Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Erlend Løland Gundersen

Patient Adaptive Imaging in Echocardiography

Unfiltered Beamformed IQ data to Despeckled GE Vivid E95 Processed Images with Deep Learning

Master's thesis in Electronics System Design and Innovation
Supervisor: Svein Erik Måsøy
Co-supervisor: Erik Smistad
June 2022

NTNU
Norwegian University of Science and Technology

Erlend Løland Gundersen

Patient Adaptive Imaging in Echocardiography

Unfiltered Beamformed IQ data to Despeckled GE Vivid E95 Processed Images with Deep Learning

Master's thesis in Electronics System Design and Innovation
Supervisor: Svein Erik Måsøy
Co-supervisor: Erik Smistad
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis proposes a deep learning-based method for generalization of the native B-mode signal processing of the GE Vivid E95 ultrasound scanner, for the purpose of improving image quality in echocardiography. A second method was developed for speckle noise reduction. Image enhancements may facilitate early-stage cancer tumor detection and better diagnosis of some diseases.

Ultrasound image quality is dependent on signal processing chains with manually tuned parameters. A data-driven solution may improve band-pass filtering, time frequency compensation, image filtering, and other signal processing operations by making them patient adaptive. Despeckling may facilitate better visual perception in the assessment of ultrasound images.

Using a supervised learning algorithm, models were trained to map unfiltered beamformed in-phase and quadrature component (IQ) data to E95 processed images. The dataset consisted of $7,424$ real cardiac images and corresponding IQ data from 22 patients, acquired with a 4Vc-D 4D probe. Despeckling was achieved by mapping IQ data from one elevation plane to compounded B-mode images. This way, the speckle reduction effect of elevation compounding could be obtained when using an M5Sc-D probe without this functionality.

Pre-processing was critical for achieving good results. The data was upsampled with bilinear interpolation, scaled with global maximum absolute scaling, and the IQ data was represented in Cartesian form.

Hundreds of convolutional neural networks were tested. Using the Adam optimizer, the models were trained with gradient descent of loss functions based on the structural similarity index measure (SSIM) and the peak signal-to-noise ratio (PSNR). Qualitative evaluation was performed by examining scan converted images, difference plots, and histograms. Performance was measured quantitatively by the SSIM and PSNR metrics.

The results showed that a small Inception U-Net can replicate the Vivid E95 native signal processing. Evaluation of a 10% test dataset gave an SSIM value of $(98.8 \pm 0.50)\%$ and a PSNR value of (35.8 ± 1.38) dB. Visual inspections showed that the predicted images were nearly identical to the E95 images. Testing the replication model on M5Sc-D probe data showed an increased contrast resolution but altered speckles. The despeckling model produced smeared images with reduced speckle noise at the cost of lowering the contrast resolution.

Sammendrag

Denne oppgaven foreslår en dyp læring-basert metode for generalisering av B-modus-signalbehandlingen til Vivid E95 ultralydskanneren fra GE, med det formål å forbedre bildekvaliteten i ekkokardiografi. En annen metode ble utviklet for flekkstøyreduksjon. Bildeforbedringer kan gjøre det lettere å oppdage kreftsvulster i tidlig stadium, og det kan bedre diagnostisering av enkelte sykdommer.

Ultralydbildekvalitet er avhengig av signalbehandlingskjeder med manuelt innstilte parametere. En datadrevet løsning kan forbedre båndpassfiltrering, tidsfrekvenskompensasjon, bildefiltrering og andre signalbehandlingsoperasjoner ved å gjøre dem pasientadaptive. Fjerning av flekkstøy kan legge til rette for bedre visuell persepsjon ved vurdering av ultralydbilder.

Maskinlæringsmodeller ble trent til å transformere ufiltrert stråleformet infase- og kvadraturkomponentdata (IQ) til E95-prosesserte bilder ved hjelp av en overvåket-læringsalgoritme. Datasettet besto av 7424 ekte hjertebilder og tilsvarende IQ-data fra 22 pasienter, tatt opp med en 4Vc-D 4D-probe. Flekkstøyreduksjonsnettverket ble trent til å gå fra IQ-data fra ett høydeplan til B-modusbilder av to sammenslåtte plan. På denne måten kunne flekkstøyreduksjonseffekten av høydeplansammenslåing oppnås med en M5Sc-D-probe som ikke har denne funksjonaliteten.

Forbehandling av data var avgjørende for å oppnå gode resultater. Dataene ble oppsamplet med bilineær interpolasjon, skalert med global maksimal absolutt skalering, og IQ-dataene ble representert på kartesisk form.

Hundrevis av konvolusjonelle nevrale nettverk ble testet. Modellene ble trent med gradientnedstigning av tapsfunksjoner basert på strukturell likhetsindeks (SSIM) og topp signal-til-støy forhold (PSNR) med Adam-optimalisatoren. Kvalitativ evaluering ble utført ved å inspisere skan-konverterte bilder, differensesplott og histogrammer. Ytelse ble målt kvantitativt med SSIM og PSNR.

Resultatene viste at et lite Inception U-Net kan brukes til å gjenskape signalbehandlingskjeden i Vivid E95-scanneren. Evaluering på et 10% testdatasett ga en SSIM-verdi på $(98,8 \pm 0,50)\%$ og en PSNR-verdi på $(35,8 \pm 1,38)$ dB. Visuelt var de predikerte bildene nesten identiske med E95-bildene. Testing av replikasjonsmodellen på data fra M5Sc-D-proben viste en økt kontrastoppløsning, men en endret flekkstøy. Flekkstøyreduksjonsmetoden ga glattede bilder med redusert flekkstøy på bekostning av kontrastoppløsningen.

Preface

This thesis was written in the course TFE4930 Electronic Systems Design and Innovation, Master's Thesis. It was the final criterion required to complete the MSc degree in Electronics Systems Design and Innovation at the Norwegian University of Science and Technology. The research was conducted while taking part in the Center for Innovative Ultrasound Solutions (CIUS) project.

Problem formulation:

> Develop a patient adaptive coherence imaging technique for improved image quality.

Acknowledgments

I want to thank:

- my supervisor Svein-Erik Måsøy for invaluable advice, continuous support, and honest feedback.
- my co-supervisor Erik Smistad for advice on machine learning and the FAST framework.
- Thomas Grønli for helping me with the LINUX server.
- Tore Grüner Bjåstad for advising me on how to align IQ data with GE Vivid E95 processed images.
- Tomáš Macholda for programming tips and his interest in my project.
- Otakar Kuchař for discussions about U-Nets, speckle generation algorithms, filter-based despeckling methods, and super-resolution.
- my classmates Ahmed Isifan, Dāvis Kļaviņš, Endre Dåvøy, Sivert Rosberg, Tobias Kristensen, and Torstein Nordgård-Hansen.
- Baptiste Martin for helping me take some breaks once in a while.
- my family for their tremendous support.

Colophon

The background part of the introduction (Section 1.1), the theory about acoustics (Section 2.1), and the deep learning theory (Section 2.3) contain modified or copied text from my specialization project: Gundersen (2021) [1]. A reference used in an entire paragraph is only mentioned the first time it is referred to. When a whole section uses the same reference it is explicitly mentioned at the beginning. Abbreviations, citations, and references to figures, tables, chapters, and sections are clickable hyperlinks. Software names are written in the SMALL CAPS text format. All ultrasound images are displayed using a colormap from GE. Scan converted images are displayed with the correct aspect ratio.

Table of Contents

List of Tables

List of Figures

Abbreviations

Adam Adaptive Moment Estimation. 23, 24, 45, 77
AE Autoencoder. 36, 37, 43, 50, 72
ANN Artificial Neural Network. 20, 21, 26–28, 30
API Application Programming Interface. 29, 40, 43, 46

CNN Convolutional Neural Network. 3, 4, 26, 30–33, 37, 43, 46, 70, 72

DL Deep Learning. 3–5, 18, 75
DNN Deep Neural Network. 3–6, 27, 36, 37, 40, 42, 77

FCNN Fully Convolutional Neural Network. 37, 43

GAN Generative Adversarial Network. 3, 4, 37
GPU Graphics Processing Unit. 7, 40, 71

IQ In-phase and Quadrature components. 3, 5, 6, 14, 15, 17, 39, 41–43, 45–49, 51, 70, 71, 75–77

ML Machine Learning. 3, 5, 6, 18, 19, 22, 24, 27, 28, 30, 39, 40, 45–47, 49, 69–71, 75, 78

PSNR Peak signal-to-noise ratio. 5, 25, 45–47, 49–51, 54, 66, 67, 71, 73, 74, 77

RAM Random-Access Memory. 7, 27, 40, 41
ReLU Rectified Linear Unit. 21, 34, 44, 50, 72
RF Radio Frequency. 4, 13, 14

SNR Signal-to-noise ratio. 2, 9, 11, 12, 14
SSIM Structural Similarity Index Measure. 5, 25, 26, 43, 45–51, 63, 66, 67, 71–74, 77

TFC Time Frequency Compensation. 15, 77
TGC Time Gain Compensation. 13, 16, 75, 77, 78
THI Tissue Harmonic Imaging. 2, 3, 12, 15

WGN White Gaussian Noise. 29, 70

Chapter 1

Introduction

Medical ultrasound scanners use signal processing algorithms with carefully tuned parameters to achieve reliable images [2]. The state-of-the-art solutions are based on complex statistical models, making the optimal choice of parameters sensitive to each patient. Using deep learning, data-driven neural networks may be used to replace the traditional signal processing. Due to their parallel processing, such networks may facilitate ultra-fast signal processing.

Another problem in medical ultrasound imaging is speckle noise and other noise types degrading the image quality. Speckle noise is difficult to remove without removing details of interest from the image. Most sophisticated despeckling algorithms are time-consuming due to their high computational costs. Deep learning-based despeckling may be faster and have improved performance compared to traditional filter-based approaches.

Section 1.1 is an extended and modified version of text from Gundersen (2021) [1].

1.1 Background

Diagnostic ultrasound is a non-invasive technique for imaging the inside of the body. It is used to study the heart, developing fetus, abdominal organs, blood vessels, and more [3].

Ultrasound is one of the most performed medical imaging modalities [4], and the area is under constant development. According to Fortune Business Insights, [5] the global ultrasound equipment market is projected to grow from $\$7.80$ billion in 2021 to $\$12.93$ billion in 2028. Artificial intelligence (AI) plays a substantial role in this, fueling market growth. In medical imaging, AI is, among other things, being used for more time-efficient examinations. Further, it is used to enhance image quality, often by reducing human errors.

Compared to other imaging modalities, ultrasound has some significant benefits. First of all, its relatively cheap cost makes it quite accessible [6]. Secondly,

ultrasound waves are acoustic, and the risk of damaging the patient is minimal [7]. In contrast, other imaging technologies like X-ray and computed tomography (CT) exploit potentially harmful ionizing radiation that increases cancer risk. Further, ultrasound captures images in real-time, making it suited for monitoring anatomic functions like heart movement.

Despite these benefits, ultrasound imaging faces the challenges of frequent poor resolution, images degraded by high levels of acoustic noise, speckle noise, and penetration depth limitations [3], [8].

These three problems are all reduced by Tissue Harmonic Imaging (THI). Further, it reduces reverberations and artifacts caused by grating lobes and side lobes [9]. Because of this, THI has become a default mode, as the method has proven to be advantageous in various applications.

1.1.1 Problem formulation

Despite the use of THI, image quality is still insufficient for diagnostic use for some patients. Image quality varies considerably from patient to patient, being highly dependent on specific anatomy [10], acoustic parameters of the tissue, and tissue inhomogeneities. Further, acquisition parameters play a significant role, but the clinician may optimize these.

Imaging obese patients is particularly challenging. A substantial factor for this is the trade-off between resolution and penetration [11]. Higher frequencies yield a better resolution but decrease penetration depth. In echocardiography, the area of interest is at high depths, meaning a compromise with resolution must be made to image deep enough. The consequences of this sacrifice are worse when imaging obese patients, as the pulses have to travel further through attenuating tissue.

The two primary causes of signal attenuation: scattering and absorption, show significant variations between individuals in soft tissue [12]. Due to the frequency-dependency of absorption, the frequency spectra of received ultrasound pulses are shifted towards the zero-frequency. The shift may vary from patient to patient, but the band-pass filtering in THI is always performed in the same way, making it sub-optimal. To optimize the filtering, detailed knowledge about the frequency content of the nonlinear ultrasound field is required. A filter with a better placed pass-band and stop-band would keep the frequencies of interest and suppress noise. This would improve both contrast resolution and Signal-to-noise ratio (SNR).

In addition to the filtering in THI, there are other manually tweaked parameters in the ultrasound signal processing chain. Some examples are filter cut-off frequencies, filter bandwidths, and parameters of image enhancement filters. As image quality is highly dependent on the signal processing algorithms [2], these parameters directly affect the image quality. Further, ultrasound signal

processing chains are often specific for each application.

Another challenge in ultrasound imaging is speckle noise. Unlike acoustic noise, speckle noise is correlated, making it challenging to reduce. Various speckle reduction methods exist, but many are time-consuming, computationally expensive, and decrease the frame rate. Some speckle reduction techniques do not support the use of cardiac probes. Faster spatial despeckling filters often cause blurring.

1.1.2 Related work

In recent years, more papers have been published regarding the use of Machine Learning (ML) in the field of medical ultrasound imaging. Various Deep Neural Networks (DNNs) have been developed to replace traditional signal processing chains. This includes the replacement of time-of-flight correction, beamsumming, filtering, envelope detection, logarithmic compression, speckle reduction, and more. Multiple papers presented in the following were listed by Luijten et al. (2022) [2] in their overview of Deep Learning (DL)-based ultrasound signal processing techniques.

Gundersen (2021) [1] showed that a simple DNN may be used to estimate the fundamental and the second harmonic center frequency of simulated nonlinear on-axis ultrasound fields. Detailed knowledge about the specific frequency content of the received signal may be used to optimize the filtering in THI. The paper showed that this may be done using simulated data.

Nair et al. (2020) [13] proposed a U-Net model that transforms raw channel In-phase and Quadrature components (IQ) data into B-mode images. The model performs delay-and-sum beamforming, filtering, envelope detection, and logarithmic compression. The training data was limited to synthetic data only. Despite this, the method showed promising results on both phantoms and in-vivo data. The test cases were limited to breast cyst examinations. In addition to outputting B-mode images, the U-Net had a second decoder part for cyst segmentation. Due to the high dimensionality of the data, the IQ data was heavily downsampled to 256x128x2, where the channels were the two IQ components. Further, no form of compounding nor speckle reduction was included in the model.

Karaoğlu et al. (2021) [14] compared different Convolutional Neural Network (CNN) architectures for mapping IQ data to B-mode images. The comparison showed that U-Nets and Generative Adversarial Networks (GANs) gave the best results.

Replacing beamforming with a U-Net is an unconventional application of the architecture. Much of the efficiency of CNNs is due to their property of invariance to local translation. As beamforming is highly geometry-dependent, learning such operations using a CNN is not trivial. Most papers published on

the topic have used traditional approaches for time-of-flight correction and focused the DL part on the beamsumming [13]. Many of these efforts aimed to accelerate time-consuming adaptive beamforming methods, improve spatial resolution, and increase the contrast resolution.

Khan et al. (2020) [15] proposed a method that replaces beamsumming and envelope detection by a CNN on subsampled Radio Frequency (RF) data. The work showed that the network could be optimized for different imaging settings with a style code vector in latent space. This way, one model may be used for various setting combinations. As signal processing pipelines in ultrasound imaging are optimized for specific applications, this could facilitate a more uniform solution.

In contrast to most efforts on replacing beamformers with a DNNs, Hyun et al. (2019) [16] included speckle reduction in the network. The method outperformed the traditional delay-and-sum beamformer. Similarly to Nair et al. (2020) [13], the network was trained on synthetic data, and the test cases were limited to breast cyst images.

Other DL-based despeckling attempts have focused on using CNNs to improve multiple-angle compounding. Jansen et al. (2021) [17] proposed an angular compounding method using the Radon transform. Results showed that by representing the image in the Radon domain, clutter may be removed by the DNN. The network revealed obscured structures and improved contrast resolution.

Gasse et al. (2017) [18] explored using a DNN to achieve the despeckling benefits of angular compounding when inputting a single image. This was done by training the network to map single images to compounded images. The results were promising, and the network offered a speed-up compared to traditional angular compounding that requires multiple emissions per image.

As sophisticated algorithmic despeckling methods often are time-consuming and computationally expensive, efforts have been made to replicate slow despeckling algorithms using DL. Dietrichson et al. (2018) [19] proposed a GAN that replicates the optimized non-local low-rank (NLLR) speckle reduction algorithm by Zhu et al. (2017) [20]. The results showed that the GAN successfully learned despeckling. Although the model was trained on cardiac images, tests on other body parts indicated that the method learned general speckle reduction. A model of reduced capacity was tested in a real-time implementation.

Further, GANs have been used to improve resolution. Choi et al. (2018) suggested a method that improves spatial resolution by mapping low-resolution images to high-resolution images using super-resolution. Such an approach may obtain high-quality images from a low-end ultrasound system.

A common factor for most of the presented DL methods is that they rely on supervised learning algorithms. This requires large amounts of target data, which may be challenging to obtain.

1.2 Objectives

Goals:

- Develop a DNN that learns and generalizes the native signal processing of the GE Vivid E95 cardiovascular ultrasound system with the GE 4Vc-D 4D probe. This includes band-pass filtering with time frequency compensation, envelope detection, frequency compounding, elevation compounding, logarithmic compression, image enhancement filtering, and more.
- Make a modified version of the DNN that performs despeckling on data acquired with the GE M5Sc-D probe.

1.3 Approach

A ML model was designed to learn and generalize the GE Vivid E95 native signal processing. This was done using a supervised learning algorithm to map unfiltered beamformed IQ data to GE Vivid E95 processed images.

The desired speckle reduction behavior was achieved by training the ML model to reconstruct elevation compounded images while only providing IQ data from one elevation plane. The model was then used to predict images from IQ data acquired with the M5Sc-D probe that does not support elevation compounding.

Predicted images were compared to their corresponding ground truth E95 images to evaluate the results. Structural Similarity Index Measure (SSIM) and Peak signal-to-noise ratio (PSNR) were used to quantify the similarity between the images. Visual comparisons of images and histograms were performed to evaluate their similarities and differences qualitatively.

1.4 Contributions

This thesis explores different aspects of replacing the ultrasound signal processing chain with a DNN. It investigates the possible advantages of such an approach and addresses the challenges.

Previous papers on DL-based reconstruction of B-mode images from IQ data have trained ML models using simulated breast cysts imaging data [13] or a small dataset from nerve imaging [14].

Large amounts of unbiased data are required to achieve robust solutions that generalize well. Little research has been conducted in the field of ML in medical ultrasound with access to real data. This work uses a large dataset of high quality from clinical cardiac imaging. The thesis investigates what is possible to achieve with a significant amount of real data.

Further, this thesis test the generalization ability of the proposed ML models by evaluating their performance on IQ data acquired using another probe.

By replacing the native GE Vivid E95 processing with a DNN, the ML model may learn to generalize the ultrasound signal processing steps. A data-driven solution may improve steps such as band-pass filtering, envelope detection, and logarithmic compression. Further, it may enhance speckle reduction by optimizing different types of compounding. More advanced signal processing techniques such as time frequency compensation (TFC) may be more precise if replaced with a data-adaptive DNN.

Gasse et al. (2017) [18] trained a DNN to single-frame images to multiple-angle compounded images to reduce speckle noise. This thesis investigates a similar approach for elevation compounding.

Lastly, autoencoders have been of little use in medical ultrasound imaging. This report investigates an unconventional use of the U-Net architecture.

1.5 Limitations

A ML model is often treated as a "black box" with no guaranteed behavior [2]. Training with a diverse dataset covering most possible scenarios will increase generalization, but the ML model may still encounter outlier cases and produce unexpected outputs. In a clinical setting, the ultrasound scanner must work as expected.

The dataset used in this work was limited to cardiac images from 22 patients. Because of this, the model may have learned patient-specific features. As all the images were of the heart, the trained model may not perform as well when imaging other body parts or using different acquisition parameters.

Due to the nature of the IQ data, data augmentation was challenging. Many image augmentation types like random flipping and rotation could not be used as they would produce unrealistic augmentations.

A disadvantage of speckle reduction is how it negatively impacts speckle tracking methods. As speckle tracking relies on the fact that speckle noise is correlated between consecutive frames, reducing the speckle noise will make it more difficult to follow the tissue movement using the speckle pattern [11]. Because of this, the proposed despeckling ML model may negatively impact the performance if used in applications like cardiac strain measurements.

Another limitation of the despeckling method is that it ignores one of the elevation planes of the IQ data when fitting the model. The network input effectively contains less information than the ground truth E95 image. Since elevation compounding relies on the decorrelation of speckles between different elevation planes, such an approach may result in an unexpected behavior where the network learns to add or remove information.

The large dimensionality of the data caused Random-Access Memory (RAM) limitations in the Graphics Processing Unit (GPU). For this reason, the batch size had to be small. Because of this, wider models (with more filters) could not be tested with high image resolution.

1.6 Outline

The remaining of the thesis is as follows: Chapter 2 provides Theory about acoustics, medical ultrasound imaging, ultrasound signal processing, and deep learning. Chapter 3 describes the Method and Implementation, Chapter 4 presents the Results and Observations, Chapter 5 is a Discussion of the results, Chapter 6 is a Conclusion, and Suggestions for Future Work are listed in Chapter 7. References are listed at the end.

Chapter 2

Theory

Some theory is needed to understand the remaining of this thesis. This section gives a brief theoretical review of relevant concepts in acoustics, ultrasound imaging, ultrasound signal processing, and deep learning.

Section 2.1 and Section 2.3 are based on modified or copied text from Gundersen (2021) [1].

2.1 Acoustics and medical ultrasound imaging

Section 2.1 is based on Angelsen (2000) [3], [12]. Other sources used are mentioned explicitly.

2.1.1 Waves, reflection, and scattering

Acoustic waves are longitudinal pressure oscillations moving through a medium in the form of compressions (high pressure) and rarefactions (low pressure). Ultrasound waves are acoustic waves with frequencies above 20 kHz, which are inaudible to the human ear. In medical ultrasound, the frequencies range between 1 to 20 MHz, depending on the application.

The speed of sound in a material is inversely proportional to the square root of the product of the material compressibility κ and mass density ρ. In medical ultrasound imaging, the speed of sound is usually assumed to be constant, equal to 1540 m/s. In reality, the difference in speed of sound in human fat and human muscle is in the order of 100 m/s.

Acoustic impedance Z is defined as the product of the mass density ρ and the speed of sound c. The acoustic impedance is crucial when describing what happens to a wave propagating from one medium to another, as differences in acoustic impedance cause reflections and scattering of the wave. If a pulse propagates between two materials with very different acoustic impedances, most of

the energy at the intersection is reflected or scattered. This is referred to as acoustic mismatch.

Reflection occurs when a wave hits a boundary between materials of different acoustic impedances with a large surface relative to its wavelength. The larger the acoustic mismatch is, the more energy gets reflected. An important notice is the frequency-independence of reflections.

Scattering, however, is highly dependent on frequency and occurs when a wave hits a small object relative to its wavelength. Such objects are referred to as scatterers and act as point sources redirecting acoustic energy in all directions. Scatterers are tiny variations in acoustic impedance.

2.1.2 Pulse-echo imaging

In pulse-echo imaging, a series of pulses are transmitted from a transducer into tissue and gets scattered and reflected due to variations in acoustic impedance. Some of the acoustic energy returns to the transducer and is recorded as oscillations. By assuming a constant speed of sound, the time delay between transmit and receive is used to calculate the depth of where the reflection took place. In a more realistic scenario, there are other factors in play. Various artifacts occur, like shadowing and reverberations. Phase-front aberration occurs due to a non-constant speed of sound in tissue.

2.1.3 Transducers

The part of the ultrasound system that transmits and receives ultrasound waves is referred to as the transducer (or probe). It consists of three main components: a piezoelectric crystal, a backing layer, and a matching layer. The piezoelectric crystal transforms (or converts) electrical energy into mechanical energy in the form of acoustic waves and vice versa. Behind the piezoelectric crystal is the backing layer, which reduces ringing by absorbing acoustic energy transmitted backward. In front of the crystal is the matching layer. It minimizes the amount of energy lost due to acoustic impedance mismatch between the piezoelectric crystal and the tissue of the patient.

Depending on the application, clinicians use different transducers. They differ in construction based on the footprint (or aperture size), arrangement of the piezoelectric crystal, and which frequency band they are designed for. Among the most popular are linear, curvilinear, and phased array probes. This report focuses on phased array transducers.

For phased arrays (or cardiac probes), used in heart imaging, there is a significant trade-off between penetration and resolution. Lower frequencies must be used to image deep enough to get sufficient contrast resolution and SNR. Further, the footprint of cardiac probes is small, designed to fit in the inter-

costal space (between the ribs). Ribs have a high acoustic impedance compared to soft tissue, resulting in strong reflections causing shadowing artifacts in the image. A small aperture yields poor lateral and elevation resolution.

More advanced ultrasound probes support three-dimensional imaging by stacking piezoelectric elements in the elevation dimension. Probes that support three-dimensional imaging are often referred to as 4D probes.

2.1.4 Attenuation

Due to attenuation, only a tiny amount of the transmitted acoustic energy returns to the transducer. The rest is mainly lost due to absorption and scattering. Other factors are transmission, refraction, and geometrical dispersion, which are not further discussed in this report.

Attenuation due to absorption is the phenomenon of mechanical energy being converted to heat. As a pulse propagates through an absorptive medium, the amplitude is lowered. Absorption is frequency-dependent, and the further a wave propagates in an absorbing material, the more attenuated it becomes. Absorption is quantified with the coefficient α, given in dB/cm/MHz, and is a material-dependent parameter. As absorption is frequency-dependent, it lowers the center frequency of a pulsed waved propagating [11]. The frequency shift becomes more significant for higher center frequencies. In soft tissue, the attenuation due to absorption is about 0.5 dB/cm/MHz for one-way propagation.

2.1.5 Tissue

Biological tissue is far from homogeneous, consisting of various tissue types like muscle tissue, connective tissue, and fat. It may be simulated as a collection of point scatterers. An ultrasound wave propagating through tissue gets attenuated as energy is scattered in random directions. Following the superposition principle, the total scattered signal is computed as the sum of all the scattered signals from each point scatterers in a volume.

Attenuation due to absorption varies from tissue to tissue. Water is a material with little absorption, but lungs filled with air and bone are highly attenuating tissue types.

Mixtures of fat and muscle cause reverberations. These are multiple echoes caused by the acoustic mismatch between the two tissue types. Further, subcutaneous fat leads to images corrupted by acoustic noise.

People have different tissue properties and structures, so image quality varies from patient to patient.

