DISTINGUISHING CATHETERS FROM THEIR ECHOES IN ULTRASOUND IMAGES

A Thesis

Submitted to the Faculty of Graduate Studies and Research

In Partial Fulfillment of the Requirements

For the Degree of

Master of Science

in

Computer Science

University of Regina

By

Gopal Chandra Bala

Regina, Saskatchewan

September 2020

Abstract

In brachytherapy of prostate cancer treatment, needles, called catheters, are inserted into the prostate of a patient. The accuracy of the catheters' position is critical for the effectiveness of the treatment. Ultrasound images are commonly used to measure their positions. However, scattering of ultrasound signal in human tissue produces echoes of catheters that are sometimes, to the human eye, difficult to distinguish. The primary goal of this research is to investigate a novel approach for distinguishing ultrasound image regions corresponding to actual objects from the regions corresponding to their echoes. One of the challenges in ultrasound image analysis is the level of noise that is significantly higher than for other types of medical images, such as CT and MRI. Analysis performed directly on the intensity content of the regions of interest is called the spatial domain approach. Due to the complexity of the problem, Machine Learning has recently become an attractive technique to tackle this problem. On the other hand, it is worth to investigate frequency domain approaches due to their notable success in signal processing, e.g., in voice recognition. To compute the Fourier transform of a local image region, a window function is applied to mask off the remaining area of the image. The classic method for local frequency analysis is the Gabor transform that employs a Gaussian function as the window mask. However, the frequency coefficients corresponding to the intensity content in the region of interest are tangled together (through convolution) with the frequency coefficients of the Gaussian window in the frequency domain. A de-convolution algorithm, called Probing Detector, is utilized to reconstruct the frequency coefficients corresponding to the image content iteratively in the order of magnitude of the coefficients. This research proposes a set of novel feature vectors based on the reconstructed frequency coefficients in the frequency domain. These features are used as the input data to

a Neural Network classifier named *Probing FCNN*. The features derived from the frequency domain can potentially be more robust to noise in the image. At the same time, they can reduce significantly the sample size and training time required by the learning process. The results from our initial experiments are very promising.

Acknowledgment

I sincerely express my gratitude towards Dr. Sandra Zilles and Dr. Xue Dong Yang for their creative perspective and continuous support that stood out as the constant source of motivation during my research. Their deep insight and patience have encouraged me to explore various aspects of the topic by myself in the course of my research rather than following step-by-step instructions. This has not only enhanced my ability to think and solve problems from my perspective but has also enabled me to have a thorough understanding of my research project while working on it. I would also like to thank Dr. Sandra Zilles for bearing the funds for my research in the first two semesters.

I am truly thankful to Saskatchewan Cancer Agency for providing me with the funding and the necessary datasets without which it would have been difficult to obtain the aforementioned results from my research. In addition to this, I would also like to thank Dr. Derek Liu for his technical insights regarding the generated datasets which have helped us to achieve a deeper understanding of the data during the research.

Table of Contents

List of Tables

List of Figures

Chapter One

INTRODUCTION

Image processing and machine learning techniques have been successfully applied in the context of various prediction tasks pertaining to images, such as image segmentation, object detection, and image classification [1]. Generally, noise in images can make such tasks difficult, in particular when there are no effective methods known for filtering out the noise in images. One type of image that is typically hard to analyze due to noise is ultrasound images [2]. This thesis deals with the detection of objects in ultrasound images, using a combination of image processing and machine learning techniques.

1.1 Motivation and Problem Description

In brachytherapy of prostate cancer patients, catheters are inserted into the prostate in order to administer the radioactive treatment. In ultrasound images captured during the therapy session, the difference between the images of the catheters and their echoes is very subtle and often indistinguishable to the human eye. The echoes created due to the scattering of the ultrasound signal look very similar to the catheters in the vast majority of cases. For this reason, distinguishing the catheters from their echoes manually is a time-consuming process. After capturing the ultrasound images of the prostate cancer patients, the doctors spend substantial amounts of time annotating the images. Figure 1.1 shows a manually

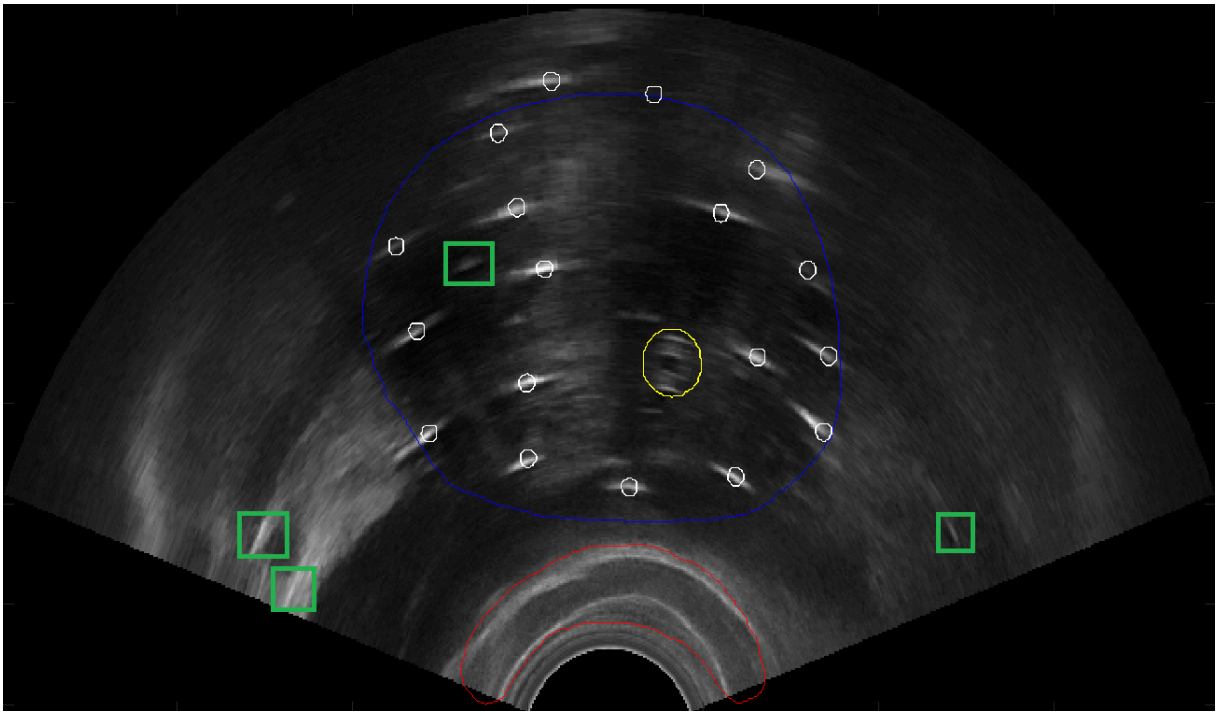


Figure 1.1 A labelled ultrasound image; Catheter: labelled with white circles; Prostate: labelled with blue circle; Rectum: labelled with yellow circle; Urethra: labelled with red color; Echo: labelled with green rectangles; source: Saskatchewan Cancer Agency.

annotated ultrasound image. In this image, the catheters are the bright regions annotated with the white circles. Besides the annotated bright regions, some other bright regions (annotated with the green rectangles) also appear throughout the ultrasound image which are often echoes. An automated system that is capable of learning from existing annotated ultrasound images could be very useful for the automatic detection of catheters in these ultrasound images.

When capturing an ultrasound image, speckle noise is added in with the original ultrasound signal. The added speckle noise is considered as one of the major obstacles for analyzing ultrasound images [3]. For this reason, the presence of speckle noise in both catheters and echoes makes it difficult to construct an automated system for distinguishing the catheters from their echoes. In spite of the fact that the catheters and echoes look very similar, the difference between these two objects lies in the ultrasound frequency [4]. The

catheter detection process can be made more effective by removing the speckle noise from both catheters and echoes. The goal of this thesis is to build and evaluate a predictor that can detect catheters and echoes. The predictor takes as input a small region of an ultrasound image that shows a bright spot that could be either a catheter or its echo. For each prediction task, the predictor outputs two confidence values where one confidence value represents the probability of a catheter and another confidence value represents the probability of an echo. These confidence values can potentially be useful for segmenting ultrasound images.

1.2 Approach

The process of building the predictor can be divided into two modules: (*i*) feature extraction based on image processing techniques and (*ii*) building a machine learning classifier. A de-convolution algorithm called probing detector [4] is applied to the ultrasound images that takes as input a small region (e.g., 33×33 pixels) of an ultrasound image. The de-convolution algorithm removes the speckle noise from the small region and constructs a set of frequency coefficients along with their positions (x, y-coordinates) in a two-dimensional space out of this region. The constructed frequency coefficients essentially represent the original ultrasound frequency of that region [4]. A set of features is computed using the positions of the constructed frequency coefficients for each region.

The machine learning classifier is built using a Neural Network learning algorithm. The computed features using the image processing techniques are used as the input data to the classifier. The Neural Network is composed of multiple layers where each layer contains a specific number of neurons which are the computational units. The learning parameters of this network are trained, resulting in a model that can take new data as input to make predictions. The Neural Network model is evaluated based on the training loss, training accuracy, and test accuracy.

1.3 Previous Work on Analyzing Ultrasound Images

A substantial amount of research has been conducted on ultrasound image processing in the past decade [5, 2, 6, 3, 7]. The research works on speckle noise reduction from ultrasound images and segmentation of images obtained through echocardiography (ultrasound imaging of the heart) are relevant to this thesis [6]. Several de-speckling techniques have been proposed to achieve the goal of speckle noise reduction from ultrasound images, for example, frequency compounding [7], and wavelet-based de-convolution [8].

In [7], the frequency compounding technique is introduced which is used for speckle noise suppression and contrast enhancement of ultrasound images. The frequency compounding technique suppresses speckle noise but it reduces the axial resolution of the ultrasound images. The axial resolution, also known as longitudinal, depth, or linear resolution, represents the number of pixels parallel to the ultrasound beam. The reduction of axial resolution causes the loss of ultrasound image pixels and makes an ultrasound image difficult to analyze. Frequency compounding divides the original ultrasound spectrum into multiple sub-bands. Each subband represents an individual ultrasound image. When all subbands are combined, the resulting ultrasound image contains less speckle noise compared to the original ultrasound image.

Echocardiography has been a major application area for image segmentation, in particular, using ultrasound images of the left ventricle (heart chamber) [6]. Tracking the endocardium motion (caused by the tissue or the blood pool) and then segmenting it are the core research topics in this case. One of the effective approaches proposed to achieve this goal is region-based segmentation using Artificial Neural Network (ANN) based methods [6]. For example, Binder et al. [5] designed a Neural Network that was trained using 369 sample regions for each training image where 279 regions represented the endocardium motion caused by the tissue and 90 regions represented the endocardium motion caused by the blood pool. The sample regions were 7×7 pixels and manually extracted from the training

images. For segmenting a new ultrasound image, first, the endocardial border (369 sample regions) was extracted and then the extracted border region was passed through the trained Neural Network to detect whether the border region represents endocardium motion caused by the tissue or the blood pool. If the Neural Network detected that the border region represented endocardium motion caused by the tissue, then 279 regions were segmented which highlighted the tissue region. Otherwise, 90 regions were segmented which highlighted the blood pool region.

The wavelet-based de-convolution [8] is another proposed approach to reconstruct an ultrasound image. In wavelet-based de-convolution, the low-pass filter and the high-pass filter are applied to the original image. The low-pass filter preserves the low frequencies which represent an approximate original image and the high-pass filter preserves the high frequencies which represent the horizontal details of the image. The low-pass and high-pass filters are applied again on the resulting two images. Applying low-pass and high-pass filters on the approximate original image generates two images where the first image is again the approximate image and the second image preserves the vertical details. Applying low-pass and high-pass filters on the horizontal detail image generates another two images where the first image preserves the horizontal details and the second image preserves the diagonal details. At this point, the total number of generated images is four. These four images are combined again to reconstruct the original images. After applying the wavelet-based de-convolution on an ultrasound image, the amount of speckle noise measured in the reconstructed image is less than the amount of speckle noise measured in the original image.

Although frequency compounding [7] and wavelet-based de-convolution [8] are effective in terms of suppressing speckle noise, they cannot be used for the experiment conducted in this thesis because their output type differs from that of the probing detector de-convolution [4]. However, these de-speckling techniques provide a deep insight into speckle noise in ultrasound images besides explaining the technical details. Furthermore, the research work on the segmentation of Echocardiography images using the Neural Network based approach

follows the local region based training strategy [6]. The *Probing FCNN* classifier designed in this thesis is also trained and tested using local regions of interest extracted from ultrasound images.

1.4 Contributions of this Thesis

This thesis makes two contributions to the fields of computer science and medical image analysis. First, the thesis combines a previously proposed de-convolution algorithm called probing detector [4] with a Neural Network classifier. The spectrum analysis section (Section 3.4) in this thesis illustrates the reasoning for extracting four latent features from the output of the de-convolution algorithm. The computational procedures of these four features are explained in Section 3.3. The extracted features for each small region of an ultrasound image provide simple and effective numerical data which are free of speckle noise. These features help the Neural Network classifier designed in this thesis to be trained with less computational complexity. This procedure allows us to evaluate the performance of the Neural Network using different volumes of data and different parameter settings.

Second, the classifier helps to distinguish the catheters from their echoes for each ultrasound image. At least partially automating this distinguishing process is the major goal in this thesis, which thus works toward computer-aided medical ultrasound image diagnostics. Using the output of the Neural Network classifier, a rectangular bounding box is drawn on top of each region of interest in the final output. Each rectangular bounding box is labelled with a prediction whether it contains a catheter or an echo. Such a tool can help in ultrasound image diagnostics for making subsequent decisions on the labeled regions of interest.

1.5 Structure of this Thesis

In Chapter 2, we introduce the technical background that is necessary for image analysis in this thesis. The Fast Fourier transform (FFT) is used by the de-convolution algorithm in this thesis. The fundamentals of Fourier transform are discussed before FFT, in order to explain the role of the Fourier transform in image processing. A de-convolution algorithm called probing detector and its implementation on ultrasound images are explained in Chapter 3. This chapter also shows how to compute a set of features using the output of the de-convolution algorithm after applying it on the ultrasound images as well as the reason for computing such features through spectrum analysis. In Chapter 4, the details on designing a Neural Network classifier are explained. The experimental results and analysis part is presented in Chapter 5. Finally, Chapter 6 concludes the thesis by discussing the limitations of our approach as well as potential future research directions.

Chapter Two

BACKGROUND STUDY

An image can simply be defined as a two-dimensional matrix where each entry is the intensity of the pixel at the corresponding image location. In image processing, an image is studied either in the spatial domain or in the frequency domain. The spatial domain represents the regular form of an image where a captured object is directly visible. On the other hand, the frequency domain represents the transformed version of an image where each spatial pixel intensity value is transformed into a particular frequency value [9]. In the frequency domain, many image processing operations, for example, filtering, convolution, and de-convolution, can be implemented efficiently with significantly less time complexity. For this reason, the Fourier transform is considered an important tool in image processing which can transform an image from the spatial domain to the frequency domain image. The Fourier transform allows us to access the geometric properties of an image in the spatial domain. Because the Fourier transform outputs a set of frequency spectra, processing frequency values in the frequency domain causes changes in geometric composition in the spatial domain [9]. This chapter explains the fundamentals of the Fourier transform, the Fast Fourier transform, an intuition about how the Fourier transform facilitates both convolution and de-convolution computation and the role of the Fourier transform in local frequency analysis.

2.1 Fundamentals of the Fourier Transform

The principle of the Fourier transform is that it transforms any signal in a time domain into a sum of sinusoids [10]. A sinusoid signal can be described by its amplitude, frequency, and phase angle. If a one-dimensional, continuous and integrable function is denoted by $f(x)$, then the forward Fourier transform of $f(x)$ is computed as shown in Equation (2.1) [9]. Given the forward Fourier transform $F(u)$ of a continuous and integrable function $f(x)$, the inverse Fourier transform of $F(u)$ is computed as shown in Equation (2.2) which recovers the original one-dimensional function $f(x)$. Here, x represents the coordinates in the time domain and u represents the coordinates in the frequency domain.

$$F(u) = \int_0^\infty f(x) \cdot e^{-i2\pi ux} dx \tag{2.1}$$

$$f(x) = \int_0^\infty F(u) \cdot e^{i2\pi ux} du \tag{2.2}$$

In Equation (2.1), the forward Fourier transform turns a function of time $f(x)$ into a function of frequency $F(u)$. On the other hand, Equation (2.2) does the reverse. It turns a function of frequency $F(u)$ into a function of time $f(x)$. The process of turning one type of a function into another type is called transformation. In both Equation (2.1) and Equation (2.2), we assume that $f(x)$ is real-valued and that $x, u \in R$.

The function of frequency $F(u)$ is complex-valued and is a combination of a real part $Re(u)$ and an imaginary part $Im(u)$ (as shown in Equation (2.3)).

$$F(u) = Re(u) + i \cdot Im(u) \tag{2.3}$$

The magnitude and phase angle of a signal can be described by these real and imaginary components [9]. If we plot a complex-valued $F(u)$ in a polar coordinate form (Re, Im), then the magnitude is the Euclidean distance between the center and the plotted point. The

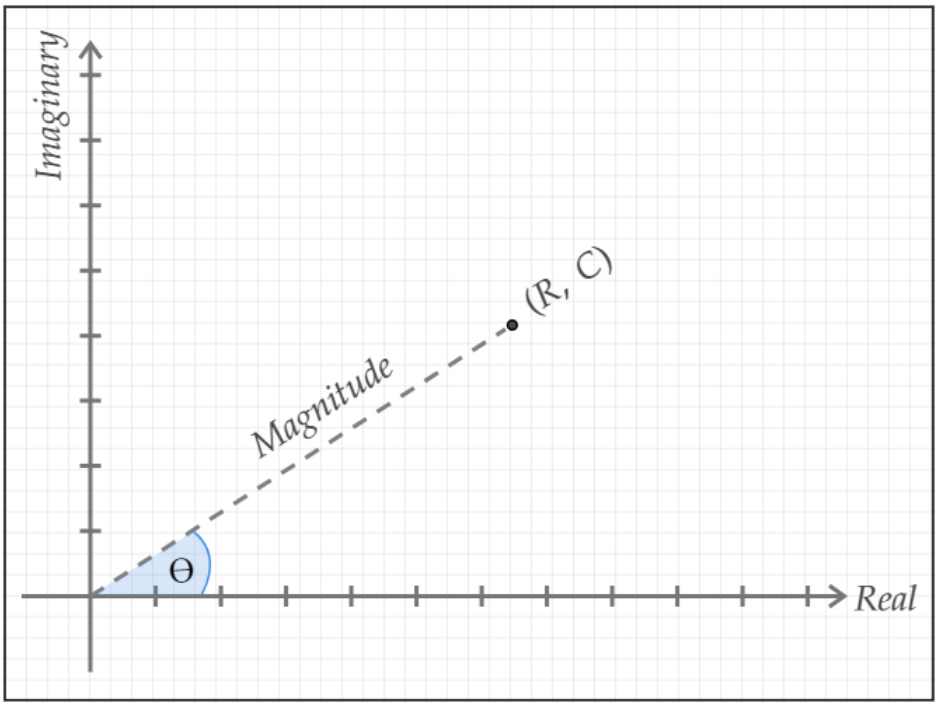


Figure 2.1 Representation of complex-valued $F(u)$ in polar coordinate form.

phase angle θ is the amount of clockwise rotation of the plotted point from the positive real axis. This is illustrated in Figure 2.1.

