Clemson University

TigerPrints

All Theses

Theses

12-2023

Detection of Myofascial Trigger Points With Ultrasound Imaging and Machine Learning

Benjamin Formby
bformby@clemson.edu

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses

Part of the Other Computer Engineering Commons

Recommended Citation
Formby, Benjamin, "Detection of Myofascial Trigger Points With Ultrasound Imaging and Machine Learning" (2023). *All Theses*. 4182.
https://tigerprints.clemson.edu/all_theses/4182

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

Detection of Myofascial Trigger Points with Ultrasound Imaging and Machine Learning

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

---

by
Benjamin Greer Knox Formby
December 2023

Accepted by:
Dr. Kuang-Ching Wang, Committee Chair
Dr. Adam Hoover
Dr. Yongkai Wu

Abstract

Myofascial Pain Syndrome (MPS) is a common chronic muscle pain disorder that affects a large portion of the global population, seen in 85-93% of patients in specialty pain clinics [10]. MPS is characterized by hard, palpable nodules caused by a stiffened taut band of muscle fibers. These nodules are referred to as Myofascial Trigger Points (MTrPs) and can be classified by two states: active MTrPs (A-MTrPs) and latent MtrPs (L-MTrPs). Treatment for MPS involves massage therapy, acupuncture, and injections or painkillers. Given the subjectivity of patient pain quantification, MPS can often lead to mistreatment or drug misuse. A deterministic way to quantify the pain is needed for better diagnosis and treatment.

Various medical imaging technologies have been used to try to find quantifiable and measurable biomarkers of MTrPs. Ultrasound imaging, with it's accessibility and variety of modalities, has shown significant findings in identifying MTrPs. Elastography ultrasound, which is used for measuring stiffness in soft tissues, has shown that MTrPs tend to be stiffer than normal muscle tissue. Doppler ultrasound has shown that bloodflow velocities differ significantly in areas surrounding MTrPs. MTrPs have been identified in standard B-mode grayscale ultrasound, but have varying conclusions with some studies identifying them as dark hypoechoic blobs and other studies showing them as bright hyperechoic blobs. Despite these discoveries, there is a high variance among results with no correlations to severity or pain.

As a step towards quantifying the pain associated with MTrPs, this work aims to introduce a machine learning approach using image processing with texture recognition to detect MTrPs in B-mode ultrasound. A texture recognition algorithm called Gray Level Co-Occurrence Matrix (GLCM) is used to extract texture features from the B-mode ultrasound image. Feature maps are generated to emphasize these texture features in an image format in anticipation that a deep convolutional neural network will be able to correlate the features with the presence of a MTrP. The GLCM feature maps are compared to the elastography ultrasound to determine any correlations with muscle stiffness and then evaluated in the presence of MTrPs. The feature map generation is accelerated with a GPU-based implementation for the goal of real-time processing and inference of the machine learning model. Finally, two deep learning models are implemented to detect MTrPs comparing the effect of using GLCM feature maps of B-mode ultrasound to emphasize texture features for machine learning model inputs.

Dedication

I would like to dedicate this work to my loving parents. They have always supported me and loved me unconditionally, teaching me to do my best, encourage others, and help others to do their best, which I strive to do everyday.

Acknowledgments

First and foremost, I would like to express my gratitude to my advisor, Dr. KC Wang. He has been vital to my development as a graduate student and researcher, providing insight and encouragement to help me learn and pursue my interests. I'm very appreciative of the time and resources he has invested in me to enable me to make it this far.

I would also like to express my gratitude towards my committee, Dr. Adam Hoover and Dr. Yongkai Wu. During both my undergraduate and graduate careers, it is through their classes and teaching that I was inspired and motivated towards the computer vision and machine learning fields. I would like to thank them for their time and wisdom that they have shared with me.

Next, I would like to thank other Clemson University faculty and collaborators on this work. It would not have been possible without their contributions. Dr. Hai Xiao and his lab have played an important role in collaboration with this project. Dr. Hudson Smith has provided much wisdom and insight throughout the research process in this project. Dr. Scott Annett from Prisma Health and Dr. Hui-Lin Tsai from Clemson University have been critical in providing knowledge on the clinical background and concepts of this project, as well as their time spent assisting in data collection and medical expertise. This work would also not be possible without the teaching and guidance from Dr. Melissa Smith.

The work from this project was in collaboration with and in part funded by NSF FABRIC (#1935966), NSF CloudLab (#2027208), NIH COBRE Team Science Award (#3P20GM121342-06S1), and the CU-Prisma Innovation Maturation Fund.

Additionally, I would like to thank my peers and fellow students who have encouraged me and provided technical support throughout the project, namely Acheme Acheme, Sarah Maxwell, and Victoria Hemphill.

Last, but certainly not least, I would like to give my thanks and appreciation to all of my friends and loved ones that have supported me and encouraged me throughout. All of my family, my fiancée Grace, and close friends have shown unconditional love that has motivated me to persevere. I'm thankful to God for how He has blessed me with this opportunity, for I certainly would not be able to accomplish it without Him.

