images appear post processing, while Figure 5.9 shows how the image would look to the technician gathering the data.
- **bloodMask:** A blood mask for the brain that is derived from the CT data that gets registered and paired with the ultrasound image. Used in detecting blood in the ultrasound data. The data is either 0 or 1, 1 meaning that there is blood in the pixel. Figure Figure 5.5 both depict how this data appears, the first in the form of a CT slice, the second in a conical view of mocked bMode.
- **bloodMaskThick:** Multiple neighboring planes of bloodMask CT data that have been averaged into one. The pixel values range from 0 to 1 while 1 indicates blood being present. Essentially, this piece of data allows for multiple slices of CT to be stacked into one image, where you can view depth of bleed. It can be read like a topological map. Figure 5.6 shows how the thick blood mask data appears, while Figure 5.7 indicates a simulated view from an ultrasound device.
- **bModeNorm:** bMode data that has been normalized to the patient's cardiac cycle. Visualized as a 4-dimensional array of data where the last 2 dimensions are time within a cardia cycle and the cardia cycle itself. Figure 5.10 demonstrates how the normalized brightness image appears, while Figure 5.11 creates a more realistic view of the data.
- **brainMask:** Shows the patient's brain mass based on CT data that was registered to the ultrasound image. This data does not show blood, skull, or any other information. It is strictly extracted brain mass from the CT. Figure 5.12 depicts how the brain mask appears

after being processed, while Figure 5.13 shows how it would look from the viewpoint of ultrasound.

- **dataCT:** Full CT data that have been sliced and matched up with the ultrasound data. The data is represented in Hounsfield units with and offset of 1024. Figure 5.14 depicts the full CT data in the form of an image, while Figure 5.15 gives the perception of an ultrasound image that contains this data.
- **displacement:** A 3-dimensional matrix that consists of displacement in the form of millimeters. The data was augmented by use of a common mode motion correction and passed through a filter. Figure 5.16 shows an image that describes how the displacement appears after augmentation, while Figure 5.17 shows the augmented data as it would have been seen in the ultrasound device.
- **discplacementNorm:** Regular displacement data that is broken into cardiac cycles to fit the number of sample sizes. Figure 5.18 shows how the cardiac cycle normalized displacement data appears, while Figure 5.19 indicates the view from an ultrasound device.
- **hr:** The heart rate during each scan in beats per second (bps)
- **hrTimes:** The estimated time for beginning and ending for a cardiac cycle while being scanned. The data is an estimation based on displacement of brain mass with blood.
- **skullMask:** Is CT derived data that contains where the patient's skull is. The data is registered to the specific ultrasound scan. The data is represented in a 2-dimensional array where the pixel values are either 0 or 1, 1 being skull. Figure 5.20 gives an example of skullMask data, while Figure 5.21 depicts how it appears from the ultrasound view.
- **skullMaskThick:** Is CT derived that contains the average of a patient's skull. The data is registered to the specific ultrasound scan. The data is represented in a 2-dimensional array that contains pixel values between 0 and 1. The skull mask thick data can be read like a topological map, where darker sections are more concentrated levels of skull. Figure 5.22 shows an example of skullMaskThick data post processing, while Figure 5.23 lines up the data to be viewed from an ultrasound device.
- **timeAxis:** Array containing the timestamps in seconds for each of the ultrasound frames.
- **transPos:** A coordinated position (*x, y, z*) and orientation of the transducer (roll, pitch, yaw) in CT space.

- **ventMask:** A CT mask containing ventricles that is registered to the patient. The values are 0 and 1, 1 being ventricle. Figure 5.24 indicates where a ventricle was found in the form of an image, while Figure 5.25 depicts the ventricle mask simulated through an ultrasound view.
- **ventMaskThick:** A CT derived averaged ventricle mask that is registered to the patient. The values range from 0 to 1. Closer to 1 being more densely found ventricle. Much like previous thick masks, the data can be read where it is more intense, there is more ventricle. The image in Figure 5.26 depicts an example of where ventricles are, while Figure 5.27 indicates where they are when simulated by an ultrasound view.
- **xAxis:** Size of the *x* axis in all collected data
- **zAxis:** Size of the *z* axis in all collected data

2.3 Related Experiments

Due to the continuing aspect of the TBI Project, it is important to note that over time, some experiments have been less related than others to the work discussed in this Thesis. In addition, the data provided in this research project has been used for various purposes for which they were not originally intended. While the data was initially collected for brain related bleed detection, the dataset was repurposed into other projects. Bleed detection has been the prominent focus of the TBI research, but there has not been any substantial progress towards developing a realistic bleed detecting algorithm due to the nature of the data and problem.

2.3.1 *Deep Learning Methods to Identify Intercranial Hemorrhage using Tissue Pulsatility Ultrasound Imaging*

Concurrent work on this project was done that focused on the same areas as this paper, however, the approaches taken were vastly different. The goal of this related research was to develop segmentation models that can be used to find intracranial hemorrhaging caused by TBI, skull, and ventricles. The models used include U-Net, U-Net++, Attention U-Net, and some use of Cascade models[12]–[15]. U-Net based models have been successfully used in the area of medical imaging. Preliminary experiments indicated that U-Net can effectively segment the skull. U-Net++ can segment brain tissue using a high *dice score*, a metric commonly used in image classification with

deep learning, ranging from 0 to 1 (where 1 corresponds to a pixel perfect match between the model output and a given ground truth annotation)[16]**.** Cascade model ideas will be applied to the task of locating ventricles.

2.3.2 *Generative Adversarial Networks*

Generative Adversarial Networks or GANs can be used to transform images from one type into another, often called image to image translation[17], [18]. A GAN contains a Generator (G) and a discriminator (D), where the generators primary focus is to learn how to transform an input into specified output. The discriminators primary focus is to determine whether the generators images are comparable to the actual paired output image. A conditional GAN or cGAN learns a conditional generative model, meaning that the network can be constructed by feeding data *y* into G and D, making it particularly good for image-to-image translation[18]. A cGAN can essentially pass in the input label to the D, whereas in a typical non conditional GAN the discriminator does not receive that information. In Figure 2.4 a cGAN is training on shoe outlines and colored in shoes, in this example, the discriminator is also fed the shoe outline.

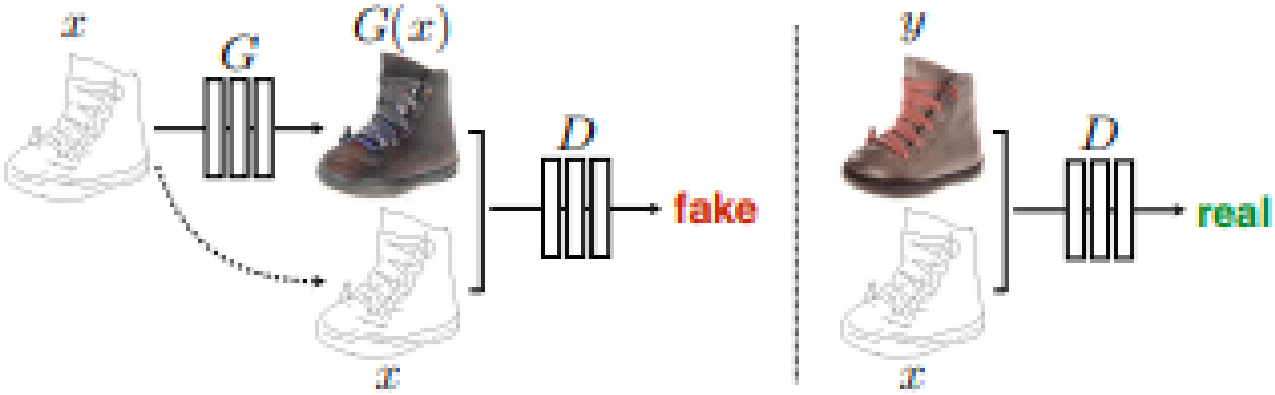

Figure 2.4: Example of a Conditional GANs Discriminator Seeing Input from Image to Image Translation Paper[18]

Pix2Pix[18] This model utilizes paired data to solve an image-to-image translation problem, for example, taking a satellite image of a city and creating an outline of all the roads as seen in Figure 2.5. To create this model, one would need examples of the output and the input so that the model can create a mapping between the two. Traditionally, a GAN doesn't have a loss function, but in

the Image-to-Image whitepaper, they utilized a cGAN that used both L1 and L2 loss, which created a function as $G^* = \arg\min(\mathrm{G})\max(\mathrm{D})\mathrm{LcGAN}(\mathrm{G},\mathrm{D}) + \lambda\mathrm{LL1}(\mathrm{G})$ [18]. The Image-Image research paper noted that there is a higher success when using loss function to guide the generator and discriminator to beat each other. Additionally, loss functions provided a better means of refining the quality of images, making them appear more as their target.

Figure 2.5: Pix2Pix Aerial photo to outline and vice versa [18]

StyleGan2-ada[19] GANs can take a very large amount of data to train, particularly when more complex image to image translations are needed. Using too little data will most often lead to the discriminator overfitting which destabilizes the model and can lead to mode collapse[19]. StyleGan2-ada, created by scientists and engineers at NVIDIA, allowed for generation of synthetic images using smaller datasets, a task that previously required a significant amount of data. In chapter 3.3, this model will be elaborated on and its use cases pertaining to this project.

3 METHODOLOGY

3.1 PROJECT ARCHITECTURE

When designing a system, it is important to understand all the components that will be necessary. In TBI's case, there is a lot of data that needs to be collected, processed, and sent to their proper locations before they reach any machine learning algorithms. Previous contributions have been made to data collection and processing, but still requires further refinement and augmentation before making it to the network. For that reason, the project architecture will be split into subcomponents titled hardware, data processing, machine learning data processing, and machine learning model. Hardware and data processing were preexisting systems that are relevant to understanding the later stages of the projects architecture. This thesis work will follow the flow depicted in Figure 3.1, which portrays the high-level architecture of the TBI project. While this thesis work did not include data collection, some parts of it will be described in this chapter in order to provide a deeper understanding of the subject from a high-level perspective, as well as setting context for analysis and conclusions. The following subsections present a break-down of relevant aspects of the architecture, such as the hardware, data operations and ML.

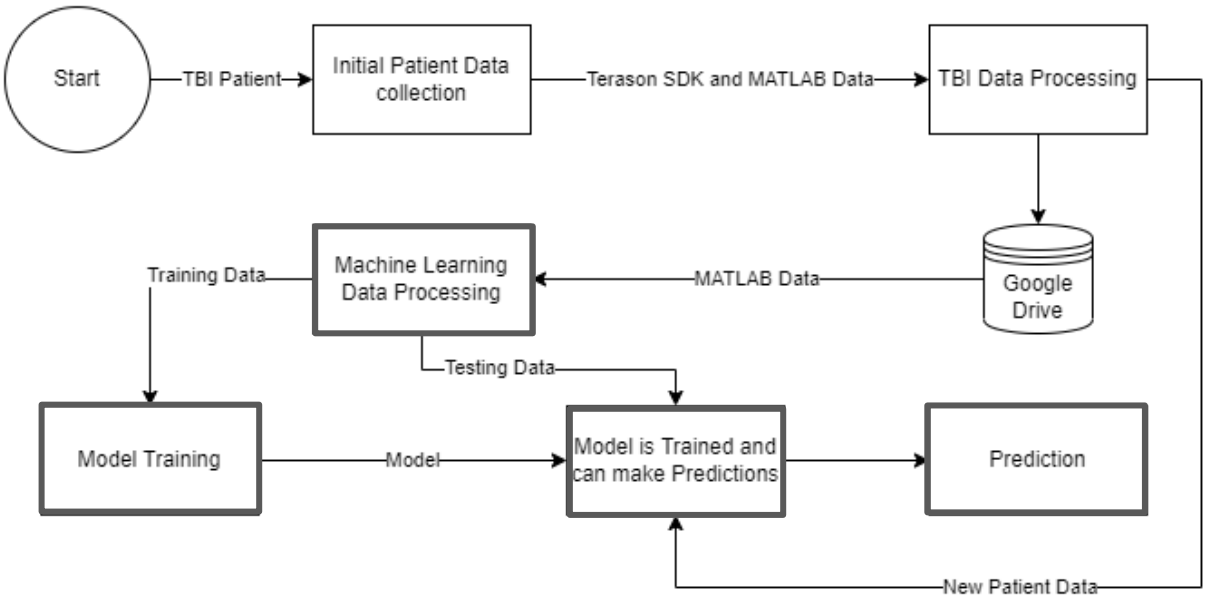

Figure 3.1: High Level Architecture: The emphasized blocks (and associated arrows), indicate the focus work of this thesis. Data collection was a pre-defined process.

3.1.1 *Hardware System*

There were two primary hardware workflows for the Pulsatility project, the first being CT hardware setup and the second ultrasound hardware setup. The CT setup was done in hospitals where patients were admitted and sent to get processed. Figure 3.2 depicts the system diagram for CT related collections. The figure assumes that the patient has been evaluated and does have TBI prior to making the collection. In the diagram, the CT machine scans the patient and collects data. After the initial collection, the data is processed and broken into the previously mentioned masks and datatypes. After processing, the data is stored and used for detection purposes. Figure 3.3 depicts the ultrasound data collection. The hardware setup for this collection method utilizes a Terason Ultrasound device and transducer to collect data. Figure 3.4 elaborate further on the specifics of collection. While the data is being collected, it is read in real time by a computer and processed. Following the data collection and processing, the CT and ultrasound data are registered together, that is matching them side by side. In other words, if there a CT and ultrasound image are paired, it will represent the same angle and location as the other.

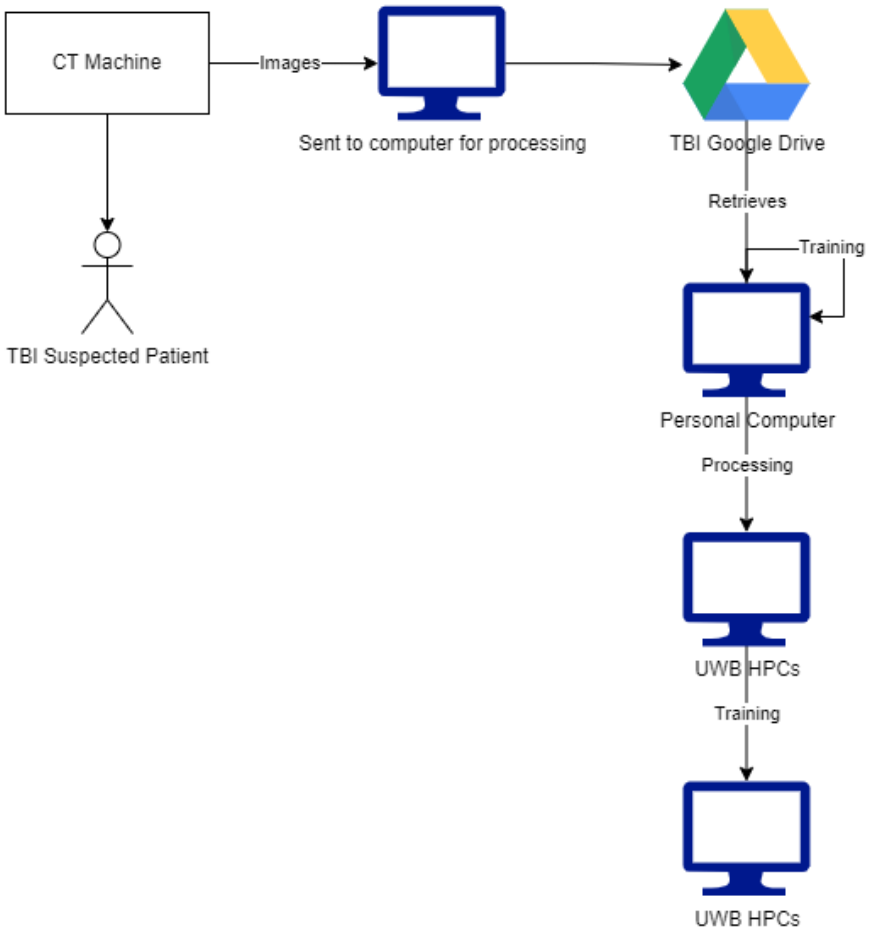

Figure 3.2: CT workflow: from data collection and processing (in a hospital setting) to training and testing in a computationally powerful machine. Main focus of this work starts from the data retrieval from the team's Google Drive.

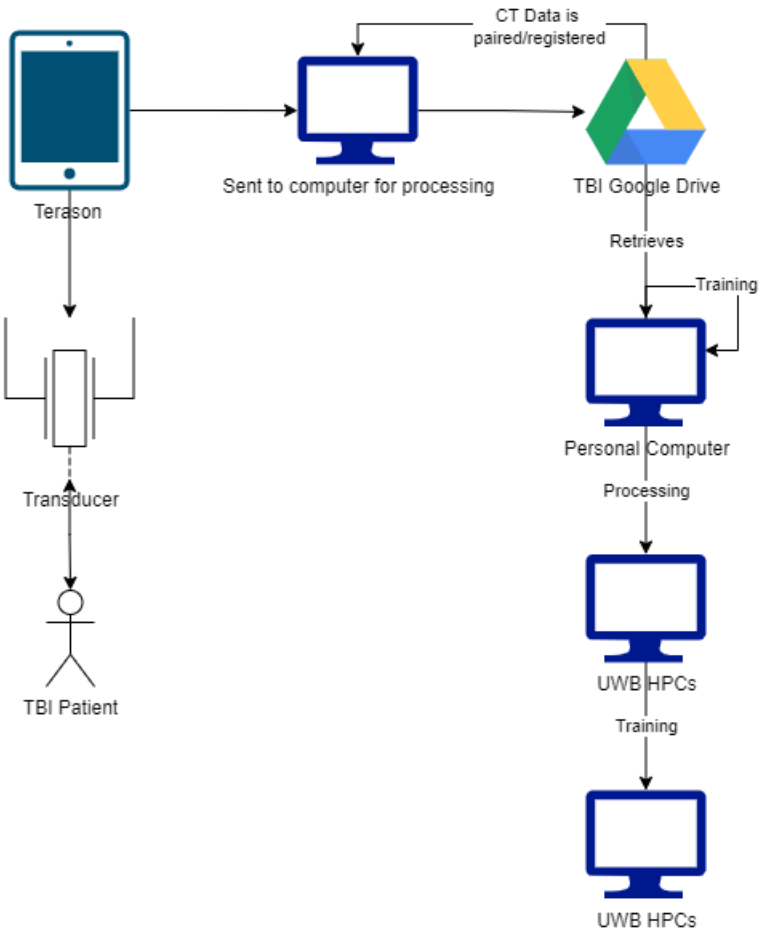

Figure 3.3 Ultrasound Hardware workflow: similar to the CT diagram above, focus of this work starts after retrieval of data from the TBI Google Drive.

3.1.2 *Data Collection and Processing System*

The data collection associated with the ultrasound device had MATLAB code running locally that allowed for streaming. Once the data was streamed, it was later processed and refined to show more relevant details for TBI. The data collection process can be seen in Figure 3.4 as provided by John Kucewicz in [9]. Figure 3.5 has been created as part of this research to provide a larger view of how the data is collected and on what timeline, as well as to give a better understanding of this process in the context of this project. While the wide scope data collection diagram is straightforward there are some components worth mentioning:

- Blood Contained in Scans: Is a Boolean statement where a detected bleed will continue to be a candidate for TBI data collection while false indicates they are no longer meeting the necessary requirements.
- Treatment (Surgery): Occurs and is meant to treat the patient an ensure they survive.
- Is Patient Awake: Determines if the patient is awake and if not, apply an unknown delay until they are awake.
- Is Patient Consenting to Data Collection: If the patient consents, they continue into ultrasound data collection, if not they are removed from the study.
- Ultrasound Data Collection: The patient, after surgery and an unknown delay in time, has their ultrasound images taken and added and moved on to the next step.
- Data Processing: Applies a black box approach that indicates the data processing diagram.

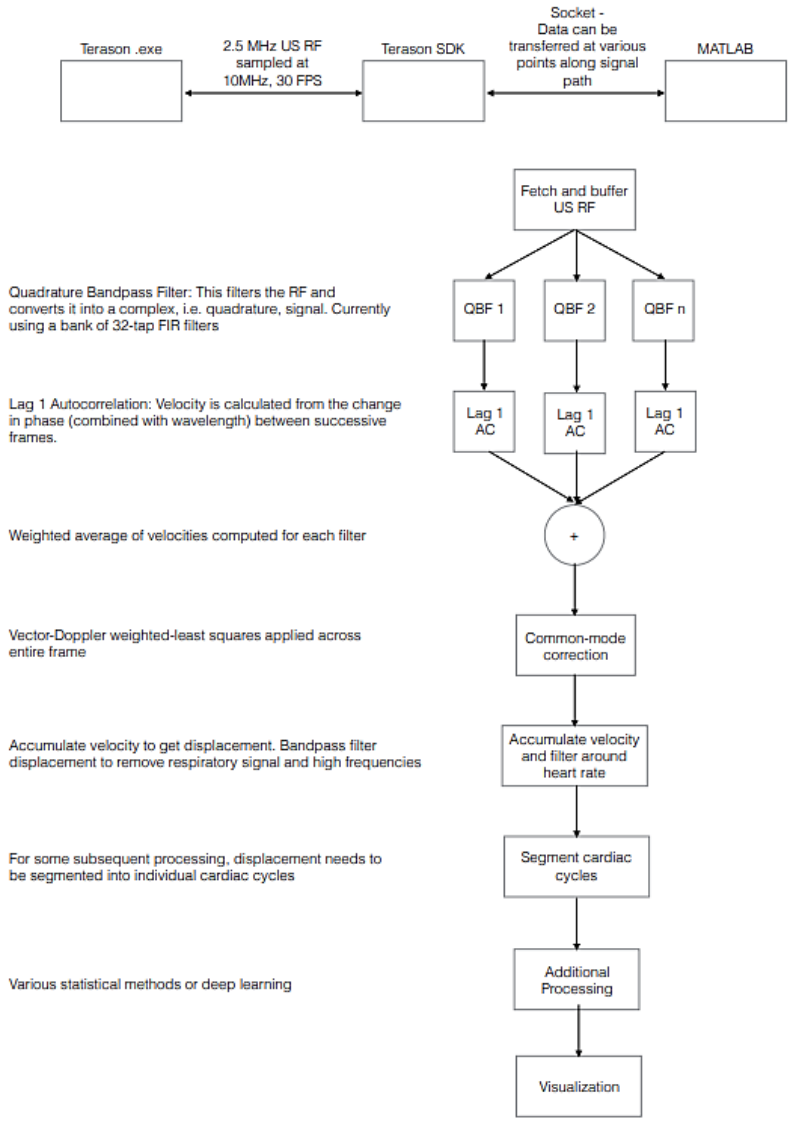

Figure 3.4 Data Processing Architecture as explained by John Kucewicz and contained in the TBI team's document repository [9], [10].

Figure 3.5 Patient Data Collection Flow.

3.1.3 *Machine Learning and Data Processing System*

The data that was collected by the team was in a state that did not easily interact with any programming languages or tools outside of MATLAB. It was important that the data was reprocessed to train and build the machine learning models. Python was utilized as the primary language throughout this process due to its presence in the machine learning community. Many

ML frameworks and associated tools have been developed for python, which provides the ability for support and ease of use. Figure 3.6 depicts how the data is processed and fed into the machine learning algorithms. Initially, the data starts in the google drive account, but is read depending on which detection method is desired. The Synthetic data generation module contains 2 distinct types of generation, both bMode and bModeBlood. The other detection methods all focus on skull, ventricle, or blood detection. Data processing or DP depicts a high-level view of what happens behind the scenes when processing the MAT files. After the DP step, models are optimized, and data is stored in relevant locations. Once optimizing training is started and data collection will occur. The models all follow a similar approach and flow but uses data that is needed for the specific model.

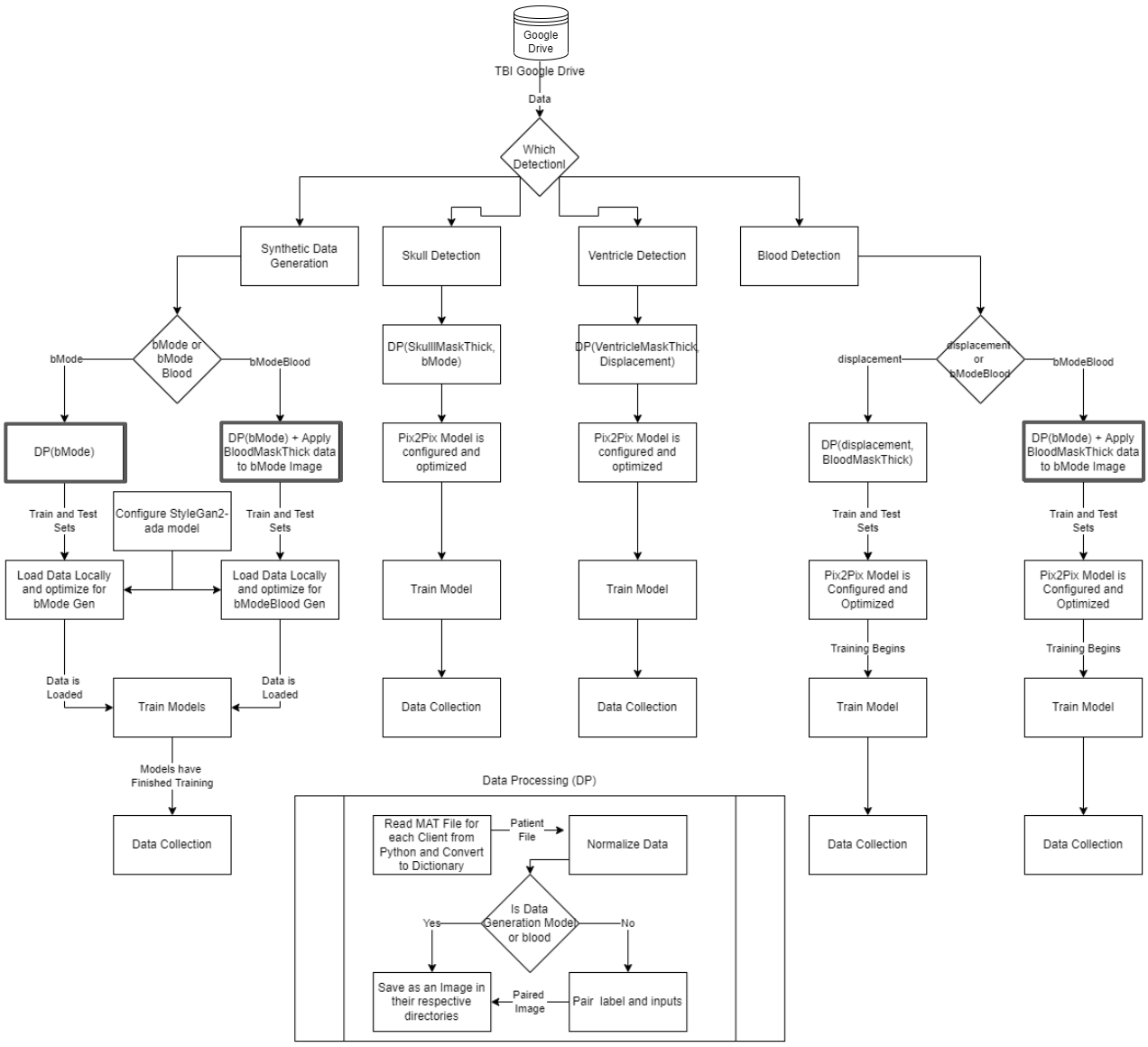

Figure 3.6 Machine Learning and Data System Diagram: this diagram encompasses the data scope of this thesis research. Sub-image titled Data Processing contains detail of what this process entails at the blocks marked with darker outlines.

3.1.4 *System Architecture as a Whole*

The machine learning project requires significant attention when processing data, pairing data, building models, configuring models, and collecting metrics. Figure 3.7 depicts how all the components work together, while the following list indicates the steps done:

1. Patient intake
2. CT Scan

3. Treatment
4. Conscious permission to collect Ultrasound Data
5. Ultrasound Collection
6. John Kucewicz Ultrasound data signal processing
7. Read MATLAB files
8. Normalize datatypes that are used for detection models
    a. Pair data with their proper counterparts
    b. Split data into Train and Test
9. Train Models
10. Collect Data using Test Set

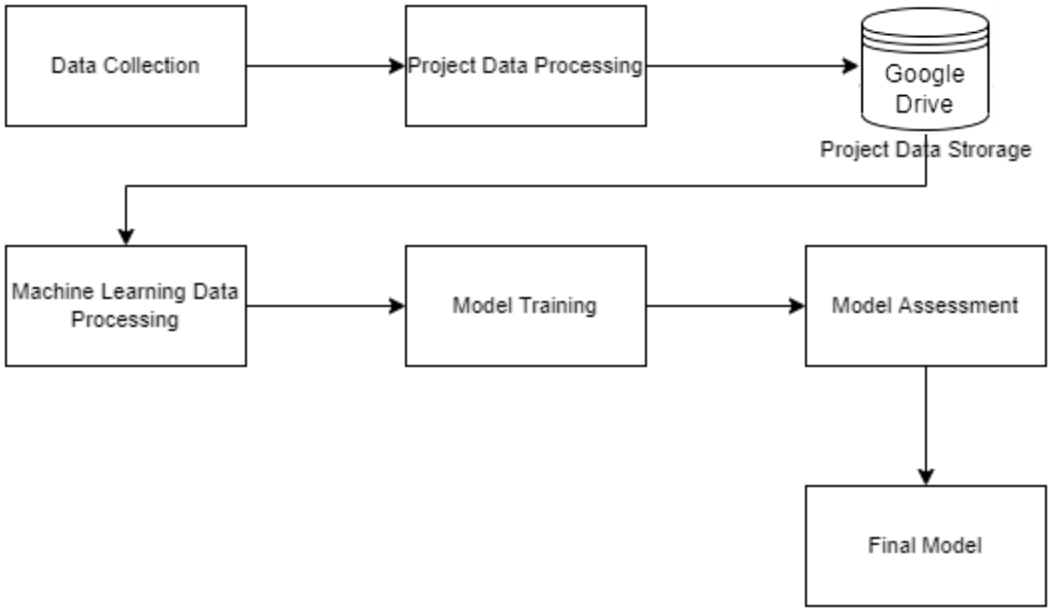

Figure 3.7 System Architecture for this research: summarizing relevant work and process flow between modules.

3.2 Data Processing

Each detection model required different approaches to extract information that could be used to help the model better identify its goal. The necessary futures and datatypes for the project were:

- bMode

- Displacement
- Skull Mask
- Ventricle Mask
- Blood Mask

bMode and Displacement were both utilized throughout each detection algorithm, but each algorithm had its own unique target data that was necessary to find what it was detecting. Table 3.1 displays whether the data was used in the project and gives a detailed analysis as to why it was or was not.

Table 3.1: TBI Data Usage

Data Type	**Used in Project**	**Analysis**
bMode	Yes	bMode was the most promising datatype when going into the project. It provided a fundamental image inside of the human head. While the data collection may be inconsistent, it captured detailed results of the human head. The brightness images showed visible skull, ventricles, and some potential blood while being examined by the teams' medical researchers.
Displacement	Yes	The displacement data was found to be a likely candidate that contains both bleeding and ventricles. While the initial focus was on bMode, displacement was also used train blood and ventricle detection models.
SkullMaskThick	Yes	SkullMaskThick was imperative when detecting skull. This CT derived datatype provided a detailed look into the patient's head. Showing clear indicators of where their skull was in the image. The mapped data with the patient allowed for semi accurate side by side comparison of where the skull is and how it should map from bMode to skullMask.
VentricleMaskThick	Yes	The ventricle data provided a basis to build ventricle detection from. The mask shows indicates where the ventricles are in the patient's head, and thus could be extrapolated to the ultrasound collected data.

BloodMaskThick	Yes	The bloodMask provided the teams earliest sample on each patient, giving the most important piece of data. The blood indicated in the bloodMask is the most realistic representation of what a natural bleed looks like when taken out of a head trauma causing experience. The pixels in this CT derived data clearly showed where each patient was bleeding, which indicated it would give us the highest chance of training an accurate model.
All CT Data	No	The full CT data was not used in, rather just the mask derivatives.
NormMask	No	A normalized CT image.
BrainMask	No	Contains only the brain, removing all other information from the CT image.

While each data point needed its own specific processing approach, there is one shared step that each needed. When processing images, it is imperative that they are normalized. Image normalization is the process of changing a range of pixel values or intensities to a specific band. Once processing has occurred to convert the datatypes into a RGB readable type, the pixels were than normalized. Without normalization, some images would appear to be blank or featureless, which would negatively affect the performance of the machine learning models. Figure 3.8 depicts what an image looked like prior to normalization, while Figure 3.9 shows how it appears post normalization.

Figure 3.8: Blank Appearing Data Due to Not Normalizing: the image is literally blank, grey rectangle has just been added here for readability purposes.

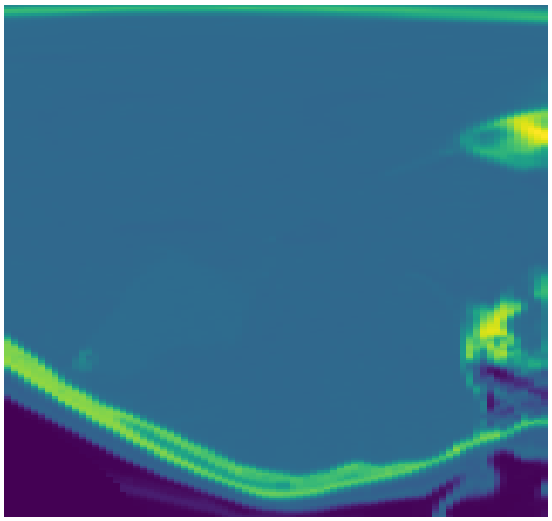

Figure 3.9: Normalized CT Data

3.2.1 *bMode and Displacement Processing*

The original format that the data was processed in was a MATLAB 3-dimensional object where the first 2 dimensions were image specific, and the 3rd dimension indicated the image number. To process the data for the machine learning model training data and input data the following steps were taken:

1. Load the matrix into a SciPy dictionary[20]
2. Create a bMode/displacement NumPy array utilizing a MATLAB converted dictionary[21]
3. Use interlinear interpolation to resize the image to 256 pixels by 256 pixels
4. Calculate the mean for the stack of bMode/displacement images to form a single datapoint

$$\frac{1}{N}\sum_{i=1}^{N} X_i$$

5. Normalize the pixel values to be in range of 0 to 255 and in the BGR color channels
6. Stack image with target data

The resulting data for bMode appears in Figure 3.10 and displacement in Figure 3.11.