Here, the magnitude is given by $r = \sqrt{(Re)^2 + (Im)^2}$ and the phase angle is given by $\theta = tan^{-1}(\frac{Im}{Re})$. In Equations (2.1) and (2.2), the complex part can be handled by representing it in complex exponential form, known as Euler's formula [9]:

$$e^{j\theta} = cos\theta + i \cdot sin\theta \tag{2.4}$$

In this way, using Euler's formula in the Fourier transform, the time domain signal gets decomposed into a set of sinusoids of different frequencies. This transformation can be extended to image processing working with a two-dimensional function $f(x, y)$ where x and y are the spatial coordinates. The equations for both the forward Fourier transform $F(u, v)$ and the inverse Fourier transform $f(x, y)$ in the two-dimensional space are [9]:

$$F(u,v) = \int_0^\infty \int_0^\infty f(x,y) \cdot e^{-i2\pi(ux+vy)} dxdy \tag{2.5}$$

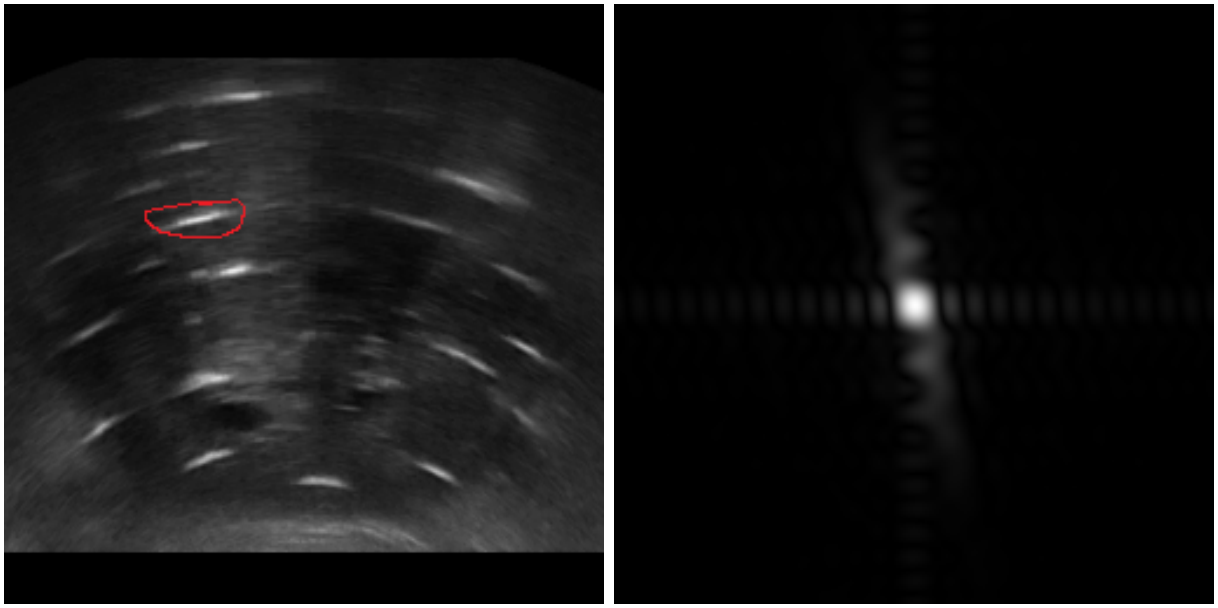
$$f(x,y) = \int_0^\infty \int_0^\infty F(u,v) \cdot e^{i2\pi(ux+vy)} dudv \tag{2.6}$$

An image is essentially a two-dimensional matrix where each value in the matrix represents the corresponding pixel intensity of the image. For this reason, in image processing, the two-dimensional discrete Fourier transform is used instead of the continuous Fourier transform. We always generate discrete data on a signal within a finite time frame and provide these data to the discrete Fourier transform as input. To compute both the forward discrete Fourier transform (Equation (2.7)) and the inverse discrete Fourier transform (Equation (2.8)) for a two-dimensional matrix, rather than running an integral computation, we perform a summation computation in two steps. Firstly, we evaluate a summation of frequencies for N samples indexed 0 through $N-1$ where N is the number of columns in the input matrix. Secondly, we evaluate another summation of frequencies for M samples indexed 0 through $M-1$ where M is the number of rows. In Equation (2.7), the final summation is divided by MN to normalize the output.

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cdot e^{-i2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{2.7}$$

$$f(x,y) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} F(u,v) \cdot e^{i2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{2.8}$$

The forward Fourier transform maps a function in the spatial domain into a function in the frequency domain. We already know that using Euler's formula, a set of polar coordinates are generated from a sinusoid and these polar coordinates are used to compute a set of magnitudes and phase angles. These magnitudes and phase angles represent a frequency spectrum in the frequency domain. Since a magnitude value represents the power of a signal for a particular frequency value, it is considered as an important feature in image processing



(a) Region of interest. **(b)** Generated magnitude.

Figure 2.2 Generated magnitude after applying the Fourier transform.

research. Applying the Fourier transform on a two-dimensional image matrix decomposes it into complex-valued exponential functions. As soon as the transformation is completed, the spatial domain image gets mapped into the frequency domain where each value shows the energy for a particular frequency. In this case, the resulting Fourier spectrum explains the distribution of energy for the image and this distribution plays an important role in image analysis. Figures 2.2(a) and 2.2(b) show a region of interest in an ultrasound image and the magnitude of its Fourier transform, respectively. The sub-image inside the red contour in Figure 2.2(a) is the region of interest.

For a two-dimensional signal, the output is a two-dimensional matrix where each entry is a complex number. To visualize a value or energy, we compute the magnitude of the complex value. Scaling the magnitude to the $0 - 255$ range creates a bright point in a two-dimensional plane. The brightness depends on the computed magnitude value. If the magnitude value is high, then the point is bright which represents high energy. On the other hand, if the magnitude value is low, then the point is faint which represents low energy. Each complex value in an output matrix of the Fourier transform and its visualized magnitude

are called a frequency coefficient. For the low frequency range, the frequency coefficients is larger compared to the frequency coefficients in high frequency range [4]. The crucial image energy in the frequency coefficients are located in the low frequency region [11]. On the other hand, the high frequency region with less energy contains fine details of the image, for example, edge and contour information of an object. The generated magnitude of the frequency coefficients in Figure 2.2(b) contains all information of the region of interest in the frequency domain. All analyses in this thesis are therefore conducted using the magnitude of frequency coefficients. The reason for the popularity of the Fourier transform in image processing is that it is easier to perform analytical tasks in the frequency domain than in the spatial domain. Besides, some computations, for example, convolution, are less expensive in the frequency domain [9].

2.2 Fast Fourier Transform (FFT)

The discrete Fourier transform converts a set of elements from the time domain to the frequency domain, or vice-versa [12]. In order to get an understanding of the time complexity of the Fourier transform, let's consider the one-dimensional discrete Fourier transform on elements $x_0, x_1, ..., x_{N-1}$, given on the following equation:

$$f(u) = \sum_{j=0}^{N-1} f(x_j) \cdot e^{-i2\pi(\frac{ux_j}{N})} \tag{2.9}$$

In Equation (2.9), every complex valued element of the frequency domain array $F(u)$ is computed by adding the product of every input element (x_j) and an exponential term of the spatial domain array $f(x)$. Thus, the time complexity of the discrete Fourier transform operation for N sample elements is $O(N^2)$. When the size of N is large, the $O(N^2)$ time complexity becomes problematic.

The FFT algorithm proposed by Cooley and Tucky [13] significantly reduces the time complexity of the discrete Fourier transform. The algorithm follows the divide-and-conquer

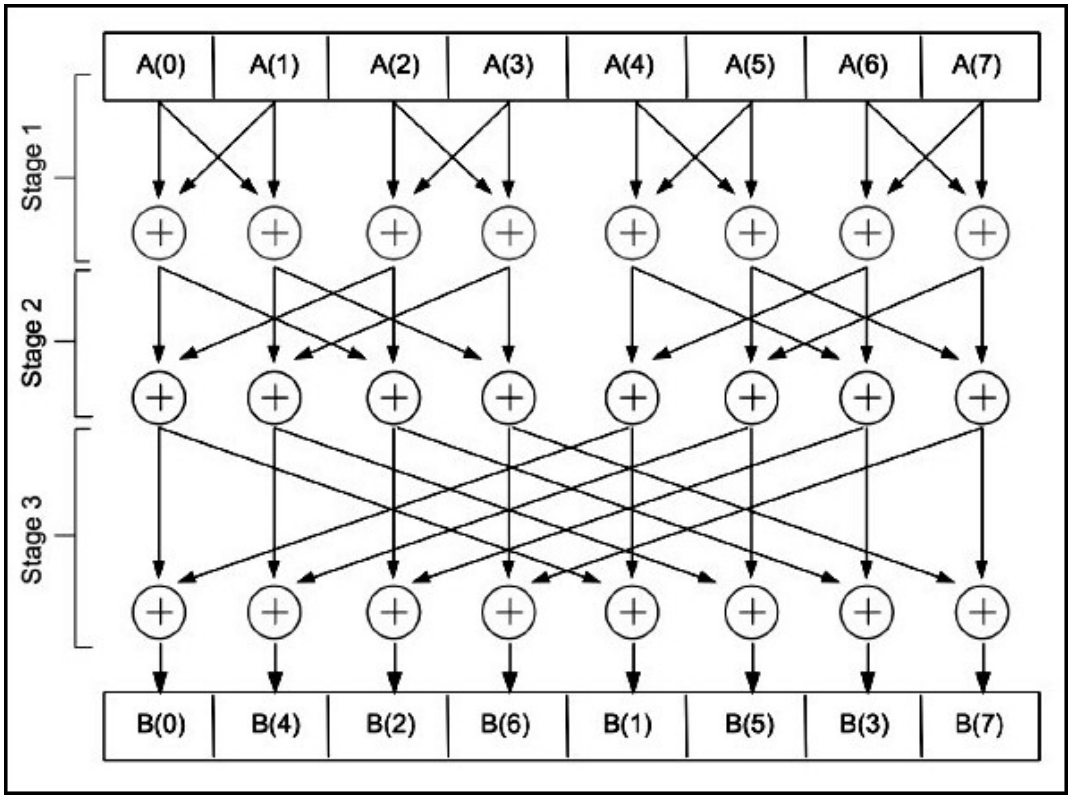


Figure 2.3 Cooley-Tukey FFT algorithm, source [12].

method. The time complexity of FFT to compute the discrete Fourier transform of any size array is $O(Nlog(N))$ where N is the size of an array. Such a significant reduction in computational time complexity made the Fourier transform an elegant solution to many computationally complex problems. The simplest and best-known Cooley-Tucky algorithm, called the Radix-2 Decimation in Time (Radix-2 DIT), is used widely for computing FFT [13].

Let's consider an array of eight elements (0 to 7). In the first step, the array is shuffled using bit reverse ordering. The bit reverse ordering approach reverses bit strings of every array element to get a new order. For example, after reversing 011, a bit string representing element 3, we get 110 which represents element 6. Therefore, the original order 3 is now transformed to order 6. In the second step, all these elements are combined back together. This step can be visualized intuitively using a butterfly diagram as shown in Figure 2.3. The

butterfly diagram represents the flow of data with time. The idea here is that in the first iteration, we combine pairs of elements together. In the second iteration, we combine pairs of pairs and in the third iteration, we combine pairs of pairs of pairs. Each combination is called a butterfly. Each butterfly represents a small Fourier transform that avoids unnecessary matrix computations.

2.3 Convolution Theorem

Convolution can be defined as a mathematical operation where two signals are combined together to produce a third one [14]. It is an integral operation used in image processing to perform filter operations, for example, object edge detection, image sharpening, image smoothing, etc. Suppose, we have two functions $f(x)$ and $g(x)$ where $f(x)$ is an input function and $g(x)$ is a kernel. For a one-dimensional or a two-dimensional case, both input function and kernel represent a one-dimensional or a two-dimensional matrix. A kernel can be defined as a matrix that is smaller in size than the input matrix. The kernel is moved throughout the input matrix and for each position of the kernel, the convolution operation is performed. The convolution between these two matrices ($f(x)$ and $g(x)$) can be defined as [11]:

$$h(x) = f(x) \otimes g(x) = \int_{-\infty}^{\infty} f(t) \cdot g(x-t)dt \tag{2.10}$$

In the above equation, t represents time in the continuous case. As an image has a finite number of data points, digital image processing uses the two-dimensional discrete form of convolution given in Equation (2.11) [9].

$$h(x,y) = f(x,y) \otimes g(x,y) = \sum_{l=0}^{y-1} \sum_{k=0}^{x-1} f(k,l) \cdot g(x-k, y-l) \tag{2.11}$$

In two dimensions, suppose, the Fourier transform of $f(x,y)$ and $g(x,y)$ in the spatial domain are $F(u,v)$ and $G(u,v)$ in the frequency domain. According to the convolution

theorem, the convolution result between $f(x, y)$ and $g(x, y)$ in the spatial domain is equivalent to the dot product between $F(u, v)$ and $G(u, v)$ in the frequency domain and vice versa [9].

$$f(x, y) \otimes g(x, y) = F(u, v) \cdot G(u, v) \tag{2.12}$$

$$f(x, y) \cdot g(x, y) = F(u, v) \otimes G(u, v) \tag{2.13}$$

The convolution theorem suggests that a convolution computation in one domain can easily be achieved by computing a simple multiplication in another domain. This property is crucial for computational efficiency in image analysis.

2.4 De-Convolution

In the past decades, a substantial amount of research has been conducted on de-convolution. In [15], a few popular de-convolution approaches are studied, for example, simulated annealing [16], and Wiener de-convolution [17]. In this section, the fundamental concept of de-convolution is explained.

In a newly generated image obtained through a convolution process, the value for each pixel is calculated as a weighted average of a set of neighbouring pixels of an original image. For this reason, it is difficult to apply de-convolution on a convoluted image to reconstruct the original image in the spatial domain. According to the convolution theorem, the convolution in the spatial domain is equivalent to the dot product in the frequency domain and vice versa. Therefore, de-convolution in the spatial domain can be achieved in the frequency domain by performing a division operation.

Let us consider image restoration as a de-convolution example. Suppose that $f(x, y)$ denotes an original image, $d(x, y)$ denotes an observed degraded image, and $g(x, y)$ denotes a kernel which generates a blurring effect by sliding the kernel across the image. Computing convolution between $f(x, y)$ and $g(x, y)$ in the spatial domain or computing dot multiplication

between between $F(u, v)$ and $G(u, v)$ in the frequency domain generates the degraded image $D(u, v)$. According to the Convolution Theorem, the relationships between computations in the spatial domain and computations in the frequency domain are:

$$d(x, y) = f(x, y) \otimes g(x, y) \tag{2.14}$$

$$D(u, v) = F(u, v) \cdot G(u, v) \tag{2.15}$$

In Equation (2.15), $D(u, v)$, $F(u, v)$, and $G(u, v)$ are the Fourier transform of $d(x, y)$, $f(x, y)$, and $g(x, y)$, respectively. This equation illustrates that the convolution between $f(x, y)$ and $g(x, y)$ in the spatial domain can easily be achieved by performing simple dot multiplication between $F(u, v)$ and $G(u, v)$ in the frequency domain. In this example, the original image can be restored from the convoluted degraded image if the inverse kernel of $G(u, v)$ is known. An inverse kernel is a matrix of the equivalent dimension of the original kernel which is a Gaussian kernel in this example. It can reverse the effect of an original kernel. Suppose, the inverse filter is $H(u, v)$, then the restored original image $D'(u, v)$ is [4]:

$$D'(u, v) = \frac{F(u, v) \cdot G(u, v)}{H(u, v)} \tag{2.16}$$

In de-convolution, the major goal is to estimate an inverse filter $H(u, v)$. In principle, it is difficult to reconstruct an optimal image from a convoluted degraded image due to the added noise $N(u, v)$ with the original image and computational complexity of $H(u, v)$. The initial value of $H(u, v)$ is not known in most cases. Several algorithms have been proposed to estimate the inverse filter utilizing an approach called blind de-convolution [15]. In blind de-convolution, a random value is assigned to $H(u, v)$ initially. After each iteration, the reconstructed image is compared with the original image and this comparison returns an error rate. A single iteration performs de-convolution for all possible kernel positions throughout an entire image. A larger error rate explains that the newly reconstructed image

is less similar to the original image. On the other hand, a smaller error rate explains that the newly reconstructed image is more similar to the original image. The target in blind de-convolution is to find a smaller error rate and thus to approximate the original image more closely with the reconstructed image.

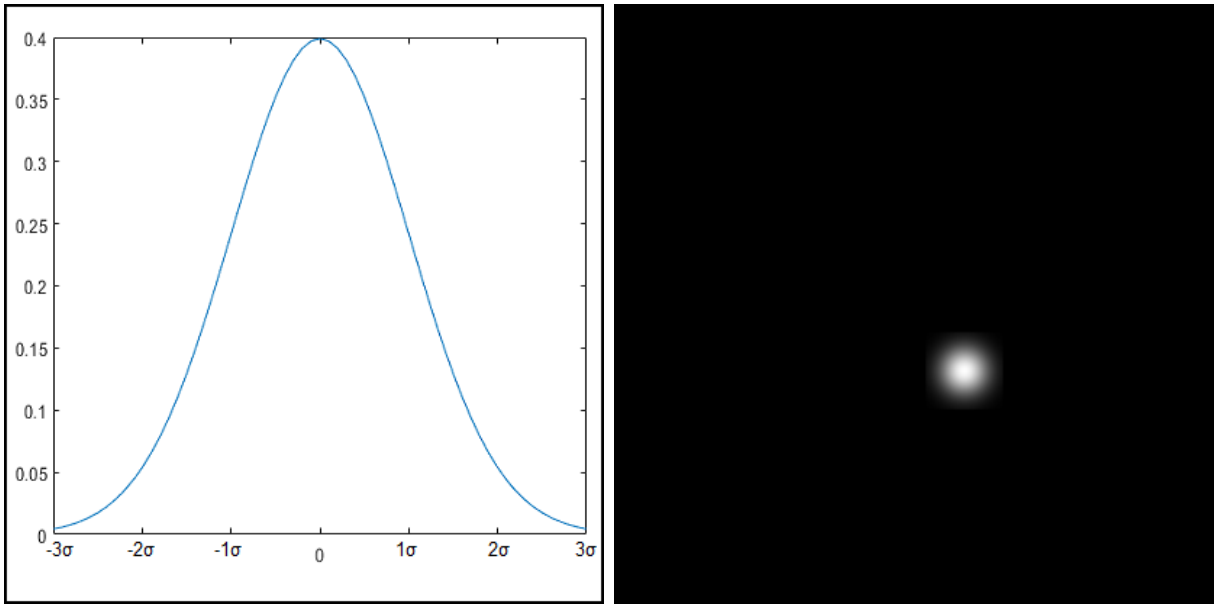
2.5 Gaussian Kernel

The Gaussian kernel is widely used in image processing, for example, noise reduction, image blurring, reconstruction of a blurred image, etc. In this thesis, the Gaussian kernel is used to restore the original frequency spectrum by reducing noise from a region of interest. For a one-dimensional signal, the Gaussian kernel function can be defined as [9]:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(\frac{x^2}{2\sigma^2})} \tag{2.17}$$

Here σ is the standard deviation of the Gaussian function. The standard range of the distance of data points from the mean for a Gaussian kernel is -3σ to $+3\sigma$ [4]. When computing the values of a one-dimensional Gaussian kernel of size N, the one-dimensional matrix holds the lowest values in the first (0) index and the last $(N-1)$ index. These lowest values explain that their distance from the mean is maximum. In this matrix, the mean value is always found in the middle $(N-1)/2$ index. If we compute a Gaussian kernel within the range of -3σ to $+3\sigma$, all the indices of the kernel are filled up with the values between this range. For a two-dimensional case, we compute an $N \times N$ matrix following a similar approach. The reason for considering -3σ to $+3\sigma$ as a standard range is that the values beyond this range are very small (close to zero) and considered as 0 (as shown in Figure 2.4 (a)).

The two-dimensional Gaussian kernel is used in digital image processing (Figure 2.4 (b)). While the Gaussian kernel is applied to an image, the center of the kernel and the center of a region of interest are located in the same position [4]. The rest of the image pixel values are



(a) 1D Gaussian kernel with $\sigma = 1$. **(b)** 2D (33×33) Gaussian kernel.

Figure 2.4 1D and 2D Gaussian kernel.

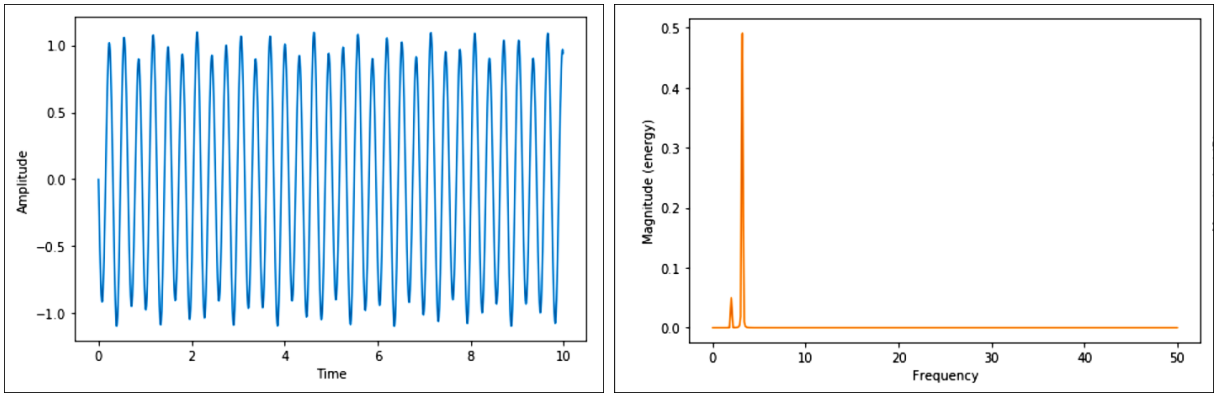
set to 0 in order to avoid interference from outside of the region of interest. Suppose, $f(x, y)$ is an image with a region of interest centered at (x_0, y_0). A two-dimensional Gaussian kernel $g(x, y, x_0, y_0)$ with the same center coordinate (x_0, y_0) can be defined as [9]:

$$g(x, y, x_0, y_0) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2})} \tag{2.18}$$

By changing the (x_0, y_0) coordinate values, the Gaussian kernel can be moved across an image. In Section 3.1, the usage of the Gaussian kernel will be explained in detail in terms of the original frequency spectrum reconstruction.