Table of Contents

List of Tables

List of Figures

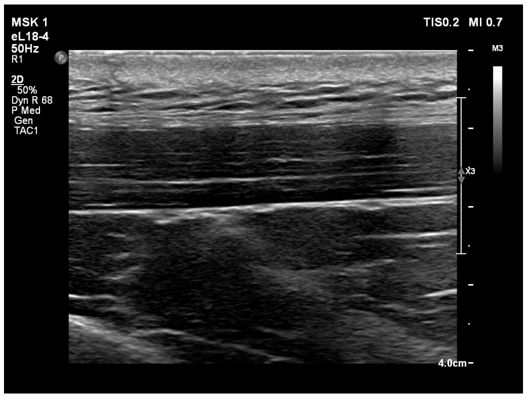
Chapter 1

Introduction

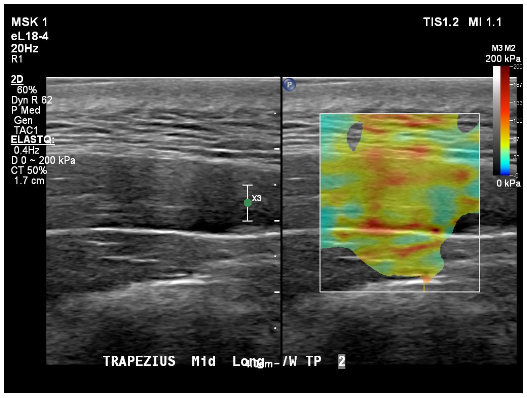
Muscle pain affects nearly all of the global population. It has a variety of causes including muscle overuse, stress, injury, or even more serious disorders or illnesses. Chronic muscle pain, commonly diagnosed as Myofascial Pain Syndrome (MPS), is characterized by stressed and stiffened bundles of muscle fibers that are hard and palpable, known as Myofascial Trigger Points (MTrPs). Quantifying the pain is generally a subjective task since there are no defining, quantifiable biomarkers to measure with medical imaging. This work proposes a system combining image texture analysis of ultrasound imaging, GPU acceleration, and machine learning to make a step towards pain quantification by detection of Myofascial Trigger Points (MTrPs).

1.1 Ultrasound Imaging Modalities

Ultrasound imaging is widely used for various medical imaging applications such as tumor and cyst detection [20], cardiography, prenatal testing, emergency treatment, and sports medicine [12],[8]. Ultrasound technology is relatively inexpensive and easily accessible compared to other medical imaging technologies (X-Ray, MRI, CT, etc.), making it well suited for these applications. Additionally, ultrasound technology continues to be improved with continuous developments in signal processing and noise reduction [18]. There are several types of ultrasound imaging including B-mode, elastography, and echocardiography. B-mode ultrasound is a two-dimensional representation of the ultrasound echoes acquired from the scanning, which is the most commonly used mode and ideal for general purpose imaging. While B-mode ultrasound is used in many applications, it is most commonly used for prenatal testing, emergency care, and sports medicine. For emergency care, B-mode ultrasound is used for bedside examinations that require fast diagnosis and treatment like an injury that may have caused internal bleeding, for example. For sports medicine applications, B-mode is often used for muscle and joint examination for injuries involving tendon or muscle tears. The B-mode ultrasound is the most widely used type of ultrasound because of these applications that need fast treatment and quick imaging. Echocardiography uses Doppler ultrasound technology to show bloodflow in the heart and cardiovascular system. Echocardiography is used in treatment of heart conditions and even in prenatal care for monitoring a fetus' heart health [27],[31]. Lastly, elastography ultrasound measures stiffness of soft tissues or organs in the body [5],[14],[48]. Elastography is most commonly used for testing of the liver and kidneys for any scar tissue that may be result of some disease or disorder. Elastography is also sometimes used for cancer detection in the liver, as well as breast cancer and prostate cancer because cancerous tissues are known to be stiffer in many of these cases [36],[11],[29].



(a) B-Mode ultrasound imaging of trapezius muscle.

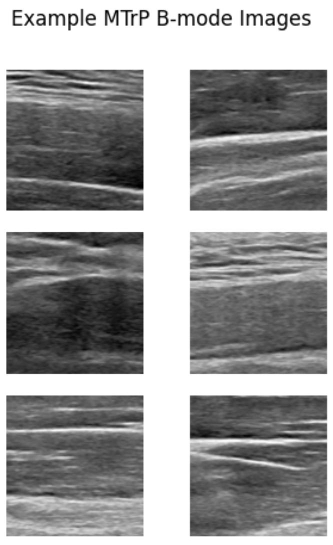


(b) Shear-wave elastography ultrasound imaging of trapezius muscle.

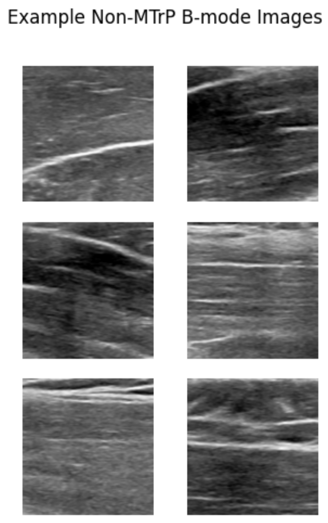
Figure 1.1: Example images of (a) B-mode ultrasound imaging and (b) shear-wave elastography ultrasound.