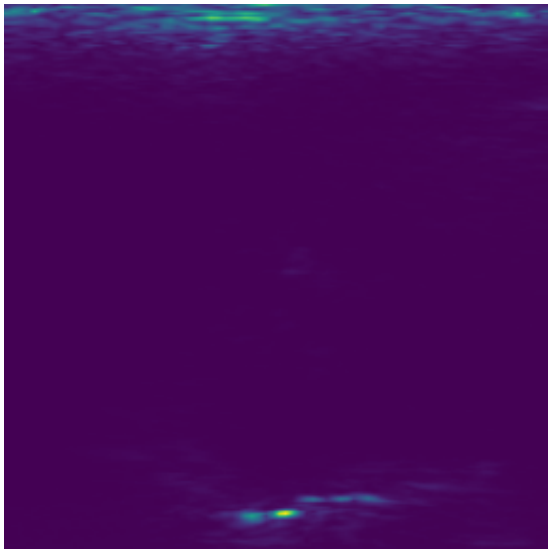

Figure 3.10: Demonstrates a post processing image of bMode, where color indicates an object inside the head

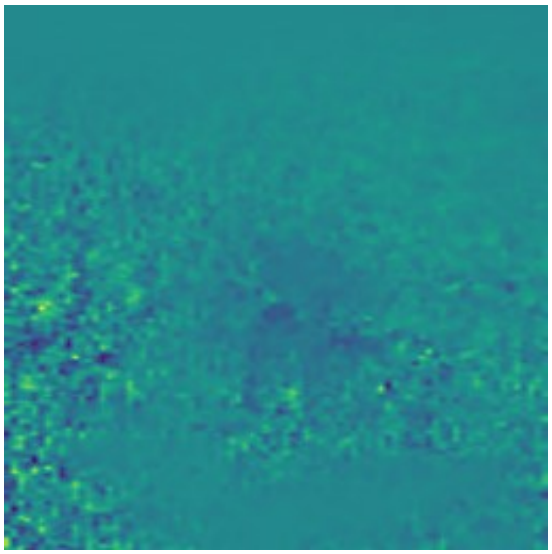

Figure 3.11: Demonstrates a post processing image of displacement

3.2.2 *Skull and Ventricle Processing*

The original MATLAB data was converted for skullMaskThick and ventricles. SkullMask thick was favored over skullMask because it was averaged over multiple slices that would better represent the average thick bMode data. Each datatype was converted into a python dictionary where the pixel values were normalized between 0 and 255. Figure 3.12 and Figure 3.13 show both what the skull and ventricle data appeared as following these processing steps. The following

steps were done for the training patients and testing patients in separate runs and stored in separate locations.

1. Load MATLAB data as a python dictionary
2. Convert the skullMaskThick or ventMask data into a NumPy array
3. Normalize each index between 0 and 255
4. Stack skullMask with cleaned bMode and stack ventMask with displacement data

The result of data processing were these four folders:

- Training bMode and SkullMask
- Testing bMode and SkullMask
- Training displacement and Ventricles
- Testing displacement and Ventricles

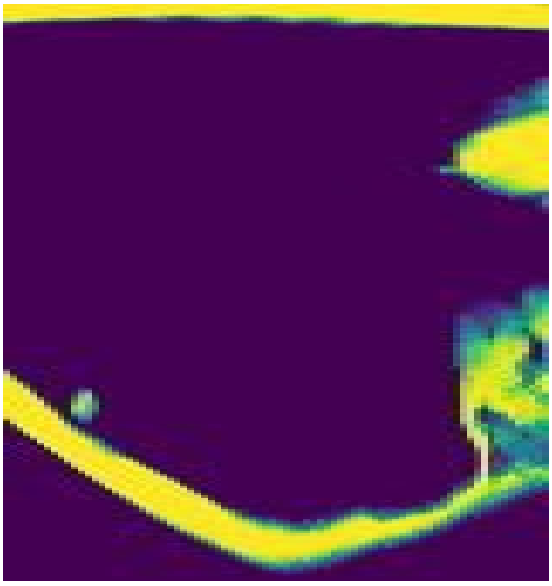

Figure 3.12: skullMaskThick data following skull data processing

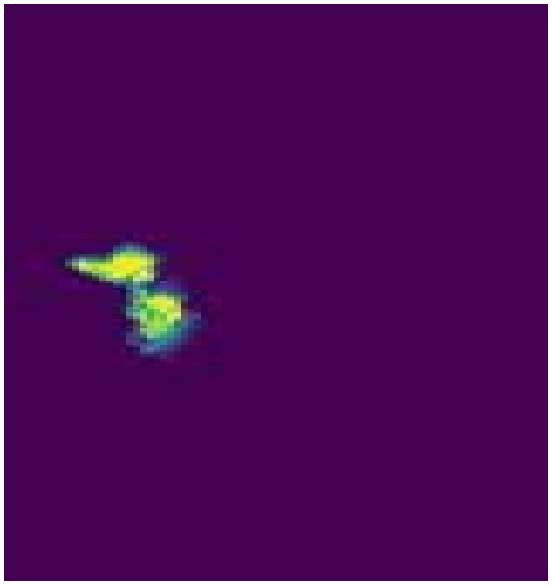

Figure 3.13: ventricleMaskThick data following data processing

3.2.3 *Blood Processing*

The bloodMask data that contains IA and EA bleeds was processed using similar steps as Skull and Ventricle processing with a few exceptions:

1. Load Matlab data for the patient as a python dictionary
2. Convert the bloodMaskThick data into a numpy array
3. Normalize each index between 0 and 255
4. OpenCV addweighted where bMode is the predominant image and the bloodMask is being layered over the top. The equation and variable definitions can be seen as $g(x)=(1-\alpha)f_0(x)+\alpha f_1(x)$ [22]
5. Pair both samples together to create our training data

The second blood detection model utilized displacement and bloodMask data which utilized the same approach as ventricle and displacement processing.

3.2.4 *Preparing Data for Training*

Once the data is extracted from the MATLAB files, normalized, and resized, it is necessary to build datasets that will be used for various machine learning models. Each type of feature that is being recognized needed its own model, and therefore its own unique dataset. The following

data types were paired for each specific feature that was being detected which can be seen in Table 3.2.

Table 3.2: Model and Datatypes Used

Model	Data Types	Reasoning
Skull Detection	bMode thickSkullMask	• bMode is one of two primary datatypes in the ultrasound dataset that reflects the contents of what is being analyzed. After some processing, bMode appeared to show a higher quality data sample for the task at hand (ie general shapes were outlined better than displacement). • thickSkullMask was used as our target data. This information as captured from a CT machine and depicted the patient's skull in a clear manner. Each cross section of skullMaskThick has been mapped to the corresponding bMode image. • Following the preprocessing steps for both bMode and thick skullMask, both data types were renormalized so that one images features were not greater than the other and paired together. For each patient there are around 100 total bMode and thickSkullMask pairs. Figure 3.14 demonstrates what a pair looks like. The right half of the image contains the input data which in this case is bMode. The left half contains the data we are trying to find in the bMode image, or in other words it is the target data.
Ventricle Detection	bMode thickVentricleMask	• bMode was the recommended and preferred data type when determining where the ventricles are in the patient. Shape regression has been used in the past to determine where ventricles are in echocardiograms with marginal success[23]. Upon further processing, it was determined that they were difficult to see, so a second test was created with displacement and thickVentricleMasks. • thickVentricleMask is derived from the CT data and was used over a ventricleMask because it showed greater detail and was more likely to be correctly approximated in the

Model	Data Types	Reasoning
		same region of the brain. The thickVentricleMask provides a detailed view of and where the ventricles are in each patient. • Both data types were re-normalized and paired together. Each patient had enough data for approximately 100 thickVentricleMask and bMode pairs. Figure 3.15 shows the paired data where the left image is a ventricle mask, and the right side is a bMode image.
	Displacement thickVentricleMask	• Displacement images showed promise indicating where ventricles may be. Whereas the displacement had noticeable pixelization in some instances in the region where a ventricle may lie. • The data was paired together where the ventricles are on the left side of the image and the displacement on the right side. Both normalized as to not unbalance the image in one way or another. Figure 3.16 contains 2 examples that depict how the training pairs appeared when going into the model.
Bleed Detection	bMode displacement bloodMaskThick	• The bMode data did not contain or contained very little bleed information. Therefore, to simulate the detection of bleeding, the CT data bloodMaskThick is layered over the bMode data. Creating a synthetic data sample of what bleeding may look like. After layering the data as seen in Figure 3.17, the final paired data was created using bModeBleed as the target and bMode as the input which can be seen in Figure 3.18. • The second blood detection model would use displacement and the bloodMaskThick data. Referring to the ventricle detection setup, the displacement data is read from a matlab file and converted into a numpy array and normalized. The bloodMaskThick data was read from the matlab file and converted into a numpy array where it was than paired with displacement data. Figure 3.19 depicts how this data was

Model	Data Types	Reasoning
		paired together, with the blood mask on the left, and the displacement data on the right.
Synthetic Data Generation	bloodMaskThick bMode	• The data being used for synthetic generation is focused around bMode and bModeBlood as seen in Figure 3.18. The same data was used, except for pairing the data. Each data type was kept in the same indices for the StyleGan2-ada model.

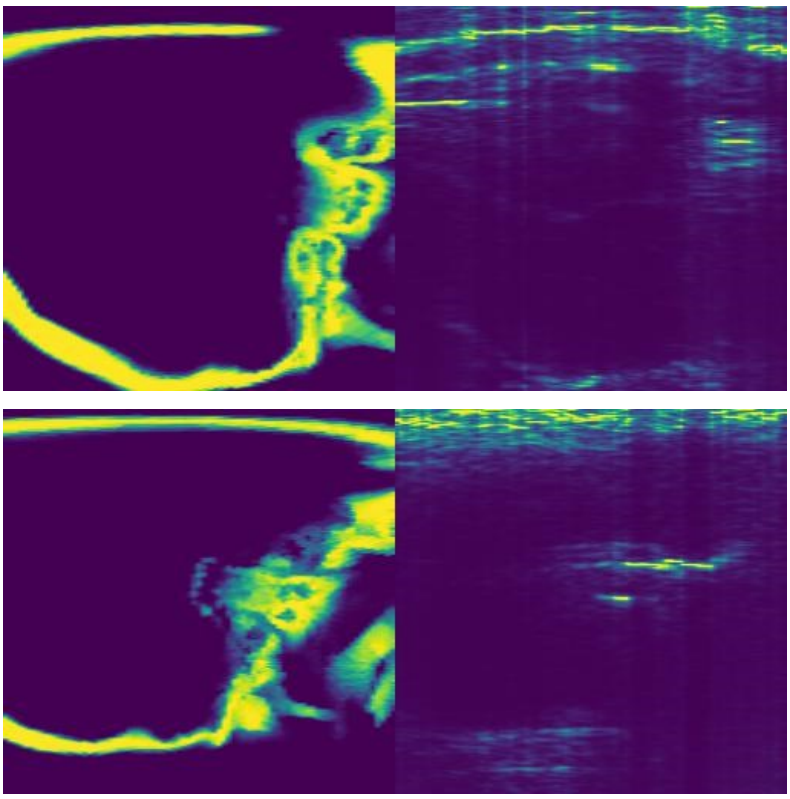

Figure 3.14: Contains 2 examples of skullMask and bMode training pairs

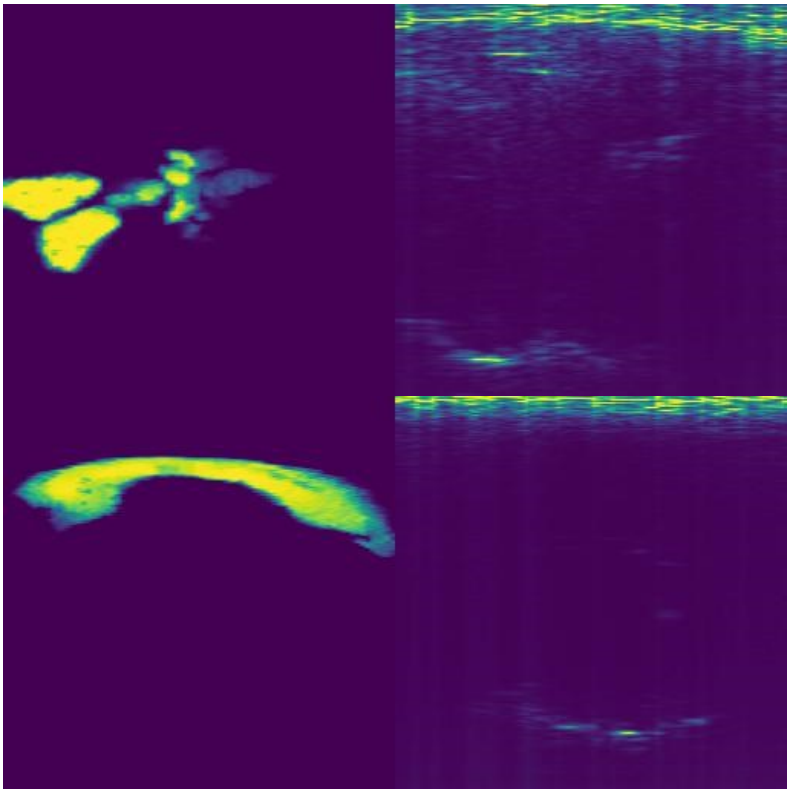

Figure 3.15: Contains 2 examples of ventricle bMode training pairs

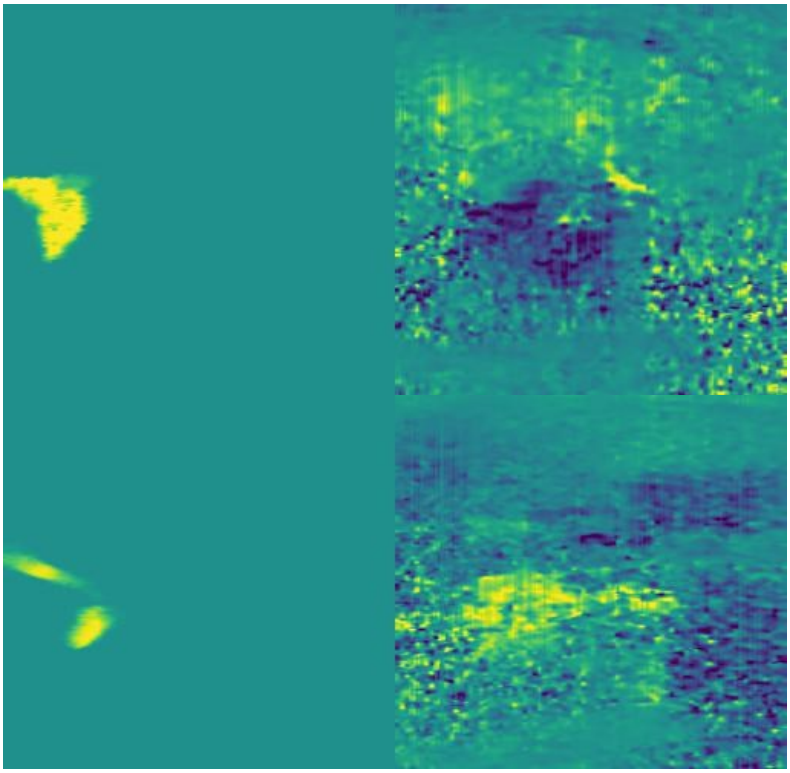

Figure 3.16: Contains 2 example of ventricles and displacement training pairs

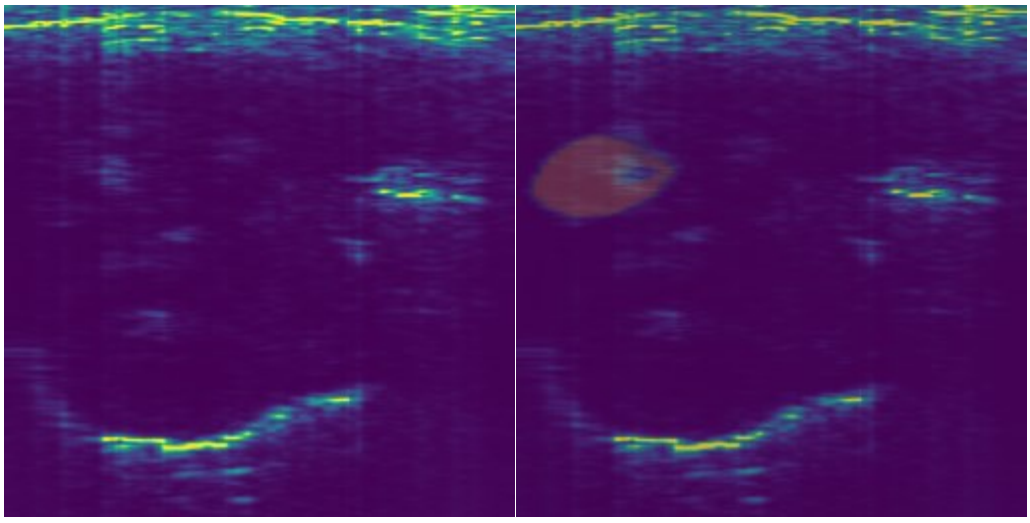

Figure 3.17: Before and After BloodMaskThick was layered on bMode

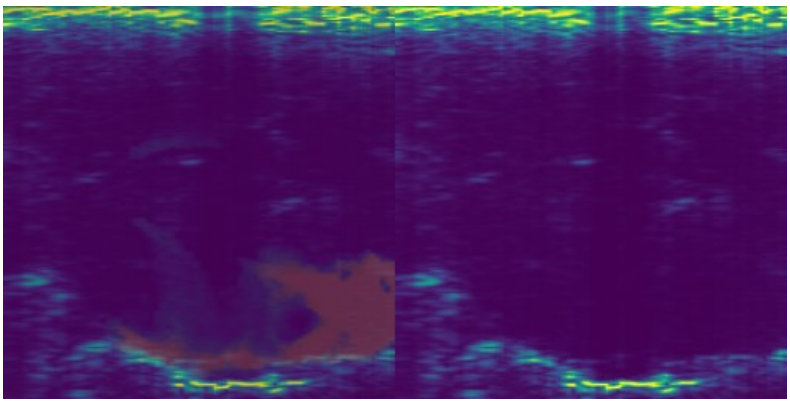

Figure 3.18: Paired bModeBlood and bMode Image

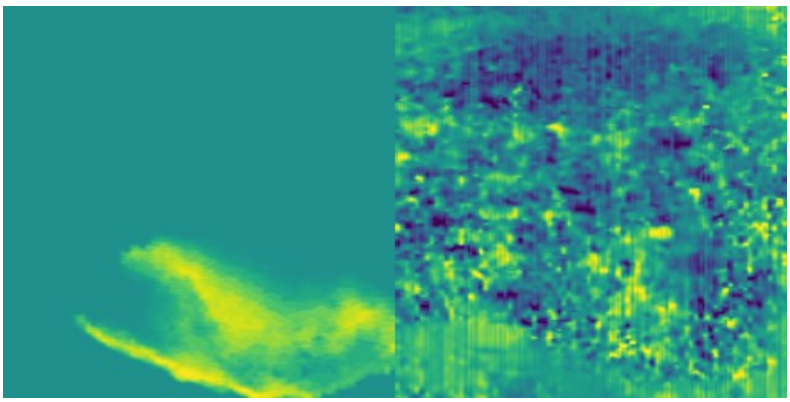

Figure 3.19: Paired bleed and displacement data

3.3 Synthetic Data Generation

Current projections have the world on track to have generated 463 exabytes of data by the year 2025, but just because there will be more data, doesn't mean that we will have the resources to train new models[24]. To elaborate, detecting bleeding in the human head using bMode is very niche which results in a deficiency of data. Eventually, this data may be collected, but that is on an unknown timeline, and therefore requires synthetic data generation. Creating data is commonplace amongst all machine learning and data driven projects. Synthetic data generation, if done correctly, can contribute to simulating a much larger dataset and thus helping train models to converge on a clearer result. While detecting liver lesions, a research team collected 182 CT scans and synthetically generated more data which resulted in significantly better results when detecting cysts, metastases, and hemangiomas [25].

Data augmentation and synthetic generation play a significant role across many statistical and machine learning fields today. Generative adversarial networks have played a key role in producing a new way to generate synthetic data. Until recently, many machine learning models would focus on augmenting the data (flipping, random noise, rotation, etc)[26]. As computational resources have increased, so to have the capabilities to generate synthetic data. In projects where data is scarce, or there is insufficient information on what you are classifying, data augmentation and generation can be used to help increase the size of your dataset. While typically, synthetically generated data may play an important role in acting as a validation set, in this project, the synthetic data will be mixed to help train models and be used in testing[26].

There are many ways to create synthetic data that are commonly used by industry and researchers today, but the primary generation technique used in this project will focus on using GANs and minimal data augmentation. Typically, data generation can be used to speed up object recognition and other classification problems, in this case, data generation will be used to help diversify and increase the size of the datasets[27]. Training generative adversarial networks with a small dataset can lead to the discriminator overfitting, which in turn causes overfitting. Utilizing a method as seen in Pix2Pix and Stylegan2-ada augments the data prior to training[18], [28]. Augmenting the discriminator increases the stability of smaller (less data) networks. The model being used is titled Stylegan2-ada, which was developed by NVIDIA to generate fake faces[28]. Stylegan2-ada is very similar to Stylegan2 but was able to converge on similar results while using orders of magnitude

less data as a result of augmenting the discriminators input[28]. Figure 3.20 depicts the networks high level architecture and how augmentation can be viewed. In the following Figure 3.21 and Figure 3.22 there is a depiction of the Generator and Discriminator. The generator scales the output back up to a 256 by 256 image where it is then passed to the Discriminator for testing.

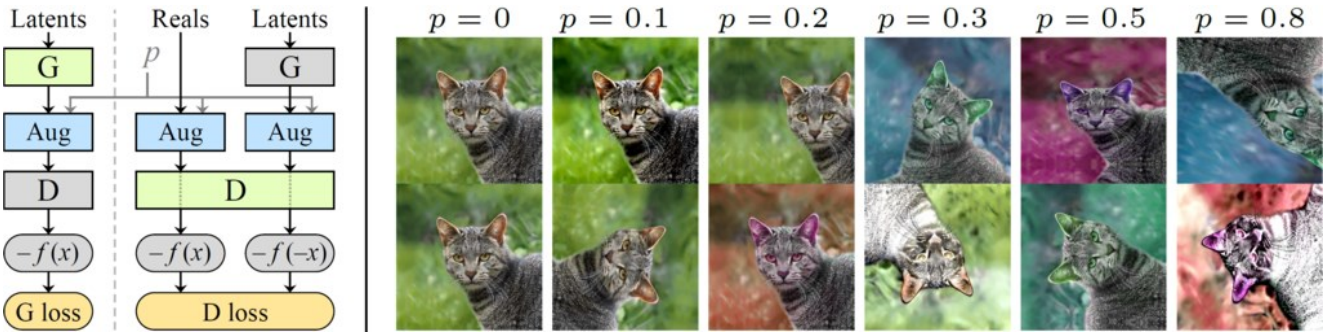

Figure 3.20: Nvidia Stylegan2-ada architecture and Randomness Scale (p)[19]

Generator	Parameters	Buffers	Output shape	Datatype
---	---	---	---	---
mapping.fc0	262656	-	[16, 512]	float32
mapping.fc1	262656	-	[16, 512]	float32
mapping	-	512	[16, 14, 512]	float32
synthesis.b4.conv1	2622465	32	[16, 512, 4, 4]	float32
synthesis.b4.torgb	264195	-	[16, 3, 4, 4]	float32
synthesis.b4:0	8192	16	[16, 512, 4, 4]	float32
synthesis.b4:1	-	-	[16, 512, 4, 4]	float32
synthesis.b8.conv0	2622465	80	[16, 512, 8, 8]	float32
synthesis.b8.conv1	2622465	80	[16, 512, 8, 8]	float32
synthesis.b8.torgb	264195	-	[16, 3, 8, 8]	float32
synthesis.b8:0	-	16	[16, 512, 8, 8]	float32
synthesis.b8:1	-	-	[16, 512, 8, 8]	float32
synthesis.b16.conv0	2622465	272	[16, 512, 16, 16]	float32
synthesis.b16.conv1	2622465	272	[16, 512, 16, 16]	float32
synthesis.b16.torgb	264195	-	[16, 3, 16, 16]	float32
synthesis.b16:0	-	16	[16, 512, 16, 16]	float32
synthesis.b16:1	-	-	[16, 512, 16, 16]	float32
synthesis.b32.conv0	2622465	1040	[16, 512, 32, 32]	float16
synthesis.b32.conv1	2622465	1040	[16, 512, 32, 32]	float16
synthesis.b32.torgb	264195	-	[16, 3, 32, 32]	float16
synthesis.b32:0	-	16	[16, 512, 32, 32]	float16
synthesis.b32:1	-	-	[16, 512, 32, 32]	float32
synthesis.b64.conv0	1442561	4112	[16, 256, 64, 64]	float16
synthesis.b64.conv1	721409	4112	[16, 256, 64, 64]	float16
synthesis.b64.torgb	132099	-	[16, 3, 64, 64]	float16
synthesis.b64:0	-	16	[16, 256, 64, 64]	float16
synthesis.b64:1	-	-	[16, 256, 64, 64]	float32
synthesis.b128.conv0	426369	16400	[16, 128, 128, 128]	float16
synthesis.b128.conv1	213249	16400	[16, 128, 128, 128]	float16
synthesis.b128.torgb	66051	-	[16, 3, 128, 128]	float16
synthesis.b128:0	-	16	[16, 128, 128, 128]	float16
synthesis.b128:1	-	-	[16, 128, 128, 128]	float32
synthesis.b256.conv0	139457	65552	[16, 64, 256, 256]	float16
synthesis.b256.conv1	69761	65552	[16, 64, 256, 256]	float16
synthesis.b256.torgb	33027	-	[16, 3, 256, 256]	float16
synthesis.b256:0	-	16	[16, 64, 256, 256]	float16
synthesis.b256:1	-	-	[16, 64, 256, 256]	float32
---	---	---	---	---
Total	23191522	175568	-	-

Figure 3.21: Stylegan2-ada Generator architecture for 256x256 images [19]

Discriminator	Parameters	Buffers	Output shape	Datatype
b256.fromrgb	256	16	[16, 64, 256, 256]	float16
b256.skip	8192	16	[16, 128, 128, 128]	float16
b256.conv0	36928	16	[16, 64, 256, 256]	float16
b256.conv1	73856	16	[16, 128, 128, 128]	float16
b256	-	16	[16, 128, 128, 128]	float16
b128.skip	32768	16	[16, 256, 64, 64]	float16
b128.conv0	147584	16	[16, 128, 128, 128]	float16
b128.conv1	295168	16	[16, 256, 64, 64]	float16
b128	-	16	[16, 256, 64, 64]	float16
b64.skip	131072	16	[16, 512, 32, 32]	float16
b64.conv0	590080	16	[16, 256, 64, 64]	float16
b64.conv1	1180160	16	[16, 512, 32, 32]	float16
b64	-	16	[16, 512, 32, 32]	float16
b32.skip	262144	16	[16, 512, 16, 16]	float16
b32.conv0	2359808	16	[16, 512, 32, 32]	float16
b32.conv1	2359808	16	[16, 512, 16, 16]	float16
b32	-	16	[16, 512, 16, 16]	float16
b16.skip	262144	16	[16, 512, 8, 8]	float32
b16.conv0	2359808	16	[16, 512, 16, 16]	float32
b16.conv1	2359808	16	[16, 512, 8, 8]	float32
b16	-	16	[16, 512, 8, 8]	float32
b8.skip	262144	16	[16, 512, 4, 4]	float32
b8.conv0	2359808	16	[16, 512, 8, 8]	float32
b8.conv1	2359808	16	[16, 512, 4, 4]	float32
b8	-	16	[16, 512, 4, 4]	float32
b4.mbstd	-	-	[16, 513, 4, 4]	float32
b4.conv	2364416	16	[16, 512, 4, 4]	float32
b4.fc	4194816	-	[16, 512]	float32
b4.out	513	-	[16, 1]	float32
---	---	---	---	---
Total	24001089	416	-	-

Figure 3.22: Stylegan2-ada Discriminator architecture for 256x256 images

3.4 DETECTION

Each detection problem utilized the same model, but with new data and techniques. The desired outcome for skull detection was an image that contained where pieces of the human skull where contained. Ventricle detection also followed a similar pattern, wanting an output that contains a realistic map of where ventricles are in ultrasound collected data. Detecting bleeding was slightly different in that there were two models were created, the first utilized bMode with overlayed bleed,

and the second uses displacement data. Each model utilized the recommended parameters to start but configured as necessary to help produce their desired outcomes.

3.4.1 *Skull Detection*

When examining data, it was clear that human skull is visible when using the ultrasound device, however it frequently missed gaps and other key features that were shown in the CT data. Cross examining the ultrasound collected data with the CT skull mask, it was clear that skull does not show as much promise in other pieces of information. However, it is worth noting that skull was visible in the displacement data, but research had already been done to investigate this. The bMode data clearly showed semblance and shadows of where skull lies in the human head. However, it lacked clarity and certain thinner pieces of bone. Figure 3.23 and Figure 3.24 respectively depict good and bad examples of skull being visible in the data. In the good example, skull can be seen along the top and bottom portions of the brightness image. While the image may not show a full picture of skull, it is certainly enough for a medical professional to infer on. In the poor example, the brightness image shows some of the top and bottom of the skull mask but misses almost all the midsection of the image. The bMode would not be enough for a medical professional to infer where the skull is for the patient.

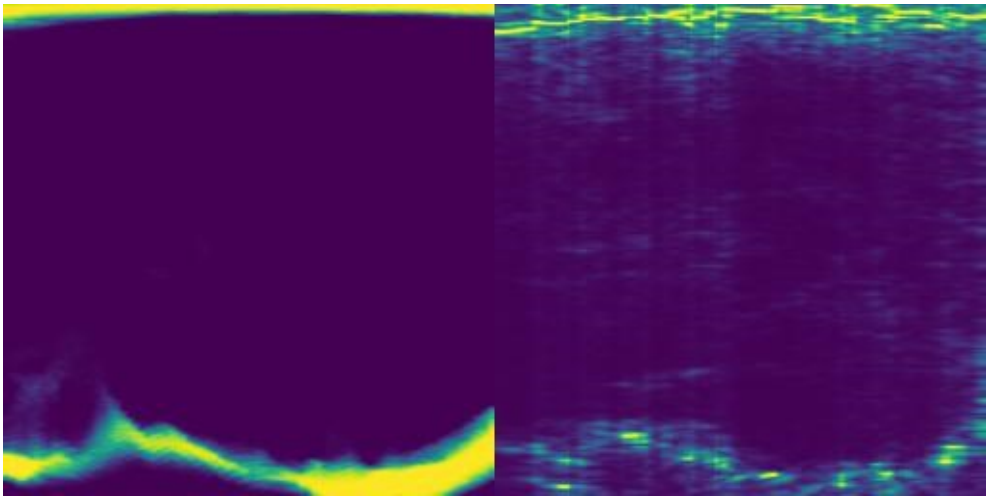

Figure 3.23: An example of a patient's skull as shown from CT to bMode. In this example, the patient skull is clearly visible in the bMode data.

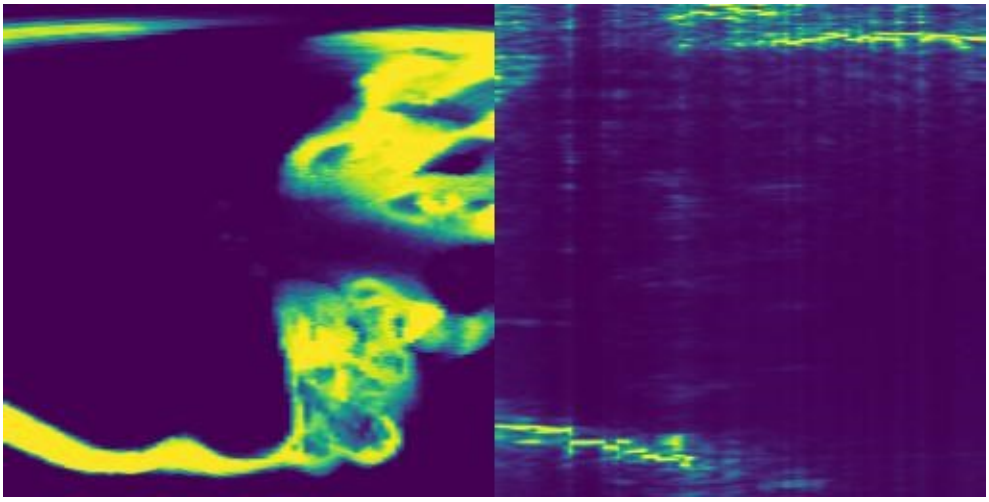

Figure 3.24: An example of a patient's skull as shown from CT to bMode. In this example, the patient skull is no visible in the bMode data.

For skull detection, a designated skull mask output was desired, which meant that some form of image-to-image translation model would be necessary for detecting the skull. To get more realistic results, a generative adversarial network would be optimal due to the way it functions. Using a discriminator and generator to train 2 networks. The discriminator would determine if the generators output was a real image or if it was fake. Depending on that information, the generators weights would be modified to better tune its accuracy. Skull detection utilized a Pix2Pix style network which was created by AI researchers at UC Berkley[18]. Pix2Pix was created to map an input image to an expected output image, and when necessary, infer this output. While the current dataset contains bModes that have readily available and mapped skull mask images, this will not be the case with its intended application. Therefore, Pix2Pix can essentially create a mapping between bMode and skull mask to generate a synthetic skull mask.