2.1.6 Image quality

Image quality in medical ultrasound is commonly quantified with the three metrics: resolution, contrast, and SNR. Depending on the context, the word resolution may have different meanings. It may be used to describe temporal resolution, but in the context of image quality, it refers to spatial resolution.

Spatial resolution

Spatial resolution is a measure of how close two objects can be to each other while still being seen as two separate objects. Depending on which dimension they are to be distinguished from each other, one refers to axial resolution, azimuth resolution, or elevation resolution. The axial (or depth/range) resolution is the resolution in the direction parallel to the ultrasound beam. Azimuth and elevation resolutions are the resolutions in the x and y-direction, respectively, perpendicular to the beam and to each other. x, y, and z complete a right-handed Cartesian coordinate system where the z-axis points down in the body [11].

Resolution is measured in meters, and low values correspond to higher resolutions. In medical ultrasound resolutions are typically in the order of tens to hundreds of micrometers. If an object is smaller than the resolution, it appears larger than its actual size.

Axial resolution Δz is given as half the spatial pulse length SPL:

$$\Delta z = \frac{SPL}{2} = \frac{N_p \lambda}{2}, \tag{2.1}$$

where N_p is the number of transmitted pulse cycles.

Spatial resolution in the direction perpendicular to the axial direction is determined by the beamwidth of the transmitted beam. Azimuth resolution Δx and elevation resolution Δy are given as:

$$\Delta x = 1.2 \cdot \frac{F}{D_{az}} \cdot \lambda \quad \text{and} \quad \Delta y = 1.2 \cdot \frac{F}{D_{el}} \cdot \lambda \tag{2.2}$$

where λ is the wavelength, F is the focal depth, and D_{az} and D_{el} are the aperture sizes in the azimuth and elevation directions, respectively.

Spatial resolution is proportional to the wavelength, thus inversely proportional to the frequency. A higher frequency yields a better resolution, as the wavelength decreases. Since transmit center frequency is an acquisition parameter, resolution may be enhanced by increasing the frequency.

Contrast resolution

Contrast, or contrast resolution, is a metric describing how distinct a scatterer appears in the image in comparison to unwanted parts of the ultrasound field

like side lobes and grating lobes. It is often measured in decibels by comparing signal levels in and around the region of interest.

Signal-to-noise ratio

SNR is a measure of the difference in power level between the signal of interest and noise, often measured in decibels. When imaging at high depths the ultrasound signal may drown below the noise floor if the frequency is too high. Lowering the transmit center frequency yields better SNR, but worse resolution.

The effects of frequency-dependent attenuation

Frequency-dependent attenuation affects image quality. Since it shifts the frequencies towards the zero-frequency, it effectively degrades the axial resolution. Further, attenuation lowers the contrast as it makes received signals weaker.

2.1.7 Tissue harmonic imaging

Since the speed of sound depends on mass density, it is higher in areas of rarefaction than in areas of compression. This causes the pulse to be distorted, making it look more and more like a saw-tooth than an ideal sinusoidal. The degree of distortion increases with the propagation distance and gives rise to harmonic frequency components [4]. This is called nonlinear wave propagation.

In THI the harmonic frequency components generated from nonlinear wave propagation are utilized to gain improved image quality. The second harmonic frequency is often used as it typically has the highest intensity of the harmonics in medical ultrasound [21]. There are two main techniques used in THI to isolate harmonic frequencies: frequency band filtering and pulse phase inversion.

Since harmonic frequencies are generated as the wave travels through tissue, reverberations are reduced when ignoring the fundamental frequency components. THI gives fewer echoes from off-axis structures of high reflectivity, less multiple scattering, and suppressed grating lobes [21]. It has considerably lower sidelobe levels than the fundamental frequency.

As harmonic frequencies are higher than the fundamental, the resulting image has a better spatial resolution. This applies to axial, lateral, and elevation resolution as seen in Equation 2.1 and Equation 2.2. SNR is effectively increased as the number of measurements is about doubled using the second harmonic frequency [4].

Since the intensity of harmonic components increases with propagation distance, THI is especially useful for imaging at high depths. THI works poorly at low imaging depths. This property makes THI particularly useful in echocardiography.

2.1.8 Speckle and stochastic noise sources

The primary noise type in medical ultrasound imaging is called speckle noise. It is caused by interference between signals from closely neighboring scatterers. Speckle noise is correlated multiplicative noise following the Rayleigh distribution. In an ultrasound image, it appears as a texture-like pattern.

In addition to the correlated speckle noise, there are sources of stochastic noise in medical ultrasound imaging. One of these sources is thermal noise caused by the electronics in the ultrasound system. Low-end systems generally have more thermal noise than high-end systems. Other sources of stochastic noise are the movement of the probe and other human errors.

2.1.9 Front-end

The ultrasound scanner front-end consists of various parts. First, a transmit pulse generator generates electrical pulses for each transducer element. Time delays are applied to the pulses to focus and steer the transmitted beam. When doing pulsed wave imaging, the system waits for the received signal before transmitting again. A T/R switch ensures that the system does not transmit and receive simultaneously. As the electronics used to process the received ultrasound pulse are designed for weaker amplitudes, the T/R switch prevents the higher amplitude signals from the transmit pulse generator from entering this part of the system.

Since the received ultrasound signal is highly attenuated, a low-noise amplifier amplifies the signal amplitude to a higher voltage. This amplifies both the signal of interest and the noise. As back-scattered signals from higher depths are highly attenuated, there is a large amplitude difference between the received signals from shallow and deep tissue. Time Gain Compensation (TGC) is done to reduce this dynamic range by applying a time-variable gain amplification. This enables using fewer bits when sampling the signal with an analog-to-digital converter. The digitized signal is referred to as the RF signal and contains channel data of band-pass signals from all transducer array elements.

2.2 Ultrasound signal processing

Section 2.2 is based on Hedrick et al. (2005) [11].

In the mid-end and back-end of the scanner, signal processing is done to the RF channel data to obtain the displayed image. Figure 2.1 shows a brief overview of the chain of signal processing steps to form a brightness mode (B-mode) image.

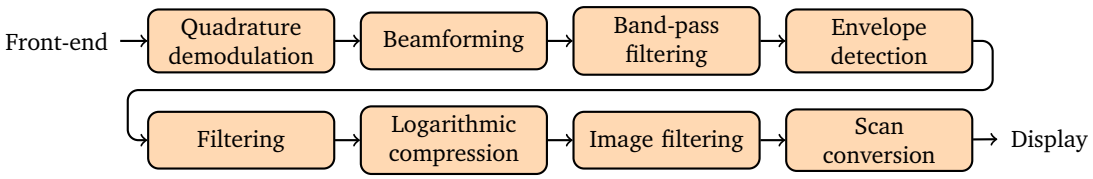


Figure 2.1: The B-mode ultrasound signal processing chain.

When displaying, additional processing is performed to facilitate better human interpretation of the images. This is not covered in this report.

2.2.1 Quadrature demodulation

Before constructing the ultrasound image from the RF data, the transducer pulse frequency (or carrier frequency) must be removed from the data [22]. This is done with quadrature demodulation. If this step is ignored, ripple artifacts will appear in the final image. The signal after quadrature demodulation is referred to as the IQ signal or the complex envelope.

Quadrature demodulation may be done in various ways, but two signal processing techniques are mainly in use.

The first approach is to calculate the Hilbert transform and downmix the signal. The Hilbert transform discards the negative frequency components, and what is left is a complex signal referred to as the pre-envelope. As this is a band-pass signal like the RF signal, downmixing with the center frequency is done to shift the frequency content to around the zero frequency in the Fourier domain. The Hilbert transform shifts the phase of the input signal by 90 degrees. When combining the shifted signal with the original, the ripples of the two signals fill each other's gaps. This gives a accurate estimate of the magnitude of the signal envelope, thus the energy. The Hilbert approach reduces ripple and provides great detail in the image. Unfortunately, it requires higher processing capabilities. Using a system with insufficient processing power may affect the frame rate [23].

The other approach is to demodulate the RF signal first and then do low-pass filtering. Like the Hilbert transform, the low-pass filter removes the negative frequency components. There is a difference between the two approaches regarding how white noise is affected. With the low-pass filter approach, the cut-off frequency may be chosen sto suppress the noise outside of the frequency band of interest. This results in a better SNR than when doing Hilbert transform followed by downmixing.

2.2.2 Beamforming

After quadrature demodulation, IQ data is beamformed to achieve receive focusing. A standard method is the delay-and-sum beamforming, where the sig-

nal from each channel is interpolated at the time-of-flight to a focal point and summed over all channels to obtain the total received signal from the point of interest. This process is repeated for all spatial points in the whole beamspace to get what is referred to as the beamformed IQ signal.

Since the IQ signal consists of complex-valued numbers, it may be represented in Cartesian or polar form.

2.2.3 Band-pass filtering

The beamformed iq data is band-pass filtered to suppress noise outside the frequencies of interest. The filtering is often implemented by mixing and low-pass filtering. Further, the band-pass filtering selects if imaging is done conventionally (using the fundamental frequency) or with THI (utilizing the second harmonic frequency).

Time frequency compensation

Due to the frequency shift caused by frequency- and depth-dependent attenuation, a fixed filter is sub-optimal. Time frequency compensation (Time Frequency Compensation (TFC)) is a technique that compensates for this effect. It is commonly done by changing the center frequency of the band-pass filter as a function of depth [24]. Estimates of the frequency shift may be used to optimize the TFC further.

2.2.4 Envelope detection

Envelope detection is then performed on the filtered IQ signal to get the magnitude of the complex signal. Traditionally, this is done by calculating the absolute value of the IQ signal. The speckle pattern appears after envelope detection.

Before further signal processing, the image may be upsampled by interpolation or low-pass filtered with decimation to reduce the data size.

2.2.5 Logarithmic compression

The received signal consists of both strong and weak back-scattered echoes, meaning it has a high dynamic range. Because of attenuation, reflections from deep tissue are weaker than those from shallow tissue. Dynamic range compression is applied to the envelope detected signal for simultaneous visualization of both strong and weak scatterers in the same image.

A common way of doing logarithmic compression is by converting the amplitude of the envelope to the decibel scale. Mathematically, this is done by calculating $20 \cdot \log_{10}(\cdot)$.

The dynamic range of the sampled data depends on the number of bits in the analog-to-digital converters, the TGC, and the penetration depth. When displaying the image, further compression is applied to obtain an image with a dynamic range suited for display. Typically the data is compressed to 8 bits with pixel values ranging from 0 to 255.

After the logarithmic compression, image enhancement filtering is often performed to make borders and structures smooth.

2.2.6 Scan conversion

When imaging using a linear probe, all scan lines are parallel. This means structures in the image have the right size relative to each other. This is not the case with a phased array. In this case, signals are received from different angles within a given sector, and the scan line spacing increases with depth. This cause shallow structures in beamspace to appear wider than they are. A geometrical transformation must be performed to obtain an image with the correct relative dimensions. Such a transformation is referred to as scan conversion.

In short, scan conversion is done by interpolating the data from a polar coordinate system to Cartesian coordinates. Typically bilinear interpolation is used, using the four nearest neighbors.

By convention, the scan converted image is flipped horizontally before it is displayed.

2.2.7 Dynamic range and gain

The scanner operator may tweak two settings called dynamic range and overall image gain when displaying the ultrasound images.

The dynamic range determines the total number of shades of gray. Increasing the dynamic range yields more grayscale levels in the chosen colormap. The result of this is smoother images with an increased level of detail. Decreasing the dynamic range increases the contrast resolution as fewer grayscale values are available.

Adjustment of the gain may be made to control the overall brightness of the ultrasound image. If the image appears dark, increasing the gain will reveal weaker signals in previously too dark regions.

2.2.8 Speckle reduction by compounding

In addition to the signal processing steps mentioned above, additional operations are implemented for speckle reduction. Three speckle reduction methods based on compounding are elevation compounding, multiple-angle compounding, and frequency compounding [23].

Elevation compounding

Elevation compounding is based on the fact that speckle is uncorrelated between different spatial locations [25]. Summation of non-coherent images created by neighboring elevation planes may be used to reduce the speckle noise (see Figure 2.2). As the method requires multiple elements in both the azimuth and elevation direction of the transducer array, it may only be implemented when using 4D probes.

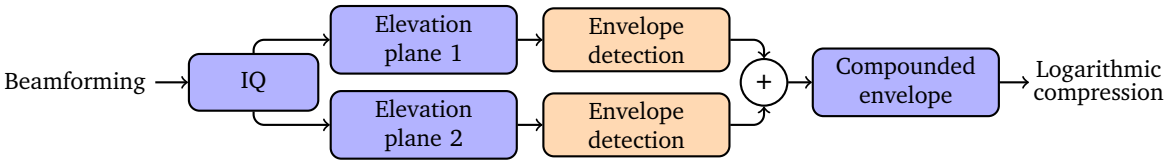


Figure 2.2: Elevation compounding using two elevation planes.

Multiple-angle compounding

Multiple-angle compounding is another speckle reduction method. The idea is to combine images from different angles, as speckle realizations are uncorrelated between images from different angles [26]. Rotation is compensated for by multiplication with two-dimensional rotation matrices. As multiple images must be formed, multiple-angle compounding affects the temporal resolution. Compounding images from different angles requires the use of a linear probe. It may not be done with the phased array transducers used in echocardiography.

In addition to reducing speckle noise, multiple-angle compounding sharpens the edges in the image. The most reflection from a boundary is achieved when imaging perpendicular to the boundary. If the ultrasound beam is parallel with a boundary, less energy returns to the transducer, and the edge in the image appears blurry. Imaging from different angles will increase sharpness due to the increase in reflected energy from the boundary.

Frequency compounding

Frequency compounding is based on the fact that speckle is frequency-dependent and looks different when imaging at different frequencies [26]. Averaging over images created using different frequency bands reduces the speckle noise.

Implementation is done by band-pass filtering the beamformed IQ data in parallel with multiple filters with partially overlapping pass-bands. Envelope detection is then performed on each of the filtered signals. The envelopes are then summed into one compounded envelope. Typically, band-pass filtering is done by mixing low-pass filtering.

Figure 2.3 shows an example of frequency compounding using two frequency spectrum bands.

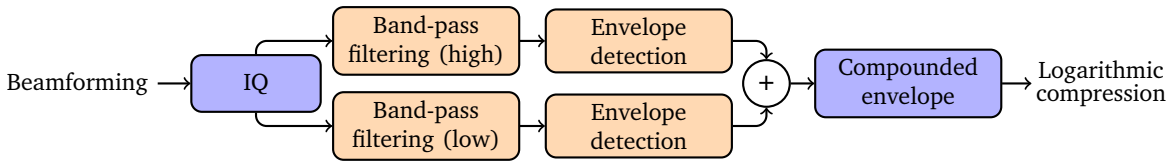


Figure 2.3: Frequency compounding using two overlapping frequency spectrum bands.

Spatial filtering

In addition to compounding-based methods, spatial filtering may be used to perform despeckling.

When using traditional filters, there is often a trade-off between reducing speckle and noise and losing sharpness due to smoothing of the image [24]. Ideally, the despeckling and denoising should be done without blurring the image and losing details of structures of interest.

More sophisticated despeckling filters are often time-consuming, causing problems with real-time implementation.

Examples of spatial filters that may be used for despeckling are Gaussian blurring, bilateral filtering, Wiener filtering, non-local means filtering, wavelet filtering, anisotropic diffusion filtering, and Laplacian pyramid-based nonlinear diffusion. All these filters must be tweaked manually.

2.3 Deep learning

The primary source used in Section 2.3 is the *Deep Learning Book* by Goodfellow et al. (2006) [27]. Other sources used are mentioned explicitly.

Deep learning (DL) is a subfield of ML that uses artificial neural networks (ANNs) inspired by the behavior of the human brain [28]. To understand DL, one must first understand the basic principles of ML.

2.3.1 Learning algorithms

ML may be divided into supervised, unsupervised, and reinforcement learning. Unsupervised and reinforcement learning will not be further discussed in this report.

The objective of supervised learning is to find a model or function f that maps some features X to corresponding labels y (see Equation 2.3). The features are the variable inputs of the model, often consisting of a collection of observations that are either raw or processed data. The labels (or targets) are the desired outputs corresponding to the observations; hence the name supervised learning.

$$y = f(X) \tag{2.3}$$

After training the model, the mapping function f may be used to make predictions $\hat{y}$ from unseen data, meaning data not used when training the model. In some cases, the model may be repeated multiple times to amplify its effect.

Supervised learning is typically used for two problems: classification and regression. When labels y are categorical, the problem is a classification problem, where the predicted label $\hat{y}$ can only fall into one of the available categories. In contrast to classification problems, the output from a regression problem may have arbitrary values.

2.3.2 Datasets

A dataset is a collection of data pieces the computer treats as one unit. In supervised learning, the term dataset includes features X and the corresponding labels y.

When choosing or creating a dataset for ML, the quantity and quality of the data is critical. More data is generally better, as the model will see more cases, increasing its robustness and generalization ability. A model that produces accurate predictions on unseen data is said to generalize. For the training to be smooth and fast, the dataset must be relevant and well-balanced. The dataset should cover most of the possible examples typical for the given problem. Irrelevant data may make the model learn undesired patterns. This may be solved by cleaning the dataset. If the dataset is not well-balanced, the trained model may be biased to focus more on some features than others.

An easy way to make a cheap, clean, large, and well-balanced dataset is to generate a synthetic (or fake) dataset. This may be an easy way to start a ML project, but it may not represent real data. A model trained using synthetic data may not perform as well on real data. The fake data may not be predictable or too predictable.

Generally, the dataset is split into three subsets: a training set, a validation set, and a test set. The training set is usually the largest of the three subsets and is used to train the ML model. During the training process, the validation set may be used to evaluate the performance of the model. It is not used to train the model but may be used to decide model hyperparameters. After training, the features from the test set are used as the input in the mapping function to verify that the model works well on unseen data, meaning data not used in the training process. The predictions $\hat{y}$ based on the unseen test data are then compared to the true labels y to determine model performance.

2.3.3 Artificial neural networks

An Artificial Neural Network (ANN) is a mapping like the one described in Equation 2.3.

Figure 2.4 shows a possible realization of an ANN. This specific model has an input layer with five nodes, one hidden layer with three nodes, and an output layer with one node. The layers are dense (or fully connected), meaning each node is linked to all nodes in the following layer.

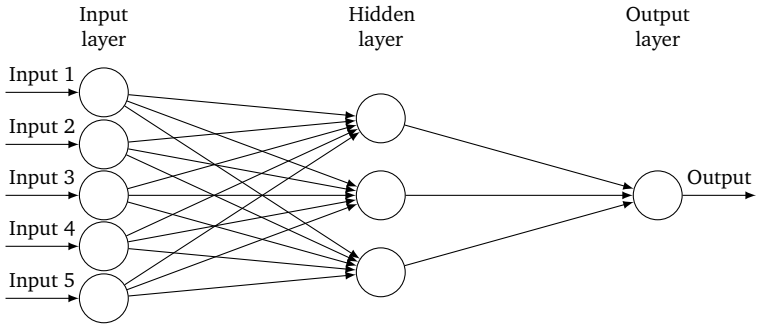


Figure 2.4: An example of an ANN.

The word deep in deep learning refers to neural networks with more than two hidden layers [29]. Increasing the number of nodes and hidden layers in an ANN increases its capacity. A higher capacity means more flexibility to express the relation between inputs and outputs. Generalization may be improved by adding more layers while holding the number of parameters constant [30].

Figure 2.5 shows how the output is calculated from the inputs of each node.

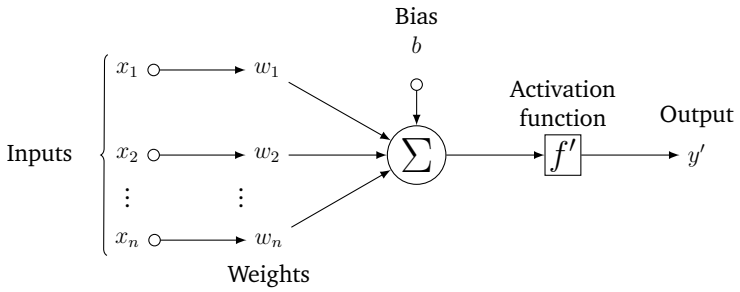


Figure 2.5: The calculation of the output y' from one node in an ANN.

The output value y' from each processing node is calculated using an activation function f' on a weighted sum of all the inputs x_i and a bias b:

$$y' = f'\left(\sum_i (w_i \cdot x_i) + b\right) \tag{2.4}$$

where w_i are the weights for each of the inputs.

Some commonly used activation functions are the linear function (that outputs the input), hyperbolic tangent ($\tanh$), sigmoid (σ), and Rectified Linear Unit (ReLU). For classification tasks, the softmax activation function is a popular choice. Equation 2.5 shows the expressions for sigmoid and ReLU.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad \text{ReLU}(z) = \max(0, z) \tag{2.5}$$

In the input layer and hidden layers, nonlinear activation functions are preferred. They allow the network to learn more complex data structures. Examples of such functions are hyperbolic tangent, sigmoid, and ReLU.

In the last years, the ReLU activation function has gained increased popularity. It makes negative values equal to zero and is linear for positive values. This makes the computational cost lower than for the sigmoid, which requires exponential calculations. The semi-linearity of the ReLU is desired as ANNs generally are easier to optimize if they have a closer to linear behavior. Since the weights of an ANN may have negative values, the ReLU activation function can be used with negative input values despite throwing away negative values.

The activation function in the output layer should be chosen depending on the span of the desired output values. In regression problems where the output should be of any arbitrary value, activation functions such as the linear function or ReLU may be suited. If the outputs should be confined in the interval of -1 to 1 or 0 to 1, the hyperbolic tangent or the sigmoid activation function may be used, respectively.

2.3.4 Loss functions

Before training an ANN, a loss function l must be chosen for the algorithm to know when it is improving. The loss function (or objective function) is a measure of the discrepancy between the predictions $\hat{y}$ and the labels y from a training set.

According to the "no free lunch" (NFL) theorem, the performance of all supervised optimization algorithms is equally good when averaged over all possible problems [31]. If a loss function works well for one problem, it might not work that well for some other problems. The choice of loss function depends on the application.

In regression problems a loss function commonly used is the quadratic loss:

$$l(y, \hat{y}) = (y - \hat{y})^2. \tag{2.6}$$

Each n-th prediction $\hat{y}_n$ in a set of N predictions, is dependent on the weights W, biases B and inputs x_n:

$$\hat{y}_n = f\left(x_n; W, B\right). \tag{2.7}$$

A total loss (or empirical risk) L is defined as the average loss over the data points:

$$L(W, B, X, y) = \frac{1}{N} \sum_{n=1}^{N} l\left(y_n, f\left(x_n; W, B\right)\right). \tag{2.8}$$

Using the quadratic loss function, the total loss becomes the mean squared error (MSE).

The goal is to minimize the total loss function with respect to the weights and biases:

$$\hat{W}, \hat{B} = \underset{W,B}{\arg\min}\, L(W, B, X, y). \tag{2.9}$$

This is done using a numerical search, as it cannot be solved in a closed-form.

2.3.5 Training an artificial neural network

Training (or fitting) a ML model is changing the parameter values of the weights W and biases B to minimize the total loss L. This is done by forward propagating the inputs X through the network, calculating the total loss, and then backpropagating through the network to update the parameters such that the total loss is lowered. This process is repeated as the model continues to learn. After multiple training iterations, the total loss is expected to decrease, meaning the neural network should produce more desirable outputs.

In each iteration, the gradients of the total loss with respect to the weights and biases are calculated. The network parameters are then updated through what is called gradient descent. This is done by subtracting a scaled version of the negative gradient from each parameter (see Equation 2.10). The scaling factor or step size is referred to as the learning rate η, determining how fast the network learns.

$$\begin{aligned} W &= W - \eta \cdot \nabla_W L \\ B &= B - \eta \cdot \nabla_B L \end{aligned} \tag{2.10}$$

The ultimate goal is to reach a global minimum of the total loss function where the gradients are zero. In practice, the algorithm often converges to a local minimum. Some local minima are better than others; thus, the aim is to reach the lowest one possible.

When training, one pass of the whole dataset is called an epoch. As there is no guarantee for the gradient descent to converge well on the first try, the gradient descent is repeated through multiple epochs to minimize the total loss further.

Parameter initialization

Before the first training iteration, the weights and biases of the network must be initialized. If done correctly, the gradient descent algorithm will converge faster to a minimum [32]. Parameter initialization may be done in various ways, but the most common approach is to assign random values or zero. Usually, zero initialization is used for biases, which means they have no effect. Further, the initial weights are assigned random values. If the weights were initialized with zero, the derivative with respect to the loss function would be equal for all weights; thus, the performance would be no better than that of a linear model.

Gradient descent variations

Three variants of gradient descent are differentiated by the amount of data used when computing the gradients [33].

The first is batch gradient descent, where the gradients are computed using the whole training set. This is rarely used as it is slow and may be limited by the available memory.

Another approach is stochastic gradient descent (SGD), where the gradients are calculated for each training sample. This causes large fluctuations in the loss function during training as the variance is increased only using one sample at a time. A benefit of such fluctuations is the ability to escape undesired local minima. When approaching the very minimum, fluctuations become disadvantageous as the SGD overshoots the minimum. The issue is partly solved by decreasing the learning rate, which increases the training time.

Mini-batch gradient descent combines the best aspects of the previous two variants, making it the most widely used. By updating the parameters for a part of the training data at a time, the variance is reduced compared to SGD, and the training is more time-efficient than with batch gradient descent. The amount of data samples in each update is called the batch size. Smaller batches find minimums that generalize better on the validation and test sets [32].

The term SGD is commonly used when doing mini-batch gradient descent, not only actual stochastic gradient descent.

The Adam optimizer

Variations of the gradient descent algorithm have been developed to optimize the training process. Some of these include momentum updating weights and biases depending on previous gradient descent steps. In this report, the attention is focused on the Adaptive Moment Estimation (Adam) optimizer.

Adam is an adaptive optimizer, meaning the learning rate changes adaptively during the training process. RMSprop is another optimizer that divides the gradient by a moving average of the square of previous gradients. The Adam

optimizer further adds an exponentially decaying average of past gradients. It introduces two new parameters β_1 and β_2 that are exponential decay rates for the first- and second-order moment estimates. For numerical stability, a small number ϵ is added to the denominator.

Below is the pseudocode for the Adam optimizer, copied from the original paper [34].

α : Stepsize / learning rate

β_1, β_2 : Exponential decay rates for the moment estimates

ϵ : Small constant for stability

$f(\theta)$: Stochastic objective function with parameters θ

θ_0 : Initial parameter vector

$m_0 \leftarrow 0$ (Initialize first moment vector)

$v_0 \leftarrow 0$ (Initialize second moment vector)

$t \leftarrow 0$ (Initialize timestep)

while θ_t not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)

$\widehat{m}_t \leftarrow m_t / \left(1 - \beta_1^t\right)$ (Compute bias-corrected first moment estimate)

$\widehat{v}_t \leftarrow v_t / \left(1 - \beta_2^t\right)$ (Compute bias-corrected second raw moment estimate)

$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / \left(\sqrt{\widehat{v}_t} + \epsilon\right)$ (Update parameters)

end while

return θ_t (Resulting parameters)

AdaMax is a variation of the Adam optimizer where the second-order moment is replaced by the infinite-order moment. Many other variations of the Adam optimizer exist, and some are better suited for specific problems than others.