2.6 Local Frequency Analysis

In signal processing, a stationary signal results from repeating a single signal unit, for example, a sinusoid, across the complete spatial domain. On the other hand, a non-stationary signal, for example, an image, is composed of discrete signal units that appear in a particular time interval [18].



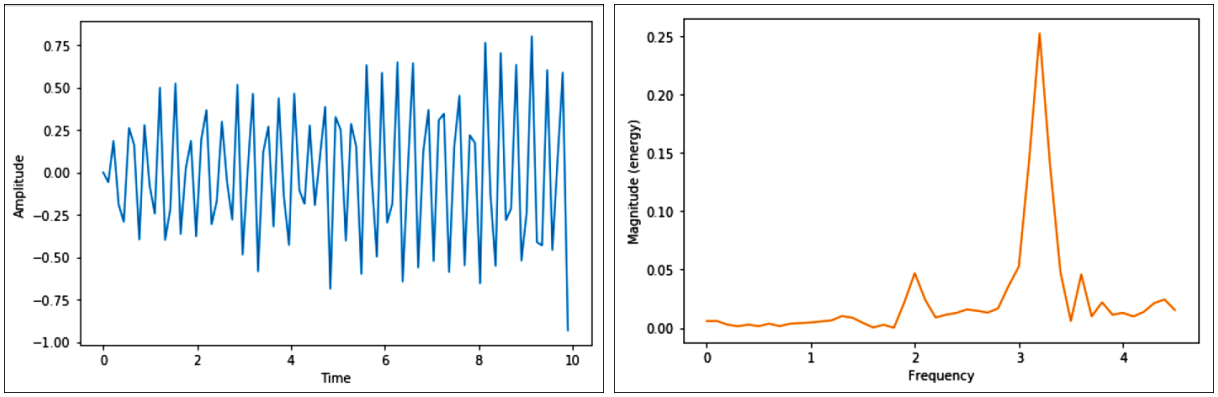
(a) A spatial domain signal. **(b)** Magnitude in the frequency domain.

Figure 2.5 Computed magnitude of a stationary signal, source [4].

To map a signal from the spatial domain to the frequency domain using the Fourier transform, the signal components are multiplied by sinusoids with discrete frequency values. Then performing an integration throughout the entire period outputs the frequency coefficients of the signal. Thus, applying the Fourier transform on a stationary signal (shown in Figure 2.5 (a)) one can compute its magnitude as a primary frequency response (shown in Figure 2.5 (b)) [18] which shows energy distribution.

Unlike frequency components in a stationary signal, frequency components in a non-stationary signal are varied throughout an entire time interval. Figure 2.6 (a) shows a non-stationary signal and Figure 2.6 (b) shows the magnitude of the signal that is achieved through the Fourier transform. The important observation here is that the computed magnitude shown in Figure 2.6 (b) has a similarity with the computed magnitude shown in Figure 2.5 (b). This observation suggests that the Fourier transform can find a particular magnitude in both stationary and non-stationary signals. However, for a non-stationary signal, the Fourier transform cannot explain a set of frequency components located at a particular time period.

In order to perform such local frequency analysis on a non-stationary signal, short term Fourier transform (STFT) is considered an effective method [4]. The STFT method is applied to a specific portion of an entire signal. This method uses a window function for pooling



(a) A spatial domain non-stationary signal. **(b)** Magnitude in the frequency domain.

Figure 2.6 Computed magnitude of non-stationary signal, source [4].

a specific portion. Utilizing this method, a signal can be segmented into multiple smaller regions known as regions of interest [4]. The standard Fourier transform is applied to each region of interest, which results in region-specific frequency information.

The size of a window function has to be fixed for multiple regions of interest throughout a signal because the frequency spectrum depends on the window size. Let us consider the one-dimensional discrete Fourier transform given in Equation (2.9) to understand the relationship between window size and the frequency coefficients. Applying a window on top of a signal reduces the dimension (number of discrete sample data) of the pooled region of interest. With this reduction of discrete sample data N, the frequency resolution (range of u) is also reduced. Besides, the sampling location is also changed with the change of window size. For example, Tables 2.1 and 2.2 show frequency values for window sizes $N = 4$ and $N = 6$, respectively.

Tables 2.1 and 2.2 suggest that energy for a specific frequency u in one table cannot be mapped to the same frequency in another table because of different N values. For this reason, the size of a window function has to be fixed for multiple regions of interest throughout a signal.

u	0	1	2	3
$2\pi\left(\frac{ux}{N}\right)$	0	$2\pi\left(\frac{x}{4}\right)$	$2\pi\left(\frac{2x}{4}\right)$	$2\pi\left(\frac{3x}{4}\right)$

Table 2.1 Frequency spectrum for 4 samples.

u	0	1	2	3	4	5
$2\pi\left(\frac{ux}{N}\right)$	0	$2\pi\left(\frac{x}{6}\right)$	$2\pi\left(\frac{2x}{6}\right)$	$2\pi\left(\frac{3x}{6}\right)$	$2\pi\left(\frac{4x}{6}\right)$	$2\pi\left(\frac{5x}{6}\right)$

Table 2.2 Frequency spectrum for 6 samples.

2.7 Speckle Noise in Ultrasound Images

In medical diagnosis, ultrasound images are widely used because of low cost, non-invasive capture, and the capability of capturing real-time images. However, there are a number of drawbacks that include: added noise from ultrasound equipment, environmental noise, and interference patterns. These types of noise are multiplicative in nature and commonly known as speckle noise [2]. They reduce ultrasound image resolution and contrast which makes an ultrasound image difficult to analyze. Multiplicative noise is modeled as the dot product of an original image and noise. Suppose, $f(x, y)$ is an original image, $n(x, y)$ is speckle noise, then a observed noisy speckle image $h(x, y)$ is [2]:

$$h(x, y) = f(x, y) \cdot n(x, y) \tag{2.19}$$

2.8 Classification Predictive Modeling

Classification can be defined as a task performed by using specific machine learning algorithms that learn the way of assigning a class label to the input features [19]. In supervised machine learning, the data set contains both input variables and output variables and a learning algorithm learns a mapping from the input to the output [19]. The input variables and their corresponding outputs are commonly referred to as features and class labels. Clas-

sification is a predictive modeling problem where the class labels are predicted for the given features, for example, predicting a tumor as malignant or benign given the properties of the tumor [1]. From a modeling perspective, the process of classification requires training data which is composed of features and class labels. The training data must be sufficient in volume in order to represent all possible aspects of a classification problem appropriately [14]. Some of the major machine learning predictors for classification are Neural Network, k-Nearest Neighbors, Decision Trees, and Random Forest. When dealing with complex problems such as image classification, natural language processing, and speech recognition, often Neural Networks are the predictors of choice [1].

The Neural Network model is evaluated based on training loss, training accuracy, and test accuracy. The training loss and training accuracy are estimated throughout the training process for each computational cycle. In a computational cycle, the model performs all necessary computations on the entire training data set [14]. At the end of a computational cycle, the training loss of the model for that particular cycle is estimated using a loss function (Section 4.3). For estimating the training accuracy, the partially trained model at that particular cycle makes predictions on the training data and then compares the predictions with the ground truths which results in values within the range of 0 to 1. Therefore, multiple training loss values and training accuracy values are estimated depending on the total number of defined computational cycles throughout the training process [14]. The values of the training loss and the training accuracy estimated in the last computational cycle are considered as the final values. The test accuracy is estimated when the trained model performs the predictions on the test data. To estimate the test accuracy, the trained model makes predictions on the test data and then compares the predictions with the ground truths.

Chapter Three

SPECTRUM CONSTRUCTION AND ANALYSIS

In order to classify catheters and echoes in an ultrasound image using frequency spectrum analysis, it is necessary to obtain the Fourier transforms from local regions of interest. The classic method for local frequency spectrum analysis is the Gabor transform [20]. It applies a Gaussian function to a local image region as a window and the standard Fourier transform is applied to the masked local image region. As shown in [20] for the first time, the Gabor transform is the result of convolution in the frequency domain between the Fourier transform of the local image content and the Fourier transform of the Gaussian mask function. Therefore, the problem of separating the Fourier transform of the local image content from the Gabor transform is a de-convolution problem. However, this is fundamentally different from the traditional de-convolution problem as reviewed in Chapter 2. The traditional problem is that the convolution happens in the spatial domain, i.e., $f(x) \otimes g(x)$. In our problem, the convolution happens in the frequency domain, i.e., $F(u) \otimes G(u)$. The traditional de-convolution methods are meaningless in our case. That means, we must perform de-convolution directly in the frequency domain, rather than indirectly in the spatial domain.

The probing detector proposed in [4] solved this problem by introducing a novel de-convolution method for reconstruction of the frequency coefficients of $F(u)$ iteratively in the

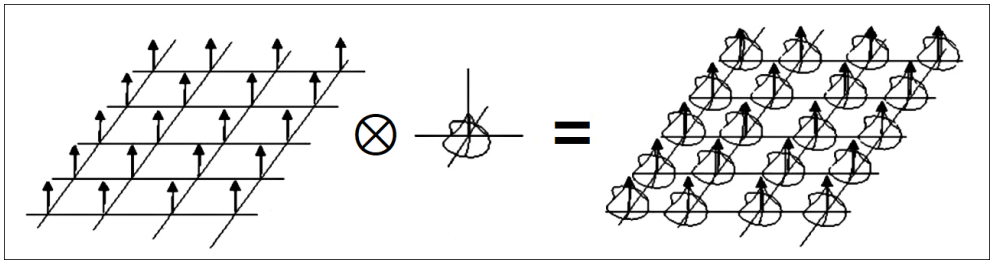
order of magnitude of the coefficients. The frequency coefficients are analyzed and several features are derived from it. Those features are used to train a Neural Network for the purpose of the classification of catheters and echoes.

3.1 Probing Detector

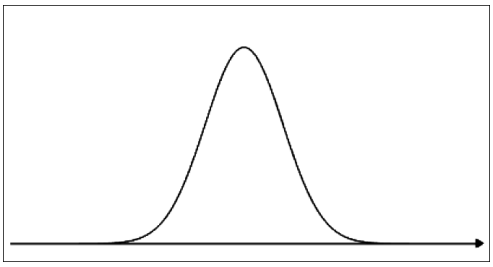
The probing detector [4] uses a Gaussian function (Section 2.5) as a window function similar to the Gabor transform [20]. According to the convolution theorem, a simple dot operation between an image $f(x, y)$ and a Gaussian window $g(x, y)$ in the spatial domain is equivalent to a convolution operation, $F(u) \otimes G(u)$, in the frequency domain [9]. This is illustrated by Figure 3.1(a). On the left hand side, a matrix of discrete values is the Fourier transform $F(u, v)$ of local $f(x, y)$ under the Gaussian window and the other is $G(u, v)$ which is the Fourier transform of the Gaussian function $g(x, y)$. The right hand side is the result of the convolution between $F(u, v)$ and $G(u, v)$. Suppose the dot operation between $f(x, y)$ and $g(x, y)$ results in $h(x, y)$ ($h(x, y) = f(x, y) \cdot g(x, y)$). The Gabor transform [20] is the Fourier transform of $h(x, y)$, thus $H(u, v) = F(u) \otimes G(u)$. The objective of the probing detector is to separate $F(u, v)$ from $H(u, v)$, through a de-convolution process directly in the frequency domain.

To better understand the concept of probing detector, a simpler 1D example is given in Figures 3.1(b) to 3.1(d). Figure 3.1(b) shows the Fourier transform of a Gaussian function $G(u)$, 3.1(c) shows the Fourier transform of an input signal $F(u)$, and 3.1(d) shows the convolution result between $F(u)$ and $G(u)$ which is denoted by $H(u)$. Note that the Fourier transform of a Gaussian function is still a Gaussian function.

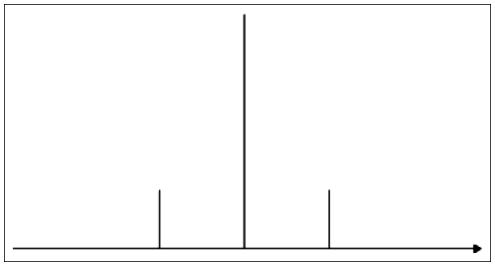
Given a convoluted signal $H(u)$, the goal of the probing detector algorithm is to reconstruct the original signal $F(u)$. The probing detector algorithm estimates each element of $H(u)$ based on the order of magnitude of the frequency coefficients in $H(u)$. First, we prespecify the number N of frequency coefficients to be reconstructed. Afterwards, one searches



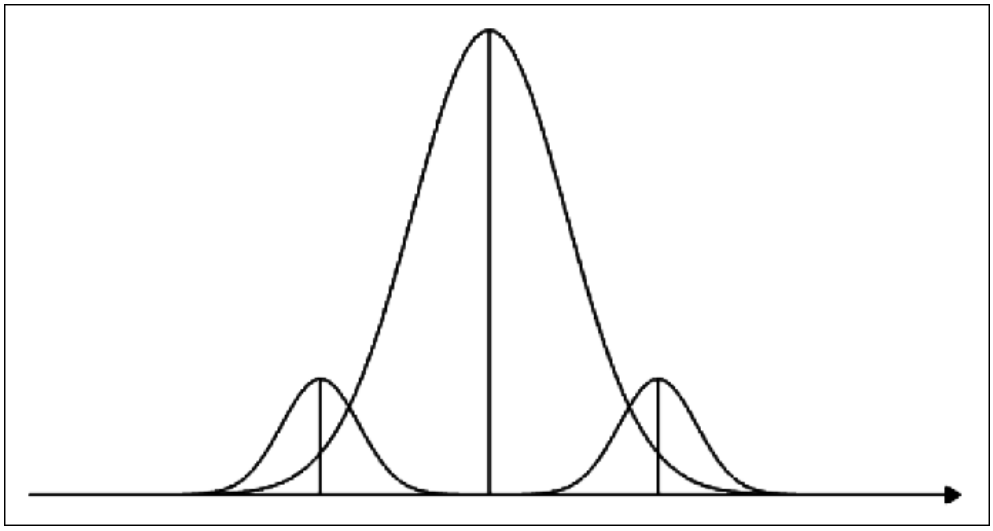
(a) Convolution visualization for a two-dimensional image signal.



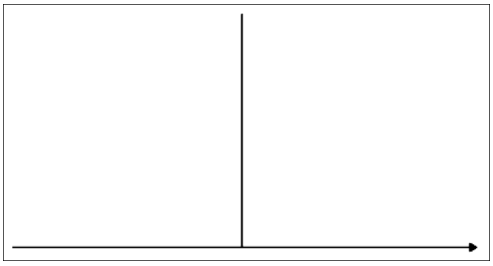
(b) Fourier transform of Gaussian function.



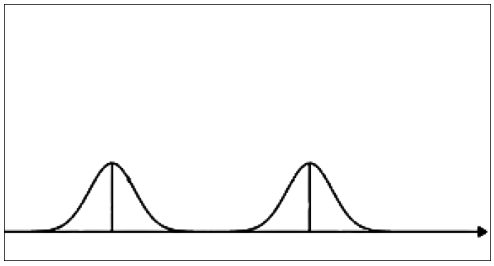
(c) Fourier transform of an input signal.



(d) Convolution of (b) and (c).

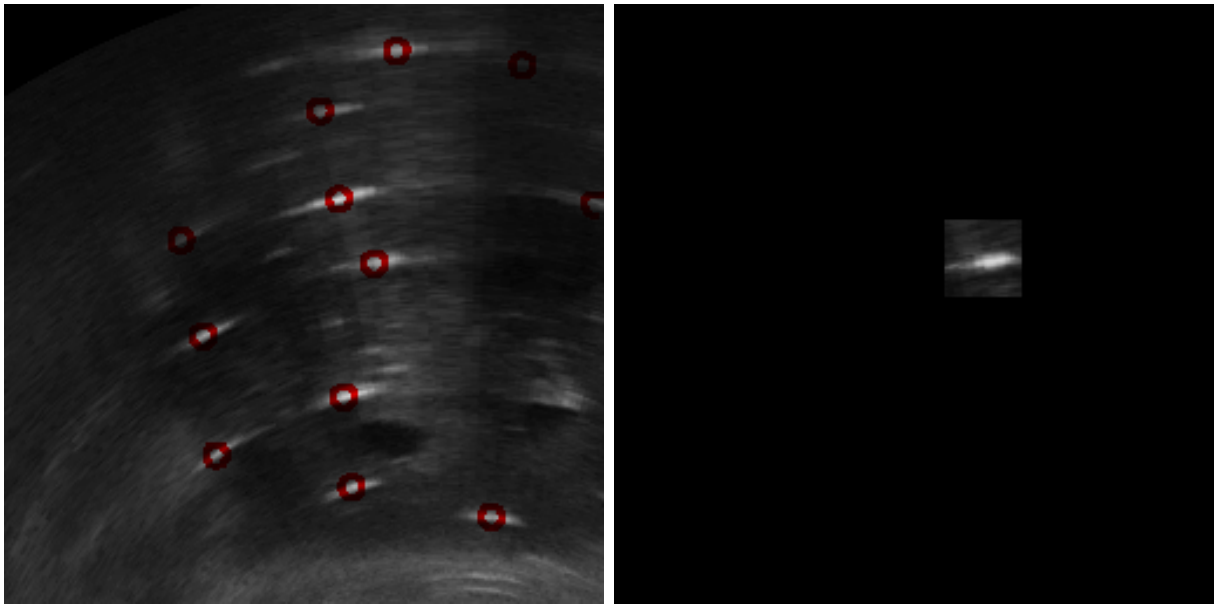


(e) First reconstructed spectrum.



(f) Remaining $H(u)$.

Figure 3.1 Visualization of convolution process, source [4].



(a) Labeled catheters. **(b)** A catheter as region of interest.

Figure 3.2 Catheters in an ultrasound image and a pooled region of interest.

through $H(u)$ for the frequency coefficient with the largest magnitude and its corresponding location u. The largest value of $H(u)$ is then stored in $F(u)$ at the same location u (as shown in Figure 3.1(e)). Then, its contribution in $H(u)$ is removed by the following two steps. First, $G(u)$ is scaled using the largest frequency coefficient as a scalar; and second, the scaled frequency coefficients of $G(u)$ are then subtracted from $H(u)$. Before subtracting, the center of the Gaussian kernel and the largest frequency coefficient have to be in the same position.

One iteration is completed when the subtraction process is done. The above process is repeated N times. For two-dimensional ultrasound images, applying probing detector de-convolution to a region of interest reconstructs the original frequency coefficients. The computational steps of the complete probing detector algorithm [4] for the two-dimensional case are as follows.

> **Input:** Image $f(x, y)$, Gaussian window $g(x, y)$, probing location (x_0, y_0), window size σ.

Output: Reconstructed frequency coefficients $F(u, v)$.

Step 1: Initialize a two-dimensional array $F(u, v)$ and set the number N of total iterations.

Step 2: Compute a Gaussian window $g(x, y)$ given the probing location (x_0, y_0) and window size σ.

Step 3: Compute the dot product of $f(x, y)$ and $g(x, y)$, denoted as $h(x, y)$. $h(x, y) = f(x, y) \cdot g(x, y)$

Step 4: Compute $H(u, v)$, the Fourier transform of $h(x, y)$.

Step 5: Compute $G(u, v)$, the Fourier transform of the Gaussian window $g(x, y)$.

Step 6: For each iteration:

(a) Find the largest frequency coefficient in $H(u, v)$ and store it in $F(u, v)$.