Training the Pix2Pix network required the skull mask and bMode data to be paired as previously discussed but leaving out a patient's data to test with. While, commonly, more data would be saved to test against, the amount of data provided in this project was not sufficient. The model performed considerably better when using all but one patient file. However, given more data, having a larger testing set would provide more realistic and better assumptions as to how the model performs. After pairing the data, the Pix2Pix network would be fine-tuned to run on a specific GPU, i.e., more VRAM meant larger models could be trained. Additionally, a new function was added where the Fréchet Inception Distance (FID) could be calculated on an epoch-

by-epoch basis[29]. Which allowed the trainer to determine how well the model was functioning or if it needed to be stopped and adjusted.

Assessing the Pix2Pix model was relatively difficult when you are assessing how an image looks based on another model's prediction for that image. The discriminator is not a good metric as to how the generator is performing because the discriminator can also be wrong. Therefore, additional metrics would be added to assess the quality of the GAN, which in this case is an FID score. An FID score is the measurement of the difference between 2 distributions of feature vectors[29]. For example, how well does the output of the generator match the target output. The FID would be calculated which would give an accurate and measurable representation of how the generator and discriminator are functioning. The FID was calculated by using the test patient's original skull mask and the Pix2Pix generated skull mask.

3.4.2 *Ventricle Detection*

Upon initial inspection of the bMode data, it was clear that ventricles were not visible, or at least not to the untrained eye. After some discussion surrounding what data could include ventricles and what data may not, the decision to move forward with displacement data was made. Figure 3.25 depicts the initial proposed dataset pairing for detection. Within this image, it is clear the ventricles are not visible. Figure 3.26 contains a ventricle mask that has been paired with a displacement image. Notice that the ventricle is not entirely clear, but the consensus of the team was to focus on displacement data. Much like Skull Detection, a pix2pix model was used to determine where the ventricles resided in the patients. The only difference between the skull detection and ventricle detection lies in the data that is being used. However, it is worth noting that further network tuning is required. Fréchet inception distance is being used to calculate how representative the output of the models is. Similarly, to Skull detection, the test patient was left out of model training and used for FID calculations. These calculations were done by comparing the original patient's ventricle mask to the network produced ventricle mask.

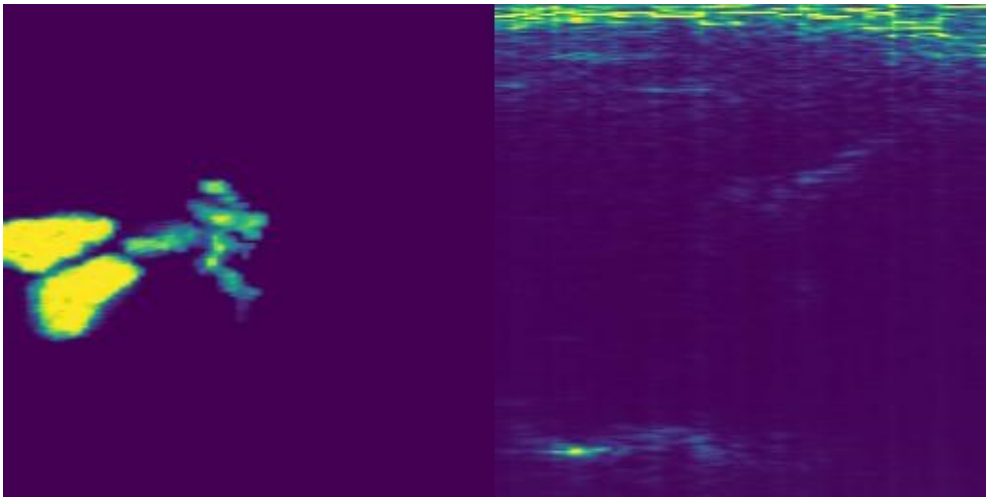

Figure 3.25: Non-visible Ventricles in bMode

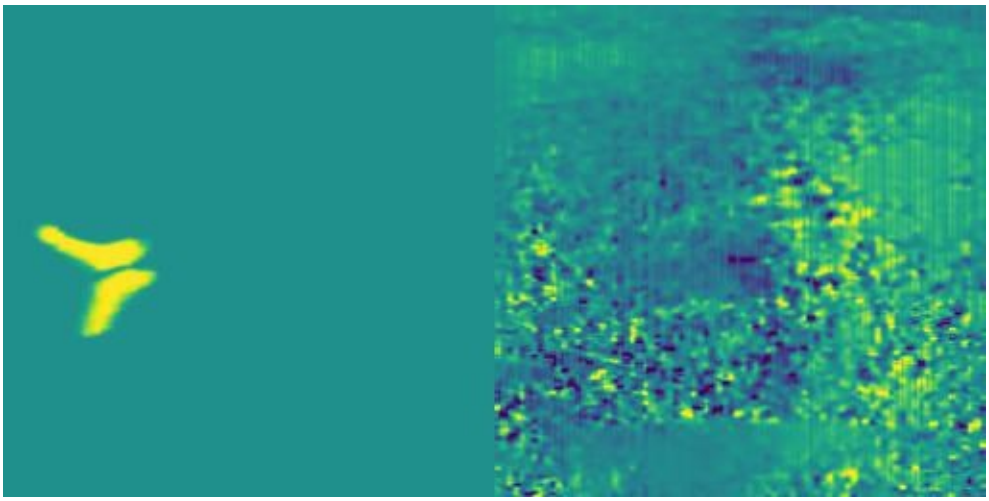

Figure 3.26: Potentially Visible Ventricle in Displacement

3.4.3 *Blood Detection*

After initial inspection of ultrasound data, it was difficult to determine if any of the images contained bleeding. Therefore, I relied on guidance from the research group to evaluate which data would be best used to create a model to find bleeding. After deliberation, displacement and bloodMask appeared to be the most likely candidates to detect bleeding with. The model, which was also debated, was eventually decided upon, we utilized the Pix2Pix network, which would in turn allow the usage of the previously written networks. However, it is worth mention that 2 separate models were created to determine where bleeding is, utilizing two separate datasets:

- Model A: utilized bMode and layered bleed bMode to detect bleeding which can be seen in image Figure 3.27.

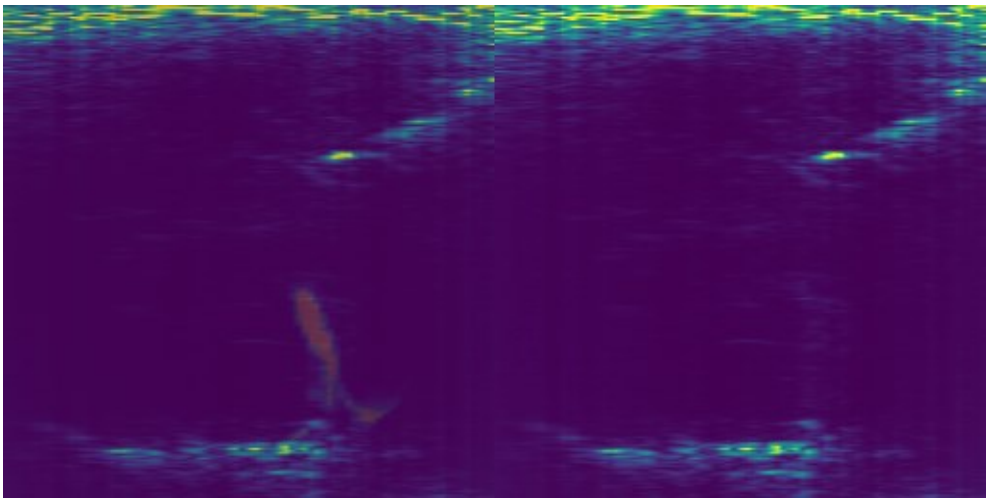

Figure 3.27: bMode Bleed and bMode Pair

- Model B: utilized the recommended displacement and bloodMaskThick data to detect bleeding which can be seen in Figure 3.28.

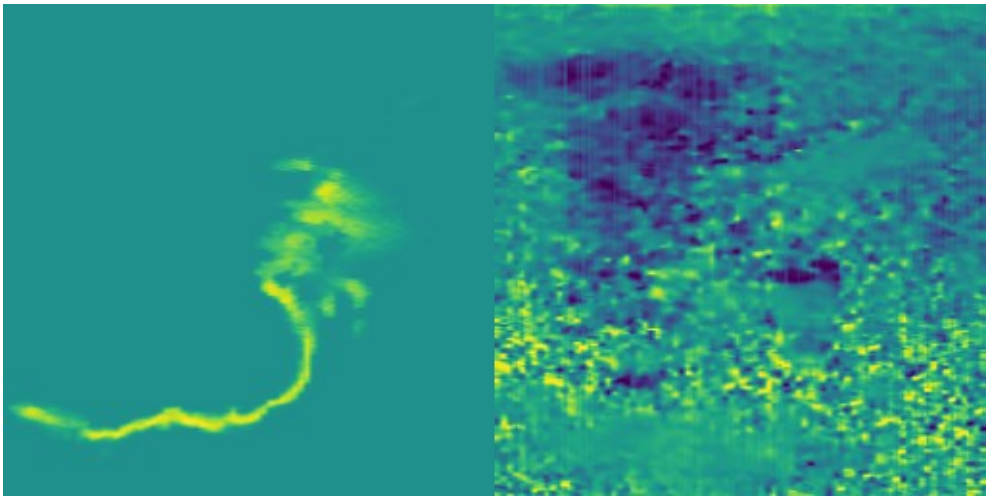

Figure 3.28: bloodMaskThick and Displacement Pairing

The model itself followed the same outline as previously defined in Skull and ventricle detection, with the exception of changing learning rates, batch sizes, and data. Additionally, new metrics would be emitted due to the growth and knowledge gained throughout the project.

3.4.4 *Detection Training*

While each detection model started with similar hyper parameters, to progress further they would need additional modification as training continued. GANs can be quite difficult to train, requiring a significant amount of handholding to make meaningful progress. Often with small datasets, a GAN can enter mode collapse, which at a high-level view means the generator and discriminator are no longer functioning correctly. In this case, the discriminator is classifying everything as been either true or negative. If the discriminator can only make one case, the generator model will start to produce the same image or images repetitively. To avoid mode collapse, a variable learning rate is often used, this ensures that neither model gets stuck in a local minimum, and if it does, it can get out. Using a checkpointing system that saves the model allows for a roll back to a functioning model before mode collapse begins. In doing so, this allows for modification of the learning rate and the restarting of training. The result of which is a model that will continue training and hopefully making better predictions. However, if results do not start improving or mode collapse begins again, the model may have reached its peak performance.

3.5 RESULT ANALYSIS

Image to image translation using conditional adversarial networks is a newer application of machine learning, which leads to fewer ways to analyze your models. Due to the nature of image-to-image translation, it can be difficult to assess whether a GAN is producing an accurate image, therefore, new metrics have been worked on to determine how effective image to image translation models are. There are many research papers that are being written that are meant to help determine how these models do, but for the most part, these methods are not yet something that can be implemented. NVIDA has produced many metrics with their StyleGan3 but are not readily available. While image translation is part of the project, so too is bleed detection. For this project, Table 3.3 depicts the metrics that will be used to evaluate both Pix2Pix and StyleGan2-ada.

Table 3.3: Metrics for Model Analysis

Metric	**Applicable Model**	**Description**
Fréchet Inception Distance (FID)[30]	StyleGan2-ada, Pix2Pix	Calculates the distance between feature vectors in real and generated images. Higher FID scores indicate that the 2 images being compared are less alike. A score of 0.0 would indicate that both images are identical. The lower the FID the more a like the images are[31]. Figure 3.29 depicts how FID represents an image based on how distorted / different it is. The FID is used to determine how well our generation and prediction models perform. Giving insight into either the generated image or predicted images relevance to the original data.
Kernel Inception Distance (KID)[32]	StyleGan2-ada	Measures how dissimilar two probability distribution are. KID compares the skewness, mean, and variance of two images[32]. A number closer to zero signifies that the two images being compared are more like each other. The KID scores provide a means of determining how realistic the StyleGan2-adas models generated images are.
Precision and Recall [33]	StyleGan2-ada	**Precision** is used to evaluate the number of relevant generated images, e.g., how well do they portray a real image and what percentage show that. **Recall** what portion of positives images were identified correctly.
Mean Square Error (MSE)	Pix2Pix	Measures how closely related the target and generated images are.
Root Mean Square Error (RMSE)	Pix2Pix	Measures the distance between how closely related the target and generated data.

Metric	Applicable Model	Description
Multiscale Structural Similarity (MS-SSIM)[34]	Pix2Pix	Used to measure the similarity between 2 images. A value that is closer to 1 indicates a better performing generator. The following equations define various components of SSIM [34] $$l(\mathbf{x},\mathbf{y}) = \frac{2\,\mu_x\,\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1},$$ $$c(\mathbf{x},\mathbf{y}) = \frac{2\,\sigma_x\,\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2},$$ $$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\,\sigma_y + C_3},$$ Multi-scale structural similarity can be defined as [34] $$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = [l_M(\mathbf{x},\mathbf{y})]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(\mathbf{x},\mathbf{y})]^{\beta_j} [s_j(\mathbf{x},\mathbf{y})]^{\gamma_j}$$
Color Histogram	StyleGan2-ada	A color histogram is used to depict the pixel distributions from synthetically generated images and natural images that were collected. The histogram will give a general idea of where and how pixels are distributed.

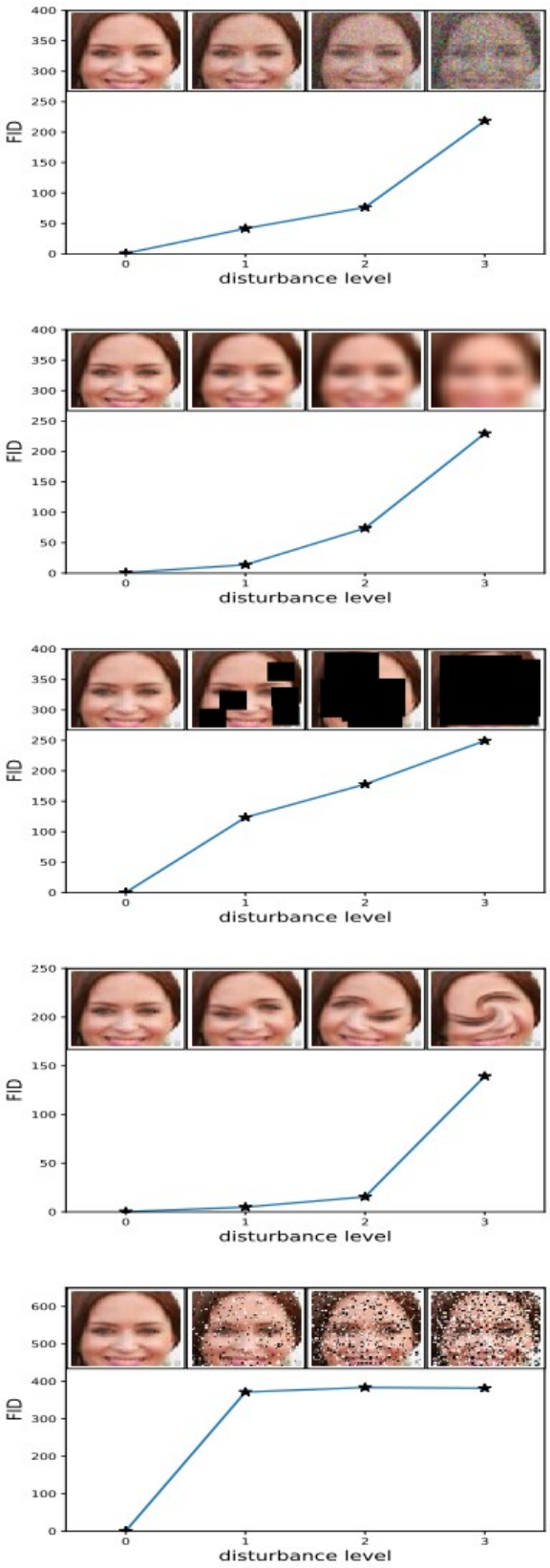

Figure 3.29: FID and Disturbance Level[30]

4 EXPERIMENTS AND RESULTS

4.1 DATA GENERATION

4.1.1 *Experiment Setup*

The synthetic data was generated utilizing the bMode and bMode masked bleed data as training data. The original dataset had approximately 2000 samples in total, which has grown to a substantially larger size since utilizing the Stylegan2-ada network. Both models were built and trained in separate operating system environments requiring several tweaks work at an optimal pace. bMode bleed data was trained on the University of Washingtons Otachi machine, utilizing all its resources, while bMode was trained locally utilizing all the available resources. Once the machines were determined, the appropriate software as defined by Nvidia was installed that would allow these networks to function[19]. Table 4.1 depicts how and where networks were trained and what the associated resources looked like.

Table 4.1: Machines used for training the Stylegan2-ada model

Training Location	**Operating System**	**Resources**	**Model**
University of Washington Otachi	Debian	2 Nvidia V100 12 gb	bModeBleed
Lambda Labs	Debian	8 Nvidia V100 12 gb	bMode but did not finish
Personal Computer	Windows 11	1 Nvidia 3090 24 gb	bMode

Prior to training, the data involved needed to be processed and placed into specific datasets that were compatible with the Stylegan2-ada network. bMode and bloodMaskThick were preprocessed and stored as jpg images. Following the creation of both datasets, bModeBleed was created by layering the bloodMaskThick data over its respective bMode jpg image. The result of these tasks were two datasets, one with bMode only data and one with bModeBleed data. Once these datasets were created, they were processed by the Stylegan2-ada networks dataset tool, which ensures that the images are all of the same size and splits them into subsets for further computation. A dataset

file is created that would allow for the addition of labelling if a project desired. The dataset appeared as:

- [folder] bModeDataset
  - [folder]0000
    - [.png] Contains bMode Images
  - [folder] 0001
    - [.png] Contains bMode Images
  - [.json] Dataset

bModeBleed had a similar structured dataset that was created by the dataset tool. Post dataset creation, the data was copied to their respective machine so that training could commence.

4.1.2 *Training*

Training a GAN may require a large amount of computational resources, for instance, Stylegan2-ada proved to be a good example of A computationally expensive model. NVIDIA found that their own trials for the size of images (256 x 256) took 6 days and 21 hours to train prior to converging on their machines[19]. In our work, the bModeBleed model trained for approximately 5 days with some interruptions due to network technical issues. Note that for this reason, in some results, metrics for the training session to be broken into different parts (e.g., loss functions, FIDs, didn't start every time at their initial values after the training was restarted). This did not impact results obtained. bMode training was done locally for 2 days and was interrupted by unexpected crashes associated to Windows 11. Cloud services company Lambda Labs[4] was also tried for training bMode data, however, we found that a local system was sufficient for converging on results that were considered realistic by the TBI team with expertise in reading brain images[35].

In Stylegan models, a tick is an iteration count, i.e., every tick information gets displayed. While training our GAN, each tick changed the augmentation probability of the data, in other words, tick 1 may have an augmentation probability of 0.037 while tick 34 could have a probability of 0.654. As such, augmentation probability increases the chances of the image being augmented and by how much. Every 50 ticks, the training is paused, and FID scores are calculated. During

[4] Lambda Labs is a company that provides cloud services specific for deep learning tasks. This resource was used as an exploratory environment only.

this calculation, the model is assessed to determine how realistic the images are that are being produced. The result is a score and collection of images that depict both quantitative and visual representation of the model performance.

4.1.3 *Results*

Following the training of both bMode and bMode bleed training, metrics were calculated to help give a quantitative measurement of how the models are performing. That is, are they creating accurate synthetic data, and if not, what sort of issues may be occurring. The results presented reflect the steps as seen in the Methodology and experimentation setup sections. However, due to the nature of training, models were trained in a stop and go fashion, meaning that they were intermittently paused. Utilizing Tensorboard, metrics can be emitted, but they are broken into as many separate graphs as there were interruptions[36]. Therefore, the results and data presented here will be from the final training session of both bMode and bMode bleed models.

4.1.3.1 bMode StyleGan2-ada

Figure 4.1 and Figure 4.2 contain both initial and final images that were generated by the model, which gives clarity on the difference between the initial weights and how they performed post training. Utilizing the generator and random seeds, synthetic images were created to compute a color histogram as seen in Figure 4.4 where Figure 4.3 is the color histogram for the original dataset. These two figures are meant to be used together for comparison on how close generated images are to originals. In our experiments, these figures indicate that overall, the synthetic images behave similarly to the original ones, showing that it is possible to create a range of suitable fake images for some TBI research targets.

Table 4.2: StyleGan2-ada bMode Initial and Final Scores shows both initial and final metrics for the model, which is used to indicate how it performed. Figure 4.4 -

Figure 4.8 show how the model performed over time while it trained. These metrics were used to monitor the process of the model and ensure that mode collapse was not approaching or occurring.

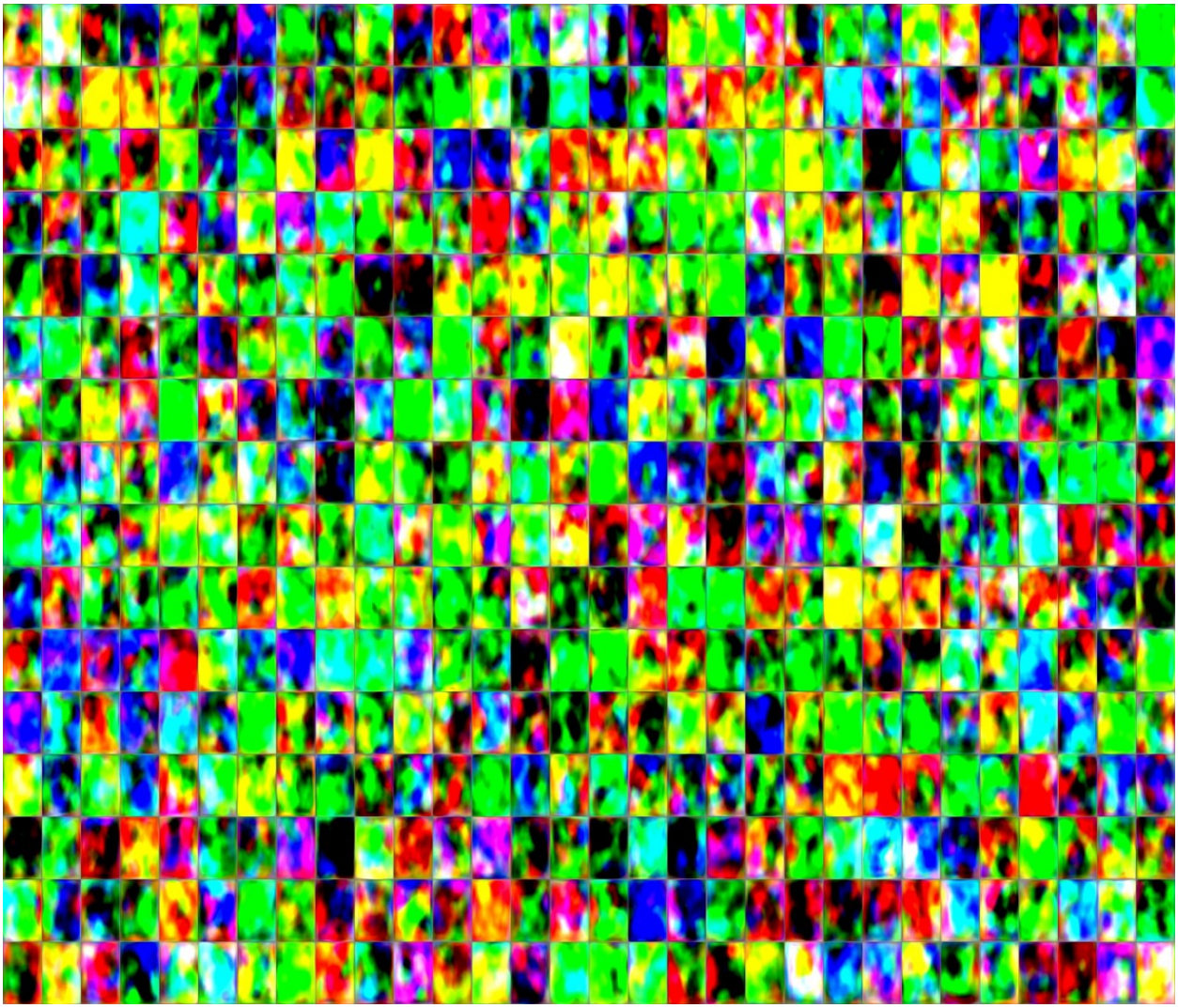

Figure 4.1: Initial StyleGan2-ada Generated bMode Images: This collage of images was created by the initial weights that were randomly selected by the model. In this case, it is expected that the outputs do not resemble the desired results. Using this approach gives a contrast between the initial and final place of the model.

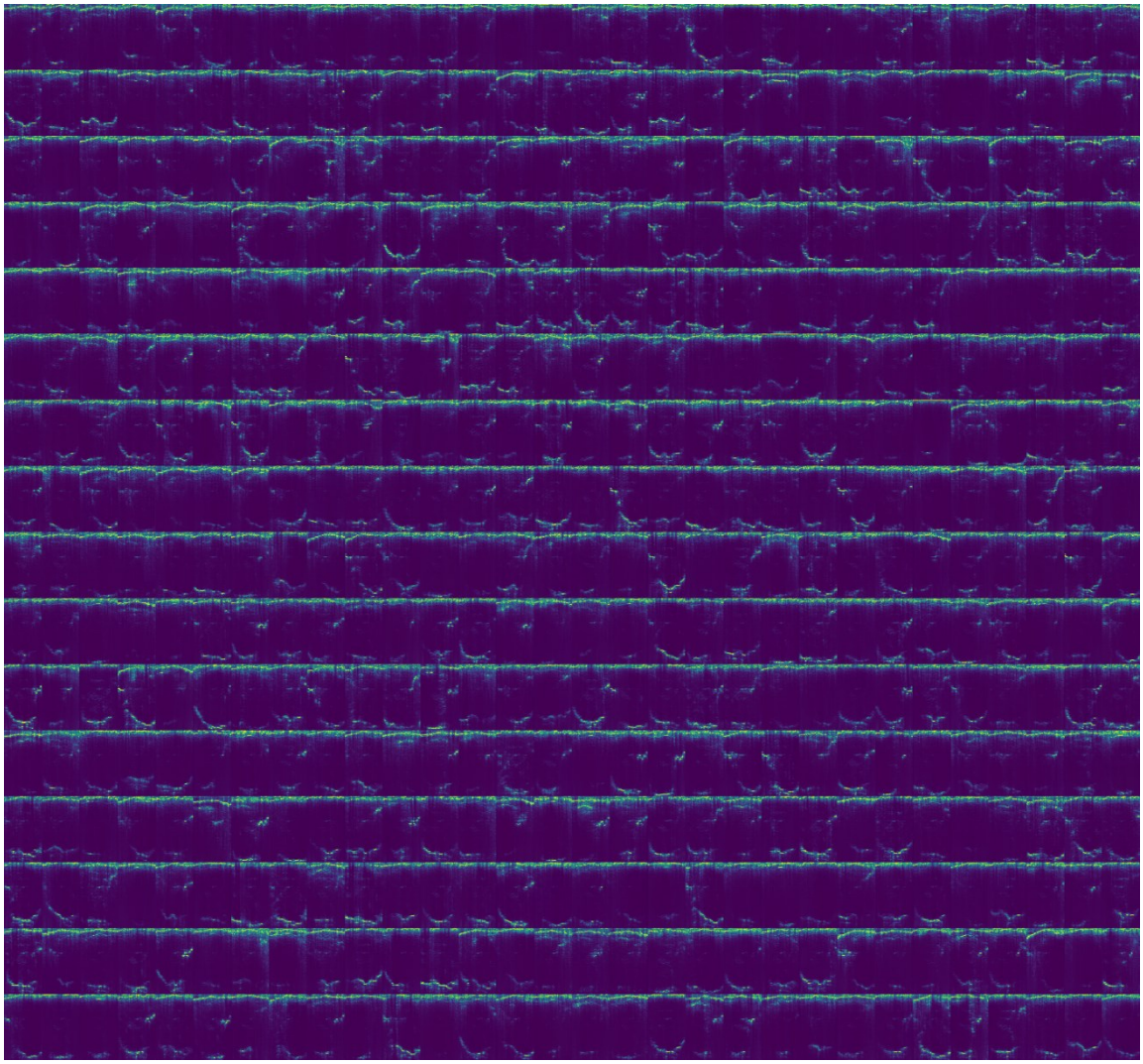

Figure 4.2: Final StyleGan2-ada Generated Fakes: The collage was generated after the final iteration of training, that is, these results are our final images. Each image in this case is randomly generated and depicts what the model believes the initial data appeared as. The synthetic images, upon inspection, appear to be representative of the initial dataset, but with a higher level of variance. The generated images will help provide a larger training data set size in future Pulsatility projects.

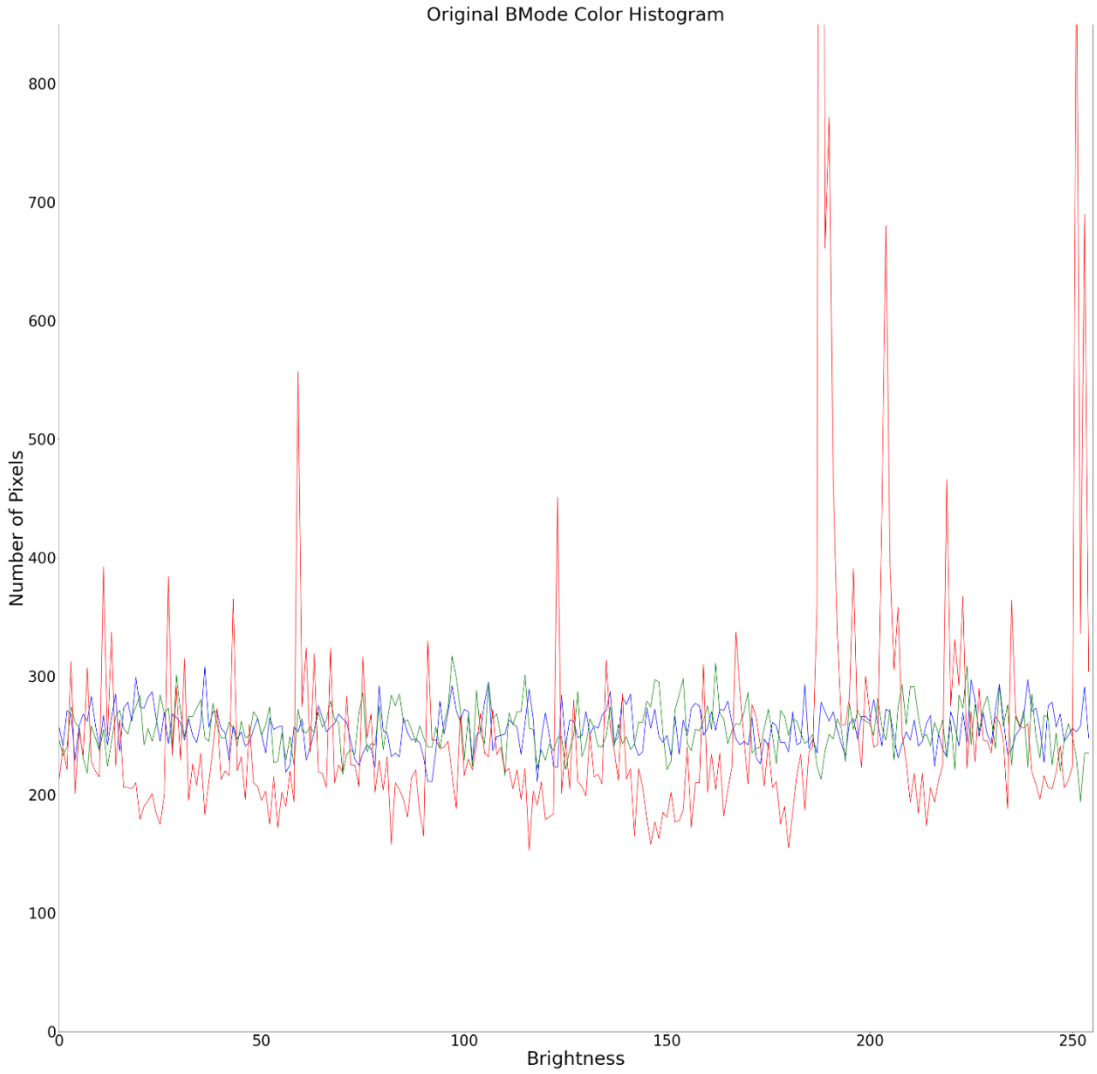

Figure 4.3: Original bMode Color "Histogram": Average distributions of all bMode images in the dataset. A color histogram can be used to show where the distribution of pixels is in a given image or set of images. Each color is split into its three RGB components, and there is one trace for each red, green, and blue intensities. The lines then represent colors distributions where the brightness values represent the intensity of pixels. The y axis indicates the number of pixels at that respective brightness level.