2.3.6 Metrics of performance

When evaluating the performance of a ML model, various metrics are used, depending on the type of problem. Accuracy, precision, and recall are commonly used when doing a classification. These metrics may not be used for regression

problems, as the predictions are not belonging to any classes. For regression, the MSE is widely used:

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} [y_n - \hat{y}_n]^2 . \tag{2.11}$$

Due to its quadratic terms, the MSE is sensitive to outliers.

Performance metrics are usually calculated after each update of parameters and epoch during training. They are calculated using the training set and a validation set during the training. After the training, the metrics are computed by evaluation of model performance on the test set.

Peak signal-to-noise ratio

When dealing with image data, the MSE may be converted to the decibel scale relative to the maximum pixel value of the original (or reference) image I. Doing so makes the metric independent of the number of bits used to store the image. Further, it compresses the number to fewer significant digits. This is referred to as PSNR and is defined as:

$$\mathrm{PSNR} = 10 \cdot \log_{10} \left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}} \right) . \tag{2.12}$$

The MSE in the PSNR expression is the average MSE in the image. Mathematically, it may be formulated as:

$$\mathrm{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 , \tag{2.13}$$

where the sums iterate the image rows m and columns n, respectively.

In lossy compression, PSNR values are typically between 30 and 50 dB, where higher values are better.

Structural similarity index measure

Some problems may occur when quantitatively determining the similarity between two images using the previously mentioned metrics. As they compare images pixel-wise, variations in brightness and contrast, small spatial shifts, and slight rotation of structures may heavily decrease the similarity value. The structural similarity index measure (SSIM) addresses these problems by emphasizing the perspective aspects closer to how humans interpret the similarity between two images. SSIM is given as a percentage between 0 and 100%, where image pairs with values above 95% are considered close to identical, depending on the image type [35].

The SSIM metric consists of three components: luminance (l), contrast (c), and structure (s) [36]. Equation 2.14 shows the expression of each individual component:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}, \quad c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}, \quad s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}, \tag{2.14}$$

where μ_x and μ_y are the means and σ_x^2 and σ_y^2 are variances of x and y, and σ_{xy} is the covariance between the two. $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$, and $c_3 = c_2/2$ are constants that stabilize the fraction and prevent division by zero. L is the dynamic range of the pixel-values. For images saved as unsigned integers using 8 bits the dynamic range is 255. The default values for the stability constants are $k_1 = 0.01$ and $k_2 = 0.03$.

The SSIM is given as a weighted combination of these three components:

$$\mathrm{SSIM}(x,y) = \left[l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma\right]. \tag{2.15}$$

By weighting all components equally ($\alpha = \beta = \gamma = 1$), the expression for SSIM simplifies to:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)\,(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)}. \tag{2.16}$$

Variations of SSIM have been suggested, such as multiscale SSIM. It combines the SSIM values calculated using the original image and downsampled versions of the image.

Zhao et al. (2018) [37] suggested using SSIM as a loss function when training CNNs, showing promising results.

2.3.7 Overtraining and overfitting

According to Tetko et al. (1995) [38], a neural network with a sufficient amount of nodes in the hidden layer can memorize an arbitrary training set, including noise. A model should not remember the examples but recognize the underlying patterns relating the input to the output. The term overfitting refers to exceeding the optimal capacity of an ANN, meaning the network consists of too many parameters [38]. Overfitting may be solved by training on more data, which generally improves the accuracy of a model. This is not always possible.

If the number of epochs becomes too large, meaning the network is trained for too long, its performance gets worse than if it was trained with fewer epochs [38]. This is referred to as overtraining and is seen as an increase in the generalization gap, meaning the deviation between training performance and validation performance. This gap may be visualized using learning curves, which

are metrics plots as a function of epochs. It is a standard procedure to plot the training and validation metrics together to inspect if the model generalizes well.

Overtraining may be avoided by adjusting the number of nodes and hidden layers. The disadvantage is that the model must then be retrained every time the network topology is changed. In practice, regularization is the preferred alternative.

2.3.8 Regularization

Regularization methods help the model generalize better by slightly modifying the learning algorithm.

A way of avoiding overtraining is to observe the learning curves to decide when is the best time to stop training. With early stopping, the training ends when a validation metric of choice has not decreased during a given amount of epochs, referred to as the patience.

Another widely used regularization method is dropout regularization, which sets a probability of ignoring nodes in the hidden layers. Effectively the network becomes smaller, but this form of regularization has proven to improve the performance of ANNs notably. When using dropout regularization, the network can not rely on specific nodes as they may be being ignored. This way, the weights are more evenly distributed across all features, improving generalization.

2.3.9 Batch normalization

Batch normalization is a technique that standardizes the input of a layer for each mini-batch. It may stabilize the training and reduce the number of epochs required, meaning faster convergence. Another advantage is that batch normalization decreases the importance of the initial model weights. Further, it requires less data for generalization.

As batch normalization requires the calculation of the average value of the batch, it becomes less stable for small batch sizes. The additional computations of the normalization increase the training time. For some DNN architectures, batch normalization performs poorly. An example of this is Recurrent Neural Networks (see Section 2.4.2).

2.3.10 Data pipelines

Small datasets that fit within the available RAM may be loaded directly into memory when training a ML model. This is often not the case for larger datasets. Instead, the data may be streamed through a data pipeline. Data pipelines reduce the memory usage when training, as only portions of the data, are loaded

at once. Despite this, the rate of loading data is often the performance bottleneck during training.

Figure 2.6 shows an example of a data pipeline consisting of two parts. In the first part, data loading and pre-processing are performed. Both of these steps are deterministic. Data loading and pre-processing may be time-consuming procedures. By caching the pre-processed data, this first part of the pipeline may only be executed once during the first training epoch [39]. In the following training epochs, the pre-processed data may be loaded directly from the cache instead of repeating the initial steps. The second part of the pipeline has a stochastic behavior caused by the shuffling and the data augmentation. Since the shuffling and augmentation differ for every epoch, these steps are repeated throughout the training process.

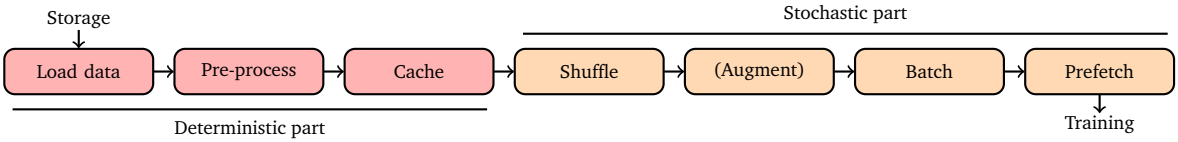


Figure 2.6: An example of a data pipeline. Data loading, pre-processing, and caching (in red) are only performed once. Shuffling, batching, augmentation, and prefetching (in orange) are repeated during training.

After reading the data from the cache, pre-processing is done to format the raw data in a format that suits the ML model. It may include resizing and scaling. If done correctly, pre-processing will improve the model performance.

Scaling

Data used in ML can have various units, meaning different features and labels can have their own scales. In an ANN this becomes problematic when the model tries to find the mapping function from inputs to outputs. Differently scaled data results in some weights being much larger than others, making nodes more sensitive to some of their inputs [40]. Such a network is unstable with lower overall performance and ability to generalize. To avoid this problem the features and labels are scaled before training. As the network outputs scaled values, post-processing is done to rescale the predicted labels to their original scales.

A commonly used scaling technique is maximum absolute scaling:

$$x_{scaled} = \frac{x}{\max\left(|x|\right)}. \tag{2.17}$$

Such scaling outputs values with an absolute value less than or equal to 1. If the data only consists of positive values it outputs values between 0 and 1.

Shuffling

When loading data from the cache a shuffle buffer of a given size is filled. Shuffling of the dataset is typically done when doing batch gradient descent or mini-batch gradient descent to ensure that the batch used to compute the gradients is representative of the overall distribution of the data. By doing this the variance is reduced and the model remains general, resulting in less overfitting. An obvious example is a case of classification with a dataset sorted by class. If not shuffled, the gradient will not be calculated based on a batch that is representative of the overall distribution of the data.

Data augmentation

After shuffling, the data may be augmented. Data augmentation is the process of transforming the data such that the features are changed while preserving the way they correspond with the labels. Because of this, data augmentation requires domain knowledge. Augmented data should be realistic, meaning it should be similar to examples typical for the given problem. There is no universal data augmentation that performs well for all tasks.

Data augmentation is particularly useful when training on small datasets, as it may be used to "generate" more training examples. This may improve generalization and make the training more robust by decreasing the risk of remembering examples from the training dataset [41].

Noise augmentation with White Gaussian Noise (WGN) is a commonly used augmentation type. Depending of the nature of the data it may be added or multiplied to the data. These approaches are referred to as additive WGN and multiplicative WGN, respectively. Traditionally noise is added to input data, but it may be added to activation functions, gradients, weights, and biases.

For image data, common augmentations are random rotation, shifting, zooming, and flipping. If dealing with a segmentation task, such augmentations may generate examples mimicking viewing an object from different angles, which may increase generalization and robustness. Some other image augmentations are random brightness, contrast, and blurring.

Batching and prefetching

After data augmentation the data is loaded in batches. In order to speed up the rate at which data is fed to the training algorithm, prefetching may be used. Prefetching overlaps the preparation of data from the pipeline with the training. This way the time it takes to extract data is reduced. Application Programming Interfaces (APIs) like TENSORFLOW and PYTORCH allow for parallelization of prefetching and other parts of the data pipeline.

2.3.11 Convolutional neural networks

A convolutional neural network (CNN) is an ANN that contains one or more convolution layers. Such layers are typically used if the inputs are structured in a grid shape like for image data. The use of CNNs is not limited to two-dimensional data.

Convolution layers perform convolution operations, which in the context of CNNs involves multiplication between inputs and weights like in dense layers. Technically, this operation is cross-correlation, but in the context of ML the term convolution is widely used.

For two-dimensional input data, the multiplication is performed between the inputs and a two-dimensional array of weights called a kernel or a filter (see Figure 2.7, inspired by [42]). The size of each kernel must be smaller than the dimensions of the input data and is called the kernel size. Typically each convolutional layer consists of multiple filters performing element-wise multiplications with the inputs. The results of all the multiplications are summed to a single value. This is done repetitively by systematically sweeping over overlapping segments of the input data. The step size of the sweeping is determined by the stride. The output from a convolutional layer is called a feature map. The amount of filters determines the number of channels of the output.

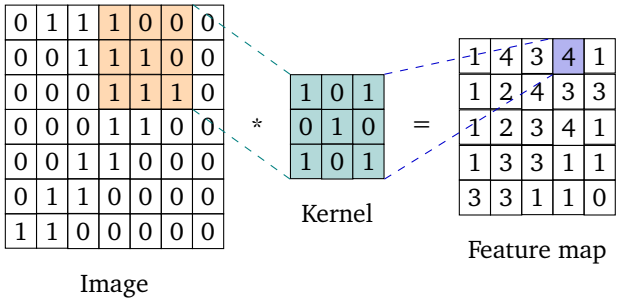


Figure 2.7: Convolution with a $3x3$ kernel and a stride of 1.

In order to obtain the same amount of samples in the height and width dimensions at input and output padding may be used. This is typically done by adding zeros. The shape of the feature map from a convolutional layer is dependent on the kernel size, the number of filters, stride, input dimension, and chosen padding type.

Deeper networks may be created by stacking multiple convolutional layers. This increases the complexity of the model, giving it more parameters to express patterns between the inputs and the outputs. The width of a network is determined by the number of filters.

Traditionally, CNNs have been used for feature extraction [43]. Dense layers have then been added as the remaining hidden layers and output layer. Flattening is done to make the data one-dimensional to connect the feature extractor

part and the dense layers. This may be problematic for images of large input sizes, as the connecting dense layers may have a large number of nodes, resulting in large and slow networks.

Much of the efficiency of CNNs comes from their ability to detect the presence of a specific feature of interest, regardless of its location in the image. This is referred to as translation (or spatial) invariance [44].

Pointwise convolution is a special kind of convolution using a 1x1 kernel. For image data, such a convolution maps each input pixel with all its channels to its corresponding output pixel [45]. This way, a stack of feature maps may be reduced by a linear projection. Unlike when using kernels of larger sizes, the surrounding pixels are ignored. Pointwise convolution has the advantage of reducing the number of depth channels. This saves time and speeds up the convolution operation by avoiding multiplication between volumes of large depths. It is common practice to use pointwise convolution in the output layer of a CNN.

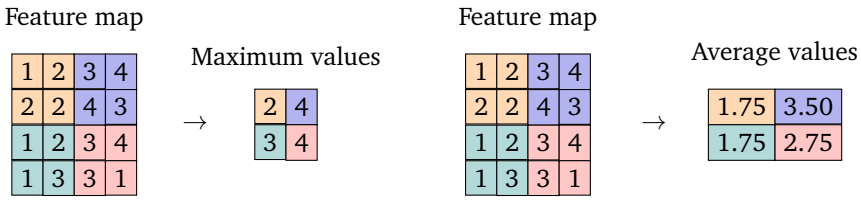
2.3.12 Downsampling and upsampling layers

Downsampling layers

After convolution layers, it is common to add a downsampling layer that reduces the dimensionality of the feature map. By doing so, the number of parameters is reduced. This results in a faster model with a smaller chance of overfitting. In this context, the word "depth" describes the number of downsampling layers in the model.

A window of a given size slides over the feature map with a chosen stride and outputs one value for each step, depending on the type of downsampling. Each feature map is downsampled independently. The height and width of the feature map are reduced, but the depth is unchanged. Usually, the stride is 2, and the window size is 2x2 such that the windows do not overlap. This effectively halves the feature map in the height and width dimensions. The idea is that the downsampled feature map keeps essential information.

Maximum pooling and average pooling are two common types of downsampling, taking the maximum or average value in each pooling window, respectively (see Figure 2.8). Unlike convolution layers, pooling layers have no parameters.



(a) Maximum pooling. (b) Average pooling.

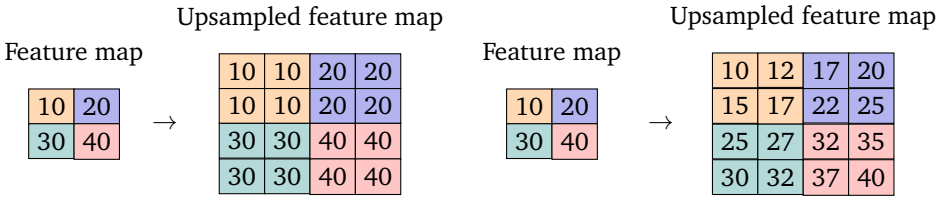
Figure 2.8: Two types of pooling: (a) maximum pooling and (b) average pooling. In both cases, the pooling window is $2x2$, and the stride is 2.

Strided convolution is an alternative way to downsample a feature map. Pooling is a fixed operation. Replacing it with a convolution layer with a stride of 2 may be used to learn any kind of pooling. This allows the model to express more complex pooling operations. According to Springenberg et al. (2015) [46], a network with strided convolutions may improve overall accuracy and stabilize the performance of a model. A disadvantage of strided convolution is the additional trainable parameters.

Upsampling layers

In contrast to pooling layers that downsample a feature map, upsampling layers increase their height and width. The number of channels of the feature map is held constant.

There are various ways of doing two-dimensional upsampling. Some popular methods are nearest-neighbor and bilinear interpolation (see Figure 2.9). More sophisticated interpolation techniques exist, like bicubic, but they require more time-consuming computations. Such techniques may not be desired in CNNs as they will slow down the model.



(a) Nearest-neighbor interpolation. (b) Bilinear interpolation.

Figure 2.9: Two types of upsampling techniques: (a) nearest-neighbor interpolation and (b) bilinear interpolation. The interpolation factor is 2.

Nearest-neighbor interpolation copies the value from the nearest pixel. By duplicating existing pixels, the resulting image appears blocky. Bilinear interpolation, on the other hand, uses linear interpolations to nearby pixels to calculate

the value of a given pixel. This makes it more suited for data of a continuous nature. Bilinear interpolation is performed by linear interpolation in one direction followed by another linear interpolation in the other direction [47].

As an alternative to interpolation, a transposed convolution (or deconvolution) layer may be used for upsampling (see Figure 2.10). It may be thought of as inverse convolution. Transposed convolution provides more flexibility than fixed interpolation, allowing a model to learn how to upsample the feature maps and fill in details [48].

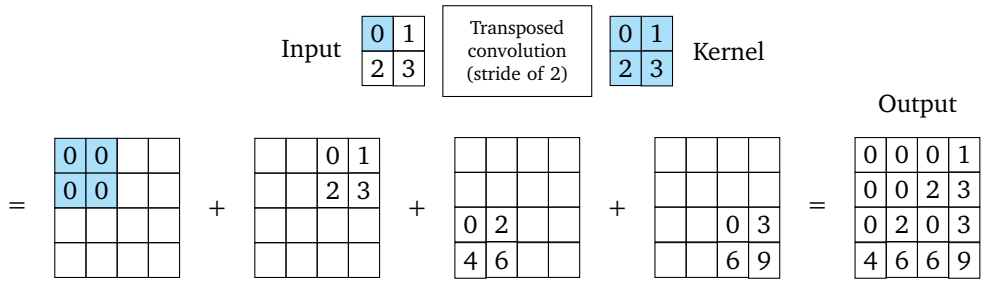


Figure 2.10: Transposed convolution with a $2x2$ kernel and a stride of 2.

2.4 Convolutional neural network architectures

CNNs are used for a wide range of applications, ranging from weather forecasting to the generation of fake videos. There exist many CNN architectures, and some are better suited for specific problems than others. Figure 2.11 shows a default convolution module, the fundamental building block of many CNN architectures.

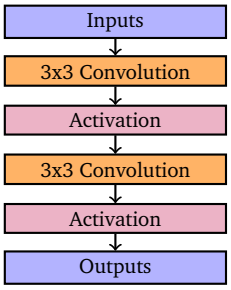


Figure 2.11: The default convolution module with two 3x3 convolution layers and two activation function layers.

2.4.1 VGG-16 and VGG-19

The VGG (also referred to as OxfordNet), is one of the most well-known CNN architectures. The original paper [49] includes two models: VGG-16 (see Fig-

ure 2.12) and VGG-19, of depth 16 and 19, respectively. Every convolution layer has a 3x3 kernel size with a stride and padding of 1. Maximum pooling with a pool size of 2x2 and a stride of 2 is used to downsample the feature maps, which decreases the number of parameters. The number of filters is doubled after each downsampling layer.

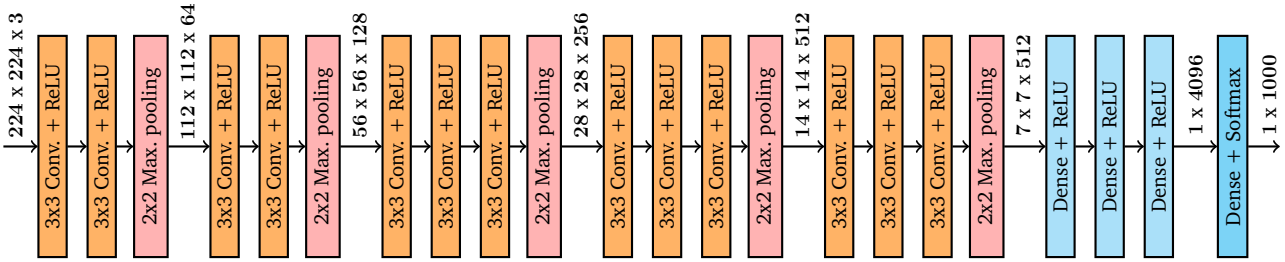


Figure 2.12: The VGG-16 architecture.

The VGG architecture was proposed for the ImageNet challenge [50] in 2013, where the goal was to classify $1,000$ classes from a large image dataset. The images were 3 channel RGB images cropped to 224x224 pixels. As the output should be a probability, the softmax activation function was used. VGG gained popularity due to its simplicity. Its 3x3 convolutions made the VGG faster than the previous winners: AlexNet [51] and ZF Net, [52] which used 11x11 and 7x7 convolutions, respectively. The combination of smaller kernels and more layers increases the non-linearity of the model, which is positive.

Due to the high number of convolution layers in the VGG architecture, it suffers from the vanishing gradient problem. When training a model using gradient descent and backpropagation, the parameters are updated according to Equation 2.10. During backpropagation, the gradients are calculated moving from the output layer to the input layer, layer by layer. Following the chain rule, the gradients get multiplied all the way to the initial layer. If a network becomes too deep and its activation functions output numbers between -1 and 1, the gradients of the loss function will approach zero for the first layers. Effectively, this means that these parameters are not being updated.

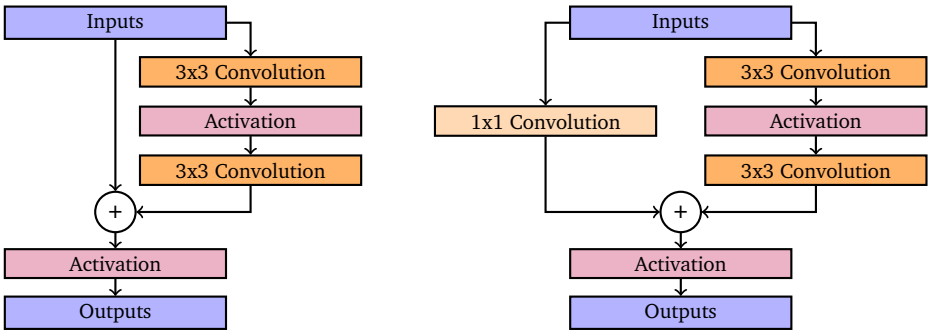
The vanishing gradient problem may be solved by the use of other activation functions that do not restrict the outputs to numbers of an absolute value smaller than 1. An example of such an activation function is ReLU, which avoids small gradients.

2.4.2 Residual neural networks

Another solution to the vanishing gradient problem is the use of Residual Neural Networks (ResNets) [53]. Such networks provide residual connections straight

to earlier layers. This way backpropagation may be done through the residual paths, avoiding some activation functions that "squeeze" the gradients.

Figure 2.13 (inspired by [54]) shows two different types of residual modules: with and without pointwise convolution on the residual connection. By adding pointwise convolution with the equal number of filters as the 3x3 convolution layers, one may eliminate the issue of having a different number of channels when doing the summation.



(a) Residual convolution module.

(b) Residual convolution module with 1x1 convolution.

Figure 2.13: Two residual modules with and without pointwise convolution on the residual connection.

2.4.3 Inception

Unlike VGG and ResNet, the Inception [55] architecture focuses on going wider instead of deeper to improve model performance. As the size of features in an image may vary, a kernel size of 3x3 pixels is not always the best option. Small and large kernels are preferred for locally and globally distributed information, respectively. The Inception architecture uses differently sized filters in the same convolution module. Figure 2.14 shows the first version of the Inception module: Inception v1. The output of the module is the concatenation along the filter dimension of the outputs of each convolution and maximum pooling layer.

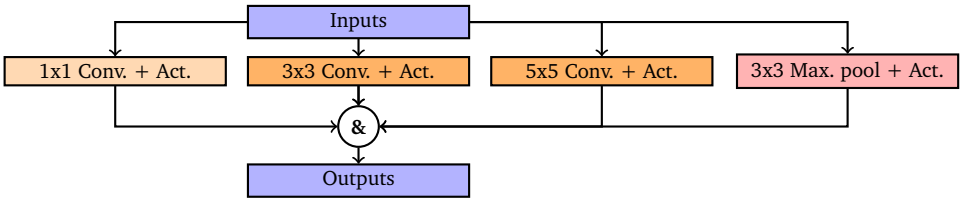


Figure 2.14: The Inception v1 module.

Due to the large 5x5 kernel in the Inception module, the number of model parameters and computational cost increase rapidly as modules are stacked. A

reduced version of the Inception v1 module solves this problem by introducing pointwise convolutions for dimensionality reduction (see Figure 2.15).

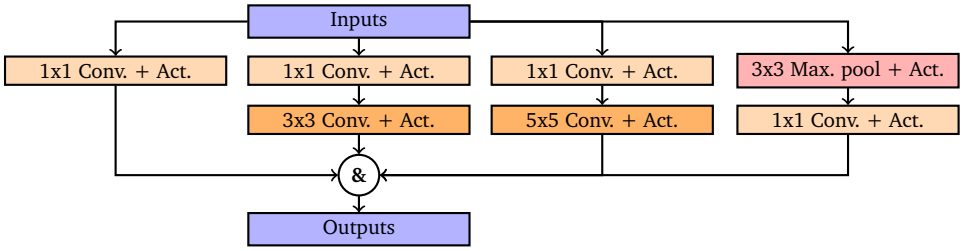


Figure 2.15: The Inception v1 module with dimension reduction by pointwise convolutions.

Inception v2 [56] replace the 5x5 convolution layer with two 3x3 convolution layer in series. This lowers the computational cost as larger kernels are more computationally expensive. Another variation of the Inception v2 module further decrease the computational cost by factorizing the 3x3 convolutions. As Figure 2.16 shows, one 3x3 convolution layer may be factorized into one 1x3 convolution layer followed by a 3x1 convolution layer.

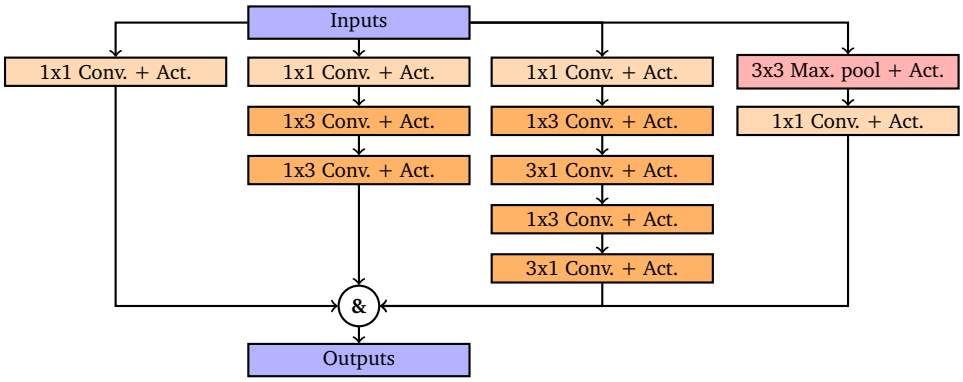


Figure 2.16: The Inception v2 module with factorization.