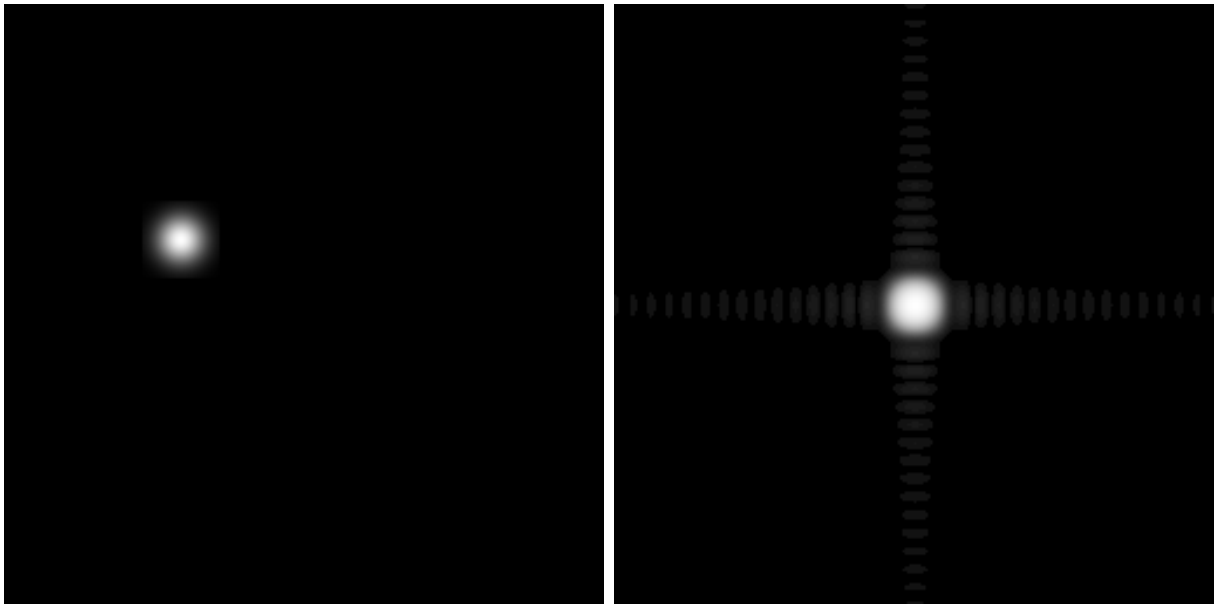
(b) Scale every element of $G(u, v)$ using the largest frequency coefficient as a scalar and store it in $G'(u, v)$.

(d) Remove $G'(u, v)$ from $H(u, v)$.

Step 7: Return $F(u, v)$.

3.2 Ultrasound Image Spectrum Construction

Ultrasound images recorded during brachytherapy of prostate cancer patients have four labels: Catheter, Prostate, Rectum, and Urethra. In this thesis, our primary focus is on catheters and their echoes. Echoes are created at the time of capturing catheter images due to the scattering property of the ultrasound signal [21]. Both catheter and echo are considered as regions of interest. The center of a catheter or echo is considered as the center of a region of interest. An ultrasound image with labeled catheters is shown in Figure 3.2(a) and a pooled region of interest is shown in Figure 3.2(b). A Gaussian window of size 33×33 pixels is used for the region of interest pooling. When selecting a Gaussian window, it is necessary to consider two major criteria: (i) the window should capture the maximum bright



(a) A Gaussian window. **(b)** Spectrum of the Gaussian window.

Figure 3.3 Representation of a two-dimensional Gaussian function and its magnitude.

region of a catheter or an echo and (ii) it should capture less fainted surrounding region of a catheter or an echo. Most of the catheters and echo fit well inside of a Gaussian window of size 33×33. The region of interest pooling retains the image content within the Gaussian window and sets all the pixels outside of the window to zero.

Figure 3.3(a) and Figure 3.3(b) show a Gaussian window and the Fourier transform of the Gaussian window. The dimension and center of the Gaussian window are the same as for the window used during the region of interest pooling.

Before performing the iterative step (Step 6) in the probing detector De-Convolution, we specify the total number of iterations N to be 30 in this case. The reason for choosing $N = 30$ iterations is that after 30 iterations, the value of the frequency coefficient gets close to 0 as shown in Table 3.1. Upon completion of the first iteration, the largest frequency coefficient is found at the center $(u = 0, v = 0)$ as the spectrum intensity is maximal in this position. Figure 3.4 shows the reconstructed frequency coefficients for $N = 30$ iterations and Table 3.1 shows the frequency coefficient values and their associated coordinates.

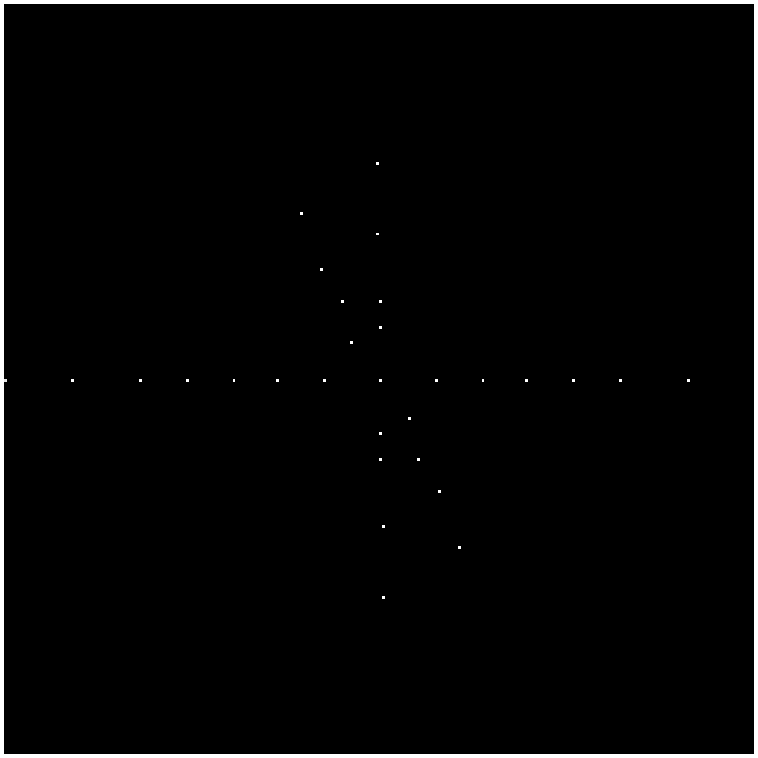


Figure 3.4 Reconstructed frequency coefficients.

From Table 3.1, we can see that after one iteration, the frequency coefficient value drops drastically from 0.834244 to 0.176501. After 10 iterations, the frequency coefficient value gets substantially smaller (0.047875). This suggests that the first 30 frequency coefficients can represent the original ultrasound signal with a very small error rate which is negligible for our purpose.

3.3 Feature Map Computation

The frequency coefficient table (given in Table 3.1) shows the frequency coefficients at particular coordinates as the output of the probing detector de-convolution. In Section 3.4, plotting a set of spectrum coordinates showed that catheter and echo have different characteristics in scattering. It suggests that the coordinates may help, along with the frequency coefficients, to distinguish catheters from their echoes. Some new features can be computed

Iteration	Coordinate	Coefficients
1	(0,0)	0.834244
2	(-18,-1)	0.176501
5	(-34,1)	0.100592
10	(-50,1)	0.047875
15	(0,-51)	0.030766
20	(-81,0)	0.020808
25	(0, 120)	0.017281
30	(112, 1)	0.006152
35	(-97, 0)	0.002007
40	(43,-13)	0.001161

Table 3.1 Reconstructed frequency coefficients and their positions.

using these coordinate values, namely *spectral rotation angle*, *spectral deviation angle*, *sector index*, and *distance from the origin*. In what follows, these four features will be explained in detail.

3.3.1 Spectral Rotation Angle

Spectral rotation angle is a measured angle that shows how much a frequency coefficient is anti-clockwise rotated from its positive x-axis. Computing *spectral rotation angle* is equivalent to computing phase angle in the frequency domain (as shown in Figure 3.5). The frequency coefficients are scattered in a two-dimensional plane. Due to this scattering, the values of *spectral rotation angle* are different for each frequency coefficient unless they are located on the same row.

A large fraction of the frequency coefficients for most of the catheters are scattered away from the x- and y-axis. On the other hand, the majority of the frequency coefficients for

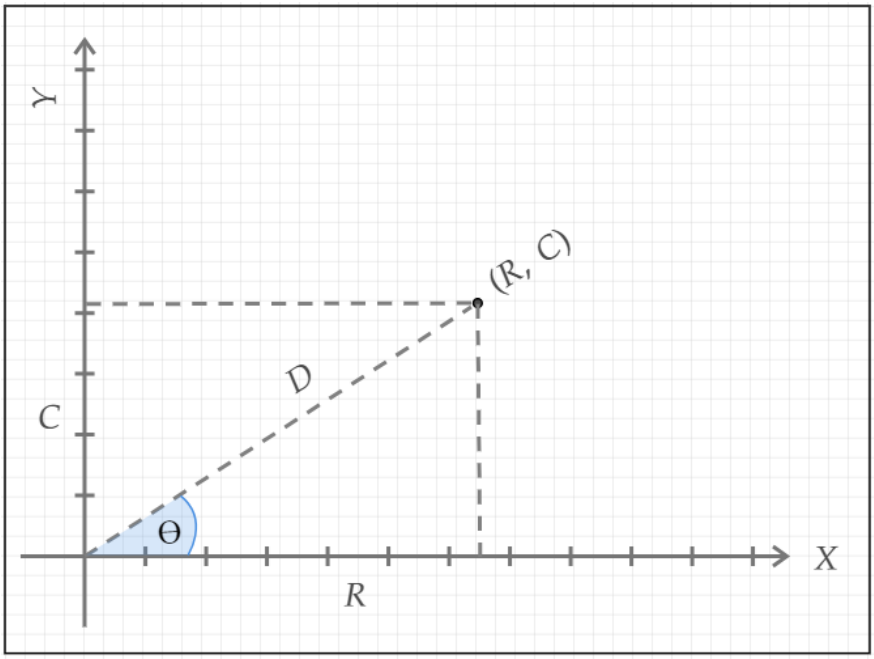


Figure 3.5 Computation of spectral rotation angle.

most of the echoes are scattered on the axis. For this scattering pattern, *spectral rotation angle* is considered as an important feature for neural network classification in this thesis.

3.3.2 Spectral Deviation Angle

Connecting three consecutive frequency coefficients generates two lines. A *spectral deviation angle* shows how much the second line changes direction compared to the first one. In Figure 3.6, θ_1 is a *spectral deviation angle* and P_1, P_2, and P_3 are three consecutive frequency coefficients. When we compute a *spectral deviation angle*, we always assume that the angle belongs to the second spectrum (P_2). In order to make this feature more significant, we have included a positive or negative direction sign to the angle where a positive value represents anti-clockwise rotation and a negative value represents clockwise rotation.

To compute a *spectral deviation angle*, let us draw two lines X_1 and X_2 that contain frequency coefficients P_1 and P_2. Both X_1 and X_2 are parallel to the x-axis. Therefore, we can compute *spectral deviation angle* (θ_1) by subtracting θ_3 from θ_2 ($\theta_1 = \theta_3 - \theta_2$). Suppose,

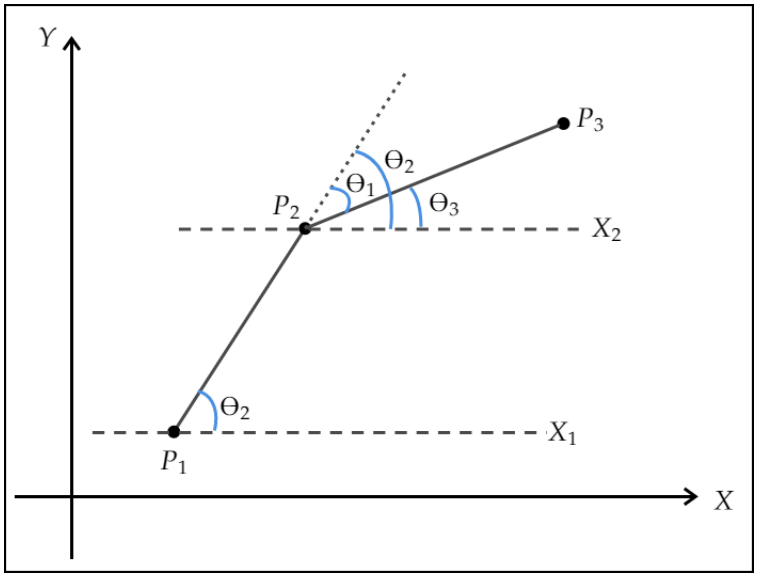


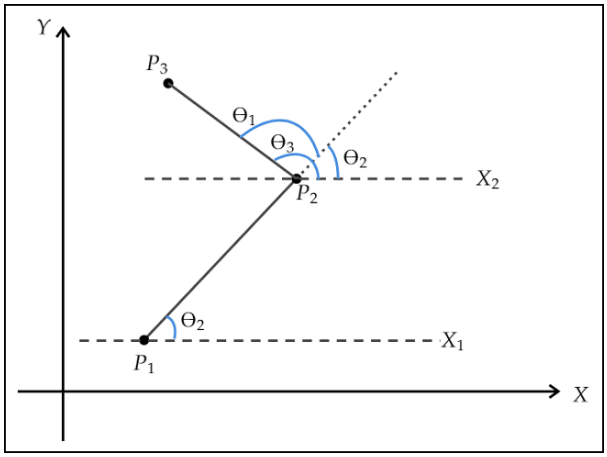
Figure 3.6 Case 1: Computation of spectral deviation angle.

coordinates of the P_1, P_2, P_3, X_1, and X_2 are (x_1, y_1), (x_2, y_2), (x_3, y_3), (x'_1, y'_1), and (x'_2, y'_2). We need to compute the slope of the lines P_1X_1, P_1P_2, P_2P_3, and P_2X_2 to compute the angles θ_2 and θ_3.

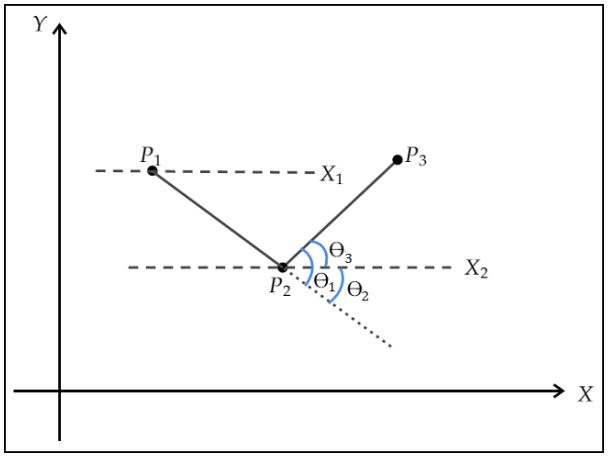
The slope of line P_1X_1 is: $m_1 = \frac{y'_1 - y_1}{x'_1 - x_1}$.
The slope of line P_1P_2 is: $m_2 = \frac{y_2 - y_1}{x_2 - x_1}$.
The slope of line P_2P_3 is: $m_3 = \frac{y_3 - y_2}{x_3 - x_2}$.
The slope of line P_2X_2 is: $m_4 = \frac{y'_2 - y_2}{x'_2 - x_2}$.

The interior angle between lines P_1X_1 and P_1P_2 is: $\theta_2 = tan^{-1}(\left|\frac{m_2 - m_1}{1 + m_1 m_2}\right|)$.
And, the interior angle between lines P_2X_2 and P_2P_3 is: $\theta_3 = tan^{-1}(\left|\frac{m_4 - m_3}{1 + m_3 m_4}\right|)$.

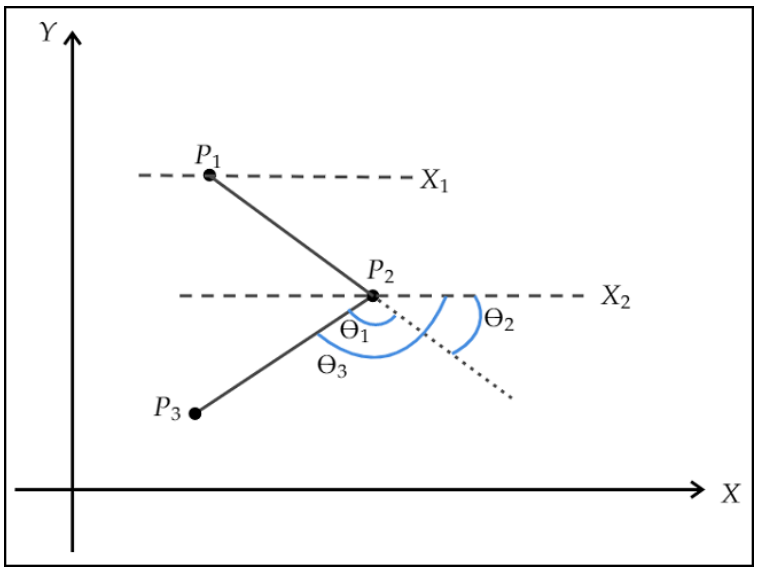
To compute the clockwise or anti-clockwise direction of a *spectral deviation angle*, we need to compute the signs of θ_2 and θ_3. In order to compute the signs of θ_2 and θ_3, we consider X_2 as a local x-axis that contains the second frequency coefficient P_2. If we extend the line P_1P_2, it creates θ_1 between the extended P_1P_2 and P_2P_3, θ_2 between the extended



(a) Case 2: Computation of spectral deviation angle.



(b) Case 3: Computation of spectral deviation angle.



(c) Case 4: Computation of spectral deviation angle.

Figure 3.7 Spectral deviation angle.

P_1P_2 and the local x-axis denoted as X_2, and θ_3 between P_2P_3 and X_2 as shown in Figure 3.7. If θ_2 and θ_3 are located on top of the local x-axis X_2 (Figure 3.6, Figure 3.7(a)), then the angles θ_2 and θ_3 are considered as positive angles. Therefore, the computed θ_1 in this case is negative, representing clockwise direction. On the other hand, if the angles θ_2 and θ_3 are located down the local x-axis X_2 (Figure 3.7(c)), then θ_2 and θ_3 are considered as negative angles. Hence, the direction of the angle θ_1 is anti-clockwise.

In Figure 3.7(b), θ_2 is located down the local x-axis and θ_3 is located on top of the local

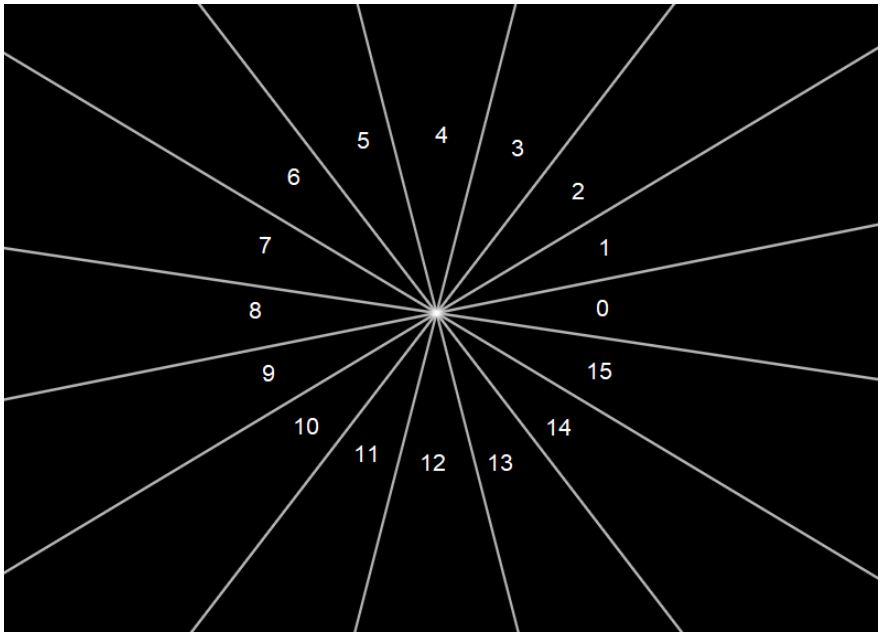


Figure 3.8 Sector index computation.

x-axis. Therefore, θ_2 is negative and θ_3 is positive. If the computed angle θ_1 is positive, then the direction of θ_1 is anti-clockwise. Otherwise, the direction of θ_1 is clockwise.

3.3.3 Sector Index

A *sector index* illustrates which frequency coefficient belongs to which sector in a two-dimensional plane. Sectors are created by dividing a two-dimensional plane into multiple smaller sections. Sector indexing is an approach of classifying the frequency coefficients by the angle defined by $arctan(y/x)$ where (x, y) is a coordinate of a frequency coefficient [11]. To compute the *sector index* of a set of frequency coefficients of a catheter or echo, we have used sixteen sectors as shown in Figure 3.8.