1.2 Myofascial Pain Syndrome

Within the sports medicine domain, muscle pain is a common problem among the majority of the global population. The main challenge of treating muscle pain is the ability to detect and quantify the pain through medical imaging like ultrasound. Recurrent muscle pain is typically diagnosed as Myofascial Pain Syndrome (MPS). MPS is a chronic pain disorder that can be characterized by painful nodules located within muscle tissues that are believed to be caused by a dysfunction of the myofascial tissues of a muscle. These nodules are referred to as Myofascial Trigger Points (MTrPs) and are often hard palpable spots in the muscle that are painful upon compression [44],[43]. MTrPs can be either active or latent, where active is the painful state and latent is the non-painful state of the trigger point. In clinical use, there are no known biomarkers of MPS or trigger points, making it difficult to quantify the pain associated with them. Figure 1.2 displays a few example B-mode ultrasound images with and without MTrPs to demonstrate the challenge of identification and quantification. These example images were obtained in the data collection process which is described in a later section. The cause of MPS is not well understood and the exact pathophysiology of what is occurring in the muscle tissue itself is unknown [43]. The pain associated with these trigger points is subjectively determined by the patient's pain description. Treatment for MPS can involve acupuncture or massage therapy, but mostly involves injections or painkillers. The mistreatment of muscle pain and MPS leads to drug misuse and even addiction among patients suffering from this condition. Quantifying muscle pain would greatly improve MPS patients' diagnoses, and therefore improve treatment and reduce drug misuse.



(a) With MTrPs.



(b) Without MTrPs.

Figure 1.2: Example images of B-mode ultrasound imaging with (a) MTrPs and (b) without MTrPs.

1.3 Machine Learning in Medical Imaging Applications

Artificial intelligence and machine learning systems have been exploding in popularity among academia, industry, and even healthcare. Artificial intelligence is used in many healthcare applications including tumor detection, segmentation of medical imaging, and noise reduction of medical imaging. Most of these applications could be summarized as computer aided diagnosis (CAD) which describes an intelligent system used for aiding in the interpretation of medical imaging or data. MTrP detection and quantification can benefit from artificial intelligence, as well. The task may be possible with the proper combination and utilization of image processing and machine learning techniques.

1.3.1 Image Processing of Ultrasound and Other Medical Imaging

It is well-known that ultrasound is a noisy medium for medical imaging. Despite constant developments in ultrasound signal processing and noise reduction, ultrasound remains to be a delicate form of imaging that depends on many variables such as angle of inclination, rotation angle, and pressure. Many of these variables are dependent on operator handling rather than technology. For this reason, feature extraction is a difficult task in ultrasound imaging and image processing is a crucial step in the machine learning pipeline. Depending on the task, further noise reduction or filters may be applied to focus on certain aspects of an image such as edges or brightness. Edge detection algorithms are commonly used for segmentation and contouring tasks [21]. Feature enhancement filters or algorithms like histogram equalization [46] can be used to enhance contrast in images. More complex algorithms like shape detection or texture analysis algorithms may be used to find objects or regions of interest in images.

For the purpose of muscle pain measurement, texture recognition algorithms are presumed to be the most beneficial step towards identifying areas of interest in the muscle. Texture analysis algorithms like Gray Level Co-Occurence Matricies (GLCM) and Local Binary Patterns are popular methods of identifying patterns in image pixel data, which make up the image textures [17],[20]. Features are extracted from these algorithms which assist in describing and interpreting images. These features can be used to provide feedback to medical specialists by providing feature outputs pertaining to areas of interest, visualizations of areas of interests, or be used as inputs to a machine learning model for predictions. Furthermore, feature maps can be generated for image-like inputs of machine learning models, emphasizing desired features in an image. While deep learning models are already good at detecting textures and features in images, it is anticipated that the emphasis provided by these feature maps would improve a model's performance that is trained on very little data.

1.3.2 GPU Acceleration of Medical Image Processing

Providing feedback to medical specialists performing an ultrasound scan in real-time is important for faster, easier, and more accurate diagnosis and treatment [25]. Real-time feedback would improve efficiency of scans by providing visualizations or predictions while a scan is in process versus after. Additionally, feedback during a scan provides better interpretation for doctors and technicians since they have context of the scan in terms of probe location, orientation, and movement. Better image interpretation would lead to more accurate diagnosis.

Real-time feedback is difficult to achieve when using algorithms like GLCM to create feature maps. The GLCM algorithm is a computationally expensive algorithm [35],[38] and constructing a GLCM corresponding to every pixel in an image for a full resolution feature map makes real-time processing serially impossible. These heavy algorithms can be accelerated using graphical processing units (GPUs) to parallelize the computations that are being made. The use of GPUs for algorithm acceleration can make real-time processing and inference possible for medical specialists.

1.3.3 Machine Learning Methods for Medical Imaging Applications

In medical imaging machine learning applications, data collection can be a challenging process and machine learning requires large amounts of data. Transfer learning is a method commonly used in medical applications that addresses the issue of small datasets by pre-training a model on a different dataset. In many cases, it has been shown to improve results by initializing a model with weights set by another dataset and then train the model on the applied dataset. While it would always be better to train a model on the same type of data, transfer learning is often a better option when data is limited and hard to collect, like it is with many medical imaging applications. Data augmentation is another machine learning strategy that uses transformations on training images to increase the training set sample size. For medical imaging applications like this one, deep convolutional neural networks (CNNs) have proven to be very effective in extracting image features in classification and detection tasks.