2.4.4 Autoencoders

An autoencoder (Autoencoder (AE)) is a DNN that is designed to learn a lower-dimensional representation of a higher-dimensional data. AEs map an input to an output by forcing the data through a compressed representation. This way the network preserves the most important features. By compressing the data, the risk of data memorization and overfitting is reduced.

AEs consist of three parts: an encoder, a bottleneck, and a decoder.

The task of the encoder is to compress the input data to a lower-dimensional representation. For CNNs, this corresponds to a feature map of a lower resolution. Usually the encoder is built of convolution modules followed by pooling layers. In this context the word deep does not only refer to the amount of convolution layers, but also to the amount of downsampling layers in the CNN. A model of depth 3 has 3 downsampling layers.

Followed by the encoder part is the bottleneck, containing the lower-dimensional representation of the data. The size of the bottleneck decides the degree of compression. As the compression type in AEs is lossy compression, the size of the bottleneck is critical for determining how much information is preserved. Reconstructed images from an AE may have disrupted structural features [14].

The bottleneck is followed by the encoder, which is essentially the opposite of the encoder. In most cases it is designed to decompress the data back to the dimensions of the input. Usually the decoder consists of convolution modules and upsampling layers.

If the dimensionality of the input data is large, the AE must be deep in order to add a bottleneck of dense layers. Otherwise the amount of model parameters will be too large when connecting the last convolution layer to a dense layer with flattening. A way of overcoming this issue is the use of a Fully Convolutional Neural Networks (FCNNs) that do not contain dense layers [57]. This allows the input of the network to be of any arbitrary size. Figure 2.17 shows an example of a VGG-based fully convolutional AE of depth 2.

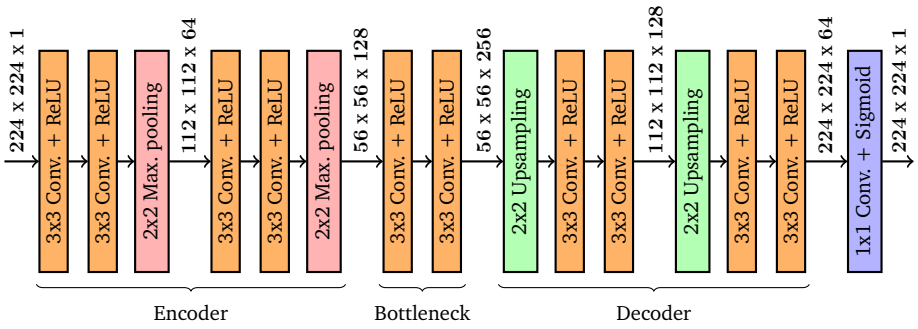


Figure 2.17: A fully convolutional autoencoder of depth 2 based on the VGG model.

AEs are used in various applications. They are used for anomaly detection, denoising, pre-training, and are the basis of GANs and other generative DNNs.

2.4.5 U-Net

A U-Net is a variation of the standard AE that introduces skip connections to overcome the loss of details due to the lossy compression of AEs [58]. The U-

Net architecture was designed for image-to-image segmentation problems but has proved useful in a wide range of applications.

The skip connections of the U-Net copy information from the encoder to the decoder. Adding an alternative path to the one through the bottleneck may preserve detailed information from the input to the output.

The name of the architecture comes from its U-looking shape (see Figure 2.18).

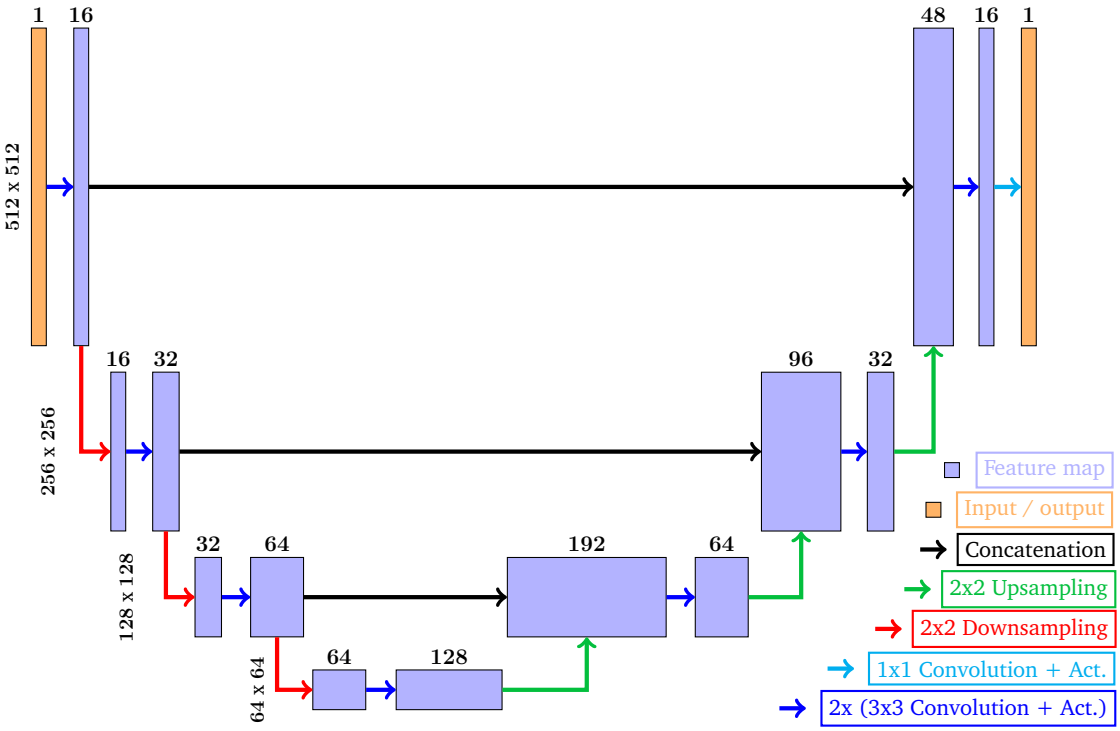


Figure 2.18: A U-Net with a depth of 3. The initial convolution module has 16 filters, doubling the number of filters for each depth increment. The image dimensions are halved after each downsampling. Feature maps from the encoder and the bottleneck are concatenated in the filter dimension.

Many variations of the U-Net architecture exist. Two examples are Residual U-Net and Inception U-Net, which replace the default convolution modules with residual modules and inception modules, respectively.

Chapter 3

Method description and implementation

This chapter describes the method and implementation used to learn and generalize the native signal processing of the GE Vivid E95 cardiovascular ultrasound system using deep learning. The chapter presents the dataset used, the data pipeline, how the ML model was developed, and how it was modified to perform despeckling. Lastly, the chapter describes how the obtained results were evaluated.

Figure 3.1 shows an overview of the Vivid E95 native signal processing chain.

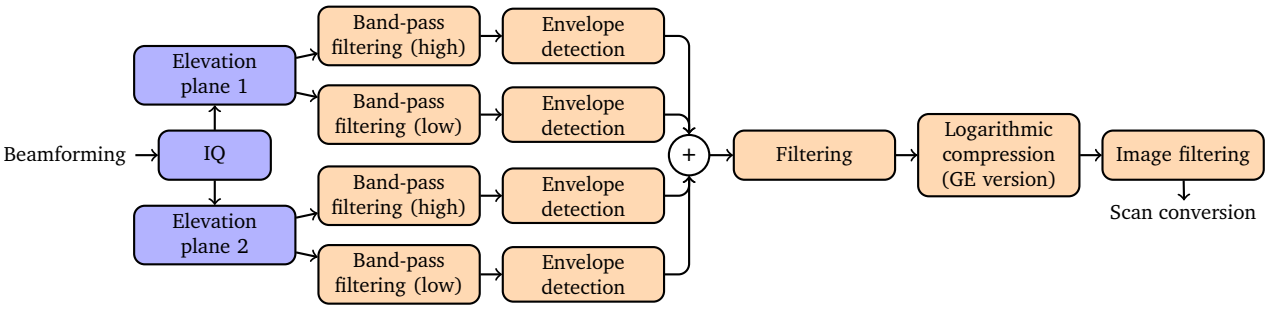


Figure 3.1: A high-level overview of the native B-mode ultrasound signal processing chain in the GE Vivid E95 scanner when using the GE 4Vc-D 4D probe.

When using the GE 4Vc-D 4D probe, the beamformed IQ data consists of two warped elevation planes. (See Haugen et al. (2015) [25] for more details about the warping technique.) Each plane is band-pass filtered in two overlapping frequency spectrum bands. Time frequency compensation is used, meaning the center frequency of the band-pass filter is changed to account for the frequency shift caused by attenuation. Each of the four filtered signals are then envelope detected and summed to one compounded envelope. Next, additional filtering is applied to the compounded envelope. Further, the envelope is log com-

pressed. GE uses a different logarithmic compression than $20 \cdot \log_{10}(\cdot)$. Lastly, the resulting beamspace image is filtered, and scan converted.

In short, the workflow consisted of four parts: development of a data pipeline, development of a DNN, training of the DNN, and evaluation of results. Optimizing the data pipeline and DNN was a recursive process (see Figure 3.2). The results were evaluated, and the data pipeline and the DNN were modified to refine the performance.

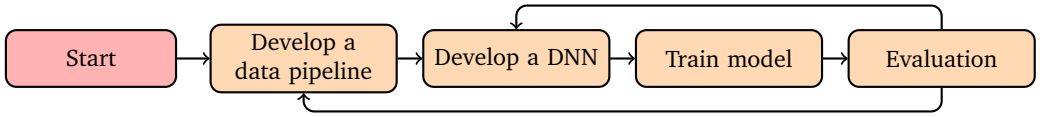


Figure 3.2: A flow chart of the recursive workflow consisting of developing a pipeline, developing a DNN, training the DNN, and evaluating the results.

The work was executed in the PYTHON programming language with the TENSORFLOW [59] and KERAS [60] high-level APIs. To speed up the process, a LINUX server with an NVIDIA Quadro P5000 GPU with 16 gigabytes of RAM was used to train the ML models.

3.1 Dataset

Figure 3.3 shows an example of scan converted images created with data from the dataset.

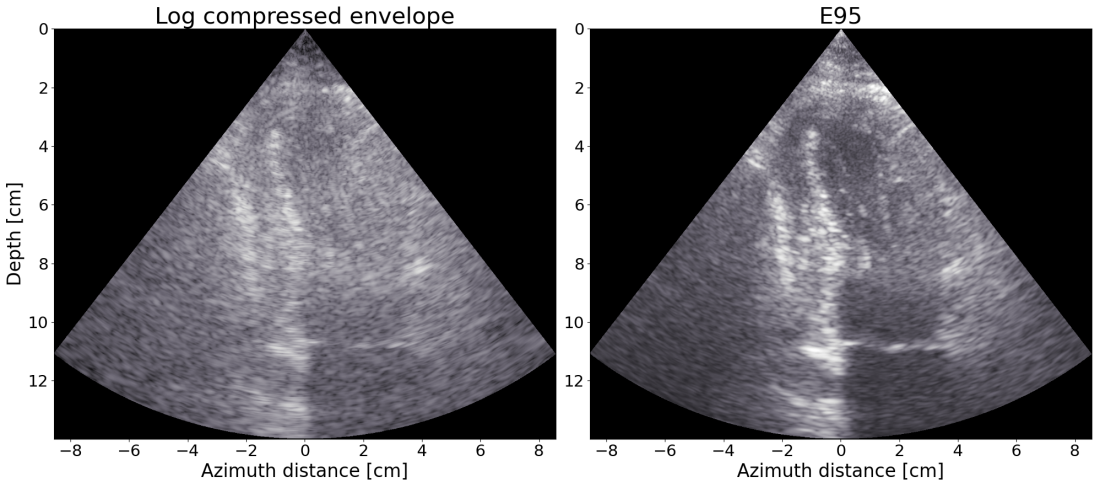


Figure 3.3: A $20 \cdot \log_{10}(\cdot)$ log compressed compounded envelope of unfiltered IQ data from two elevation planes (left) and its corresponding GE Vivid E95 processed image (right). View: A2C.

The data was already collected using the Vivid E95 cardiovascular ultrasound system with software version VIVID E95:203.75.2 from GE Vingmed Ultrasound. Data was acquired using a GE 4Vc-D cardiac sector probe and the Cardiac_E application with factory settings. The transmit center frequency was 1.4 MHz, and the receive center frequency was 2.8 MHz. The spatial extent in the depth and azimuth directions varied throughout the dataset.

The dataset consisted of 116 DICOM [61] files and files with corresponding unfiltered beamformed IQ data from 22 patients. Each file contained 64 frames, and the total number of unique images was $7,424$. All frames were split into separate files so that individual frames from different DICOM files could be shuffled. The dataset included images with the most common cardiac views: PLAX, PSAC, A4C, A2C, and ALAX. (See Østvik et al. 2019 [62] for more information about cardiac views.)

80% of the dataset was used for training, 10% for validation, and the remaining 10% was used as a test dataset. The number of images was $5,824$, 832, and 704 for the training, the validation, and the test dataset, respectively. Splitting was done at the patient level, such that data from a specific patient only belonged to one of the three datasets. The validation and test dataset had data from 2 patients each, and the training dataset contained the data from the remaining 18 patients. This was done to ensure that validation and testing were performed on unseen cases.

3.2 Data pipeline

Due to the large dataset size, a data pipeline was used to stream data when needed for training and evaluation. This way, RAM overflow issues were avoided. Figure 3.4 shows the data pipeline used for the training, validation, and test dataset.

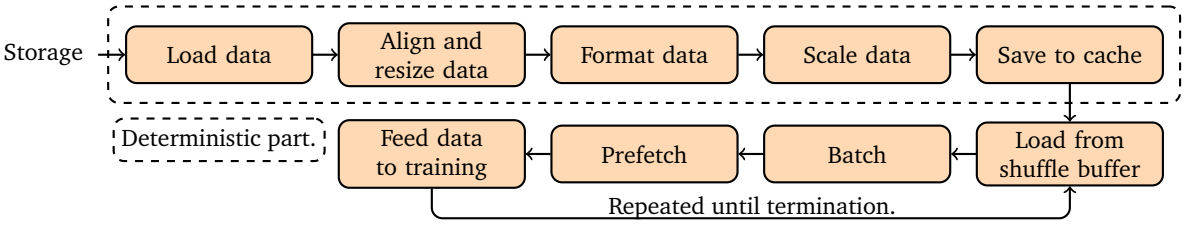


Figure 3.4: The data pipeline used for the training, validation, and test dataset. It consisted of one deterministic part for pre-processing (surrounded by the dotted rectangle) and one part of a stochastic behavior. By caching the pre-processed dataset, the deterministic part was only executed once.

The first half of the data pipeline consisted of deterministic steps for reading and pre-processing the dataset. As repeating these steps on every iteration

was unnecessary, caching was done by saving the pre-processed data to the hard drive. When the remaining stochastic part of the pipeline was executed, pre-processed data was loaded from the cache files. The randomness of the remaining of the data pipeline was introduced by shuffling.

3.2.1 Pre-processing

The spatial extent of the GE Vivid E95 processed images was slightly smaller than that of the IQ data. As a result, the network learned to stretch the IQ data in both the range and the azimuth direction. To overcome this problem, the IQ data was interpolated and cropped to overlap with the spatial extent of the E95 images. This was done by interpolation between two grids for each elevation plane separately. Both bilinear and nearest-neighbor interpolation were tested.

As a result of the variable depth and angular sector, the number of samples in the height and width dimensions varied. When training the DNNs, the inputs and outputs needed to be of a fixed shape. Three approaches were proposed to solve this problem: padding, cropping, or stretching. The solution used was stretching. The grids used in the grid interpolation were upsampled to a desired amount of samples. The grids were upsampled with spline interpolation of order 3 to 768 samples in the range and 512 samples in the azimuth direction. These numbers were chosen to be higher than the maximum amount of samples in each of the two dimensions throughout the dataset. This way, downsampling, and potential aliasing problems were avoided. To obtain outputs of the same dimensions as the inputs, the GE Vivid E95 processed images were upsampled to 768x512 pixels using bilinear interpolation.

Further, the complex IQ data was split into two channels, depending on the type of complex number representation used. When using the Cartesian format, the two channels contained the real and imaginary parts of the data. The data was split into magnitude and phase components when representing the complex iq data in polar form.

After the interpolation step, scaling was done to weigh the features equally and avoid the exploding gradient problem. Two types of scaling were tested: local and global maximum absolute scaling. The local scaling was performed by dividing each data sample by its maximum absolute value. Global scaling was done by scaling each data sample relative to the maximum absolute value of the entire dataset. The global maximum absolute value for the E95 processed images was 255 as they were stored using 8 bits. The IQ data was scaled by the maximum absolute value of the entire dataset.

Depending on the type of complex number representation used, the outcome of the scaling was different. When using the Cartesian IQ representation, the real and the imaginary part were scaled between -1 and 1. In polar form, the phase and the magnitude were scaled between -1 and 1, and 0 and 1,

respectively. The E95 images were always scaled between 0 and 1.

Before caching the pre-processed data, the elevation plane dimension and the complex number representation dimension were concatenated in the filter dimension. This was done to fit the accepted format of the two-dimension convolution layer in KERAS.

3.2.2 Shuffling, batching, and prefetching

When reading data from the cache, the data was loaded into a shuffle buffer with a size of 100 IQ and B-mode image pairs. The training and validation datasets were also shuffled before reading and pre-processing the data.

Before feeding the data to the training algorithm, the data was batched and prefetched. Due to the high resolution of the images, a batch size of 8 was chosen. This allowed using more filters in the convolution layers without running into memory overflow problems.

Prefetching was added to the end of the data pipeline to increase the speed of which data could be streamed into the training algorithm. By using the autotune option in TENSORFLOW, the level of parallelism was optimized automatically.

3.3 Development of a ML model for Vivid E95 native signal processing

Before attempting to do despeckling, the goal was to learn and generalize the Vivid E95 native B-mode signal processing as well as possible.

Hundreds of CNNs models were tested. The architectures used were variations of the standard AE, U-Net, Residual U-Net, and Inception U-Net. Batch normalization was not used due to the use of residual connections and skip connections. The networks were built using the functional API in KERAS.

To use a bottleneck of dense layers, the depth of the autoencoder would have needed to be very high to sufficiently downsample the data. As this would have resulted in a large and slow network of millions of parameters, FCNNs were used instead.

The best performing model was obtained by making one change at a time to the U-Net in Figure 3.5. The initial model had 16 filters, a depth of 3, used hyperbolic tangent as the activation function, and did downsampling and upsampling by strided convolution and transposed convolution, respectively. It used an SSIM-based loss function.

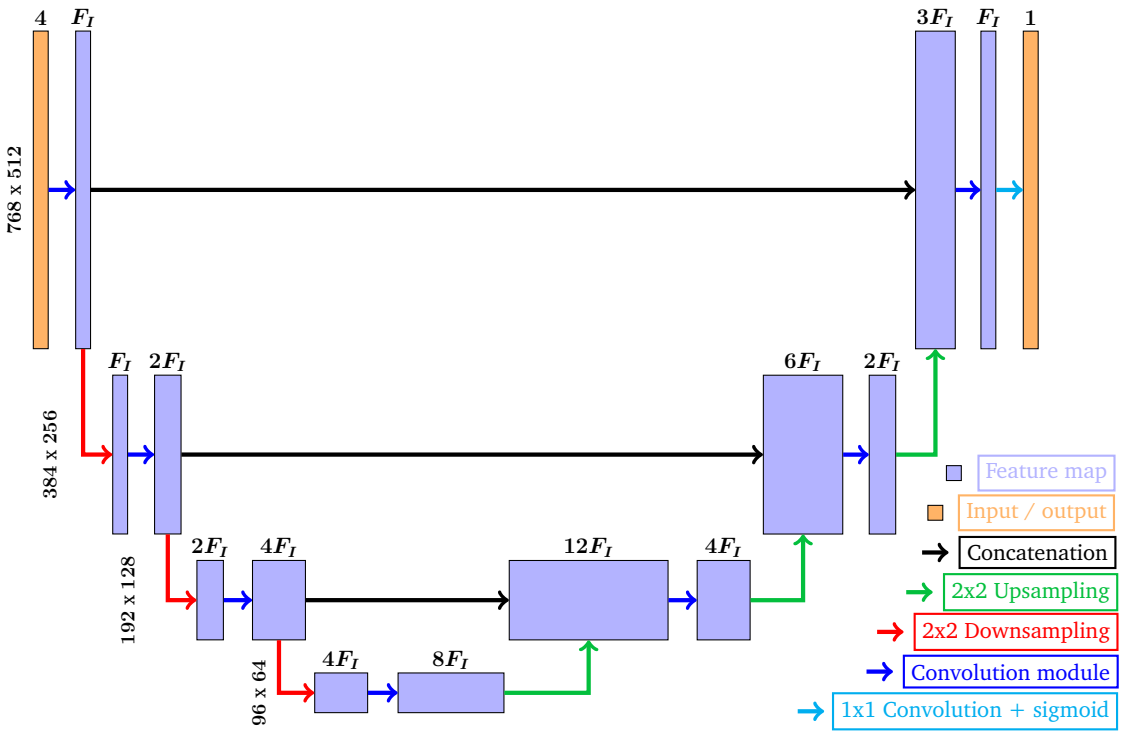


Figure 3.5: A U-Net with a depth of 3 and F_I filters.

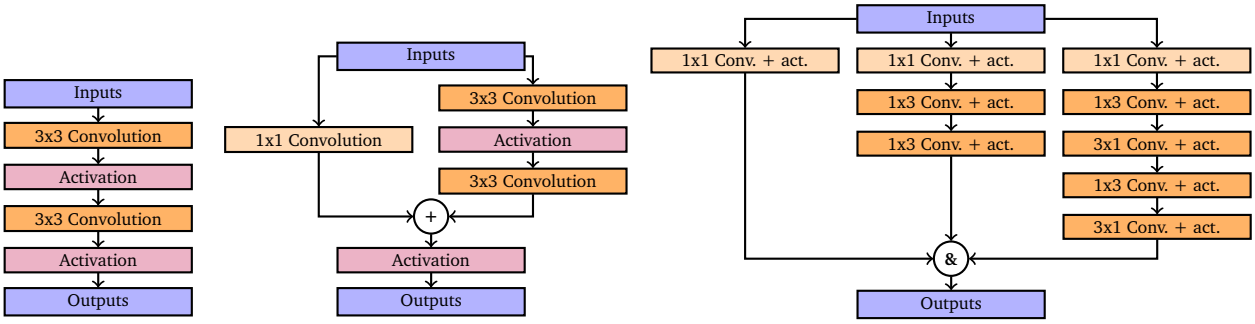
F_I was the number of filters used in the initial convolution module. This number was doubled after each depth increment, inspired by the VGG architecture. Optimization of the model capacity was done using a grid search. Combinations of the numbers of initial filters F_I and depths were tested to find a model of sufficient capacity for the given problem.

After optimizing the model capacity, different downsampling and upsampling methods were tested. Downsampling was performed using maximum pooling or strided convolution. Upsampling was done with bilinear interpolation or strided transposed convolution. The downsampling convolution layers used 3x3 kernels and a stride of 2. A 2x2 pool size was used for the maximum pooling. This way, the image height and width were halved after each downsampling layer.

The linear, sigmoid, hyperbolic tangent and ReLU activation functions were tested in the input layer and hidden layers. Sigmoid was used as the output activation function to avoid predicted values outside the range of available scaled pixel values ($[0, 1]$).

Figure 3.6 shows the three different convolution modules tested for the U-Net, the Residual U-Net, and the Inception U-Net, respectively. The default module used for the U-Net had two convolution layers with 3x3 kernels and two activation functions. The residual module had pointwise convolution on the residual connection. This was done to get outputs from the two paths of an

equal number of channels. The Inception U-Net consisted of factorized Inception v2 modules without the 3x3 maximum pooling layer (see Figure 2.16).



(a) Default module.

(b) Residual module with 1x1 convolution.

(c) Factorized Inception v2 module without maximum pooling.

Figure 3.6: Three different convolutions modules.

Three loss functions were tested: SSIM, PSNR, and the product of the two. Since the optimization algorithm was designed to minimize the loss function, some modifications had to be made to the loss functions. The SSIM loss was formulated as $(1 - \mathrm{SSIM})$, the PSNR loss as $-\mathrm{PSNR}$, and the combined loss as $-\mathrm{PSNR} \cdot \mathrm{SSIM}$. The SSIM metric was calculated with default settings using a 11x11 Gaussian filter with a width (or sigma value) of 1.5.

All models were trained with mini-batch gradient descent using the Adam optimizer with a learning rate of 0.001 and default settings: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$.

The number of training epochs was set to 200. Early stopping regularization was used to avoid overtraining by monitoring the validation loss. The patience was equal to 5 epochs. The best performing model was also trained with a different initialization seed, and an increased patience of 10 epochs to further improve its performance. Dropout regularization was not used as the learning curves showed no generalization gaps of significance.

3.4 Despeckling

The best performing ML model for learning and generalizing the GE Vivid E95 native signal processing was modified to perform despeckling.

As the M5Sc-D probe does not support elevation compounding, it produces images with more speckle noise than when using the 4Vc-D probe. The goal was to get the despeckling effect of elevation compounding without having data from two elevation planes.

This was achieved by training the ML model with 4Vc-D data to map IQ data from one elevation plane to elevation compounded E95 images.

A small dataset acquired with a GE M5Sc-D probe was used to evaluate the performance of the ML model. It consisted of 64 frames from one scan of one patient. This data was acquired using different acquisition parameters than with the 4Vc-D probe. The transmit and receive center frequencies were 1.7 MHz and 3.4 MHz, respectively.

3.5 Evaluation of performance

Since the objective was to learn and generalize the Vivid E95 native signal processing, the aim was to find a data pipeline and a ML model that maximized the SSIM and the PSNR. Both metrics were calculated in beamspace. Visual comparisons of images and histograms were made for the models that performed well quantitatively. Histograms were calculated of scan converted images after gain adjustments. The black background pixels values of the scan converted images were ignored when plotting the histograms. Predicted images were compared to their corresponding ground truth GE Vivid E95 processed images. This was necessary as it is difficult to quantify the degree of despeckling with a metric. Difference plots were used to better visualize the changes between the predicted and E95 images. When plotting difference plots, the difference was calculated in beamspace before scan conversion. Scan conversion was implemented using the FAST API [63], [64].

Histogram matching [65] was done to account for small pixel intensity differences caused by the model. The pixel intensities of the images were changed such that their histograms (meaning the distributions of their pixel intensities) matched the histograms of the E95 reference images. In doing so, the contrast and illumination of the predicted images were adjusted to match the E95 images. Histogram matching was performed in beamspace.

Visual comparisons were performed on image cases of data from the 4Vc-D and the M5Sc-D probe. Since the IQ data from the M5Sc-D probe only contained one elevation plane, the plane was duplicated to fit the input shape of the E95 replication model.