The following equation computes the *sector index* for each frequency coefficient:

$$i = tan^{-1}(\frac{y}{x})/\frac{2\pi}{n} \tag{3.1}$$

Here, i is a *sector index*, (x, y) is a coordinate of a frequency coefficient and n is the total number of sectors ($n = 16$). From a float type output of Equation 3.1, only the integer part

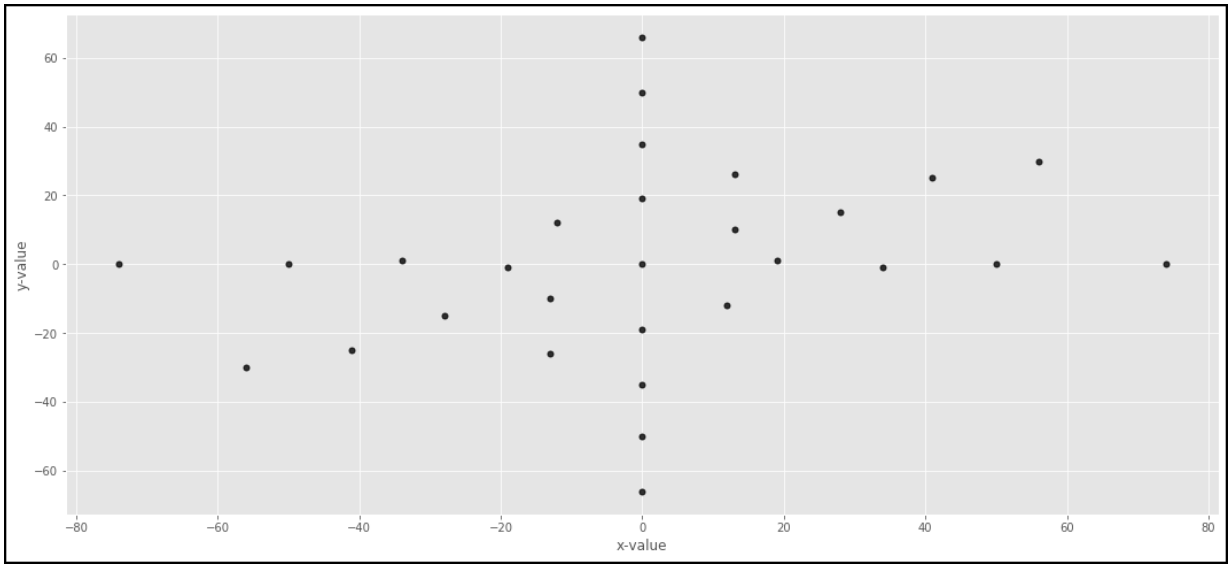
is extracted in order to represent a *sector index*.

3.4 Spectrum Analysis

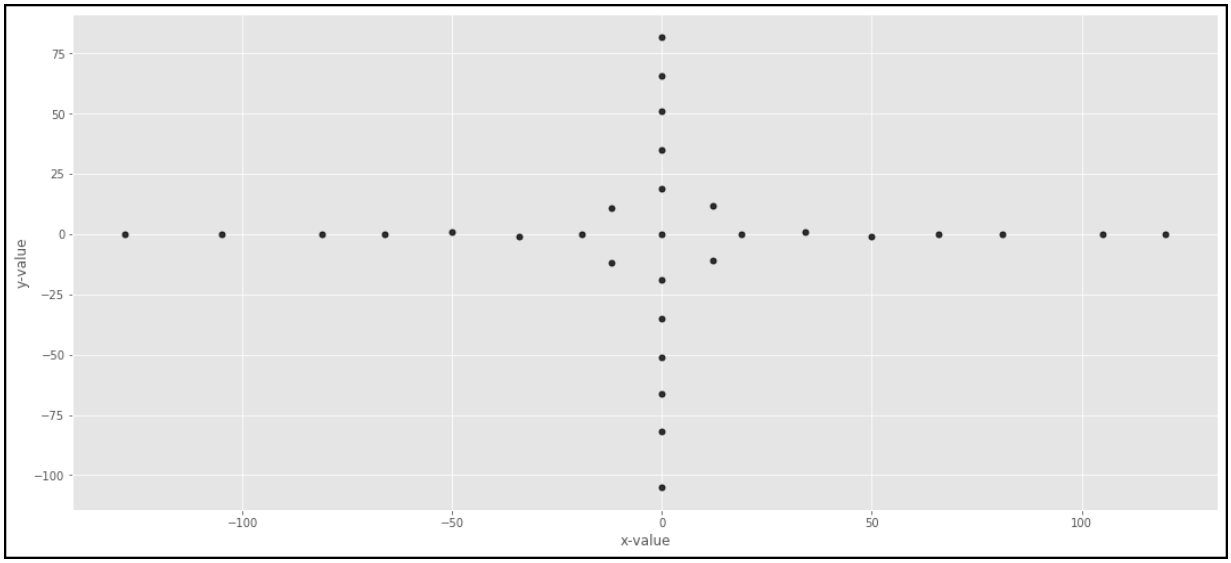
The reconstructed frequency coefficients (Figure 3.4 and Table 3.1) represent the energy distribution for an image in the frequency domain. In image analysis, the frequency coefficient is considered as an important feature along with its position in two-dimensional space. Applying the probing detector algorithm on a region of interest constructs a set of frequency coefficients as shown in Table 3.1. A region of interest can contain a catheter or an echo. Therefore, the de-convolution algorithm generates the frequency coefficients for both catheter and echo as output. To visualize energy distribution, let us plot the frequency coefficients for both catheters and echoes on a two-dimensional plane. Figure 3.9 (a) shows plotted frequency coefficients for a catheter and Figure 3.9 (b) shows plotted frequency coefficients for an echo. Each plotted point represents a constructed frequency coefficient (energy) at that specific position. The scatter plots in Figure 3.9 are generated by using the coordinates in Table 3.1. The amount of energy (value of the frequency coefficients) varies from coordinate to coordinate.

After observing scatter plots of several catheters and echoes, we hypothesize that the frequency coefficients of the majority of catheters are more scattered than those of the majority of echoes as shown in Figure 3.9. For the majority of echoes, frequency coefficients are located mostly on and close to the x- and y-axis. On the other hand, for the majority of catheters, frequency coefficients are located on the x, y-axis and also both close to as well as away from the x, y-axis. However, these patterns are not observed for all catheters and echoes. There are some catheters and echoes that have similar scattering patterns. In Figure 3.10, the scattering pattern of the plotted frequency coefficients of an echo is similar to the scattering pattern of a catheter.

The frequency coefficient scattering can be measured by computing *spectral rotation*



(a) Frequency coefficient of a catheter.



(b) Frequency coefficient of an echo.

Figure 3.9 Plotted frequency coefficient.

angle. It shows how much a frequency coefficient is scattered from the positive x-axis. The *spectral rotation angle* computation is inspired by the computation of the phase angle in the Fourier transform. The details about *spectral rotation angle* are given in Section 3.3.1.

For the purpose of designing a machine-learned classifier to classify regions of interest as catheter or echo, reconstructed frequency coefficients are considered as an important feature because they represent the energy distribution. Nevertheless, this single feature is

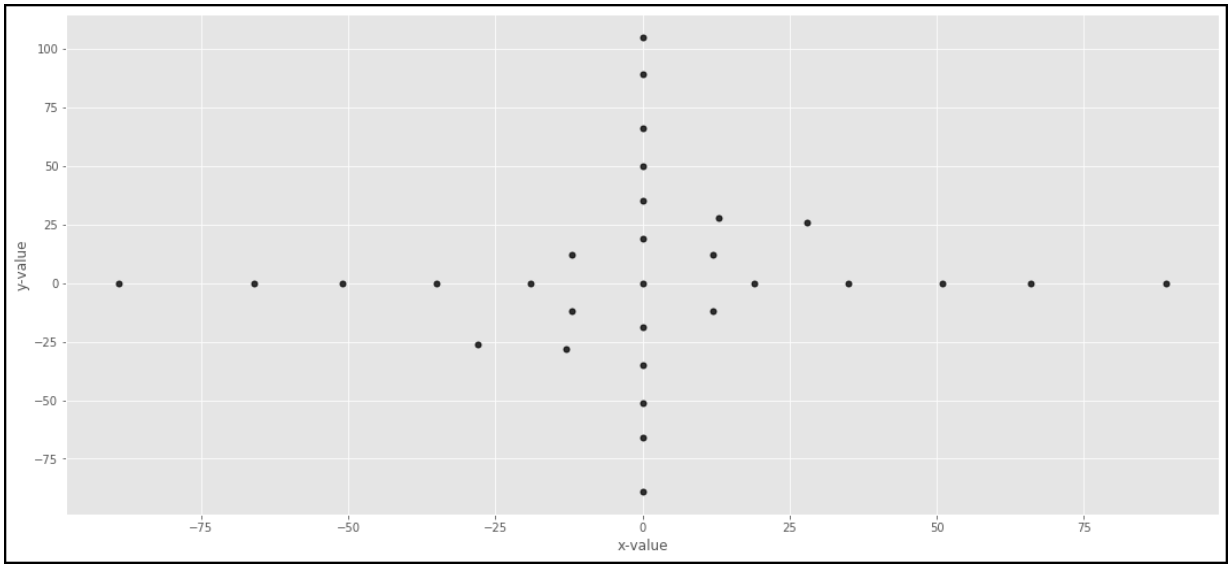
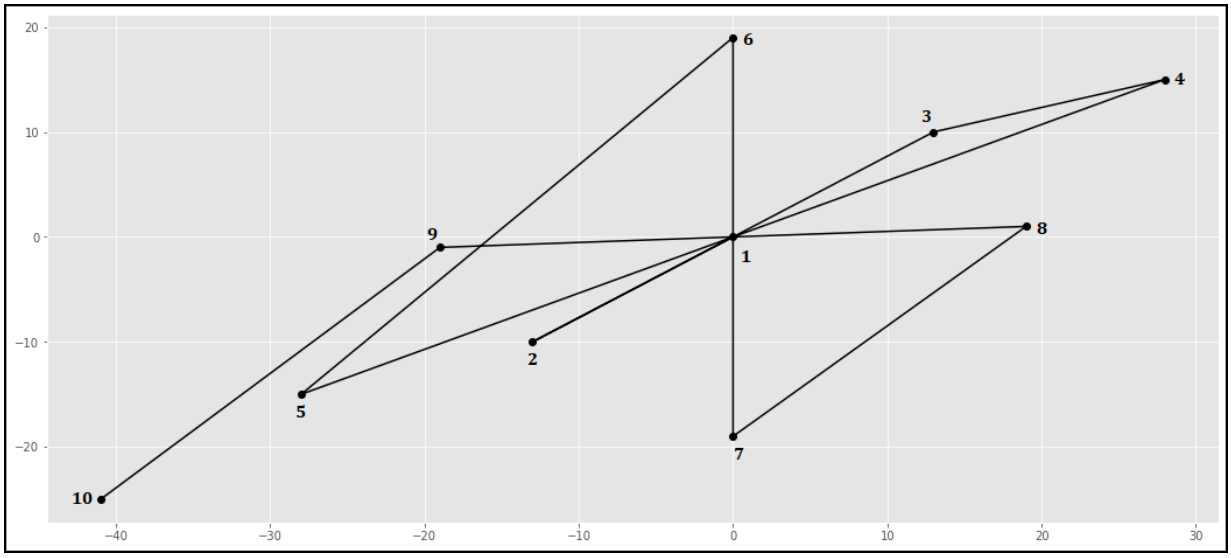


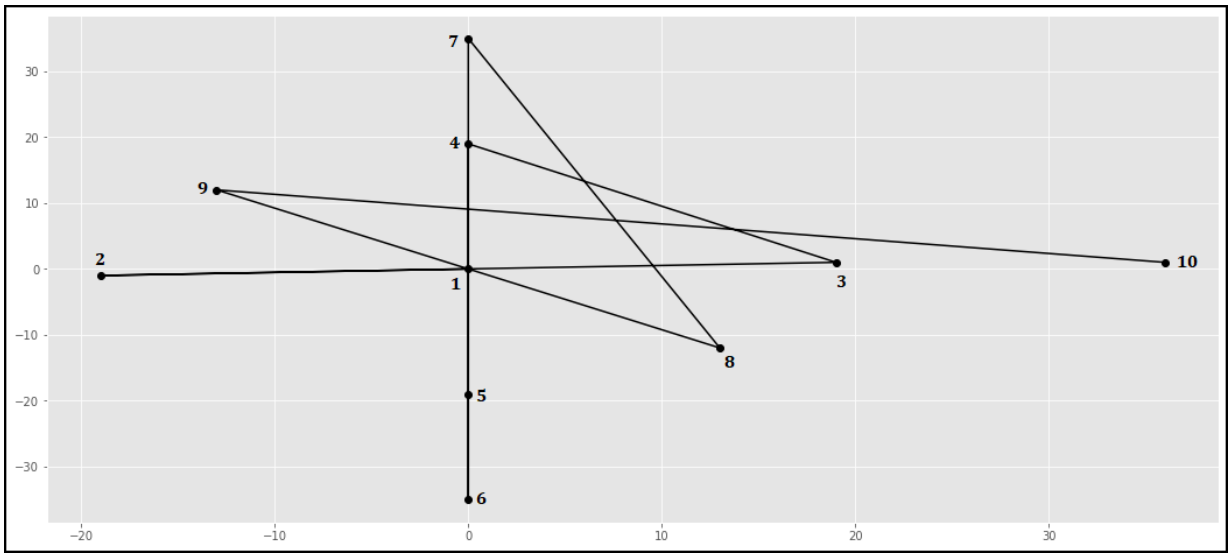
Figure 3.10 Plotted frequency coefficients of an echo.

not sufficient for performing the classification task alone with satisfactory accuracy. Although *spectral rotation angle* is not always able to generate distinct values for some catheters and echoes, it can still be used as a classification feature alongside the frequency coefficient. To conduct the classification experiment, we have used three more features called the *spectral deviation angle* (explained in Section 3.3.2), *sector index* (explained in Section 3.3.3), and *distance from the origin* (explained in Section 3.3.1). Each of the features has a significant role in the classification task. The *spectral deviation angle* is introduced because the *spectral rotation angle* cannot distinguish the scattering property of some catheters and echoes. In order to understand the reason for introducing the *spectral deviation angle* intuitively, we have drawn ten frequency coefficients for both catheter and echo in a two-dimensional plane (as shown in Figure 3.11). Here the indices of the iterations are considered as the order. For example, the frequency coefficient constructed in the 1^{st} iteration has the order 1.

After analyzing the reconstructed frequency coefficients of several catheters and echoes, we observe that the order of the reconstructed frequency coefficients is distinct for each catheter and echo. If we connect three consecutive frequency coefficients and compute how much the first line is rotated towards the second line (Figure 3.6), then the computed angle



(a) Frequency coefficient of a catheter.



(b) Frequency coefficient of an echo.

Figure 3.11 Plotted frequency coefficient.

is always unique for different catheters and echoes because of the order of the frequency coefficients. In Figure 3.11, we can see that the order of the frequency coefficients and generated *spectral deviation angle* after connecting them, are unique for each catheter or echo. The *spectral rotation angle* cannot distinguish the scattering property of some catheters and echoes. But, the *spectral deviation angles* are unique for catheters and echoes even though their scattering pattern is similar. For this reason, the *spectral deviation angle* carries

significant information in terms of distinguishing catheters from their echoes. As we will see later, this information is useful for our classification procedure.

Since the reconstructed frequency coefficients are scattered, they can be classified based on the range of their position. We can imagine this classification approach as performing two sequential tasks: firstly, splitting an entire part of a two-dimensional plane into a finite number of sub-parts; secondly, forming a different number of groups of frequency coefficients that belong to different sub-parts. Each sub-part is known as a sector. The idea here is to compute the *sector index* (Section 3.3.3) for the frequency coefficients as a classification feature. A *spectral rotation angle* is used to compute a *sector index* for a particular frequency coefficient.

Chapter Four

NEURAL NETWORK CLASSIFIER

In this chapter, the architecture of a Neural Network classifier is designed in order to train a predictor that uses our numerical features. The designed Neural Network classifier is named *Probing FCNN* in this thesis where FCNN stands for Fully Connected Neural Network. In Chapter 3, we explained the process of generating a set of frequency coefficients and associated coordinates of a catheter or an echo applying the probing detector de-convolution algorithm. The coordinates of the frequency coefficients are used to compute the numerical features. A total of 40 frequency coefficients along with their coordinates are constructed for each catheter or echo using $N = 40$ iterations in the de-convolution algorithm. As shown in Section 3.2, the reason for using $N = 40$ iterations is that the values of the frequency coefficients are close to zero after 30 or 40 iterations. A coordinate of a frequency coefficient is used to create four features: spectral rotation, spectral deviation angle, sector index, and distance from the origin. For each iteration, the maximum number of computed data points is 5 including the frequency coefficient. Therefore, for $N = 40$ iterations, the maximum number of data points for each catheter or echo is $5N$ which is 200.

In order to design a Neural Network classifier, first of all, we organize the linear units into input layer, hidden layers, and output layer. Secondly, we apply batch normalization [1] on the outputs of each hidden layer. Thirdly, we estimate the network loss and minimize it using the Gradient Descent optimization algorithm [22]. Finally, we apply regularization

techniques to improve the predicted probability for unseen data.

4.1 Linear Units Design

A Neural Network is composed of three types of layers: an input layer, hidden layers, and an output layer [1]. Figure 4.1 shows how the layers are organized in a Neural Network. The input layer holds initial data for a Neural Network and passes the data on to the next layer. This layer does not perform any computation. Therefore, no weights and biases are associated with this layer. The hidden layers are intermediate layers located between input and output layers where all major computations are performed. If a Neural Network contains multiple hidden layers, then the output of the first hidden layer is considered as the input of the second hidden layer and so on. The output layer produces a result for the given input data as a value in the interval $[0, 1]$ which describes how confident the classifier is about a particular prediction [1]. Each node in each layer is known as a neuron. Each neuron in a hidden layer or an output layer is connected with all possible neurons from the previous layer and each connection has a particular weight. The neuron then computes the sum of the weighted outputs from the previous layer and adds a bias [1]. Weights and biases are real numbers and constitute the parameters of the model that have to be learned. In the learning process, they are first initialized (e.g., by random values) and then updated so as to reduce the training error [19].

Let us establish a set of connections between two layers where the input layer contains two neurons and the output layer contains a single neuron. We can think about the second layer as a hidden layer or an output layer. This simple network is known as a Perceptron [1] and is shown in Figure 4.2. In general, the linear relationship between the input neurons and the output neuron is computed using the following equation:

$$y = b + \sum_{i=1}^{n} (w_i x_i) \tag{4.1}$$

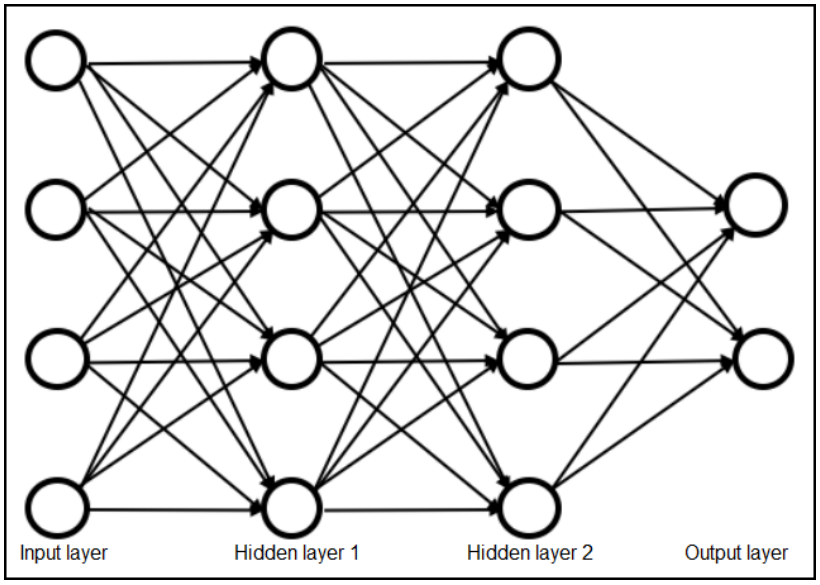


Figure 4.1 The layers in a Neural Network; source [23].

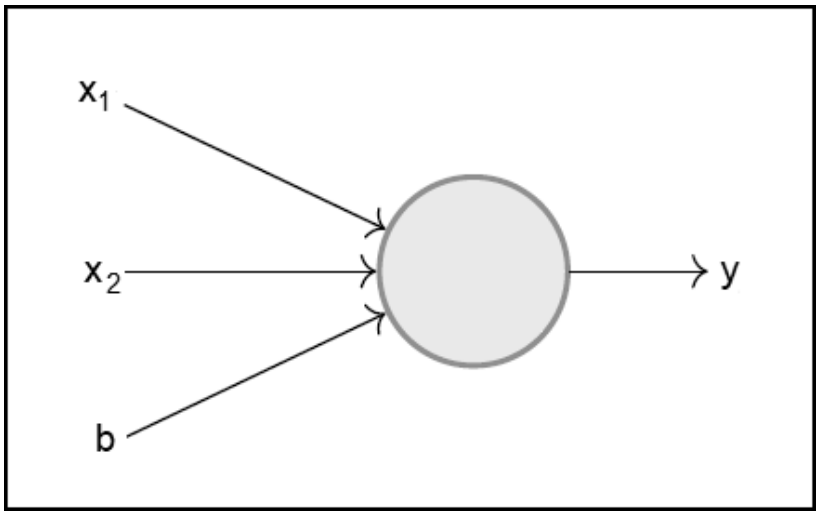


Figure 4.2 A Perceptron; source [1].