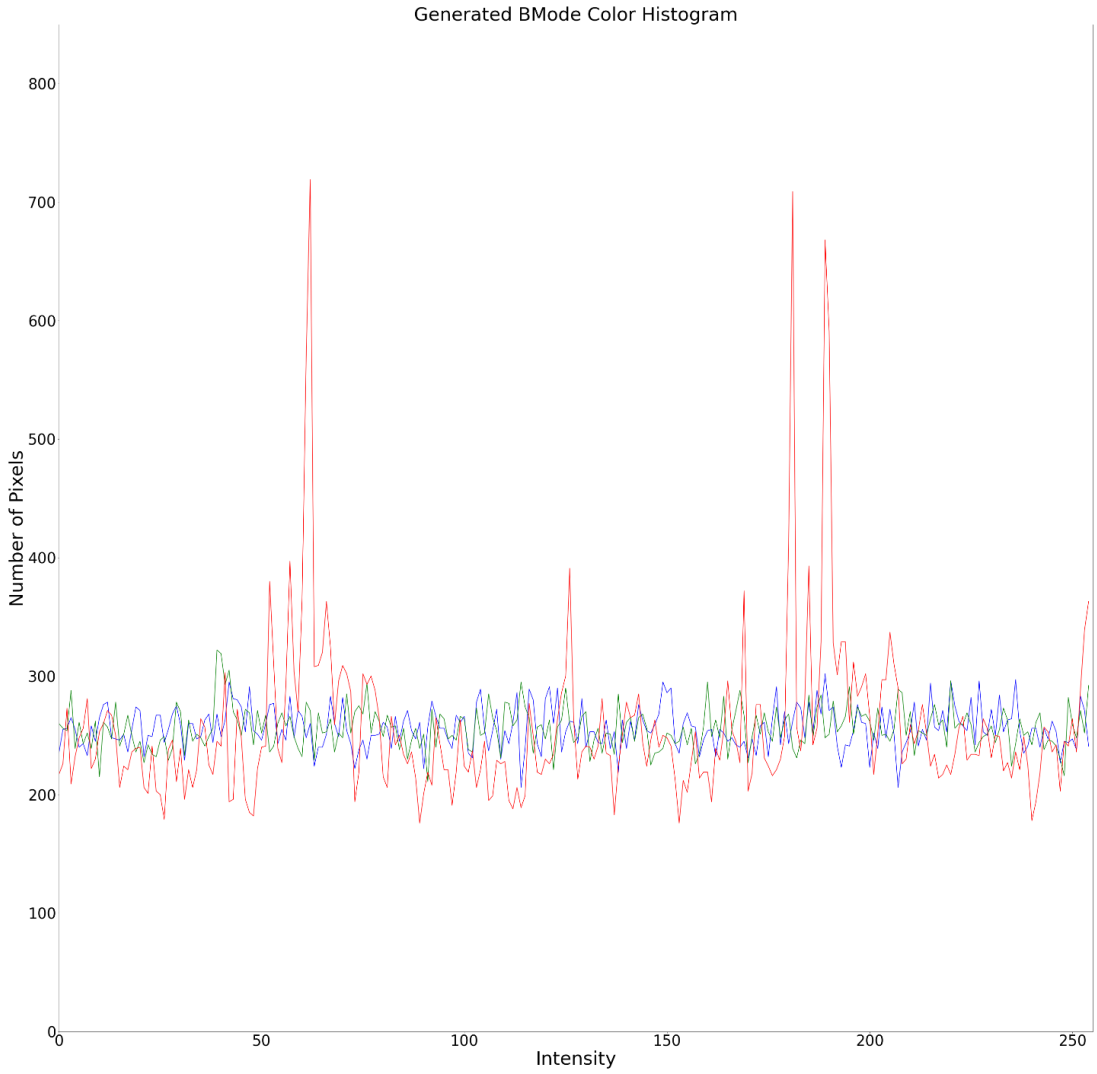

Figure 4.4: Generated bMode Color “Histogram”: Much like the previous color histogram, this image depicts the distribution of pixels and their intensities. This plot is meant to be compared with the original bMode Color Histogram, which in turn will give insight into how the pixels are distributed when produced synthetically.

Table 4.2: StyleGan2-ada bMode Initial and Final Scores

Metric	Initial Scores	Final Score
FID	287.1293	13.1112
KID	NA	0.0111
Precision	NA	0.4591
Recall	NA	0.327
Inception Score	NA	1.4801± 0.0060

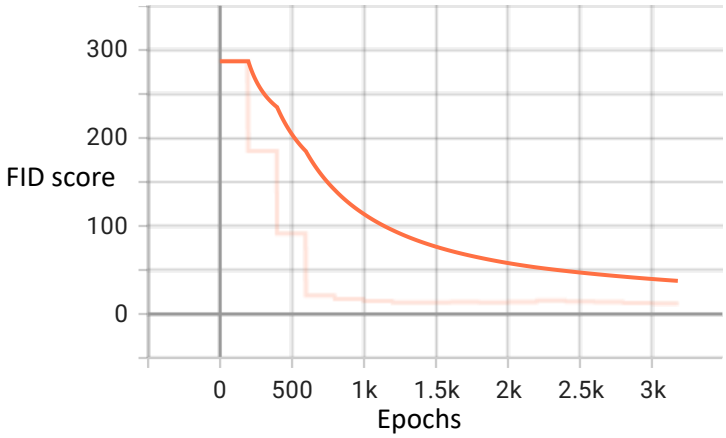

Figure 4.5: StyleGan2-ada bMode FID

Figure 4.5 indicates how the model trained over time by using the Fréchet Inception Distance or FID. In this image, we can see how closely related the generated images were at each step in the model. At the beginning of the training, there are noticeable changes (large steps), while towards the end the line shows convergence, indicating that at this point, images are very similar to each other.

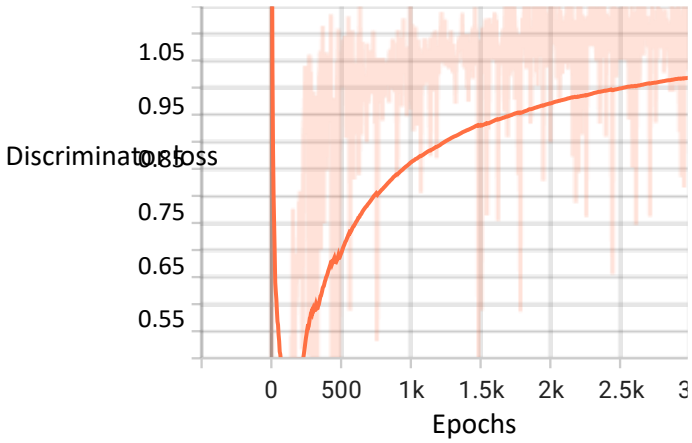

Figure 4.6: StyleGan2-ada bMode Discriminator Loss.

Figure 4.6 shows the loss of the discriminator at each step. The discriminator loss initially starts relatively high while the generator was training. After around 100 epochs, the discriminator loss begins to climb. This is expected as convergence is reached.

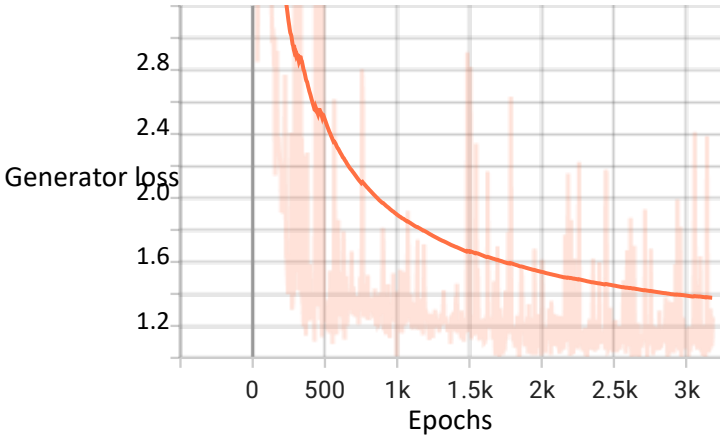

Figure 4.7: StyleGan2-ada bMode Generator Loss.

Figure 4.7 indicates how the generators loss was influenced by each training step. As training occurred, the generators loss decreased and could have continued on a downward trend, which indicates that a more representative image is being produced.

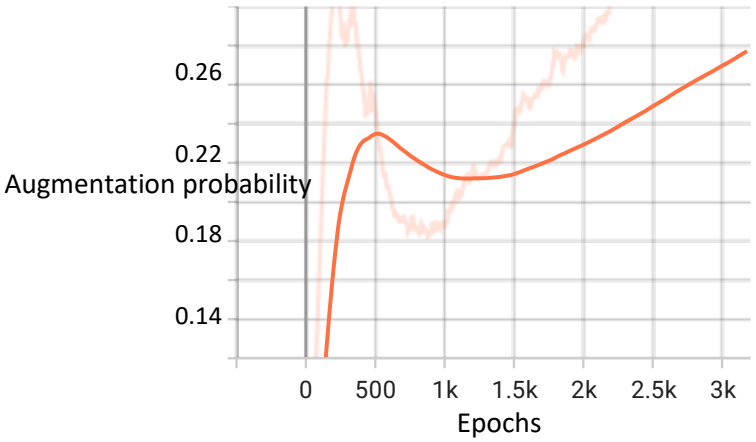

Figure 4.8 Augmentation Probability while Training: augmentation probability was used to add random noise to the training set at each epoch. The above graph indicates what the probability of random noise was applied at each step.

4.1.3.2 bMode Bleed StyleGan2-ada

Figure 4.9 contains synthetically generated images from the initial weights, while the final model was utilized to produce Figure 4.10, which contains the results of the model. The generated images are collected into one large png image and were generated in between training cycles. Each sub bMode bleed image is generated utilizing a random seed and is later used for analysis of the network. This information has been included in here due to network issues mentioned in 4.1.2, which affected the continuity of the experiments. The metrics collected in Table 4.3 depicts how the network performed after its final training period. FID was calculated after the first epoch to show the progression of the Stylegan2-ada model. All metrics besides FID were calculated on the final model. bMode Bleed Stylegan2-ada model trained with several interruptions, therefore reducing the metrics emitted to the last 7000 ticks. contains the initial and final scores for bMode Bleed Detection, while

Figure 4.11-

Figure 4.14 depict how the model trained over time.

Table 4.3: StyleGan2-ada bModeBleed Initial and Final Scores

Metric	**Initial Scores**	**Final Scores**
FID	276.3956	11.7886
KID	NA	0.0068
Precision	NA	0.4414
Recall	NA	0.3053
Inception Score	NA	1.597 ± 0.0080

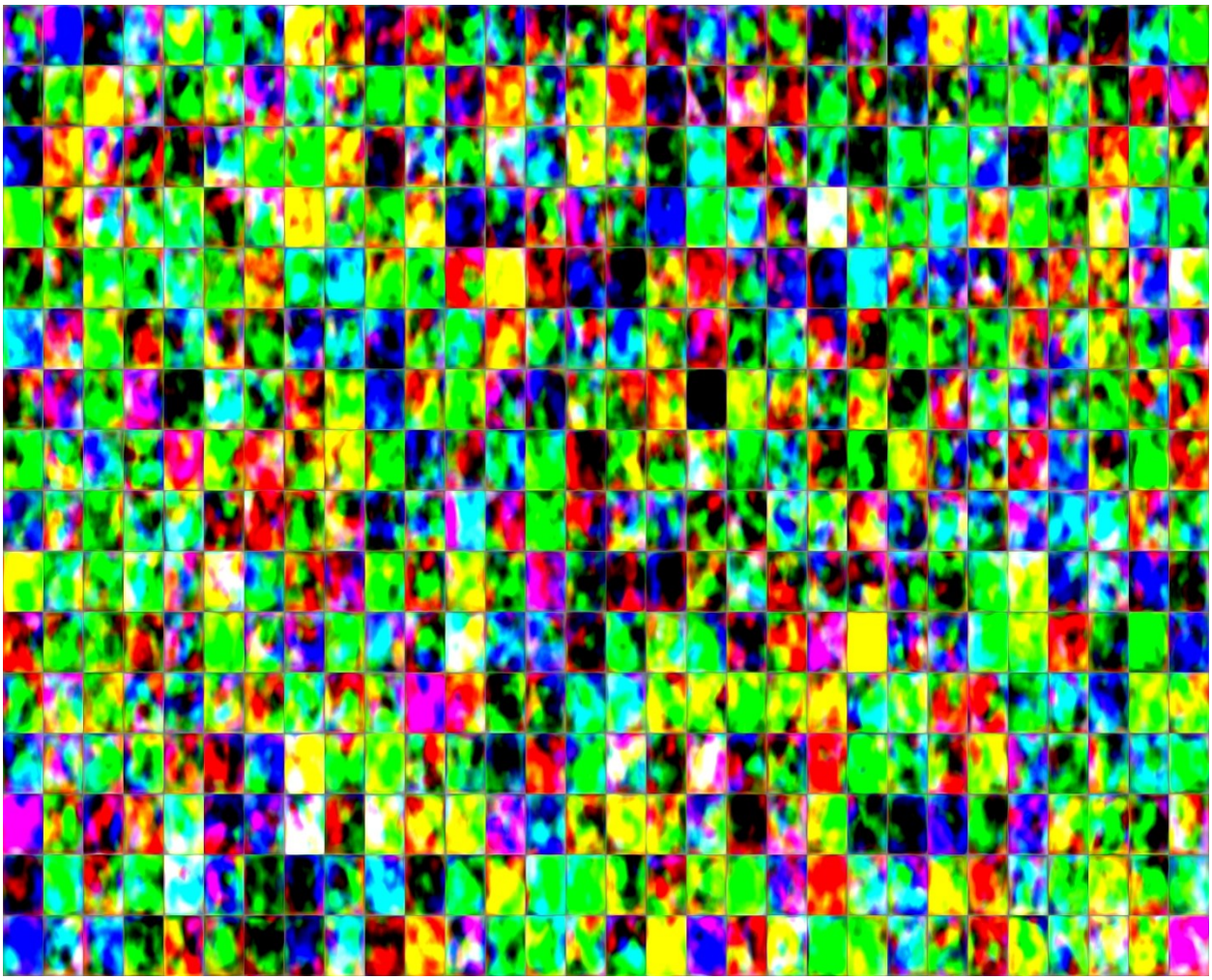

Figure 4.9: StyleGan2-ada bModeBleed: Initially Produced Images. The collage of images was created by the initial weights that were randomly selected by the model. In this case, it is

expected that the outputs do not resemble the desired results. Using this approach gives a contrast between the initial and final place of the model.

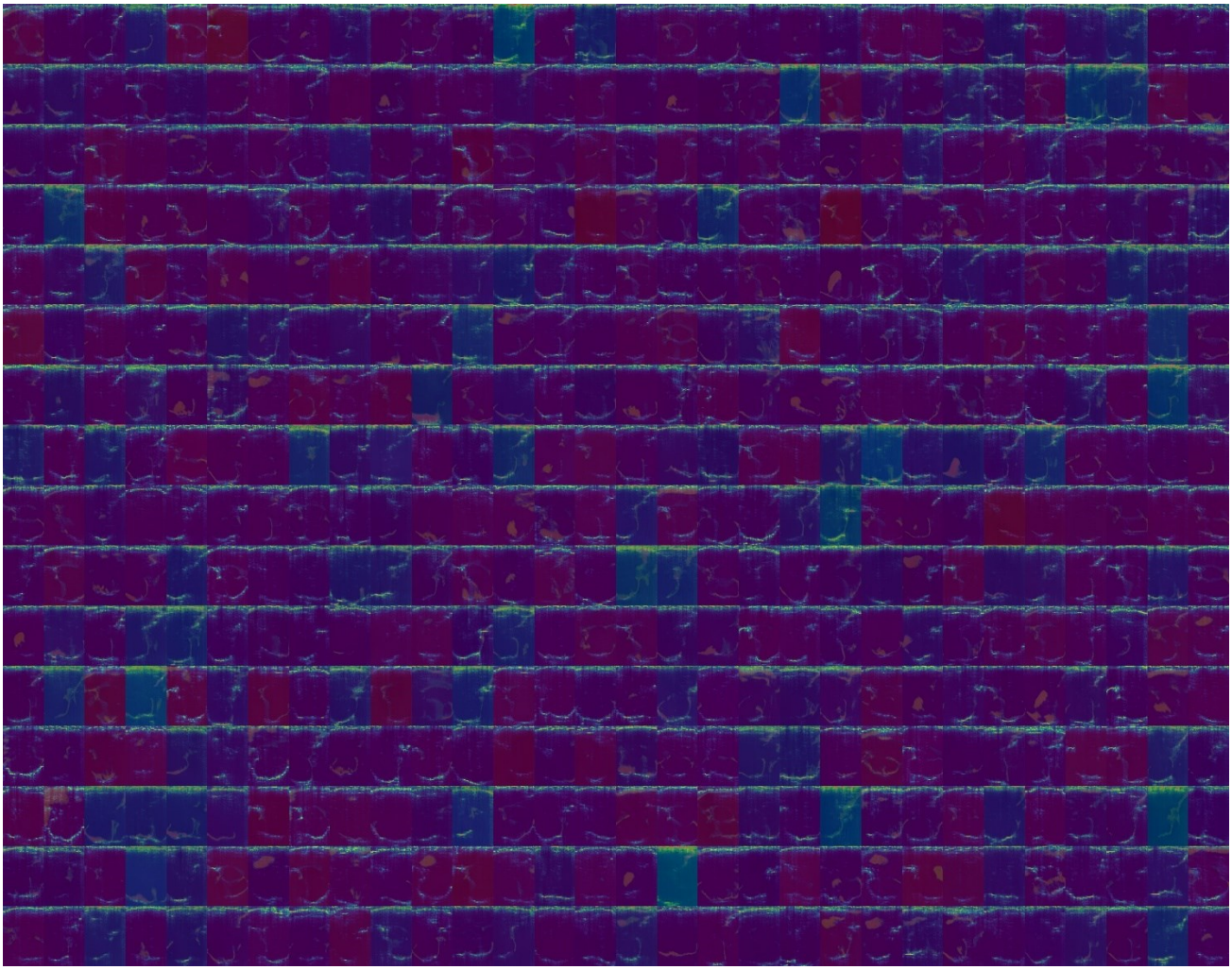

Figure 4.10 StyleGan2-ada bModeBleed Final Produced Images: This figure depicts synthetically generated bModeBleed images that have been created using the StyleGan2-ada model. The collage shows various different image types with varying blood, pixel distributions, and noise.

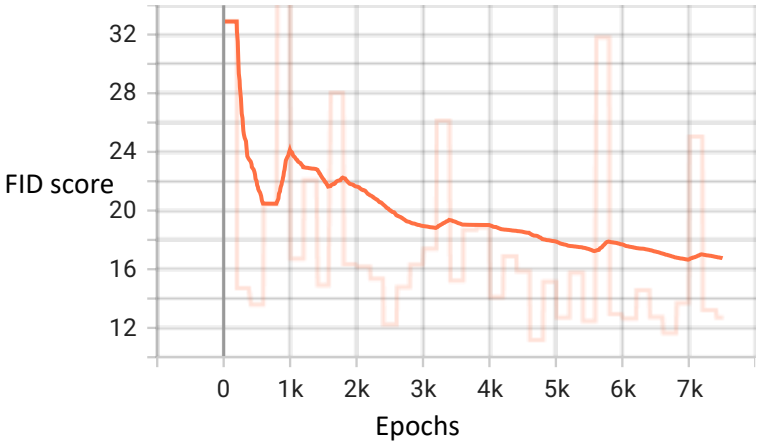

Figure 4.11: StyleGan2-ada bModeBleed FID Score.

Figure 4.11 shows how the model trained for the final epochs, it is worth mentioning that this graph does not start at the initial zeroth epoch. However, it does depict how the model trained and shows a downward trend indicating that there may be more performance left to obtain from the model.

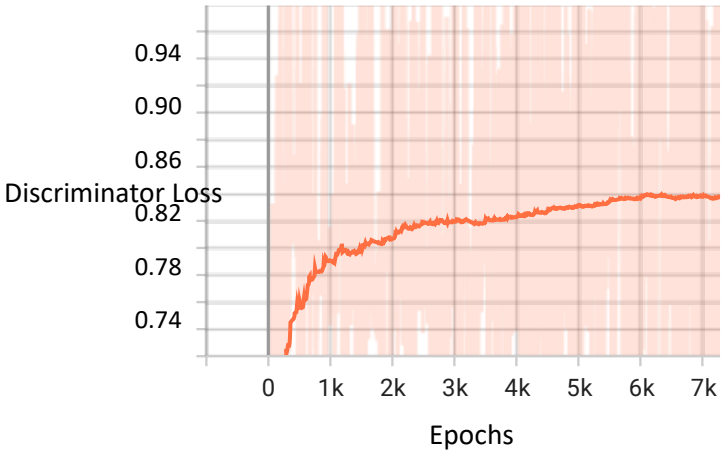

Figure 4.12: StyleGan2-ada bModeBleed Discriminator Loss.

The discriminator plot in Figure 4.12 indicates how the model trained for the final 7 thousand epochs. The result is a steady incline where the final iterations seemed to show a tapering discriminator loss.

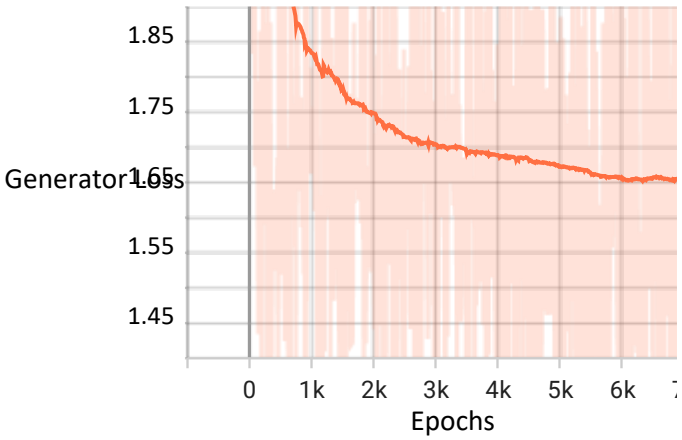

Figure 4.13: StyleGan2-ada bModeBleed Generator Loss.

The final generator loss for the final 7 thousand epochs is depicted above and can be seen as progressing downwards but tapering off in Figure 4.13. Additional training time may not have resulted in a better performing generator, however, there still could've been room for growth. Figure 4.14 below indicates how the augmentation probability for this experiment was added at each step.

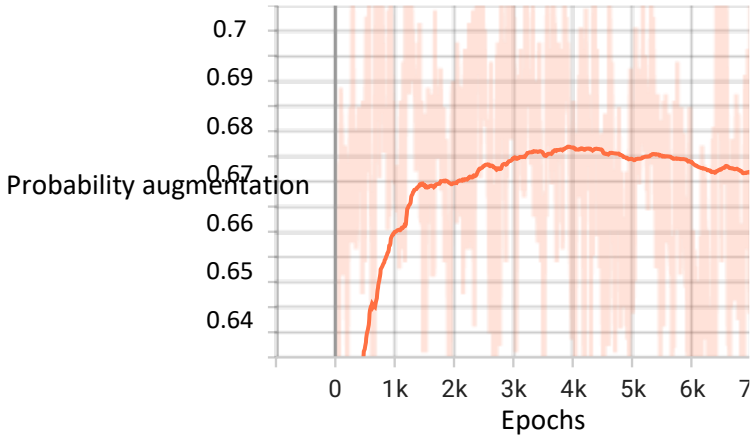

Figure 4.14: StyleGan2-ada bModeBleed Augmentation Probability.

4.2 Skull and Ventricle Detection

4.2.1 *Experiment Setup*

Skull and ventricle detection both followed a similar approach when setting up the experiments. In both cases, Anaconda was used to set up a local environment that would be used

to conduct both detection models. The model and preprocessing utilized anaconda, a research scientist focused package manager. Using anaconda, OpenCV, Pytorch, Tensorflow, and many more packages were installed which would grant access to the necessary tools to build and train a model[37], [38]. Table 4.4 depicts the specifications of the computer that would be used to prepare and train the detection models. Both models required the following data storage path:

- [folder] trainingData: Containing either bMode skull or displacement ventricle paired training data as images
- [folder] testPatients: Containing either bMode skull or displacement ventricle paired test patient data as images
- [folder] score:
    - [folder] generatedImages: where generated skullMask images were stored
    - [folder] labels: where the label images were stored
- [folder] discriminator_checkpoints: contained discriminator model checkpoints
- [folder] Generator_checkpoint: contained generator model checkpoints

Table 4.4: System Hardware used in Skull and Ventricle Detection

Hardware or Software	**Details**
Operating System	Windows 11
CPU	AMD Ryzen 3900X – 12 cores, 24 threads
RAM	DDR4 32gb
GPU	Nvidia 3090 – 24gb GDDR6X

4.2.2 *Training*

Training the models required some prelaunch steps and careful consideration of hardware specifications. Due to the limitations of hardware, the models were not trained in a distributed fashion, instead were trained individually on a single GPU. The images were trained at a size of 256 x 256 pixels, which was the maximum sized MATLAB array in the original dataset. The models were both initially started with the same parameters as seen in Table 4.5. The skull detection model trained for approximately 1 day while the ventricle detection model trained for approximately 3 days.

Generative adversarial networks can be incredibly difficult to train, requiring significant manual intervention to adjust training parameters. In each case, to avoid mode collapse or getting stuck in a local minimum, learning rates were adjusted on a case-by-case basis. In each case of mode collapse, the model's discriminator would converge on either 100% accuracy or 0% accuracy. Once hitting this, the model's performance would crash and produce the same or similar images repetitively. The case where images are repeated, or the generator is no longer function is called mode collapse. In this state, the model needed to be manual shut off and the learning rate adjusted[39]. After adjusting the learning rate, the model would be restarted from the last checkpoint prior to where the FID crashed as seen in Figure 4.18 in results.

Table 4.5: Initial Skull and Ventricle Training Parameters

Parameter	**Value**
Batch Size	32
Kernel Size	4x4
Learning Rates (both Generator and Discriminator)	2e-6

4.2.3 *Results*

During each epoch of the training process, the FID was calculated for the model to assess the performance and save a copy of the model if it surpassed the previous best FID score. While this approach took more time, it did yield a more in depth look as to how the training process went. It is worth mentioning that some models stopped training due to external factors which will change how the tensorboard graphs look. The final FID was calculated using the final generator model and was compared to the original data points target label.

4.2.3.1 Skull Detection

Skull detection produced outputs that can be seen Figure 4.21 where the left most image is bMode, the middle image is the target, and the right most image is the Generator generated image. Table 4.6 depicts the results of the model, where the score represents the final or initial metric calculation. It is worth mentioning that Tensorboard was not used until a later time, which would have been after Skull detection was conducted. However, sample training was done to depict how the training process looked. This sample run includes a scenario where mode collapse

occurs sending discriminator loss for fake images to 0 and discriminator loss for real images to 1 which can be seen in

Figure 4.19 and

Figure 4.20. It is worth noting that the FID also suffers when the mode collapse which can be seen in Figure 4.18. While training, the model's learning rate was adjusted from 2E-6 to 2E-4 in an attempt to break out of a local minima's, but the solution only worked for a short time before hitting another minima that couldn't be broken out of.

Table 4.6 Skull Detection Final Metrics

Metric	**Score**
Initial Fréchet Inception Distance	300
Final Fréchet Inception Distance	55.4725
Average MSE	2350.6086
Average RMSE	47.8996
MS-SSIM	0.6005

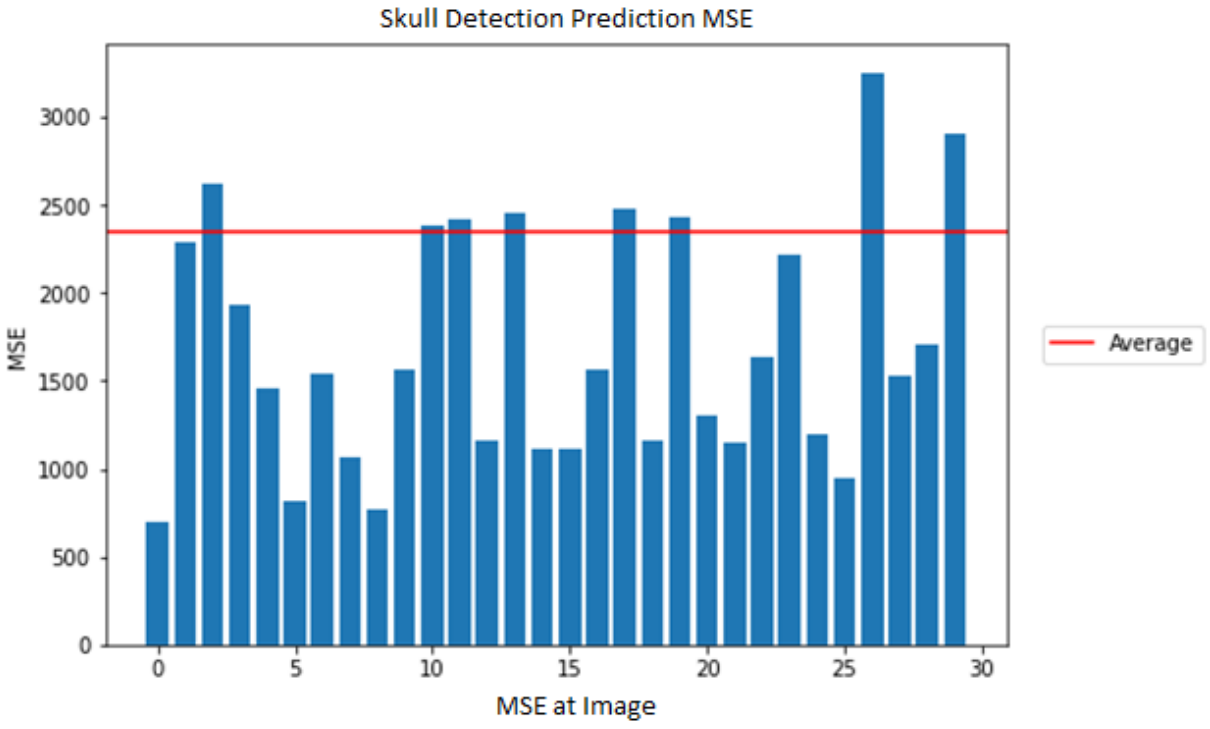

Figure 4.15: Skull Detection MSE where x is the generated or predicted image an

Figure 4.15 above indicates the mean squared error for the results obtained by the skull detection model. The red line depicts the calculated average of all the images, that is if they were all averaged together the value would have been 2350. The image shows that there were a few outliers that had worse performance, whereas there were 12 below 1500. 8 of the predictions were higher than the average, which indicates that the outliers heavily swayed the Average MSE. The MSE was calculated using the generated images and the label images.

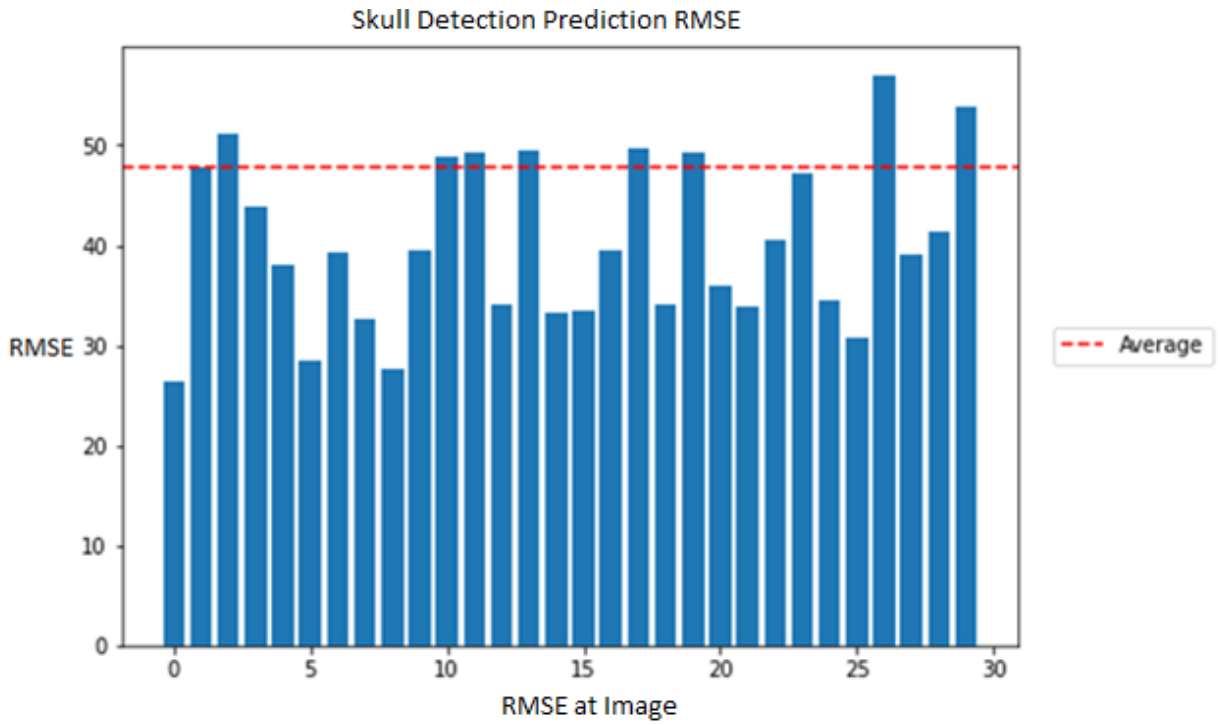

Figure 4.16: Skull Detection RMSE

Much like MSE, the RMSE in Figure 4.16 depicts the relation or difference between the original and generated values. The relationship being inspected is that of the original skull mask and the generated skull mask, which results in a depiction of how well the model performed.

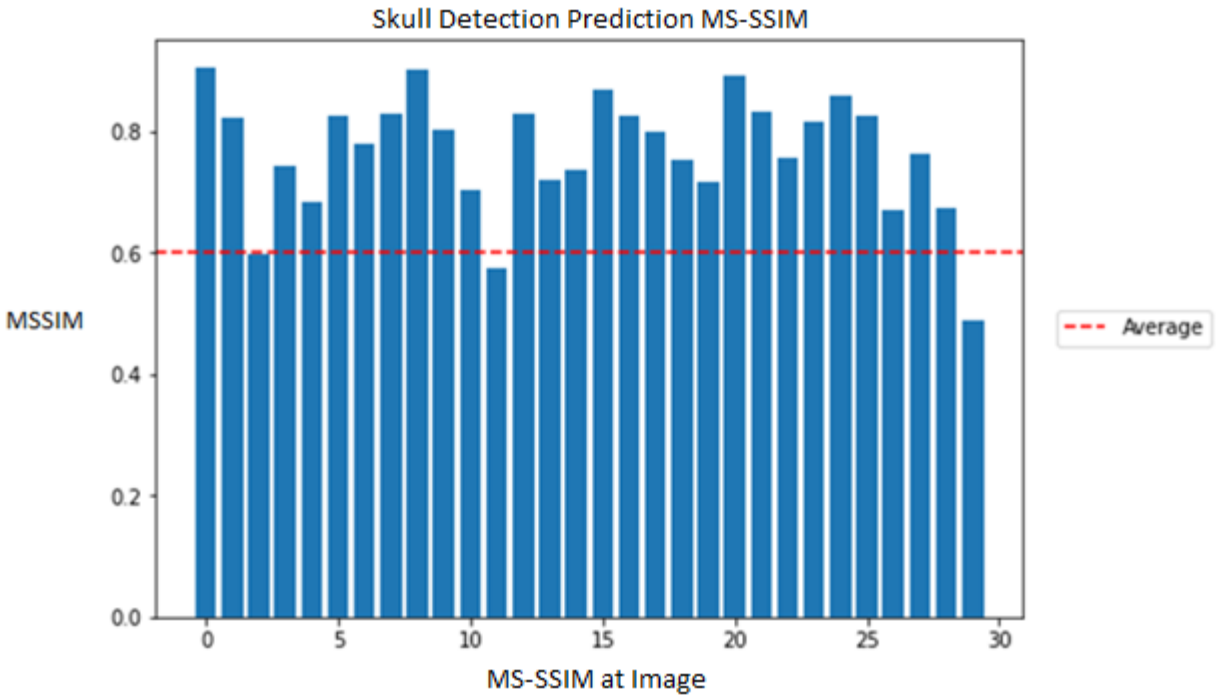

Figure 4.17: Skull Detection MS-SSIM

The MS-SSIM bar plot in Figure 4.17 indicates the perceived change in images, meaning that the label is compared to the generated skull data. A value of 1 is deemed to be the same, whereas a value of 0 would indicate that they are not at alike. The graph shows that the average value is 0.6, 26 of them being higher than the average.