The images were displayed with a full dynamic range from 0 to 255. The overall gain was increased to inspect how the ML model affected weak scatterers. For most case examples, the gain was increased to the limit of saturation. Still, in some cases, the gain had to be increased past saturation to allow for visualization of weak scatterers. The same gain adjustments were made to the E95 image, the predicted image, and the histogram matched predicted image.

The model could not be repeated multiple times as the CNN had a different number of input and output channels.

Chapter 4

Results and observations

This chapter covers the most important results and observations. The chapter starts by presenting quantitative results when using different types of pre-processing and ML models. Further, it shows examples of predicted images using the best performing ML model. Lastly, image examples predicted by the despeckling model are presented.

4.1 Learning and generalizing the Vivid E95 native signal processing

Section 4.1 presents the results of the work related to learning and generalizing the Vivid E95 native signal processing.

4.1.1 Pre-processing

Pre-processing of the IQ data was critical for the performance of the ML model.

The use of bilinear interpolation between the two grids significantly increased the SSIM and PSNR values compared to when using nearest-neighbor interpolation.

Figure 4.1 shows the learning curves of the PSNR value for the training and the validation dataset when using different types of scaling. The model trained with data scaled by global maximum absolute scaling outperformed the model trained with local maximum absolute scaled data. Global scaling resulted in the highest PSNR value. The PSNR learning curves for the validation dataset had fluctuations when using both scaling methods.

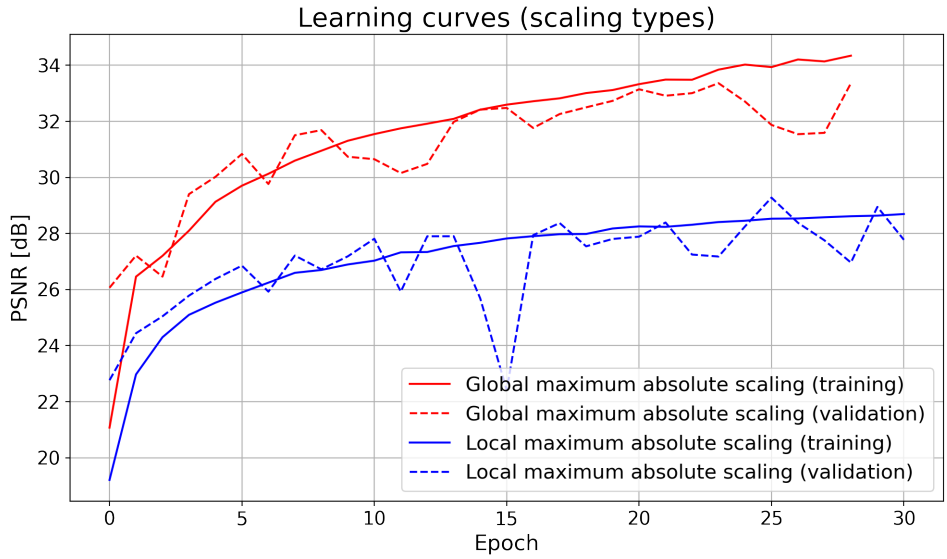


Figure 4.1: PSNR learning curves for different scaling types.

It was easier to train the network with IQ data in Cartesian form than in polar form. Figure 4.2 shows that when using the Cartesian complex number representation the SSIM value got higher and the training converged faster to a local minimum. Training with IQ data in polar form caused a significant generalization gap.

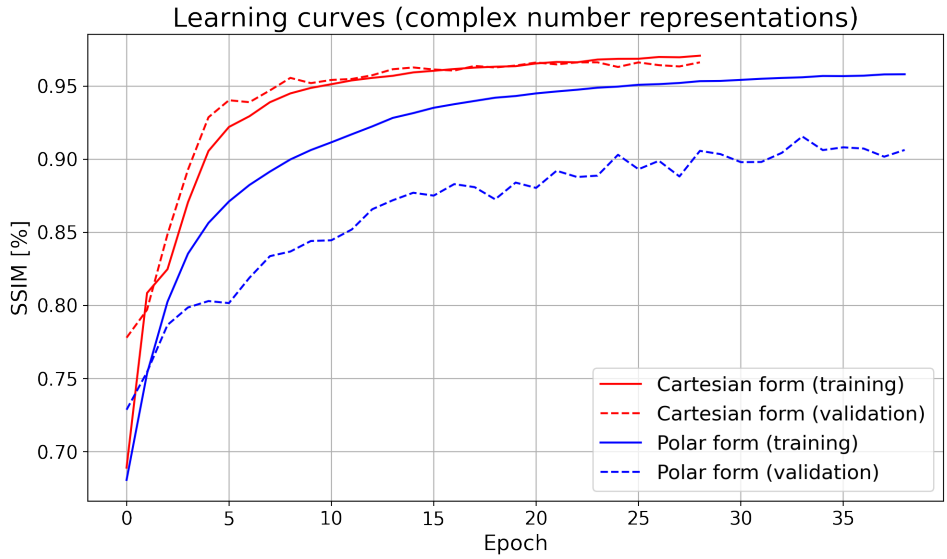


Figure 4.2: SSIM learning curves for different complex number representations.

4.1.2 ML model

All the remaining ML models were trained with IQ data in Cartesian form. The data was scaled with global maximum absolute scaling.

Table 4.1 shows that the U-Net with 16 filters, a depth of 3, and $584,433$ model parameters gave the highest SSIM and PSNR. Further, its mean SSIM value had the lowest uncertainty.

The three models with the lowest capacities, with less than $36,000$ model parameters, had significantly lower performances.

Table 4.1: SSIM and PSNR values for combinations of filters and depths.

Filters	Depth	Parameters	SSIM	PSNR
8	1	$7,865$	$(87.4 \pm 3.20)\%$	(21.6 ± 0.22) dB
8	2	$35,641$	$(91.6 \pm 2.50)\%$	(28.2 ± 0.45) dB
8	3	$146,489$	$(96.7 \pm 0.45)\%$	(31.5 ± 0.97) dB
8	4	$589,369$	$(96.9 \pm 0.49)\%$	(33.4 ± 0.82) dB
8	5	$2,359,865$	$(96.3 \pm 0.55)\%$	(31.6 ± 0.40) dB
16	1	$30,705$	$(94.2 \pm 1.40)\%$	(25.8 ± 0.27) dB
16	2	$141,553$	$(97.1 \pm 0.45)\%$	(30.7 ± 0.76) dB
16	3	$584,433$	$(97.9 \pm 0.34)\%$	(33.8 ± 0.83) dB
16	4	$2,354,929$	$(95.8 \pm 0.73)\%$	(28.7 ± 0.48) dB
16	5	$9,434,865$	$(96.1 \pm 0.86)\%$	(28.5 ± 0.59) dB

Figure 4.3 shows that the model of depth 3 converged the fastest to a local minimum among the models with 16 filters. The SSIM learning curves showed no generalization gaps.

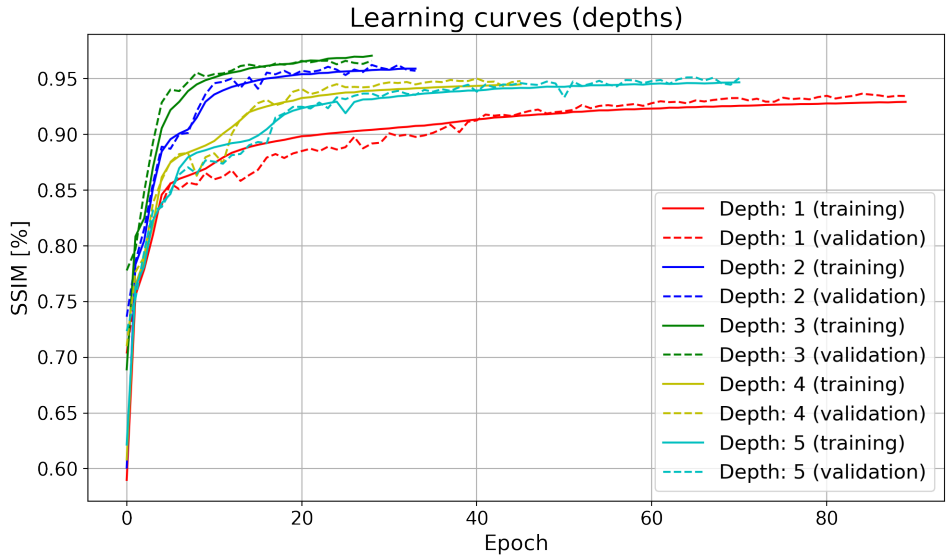


Figure 4.3: SSIM learning curves for different depths with 16 filters.

Models that used strided transposed convolution for upsampling outperformed the models that used bilinear interpolation. Table 4.2 showed that models with downsampling by maximum pooling gave higher PSNR values than when using strided convolution. The highest SSIM values were achieved by downsampling and upsampling with strided convolution and strided transposed convolution, respectively. This last option was preferred as it gave SSIM and PSNR values with lower standard deviations than the other combinations.

Table 4.2: SSIM and PSNR values for different combinations of downsampling and upsampling techniques.

Downsampling	Upsampling	Parameters	SSIM	PSNR
Maximum pooling	Bilinear interpolation	$487,441$	$(94.1 \pm 1.20)\%$	(27.1 ± 0.40) dB
Maximum pooling	Strided transposed convolution	$535,937$	$(97.7 \pm 0.64)\%$	(34.4 ± 0.97) dB
Strided convolution	Bilinear interpolation	$535,937$	$(92.2 \pm 2.61\%$	(26.6 ± 0.38) dB
Strided convolution	Strided transposed convolution	$584,433$	$(97.9 \pm 0.39)\%$	(33.8 ± 0.83) dB

Table 4.3 shows that hyperbolic tangent and ReLU were the activation functions in the input layer and hidden layers that gave the best performance. The mean SSIM and PSNR were slightly higher when using the ReLU activation function. Further, the hyperbolic tangent activation function gave a lower standard deviation for the SSIM metric.

Table 4.3: SSIM and PSNR values for different activation functions in the input layer and hidden layers.

Activation	SSIM	PSNR
Linear	$(0.284 \pm 0.023)\%$	(5.07 ± 0.21) dB
Sigmoid	$(50.9 \pm 3.10)\%$	(16.2 ± 1.20) dB
tanh	$(97.9 \pm 0.39)\%$	(33.8 ± 0.83) dB
ReLU	$(98.0 \pm 0.59)\%$	(34.0 ± 0.86) dB

The standard AE without skip connections gave significantly lower SSIM and PSNR values than the U-Net variations (see Table 4.4). Further, the Residual U-Net performed better than U-Net. Inception U-Net outperformed the other U-Net variants but had more model parameters.

Table 4.4: SSIM and PSNR values for the standard AE and different U-Net variations.

Architecture	Parameters	SSIM	PSNR
Standard AE	$536,049$	$(50.8 \pm 3.30)\%$	(16.0 ± 1.20) dB
U-Net	$584,433$	$(97.9 \pm 0.39)\%$	(33.8 ± 0.83) dB
Residual U-Net	$606,353$	$(98.2 \pm 0.38)\%$	(34.0 ± 0.92) dB
Inception U-Net	$1,024,305$	$(98.5 \pm 0.25)\%$	(34.3 ± 0.81) dB

Table 4.5 shows that the SSIM-based loss function outperformed the PSNR and the combined loss function when comparing the SSIM values. It showed

significantly lower standard deviations for the two metrics. The PSNR based loss function yielded a slightly higher PSNR value, but a lower SSIM value.

Table 4.5: SSIM and PSNR values for different loss functions.

Loss function	SSIM	PSNR
$-\,\mathrm{PSNR}$	$(96.3 \pm 0.73)\%$	(34.0 ± 0.97) dB
$1 - \mathrm{SSIM}$	$(97.9 \pm 0.39)\%$	(33.8 ± 0.83) dB
$-(\mathrm{PSNR} \cdot \mathrm{SSIM})$	$(95.9 \pm 0.62)\%$	(33.5 ± 1.05) dB

Table 4.6 shows the best performing model. After training it with an early stopping patience of 5 epochs, evaluation of the test dataset gave an SSIM value of $(98.7 \pm 0.45)\%$ and a PSNR value of (35.5 ± 1.16) dB. When training the same model with an early stopping patience of 10 epochs and a different initialization seed, the results were an SSIM value of $(98.8 \pm 0.50)\%$ and a PSNR value of (35.8 ± 1.38) dB.

Table 4.6: The best performing ML model for replication of the Vivid E95 native signal processing chain. The early stopping patience was 10 epochs.

Loss function:	$1 - \mathrm{SSIM}$ (default settings)
Optimizer:	Adam (default settings)
Architecture:	Inception U-Net (factorized v2)
Depth:	3 (number of downsampling layers)
Filters:	16 (doubled for each depth increment)
Convolution:	3x3 kernel ($stride = 1$), padding type: 'same'
Downsampling:	3x3 convolution ($stride = 2$)
Upsampling:	3x3 transposed convolution ($stride = 2$)
Activation:	ReLU
Output activation:	Sigmoid
Parameters:	$1,024,305$
SSIM:	$(98.8 \pm 0.50)\%$
PSNR:	(35.8 ± 1.38) dB

4.1.3 Predicted image cases

The model from Table 4.6 was used to predict B-mode images with IQ data from the test dataset.

Figure 4.4 shows an example of a predicted image compared to the traditional $20 \cdot \log_{10}(\cdot)$ log compressed compounded envelope image, its corresponding E95 processed image, and a histogram matched (HM) version of the predicted image.

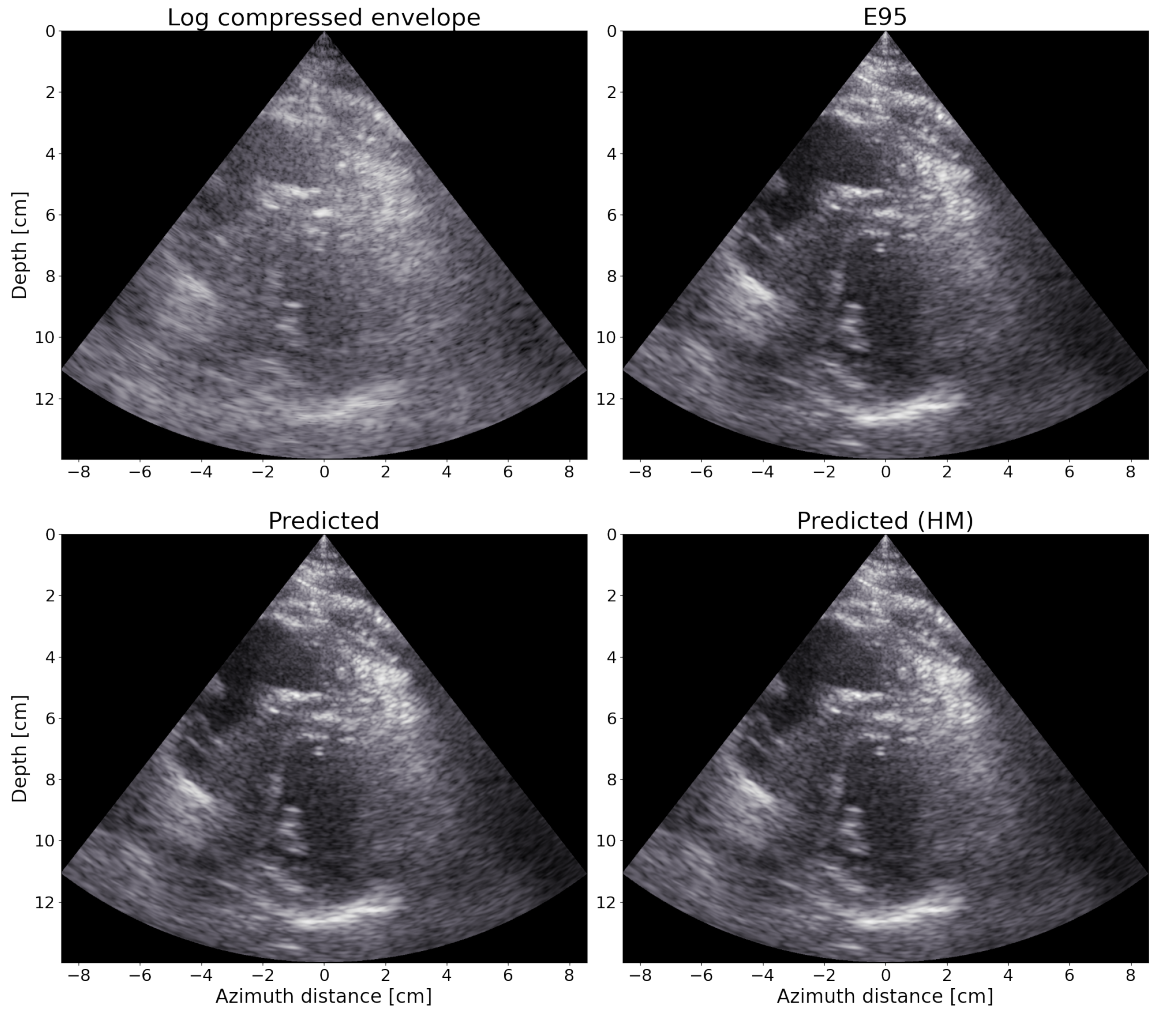


Figure 4.4: A log compressed compounded envelope, its corresponding E95 image, predicted image, and HM predicted image. Cardiac view: PSAX.

The following pages present seven case examples of predicted images for different cardiac views. Case 1 - 6 use data acquired with the 4Vc-D probe, and case 7 shows the performance of the model on data from the M5Sc-D probe.

Case 1 (4Vc-D probe)

Figure 4.5 shows an A2C cardiac view case example with $\mathrm{SSIM} = 99.1\%$.

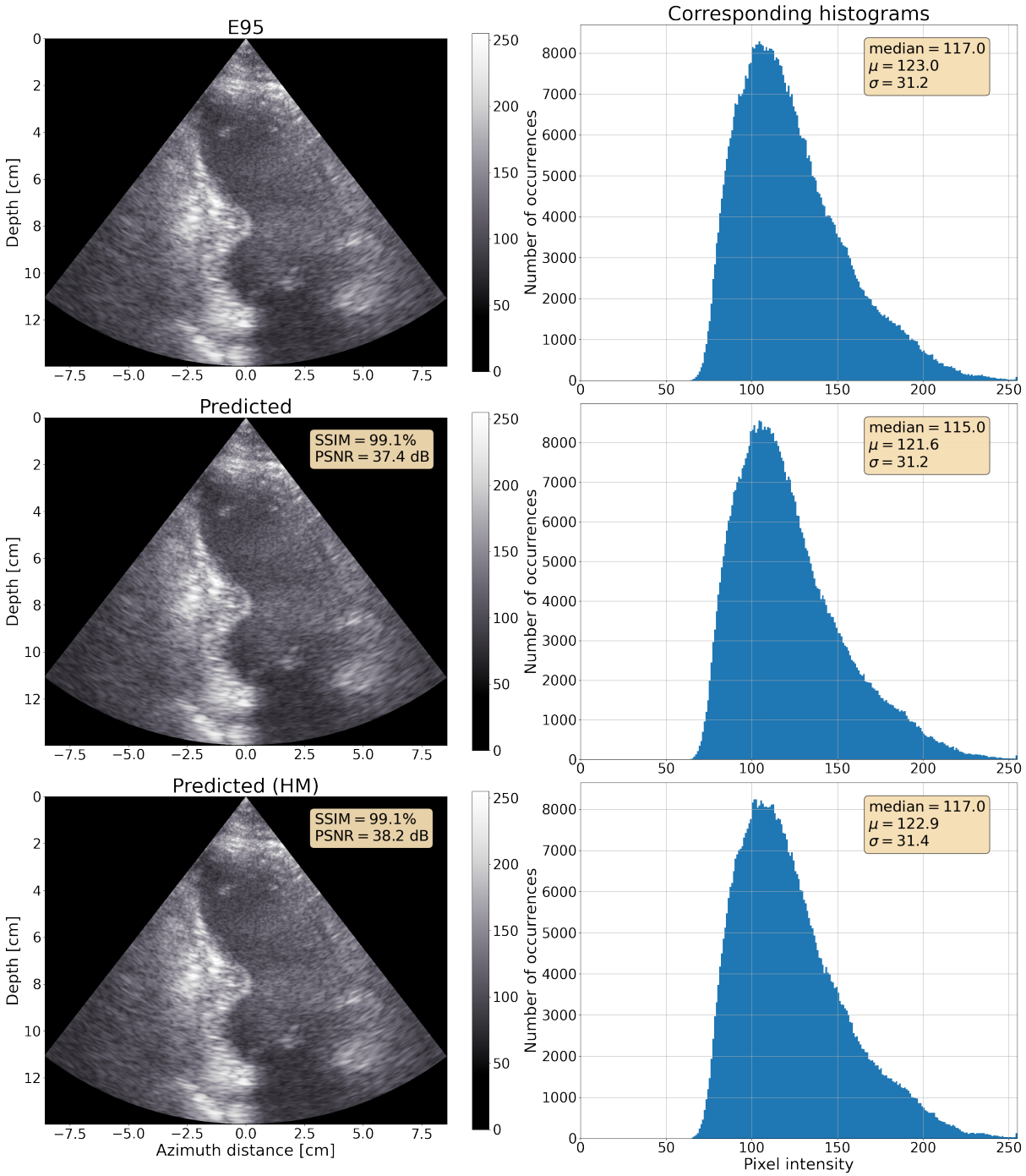


Figure 4.5: Case 1: An E95 image and its histogram compared to the corresponding predicted and HM predicted image. Cardiac view: A2C.

The histograms of the E95 image and the predicted image were almost identical, with the exception of a slightly lower mean value for the predicted image. Histogram matching compensated for this mean difference and increased the PSNR.

Figure 4.6 shows that the predicted image had increased brightness in sections of the left side of the image and in the center of the near-field (close to the transducer).

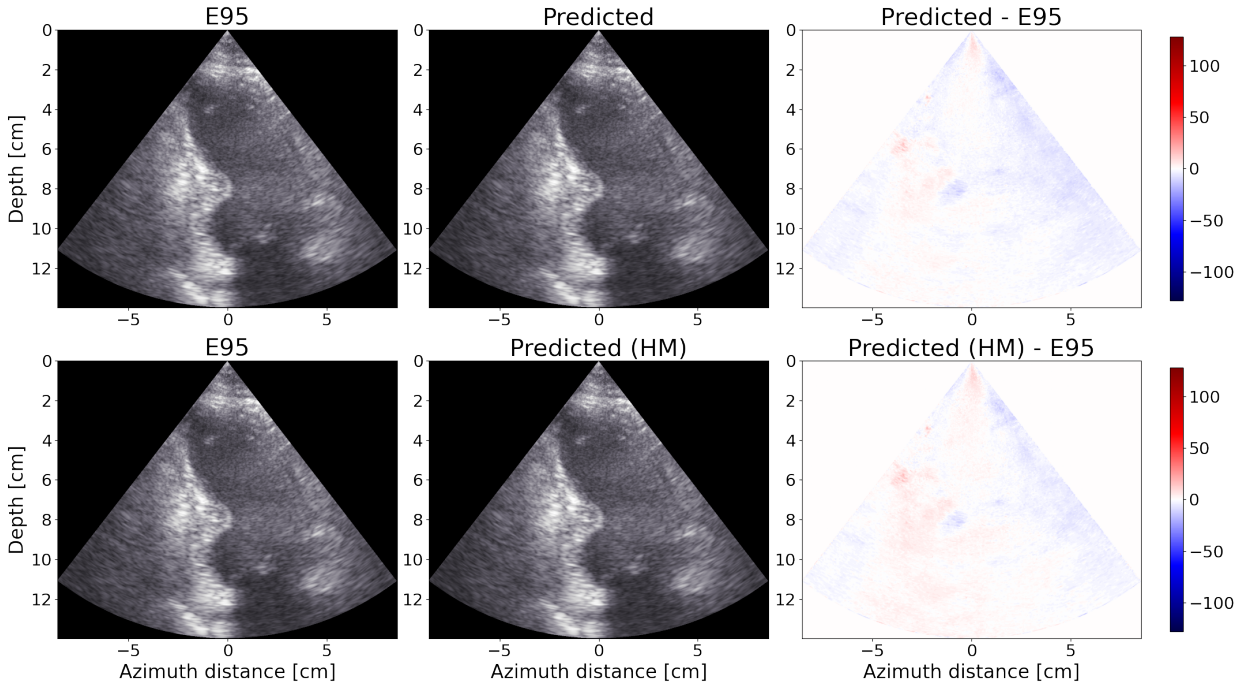


Figure 4.6: Case 1: The residual between an E95 image and its corresponding predicted image with and without histogram matching. Cardiac view: A2C.

Case 2 (4Vc-D probe)

Figure 4.7 shows a PLAX cardiac view case example with $\mathrm{SSIM} = 99.1\%$.

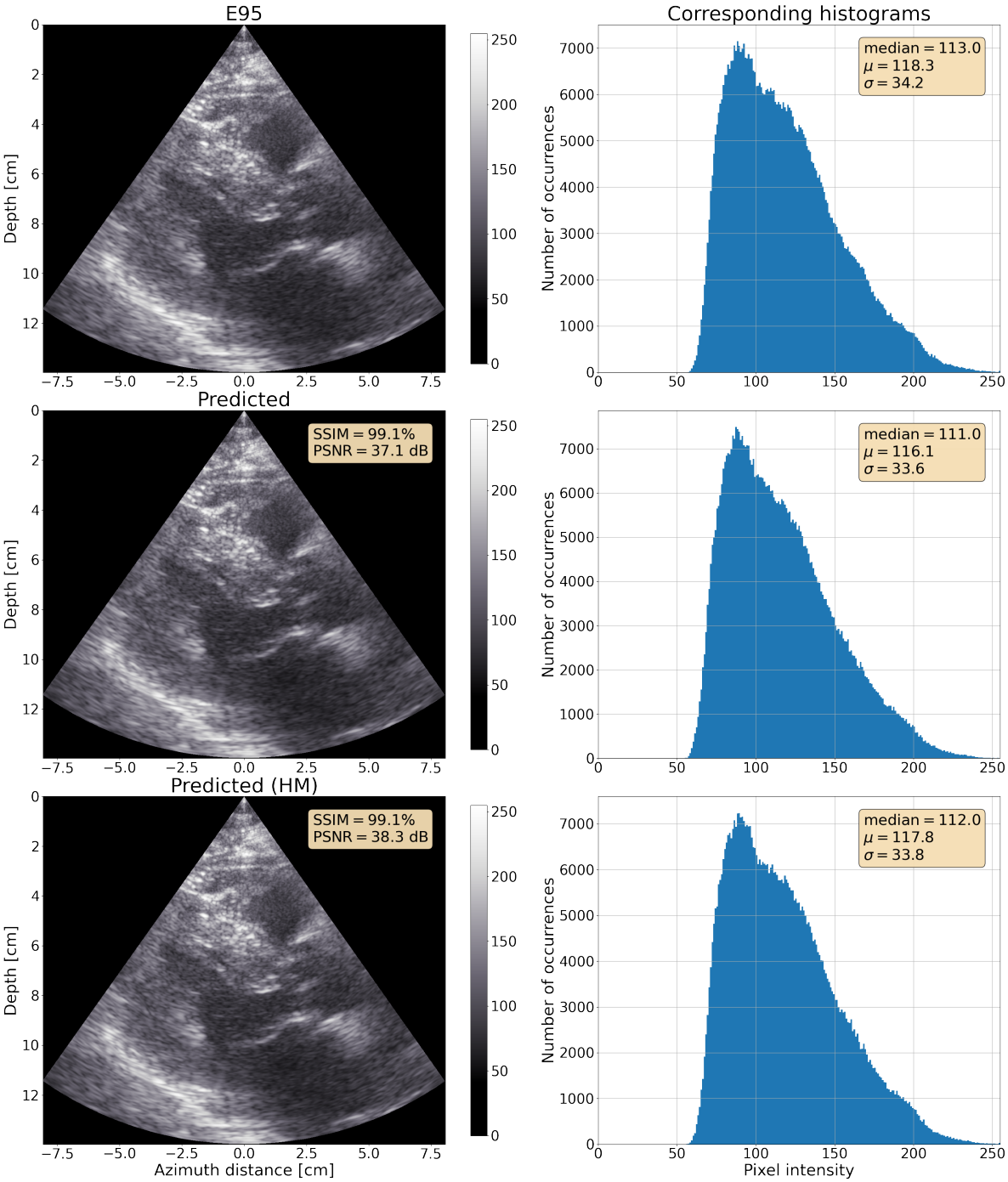


Figure 4.7: Case 2: An E95 image and its histogram compared to the corresponding predicted and HM predicted image. Cardiac view: PLAX.