Here, y is the output of the network, w_i is the weight of the i^{th} input, x_i is the value of the i^{th} neuron from the previous layer, and b is the network bias. For the network in Figure 4.2, after providing input x_1, x_2 and bias b, the output y is:

$$y = w_1 x_1 + w_2 x_2 + b \tag{4.2}$$

We can build a Neural Network model computing the y-value of different neurons from

different layers using Equation (4.1). Such a model can predict well if the provided data is linearly separable. However, computing such linear relationships among all of the layers also generates a linear output. In such a case, the model is not capable of fitting non-linear data. This problem can be fixed by introducing non-linearity into the Neural Network [19]. We can introduce non-linearity by using activation functions. An activation function is a mathematical operation that normalizes the linear output value (y) of each neuron within a certain range, for example, -1 to $+1$ or 0 to $+1$ [19]. However, some activation functions activate or deactivate a neuron based on the linear output value of a neuron instead of normalizing the value. An activation function always takes y as input. Therefore, computing the linear relationship is mandatory in the first place if we want to apply an activation function. For Backpropagation and Gradient Descent learning (Section 4.4), the output of each neuron must be differentiable in order to estimate the network error and to adjust weight and bias values reducing the estimated error [19]. Activation functions are easily differentiable which makes computations less expensive. Besides non-linearity, run time savings are another reason for using activation functions in Neural Networks. Three types of activation functions are used in this thesis to design the Neural Network architecture: Sigmoid activation function [24], Hyperbolic Tangent ($tanh$) activation function [24], and Rectified Linear Unit (ReLU) [24].

The Sigmoid activation function can be represented with an s-shaped curve (as shown in Figure 4.3) that provides outputs between 0 and $+1$. The form of the Sigmoid activation function $\sigma(y)$ is:

$$\sigma(y) = \frac{1}{1 + e^{-y}} \tag{4.3}$$

Here, y is a linear output of a neuron.

The $tanh$ activation function is also represented by an s-shaped curve as shown in Figure 4.4. Unlike the Sigmoid activation function, which ranges from 0 to $+1$, the range of the $tanh$ activation function is from -1 to $+1$. Equation (4.4) shows the form of the $tanh$ activation

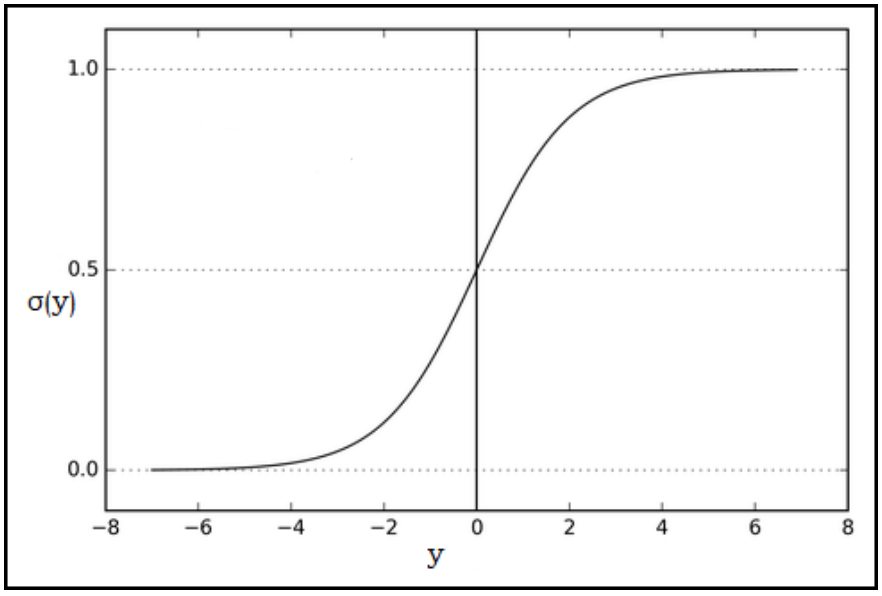


Figure 4.3 Sigmoid activation function; source [24].

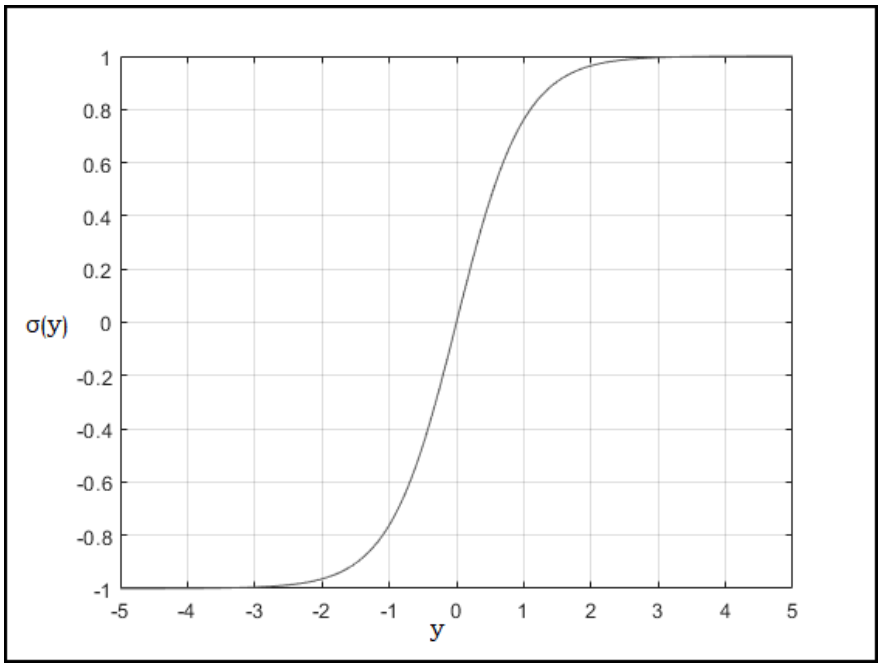


Figure 4.4 Hyperbolic Tangent ($tanh$) activation function; source [24].

function $tanh(z)$:

$$tanh(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}} \tag{4.4}$$

The third activation function used in this thesis to design the Neural Network architecture

is the Rectified Linear Unit (ReLU). The mathematical form of ReLU given in Equation (4.5) shows that the output of the ReLU $Re(y)$ is zero when the value of y is less than zero and the value of $Re(y)$ is y when the value of y is greater than or equal to zero.

$$Re(y) = \begin{cases} 0, & \text{if } y < 0 \\ \\ y, & \text{if } y \geq 0 \end{cases} \tag{4.5}$$

ReLU deactivates the neurons with negative values by turning the negative values into zero. The deactivated neurons cannot participate in the further Neural Network computations. Since the ReLU does not allow neurons with negative values to participate in the computations, it is far more computationally efficient compared to the *tanh* activation function.

The Neural Network which is designed to be fitted with ultrasound image features (computed in Section 3.3) contains four linear layers: an input layer, two hidden layers, and an output layer. The input layer can contain a maximum of 200 neurons. Each hidden layer contains roughly 70 percent of the number of neurons of the previous layer. There is no general rule regarding the number of neurons in a hidden layer. Placing more neurons in a hidden layer than in its previous layer increases computational complexity. Therefore, the first hidden layer has 140 neurons and the second hidden layer has 100 neurons.

The problem of distinguishing catheters from their echoes is a binary classification task where the given input set is classified into two groups. The data set for the classification problem also contains two class labels: 1 for the catheter and 0 for the echo. The Neural Network aims to provide two probability values for these two class labels: the probability of a catheter and the probability of an echo. For this reason, the output layer has two neurons where the first neuron contains the output of a catheter, and the second neuron contains the output of an echo. However, the second output is completely determined by the first one (e.g., $p(x)$). Because the sum of the generated probabilities in the two output

neurons is always 1, the second output can be computed by subtracting the first output from 1 $(1 - p(x))$.

4.2 Data Normalization

Data normalization is the process of scaling an entire data set into a specific range without changing the proportions of differences between data points [25]. A data set of multiple features can contain numerical values of multiple ranges. When the range of a feature is larger than the range of another associated feature, the feature with the larger range might influence the result more significantly even when the feature is less important. For example, Table 4.1 shows the range of different features in our ultrasound data set. Such differences can have a negative impact both in training and prediction. For this reason, data normalization is considered as an important step while designing a Neural Network.

In this thesis, z-score normalization [25] is applied to the ultrasound data set to perform normalization. The z-score normalization technique scales the data maintaining a standard normal distribution with mean or average $\mu = 0$ and standard deviation $\sigma = 1$ in the result. The normalized values (z scores) of a feature x are computed using the following equation:

$$z_x = \frac{x - \mu}{\sigma} \tag{4.6}$$

Here μ is the mean value and σ is the standard deviation. Suppose, one of our ultrasound features, for example, frequency coefficient x_f, has N samples. In this case, the mean is the simple average of N samples (Equation (4.7)). To compute the standard deviation of N samples, first, we compute the sum of the squared distances between each sample and the mean and then we compute the square root of the sum of squares (Equation (4.8)).

$$\mu = \frac{1}{N} \sum_{i=0}^{N-1} x_i \tag{4.7}$$

Feature	Lower End of Range	Upper End of Range
Frequency Coefficient	< 0.0055	0.8
Spectral Rotation (Radian)	0	6.3
Spectral Deviation Angle (Radian)	0	6.3
Sector Index	0	15
Distance from Origin (Pixel)	-128	$+128$

Table 4.1 Ranges of different ultrasound features.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(x_i - \mu)^2} \tag{4.8}$$

The normalization of input features needs to be performed before starting the training process of the Neural Network. We have also applied the normalization to each hidden layer. The advantage of applying normalization on hidden layers is that it speeds up the training process through loss smoothing [19]. We will not explain the loss smoothing process in this thesis; the interested reader is referred to [26] for more information.

4.3 Loss Estimation

A Neural Network is trained iteratively by estimating its loss in output prediction and then reducing the estimated loss from its output. In our Neural Network, we have used the cross-entropy loss [27], also known as the log loss. The reasons for using the cross-entropy loss are the convexity of the objective function and its relatively low sensitivity to outliers [27]. We can think of an objective function as a collection of estimated losses for multiple forward propagations (Section 4.4). The Gradient Descent optimization algorithm

(Section 4.4) is applied to the objective function through the backpropagation process to compute the minimum loss for the Neural Network. After each forward propagation, the Neural Network estimates a loss value and then performs backpropagation to minimize the estimated loss. If the objective function is convex, then the Gradient Descent algorithm can reach the global minimum which is a point in the convex objective function where the estimated loss is minimum [1]. On the other hand, if the objective function is non-convex, the Gradient Descent algorithm may become stuck in a local minimum [1]. Compared to other loss functions for the classification task, for example, Mean Squared Error (MSE), the cross-entropy loss has a greater likelihood to be convex [22].

Secondly, due to a low sensitivity to the outliers [28], the cross-entropy loss is a suitable choice for the classification task. The task of detecting catheters and their echoes in an ultrasound image is a binary classification task. If we fit a model to perform this binary classification, the first output neuron outputs the predicted probability of a catheter, and the second output neuron outputs the predicted probability of an echo. Given that the ground truths of catheters and echoes are known, the Gradient Descent algorithm (explained in Section 4.4) can evaluate how inaccurately the predicted probabilities are estimating the cross-entropy loss. The cross-entropy loss returns a higher value for a prediction with lower probability and a lower value for a prediction with higher probability. The cross-entropy loss for N predictions is computed using the following equation [27]:

$$L(p(x), y) = -\frac{1}{N}\sum_{i=0}^{N-1} y_i \cdot log(p(x_i)) + (1 - y_i) \cdot log(1 - p(x_i)) \tag{4.9}$$

Here, y_i is the ground truth where $y_i = 1$ represents a catheter, and $y_i = 0$ represents an echo, and $p(x_i)$ is the predicted probability of a catheter. Both $p(x)$ and y in $L(p(x), y)$ are vectors of size N. The predicted probability of an echo in Equation 4.9 is determined by the predicted probability of a catheter and given by the value $(1 - p(x_i))$. If the ground truth of a region of interest is $y_i = 1$ (catheter), then the second part of Equation 4.9 $((1 - y_i) \cdot log(1 - p(x_i)))$ is cancelled out and the cross-entropy loss for the catheter is

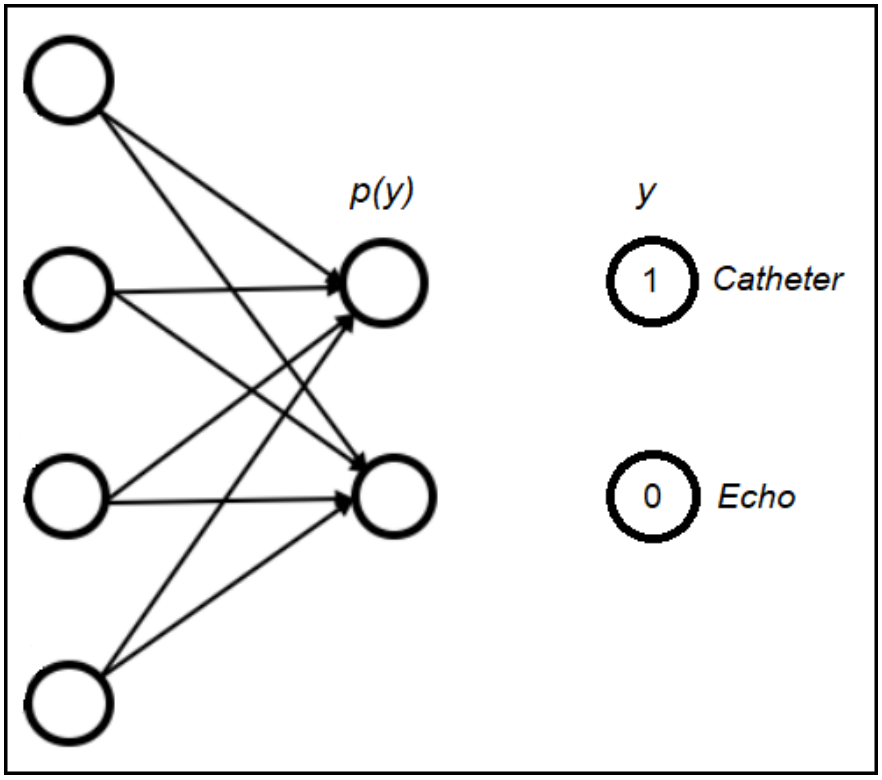


Figure 4.5 Visualization of prediction and ground truth.

computed. On the other hand, if the ground truth of a region of interest is $y_i = 0$ (echo), then the first part of Equation 4.9 ($y_i \cdot log(p(x_i))$) is cancelled out and the cross-entropy loss for the echo is computed.

For each predicted probability in the output layer, an input from which the probability was predicted, has a corresponding ground truth label as shown in Figure 4.5. The computed y values using Equation (4.1) in two output neurons are passed through the Sigmoid activation function. The Sigmoid activation function turns these y values into a probability distribution $p(y)$. Therefore, each of the two output neurons has an individual predicted probability. The predicted probability in the first neuron represents the probability of a catheter. If the predicted probability in the first neuron is a lower value, for example, 0.3, then passing the probability value $p(y) = 0.3$ and ground truth label $y = 1$ to the cross-entropy loss (Equation (4.9)) returns a higher estimated loss. On the other hand, if the predicted probability in the first neuron is a higher value, for example, 0.8, then passing the

probability value $p(y) = 0.8$ and ground truth label $y = 1$ to the Equation (4.9) returns a lower estimated loss. A similar approach is also applied to the second neuron to estimate the loss for an echo.

4.4 Gradient Descent Optimization

Gradient Descent is an optimization technique that computes the minimum loss for a Neural Network through the backpropagation process [22]. Forward propagation is the process of computing the value of y using Equation (4.1) for each neuron in each layer moving forward from the input layer to the output layer through the network [1]. For the first forward propagation, the weight and bias values are initialized randomly. On the other hand, the backpropagation does the reverse. It updates the learning parameters moving backward from the output layer to the input layer [1]. Section 4.3 shows the process of estimating the loss at the end of a forward propagation process. When training a Neural Network, the forward propagation process and the backpropagation process iterate multiple times. Thus, these processes find the minimum loss and estimate the optimal learning parameters. To obtain the optimal values for learning parameters, it is necessary to find the minimum estimated loss, which is the aim of the Gradient Descent [22].

In our Neural Network, the forward propagation process passes through two hidden layers and computes the probability in the output layer. After estimating the cross-entropy loss at the end of the forward propagation process, the backpropagation process moves one step backward from the output layer to the second hidden layer. Suppose w_j^l is a learning parameter. For the learning parameter w_j^l, the Gradient Descent finds the error rate by computing the partial derivative of the estimated loss $L(p(y), y)$ with respect to the learning parameter w_j^l. The error rate is the measurement of the partial deviation of an estimated loss from its minimum value. Each learning parameter (w_j^l) in each layer is updated by reducing the computed error rate from w_j^l, as shown in the following equation [22]:

$$w_j^l = w_j^l - \eta \cdot \frac{\partial}{\partial w_j^l}(L(p(y), y)) \tag{4.10}$$

$$b^l = b^l - \eta \cdot \frac{\partial}{\partial b^l}(L(p(y), y)) \tag{4.11}$$

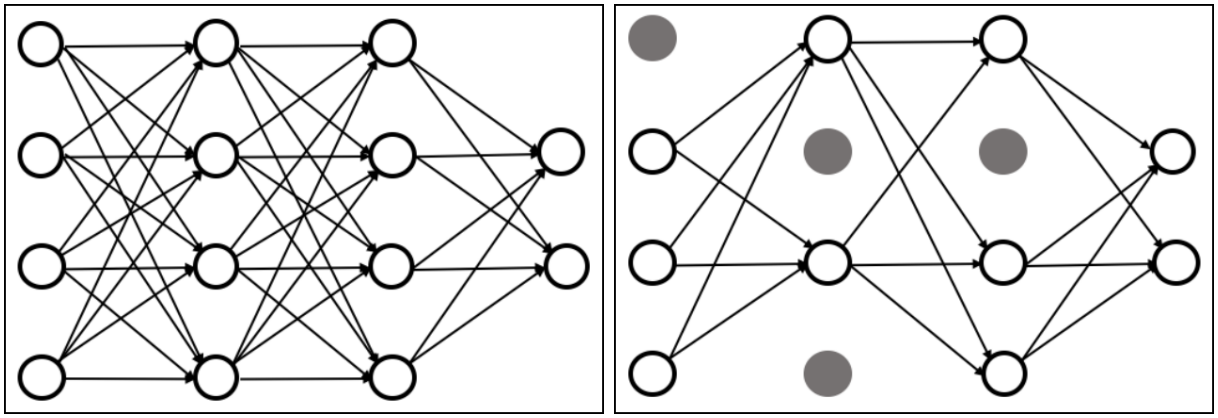
Here b^l is the bias in layer l and η is the learning rate. The learning rate is the tuning parameter in Gradient Descent which defines how fast the cross-entropy loss converges toward the minimum loss [1]. The Neural Network for ultrasound data uses a learning rate of $\eta = 0.001$. For each estimated loss, the backpropagation process keeps iterating until it reaches the input layer.

4.5 Network Regularization

In Neural Network classification, it is necessary to estimate a model that fits both the training and the testing data. A situation in which the estimated model fits well with the training data set but provides a poor prediction for the testing data set, is referred to as overfitting [1]. The regularization techniques prevent overfitting by forcing a Neural Network to be unbiased to the training data set. When a Neural Network is unbiased, the difference between training accuracy and testing accuracy is close to zero. Two major regularization techniques are dropout regularization and L_2 parameter regularization [19]. These two regularization techniques are applied to the Neural Network for ultrasound image classification.

4.5.1 Dropout Regularization

The dropout regularization technique modifies a Neural Network by randomly dropping neurons [23]. When a neuron is dropped, it loses all neural connections with previous and next layers as shown in Figure 4.6. It can be applied both on the input layer and on the hidden layers. If a Neural Network has multiple hidden layers, dropout regularization can be applied



(a) Before applying dropout. **(b)** After applying dropout.

Figure 4.6 Visualization of the dropout effect on a Neural Network; source [23].

to some layers or all of the layers, specifying a hyperparameter in the range $[0, 1]$ for each layer [1]. The hyperparameter is known as the dropout rate. The hyperparameters are the variables that define how the Neural Network will be trained. The values of hyperparameters are tunable. To perform the dropout regularization on a hidden layer, the Neural Network sets the value of the randomly selected neurons to zero during the forward propagation process.

The Neural Network for the ultrasound image data classification has two hidden layers and the dropout regularization is applied on both layers. The total number of neurons in the first hidden layer is greater than the total number of neurons in the last hidden layer. For this reason, the dropout rate is higher in the first hidden layer, where we used a value of 0.30 (30%), compared to the dropout rate in the last hidden layer which is 0.20 (20%). The Neural Network is tested using various dropout rates. The effective dropout rates of 0.30 (30%) and 0.20 (20%) are obtained through trial and error. Chapter 5 explains the details of the experiments and results.