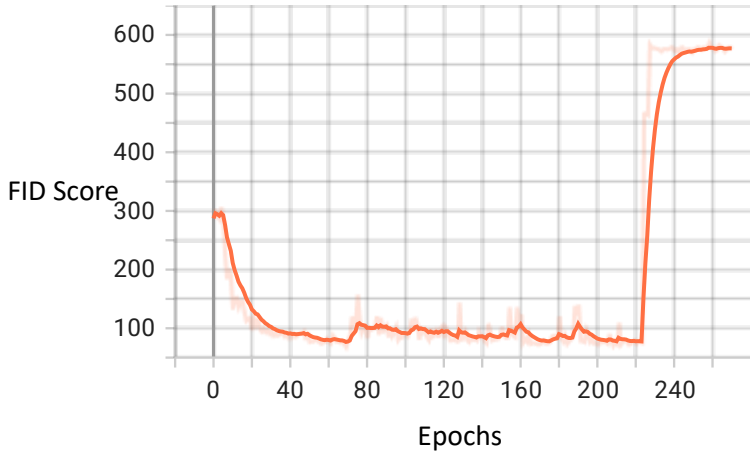

Figure 4.18: Skull Detection FID (y) at each Epoch (x)

The FID graph above indicates how the FID performed at each epoch. In this example, the GAN entered a state of mode collapse which resulted in the FID calculations spiking and the model became inaccurate.

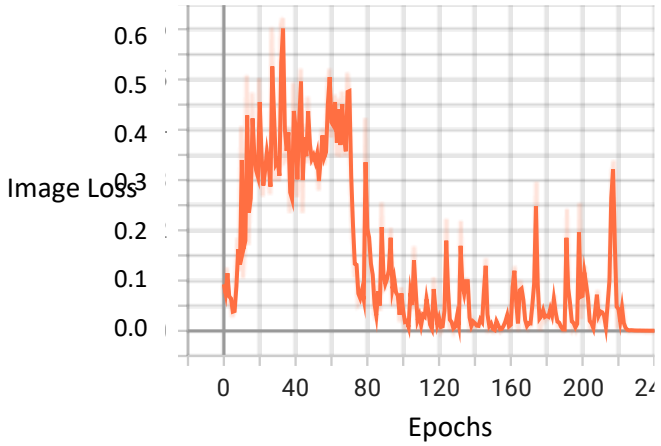

Figure 4.19: Skull Detection Fake Image Loss

In Figure 4.19 above we can see fake image loss during the final 250 epochs. As epochs went on, the fake image loss approached zero until it reached zero. Reaching zero was a result of mode collapse as indicated previously. Then, image 4.20 below, shows the real image loss, that is, the images that were real but incorrectly classified. Eventually it approached 1 indicating mode collapse, meaning that classification accuracy would always be 100%.

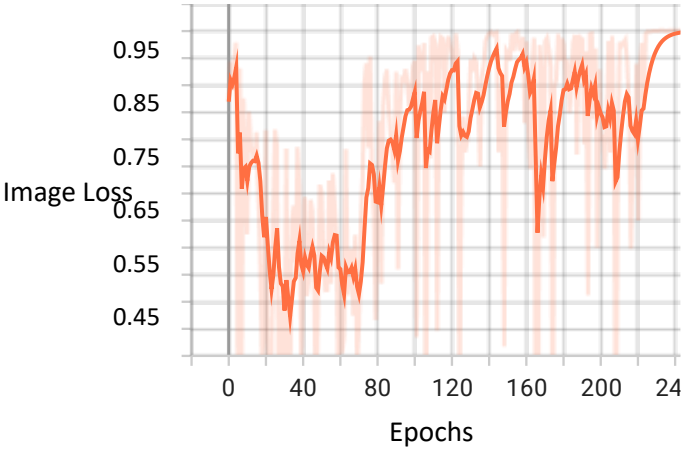

Figure 4.20: Skull Detection Real Image Loss

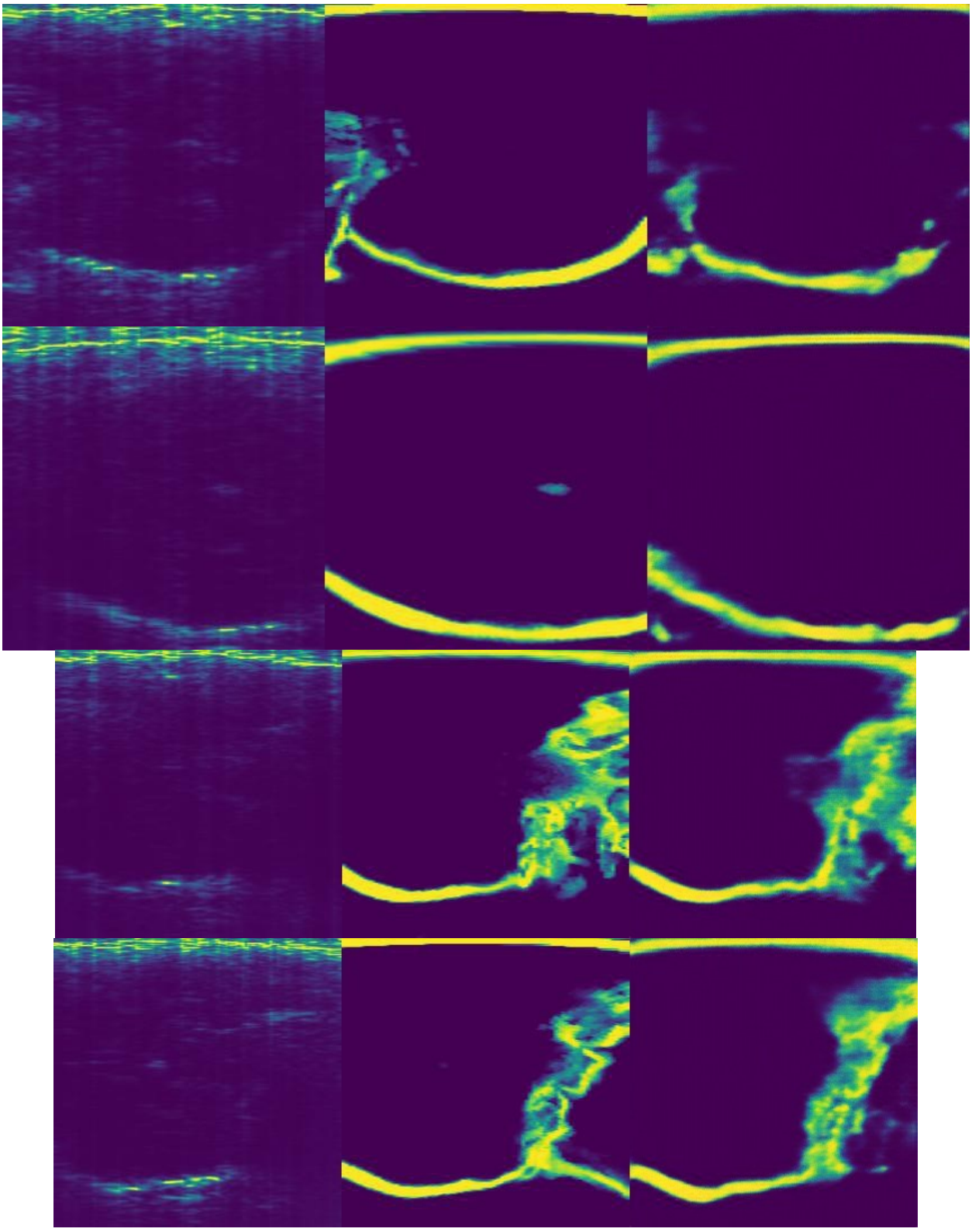

Figure 4.21: Skull Detection as input, target, prediction where rows are patients

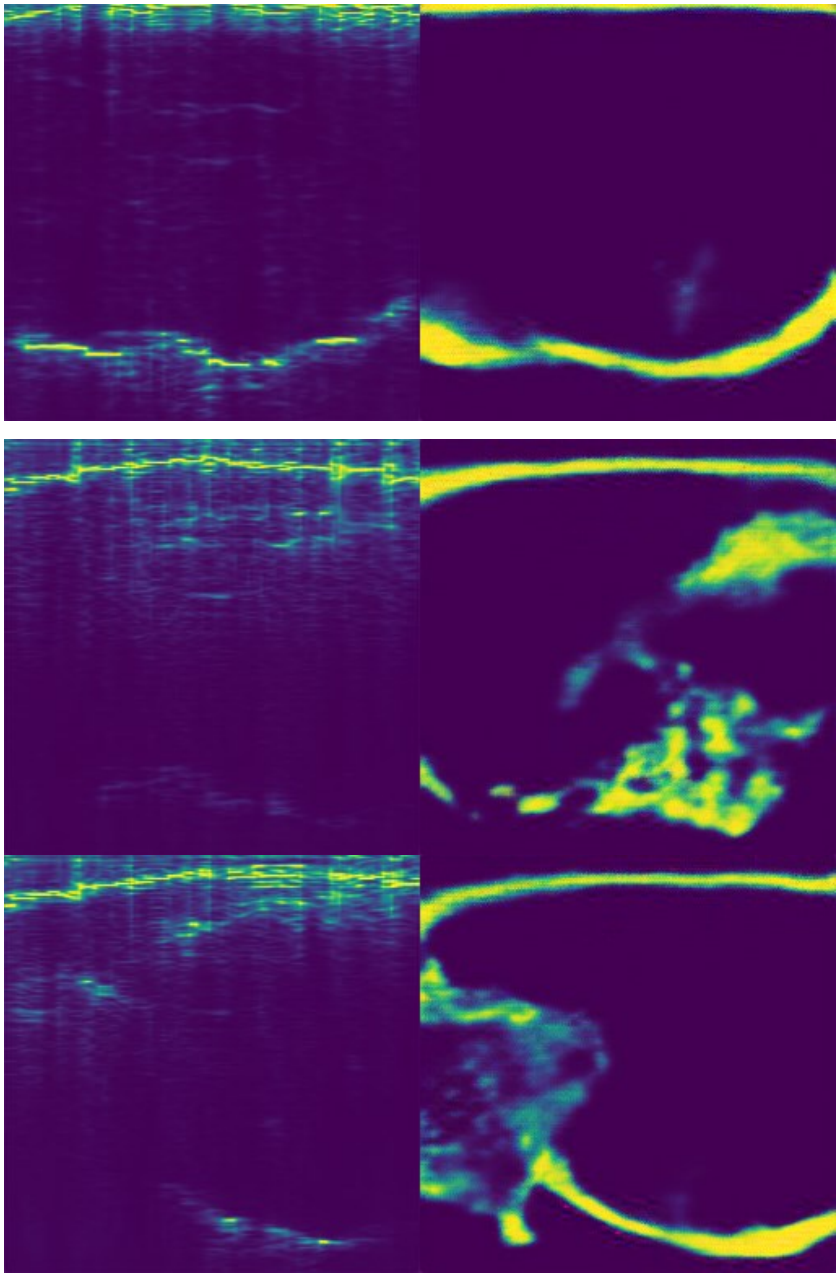

Figure 4.22: Skull Detection where left is Synthetically generated bMode by StyleGan2-ada, and the right is Pix2Pix model prediction

4.2.3.2 Ventricle Detection

Ventricle detection trained without intervention for 1000 epochs before reaching a point of potential convergence. Manual intervention was done to adjust the hyperparameters but was unsuccessful in increasing the FID scores and getting out of these potential local minima. The

model initially started at an FID score of 302 and after 1000 epochs reached a peak FID performance of 77.208 as seen in Table 4.7 and
Figure 4.26. The Discriminators real and fake loss, seen in
Figure 4.27 and
Figure 4.28 reached a point of healthy training where each maintaining approximately a 50/50 split on prediction. The produced ventricle images can be seen in Figure 4.29 and can be read as input, target, prediction.

Table 4.7: Ventricle Detection Final Metrics

Metric	**Score**
Initial Fréchet Inception Distance	302
Final Fréchet Inception Distance	77.207
Average MSE	112.2176
Average RMSE	9.2996
MS-SSIM	0.929

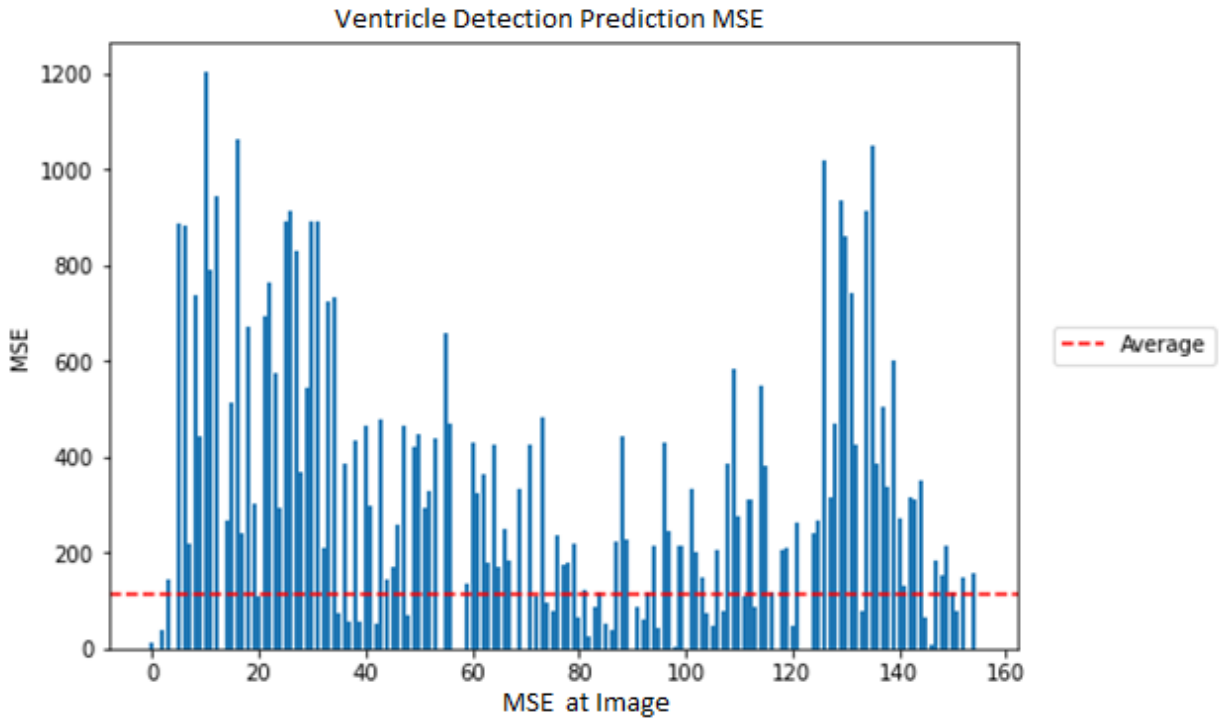

Figure 4.23: Ventricle Detection MSE

The above graph depicts the generated ventricles MSE when compared to the ventricle label. The average MSE was 112 with several outliers reaching as much as 1200.

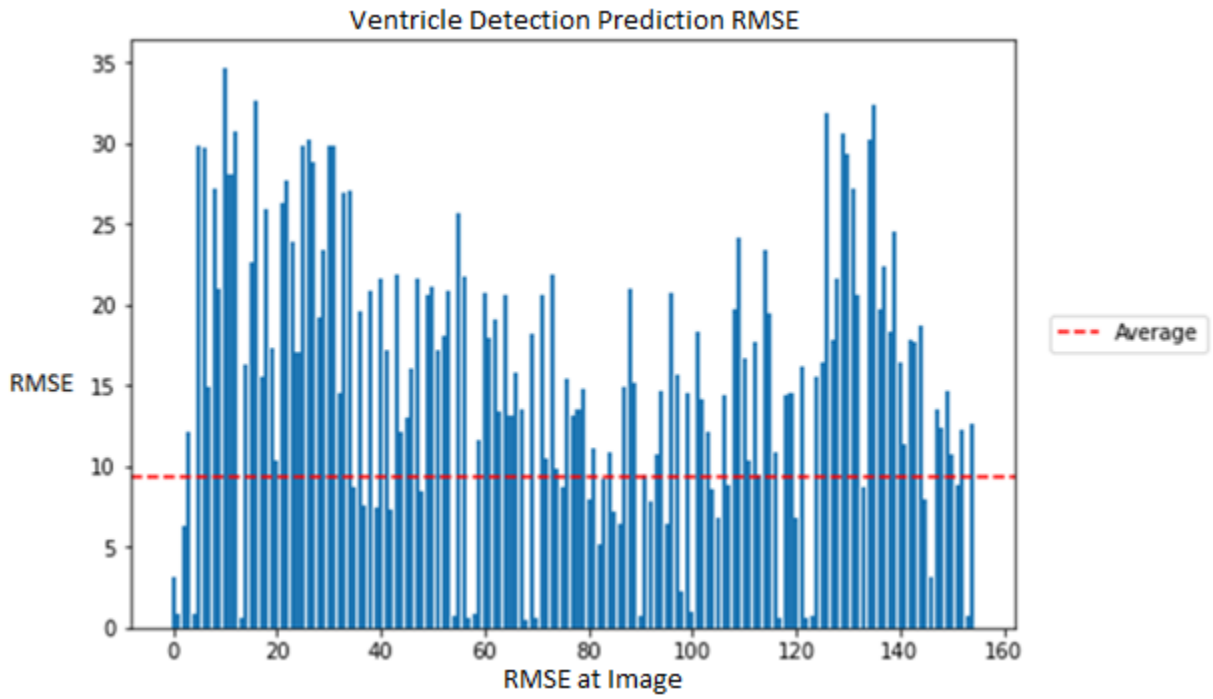

Figure 4.24: Ventricle Detection RMSE

The figure depicted above indicates the displacement of ventricles generated image and label image RMSE calculations. The average was slightly over 9 where the RMSE was distributed on both sides of the average in a equal manner.

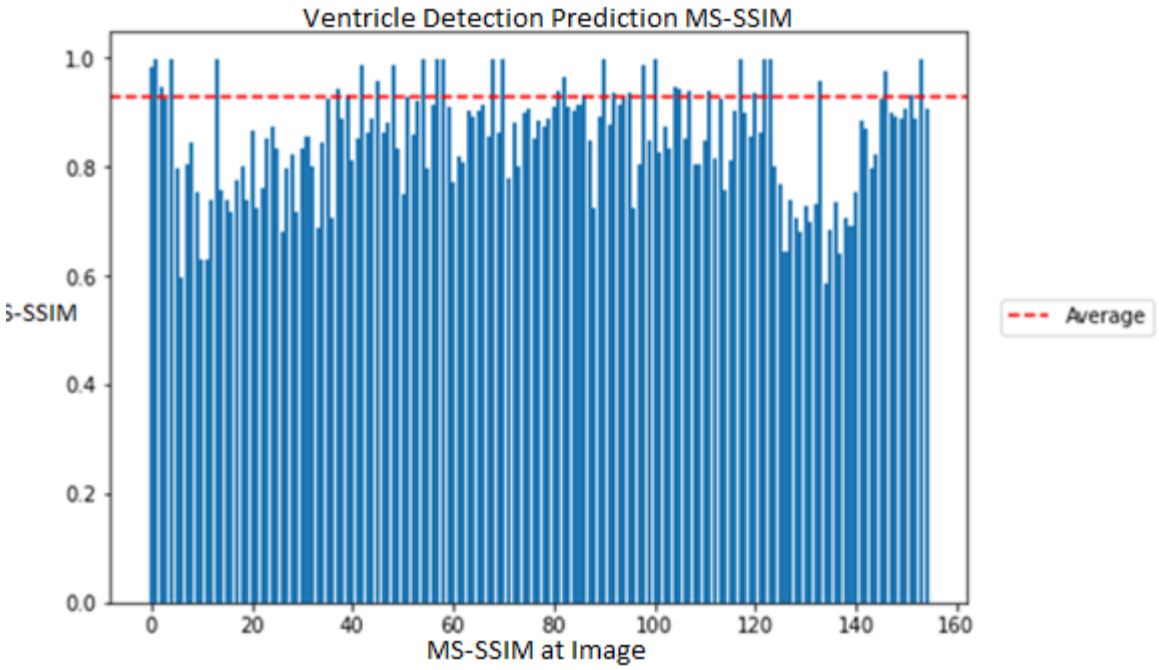

Figure 4.25: Ventricle Detection MS-SSIM

The above MS-SSIM graph was produced using the label ventricle data and generated ventricles from the test set. The average was 0.9 with most of the generated images below 0.9. The graph indicates the generated ventricles were very similar to the label data which indicates the model performed well.

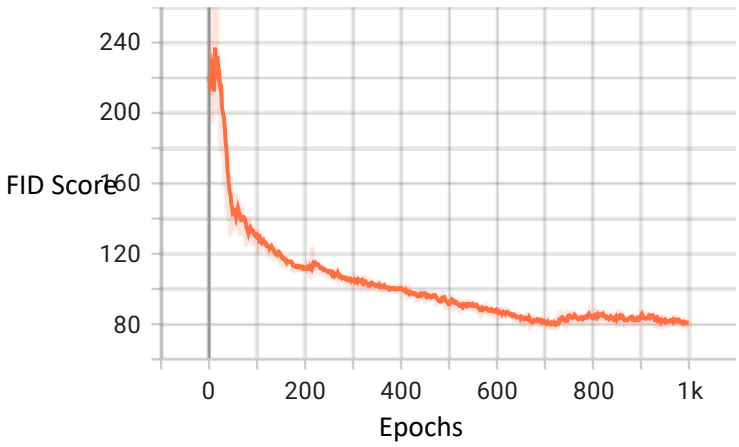

Figure 4.26: Ventricle Detection FID

The FID plot in Figure 4.26 was produced during an end-to-end training session. The FID started at 302 and ended at 77, which indicates progression while training. However, at 700 epochs, the training appeared to drop off which may indicate that a local minima or global minima may have been reached.

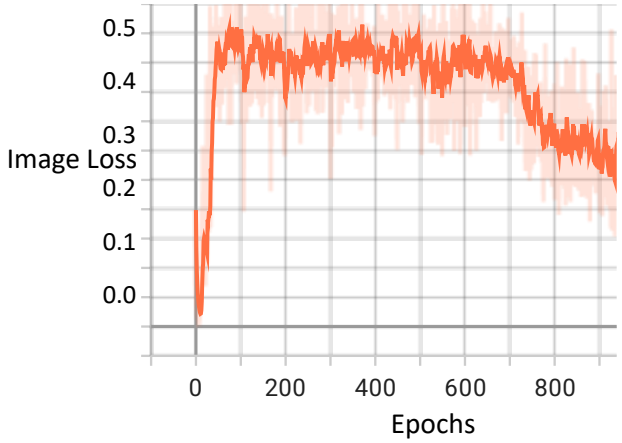

Figure 4.27: Ventricle Detection Fake Image Loss.

The fake image loss initially started near 0 and reached a healthy training state of approximately 0.5 but tapered off to 0.3 once we hit 700 epochs.

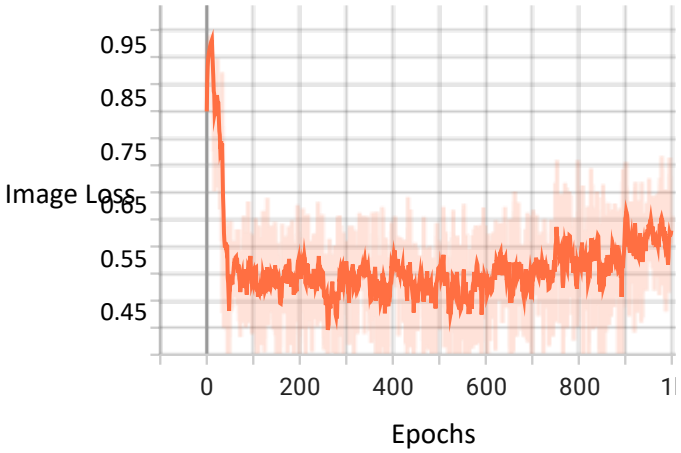

Figure 4.28: Ventricle Detection Real Image Loss where x is epoch and y is real image loss

In Figure 4.28, we can see the real image loss initially started at or near 1 and eventually settled near 0.55 once a healthy training balance was reached. The image loss was slightly higher than 50 percent during the entire training session.

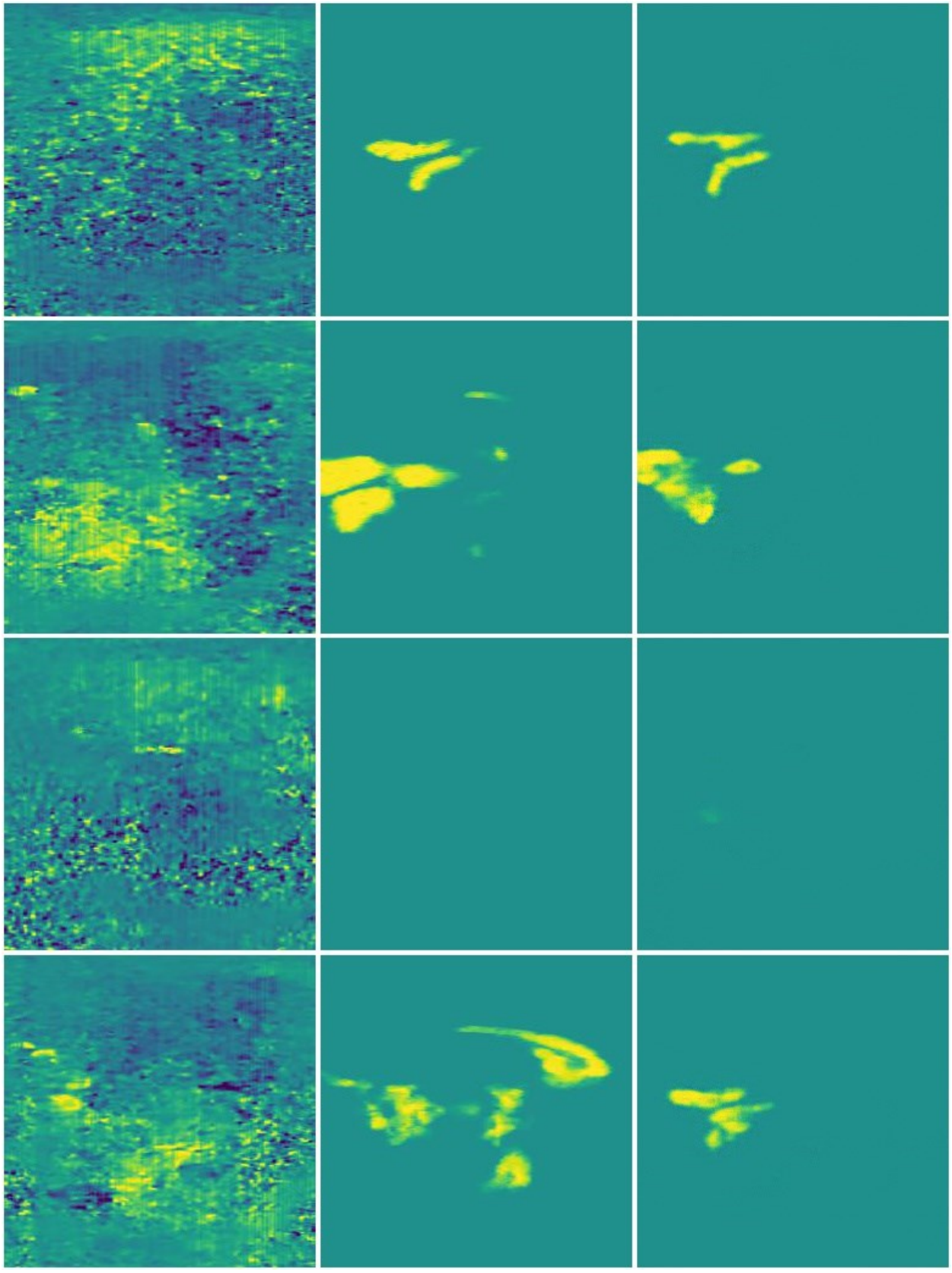

Figure 4.29: Ventricle Detection input, target, prediction where rows are patients

4.3 BLOOD DETECTION

4.3.1 *Experiment Setup*

Blood detection can be broken down into 2 distinct models, one that utilized layered bModeBleed and bMode data, while the other used displacement and bloodMaskThick. In each case, they followed the same model setup procedures, just with different data and training parameters. Table 4.8 depicts the hardware that both models trained on, however, it is worth noting that both models did not train for an adequate amount of time. Given more computational resources, it is possible that both models may progress further. Unlike previously detected skull and ventricle, these models are emitting more training data at each epoch, which in turn greatly slows down the time needed to train the models. Both models had the same folder structure that is depicted as:

- [folder] trainingData: Containing either bModeBleed and bMode or bloodMaskThick and displacement paired training data as images
- [folder] testPatients: Containing either bModeBleed and bMode or bloodMaskThick and displacement test patient data as images
- [folder] score:
    - [folder] generatedImages: where generated images were stored
    - [folder] labels: where the label images were stored
- [folder] discriminator_checkpoints: contained discriminator model checkpoints
- [folder] Generator_checkpoint: contained generator model checkpoints
    - Each model checkpoint is around 0.7 gigabytes which stopped the training process because it filled the solid-state drive.

Table 4.8: Blood Detection Hardware

Hardware or Software	**Details**
Operating System	Windows 11
CPU	AMD Ryzen 3900X – 12 cores, 24 threads
RAM	DDR4 32gb
GPU	Nvidia 3090 – 24gb GDDR6X

4.3.2 *Training*

Each model required some tweaking prior to be able to run for an extended period of time. Learning rates varied from model to model, where some would be fine, others would enter mode collapse quickly. Which can be evident based on the generator and discriminator loss, there is a trend of one or the other going to 1 or 0, where the same image would be produced repeatedly. The images trained at a size of 256 x 256 pixels, which was the maximum sized MATLAB array in the original dataset. The displacement-based model trained for approximately 1 day, and the bMode model trained for 12 hours before shutting down prematurely due to storage restraints.
Much like skull and ventricle detection, the models were difficult to train and at times would require manual intervention. Due to the interventions, the model output figures for model results may look incomplete, but the most recent training session is included. The models initial training parameter appeared as seen in Table 4.9. However, learning rates were adjusted to break out of local minimas where necessary.

Table 4.9: Blood Detection Initial Training Parameters

Parameter	**Value**
Batch Size	64
Kernel Size	4x4
Learning Rates (both Generator and Discriminator)	2e-6

4.3.3 *Results*

Both bMode and displacement-based models produced the same measurements for analysis, which includes the following:

- Average Mean Squared Error
- Discriminator Loss
- Generator Loss
- Fréchet Inception Distance

4.3.3.1 bMode Blood Detection

The bMode bleed detection model did not need any additional learning rate nor batch size adjustments. However, it was interrupted at 50 epochs because models were failing to save due to storage constraints. Files were removed from the drive, and continued training. The metrics and diagrams were from the latest iterations of bModeBleed Pix2Pix model training. Table 4.10 defines the final model results, that is after training concluded. MSE, RMSE, and MS-SSIM utilized the entire testing set to conduct results and can be found in Figure 4.30 through Figure 4.32. While training, the model emitted metrics as seen in

Figure 4.33 through

Figure 4.36 which was used to ensure training remained healthy and no intervention was needed. The test set was used with the generator to create results that can be found in Figure 4.37.

Table 4.10: bMode Blood Detection Model Results

Metric	**Score**
Initial Fréchet Inception Distance	340
Final Fréchet Inception Distance	112.376
Average MSE	264.4094
Average RMSE	15.6536
MS-SSIM	0.83

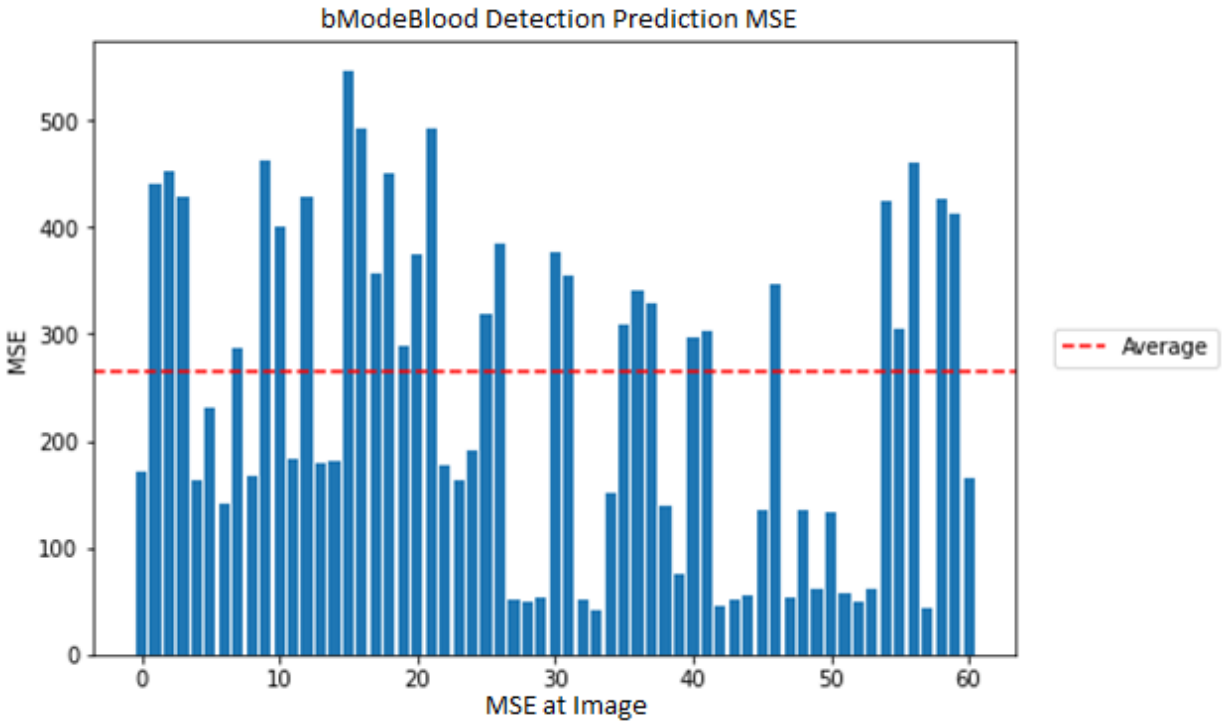

Figure 4.30: bModeBlood MSE

The above graph indicates the MSE for the final models generated images when compared to the bModeBlood label. The MSE average is 264, with a distribution that leans to being under 264. The results depict that the images, from the standpoint of MSE, are similar.