1.4 Contributions

This work proposes an end-to-end process of interpreting ultrasound image data to detect MTrPs. Under the assumption that elastography ultrasound is an accurate representation of muscle stress and stiffness, textural features are extracted through a GPU accelerated texture recognition algorithm and compared for validation. These comparisons also act as a basis for deciding what parameters are to be used when considering feature maps as machine learning inputs. The main contributions of this work include the following:

- An assessment of GLCM feature maps of B-mode ultrasound through (1) a comparison to elastography ultrasound for measuring muscle stiffness and (2) an evaluation of their correlation to the presence of MTrPs.
- A study on GPU-based implementations of accelerating the GLCM feature map generation to achieve real-time processing.
- A comparison of two deep learning approaches to detect MTrPs with and without using GLCM feature maps of B-mode ultrasound.

Chapter 2

Related Work

Recent research has shown that ultrasound is a viable option for measuring muscle stiffness and identifying MTrPs. Image processing techniques like noise reduction, segmentation, feature enhancement, and texture recognition are common in ultrasound applications as well as other medical imaging applications. Consequently, processing these images in real-time with these algorithms can be computationally expensive. Given the nature of image data, parallelization is a common method used for accelerating these algorithms. Lastly, many different machine learning approaches have been taken to tackle a variety of classification or segmentation tasks in medical imaging.

2.1 MPS Diagnosis and Ultrasound Imaging

The current process for diagnosing MPS and muscle pain requires a physical examination where a doctor must palpate the muscle tenseness or MTrPs [10]. Examiners facilitate certain stretches and movements to test mobility and reproducability of pain caused by MPS. Examiners then feel for any palpable and painful nodules, identifying any active or latent MTrPs typically in the upper back or neck. Different types of ultrasound modalities may be used to try to visualize or capture the MTrPs in imaging.

Some studies have demonstrated that the MTrPs are visible in B-mode ultrasound and elastography ultrasound [44]. This study had 9 patients that had been diagnosed with MPS and all but 2 were experiencing pain at the time of the study. Examiners performed physical examinations and identified up to 2 trigger points on each side of the upper back for each patient. Examiners also identified areas of normal muscle for comparison. MTrPs were identified as active (A-MTrP) or latent (L-MTrP) depending on if there was pain. Vibration sonoelastography ultrasound was performed on the labeled trigger point areas, as well as on the normal labeled areas. Vibration sonoelastography was the method of measuring stiffness in 2010 when the study was performed. Shear-wave elastography (SWE) is the newer mode of ultrasound used for measuring stiffness, which is used in this work. The study found that trigger points appear as hypoechoic, or darker, regions in B-mode ultrasound and show higher levels of stiffness in the elastography ultrasound. There was no significant difference in size or shape of active versus latent trigger points. [26] supports this finding, showing that MTrPs appear as as hypoechoic regions in B-mode ultrasound, however, found that active MTrPs were larger in size using elastography. This study also tried some basic image processing and filtering including an entropy filter to try to highlight the features of the MTrP, but was unable to show significant results. Contrary to [44] and [26], [41] performed a similar study using B-mode ultrasound and found a patient to have MTrPs appear as hyperechoic, or brighter white, regions. [43] uses Doppler ultrasound to examine bloodflow in and around active trigger points. This study had 16 patients with acute neck pain participate in the study. Again, examiners marked active, painful trigger points and also identified areas of normal muscle. Then, spectral doppler ultrasound was used to measure the systolic and diastolic velocities of the blood in these areas. Through a statistical analysis of the peak systolic velocity and minimum diastolic velocity, the study found significant differences between trigger points, both active and latent, and normal muscle tissues. The findings showed increased systolic velocities and reversed flow with negative diastolic velocities.

While these works have shown statistically significant results towards identifying trigger points with varying ultrasound modalities, there is still great difficulty in measuring the severity or pain of these trigger points. As shown in [44], there is high variance in size and shape of the visible regions in the ultrasound imaging. Consistent and reliable biomarkers that are quantifiable are yet to be discovered.

2.2 Image Processing and Texture Recognition Methods in Medical Imaging

There are many tools used in image processing that help accomplish a variety of tasks in computer vision. Some of these methods include noise reduction, feature enhancement, segmentation, and texture analysis.

2.2.1 Noise Reduction

Recent advancements have been made towards noise reduction in ultrasound imaging. Ultrasound imaging often suffers from a speckled noise, creating artifacts in the image that can be hard to distinguish between true features or noise. This noise could be caused by many things like stray echoes of the ultrasound or a lack of gel on the ultrasound probe [22]. The works [2] and [23] perform extensive experiments, testing up to 67 different filters to find the best at removing the noise while preserving features. Other methods have used deep learning approaches [39],[16],[46] to find the optimal way of reducing the type of noise in the image. [16] employs a convolutional autoencoder to remove speckling in mammogram and dental medical imaging datasets. [39] performs a similar approach for removing noise in echocardiography ultrasound.