Histogram matching did not make the histograms of the predicted image and the E95 image identical, but it reduced the differences between them.

Some sections of the predicted image had reduced pixel intensity compared to the E95 image (see Figure 4.8). The area in the azimuth center between the depths of 2 and 4 cm had increased brightness. Histogram matching expanded the spatial extent of this amplified area.

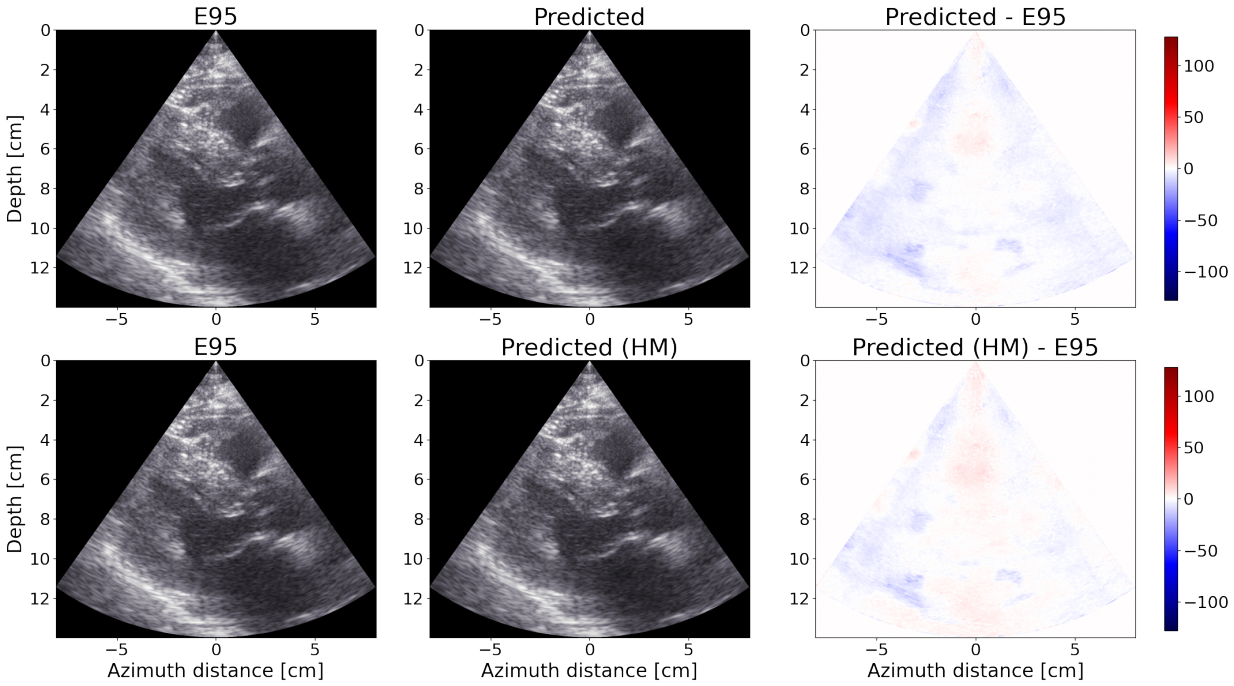


Figure 4.8: Case 2: The residual between an E95 image and its corresponding predicted image with and without histogram matching. Cardiac view: PLAX.

Case 3 (4Vc-D probe)

Figure 4.9 shows a PSAX case where the gain was increased past saturation.

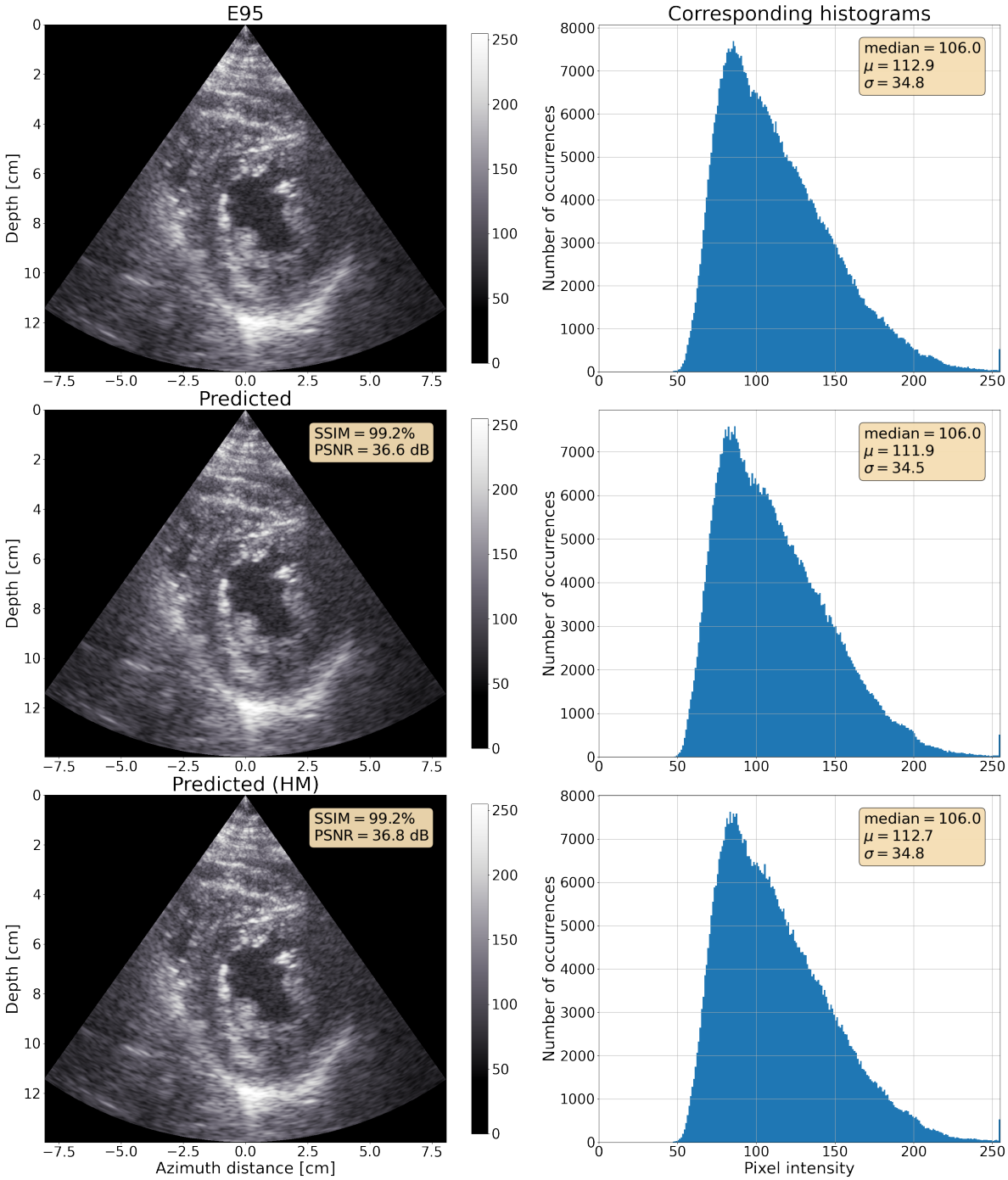


Figure 4.9: Case 3: An E95 image and its histogram compared to the corresponding predicted and HM predicted image. Cardiac view: PSAX.

The difference plots in Figure 4.10 reveal that some pixel intensities were suppressed at depth 2 cm on the left edge of the image. A similar behavior occurred on the right of the image between the depths of 10 to 14 cm. Larger areas around the image were slightly amplified.

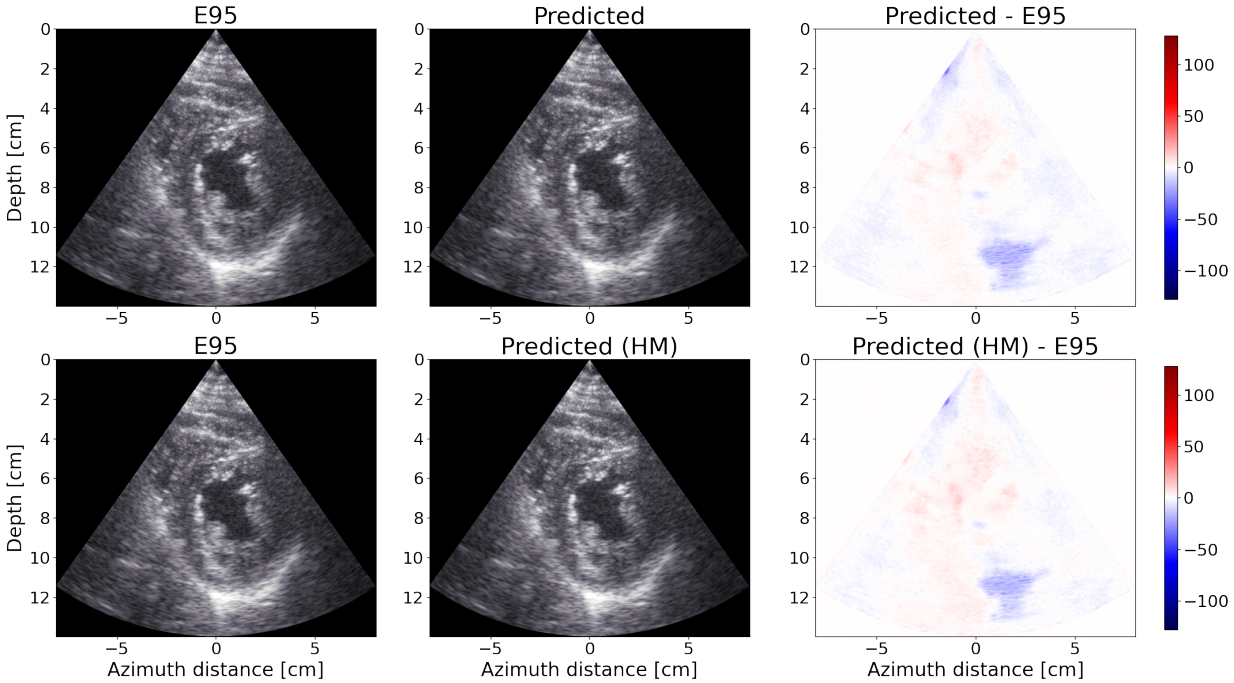


Figure 4.10: Case 3: The residual between an E95 image and its corresponding predicted image with and without histogram matching. Cardiac view: PSAX.

Case 4 (4Vc-D probe)

Figure 4.11 shows a PLAX case with gain increased past saturation.

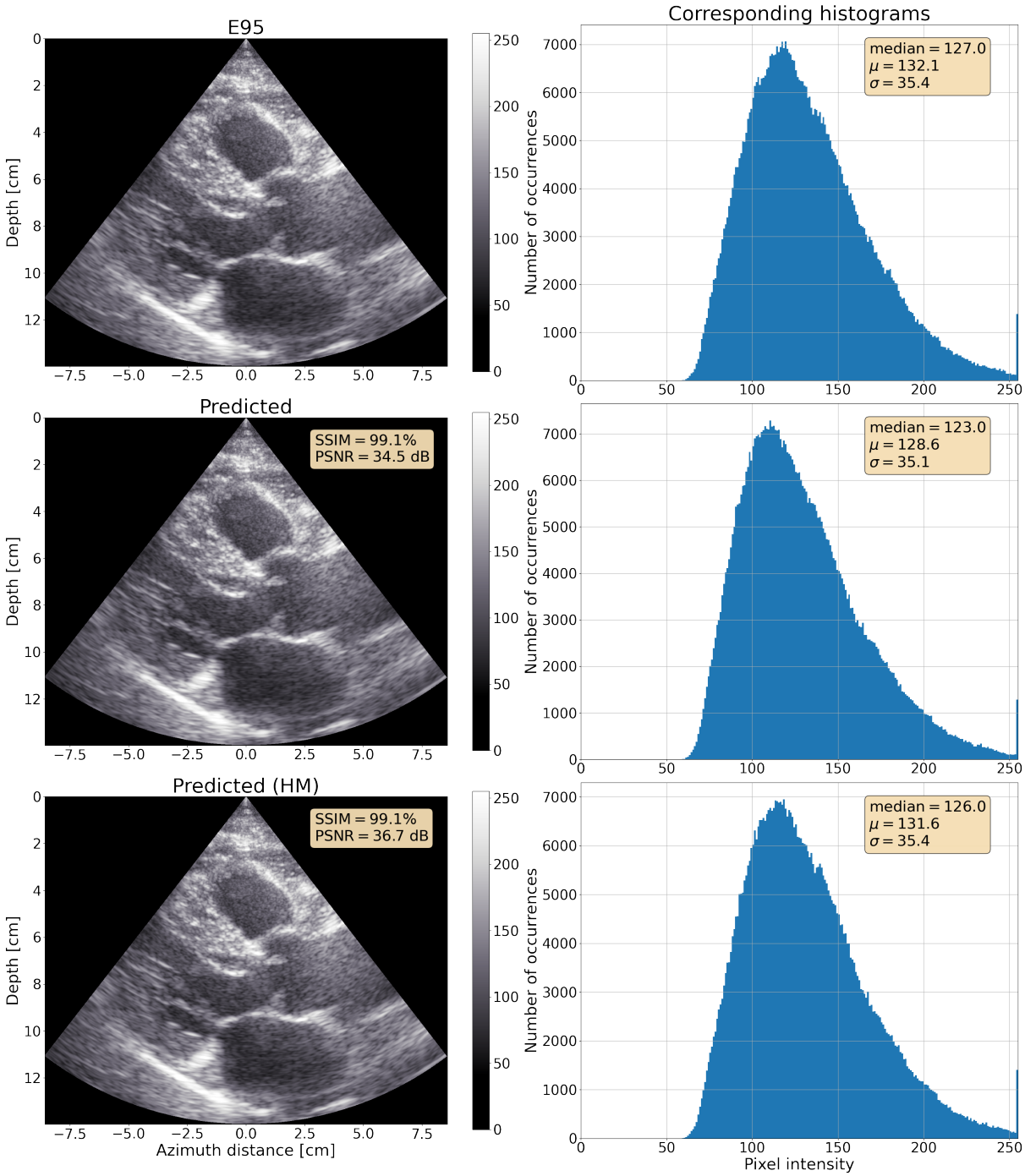


Figure 4.11: Case 4: An E95 image and its histogram compared to the corresponding predicted and HM predicted image. Cardiac view: PLAX.

Figure 4.12 shows a large suppressed area on the structure in the bottom right corner. Some smaller areas were suppressed around the predicted image in what appeared to be random locations. In the histogram matched image large areas in the image were amplified.

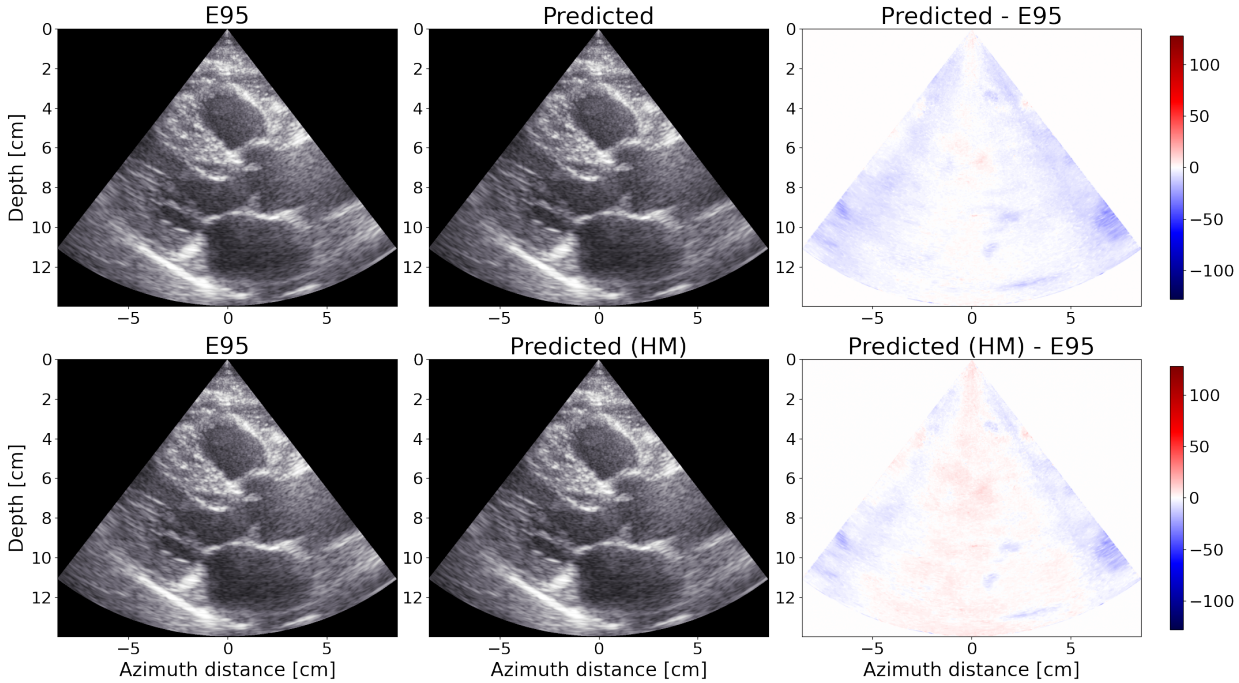


Figure 4.12: Case 4: The residual between an E95 image and its corresponding predicted image with and without histogram matching. Cardiac view: PLAX.

Case 5 (4Vc-D probe)

Figure 4.13 shows another case example with an A2C cardiac view.

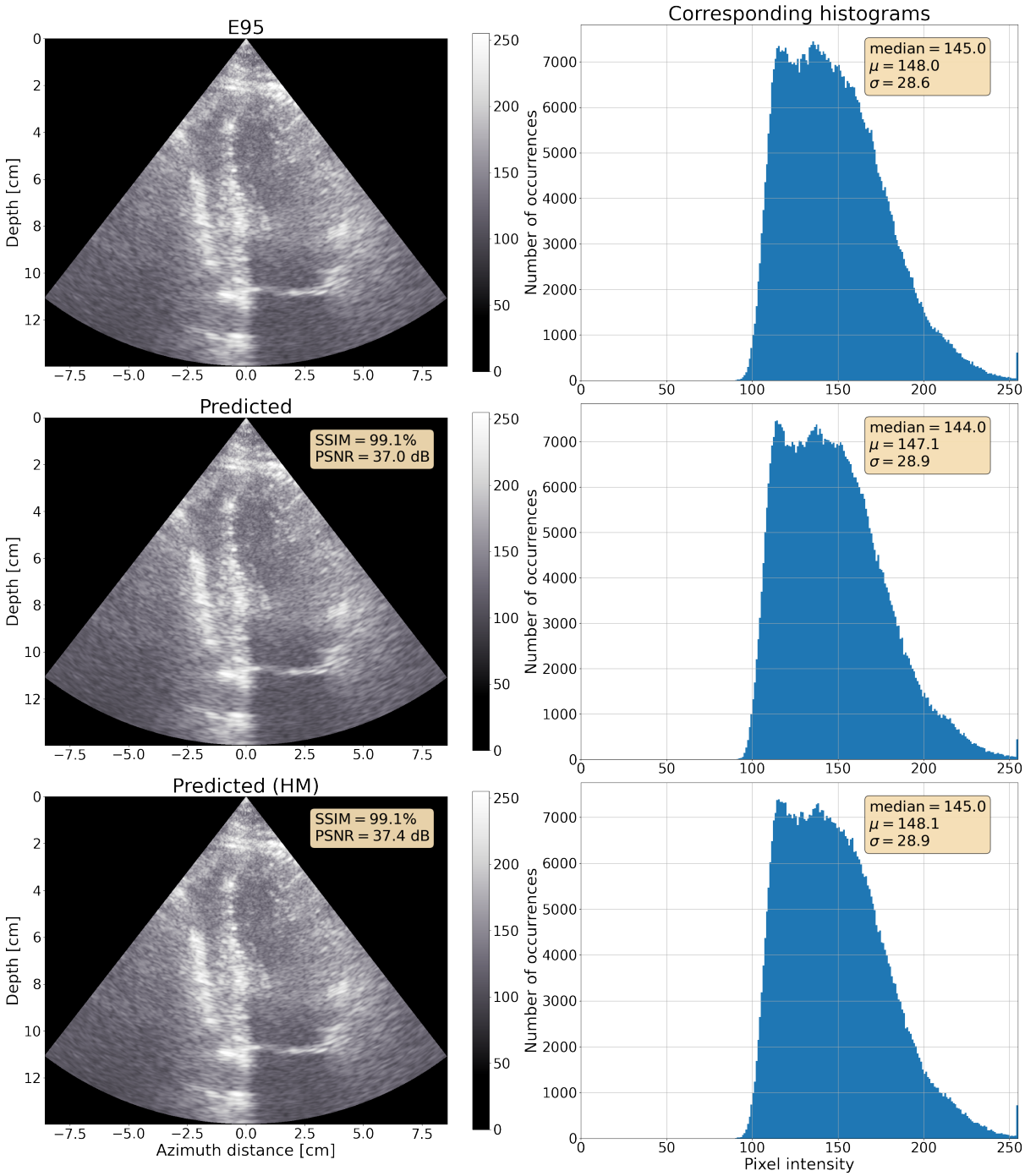


Figure 4.13: Case 5: An E95 image and its histogram compared to the corresponding predicted and HM predicted image. Cardiac view: A2C.

The difference plots in Figure 4.14 show that most of the image had the same pixel values as in the original E95 image. An exception was the bright vertical structure on the left which was amplified.

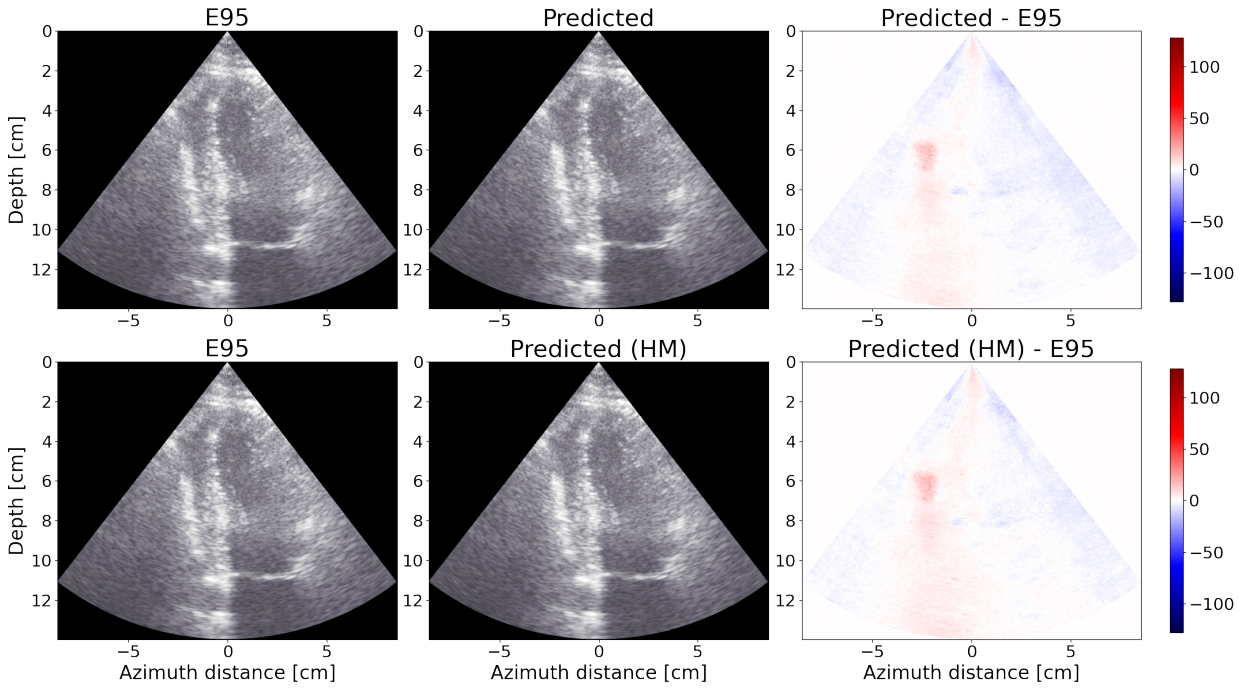


Figure 4.14: Case 5: The residual between an E95 image and its corresponding predicted image with and without histogram matching. Cardiac view: A2C.