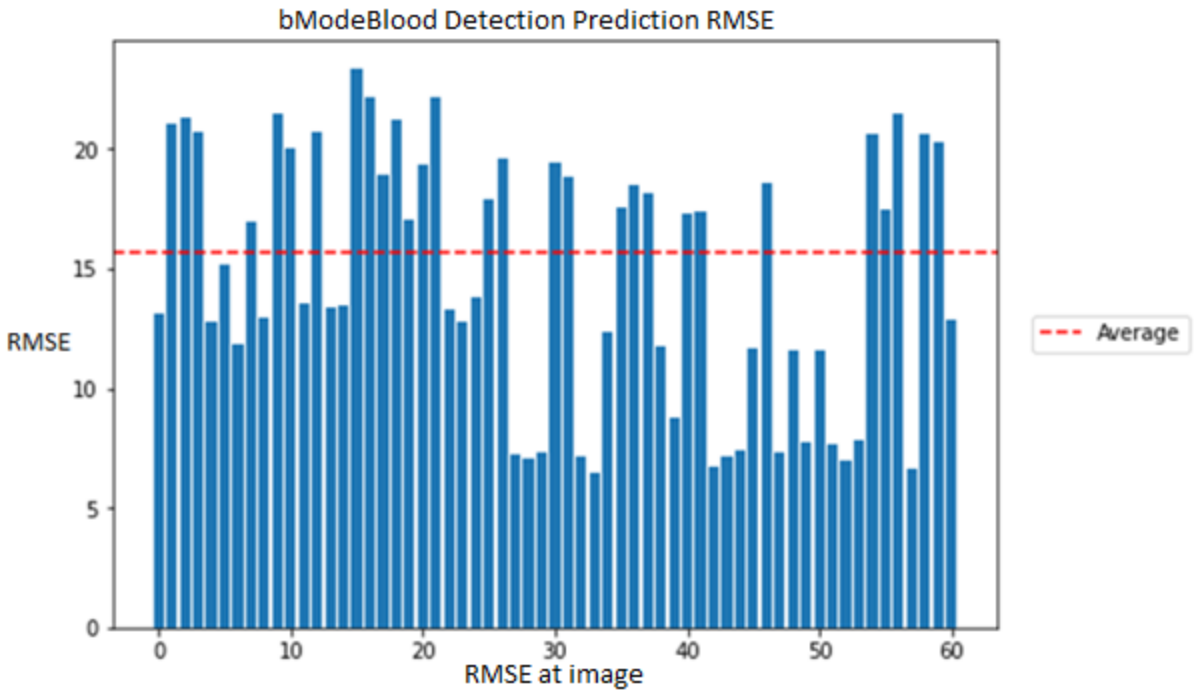

Figure 4.31: bModeBlood RMSE

The above RMSE graph indicates how the bModeBlood model performed, where the average RMSE was 15 and most of the distribution was under.

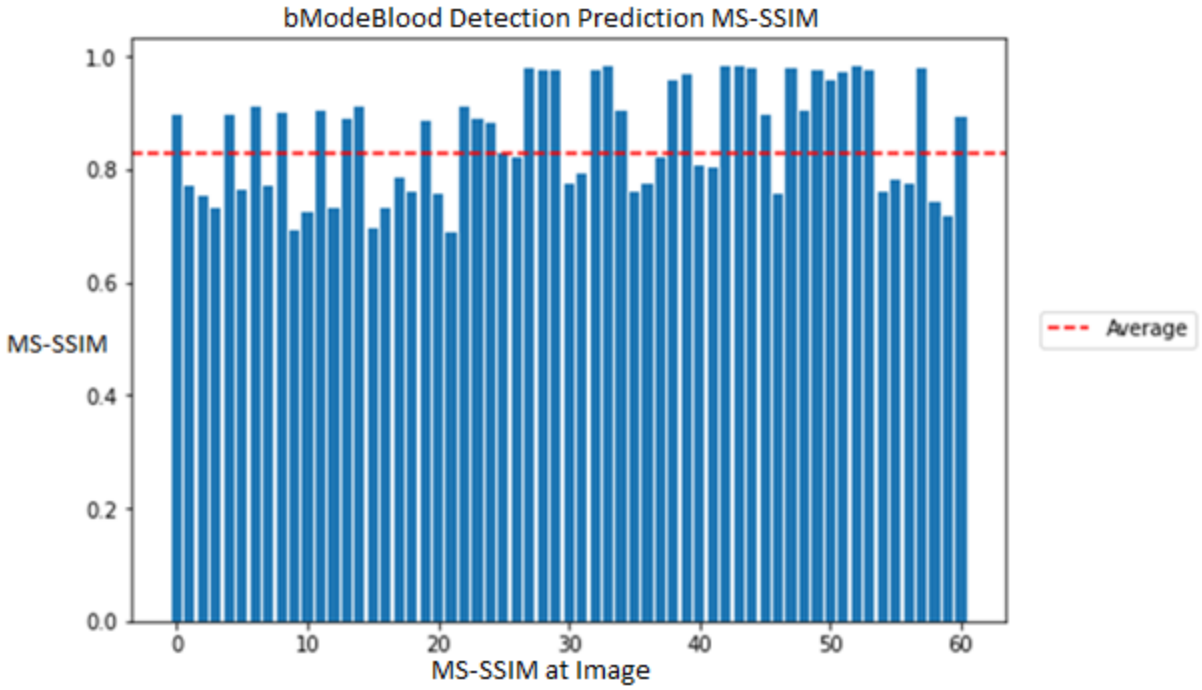

Figure 4.32: bModeBlood MS-SSIM

Figure 4.32 bar plot above shows the MS-SSIM metrics for the model's performance when comparing the generated and label data. The average was 0.83 and was well distributed above and below. The results indicate the model produced images that were structurally similar to that of the label data. In 4.33 below, we can see the FID can be seen at each epoch and initially started at 340 and finished at 112 while training for 800 epochs. The downward trend indicates that there may be more progress to be made on the model with more training time. However, the model progressed and created more realistic images with each epoch.

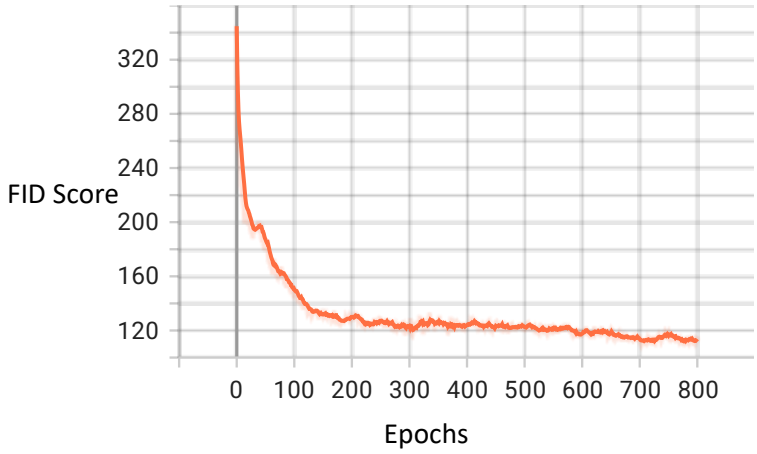

Figure 4.33: bModeBlood FID

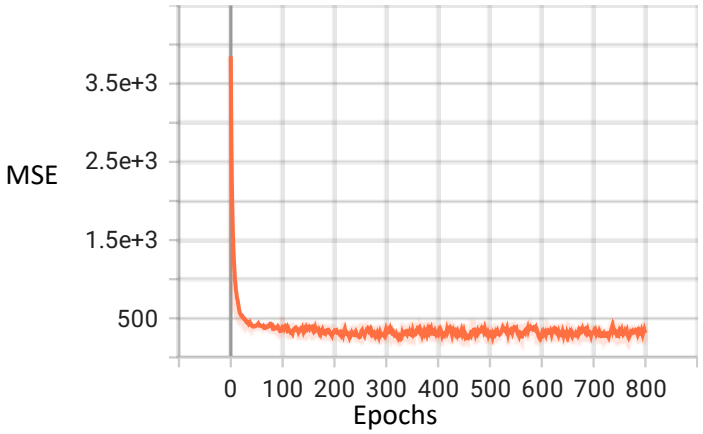

Figure 4.34: bModeBlood Training MSE

The average MSE for each epoch can be seen above in figure 4.34. Initially, the MSE was quite high, but once the 50th epoch was reached, the MSE stayed flat indicating good model convergence. Which indicates that the metric may not be as suitable for differentiate the generated images or was simply not sensitive enough. The generator loss for bModeBlood can be seen next in Figure 4.3; initially started quite high, but eventually flattened out around 300 epochs again, indicating reasonably rapid convergence.

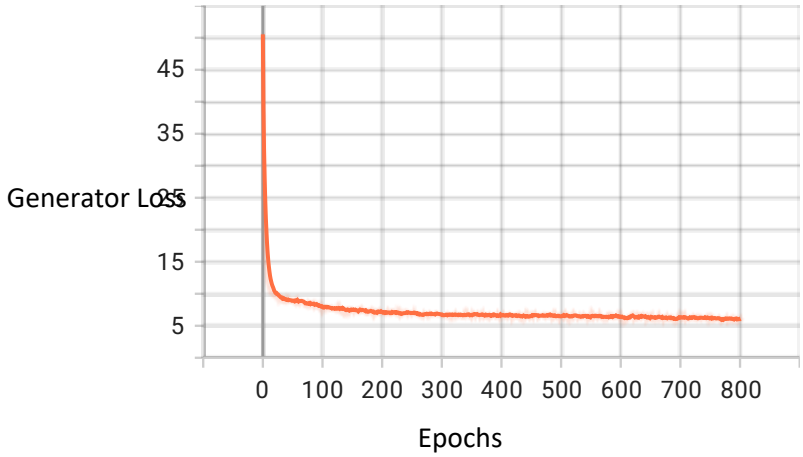

Figure 4.35 bModeBlood Training Generator Loss.

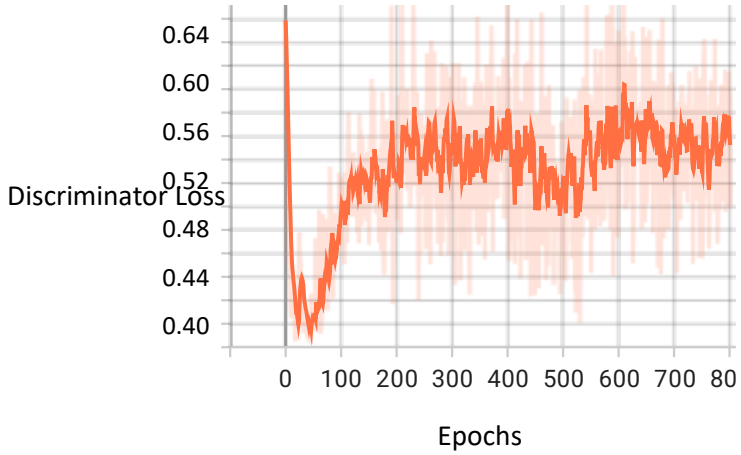

Figure 4.36 bModeBlood Training Discriminator Loss: showing how the discriminator loss started slightly higher than 0.68 and maintained a healthy training rate after the 200th epoch afterwards.

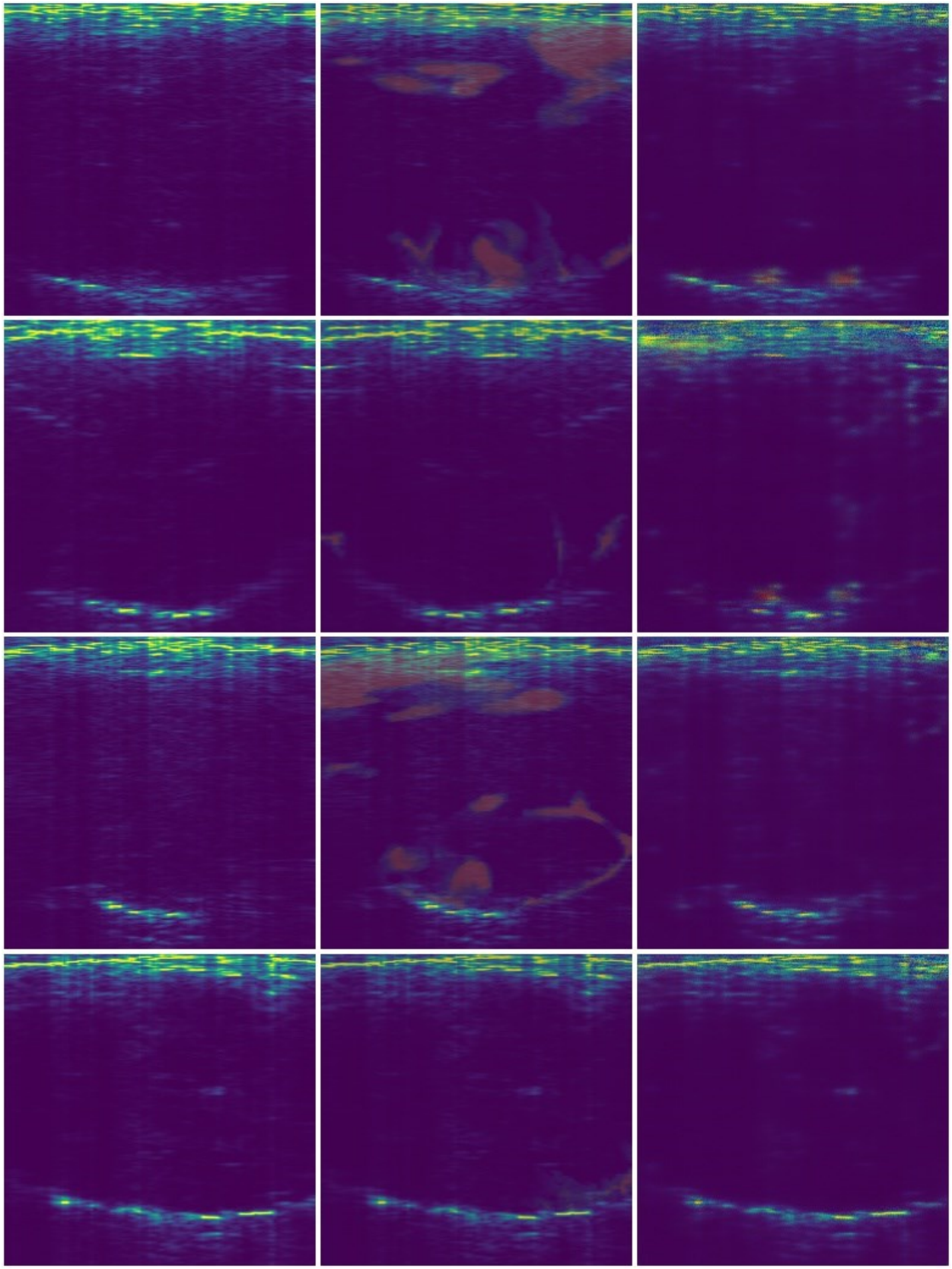

Figure 4.37: bModeBlood Detection for input, target, prediction where rows are patients

4.3.3.2 Displacement Blood Detection

The displacement blood detection Pix2Pix model required tweaking of learning rates while training, which still resulted in mode collapse, which can be seen in the figures depicted below. The final stable model, that is before FID moved to 350, was used to create the model metrics. Table 4.11 elaborates on the final performance of the model, that is after training was done, and utilized the test set. The metrics and diagrams were from the latest iterations of displacement blood Pix2Pix model training. Table 4.11 defines the final model results, that is after training concluded. MSE, RMSE, and MS-SSIM utilized the entire testing set to conduct results and can be found in Figure 4.42 through Figure 4.44. While training, the model emitted metrics as seen in Figure 4.38 through
Figure 4.41 which was used to ensure training remained healthy and no intervention was needed. The test set was used along with the generator to create results that can be found in Figure 4.45.

Table 4.11: Displacement Blood Detection Final Results

Metric	**Score**
Initial Fréchet Inception Distance	310
Final Fréchet Inception Distance	195.5018
Average MSE	37.941
Average RMSE	5.4692
MS-SSIM	0.98116

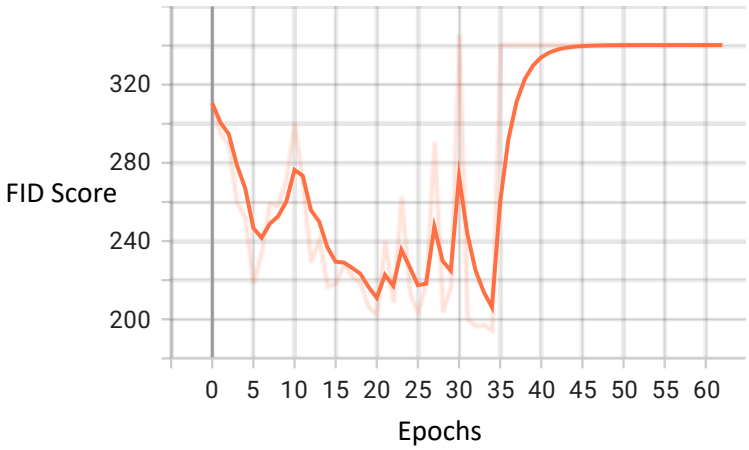

Figure 4.38: Displacement Blood Detection FID

Figure 4.38 shows the final iteration of displacement-based blood detection and shows how FID progressed with each epoch. The model did not train for a long period of time, each module iteration (change of hyperparameters) resulted in mode collapse after a short period of time. The FID started around 310 and had a low of 195 before mode collapse occurred. Figures 4.39 through 4.41 below show the corresponding training MSE, discriminator and generator loss respectively, all for this blood detection experiment.

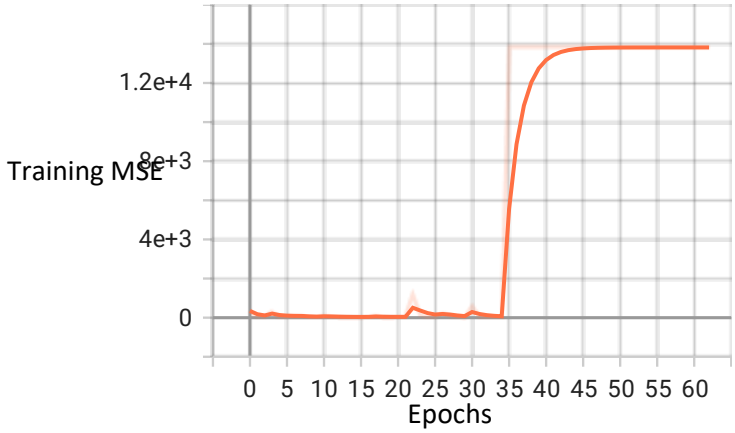

Figure 4.39 Displacement Blood Detection Training MSE

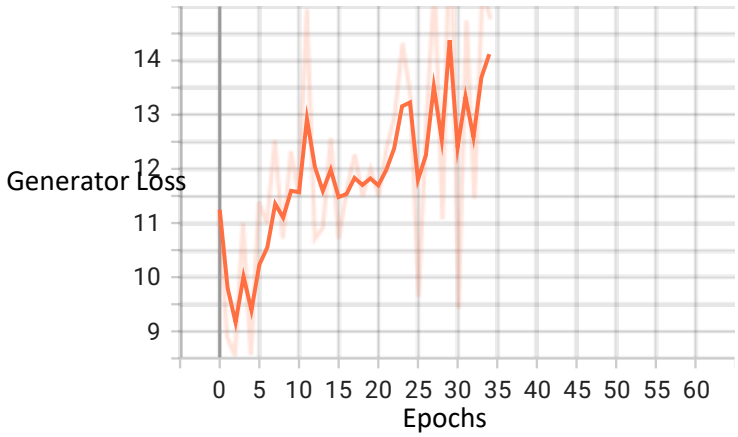

Figure 4.40 Displacement Blood Detection Training Generator Loss

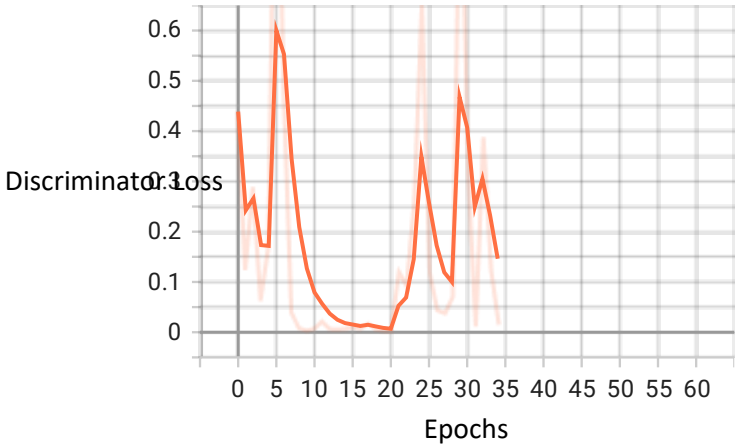

Figure 4.41 Displacement Blood Detection Training Discriminator Loss.

The following figures 4.42 through 4.44, contain bar plots corresponding to Displacement Blood detection MSE, RMSE and MS-SIMM respectively. The MSE, Figure 4.42, illustrates the generated displacement blood predictions in comparison to their label counterparts. Each generated image that was higher than approximately 50 was a generated image, the rest were poor predictions. The RMSE in Figure 4.43, shows that results had an average of 5.49 with the only blood depicting prediction being closer to 55, behavior matching to what was observed in the MSE. Lastly, Figure 4.44 contains the MS-SSIM for the displacement-based blood detection model showing an average of 0.98. However, these results may be skewed because most of the images did not produce any results that would indicate a bleed. These images suggest that this experiment was not successful and further investigation is needed.

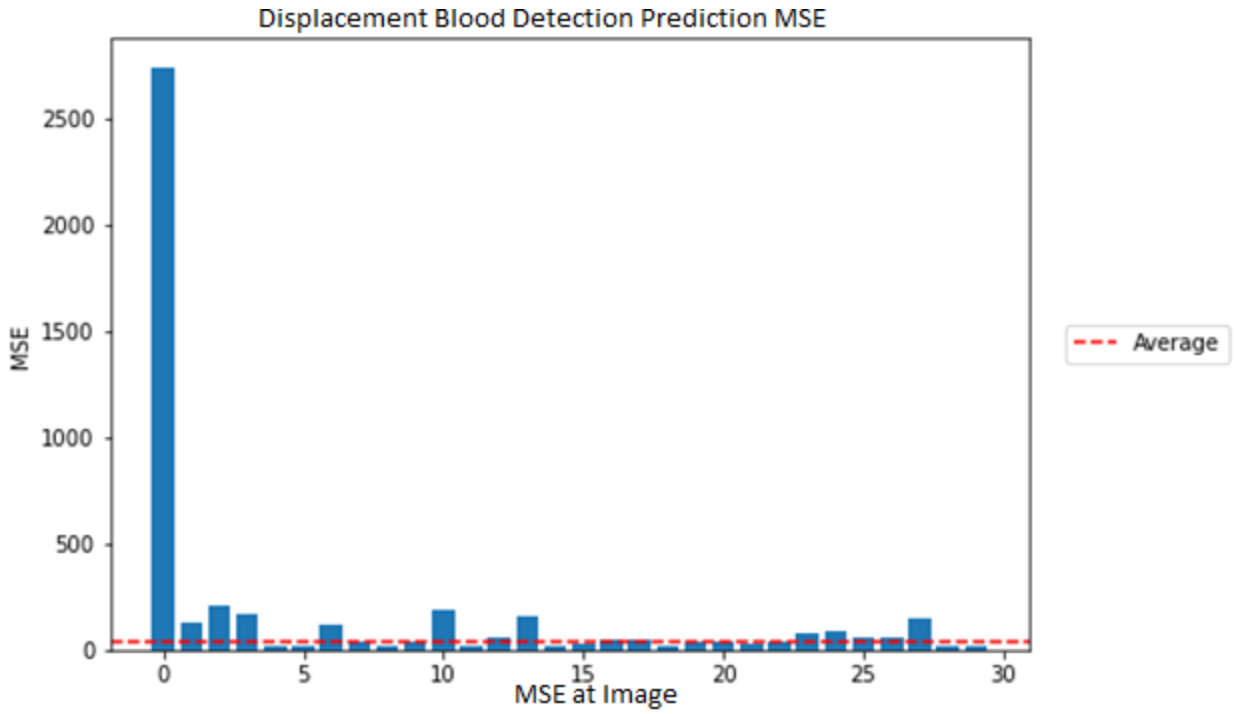

Figure 4.42 Displacement Blood Detection MSE

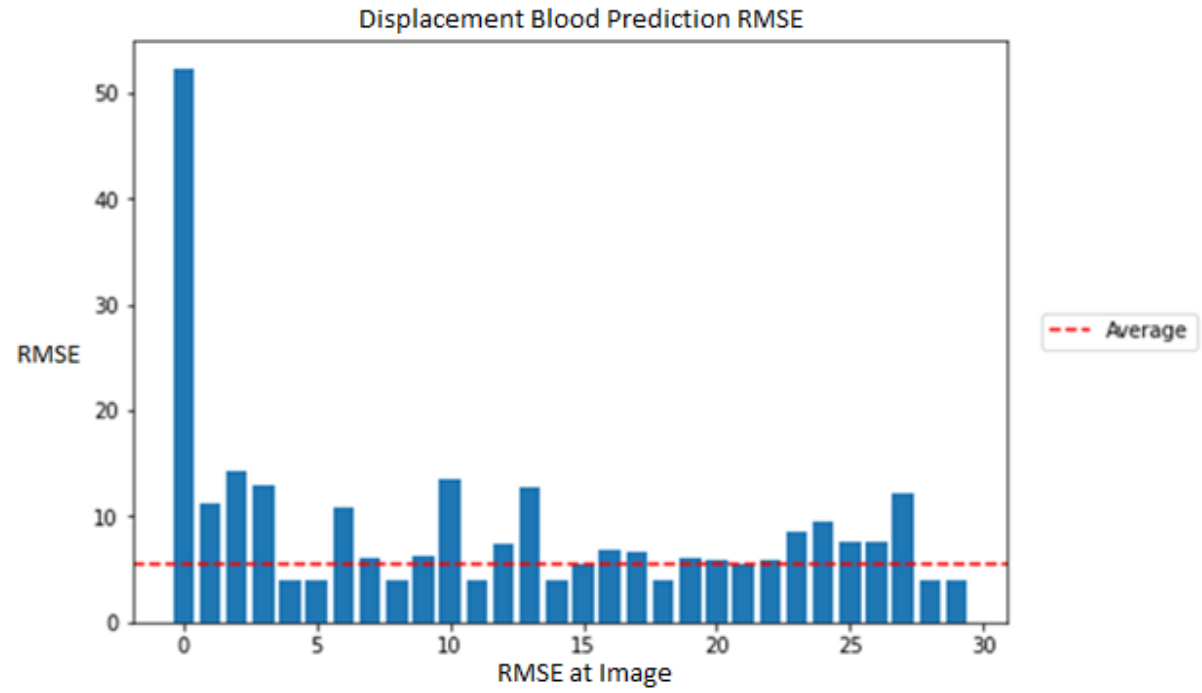

Figure 4.43 Displacement Blood Detection RMSE

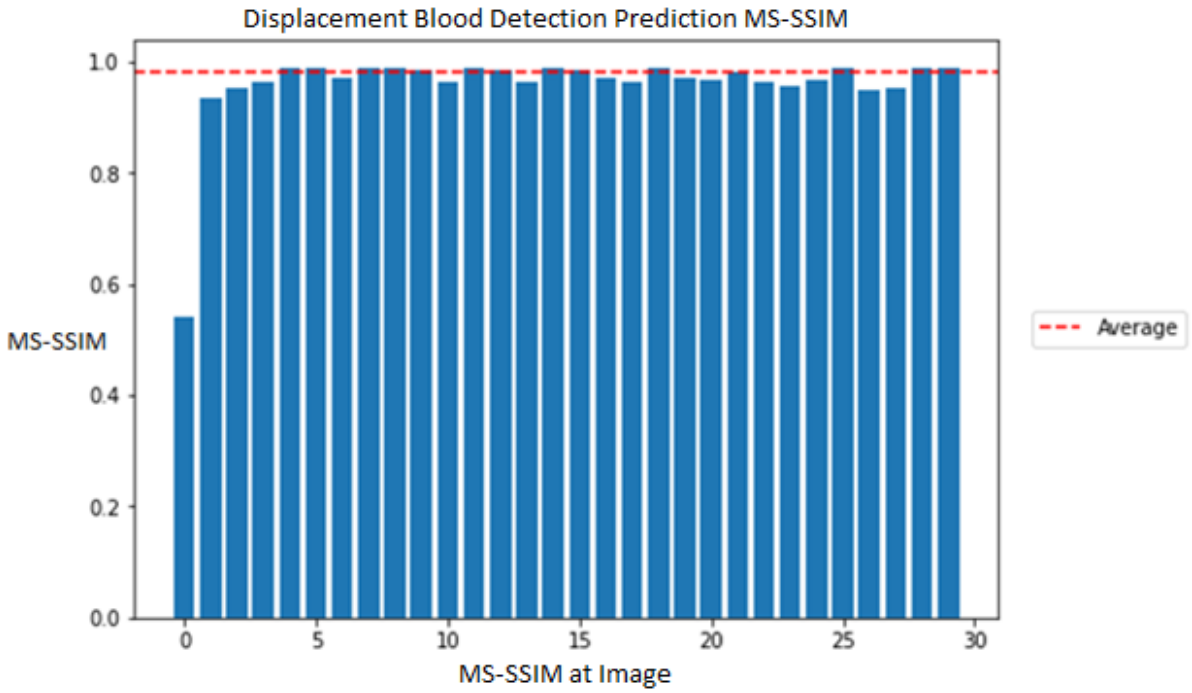

Figure 4.44 Displacement Blood Detection MS-SSIM

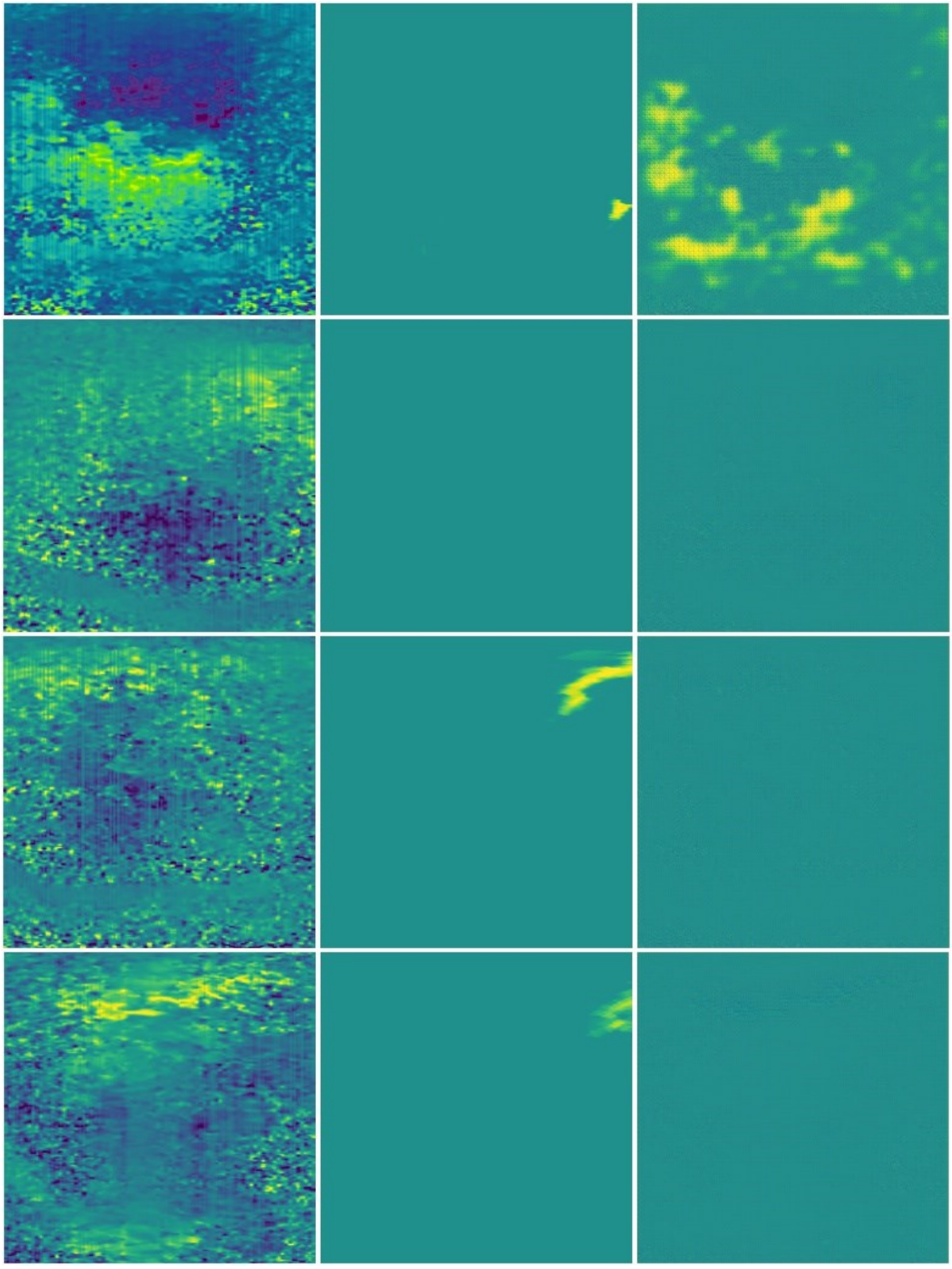

Figure 4.45: Displacement Blood Detection for input, target, prediction where rows are patients

5 DISCUSSION AND CONCLUSION

During the course of this research, we found that there were considerable obstacles when approaching machine vision problems. In this study, we were able to design and implement multiple machine learning models that allowed for specific feature detections in the human head. The conclusion section will discuss our research achievements and focus on analysis of the work done. The limitations and challenges section will focus on obstacles that had to either be overcome or outline blockers that kept us from progressing further. Furthermore, we will discuss future work that could be done to yield a more realistic outcome that is closer related to the initial goal of the research.