To apply some of these noise reduction methods to aid in MTrP detection would be difficult considering the characteristics of the trigger points in an image are still relatively unknown. Noise filtering or noise reduction may discard important information about the muscle tissue and the effects that a MTrP may have on the surrounding area.

2.2.2 Feature Enhancement

Many works have made efforts to enhance the features that may be subtly present in medical imaging. A common approach used for this is contrast enhancement or histogram equalization. [20] performs semantic classification of tumors in breast ultrasound with histogram equalization on the region of interest as a preprocessing step. Histogram equalization has many variations including Recursive Mean Separate Histogram Equalization (RMSHE) [1] and Contrast Limited Adaptive Histogram Equalization (CLAHE) [6],[4],[46]. All of these works use these histogram equalization feature enhancement algorithms as an instrumental step in their preprocessing, showing improvements in noise reduction and classification accuracies.

Feature enhancement algorithms such as these are generally an effective approach, however, similar to noise reduction methods, feature enhancement may be difficult since the features and biomarkers of MPS and general muscle stiffness is unknown in ultrasound. Using a contrast enhancement algorithm would require a careful consideration of what the lower contrast areas may mean for muscle stiffness.

2.2.3 Segmentation

Segmentation methods have been used as both preprocessing steps and as final tasks in medical imaging applications. [20] has segmentation as a final output that uses noise reduction, feature enhancement, and texture recognition to make a final segmentation of breast tumors in ultrasound. [6] on the other hand, simply uses segmentation and masking to extract regions of interest that are then used in further processing or analysis. [21] is another work with segmentation of brain tumors in MRI scans as the goal. This work employs a watershed segmentation algorithm along with a combination of noise reduction, feature enhancement, and texture recognition.

Image segmentation could be beneficial to MTrP detection by identifying regions of the image that show interesting characteristics like muscle stiffness, or regions that can be ignored such as fat tissues. However, image segmentation still needs a basis for segmenting, which becomes difficult to define because of the unknowns surrounding MTrP characteristics in ultrasound image data.

2.2.4 Texture Recognition

Finally, of the related image processing techniques that are commonly used in medical imaging, texture recognition algorithms like the Gray Level Co-Occurrence Matrix (GLCM) algorithm and Local Binary Patterns (LBP) algorithm are very useful for finding patterns within pixel data. The GLCM algorithm first founded by Haralick [17], is a matrix representation of pixel pair occurences in an image or patch of an image. Haralick features like homogeneity, dissimilarity, energy, contrast, and correlation can be statistically calculated from the resulting matrix of the GLCM algorithm. Similar to GLCM, LBP is another algorithm that creates a local representation of binary patterns around a given pixel [33]. These algorithms, especially GLCM, are very popular in feature extraction of medical imaging. Often times, GLCM is used for finding texture features of objects or regions of interest. [21] aims to classify MRI scans for brain tumor detection based on GLCM features of a segmented region. [20] uses both LBP and GLCM texture features of superpixels that make up the region of interest in order to segment breast tumors in ultrasound. [4] performs GLCM texture analysis on lung ultrasound to do a statistical comparison of features and differentiate scans with Covid-19 and pneumonia. Works like [15] and [28] have made efforts to improve the algorithm further by designing variations that are gray level invariant and higher order applications of GLCM.

Texture recognition has shown to be a very effective method for feature extraction of a objects and regions of interest. In the case of detecting MTrPs, the entire muscle is the initial region of interest, yet can have a variety of characteristics within. A texture recognition approach that would provide a global and local representation of texture patterns and would give insight to what features may describe a trigger point or muscle stiffness.

2.3 GPU Acceleration of Medical Image Processing

Because of the constant increase in computing power as Moore's Law suggests, and the popularity increase of general purpose GPUs, GPUs are becoming more widely used for medical image processing and algorithm acceleration. [25] gives a composition of works in the medical imaging community that utilize GPUs for algorithm acceleration. Some of these works include 3D reconstruction of brain MRIs and CT, tracking applications of brain fibers, segmentation of vertebrae X-Ray, and motion estimation. Speedups range in these applications from 3x faster to 60x faster and enable tools for better diagnosis and treatment of patients that weren't possible without such computing power.

More specifically to the GLCM algorithm, a few works have demonstrated acceleration approaches on both GPUs and FPGAs. [3] implements the algorithm using FPGA hardware and parallelizes the construction of just the matrices, achieving 33% decrease in runtime. Works like [35] and [38] implement the algorithm on GPU architectures and are able to achieve up to 155x and 20x, respectively. These papers both used the CUDA architecture and libraries for programming NVIDIA GPUs, parallelizing both the GLCM construction and feature calculation. It is important to note that the speedups differ because the latter used only 8 grayscale levels, resulting in smaller GLCMs and the work with higher speedup had all 256 grayscale levels. Speedups vary on many parameters, which will be discussed later in this work.

Furthermore, similar to most applications of GLCM, these works only construct a single GLCM for an image or region of interest, or they divide an image into many chunks, constructing GLCMs and calculating features for each chunk. No work was found that has explored a feature map generation of GLCM features and accelerated the feature map generation for real-time processing with the intent for image-like inputs to a convolutional neural network. Some works like [20] that utilize superpixels and determine GLCM features of these superpixels is a similar approach, but does not aim for real-time processing. [35] calculates multiple GLCMs throughout a divided image, almost indirectly creating a low resolution feature map. [38] performs GLCM calculation for every pixel in an image for synthetic aperture radar imaging, but does not explain how exactly the features are then used in classification. None of these mentioned works describe a full resolution GLCM feature map usecase in machine learning or a scalable approach using multiple GPUs for further acceleration.