Case 6 (4Vc-D probe)

Figure 4.15 shows a case with one of the lowest SSIM values of 98.3%

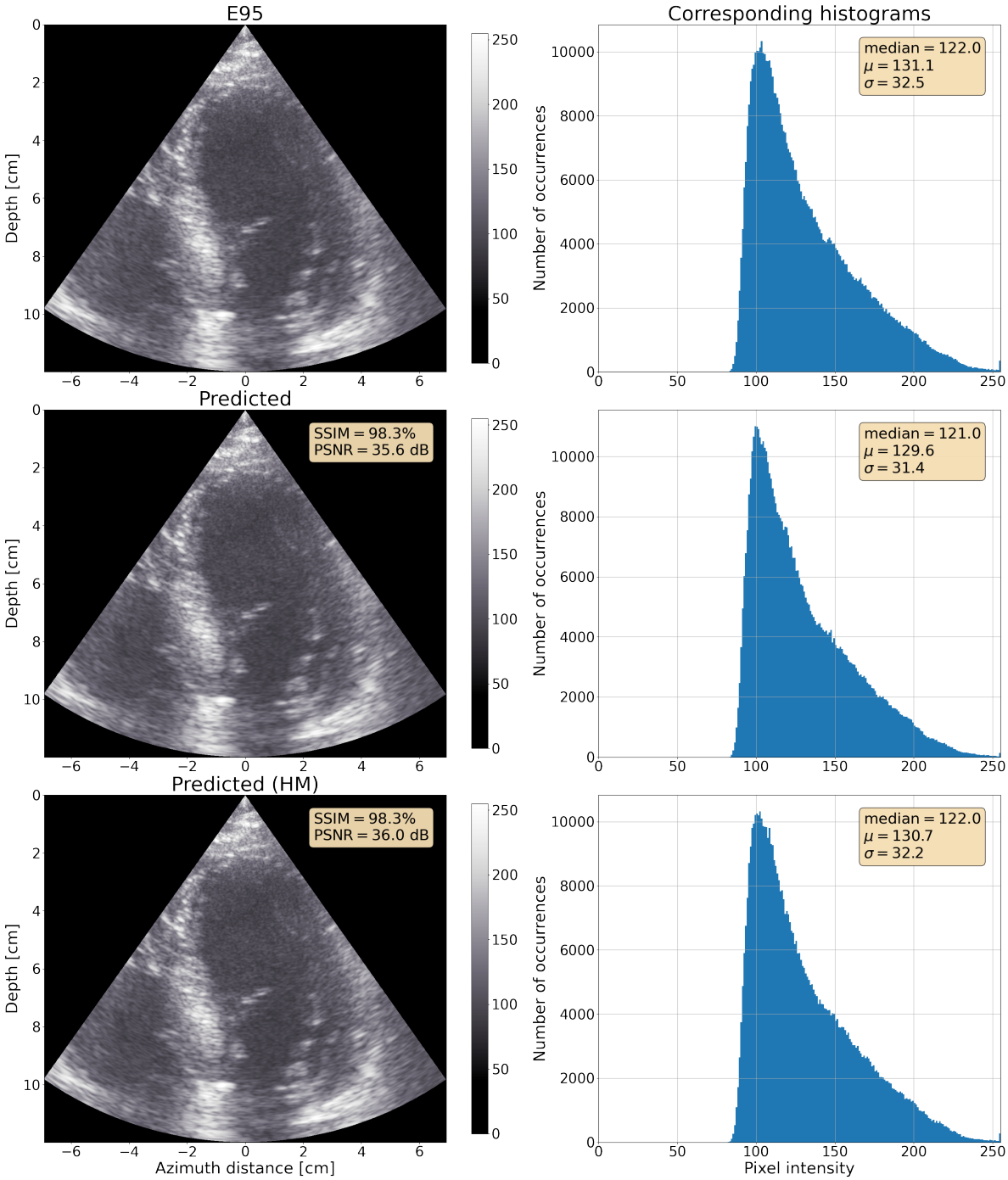


Figure 4.15: Case 6: An E95 image and its histogram compared to the corresponding predicted and HM predicted image. Cardiac view: A4C.

The difference plots in Figure 4.16 show increased pixel intensities in a large area around depth 6 cm and azimuth distance 2.5 cm. This made the vertical structure on the right side at depth 4 cm slightly more visible.

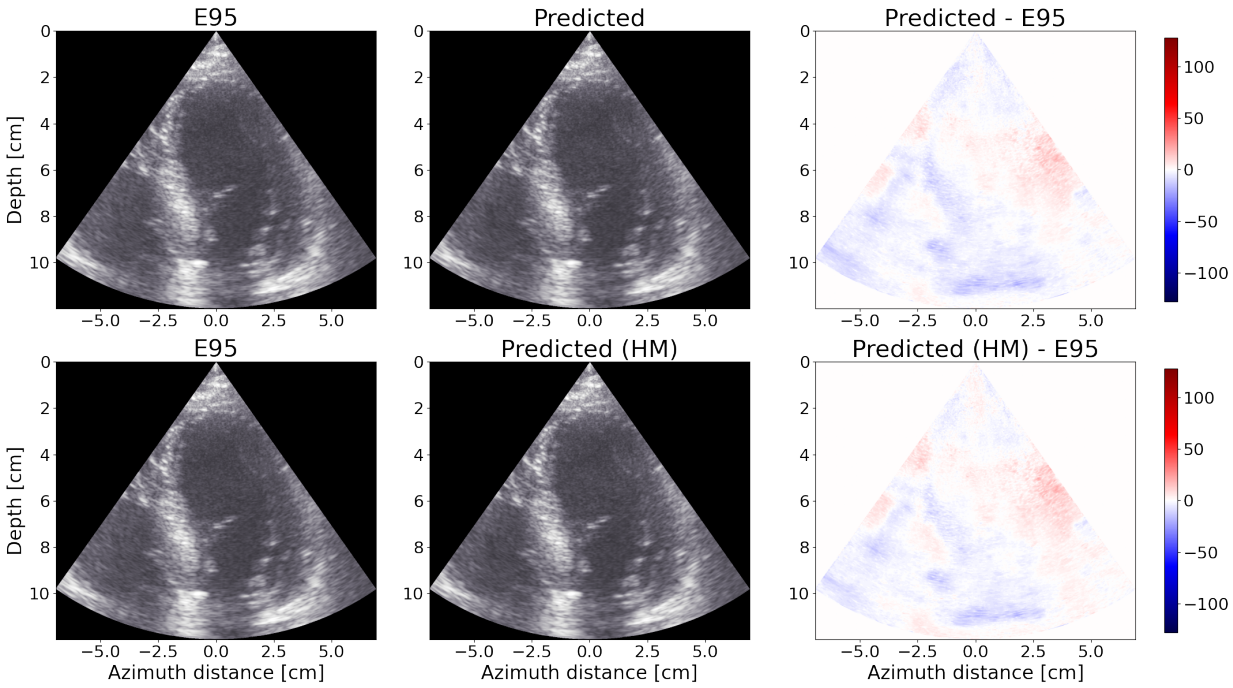


Figure 4.16: Case 6: The residual between an E95 image and its corresponding predicted image with and without histogram matching. Cardiac view: A4C.

Case 7 (M5Sc-D probe)

Figure 4.17 shows a case with data acquired using the M5Sc-D probe.

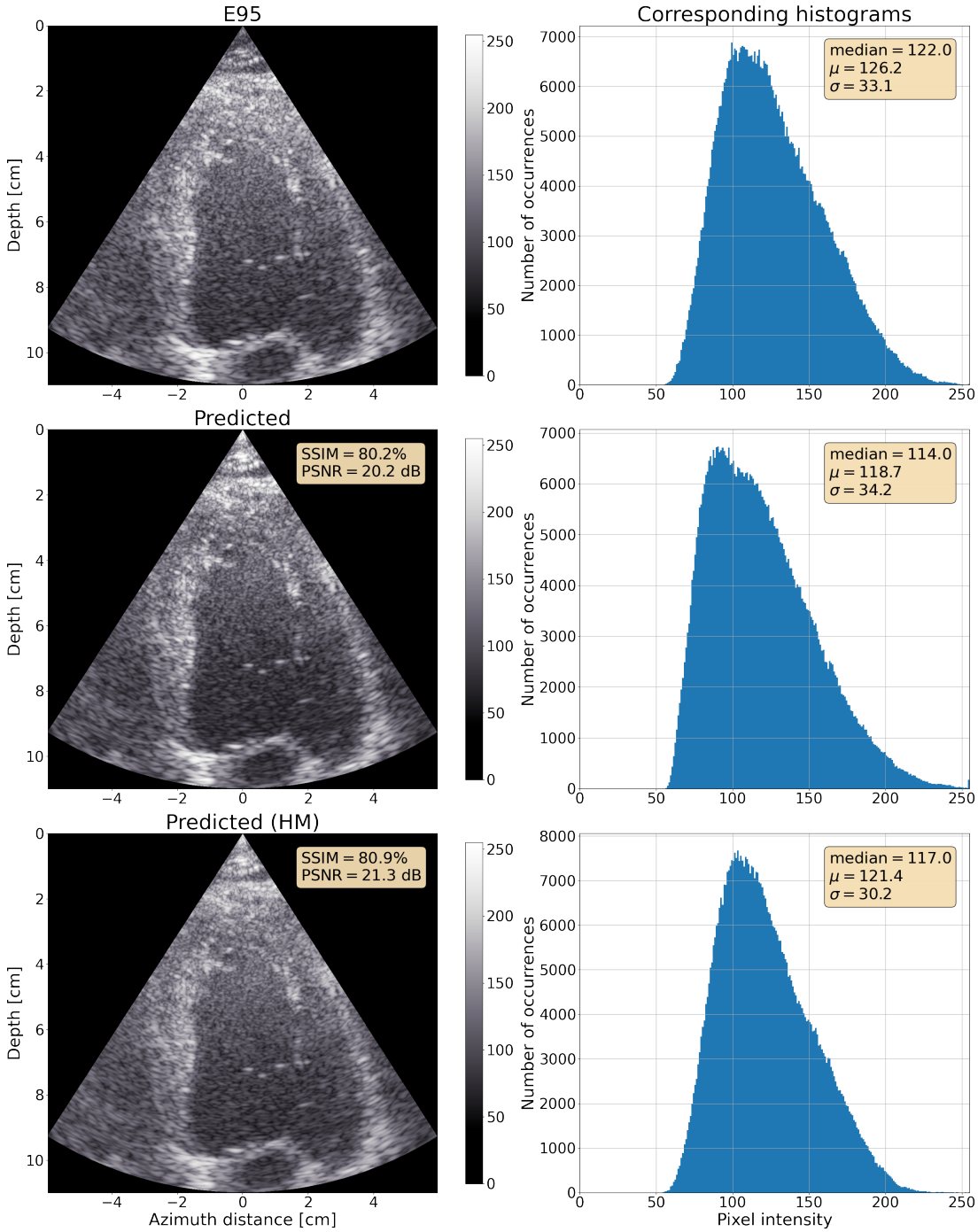


Figure 4.17: Case 7: An E95 image and its histogram compared to the corresponding predicted and HM predicted image. Cardiac view: A2C.

The predicted image had increased gain in the near-field. Further, the contrast resolution was better as the darker regions became darker relative to the bright structures. The mean and median values of predicted image were lower than those of the E95 image. After histogram matching, the darker areas became brighter. This caused a lower contrast resolution than in the E95 image.

Compared to the previous examples the SSIM and PSNR values were significantly lower. Evaluation of the M5Sc-D dataset resulted in an SSIM value of $(80.0 \pm 0.14)\%$ and a PSNR value of (20.2 ± 0.054) dB.

Figure 4.18 reveals that the region below the near-field was suppressed. Further, it shows that the speckles were altered.

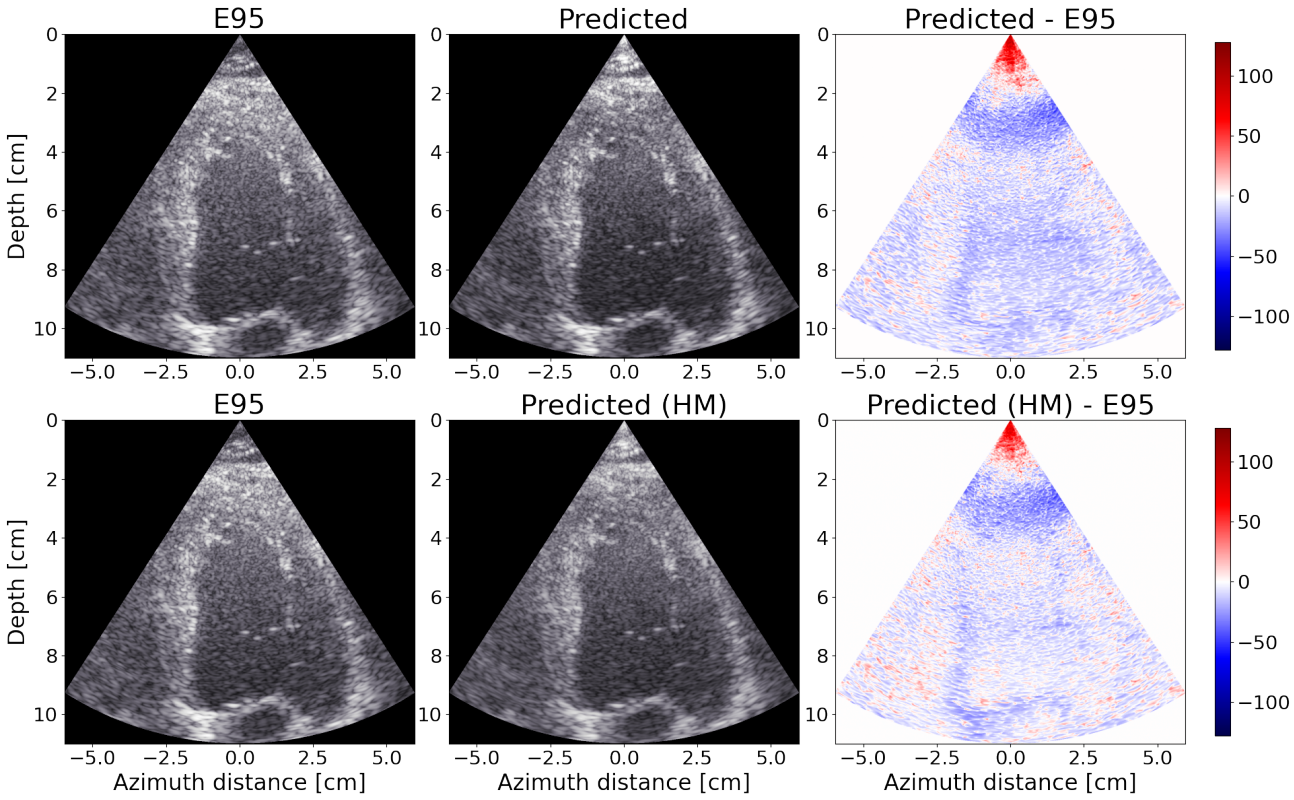


Figure 4.18: Case 7: The residual between an E95 image and its corresponding predicted image with and without histogram matching. Cardiac view: A2C.

4.2 Despeckling

The model used for despeckling was the same as the one in Table 4.6, with the exception of only having one elevation plane as input. Evaluation of the 4Vc-D test dataset gave an SSIM value of $(83.1 \pm 1.22)\%$ and a PSNR value of (28.3 ± 0.52) dB. When evaluating the model on the M5Sc-D dataset, the results

were an SSIM value of $(75.6 \pm 0.15)\%$ and a PSNR value of (19.9 ± 0.025) dB.

Figure 4.19 shows a case example when using the despeckling model.

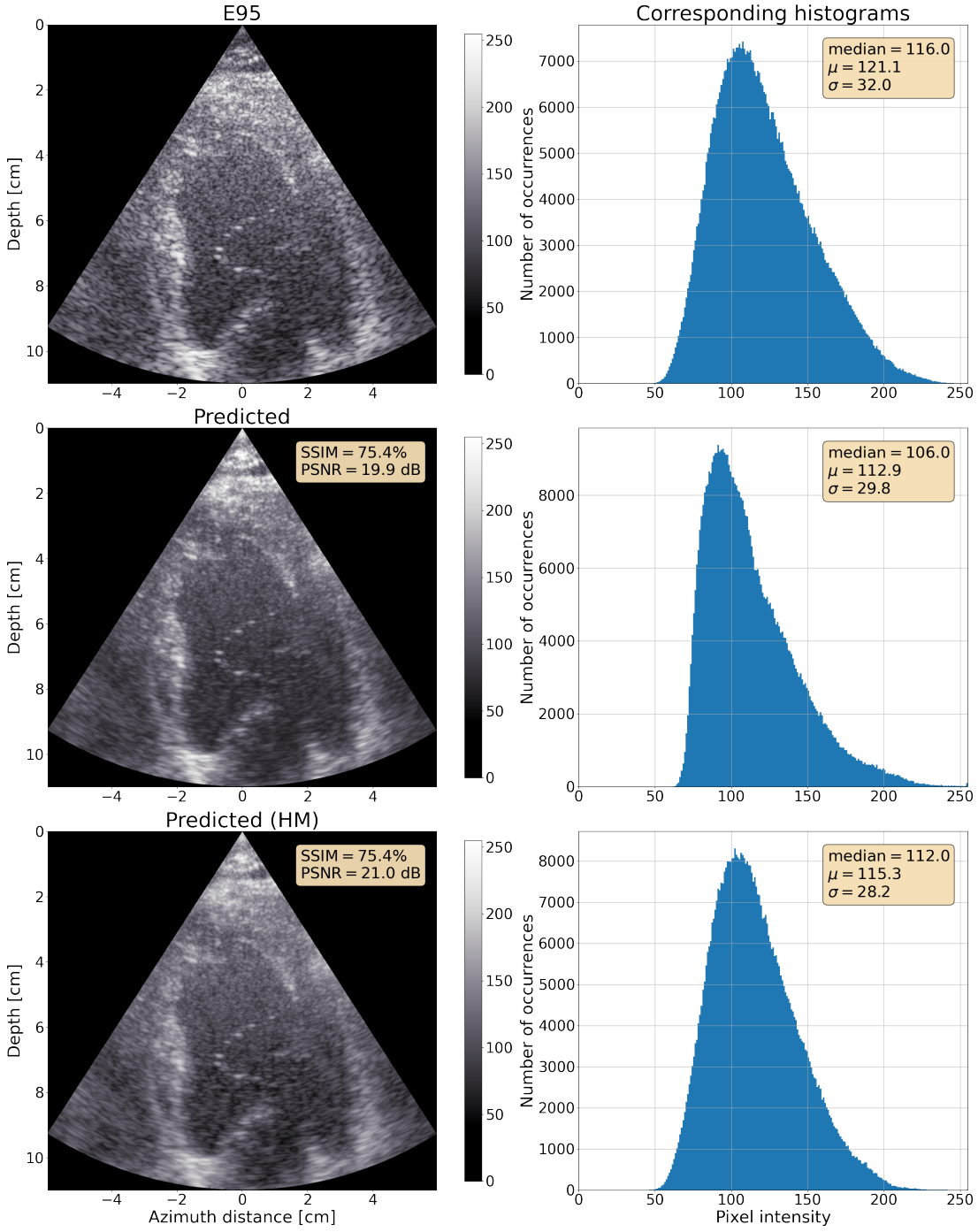


Figure 4.19: An E95 image and its histogram compared to the corresponding predicted and HM predicted image. The data was acquired using the M5Sc-D probe. Cardiac view: A2C.

The predicted image had smaller speckles and looked more smooth than the E95 image.

Histogram matching decreased the contrast resolution by lowering the brightest pixel intensities.

There were large variations between the histograms of the E95 image and the predicted image. For the predicted image, the distribution of pixel intensity values was shifted towards lower values. Further, the standard deviation of the histogram was lowered. Histogram matching in beamspace did not result in equal histograms for the scan converted images. Histogram matching further lowered the standard deviation and effectively lowered the contrast.

The difference plots in Figure 4.20 show a similar gain difference in the near-field to the one observed in Figure 4.16. Further, the difference plots show that the model suppressed speckle noise.

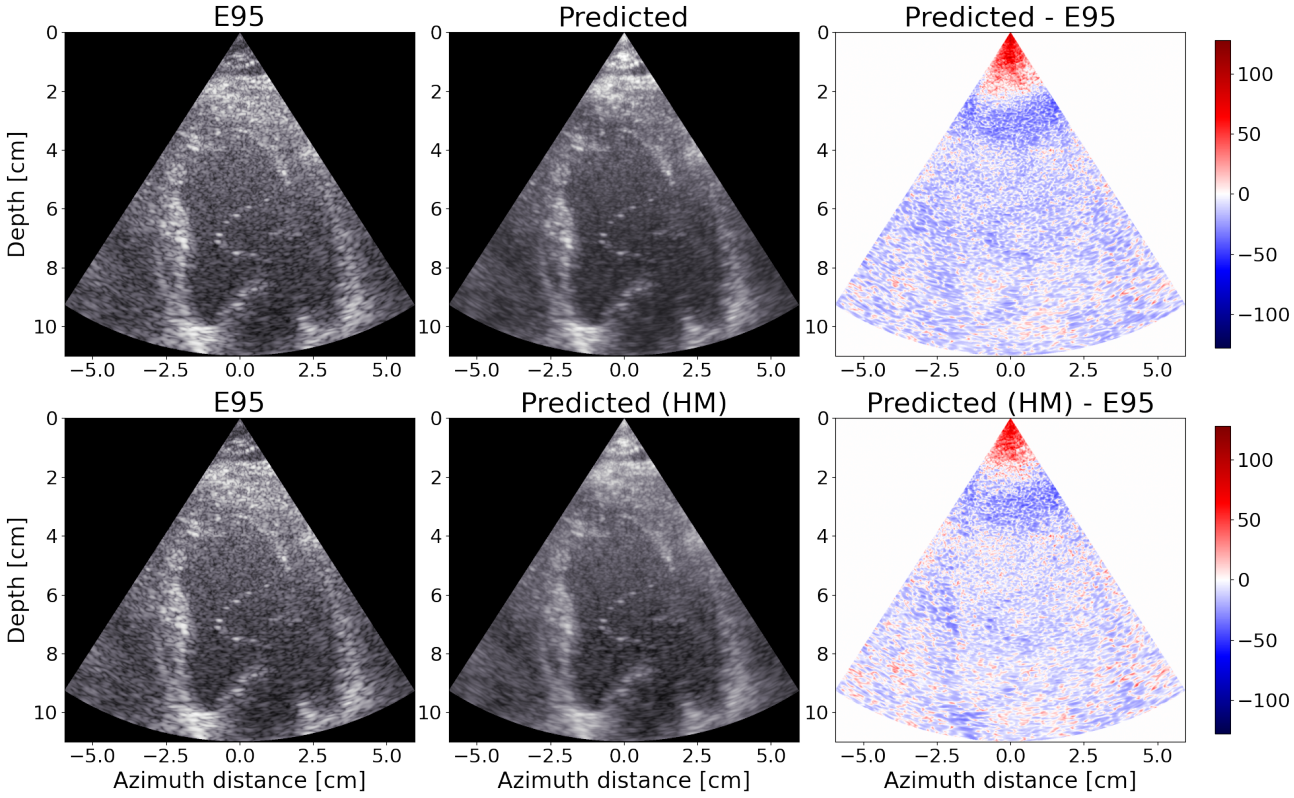


Figure 4.20: The residual between an E95 image and its corresponding predicted image with and without histogram matching. The data was acquired using the M5Sc-D probe. Cardiac view: A2C.

Chapter 5

Discussion

This chapter discusses the dataset and the stochastic part of the data pipeline. Further, it discusses the results and observations in the same order presented in Chapter 4.

5.1 Dataset and data pipeline

Visual inspections showed that the predicted images from the E95 replication model were nearly identical to the E95 images despite only using a dataset from 22 patients. This result shows that a ML model may learn the ultrasound signal processing chain with a dataset of about $7,400$ images. Still, increasing the dataset size would likely have enhanced the performance. Since the test dataset was limited to data from two patients, the results may not represent the performance of the model on a more diverse test dataset.

Since the dataset consisted of many series of 64 images that were similar, the model could have learned patient-specific features. This was not observed in the case examples. Since the ML model was only tested on cardiac images, its performance was not evaluated when imaging other body parts. The fact that shuffling was performed both before reading the data and on every training epoch may have decreased the risk of learning patient-specific features. Using a small batch size of 8 may have had a similar effect, as it reduced the chance of having images from the same DICOM file in a batch.

Further, the same transmit and receive center frequencies were used for the entire dataset. When the model was evaluated on data acquired using the M5Sc-D probe, the resulting images showed that the size and shape of the speckles were affected. This may be explained by the fact that the model was only trained for one frequency setting, but the M5Sc-D data was acquired using other transmit and receive center frequencies. Training the model with more combinations of acquisition parameters could increase robustness and model generalization, but it would require time-consuming data collection.

Data augmentation was not performed since the dataset already consisted of DICOM files containing 64 similar images from consecutive frames. This way, different realizations of similar image cases were already included in the dataset without having to augment the data. Data augmentation could have improved robustness by changing the data samples on every training epoch, but doing so was not trivial due to the nature of the IQ data. As Section 2.3.10 describes, augmented data should be realistic.

The large dynamic range of the IQ data caused noise augmentation problems. If adding WGN, the noise level would either be too high relative to weak scatterers or too low relative to the strong scatterers. As the speckle noise in ultrasound imaging is multiplicative, additive WGN was not used. Instead, noise augmentation could have been done by multiplicative WGN. This way, the noise level would have been adjusted to the strength of each scatterer. This could have affected the model's ability to map the IQ data to the correct corresponding pixel intensity values.

5.2 Learning and generalizing the Vivid E95 native signal processing

This section discusses the data pre-processing, the ML model designed to learn and generalize the E95 native signal processing, and its predicted images.

5.2.1 Pre-processing

The pre-processing steps proved to be crucial to achieve great results.

Since the spatial extents of the IQ data and the E95 images differed, the data had to be aligned. Three possible solutions were proposed: padding, cropping, or stretching. Padding could potentially confuse the network by adding irrelevant values to the input. Cropping would limit the spatial extent of the images. Stretching was chosen as the preferred method, despite affecting the sampling rate. The change in sampling rate may have made the ML model more robust to small changes in acquisition frequency.

To avoid potential aliasing problems, the data was only upsampled, not downsampled. The best results were archived when performing grid alignment and upsampling the IQ data and B-mode images with bilinear interpolation. This interpolation method was more suited than nearest-neighbor interpolation due to the continuous nature of the IQ data.

The data was upsampled to 768 samples in range and 512 samples in the azimuth direction. Typically the input dimensions of a CNN are much smaller than this. Because of this, the batch size had to be reduced. This may have had a negative impact on generalization as each mini-batch contained a small

number of different cases. Ideally, each mini-batch should be representative of the whole dataset. This could have been done with a more powerful GPU.

Scaling the data was necessary for the model to learn to map the input IQ data to the correct pixel intensity values. Each sample was scaled relative to its maximum absolute value when using local maximum absolute scaling. This made the network struggle to output the correct pixel intensities. When using local scaling, the network could not know how bright an image was relative to the other images in the dataset. The use of a global maximum absolute scaling solved this issue. In the results, this was seen as an increase in PSNR, indicating less differences in pixel intensities between the E95 image and the predicted image. The learning curves still have fluctuations. This may be explained by the fact that SSIM was used as the loss function, not the PSNR.

The best network performance was achieved when representing the IQ in Cartesian form. Figure 4.2 shows that using the Cartesian form led to a faster training convergence than when using the polar form. In addition, the learning curves showed a large generalization gap when representing the IQ data in polar form. One could think that the network would have improved performance using the polar form, as the magnitude part of this representation is identical to the envelope signal. An explanation of the poor performance may be that band-pass filtering is performed before and not after envelope detection in the ultrasound signal processing chain. The network may have struggled to band-pass filter the signal as it was already envelope detected.