The dropout regularization has an impact on the total number of learning parameters. The learning parameters are the combination of weights and biases from different layers. The dropout regularization reduces the total number of learning parameters due to the reduction of neurons. The total number of learning parameters for each layer is computed using the

following equation:

$$TotalLearningParameters = (L_c \times L_p) + 1 \tag{4.12}$$

Here L_c is the total number of neurons in the current layer, L_p is the total number of neurons in the previous layer, and $+1$ is for one added bias. After applying the dropout regularization, the total number of active neurons in the first hidden layer is 98 instead of 140 and in the second hidden layer is 80 instead of 100. The total number of learning parameters before and after applying the dropout regularization in these two hidden layers are 14,001 and 7,841 respectively. In this way, having fewer learning parameters improves not only the training time but also the accuracy on unseen data, as the resulting simpler model is less prone to overfitting [19].

4.5.2 L_2 Parameter Regularization

L_2 Parameter Regularization [19] is another regularization technique that penalizes the learning parameters of a Neural Network model. The penalizing process is performed by decreasing the values of the learning parameters. The intuition behind L_2 Parameter Regularization is that the learning parameters with smaller values result in a simpler model which leads to a better prediction for unseen data, again due to a lower tendency to overfitting.

L_2 Parameter Regularization is performed by adding the L_2 vector norm at the time of estimating the loss after each forward propagation. The L_2 vector norm computes the distance of vector elements from the origin of the vector. A regularization term λ is multiplied with the L_2 vector norm to regulate its value while adding the vector norm to the estimated loss. The L_2 vector is computed using the following equation for n vector components:

$$\|\omega\|_2 = \left(\sum_{i=0}^{n-1} |x_i|^2 \right)^{\frac{1}{2}} \tag{4.13}$$

Here, $\|\omega\|_2$ represents the L_2 vector norm and x_i are the vector components. In L_2

Parameter Regularization, the L_2 vector norm is added to the estimated cross-entropy loss regularized by the parameter λ as shown in the following equation:

$$L'(p(y), y) = L(p(y), y) + \lambda \cdot ||\omega||_2 \tag{4.14}$$

The range of the regularized parameter λ is $[0, \infty)$. This parameter is also known as weight decay. The selection of the value of weight decay is explained in Chapter 5.

Chapter Five

EXPERIMENTAL RESULTS AND ANALYSIS

We start by explaining the training of the Neural Network classifier in detail. Secondly, we describe how the trained classifier is applied to the test data set to evaluate it in terms of accuracy. Thirdly, the predicted confidence values of *Probing FCNN* and the predicted confidence values of another ultrasound image segmentation model trained by S. Tupor using a deep learning architecture are compared to test whether combining these two classifiers improves the overall accuracy of either individual model. Both Neural Network classifiers are trained and tested on the same ultrasound image data provided by the Saskatchewan Cancer Agency. The shape of the input data and the architecture of these two classifiers are different.

The five features introduced in Chapter 3 are considered as the input features for the *Probing FCNN* classifier. These features are extracted from the ultrasound images applying the probing detector de-convolution and a few other image processing techniques (Section 3.3). The input data for both training and testing is selected randomly from around 48,000 extracted data (regions of interest) of catheters and echoes.

#	Frequency Coefficients	Spectral Rotation	Spectral Deviation	Sector Index	Distance from Origin
1	0.898911	0.000000	-1.302566	0	0.000000
2	0.186340	-0.218669	0.708626	0	18.439089
3	0.185555	-0.218669	0.437338	0	18.439089
4	0.110231	-0.321751	0.682317	0	31.622777
5	0.110164	0.0721751	0.141897	1	31.622777
6	0.100436	0.060423	0.030294	1	19.000000
7	0.099856	0.000000	0.000000	0	19.000000
8	0.066827	-0.293589	0.000000	0	44.922155
9	0.066812	0.893589	0.933359	7	44.922155
10	0.050743	0.000000	0.766496	0	35.000000

Table 5.1 Sample values of the computed features for a catheter.

5.1 Feature Selection

The five extracted features from ultrasound images that are used for training and testing the *Probing FCNN* classifier are *frequency coefficients*, *spectral rotation angle*, *spectral deviation angle*, *sector index*, and *distance from the origin*. Table 5.1 shows ten tuples of sample values of these features for a catheter.

Not all of these features are used as input to the classifier since this would have a negative impact on the prediction accuracy, as can be seen from our experiments with feature map 1 below. The feature selection process is performed through the trial and error method. We choose six different feature maps and pick the one that yields the best prediction performance. For example, the second feature map is created using *frequency coefficients*, *spectral rotation angle*, and *spectral deviation angle*, and the third feature map is created using *frequency coefficients*, *spectral deviation angle*, and *sector index*. The *Probing FCNN* classifier is trained

#	**Configurations**
Feature map 1	All 5 features
Feature map 2	frequency coefficients, spectral rotation, spectral deviation angle
Feature map 3	frequency coefficients, spectral deviation angle, sector index
Feature map 4	frequency coefficients, spectral deviation angle, distance
Feature map 5	frequency coefficients, sector index, distance
Feature map 6	frequency coefficients, spectral deviation angle, sector index, distance

Table 5.2 Different feature maps created by combining different features.

separately using these two training feature maps. Two different testing feature maps are then used to compute the accuracies of two trained classifiers. Analyzing these accuracies, it has been decided which feature combinations result in the highest accurate predictions.

To create both the training and testing feature maps, the randomly selected data set is split into 40% training data, and 60% testing data. The training and testing feature maps are then created by selecting the target features. Table 5.2 shows the configurations of different feature maps created by combining different features. The obtained accuracy for each feature map is shown in Table 5.5.

5.2 Hyperparameters Settings

In the *Probing FCNN* classifier designed in Chapter 4, the total number of layers and the total number of neurons in each layer are smaller compared to the typical Neural Network

Hyperparameters	Attempted Values	Assigned Value
Epochs	50, 100, 150, 200	100
Learning rate	0.001	0.001
Weight decay	0.1, 0.01, 0.001, 0.0001	0.001
Dropout rate	(0.40, 0.30), (0.30, 0.20), (0.20, 0.10)	0.30, 0.20

Table 5.3 Used hyperparameters and their assigned values.

classifier for image classification. Such smaller numbers of layers and neurons create fewer learning parameters during the training process. The classifier is trained following the batch learning approach and using GPU computation. In batch learning, the training data is split up into n_b batches. The batch learning approach helps to find the minimum estimated training loss faster, thus reducing the training time [19]. The *Probing FCNN* classifier is trained using one GPU computation. Using multiple GPU computations in parallel, the training time can be reduced more.

The hyperparameters used for training the *Probing FCNN* classifier are number of epochs, learning rate (Section 4.4), batch size, weight decay (Section 4.5.2), and dropout rate (Section 4.5.1). The hyperparameters are the variables that define how the Neural Network will be trained. An epoch refers to one full computational cycle during the training process. In one full computational cycle, the Neural Network performs the forward propagation, loss estimation, and backpropagation through the entire training data set [1]. The hyperparameters and their assigned values are shown in Table 5.3.

The Neural Network classifier is trained by assigning different values to the number of epochs, weight decay, and dropout rate. For a higher number of epochs, for example, 150 and 200 epochs, the classifier attains a relatively high training accuracy of up to 0.87. However, the trained classifier for such a high number of epochs results in a lower test accuracy, here around 0.64. For both 50 and 100 epochs, the classifier yields consistent training and testing

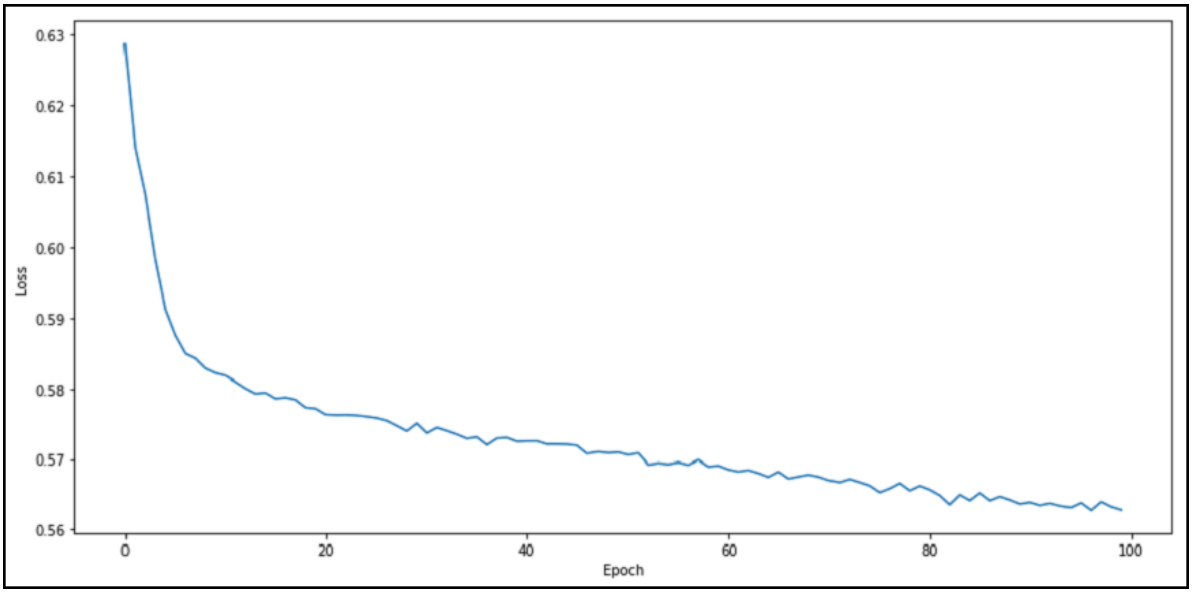
Epoch Index	Training Loss	Training Accuracy
Epoch 1	0.62868	0.68580
Epoch 20	0.57717	0.71055
Epoch 40	0.57249	0.71903
Epoch 60	0.56910	0.72445
Epoch 80	0.56624	0.72741
Epoch 100	0.56282	0.73188
Epoch 200	0.52407	0.87079

Table 5.4 Training history for feature map 4.

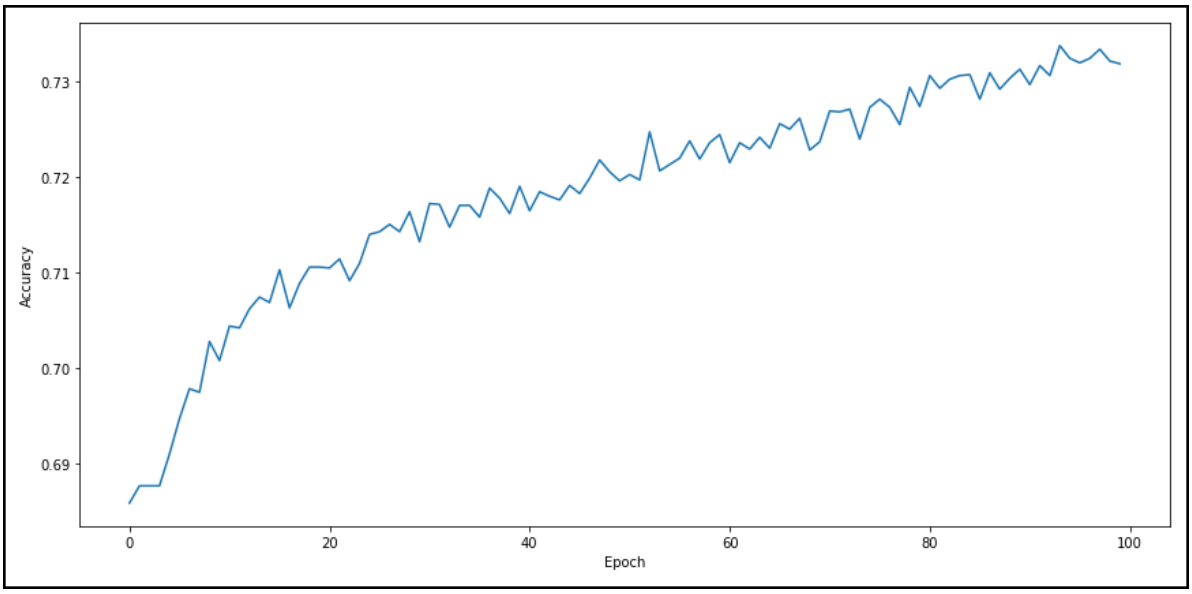
accuracy values. The learning rate defines how fast or slow the Neural Network classifier is trained. The suggested optimal value for learning rate is 0.001 [1].

It is recommended to use a value smaller than 0.1 as the weight decay value. For finding the optimal value of the weight decay hyperparameter, the *Probing FCNN* classifier is trained using four different values: 0.1, 0.01, 0.001, and 0.0001. Assigning a value that is greater than or equal to 0.1 to the weight decay causes a reduction in both training and testing accuracy. Using 0.1 as the weight decay value in the *Probing FCNN* classifier penalizes the learning parameters in such a way that the training accuracy goes down to 0.66. On the other hand, using 0.0001 as the weight decay value has very low significance in terms of penalization. Between 0.01 and 0.001, using 0.001 as the weight decay value in the *Probing FCNN* classifier generates the highest training accuracy (around 0.73) after penalizing the learning parameters.

Because the dropout deactivates the specified number of neurons in each layer, a high dropout rate causes model underfitting where the trained classifier fits very poorly with the training data [1]. The Neural Network is trained and tested using three pairs of dropout rates: $(0.40, 0.30)$, $(0.30, 0.20)$, and $(0.20, 0.10)$. The first and second elements in each pair represent the dropout rate for the first and the second layers, respectively. Using $(0.40, 0.30)$



(a) Epoch *vs* estimated training loss.



(b) Epoch *vs* training accuracy.

Figure 5.1 The plotted diagram for epoch *vs* training loss and epoch *vs* training accuracy for feature map 4.

or higher values as the dropout rates reduce both training accuracy and test accuracy. The dropout rate $(0.30, 0.20)$ generates higher accuracy than $(0.20, 0.10)$. Therefore, $(0.30, 0.20)$ is used as the dropout rate in the Neural Network. Using no dropout at all results in the overfitting problem where the training accuracy goes up to 0.84 and the test accuracy lies

below 0.67 only.

Classifiers are trained using six different feature maps given in Table 5.2, from which we pick the one yielding the maximum test accuracy. We choose feature map 4, which generates an accuracy within 0.72 to 0.74. Table 5.4 shows the training history of the classifier for the feature map 4. The plotted diagram for epoch *vs* estimated training loss and the plotting for epoch *vs* training accuracy are shown in Figure 5.1 (a) and Figure 5.1 (b), respectively.

Figure 5.1 (a) shows the reduction of estimated training loss over the epochs and Figure 5.1 (b) shows the improvement of training accuracy over the epochs. The training loss and training accuracy are computed throughout the training process for each epoch. For computing the training loss after a particular epoch, the obtained predicted probabilities of the training data and its ground truths are plugged into the loss function (Equation 4.9) which results in a training loss for that epoch. The training accuracy after a particular epoch is computed by making predictions using the trained model up to that particular epoch and then comparing the predictions with the ground truths. When the training process is completed, the *Probing FCNN* classifier stores the computed optimal values of learning parameters which are weights and biases. These learning parameters are used for computing the confidence values on the test data and making further predictions on the unseen data.

5.3 Test Results

The dimension and selected features of the test data need to be the same as for the training data to compute a valid confidence value for a region of interest. Suppose, for each catheter and echo, the dimension of the training data is 40×3 where 40 is the number of frequency coefficients constructed by the probing detector de-convolution and 3 is the number of selected features (here *frequency coefficients*, *spectral rotation angle*, and *spectral deviation angle*). To test the Neural Network, the dimension and selected features of test data for each catheter and echo have to be the same. The test results are estimated using the ran-

Feature map	Test Accuracy
Feature map 1	0.66-0.69
Feature map 2	0.68-0.69
Feature map 3	0.71-0.72
Feature map 4	0.72-0.74
Feature map 5	0.71-0.73
Feature map 6	0.69-0.72

Table 5.5 Computed accuracies for six different feature maps.

domly selected test data for the classifier trained with six different feature maps. Table 5.5 shows the computed test results for six different feature maps. The trained *Probing FCNN* classifier is tested using various random samples of size ranging from 8000 to 30,000. The computed test accuracies using these random samples of different sizes for each feature map are different in most cases and always belong to a certain range as shown in Table 5.5. The feature map 4 generates the highest prediction accuracy, using *frequency coefficients*, *spectral rotation angle*, and *spectral deviation angle* as the selected features.

To visualize the test result, the regions of interest are placed back on top of each original ultrasound image. This procedure highlights the catheters and echoes in the rectangular bounding boxes with their predicted label as shown in Figure 5.2. If the predicted region of interest is a catheter, then the corresponding bounding box is labeled as C, otherwise, the predicted region of interest is an echo and the bounding box is labeled as E. In Figure 5.2, out of ten regions of interest, the total number of true predictions is seven and the total number of false predictions is three.

The reconstructed frequency coefficients from ultrasound images applying the probing detector de-convolution algorithm and the computed features from the frequency coefficients constitute a novel contribution in distinguishing catheters from their echoes. The computed highest test accuracy, which is around 0.73, suggests that the reconstructed frequency co-

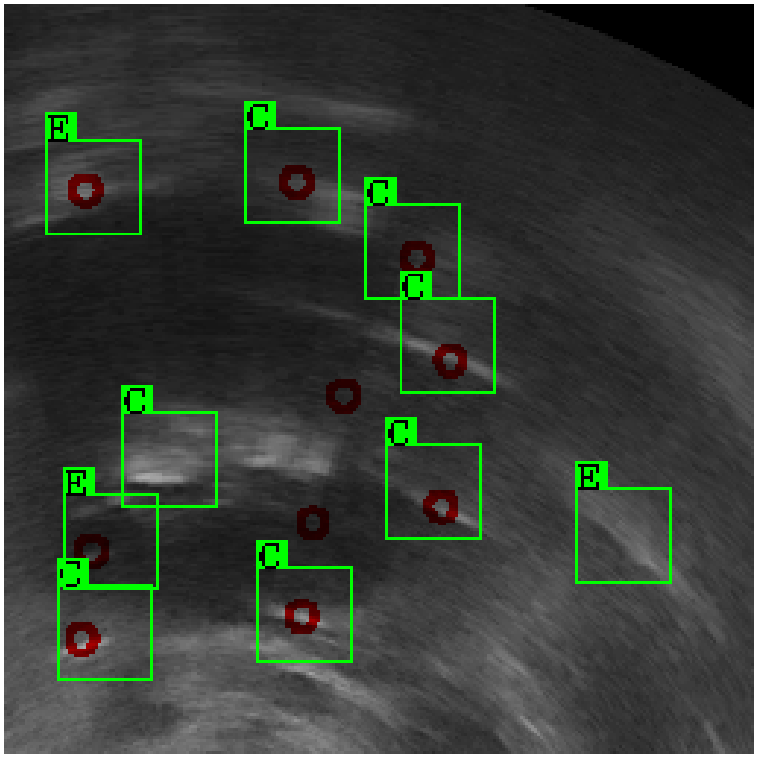


Figure 5.2 Highlighted regions of interest with predicted labels.

efficients and their location in a two-dimensional plane display a particular pattern. When training the *Probing FCNN* classifier using the extracted features, the classifier learns these underlying patterns. The learning process is performed through computing the optimal values of the learning parameters by finding the minimum estimated loss. For a new feature map, each feature is analyzed using these learning parameters to find the underlying pattern and to output the decision in predicted confidence value. The computed accuracy also suggests that the analytical tasks performed on the frequency coefficients and the other extracted features, as described in Chapter 3, is effective in terms of distinguishing catheters from their echoes.

5.4 Combining Probing FCNN with a U-Net Predictor

In a related research project conducted by S. Tupor, the same dataset as analyzed in our study was used to train a classifier that predicts the location of catheters in ultrasound images.[1] This classifier was also a Neural Network, but built using a different architecture; specifically, the *U-Net* deep learning architecture [29], which has shown great successes in medical image analysis, was deployed in this case.

For each pixel in the given input image, Tupor's classifier makes a prediction expressing how likely the corresponding location in the image is part of a catheter. As in the case of the *Probing FCNN* classifier, these predictions can be considered as confidence values.

Lead by the question whether a combination of *Probing FCNN* with Tupor's classifier would yield a more accurate predictor than either individual classifier, we performed a small comparative study. This study used a dataset of 6,660 records provided to us by Tupor in personal communication. Each record contains an ultrasound image I from the Saskatchewan Cancer Agency Dataset used in both Tupor's and our study, as well as two lists of pixel coordinates in the image I. The two lists jointly refer to all and only the locations in the image in which Tupor's classifier predicts centroids of catheters. The first list contains correct predictions, i.e., the locations in this list do in fact correspond to catheters, according to the ground truth labels. The second list contains false positives, i.e., the locations in which Tupor's classifier predicted catheters through the ground truth labels indicate a non-catheter position. For each item in either of the two lists, the confidence value predicted by Tupor's classifier is provided as well. Table 5.6 shows a sample record among the 6,660 provided records. Altogether, there are 79,979 catheter predictions contained in these 6,660 records.