2.4 Machine Learning in Ultrasound and Other Medical Imaging

Machine learning is becoming a powerful tool for medical imaging applications. Machine learning is being used as computer aided detection and diagnosis, as well as for more specific tasks like segmentation, noise reduction, tracking, and classification. Some of the types of machine learning methods used in medical imaging applications are reviewed in [37]. [37] describes both supervised and unsupervised methods including regression, classification, clustering, and reinforcement learning. More specifically, with image data, deep neural networks (DNNs) or deep convolutional neural networks (CNNs) are very popular and efficient for learning features embedded within image data. [21] uses a traditional machine learning method, a support vector machine, combined with some pre-processing methods and an extensive texture analysis to detect brain tumors. More and more medical applications are beginning to utilize these deep learning computer vision methods as described in [24], and [42]. [24] provides a review of medical imaging applications to deep learning including brain tumor segmentation, skin cancer detection, lung imaging, and more. [42] describes novel self-supervised deep learning methods like generative adversarial networks and contrastive learning methods for applications like brain tumor segmentation and other organ segmentation tasks.

A common challenge among medical imaging machine learning applications is the data collection process and dataset sizes. This is due to many factors with the largest being privacy concerns, expensive or inaccessible equipment, and labeling challenges [34],[9]. There are a few strategies that have yielded performance improvements under such circumstances and some of those include transfer learning and data augmentation.

2.4.1 Transfer Learning

Transfer learning is a method of using models that are pre-trained on a different dataset and then alter the input specifications and classification to apply to the desired dataset. Many medical imaging studies have employed this technique to overcome challenges of small datasets and demonstrated better performance when using transfer learning. [6] uses transfer learning in a tumor detection application in mammogram imaging. The dataset used in this work consists of only 300 samples and using an AlexNet architecture pretrained on ImageNet was able to achieve 99% accuracy, an increase of 15% better accuracy than a model without transfer learning. [31] tests multiple deep learning architectures pre-trained on ImageNet for an echocardiography ultrasound classification task.

The ImageNet dataset [13] has proven to be almost unmatched when it comes to pre-training and transfer learning because of the diversity and size of the dataset. ImageNet is a growing dataset that is currently 14+ million images of a wide variety of categories and subcategories. Despite the fact that ImageNet is not a medical imaging dataset, or even a grayscale dataset, it's been demonstrated that the image features learned from such a large variety of images still translate well to other applications like medical imaging. A newer dataset consisting of a variety of medical images called RadImageNet [30] has shown promising results for medical imaging applications using transfer learning. RadImageNet is a compilation of over 5 million radiological medical images including MRI, Ultrasound, CT, and PET scans. [32] compared these two datasets as a basis of transfer learning and found that both were pretty similar, but each had better performance depending on the architecture used.

Some of the most popular and powerful deep learning architectures used for transfer learning include ResNet [19], Inception [47] and VGG [45]. For image recognition, the ResNet family has consistent and exceptionally high performance on datasets like ImageNet and CIFAR. The ResNet family consists of varying depths ranging from 18 layers to 1,202 layers. The more popular variants include ResNet-34 and ResNet-50 given the balance between efficiency and accuracy.

2.4.2 Data Augmentation

Additionally, when short on data samples, data augmentation is utilized to increase the training dataset size [6]. Data augmentation involves creating new training samples by performing transformations of original samples. Transformations can be anything from a simple rotation or flip to scaling or affine transformations. These different augmentations can be classified as geometric based, color and intensity based, and non-rigid based augmentations [7]. This work tested varying augmentations on three different ultrasound datasets, seeing an increase in performance accuracy in every category of augmentation and with every dataset.

Chapter 3

Texture Analysis and GLCM Feature Maps for MTrPs

This chapter describes the texture recognition method used as the first step towards detecting MTrPs in B-mode ultrasound. It is anticipated that Gray Level Co-Occurrence Matrix (GLCM) features may benefit a machine learning model by emphasizing certain texture features, improving the efficiency and accuracy of a model that is trained on a small medical imaging dataset. Experimentation results are shown for a comparison between GLCM texture features found in B-mode ultrasound and the muscle stiffness measurements provided by shear-wave elastography (SWE) ultrasound. Then, the GLCM feature maps and elastography measurements are assessed with the presence of MTrPs.

3.1 Problem Statement

As a step towards MTrP detection, correlating GLCM feature maps of B-mode ultrasound with SWE ultrasound would validate that the use of GLCM can reproduce at least the muscle stiffness associated with MTrPs. With recent works using SWE ultrasound to measure stress in muscle tissue, this work assumes SWE ultrasound can accurately measure the stiffness of muscles. Subsequently, finding an equivalent method using only B-mode ultrasound as a source of data would be useful since elastography ultrasound is not as accessible as standard B-mode. This chapter provides (1) an assessment of the correlation between SWE ultrasound and GLCM feature maps of B-mode ultrasound, and (2) an assessment of both the SWE ultrasound and the GLCM feature maps and their correlation to MTrPs. This chapter aims to find these correlations between GLCM feature maps, SWE, and MTrPs to provide a basis for the inputs of a machine learning model.