Representing the IQ data in polar form could have facilitated the use of more complex augmentation techniques. The fact that the speckles appear after envelope detection means that the magnitude part of the IQ data could have been augmented with simulated speckle noise. This would not have been possible with IQ data represented in Cartesian form.

5.2.2 ML model

A grid search was performed to find a ML model of a sufficient capacity for the given task. Table 4.1 shows that the model of depth 3 and 16 filters gave the highest SSIM and PSNR. The model with depth 4 and 8 filters had a similar model capacity and had the second to best performance when comparing PSNR values. This indicates that these models had a suitable capacity for the given problem.

Further increasing the depth while holding the number of filters constant increased the number of model parameters. This may have caused overfitting, as the SSIM and PSNR values were lowered. Decreasing the number of model parameters to less than $100,000$ gave underfitting. Larger models with 32 filters could have further increased the performance. This could not be tested due to memory limitations. Another aspect of increasing the number of filters is how it

negatively affects the speed of the model due to a higher number of convolution operations and more model parameters. Network speed is critical as ultrasound images are formed in real-time.

Compared to the E95 replication model, the VGG-19 model used 2 more downsampling layers for images of less than half the amount of samples in the height and width dimensions. It is worth mentioning that the VGG architecture was designed for segmentation, where a higher depth may help extract the essential features. For the E95 replication model it might not have been as important to extract the essential features as it was designed to replace a signal processing chain.

In addition to the model depth, the way of downsampling and upsampling impacted the performance. The combination of downsampling by strided convolution and upsampling by strided transposed convolution yielded the highest SSIM value. This allowed the model to learn the most suited downsampling and upsampling operations instead of deterministic pooling and interpolation operations. A similar result was achieved with the model that used maximum pooling and strided transposed convolution for downsampling and upsampling, respectively. This indicates that the upsampling method accounted for most of the improvement. Average pooling may have given similar or better results than downsampling with maximum pooling.

The results in Table 4.3 show that the hyperbolic tangent was more stable than the ReLU, but the ReLU gave the highest mean SSIM value. The difference in standard deviation when using the two activation functions may have been caused by using different initialization seeds for the two training processes. The results obtained using the linear and sigmoid activation functions were significantly worse.

When using a model with a depth of 3 downsampling layers and 16 filters in the initial convolution module, the standard AE gave a mean SSIM value of 50.8%. This value was increased to 97.9% when using a U-Net with skip connections. The reason why the standard AE had such a much worse performance might have been the vanishing gradient problem. This could have been fixed by batch normalization or by choosing a non-restricting activation function like the ReLU. Batch normalization would have decreased the performance of the models with residual connections.

The U-Net style architecture was preferred over the standard AE as it preserves details across the bottleneck. Loss of high-frequency information would have resulted in images of a lower spatial resolution. Without the skip connections, the predicted images could have had disrupted features.

Replacing a signal processing chain is an unconventional use of the U-Net architecture, designed for image-to-image segmentation tasks. In this scenario, the U-Net performed time-to-space migration. Although much of the efficiency of CNNs is due to their translation invariance, the U-Net gave promising results.

Even better results were achieved by replacing the default convolution mod-

ules with residual modules or factorized Inception v2 modules. The Residual U-Net performed better than the U-Net, and the Inception U-Net further increased the SSIM and PSNR. By using Inception v2 modules, the model capacity was increased by making the network wider instead of deeper. Compared to the U-Net, the Inception U-Net had twice the number of parameters. Since the models of higher capacities Table 4.1did not have improved performance, the use of multiple kernel sizes likely accounted for the improvement. Further, the Inception U-Net increased the number of parameters without changing the depth or the number of filters in each convolution layer. This way, the capacity was increased while maintaining the optimal depth and filter combination.

Changing the loss function from SSIM to PSNR or combining the two did not improve the quantitative performance. The PSNR loss function gave a slightly higher mean PSNR value, but the difference was insignificant compared to when using the SSIM loss. Further, the PSNR correlates poorly with perceived image quality. Blurred images will still give a high PSNR value when comparing pixel by pixel. The SSIM was preferred over the other losses due to its ability to capture perceptual similarity.

Since the height and width dimensions of the images were large, the structures in the image effectively spanned over more pixels than if the dimensions were smaller. The SSIM values were calculated using the default 11x11 Gaussian filter. Changing the filter size when calculating the SSIM loss to fit the size of the features of interest could have potentially improved the results. As the structures in the images had different sizes, another solution could have been using the multiscale SSIM metric as the loss function.

The model with the highest mean SSIM value was achieved by combining the best test results. It gave an SSIM value of $(98.8 \pm 0.50)\%$ and a PSNR value of (35.8 ± 1.38) dB when evaluated on the 4Vc-D test dataset. In this case, combining the best results improved the model. This is not true in general, as the changes often are dependent on each other. An example of this is the performance of the observed depth and filter combinations.

The results differed slightly when training the model with a different initialization seed and lower early stopping patience. This indicates that there were some variations in performance from training to training. The standard deviations were different when using another initialization seed, but the mean values stayed relatively constant. Another explanation of the increased standard deviations may have been that the higher early stopping patience may have caused a slight overtraining.

5.2.3 Predicted image cases

4Vc-D probe

Case 1 - 6 showed that the predicted images and the E95 images were so similar that it was difficult to differentiate between the two without using histograms and difference plots. For all the case examples, the mean and median values of the predicted histograms were shifted towards darker pixel values. The same effect was observed in the difference plots as overweight of suppressed pixels. Histogram matching compensated for this shift by increasing the brightness in some areas. The resulting difference plots had a similar amount of suppression and amplification of pixel intensity values.

Figure 4.10 of case 3 showed suppression of parts of a structure in the bottom of the image and a similar change in the top left. These changes seemed to be stochastic. Similar behavior was observed in Figure 4.12 for case 4. In this example, the suppression decreased the brightness of a structure in the bottom right, which gave a lower contrast resolution. Such behavior may remind of that of a segmentation network, which is the conventional use of the U-Net architecture. Changing to the PSNR loss function could be another way of eliminating this behavior, as it focuses on pixel by pixel instead of structures.

Case example 6 had one of the lowest SSIM values of the test dataset, but the predicted image still looked close to identical to the E95 image. In this example, the predicted image made one of the vertical structures appear brighter. This observation might indicate that the model generalized the Vivid E95 native signal processing instead of replicating it. All the amplified areas were areas of interest with bright structures. Such an effect is positive as it highlights the regions of interest. At the same time, some thinner structures of interest were slightly suppressed, which is not desired. The focus of entire structures may have resulted from using the SSIM loss function or the segmentation-oriented U-Net architecture.

M5Sc-D probe

There were differences between the predicted images when using the 4Vc-D and the M5Sc-D probe. Evaluation of the M5Sc-D dataset gave an SSIM value of $(80.0 \pm 0.14)\%$ and a PSNR value of (20.2 ± 0.054) dB. This was significantly lower than when using the 4Vc-D probe that the model was trained for. It is worth mentioning that the test dataset of M5Sc-D data only consisted of one case of 64 frames from one patient. Evaluation of more cases would have given a better representation of the performance of the model when using the different probe.

The main contribution to the drop in quantitative performance may be explained by the gain variations in and below the near-field. This was likely

caused by using a different TGC for the M5Sc-D probe. Since the model was trained on data with the same TGC, it appears to have learned to map the pixel values to a specific intensity in this part of the image. Training the network with data acquired using different TGC settings could facilitate generalization of TGC settings.

Despite the lowered quantitative performance, visual inspections showed that the model performed well on the M5Sc-D data. Compared to the E95 image, the predicted image had improved contrast. The darker areas were suppressed, but the pixel intensities of the bright structures were maintained. This result indicates a benefit of using a state-of-the-art transducer may be learned by a ML model and transferred to a system using a less sophisticated probe.

Figure 4.16 shows changes in the speckles between the E95 image and the predicted image. This was likely caused by the differences in acquisition frequencies for the 4Vc-D and M5Sc-D data. Speckle size is frequency-dependent, and a higher frequency will give smaller speckles. The fact that the model was trained with data acquired with fixed transmit and receive center frequencies of 1.4 MHz and 2.8 MHz, respectively, may have impacted its ability to generalize the band-pass filtering for different frequencies. Because of this, the higher-frequency data from the M5Sc-D probe was unseen to the network.

5.3 Despeckling

When modifying the model to perform despeckling, the overall qualitative performance on the 4Vc-D test dataset was decreased. This was expected as only inputting one elevation plane effectively ignored half of the information used to form the elevation compounded E95 images.

Since the M5Sc-D probe does not support elevation compounding, no information was lost when inputting the IQ data into the despeckling network. Figure 4.19 shows that the technique successfully decreased the speckle noise and smeared the image, similar to the effect of elevation compounding. This result shows that DL may be used to transfer the speckle reduction behavior of elevation compounding from a 4D probe to a probe without this feature. The frame rate may be higher by using one elevation plane instead of multiple planes.

An unfavorable change in the predicted image was the decreased brightness of the structures of interest. This was observed as a shift towards lower pixel intensities in the predicted histogram. Histogram matching further reduced the brightness of these structures. This affected the contrast resolution. This undesired effect was caused by the histogram matching trying to even out the overall gain differences caused by the increased brightness in the near-field.

Figure 4.20 shows the despeckling effect as suppression of speckle-shaped structures in the difference plots. The acquisition frequency difference may also

explain the effect since Figure 4.18 shows a similar observation for the E95 replication model. At the same time, the smoothing effect was not achieved with the E95 replication model. This indicates that the despeckling model learned a similar smoothing effect to that obtained from elevation compounding.

When using elevation compounding, some scatterers may have only been visible in one of the two warped elevation planes. Because of this, the model may have learned to add or remove scatterers that were not present in the one input plane. As elevation compounding utilize the decorrelation of speckle between multiple planes, inputting data from only one warped elevation plane may cause unpredictable behavior. Ideally, the network should have been trained with inputs and outputs of the same spatial extents. This could have been done by beamforming the IQ data to one centered elevation plane instead of two warped planes.

Chapter 6

Conclusion

A DNN that replicates the GE Vivid E95 native signal processing was successfully developed. It learned band-pass filtering with TFC, envelope detection, filtering, frequency compounding, elevation compounding, GE logarithmic compression, image enhancement filtering, and more. A despeckling effect similar to that obtained by elevation compounding was successfully learned by modifying and retraining the model.

Pre-processing of the data was crucial for achieving great results. Upsampling by bilinear interpolation outperformed nearest-neighbor interpolation as it is better suited for continuous signals. The use of global maximum absolute scaling was needed to map the IQ data to the correct pixel intensity values. Representing the IQ data in Cartesian form gave the best results, but the polar form representation could potentially facilitate more data augmentations.

The best performing model was an Inception U-Net of depth 3 with 16 filters and $1,024,305$ parameters. It outperformed other U-Nets by going wider instead of deeper. Strided convolution and strided transposed convolution gave increased flexibility to learn downsampling and upsampling, respectively. The network was trained with the Adam optimizer and an SSIM-based loss function, as it focuses on structures instead of individual pixels.

Evaluation of the replication model on a test dataset consisting of 704 4Vc-D test cases gave an SSIM value of $(98.8 \pm 0.50)\%$ and a PSNR value of (35.8 ± 1.38) dB. Visual inspections showed that the predicted images were nearly identical to the ground truth E95 images. Since the test dataset was limited to data from two patients, the performance on a more diverse dataset could not be tested. Testing with M5Sc-D probe data showed an increased contrast resolution. Different TGC settings caused gain differences when using the M5Sc-D probe. Altered speckles indicated that the model struggled to generalize for different frequencies.

Predicted images from the despeckling model were smeared and had reduced speckle noise at the cost of a lowered contrast resolution.

Chapter 7

Suggestions for future work

Topic 1 Test the models in a real-time implementation.

Topic 2 Improve the generalization and robustness of the models by training them with different frequencies, TGC settings, and more image cases of different body parts.

Topic 3 Add depth information to the input of the networks to improve the time frequency compensation.

Topic 4 Train ML models to learn the signal processing of state-of-the-art ultrasound systems and transfer the benefits to low-end systems. An example may be the use of super-resolution to enhance the image resolution of handheld ultrasound systems.

References

[1] E. L. Gundersen, "Patient adaptive imaging in echocardiography: Deep learning-based estimation of fundamental and second harmonic center frequency for optimization of filtering in tissue harmonic imaging," project report in TFE4930, Department of Electronic Systems, NTNU – Norwegian University of Science and Technology, Dec. 2021.

[2] B. Luijten, N. Chennakeshava, Y. C. Eldar, M. Mischi, and R. J. G. van Sloun, "Ultrasound Signal Processing: From Models to Deep Learning," *arXiv:2204.04466*, Apr. 2022.

[3] B. Angelsen, *Ultrasound Imaging: Waves, Signals, and Signal Processing: Vol. 1: Basic Principles, Wave Generation, Propagation, and Beamforming in Homogeneous Tissue*. Trondheim: Emantec AS, 2000.

[4] A. L. Klibanov and J. A. Hossack, "Ultrasound in Radiology: from Anatomic, Functional, Molecular Imaging to Drug Delivery and Image-Guided Therapy," *Investigative radiology*, vol. 50, no. 9, pp. 657–670, 2015.

[5] Fortune Business Insights, "Ultrasound equipment market size, share & covid-19 impact analysis, by product (compact, and table-top), by application (radiology, gynecology, cardiology, point of care, urology, surgery, and others), by end user (hospitals & clinics), and regional forecast, 2021-2028." `https://www.fortunebusinessinsights.com/industry-reports/ultrasound-equipment-market-100515`. Accessed: 2022-05-12.

[6] S. M. Bierig and A. Jones, "Accuracy and Cost Comparison of Ultrasound Versus Alternative Imaging Modalities, Including CT, MR, PET, and Angiography," *Journal of Diagnostic Medical Sonography*, vol. 25, no. 3, pp. 138–144, 2009. Publisher: SAGE Publications Inc STM.

[7] Health, Center for Devices and Radiological, "Ultrasound Imaging," *FDA*, 2020.

[8] M. Ploquin, A. Basarab, and D. Kouamé, "Resolution enhancement in medical ultrasound imaging," *Journal of Medical Imaging*, vol. 2, 2015.

[9] A. Anvari, F. Forsberg, and A. E. Samir, "A Primer on the Physical Principles of Tissue Harmonic Imaging," *RadioGraphics*, vol. 35, no. 7, pp. 1955–1964, 2015. Publisher: Radiological Society of North America.

[10] S. J. M. Alqahtani, R. Welbourn, J. R. Meakin, R. M. Palfrey, S. J. Rimes, K. Thomson, and K. M. Knapp, "Increased radiation dose and projected radiation-related lifetime cancer risk in patients with obesity due to projection radiography," *Journal of Radiological Protection*, vol. 39, pp. 38–53, 2018. Publisher: IOP Publishing.

[11] W. Hedrick, D. Hykes, and D. Starchman, *Ultrasound Physics and Instrumentation*. Elsevier Mosby, 4 ed., 2005.

[12] B. Angelsen, *Ultrasound Imaging: Waves, Signals, and Signal Processing: Vol. 2: Propagation and Scattering in Heterogeneous, Nonlinear Tissue with Contrast Agent Imaging and Doppler Measurements*. Trondheim: Emantec AS, 2000.

[13] A. A. Nair, K. N. Washington, T. D. Tran, A. Reiter, and M. A. Lediju Bell, "Deep Learning to Obtain Simultaneous Image and Segmentation Outputs From a Single Input of Raw Ultrasound Channel Data," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 67, pp. 2493–2509, Dec. 2020.

[14] O. Karaoğlu, H. c. Bilge, and I. Uluer, "Removal of speckle noises from ultrasound images using five different deep learning networks," *Engineering Science and Technology, an International Journal*, 2021.

[15] S. Khan, J. Huh, and J. C. Ye, "Adaptive and Compressive Beamforming Using Deep Learning for Medical Ultrasound," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 67, pp. 1558–1572, Aug. 2020.

[16] D. Hyun, L. L. Brickson, K. T. Looby, and J. J. Dahl, "Beamforming and Speckle Reduction Using Neural Networks," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 66, pp. 898–910, May 2019.

[17] G. Jansen, N. Awasthi, H.-M. Schwab, and R. Lopata, "Enhanced Radon Domain Beamforming Using Deep-Learning-Based Plane Wave Compounding," in *2021 IEEE International Ultrasonics Symposium (IUS)*, pp. 1–4, Sept. 2021. ISSN: 1948-5727.

[18] M. Gasse, F. Millioz, E. Roux, D. Garcia, H. Liebgott, and D. Friboulet, "High-Quality Plane Wave Compounding Using Convolutional Neural Networks," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 64, pp. 1637–1639, Oct. 2017.

[19] F. Dietrichson, E. Smistad, A. Ostvik, and L. Lovstakken, "Ultrasound Speckle Reduction Using Generative Adversial Networks," in *2018 IEEE International Ultrasonics Symposium (IUS)*, pp. 1–4, Oct. 2018. ISSN: 1948-5727.

[20] L. Zhu, C.-W. Fu, M. S. Brown, and P.-A. Heng, "A Non-local Low-Rank Framework for Ultrasound Speckle Reduction," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 493–501, July 2017. ISSN: 1063-6919.

[21] F. A. Duck, "Nonlinear acoustics in diagnostic ultrasound," *Ultrasound in Medicine & Biology*, pp. 1–18, 2002.

[22] K. Maccallum, "Pros and cons of two popular ultrasound signal processing techniques." `https://starfishmedical.com/blog/pros-cons-two-popular-ultrasound-signal-processing-techniques/`. Accessed: 2022-05-06.

[23] J. Park, J. Kang, J. H. Chang, and Y. Yoo, "Speckle reduction techniques in medical ultrasound imaging," *Biomedical Engineering Letters*, vol. 4, pp. 32–40, Mar. 2014.

[24] D. M. Murtaza Ali and U. Dasgupta, "Signal processing overview of ultrasound systems for medical imaging," 2008. Publisher: Texas Instruments.

[25] G. U. Haugen, K. Kristoffersen, and A. R. Sornes, "Method and apparatus for performing ultrasound elevation compounding," Dec. 2015.

[26] C. Whatmough, J. Guitian, E. Baines, L. Benigni, P. Mahoney, P. Mantis, and C. Lamb, "Ultrasound image compounding: Effect on perceived image quality," *Veterinary radiology & ultrasound : the official journal of the American College of Veterinary Radiology and the International Veterinary Radiology Association*, vol. 48, pp. 141–5, Mar. 2007.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[28] Towards Data Science, "AI, Machine Learning, Deep Learning Explained Simply." `https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960`. Accessed: 2022-05-28.

[29] Towards Data Science, "Deep neural networks: How to define?." https://towardsdatascience.com/deep-neural-networks-how-to-define-73d87bf36421. Accessed: 2022-05-29.

[30] L. Wu, Z. Zhu, and W. Ee, "Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes," *arXiv*, June 2017.

[31] J. Brownlee, "No free lunch theorem for machine learning." https://machinelearningmastery.com/no-free-lunch-theorem-for-machine-learning/. Accessed: 2022-05-28.

[32] S. Yadav, "Weight initialization techniques in neural networks." https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78. Accessed: 2022-05-09.

[33] S. Ruder, "An overview of gradient descent optimization algorithms." https://ruder.io/optimizing-gradient-descent/index.html#stochasticgradientdescent. Accessed: 2022-06-02.

[34] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, Jan. 2017.

[35] J. Flynn, S. Ward, J. Abich IV, and D. Poole, *Image Quality Assessment Using the SSIM and the Just Noticeable Difference Paradigm*, vol. 8019. July 2013.

[36] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, Apr. 2004.

[37] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss Functions for Neural Networks for Image Processing," *arXiv:1511.08861*, Apr. 2018.

[38] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. Comparison of overfitting and overtraining," *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 5, pp. 826–833, 1995. Publisher: American Chemical Society.

[39] TensorFlow, "Better performance with the tf.data api." https://www.tensorflow.org/guide/data_performance. Accessed: 2022-04-14.

[40] C. Bishop, *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[41] Machine Learning Mastery, "Train neural networks with noise to reduce overfitting." https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/. Accessed: 2022-05-29.

[42] J. Riebesell, "Convolution operator." https://tikz.net/conv2d/. Accessed: 2022-04-13.

[43] A. Dertat, "Applied deep learning - part 4: Convolutional neural networks." https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. Accessed: 2022-04-12.

[44] J. Brownlee, "How do convolutional layers work in deep learning neural networks?." https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/. Accessed: 2022-05-28.

[45] J. Brownlee, "A gentle introduction to 1×1 convolutions to manage model complexity." https://machinelearningmastery.com/introduction-to-1x1-convolutions-to-reduce-the-complexity-of-convolutional-neural-networks/. Accessed: 2022-05-10.

[46] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," *arXiv:1412.6806*, Apr. 2015.

[47] Wikipedia, "Bilinear interpolation." https://en.wikipedia.org/wiki/Bilinear_interpolation. Accessed: 2022-05-22.

[48] Dive into Deep Learning, "Transposed convolution." https://d2l.ai/chapter_computer-vision/transposed-conv.html. Accessed: 2022-05-09.

[49] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556*, Apr. 2015.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012.

[52] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *arXiv:1311.2901*, Nov. 2013.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385*, Dec. 2015.

[54] Z. Alom, T. M. Taha, and V. K. Asari, "Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation," *arXiv:1802.06955*, p. 12, 2018.

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *arXiv:1409.4842*, Sept. 2014.

[56] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *arXiv:1512.00567*, Dec. 2015.

[57] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *arXiv:1411.4038*, Mar. 2015.

[58] N. Siddique, P. Sidike, C. Elkin, and V. Devabhaktuni, "U-Net and its variants for medical image segmentation: theory and applications," *arXiv:2011.01118*, p. 42, 2020.

[59] TensorFlow, "An end-to-end open source machine learning platform." `https://www.tensorflow.org/`. Accessed: 2022-05-22.

[60] Keras, "About keras." `https://keras.io/about/`. Accessed: 2022-05-22.

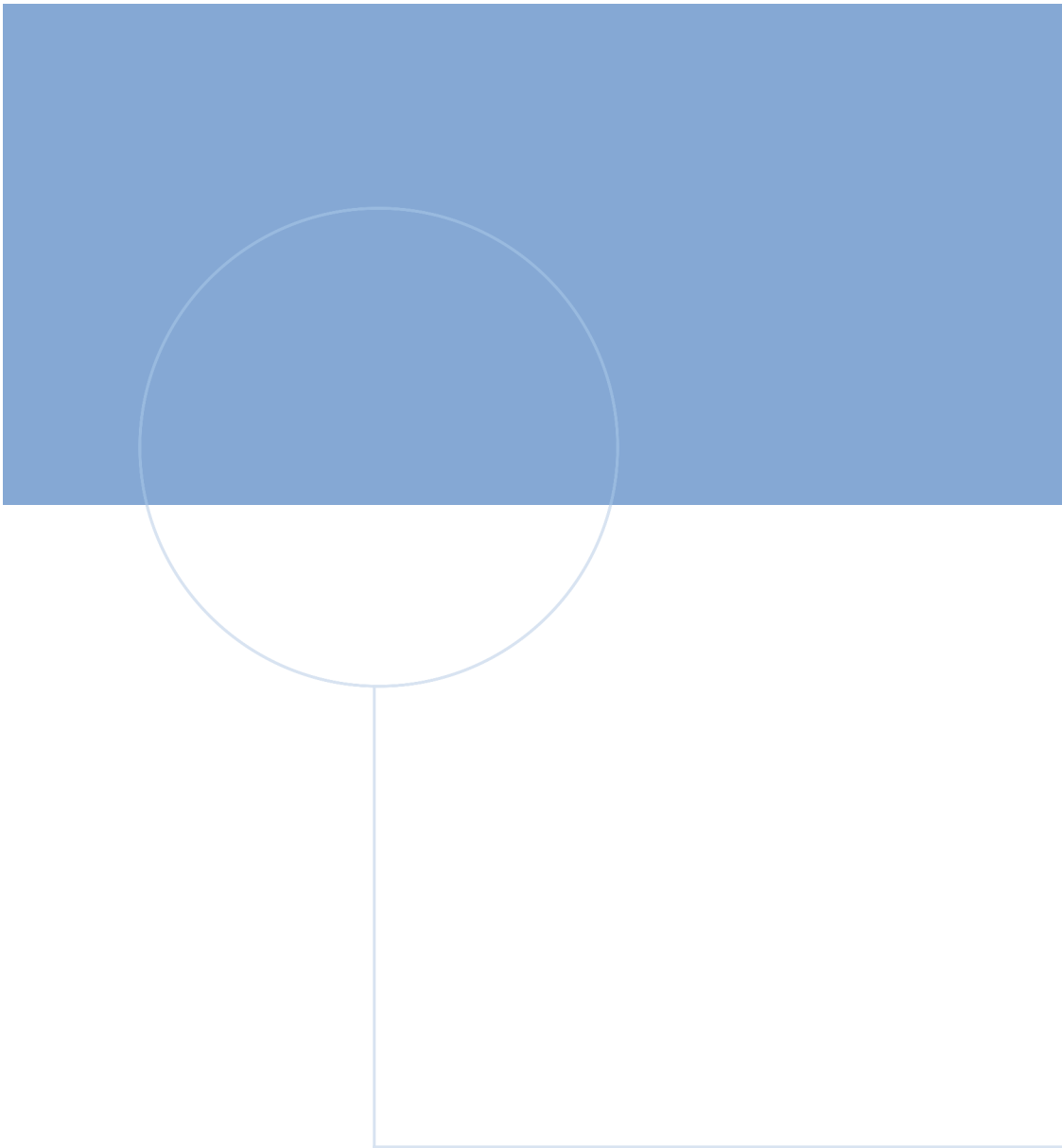
[61] The Medical Imaging Technology Association (MITA), "Dicom." `https://www.dicomstandard.org/`. Accessed: 2022-03-14.

[62] A. Østvik, E. Smistad, S. A. Aase, B. O. Haugen, and L. Lovstakken, "Real-Time Standard View Classification in Transthoracic Echocardiography Using Convolutional Neural Networks," *Ultrasound in Medicine & Biology*, vol. 45, pp. 374–384, Feb. 2019.

[63] E. Smistad, M. Bozorgi, and F. Lindseth, "FAST: framework for heterogeneous medical image computing and visualization," *International journal of computer assisted radiology and surgery*, vol. 10, 2015.

[64] E. Smistad, A. Østvik, and A. Pedersen, "High performance neural network inference, streaming, and visualization of medical images using fast," *IEEE Access*, vol. 7, 2019.

[65] Wikipedia, "Histogram matching." `https://en.wikipedia.org/w/index.php?title=Histogram_matching&oldid=1070474633`. Accessed: 2022-06-09.



NTNU

Kunnskap for en bedre verden