5.1 CONCLUSION

Next, we will focus on higher level analysis of the results to depict whether the outcome was favorable or not. Additionally, we will analyze pain points that may have detracted from our learnings.

5.1.1 *Synthetic Data Generation*

Utilizing bMode and thickBloodMask data, we were able to mimic how authentic blood data may appear, or at the very least depict an anomaly in the patient's head. Using this pseudo bleed data, we were able to produce data that could be used as a proof of concept that bleeding could be detected if it appeared in the bMode data. Using this data, we were able to train a StyleGan2-ada model which allowed us to generate synthetic data. The data that was produced was remarkably like the original dataset with an FID score of 13.11 for bMode generation and 11.79 for bModeBleed generation. While these scores are not perfect, there could be several reasons which revolve around the size of the datasets and training time. NVIDIA recommends training these models for 14 days, while the ones in this paper trained for less than 5 days each. While the synthetic data did not get utilized in this research to its full potential, it can be used in various other places. Dr. Parsons, a researcher on the team, is taking an action item to use this data with her Data Mining Class to predict where bleeds are. The synthetic data will do well to help

diversify and increase the size of their datasets allowing for different models and approaches to be taken.

5.1.2 *Skull and Ventricle Detection*

Both skull and ventricle detection were a moderate success, skull more so than ventricles. Skull detection reached a final Fréchet inception distance of 55.473, which when referencing Figure 3.29 the results are good but contain some level of noise in the images. Analyzing the images from a view perspective, one of the results stood out amongst the rest which can be seen in Figure 5.1. The figure depicts a well generated skull mask but contains what could appear as a slight blur mask. Referencing the image on the right, to the human eye, there are details contained in the generated image that are not visible in the bMode. Such a result may indicate that the model is detecting the positioning of the transducer generated image and layering a mask, albeit a good version over it. With that in mind, it leaves the question of, what would the output be if part of the skull were to be missing. It would be fascinating to gather ultrasound data with abnormalities in the dataset to train and test a new model with more variance in data.

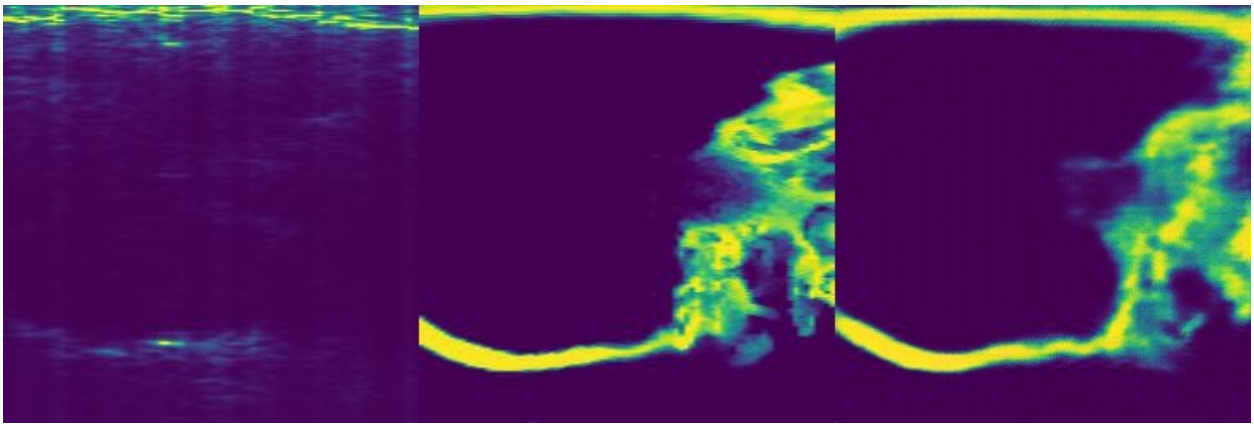

Figure 5.1: The left most image is the detected skull with some minor blurring over the middle-left structure,

Transitioning into the ventricle detection, they were slightly less impressive sitting at an FID of 77.207, however when referencing what FID scores match up with what pictures, the results are decent. They accurately predict where ventricles are and make great predictions as to the shape of the ventricle in most cases. The ventricle model even depicted a case where ventricles were not visible with decent accuracy in Figure 4.29 in the 3rd row from the top. In Figure 5.2, the rightmost

image indicates a prediction of where the ventricle is in the bMode data. The prediction appears to be quite remarkable, indicating not just the location of the ventricle, but also the shape. For the most part, the results for ventricle detection were quite good. With more data and multiple different ventricle angles, the model could be improved upon.

Figure 5.2: Displacement Ventricle Detection where the middle is CT derived and the left is predicted

5.1.3 *Blood Detection*

Blood detection was by far the most difficult feature to detect, which could be due to some factors that were beyond the control of this project. Detecting blood, we utilized 2 different data sets to predict where bleeding was occurring in patients, the two datasets being displacement paired with bloodMask and bMode paired with bModeBleed. In both cases, the results were quite poor, not giving an accurate prediction of where blood is in the brain which can be seen in Figure 5.3 and Figure 5.4. There are a few issues that can be seen in the predictions:

- Displacement Blood Predictions: The output images depict a model that cannot differentiate where bleeds are, seemingly generating random noise. The example in 5.3 shows a model that predicted that there was bleed in the input data but failed to classify any locationally relevant information.
- bModeBleed Blood Predictions: The bModeBleed model did a surprisingly good job replicating where bMode image is but failed to differentiate blood. The generated bModeBlood images overlayed blood in some cases, but not in a meaningful way.

The displacement-based model had an FID of 195.502, while the bModeBleed data had an FID score of 112.376. The low prediction accuracy could be a resultant of the data, in most cases,

ultrasounds were not taken of the patient until well after treatment. If the treatment were successful, it should have limited the amount of blood in the patient's head, which in turn would mean that blood is not contained in the ultrasound data. Coincidentally, ventricles contain fluid, something that was predictable using the same model, which leads us to believe that if blood were contained in the ultrasound data, it may be possible to predict.

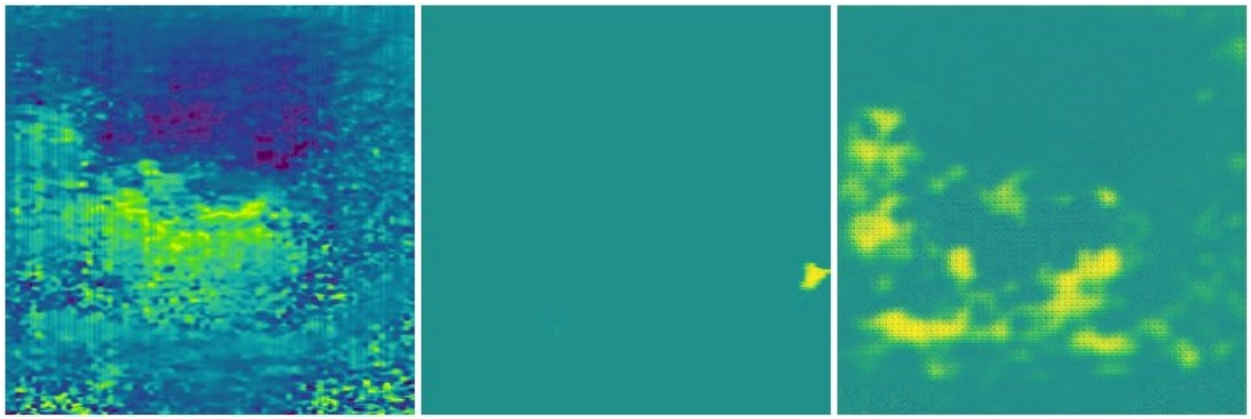

Figure 5.3 Poor Displacement Blood Predictions

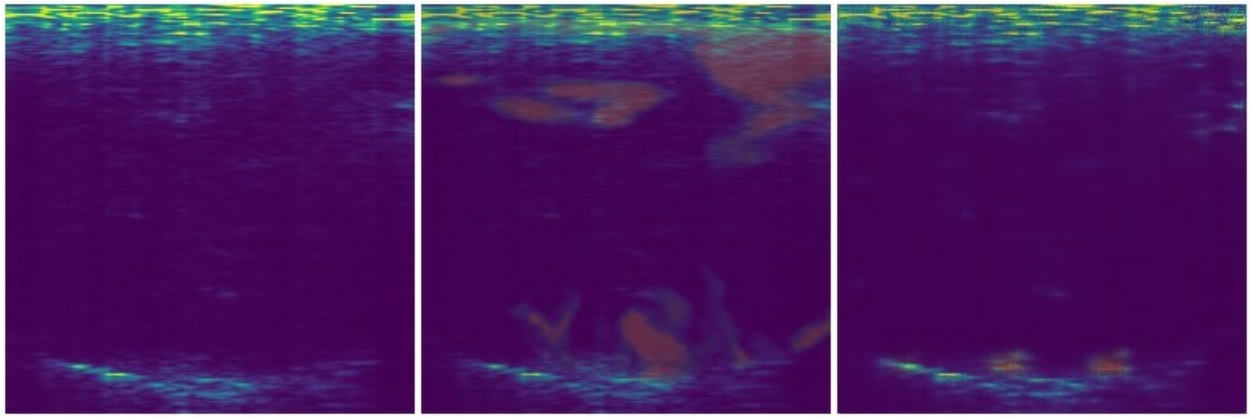

Figure 5.4 Poor bModeBlood Predictions

5.2 LIMITATIONS

5.2.1 *Data*

5.2.1.1 Collection Method

Data collection techniques had to change when COVID started in the United States, which is why around 50% patients contained less data. In addition to covid, there were several issues with the collection methodology due to the nature of the data gathered. Human life is the priority in any

medical situation, which is why in many cases the patients were treated first. Table 5.1 indicates shortcomings that revolved around data and data collection in the project.

Table 5.1: Potential Issues with Data Collection

Data Collection	**Issue**
During COVID	Due to COVID, quarantine protocols kept medical collection experts from gathering data in the same nature prior to the pandemic. Which caused an imbalance of data further shrinking the dataset for the TBI Project.
Treatment Prior to Ultrasound Data Collection	Any form of head trauma can be life threatening, which is the reason it is so important that one goes to the hospital. While not all head related traumas contain brain bleeding, a reasonable amount do. In this project, patients are administered to the hospital and put into a CT machine. The CT machine dictates a surgical procedure if there is bleeding. To increase the chances that a patient survives, the treatment occurs immediately, Pulsatility does not get a chance to collect data prior to treatment and must wait for family or patient consent. While the data is collected with the idea that it contains bleeds, that may not be reflected in the data.

5.2.1.2 Sample Size

Pulsatility collected 130 patients worth of patient data which created around two thousand independent samples of data of each type. While this may seem like a lot of data, it is not uncommon for machine learning models to train on hundreds of thousands to millions of data samples, something that we did not have access to. The largest bottleneck in this project is the lack of data, and the team's challenges to gather more. Collecting data is a challenging task, collecting medical data is more so, but collecting medical data that is so specific (which is needed in this project) almost seems impossible. To uniquely qualify for data collection, the patient must be injured in a battle situation and be consenting of data collection. The researchers settled on allowing non-combat related injuries into the pool of candidates, and after 4 years of data collection has turned up 130 patients. However, it is worth noting that skull and ventricle detection could loosen their restrictions on candidates to allow for a larger pool, however with blood, no further

changes can be made. The ideal sample size would be two thousand or more candidates before we can start making more accurate predictions.

5.2.1.3 Complexity

Business spending millions of dollars perfecting their onboarding strategies to help mitigate down time and allow employees to ramp up a fast pace. The Pulsatility project was incredibly complex and took a significant amount of time and energy to onboard to. It took approximately 5 months of reading, researching, attending meetings, and thinking, before modifying or diving into the data. Documentation was sparse and was primarily contained as tribal knowledge, meaning that there was no consolidated location of information and instead required one to ask questions. Thankfully, this changed as time went on and more documentation was produced that would allow for future students or researchers to onboard more quickly.

5.2.2 *Hardware*

The largest consumer of time in this project was training. The models took a significant amount of time to train, ranging from 3 hours to 7 days in the bModeBlood synthetic data generation model. Initially, to mitigate this time, we had planned to use cloud providers. However, after researching the cost per hour of GPU clusters, the time savings didn't make as much sense. The amount of time required to have an instance up and to train would likely cost as much or more than buying a better GPU. While a cloud-based provider doesn't make sense for an individual, we believe that an organization would be better suited for such instances. As an individual, it is hard to justify the cost of an instance rather than buying hardware for yourself and reselling it if desired.

While it sounds easy to buy hardware for a project, there is a considerable amount of work that goes into determining what you need. In this research the size of GPU caches wasn't of huge concern, however, the frequency of the ram and computational capabilities are. While training our detection models, a significant amount of time was spent transferring data to and from the GPU due to the way metrics were calculated. The FID required the test set images to be predicted and the predicted images computed against the labels. Which in turn meant that we were loading and offloading the models several times after each epoch. Knowing this meant that we needed a GPU that had enough onboard memory to store the model and train, but also have a high enough frequency that the emission of metrics does not take a large amount of time.

When using specific consumer-grade equipment training time still took a considerable amount of time. In total, training took 2 or more months when you factored in different model versions that were not talked about in this paper. However, with proper equipment, or more instances, it would be entirely possible to lower the training time to a few days. Instead of training 1 or at most 2 models at a time, we could train all of them and iterate from there.

5.3 Future Work

The thesis has gone in depth on methods to generate ultrasound data. Additionally, the work presents methodologies and approaches to detect skull, ventricles, and blood contained in ultrasound images of the human head. The research has proven that there is success in finding both skull and ventricles in ultrasound data but could easily be scaled up with a larger dataset. However, the data used in this project did not provide a sufficient basis to detect bleeding in ultrasound images of the human head. The work done can provide a starting point for future research to iterate over as more data and resources become accessible. The following list contains potential areas of research that could be conducted:

- **Blood Detection:** While blood detection was not successful in this project it may still be possible to detect bleeding. The models used in this research could be applied to the same situations given more data that is better representative of brain bleeds that are caused by a war situation.
- **Ventricle Detection:** Ventricles are quite possibly the most interesting feature that were detectable in this project. While the initial goal of Pulsatility was not to detect ventricles, we believe that ultrasound has shown significant promise in detecting them. Given a larger pool of data (2000+) patients, the shortcomings of the current model may be surpassed. That is, with more data, it may be possible to better detect the shape of a patient's ventricles.
- **Skull Detection:** Building from the previous statement, if there were more data, it could be very possible to create a better performing model. It would be interesting to add in cases where skull is not there and or there are abnormalities. While the skull detection model did well, there could be issues when cases appear that the mode has not seen before. Adding in data with missing skull / abnormal features could create a more robust model.

- **Synthetically Generated Data:** The Stylegan2-ada model showed promise in creating accurately generated data, which could be used to detect bleeding that was artificially added. Dr. Parson's data recognition class is taking an action to create a model that will use this data to determine if bMode contains or does not contain bleeding.

Moreover, the use of computer vision-based approaches to both generate and detect features of the human head using ultrasound data is plausible. However, there were many shortfalls that kept this thesis from performing to its fullest abilities, which can be attributed to the quantity and quality of data. With sufficient data, each model could be improved upon to increase the quality of the predictions and come closer to the original CT derived target data.

BIBLIOGRAPHY

[1] J. E. Risdall and D. K. Menon, “Traumatic brain injury,” *Philos Trans R Soc Lond B Biol Sci*, vol. 366, no. 1562, pp. 241–250, Jan. 2011, doi: 10.1098/rstb.2010.0230.

[2] J. J. Bazarian *et al.*, “Serum GFAP and UCH-L1 for prediction of absence of intracranial injuries on head CT (ALERT-TBI): a multicentre observational study,” *The Lancet Neurology*, vol. 17, no. 9, pp. 782–789, Sep. 2018, doi: 10.1016/S1474-4422(18)30231-X.

[3] D. E. Ross, “Review of longitudinal studies of MRI brain volumetry in patients with traumatic brain injury,” *Brain Injury*, vol. 25, no. 13–14, pp. 1271–1278, Dec. 2011, doi: 10.3109/02699052.2011.624568.

[4] T. Heye *et al.*, “The Energy Consumption of Radiology: Energy- and Cost-saving Opportunities for CT and MRI Operation,” *Radiology*, vol. 295, no. 3, pp. 593–605, Mar. 2020, doi: 10.1148/radiol.2020192084.

[5] S. Jain and L. M. Iverson, *Glasgow Coma Scale*. StatPearls Publishing, Treasure Island (FL), 2021. [Online]. Available: http://europepmc.org/abstract/MED/30020670

[6] S. Li, J. Jiao, Y. Han, and T. Weissman, “Demystifying ResNet,” *CoRR*, vol. abs/1611.01186, 2016, [Online]. Available: http://arxiv.org/abs/1611.01186

[7] Terason Division and Teratech Corporation, “Smart 3200T NexGen,” 2021. https://www.terason.com/usmart-3200t/ (accessed Nov. 06, 2021).

[8] L. A. Abbagnaro, B. B. Bauer, and E. L. Torick, “Measurements of diffraction and interaural delay of a progressive sound wave caused by the human head. II,” *J Acoust Soc Am*, vol. 58, no. 3, pp. 693–700, Sep. 1975, doi: 10.1121/1.380716.

[9] J. Kucewicz, “TBI Database Resources”.

[10] J. Kucewicz, “Kucewicz.”

[11] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[12] K. Han, Y. Li, and B. Xia, “A cascade model-aware generative adversarial example detection method,” *Tsinghua Science and Technology*, vol. 26, no. 6, pp. 800–812, 2021.

[13] “A detailed explanation of the Attention U-Net.”

[14] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, “UNet++: A Nested U-Net Architecture for Medical Image Segmentation,” *CoRR*, vol. abs/1807.10165, 2018, [Online]. Available: http://arxiv.org/abs/1807.10165

[15] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” *CoRR*, vol. abs/1505.04597, 2015, [Online]. Available: http://arxiv.org/abs/1505.04597

[16] L. Baskaran *et al.*, “Automatic segmentation of multiple cardiovascular structures from cardiac computed tomography angiography images using deep learning,” *PLOS ONE*, vol. 15, p. e0232573, Jun. 2020, doi: 10.1371/journal.pone.0232573.

[17] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.
[18] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," *CVPR*, 2017.
[19] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training Generative Adversarial Networks with Limited Data," 2020.
[20] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.
[21] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.
[22] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
[23] S. K. Zhou, "Shape regression machine and efficient segmentation of left ventricle endocardium from 2D B-mode echocardiogram," *Medical Image Analysis*, vol. 14, no. 4, pp. 563–581, Aug. 2010, doi: 10.1016/J.MEDIA.2010.04.002.
[24] B. Vuleta, " How Much Data Is Created Every Day? 27 Staggering Stats." 2021. [Online]. Available: https://seedscientific.com/how-much-data-is-created-every-day/#:~:text=The%20amount%20of%20data%20in,least%201%2C200%20petabytes%20of%20information.
[25] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic data augmentation using GAN for improved liver lesion classification," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 289–293. doi: 10.1109/ISBI.2018.8363576.
[26] M. Fintz, A. Shoshan, N. Bhonker, I. Kviatkovsky, and G. G. Medioni, "Synthetic Data for Model Selection," *CoRR*, vol. abs/2105.00717, 2021, [Online]. Available: https://arxiv.org/abs/2105.00717
[27] D. Perri, M. Simonetti, and O. Gervasi, "Synthetic Data Generation to Speed-Up the Object Recognition Pipeline," *Electronics (Basel)*, vol. 11, no. 1, 2022, doi: 10.3390/electronics11010002.
[28] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training Generative Adversarial Networks with Limited Data," 2020.
[29] E. J. Nunn, P. Khadivi, and S. Samavi, "Compound Frechet Inception Distance for Quality Assessment of GAN Created Images." arXiv, 2021. doi: 10.48550/ARXIV.2106.08575.
[30] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium," *CoRR*, vol. abs/1706.08500, 2017, [Online]. Available: http://arxiv.org/abs/1706.08500
[31] Y. Yu, W. Zhang, and Y. Deng, "Frechet Inception Distance (FID) for Evaluating GANs".
[32] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs." arXiv, 2018. doi: 10.48550/ARXIV.1801.01401.
[33] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila, "Improved Precision and Recall Metric for Assessing Generative Models." arXiv, 2019. doi: 10.48550/ARXIV.1904.06991.
[34] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, 2003, vol. 2, pp. 1398-1402 Vol.2. doi: 10.1109/ACSSC.2003.1292216.

[35] “Lambda Labs.”
[36] Martín~Abadi *et al.*, “ TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.” 2015. [Online]. Available: https://www.tensorflow.org/
[37] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[38] “Anaconda Software Distribution,” *Anaconda Documentation*. Anaconda Inc., 2020. [Online]. Available: https://docs.anaconda.com/
[39] W. Li, L. Fan, Z. Wang, C. Ma, and X. Cui, “Tackling mode collapse in multi-generator GANs with orthogonal vectors,” *Pattern Recognition*, vol. 110, p. 107646, Feb. 2021, doi: 10.1016/J.PATCOG.2020.107646.
[40] “Brain Shift and Herniation.”

APPENDIX A:

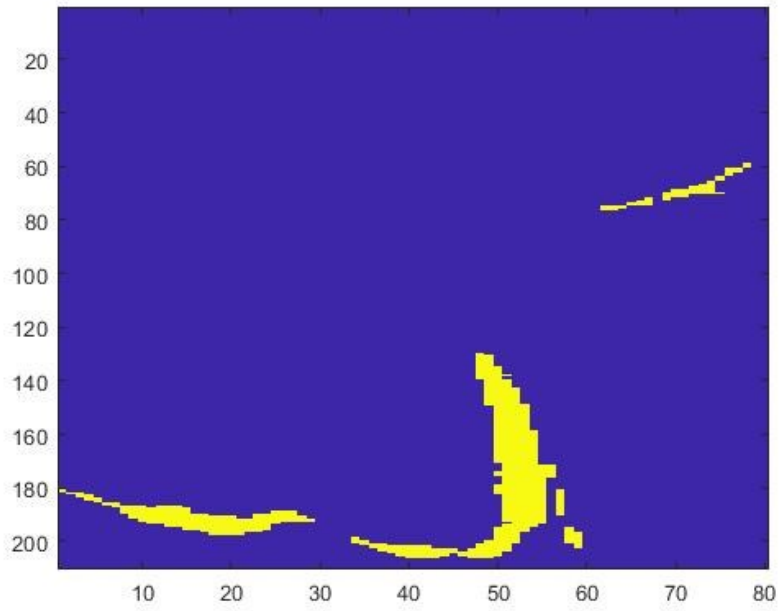

Figure A.1: CT Derived Blood Mask Variable

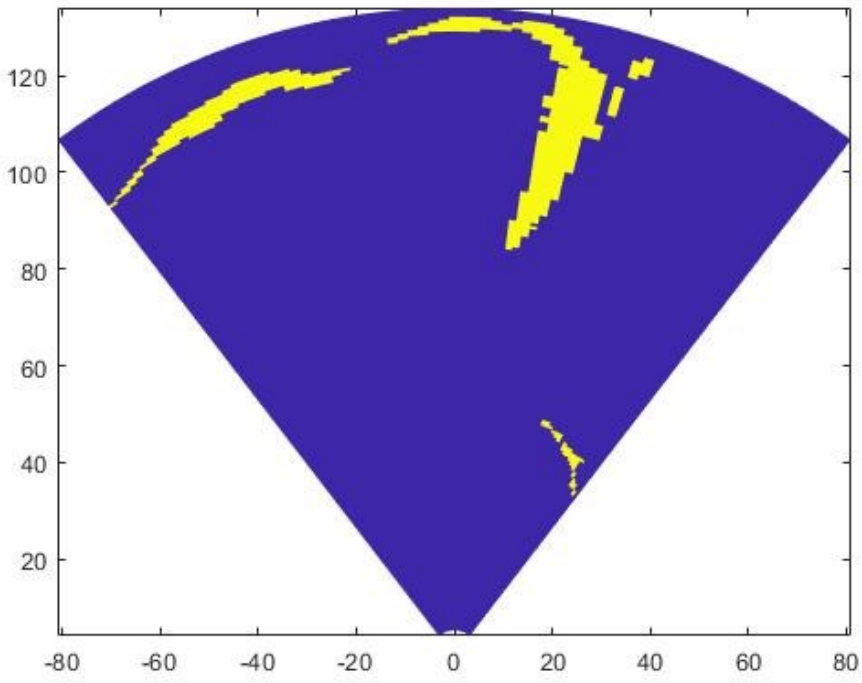

Figure 5.5: Blood Mask in the Form of Ultrasound View

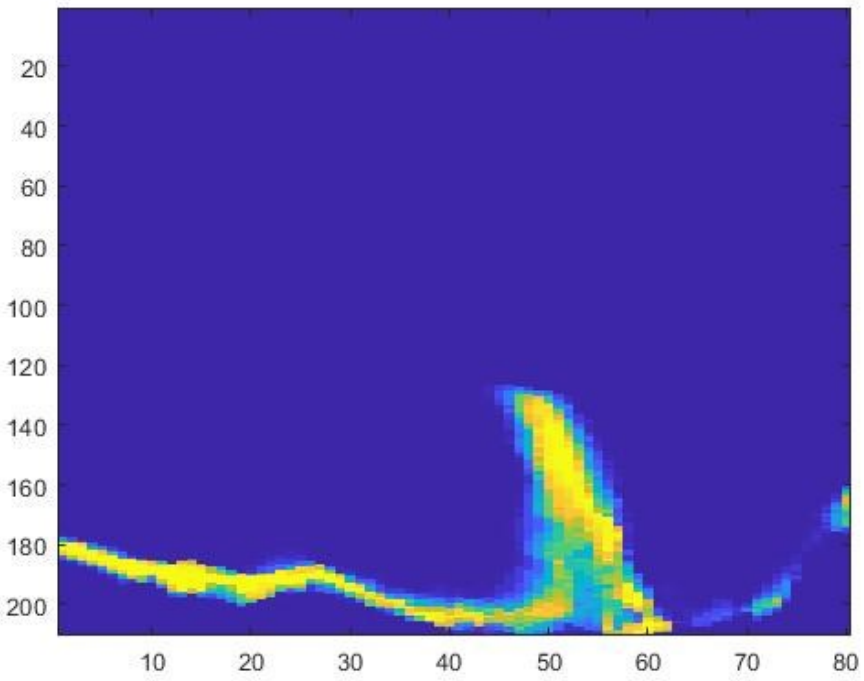

Figure 5.6 bloodMaskThick CT Data

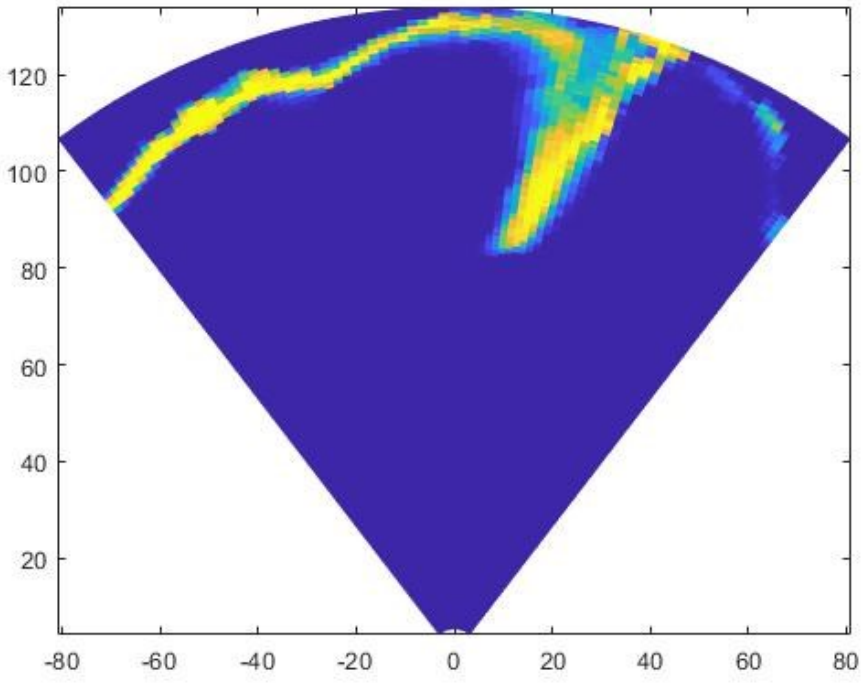

Figure 5.7: bloodMaskThick in the Form of Ultrasound View

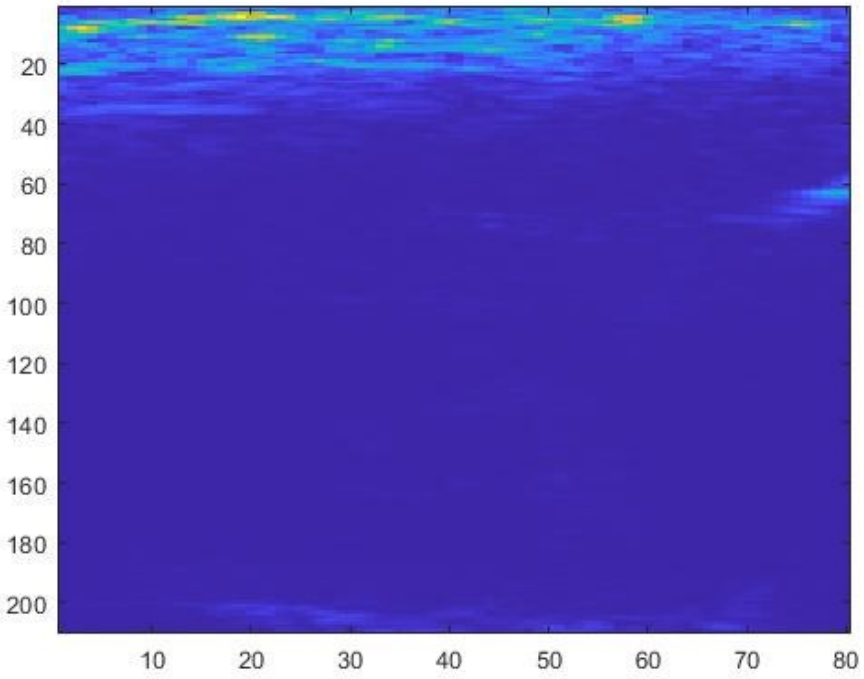

Figure 5.8: Ultrasound Collected bMode data

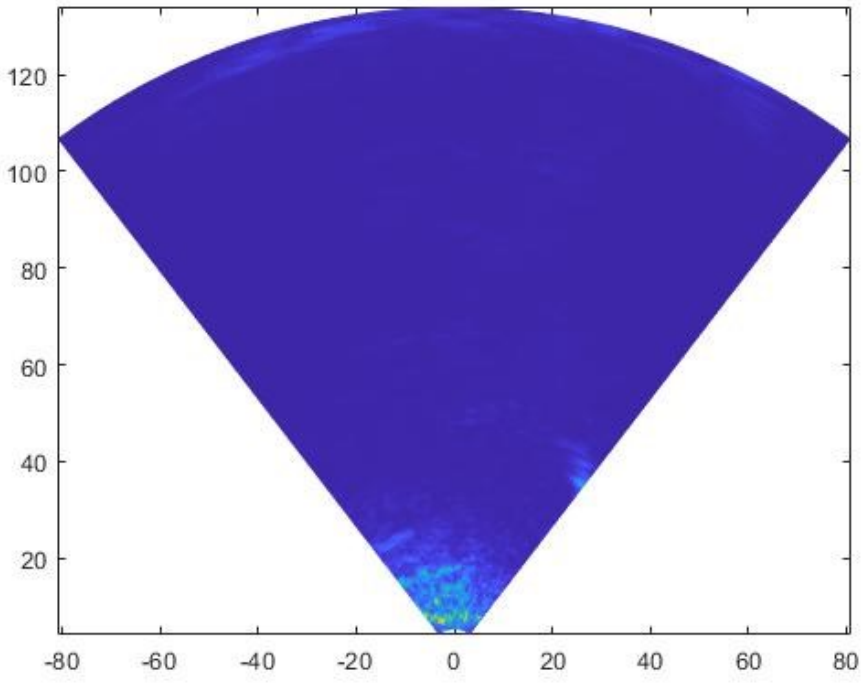

Figure 5.9: Ultrasound bMode Data as it would appear from the technician

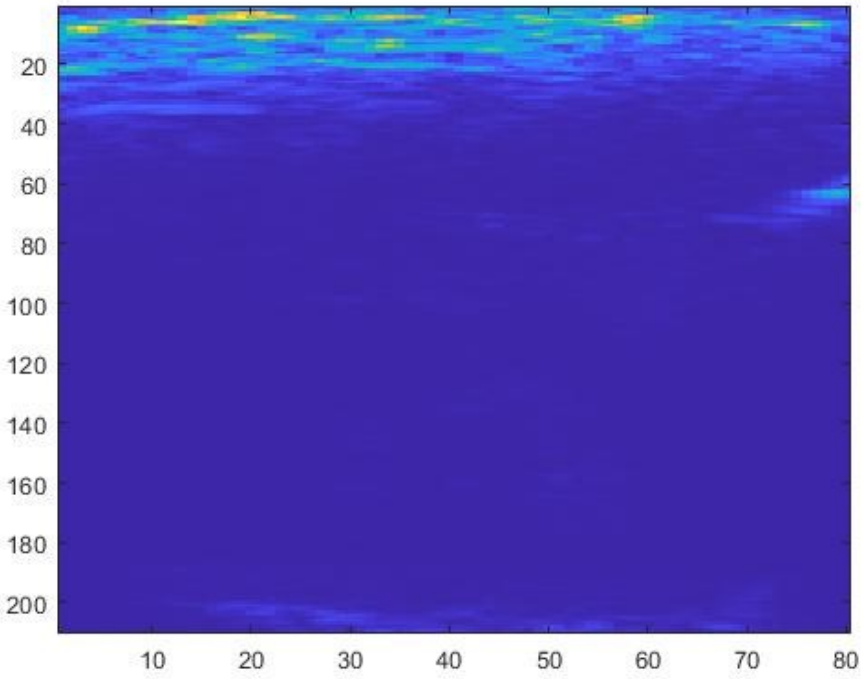

Figure 5.10: Ultrasound collected data that has been normalized

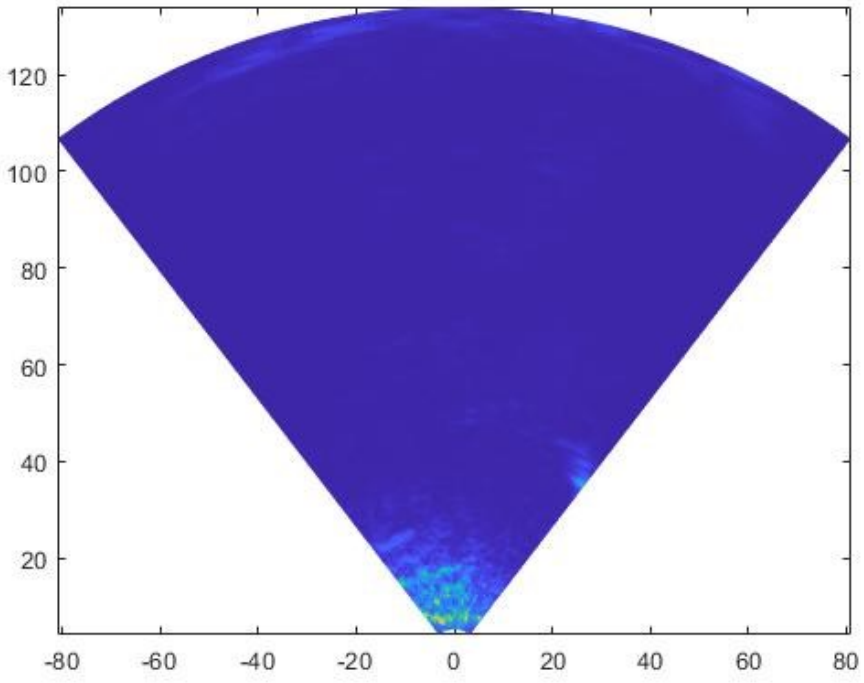

Figure 5.11: Ultrasound collected data that is viewed as it may be seen by the technician