3.2 Basis for Comparison

Texture recognition, specifically the GLCM algorithm, provides an effective way of statistically identifying patterns that occur within the image data that may not be visible to the naked eye. Based on the literature studying MPS, MTrPs may cause changes in the texture and composition of the muscle tissue. This makes GLCM a suitable candidate for texture recognition and investigating the patterns that may be present in the image data. To further visualize these features, entire feature maps can be generated corresponding pixel-by-pixel to the input ultrasound image. These feature maps provide another level of feature visualization and interpretation for both human perception and computer vision, since the pixel-by-pixel feature representations maintain a spatial relationship rather than singular feature values representing a region of interest. Therefore, for the MTrP detection, it is anticipated that the generation of these feature maps will emphasize the texture features that may correspond to the presence of a MTrP, and maintaining the nature of image data, be effective for a deep convolutional neural network.

In an effort to validate the patterns found by the GLCM feature maps, they are compared to SWE ultrasound measurements using image similarity metrics. The findings from this comparison provide an educated selection of GLCM parameters since the muscle stiffness provided by SWE elastography has been shown in literature to help characterize MTrPs. The findings will then provide a basis for the inputs of the machine learning model.

3.3 Methods

The methodologies of this chapter start with an explanation of the GLCM algorithm and how it is used to generate feature maps of the B-mode ultrasound imaging. Next, the process of correlating the generated maps using various parameters to the shear-wave elastography (SWE) is described and what metrics are used. Finally, statistical comparisons are made to determine if any patterns result from when MTrPs are present in the input images.

3.3.1 GLCM

The Gray Level Co-Occurrence Matrix (GLCM) algorithm is a matrix representation of the frequency of pixel pairs in a given image or patch of an image. A square matrix of size equal to the number of grayscale levels in the image is created, where each i^{th},j^{th} element of the matrix is the number of occurrences that a pixel value i occurs at an angle θ and distance d from a pixel value j. This is demonstrated in the figure and algorithm below (Fig. 3.1 and Alg. 1) where a 4x4 image with 3 grayscale levels is used to generate a 3x3 GLCM with at an angle θ of 0 and distance d of 1. The GLCM is then converted to a normalized GLCM where each element represents the probability of that pixel pair occurring.

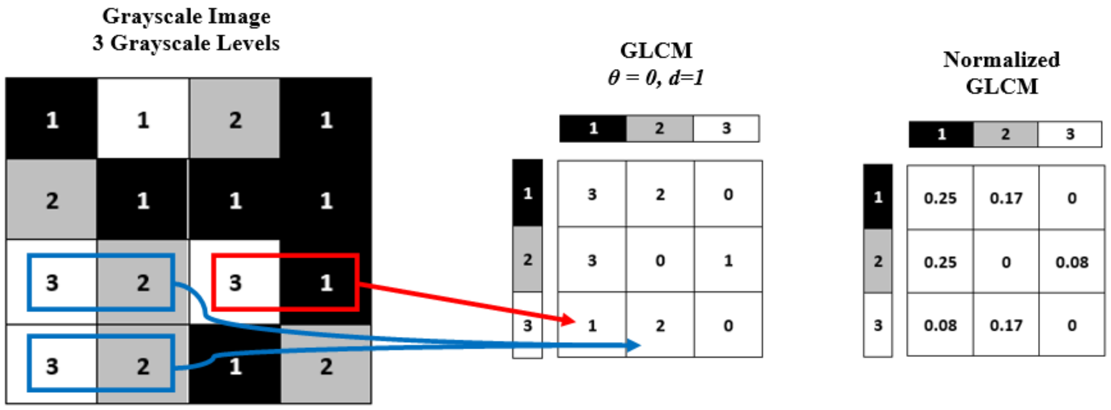


Figure 3.1: Calculation of GLCM.

Algorithm 1 GLCM calculation.

```
for $r$ in $Patch$ do
    for $c$ in $Patch$ do
        $m$ = image[r][c]
        $n$ = image[r][c] at $\theta$ with $range$
        increment $GLCM[m][n]$
    end for
end for
```

$$P(i,j) = \frac{p(i,j)}{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p(i,j)} \tag{3.1}$$

After a GLCM is constructed, statistical properties such as homogeneity, dissimilarity, energy, correlation, and contrast can be calculated as a single numerical feature value for that GLCM. The statistical features must be calculated from a normalized GLCM, matrix P. Matrix P becomes a matrix of probabilities that element $P(i,j)$ occurs in the image or patch. This probability is calculated from Equation 3.1 where $p(i,j)$ is the i,j element of the un-normalized GLCM and P is the normalized GLCM. While there are many features that can be calculated, the following five features were chosen to be used in experimentation.

$$Homogeneity = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\frac{P(i,j)}{1+(i-j)} \tag{3.2}$$

$$Dissimilarity = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|i-j|P(i,j) \tag{3.3}$$

$$Energy = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}P(i,j)^2 \tag{3.4}$$

$$Contrast = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(i-j)^2P(i,j) \tag{3.5}$$

$$Correlation = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\frac{(P(i,j)(i\times j)-(\mu_x\times\mu_y)}{(\sigma_x\times\sigma_y)} \tag{3.6}$$

These features summarize most other possible features and are sufficient for this muscle texture recognition since the features that best characterize the MTrPs are still to be determined. Applying this to a feature map approach requires a GLCM to be constructed for every pixel in the image, or at whatever desired resolution. This is described in Algorithm 2. The feature map is also normalized to a scale from 0 to 255.