We used the data provided by Tupor as follows. First, the centroid information in these records was used to extract one region of interest for each predicted catheter location

[1]Note that this classifier is not described in any publication yet and is part of ongoing research, so that we cannot provide any explicit details for it in this thesis.

Type	Centroid	Confidence
Catheter	(397, 253)	0.5764
Catheter	(251, 241)	0.6198
Catheter	(299, 230)	0.8076
Catheter	(247, 217)	0.6874
Catheter	(390, 183)	0.5491
Catheter	(436, 181)	0.8800
Catheter	(239, 181)	0.6172
Catheter	(277, 175)	0.7534
Catheter	(278, 126)	0.7442
Catheter	(241, 126)	0.7537
Catheter	(457, 103)	0.6486
Catheter	(267, 96)	0.5446
Catheter	(371, 66)	0.6464
Non-Catheter	(312, 261)	0.5758
Non-Catheter	(425, 228)	0.6634
Non-Catheter	(365, 224)	0.7787
Non-Catheter	(398, 144)	0.5760

Table 5.6 Sample prediction data for a single image, as provided by Tupor's classifier.

(both the true positives and the false positives predicted by Tupor's classifier). To provide appropriate inputs for the *Probing FCNN* classifier, each region of interest was of size 33×33 and the pixels of the underlying 256×256 image were set to zero outside this region of interest, as shown in Figure 3.2(b). A set of features was computed from each extracted region of interest as shown in Table 5.1. These features were then fed to the *Probing FCNN* classifier, which in turn generated a confidence score referring to the likelihood with which the region

Average confidence of Tupor's classifier	0.7093
Standard deviation	0.1147
Average confidence of *Probing FCNN* Classifier	0.8610
Standard deviation	0.1515

Table 5.7 Average confidence and standard deviation over all catheters that Tupor's classifier correctly labelled as catheters.

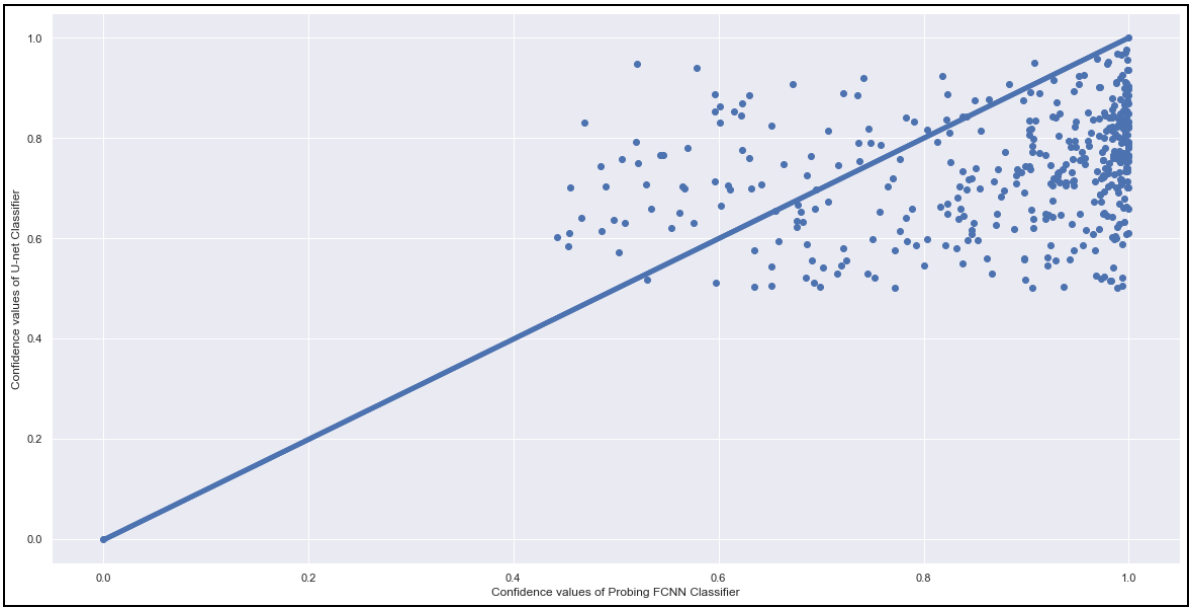
Average confidence of Tupor's classifier	0.6350
Standard deviation	0.1026
Average confidence of *Probing FCNN* Classifier	0.1228
Standard deviation	0.1340

Table 5.8 Average confidence and standard deviation over all non-catheters.

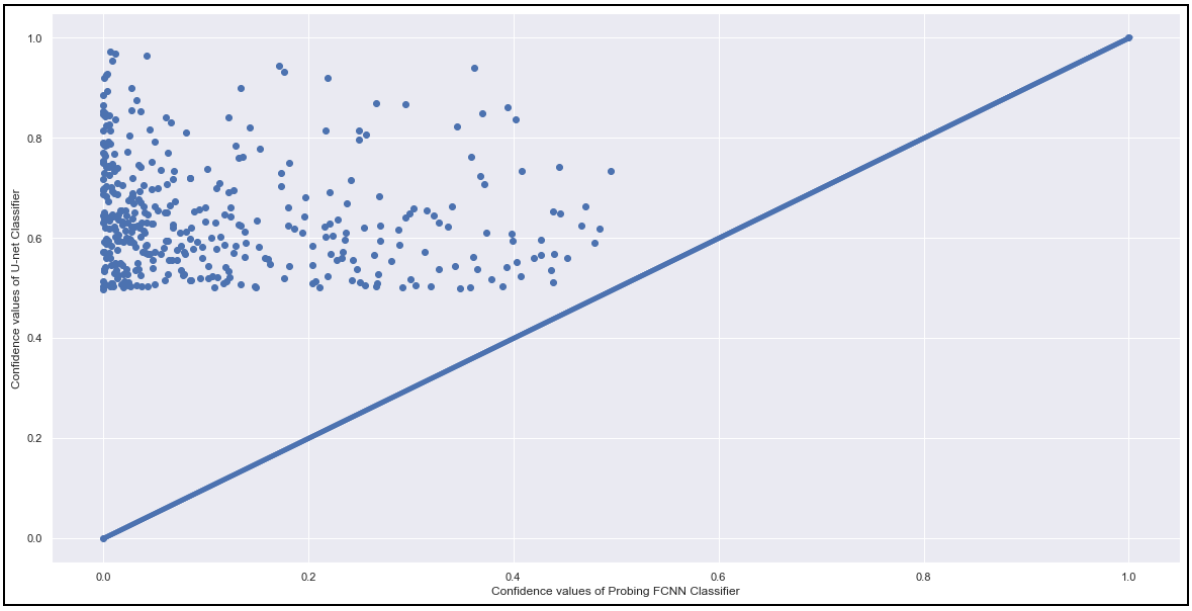
of interest shows a catheter. A number of analytical tasks were performed on the data thus acquired.

First, we averaged the confidence values of both classifiers separately for (i) the image locations that Tupor's classifier correctly predicted as catheters and (ii) the image locations that Tupor's classifiers falsely predicted as catheters. The results, shown in Tables 5.7 and 5.8, indicate that applying *Probing FCNN* after Tupor's classifier can likely be used to improve the accuracy of Tupor's classifier: while *Probing FCNN* on average has a much higher confidence on the locations that Tupor's classifier predicts as catheters, at the same time, it has a drastically smaller average catheter confidence score on the non-catheters that Tupor's classifier considered classifiers. Note that the standard deviation is slightly higher for confidence scores produced by *Probing FCNN* compared to those produced by the *U-Net* classifier. Moreover, for either classifier, predictions are scattered slightly more widely for the true catheters than for the non-catheters.

Second, we tested



(a) A Scatter plot of catheters.



(b) A Scatter plot of non-catheters.

Figure 5.3 Scatter plots for both catheters and non-catheters.

- how often the catheter confidence score returned by *Probing FCNN* was *higher* than the confidence score from Tupor's classifier, when the underlying image location did in fact show a *catheter*;

- how often the catheter confidence score returned by *Probing FCNN* was *lower* than

the confidence score from Tupor's classifier, when the underlying image location did in fact show a *non-catheter*.

For a random selection of 400 true catheter image regions (true positives in Tupor's predictions) and 400 non-catheter image regions (false positives in Tupor's predictions), the results are visualized in scatter plots in Figure 5.3. In both scatterplots, the x-axis refers to the catheter confidence score from *Probing FCNN*, while the y-axis refers to the corresponding score from Tupor's classifier; the line for $x = y$ is plotted for ease of analysis. The scatter plot for catheters shows that the vast majority of correct catheter predictions made by Tupor's classifier have a higher catheter confidence score in *Probing FCNN*. By comparison, all of the 400 sampled non-catheters receive a substantially lower score from *Probing FCNN* than from Tupor's classifier. Interestingly, almost none of the 400 catheters are given a lower confidence value than 0.5 by *Probing FCNN*, while nearly all of the 400 non-catheters are.

We repeated the experiment for other random samples of 400 catheters and 400 non-catheters, but the resulting scatterplots always looked very similar, so that only one pair of scatterplots is shown here. Computed over the whole set of 79,979 catheter predictions from Tupor's classifier, around 80% of the true positives had a higher catheter confidence score in *Probing FCNN* and around 92% of the false positives had a lower catheter confidence score in *Probing FCNN*.

The results from this analysis strongly suggest that *Probing FCNN* has successfully learned patterns based on features that are hard for the *U-Net* architecture to internalize. In particular, our frequency spectrum approach can likely lead to substantial improvements compared to the standard deep learning approaches used in medical image analysis.

Chapter Six

CONCLUSION

This thesis proposes a semi-automated system for distinguishing catheters from their echoes in ultrasound images. The probing detector de-convolution algorithm, implemented entirely in the frequency domain, is applied to small regions in the ultrasound images in order to filter out speckle noise in the images. By applying the de-convolution algorithm to a small region in an ultrasound image, a set of frequency coefficients is constructed out of that small region. The constructed frequency coefficients represent the original ultrasound frequencies corresponding to that small region which contains a catheter or an echo. A Neural Network classifier named *Probing FCNN* is designed and implemented in this thesis. It takes as input a set of features computed from the constructed frequency coefficients and classifies the given input by generating two probabilities, where one probability represents the confidence value of a catheter and the other represents the confidence value of an echo. A threshold value is used at this point to make a decision whether the given input represents a catheter or an echo. The prediction accuracy of the classifier is measured by comparing the decisions with the ground truths. Classifiers were trained with various feature maps, where each feature map is the combination of different features. The feature map resulting in the highest prediction accuracy (feature map 4) on the test data is chosen as the basis of our final *Probing FCNN* classifier. Finally, the outcome of the *Probing FCNN* classifier is compared with the outcome of another deep learning classifier trained by S. Tupor. The findings of this strongly suggest

that the overall accuracy of Tupor's classifier could be improved by combining it with the outcome of the *Probing FCNN* classifier.

6.1 Limitations of the Proposed Approach

The feature selection process in the *Probing FCNN* classifier to obtain the highest prediction accuracy is performed through trial and error. Among six different feature maps, the one that yielded the maximum predicting performance was picked. However, the underlying patterns or aspects generated by the feature map after training the predictor are not described in this thesis. Various patterns or aspects can be associated with the trained classifier, for example, patterns in the estimated loss for various random samples, patterns in learned parameters, causes of the decisions, etc. [30]. This demonstrates the lack of interpretability and explainability of the implemented classification approach. The aim of the interpretability and explainability of a classifier is to describe the internal patterns or aspects of the classifier [30]. In many cases, a trained model is considered as a black box due to the reason that the data comes in and the prediction comes out but the learned model cannot be interpreted. However, it may in some cases be necessary to provide insights about the causes of the classifier's behavior in terms of the predictions [30]. Besides, analyzing the internal aspects of a classifier usually widens the scope for improving the model performance.

The regions of interest containing catheters are extracted from annotated ultrasound images by finding the center of each annotated area. However, the echoes are not annotated in the ultrasound images. The regions of interest containing echoes are extracted by identifying the center manually. This manual extraction of echoes raises two issues. First, the manual extraction process takes a long time for a large volume of ultrasound images and second, an extracted region of interest may contain a blank region (neither echo nor catheter). Extraction of a large number of such blank regions of interest and using it during the training can result in misleading predictions. Another limitation of the proposed approach is that the

designed predictor is a binary classifier that predicts whether a given input is a catheter or an echo. The *Probing FCNN* classifier still needs to be combined with other tools to predict all locations of catheters in an ultrasound image.

Furthermore, the analytical tasks performed in this thesis between the output of *Probing FCNN* and Tupor's classifiers are limited. A full combination of these two classifiers to test whether the process improves the overall accuracy of either individual classifier is not implemented in this thesis.

6.2 Future Research

One approach for improving the performance of our model could be to use explanation tools that provide some level of interpretability to the predictions of a Neural Network. Analysis of the learned parameters of the model and their impact on different layers of the network could reveal the patterns constructed in the model. Besides predicting whether a given input represents a catheter or an echo, knowing the reasoning of the predictions helps to detect flaws in the predictor and thus to work on the performance improvement of the trained model. The reasoning of a particular prediction can be uncovered by studying the constructed patterns in the model. The widely used tools for dealing with model interpretability and explainability are SHAP and LIME [31]. The SHAP (Shapley Additive Explanation) approach uses the idea of Shapley values for measuring the influence of a feature. The Shapley values compute all possible predictions using all possible combinations of features which can guarantee prediction consistency of a model [31]. Initially, the Shapley values build and train local models for all possible combinations of features and then use these models to compute the predictions. The LIME (Local Interpretable Model-agnostic Explanations) approach describes the causes of a particular prediction by using the local model corresponding to the prediction [31]. Combining the *Probing FCNN* classifier with these tools could potentially help to ensure both prediction consistency as well as model interpretability.

Furthermore, the output of the probing detector de-convolution algorithm [4] used in this thesis can be used as a basis for creating a local image descriptor [32] for the provided ultrasound images. Detection of points of interest in an image and point of interest descriptors have become an active research topic in recent years [4]. A point of interest descriptor in an image is also known as a local image descriptor. In a local image descriptor, a point of interest is a smaller region in an image on which analytical tasks are performed [32]. The notion of point of interest is equivalent to the notion of region of interest used in this thesis. In general, a point of interest detector is used to identify equivalent regions in single or multiple images and the local image descriptor is used to characterize the identified regions. The aim of the local image descriptor is to construct a smaller vector of numeric data for each point of interest [32]. The constructed vector needs to be invariant to image rotation, photometric variations, scaling, and translation. Also, the reconstructed vector should be low-dimensional but composed of substantial information [32]. Creating an image descriptor that follows these properties can bring multiple images into correspondence. A local image descriptor can be designed and developed for the ultrasound images provided by the Saskatchewan Cancer Agency in the hope of achieving some useful results, for example, finding the similarities or dissimilarities among multiple slices of a single patient or different patients. The de-convolution used in this thesis constructs a set of frequency coefficients for each small region in an ultrasound image known as a region of interest. The constructed frequency coefficients are free of speckle noise and contain substantial frequency information. The frequency coefficients constructed by applying the de-convolution can be used as the major components for creating a local image descriptor.

REFERENCES

[1] Michael A. Nielsen. "Neural Networks and Deep Learning". *Determination Press*, www.deeplearningbook.org (retrieved on Aug 7, 2020), 2015.

[2] Oleg Michailovich and Allen Tannenbaum. "Despeckling of Medical Ultrasound Images". *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 53, no. 1. 2006, pages 64–78.

[3] Baier Rosa and Fernando Monteiro. "Performance Analysis of Speckle Ultrasound Image Filtering". *Computer Methods in Biomechanics and Biomedical Engineering: Imaging and Visualization*, vol. 4, no. 3, 2014, pages 1–9.

[4] Bingyang Liu. "Probing Detector for Image Local Frequency Analysis". Master Thesis. University of Regina, Regina, SK, Canada, 2017.

[5] Thomas Binder, Michael Süssner, Deddo Moertl, Thomas Strohmer, Helmut Baumgartner, Gerald Maurer, and Gerold Porenta. "Artificial Neural Networks and Spatial-temporal Contour Linking for Automated Endocardial Contour Detection on Echocardiograms: A Novel Approach to Determine Left Ventricularcontractile Function". *Ultrasound in Medicine and Biology*, vol. 25, no. 7, 1999, pages 1069–1076.

[6] Julia Noble and Djamal Boukerroui. "Ultrasound Image Segmentation: A Survey". *IEEE Transactions on Medical Imaging*, vol. 25, no. 8, 2006, pages 987–1010.

[7] Jose Sanchez and Michael Oelze. "An Ultrasonic Imaging Speckle-Suppression and Contrast-Enhancement Technique by Means of Frequency Compounding and Coded Excitation". *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 56, no. 7, 2009, pages 1327–1339.

[8] Nelly Pustelnik, Amel Benazza-Benhayia, Yuling Zheng, and Jean-Christophe Pesquet. "Wavelet-based Image Deconvolution and Reconstruction". *Wiley Encyclopedia of Electrical and Electronics Engineering*, vol. 48, no. 3, 2016, pages 371–375.

[9] Wilhelm Burger and Mark Burge. "Digital Image Processing: An Algorithmic Introduction Using Java". *Springer*, New York, 2008.

[10] Thomas Batard and Michel Berthier. "Spinor Fourier Transform for Image Processing". *IEEE Transactions on Signal Processing*, vol. 7, no. 4, 2013, pages 605–613.

[11] Robert Ian Cowles. "An Adjustable Gaussian Filter for Fourier Transformation of Arbitrary Image Region". Master Thesis. University of Regina, Regina, SK, Canada, 2002.

[12] Ben Karsin. "Parallel Fast Fourier Transform Literature Review". Technical Report. University of Hawaii, 2013.

[13] James W. Cooley and John W. Tukey. "An Algorithm for the Machine Calculation of Complex Fourier Series". *Mathematics of Computation*, vol. 19, 1996, pages 297–301.

[14] Sakshi Indolia, Anil Kumar Goswami, Surya Prakesh Mishra, and Pooja Asopa. "Conceptual Understanding of Convolutional Neural Network – A Deep Learning Approach". *Procedia Computer Science*, vol. 132, 2018, pages 679–688.

[15] Deepa Kundur and Dimitrios Hatzinakos. "Blind Image Deconvolution". *IEEE Signal Processing Magazine*, vol. 13, no. 3, 1996, pages 43–64.

[16] Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins. "Digital Image Processing Using Matlab". *Pearson Prentice Hall*, New York, 2004.

[17] Jacob Benesty, Jingdong Chen, Yiteng Huang, Simon Doclo, and Shoji Makino. "Study of the Wiener Filter for Noise Reduction". *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, 2006, pages 9–41.

[18] Andreas Savakis and Richard Carbone. "Discrete Wavelet Transform Core for Image Processing Applications". *International Society for Optics and Photonics*, vol. 5671, 2005, pages 142–151.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. "Deep Learning". *MIT Press*, Cambridge, Massachusetts, 2016.

[20] Qian Shie and Dapang Chen. "Discrete Gabor Transform". *IEEE Transactions on Signal Processing*, vol. 41, no. 7, 1993, pages 2429–2438.

[21] Heather Venables. "How Does Ultrasound Work?" *Institute Laboratory Animal Research*, vol. 19, 2011, pages 44–49.

[22] Jiawei Zhang. "Gradient Descent Based Optimization Algorithms for Deep Learning Models Training". *ArXiv*, vol. 1903.03614, 2019.

[23] Siyue Wang, Xiao Wang, Pu Zhao, Wujie Wen, David Kaeli, Peter Chin, and Xue Lin. "Defensive Dropout for Hardening Deep Neural Networks Under Adversarial Attacks". *ArXiv*, vol. 1809.05165, 2018.

[24] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning". *ArXiv*, vol. 1811.03378, 2018.

[25] Joaquin Sola and Joaquin Sevilla. "Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems". *IEEE Transactions on Nuclear Science*, vol. 44, no. 3, 1997, pages 1464–1468.

[26] Leonard Berrada, Andrew Zisserman, and Pawan Kumar. "Smooth Loss Functions for Deep Top-k Classification". *ArXiv*, vol. 1802.07595, 2018.

[27] Katarzyna Janocha and Wojciech Marian Czarnecki. "On Loss Functions for Deep Neural Networks in Classification". *ArXiv*, vol. 1702.05659, 2017.

[28] Ke-Lin Du and Shanmukha Swamy. "Neural Networks and Statistical Learning". *Springer*, London, 2014.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". *In Proceedings of the* 18^{th} *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pages 234–241.

[30] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. "An Overview of Interpretability of Machine Learning". *Data Science and Advanced Analytics*, vol. 1806.00069, 2018, pages 80–89.

[31] Scott Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". *ArXiv*, vol. 1705.07874, 2017.

[32] Simon Winder and Matthew Brown. "Learning Local Image Descriptors". *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pages 1–8.