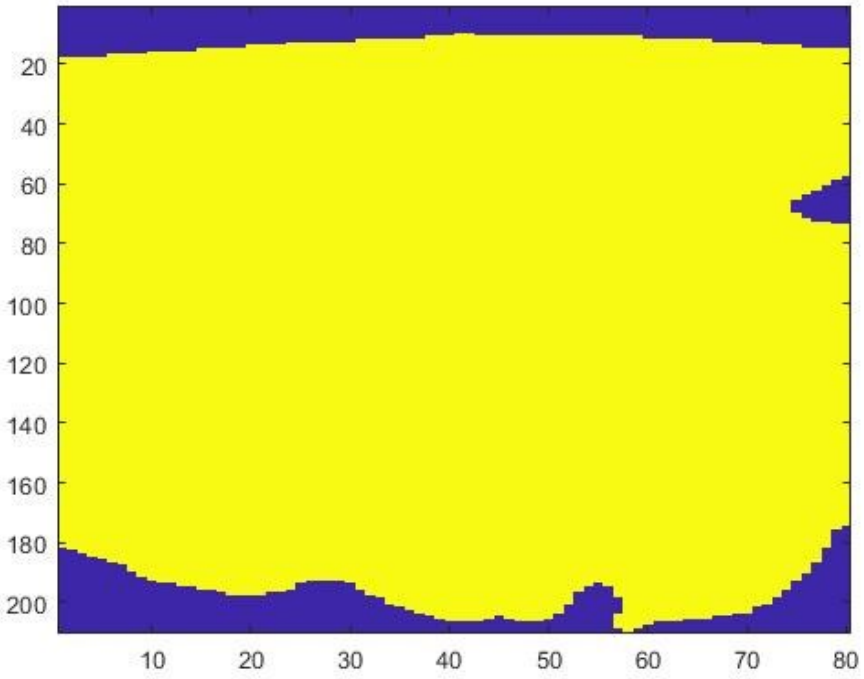

Figure 5.12: CT collected BrainMask data

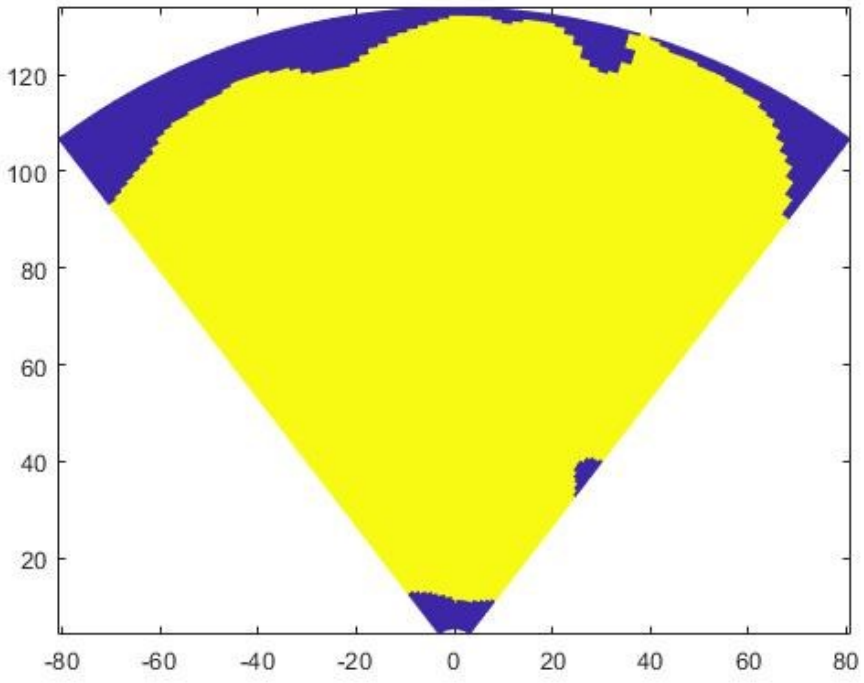

Figure 5.13: BrainMask data as it would appear from ultrasound

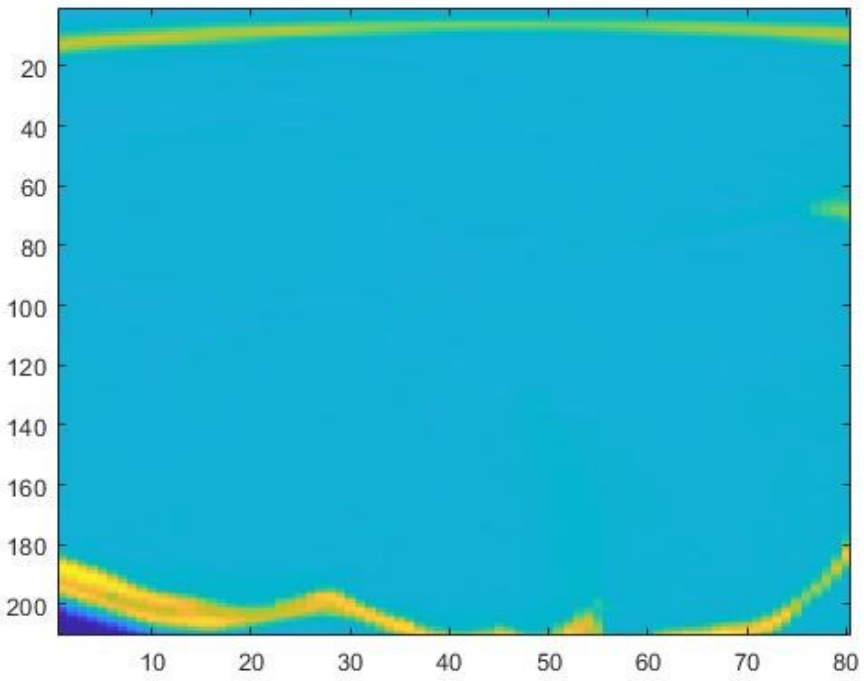

Figure 5.14: Full CT data of a patient's head

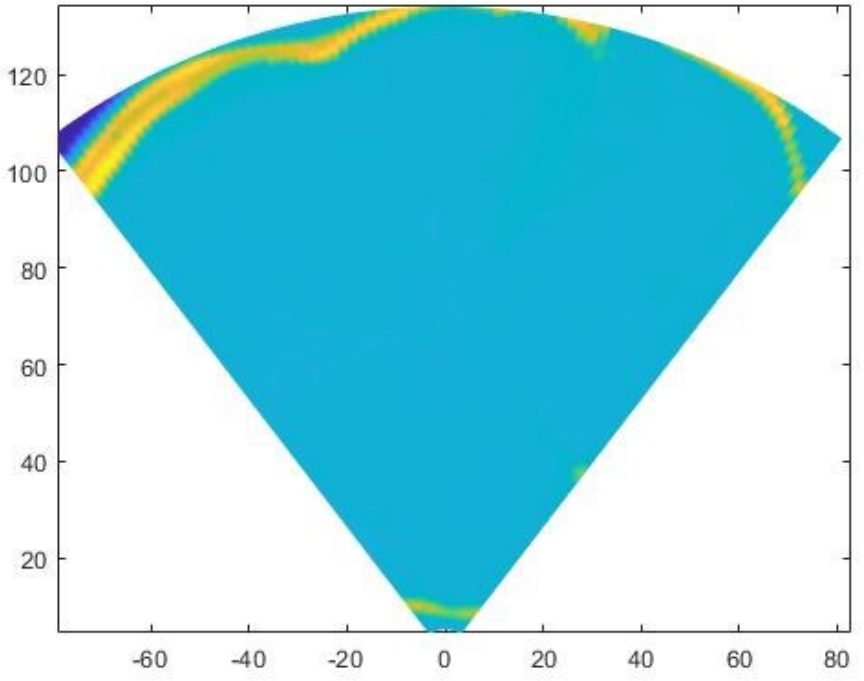

Figure 5.15: Full CT Data as seen through ultrasound view

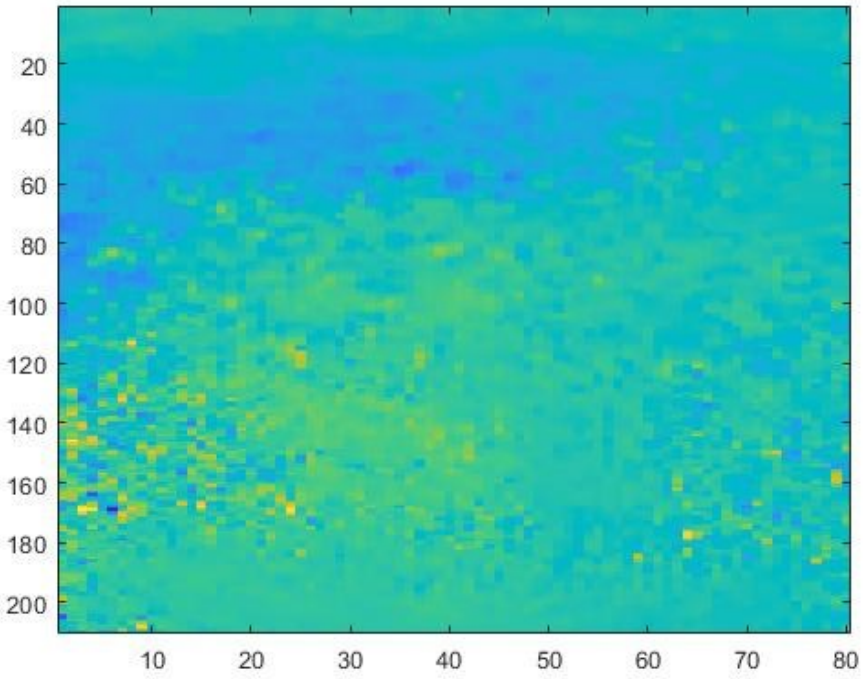

Figure 5.16: Displacement Data Collected by the Ultrasound Device

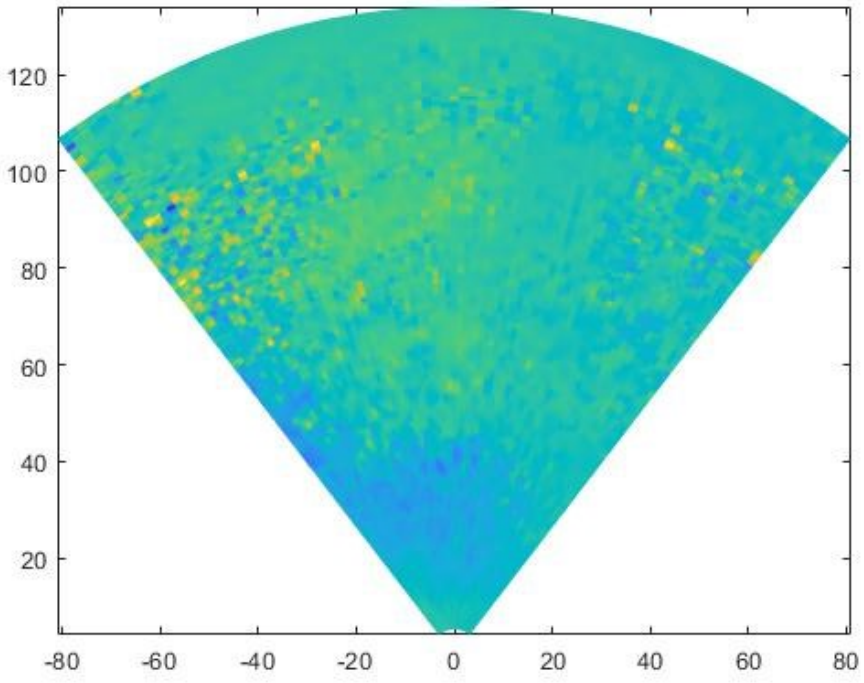

Figure 5.17: Displacement data as would be viewed from the ultrasound device

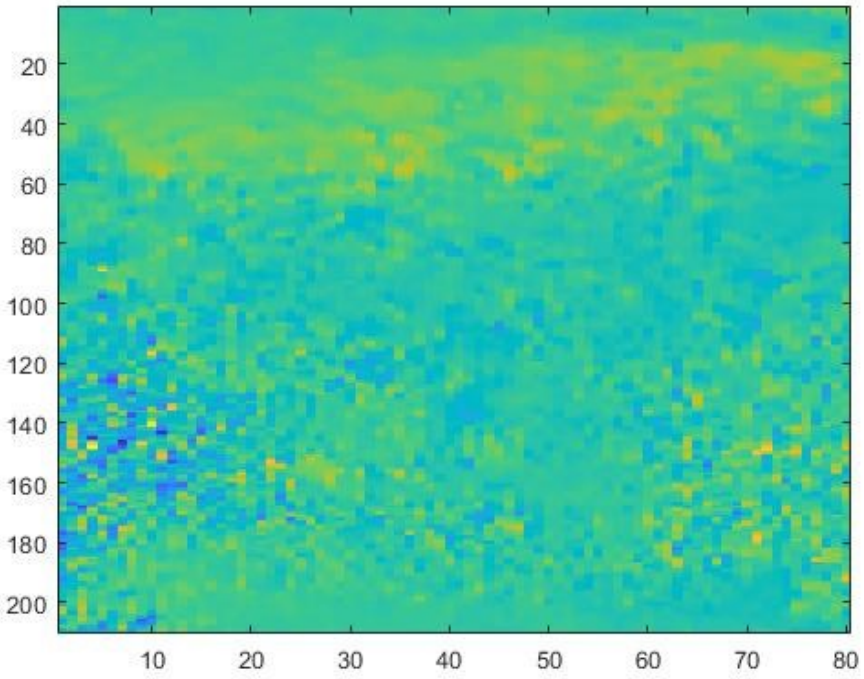

Figure 5.18: Normalized Displacement Data

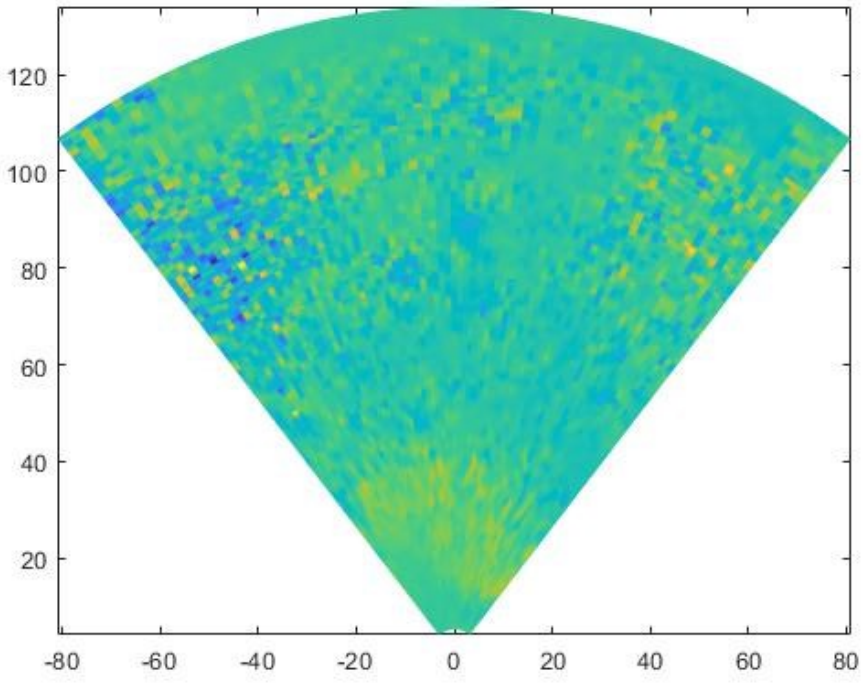

Figure 5.19: Normalize Displacement in Form of Ultrasound View

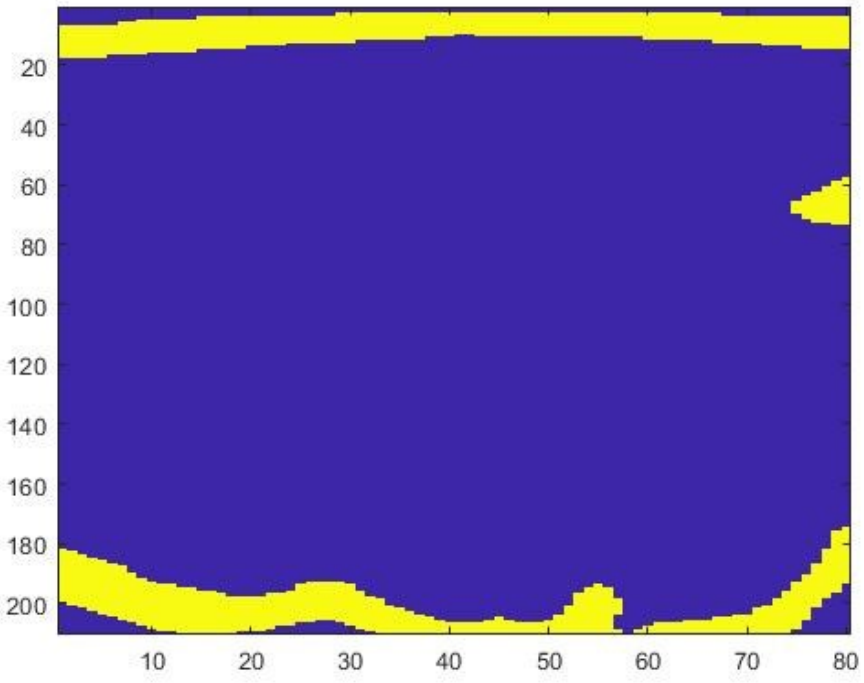

Figure 5.20: CT Derived Skull Mask

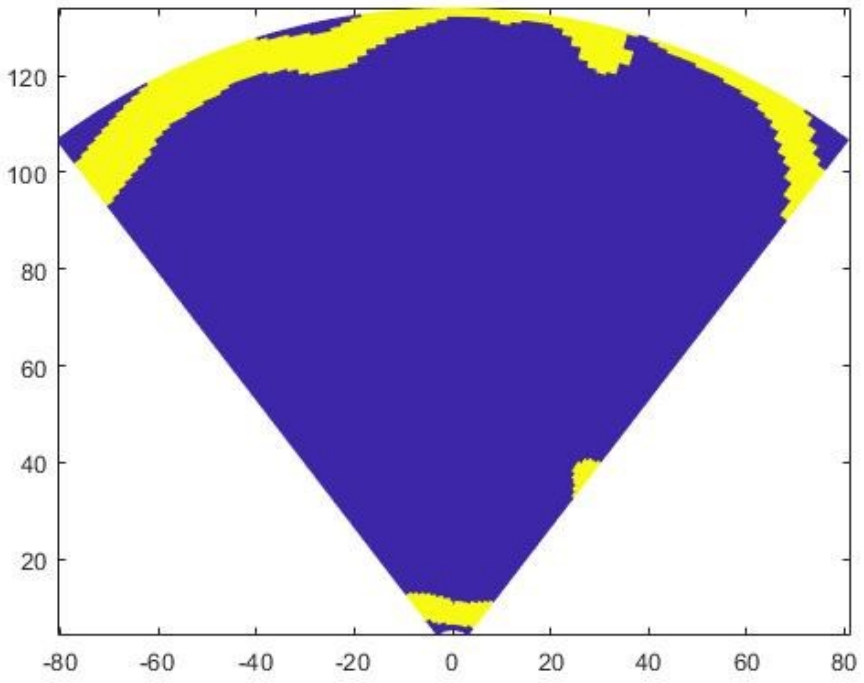

Figure 5.21: Ultrasound Point of View for Skull Mask

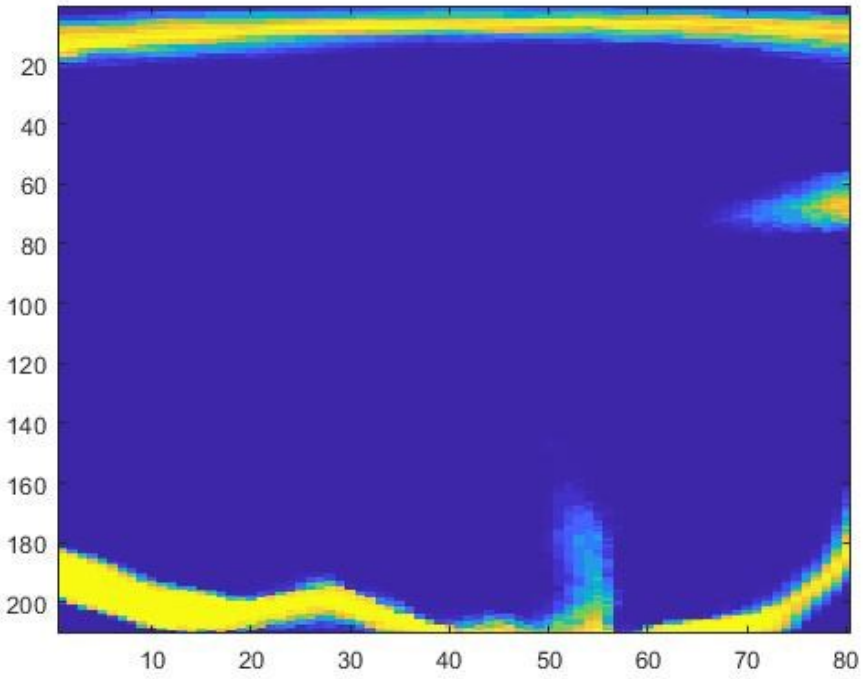

Figure 5.22: Skull Mask Thick data

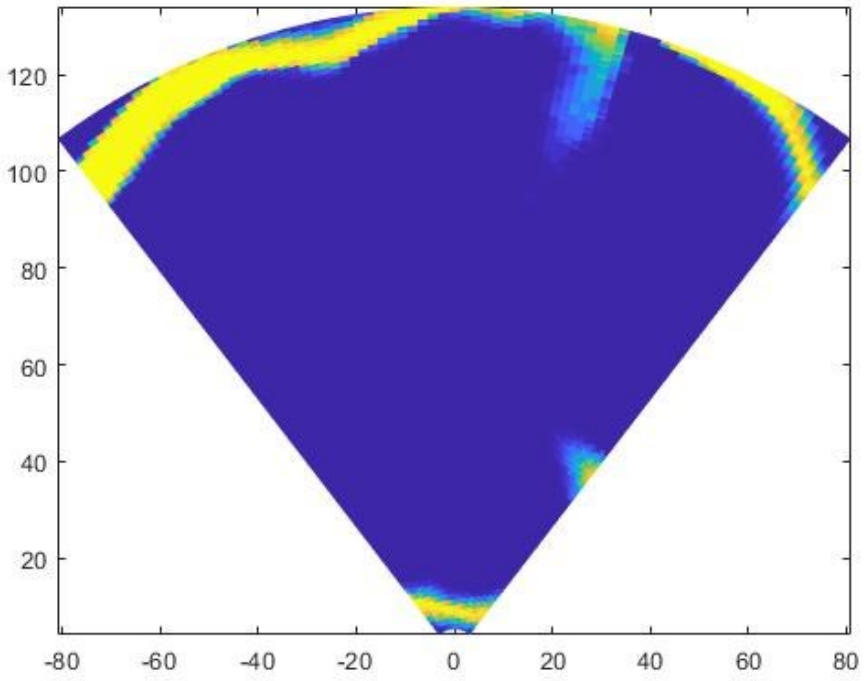

Figure 5.23: Skull Mask Thick viewed from Ultrasound

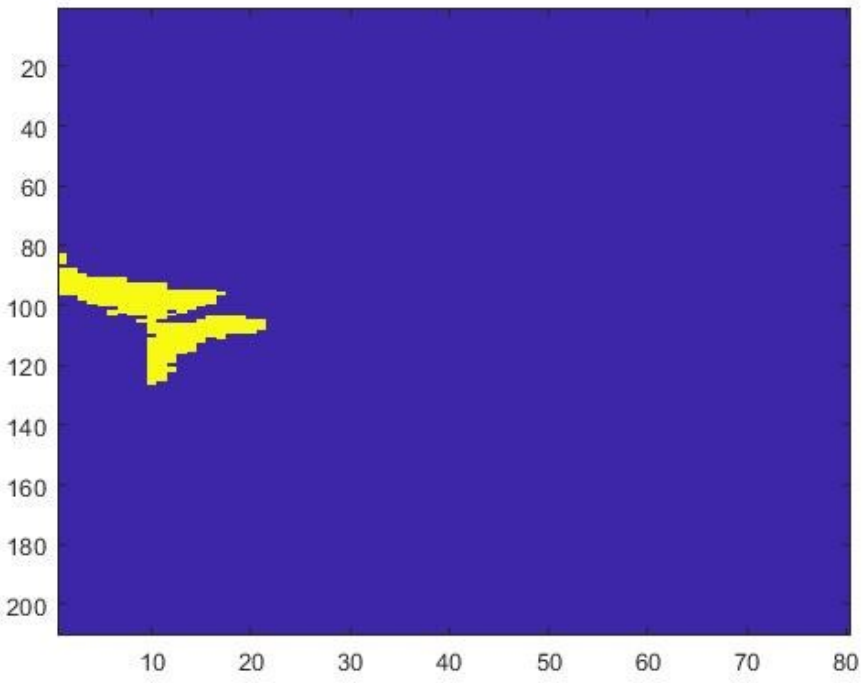

Figure 5.24: CT derived Ventricle Mask

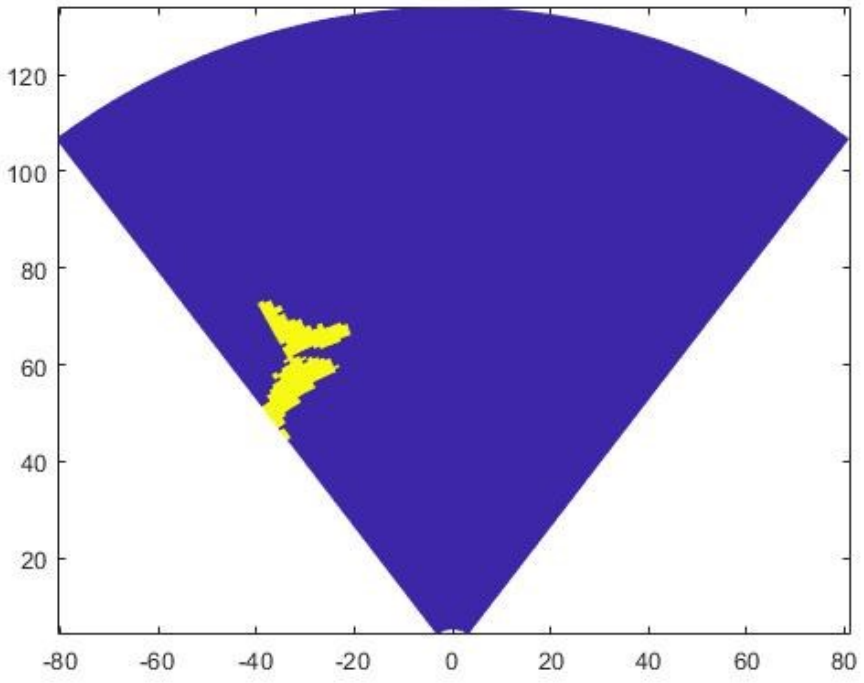

Figure 5.25: Ventricle Mask depicted from Ultrasound view

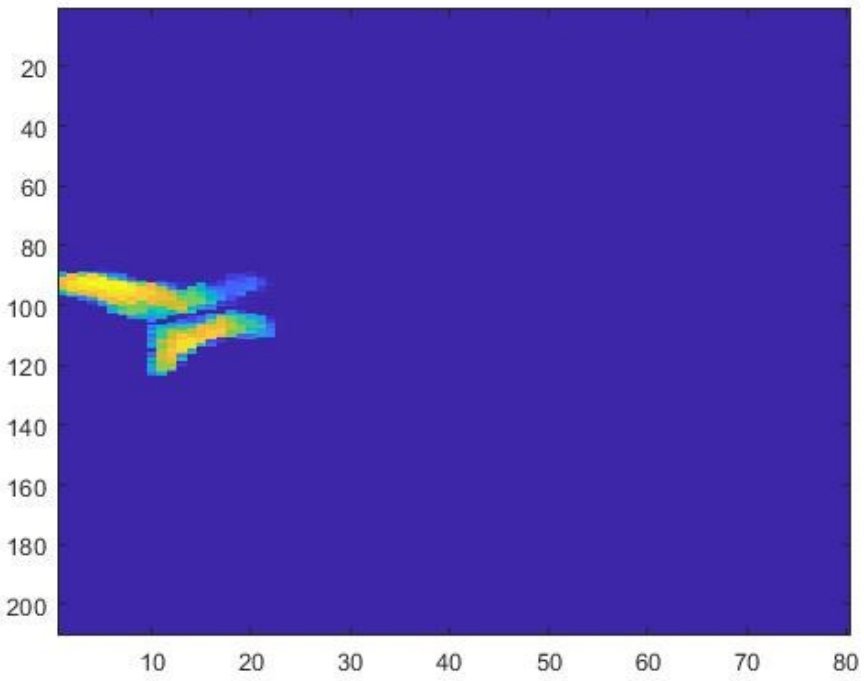

Figure 5.26: ThickVentricleMask derived from CT data

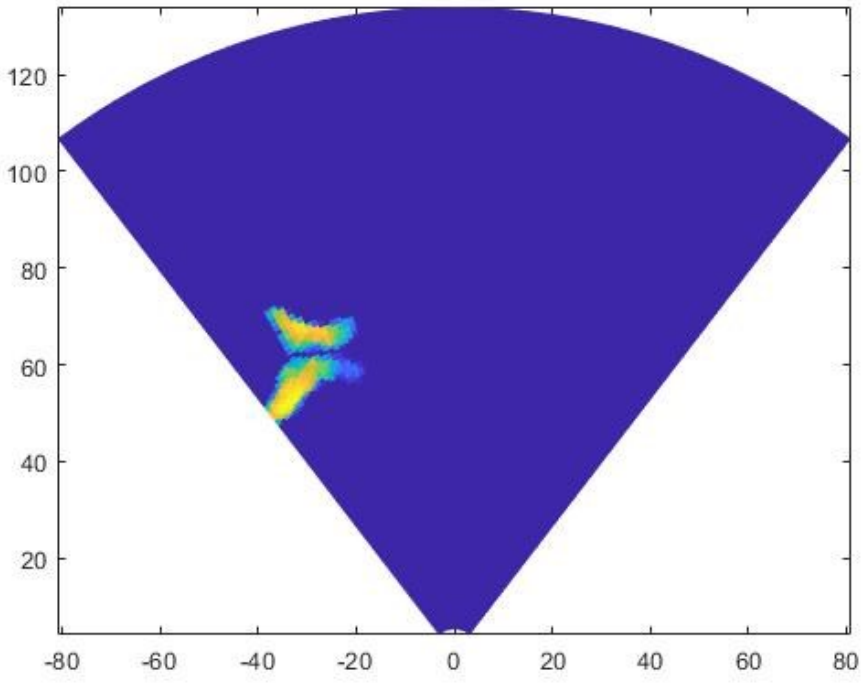

Figure 5.27: ThickVentricleMask simulated by ultrasound view