3.3.2 Comparison of SWE US and GLCM Feature Maps of B-Mode US

To determine if any of the GLCM feature maps are similar to or highlight similar features of elastography, two image similarity metrics are used: Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM) [40]. These metrics are defined by Equations 3.7 and 3.8 The shear-wave elastography used in this work is a mapping to a one-dimensional RGB colormap. Since

Algorithm 2 GLCM feature map generation.

```
for $Row$ in $Image$ do
    for $Col$ in $Image$ do
        for $r$ in $Patch$ do
            for $c$ in $Patch$ do
                $m$ = image[r][c]
                $n$ = image[r][c] at $\theta$ with $range$
                increment $GLCM[m][n]$
            end for
        end for
    end for
end for
```

not all of the region of interest of the elastography imaging fills in with a level of stiffness 1.1b, a mask is used to black out all pixels that are not given a stiffness value. Because of this, a fully black image is used in comparison as a baseline to show a minimum score. The original B-mode image is also used as a baseline for comparison. The first metric uses Mean Squared Error (MSE) to determine the average distances between pixel values of each image. Using this metric, the fully black baseline image would give the maximum MSE score possible. For the second metric, Structural Similarity Index Measure (SSIM) is used. This metric gives a score ranging from -1 to 1 where -1 is completely anti-correlated and 1 is a perfect match, 0 is dissimilar. MSE estimates an absolute error based on pixel distances between images whereas SSIM takes into account a spatial relationship to neighboring pixels consisting of three components to it's estimation: luminance, contrast, and structure. These metrics are calculated for a variety of feature maps that have varying parameters, in an effort to determine if any specific parameter values are more similar to the SWE measures than others. When making the comparison, since the feature maps are a one channel grayscale image, the feature maps are converted to a 3-channel colormap defined by the OpenCV library, which is as close to the elastography colormap as possible at this time. The colormaps are not exact, but are assumed to be the same for the purpose of this work.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{3.7}$$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.8}$$

3.3.3 MTrPs in SWE and GLCM Feature Maps of B-Mode US

For a basic statistical analysis about the presence of MTrPs, averages and standard deviations are taken of labeled elastography images and compared to that of GLCM feature maps of B-mode ultrasound. An average pixel value and its standard deviation is taken of every labeled elastography image for both elastography and the respective feature maps. These values calculated across many images are averaged, resulting in average means and average standard deviations of images labeled with and without MTrPs.

3.4 Experimentation

This section describes the procedures for the data collection process from scan to processable image data. Next, the feature map generation specifications are described and how they are evaluated.

3.4.1 Preliminaries and Data Collection

For the data collection, a Phillips Epiq Elite 7 ultrasound machine is used to perform scans of the trapezius and infraspinatus upper back muscles. These scans are performed by a medical specialist who also administers a physical examination to locate painful areas on the patient. Volunteers are taken as subjects to be scanned, some describing muscle pain in certain areas of the upper back. The medical specialist labels a scan as a trigger point if a palpable nodule is present and associated with the pain. In a single scanning session, B-mode images are taken of the trapezius and infraspinatus for both left and right sides of the back/shoulder, as well as shear-wave elastography images. Any specific areas of pain or identified trigger points are captured as well. The ultrasound scans are saved to a server as a DICOM files, which is a medical imaging standard format. From the server, scans are converted to PNG and MP4 files for images and video, respectively, using the Pydicom library for Python. Patient data is hidden for privacy.

3.4.2 GLCM Feature Map Generation and Comparison to Elastography

With the acquired scans, feature maps are generated with a combination of varying parameters of distance d, angle θ, patch size p, and grayscale levels l. The distance between pixels d was varied between values 1, 3, and 5. The angle between pixels θ was held constant at -45 degrees to capture both a vertical and horizontal features between pixel pairs. The patch size diameter p was varied with values 9, 15, and 25 pixels in some cases and 7, 15, and 31 in other cases. Lastly, grayscale levels l varied with values 8, 32, 64, 256. Five features were implemented for feature map generation including homogeneity, dissimilarity, energy, contrast, and correlation.

First, visual comparisons are made between features and the effects of the varying parameters for each feature are provided by visualizations. Next, to help determine the best parameters to use for the GLCM feature maps, the MSE and SSIM scores are calculated for each feature map with varying features, p, d, and l. The scores are calculated by comparing each feature map combination with the elastography ultrasound colormap. For determining the best p, d, and l, the homogeneity feature is used.

3.4.3 Comparison of GLCM Feature Maps and Elastography with MTrPs

In evaluation of MTrP presence, the averages and standard deviations for MTrP and no MTrP are calculated for both the elastography and each feature map, as well as the original B-mode ultrasound image as a baseline. 52 samples labeled as MTrP present and 70 samples labeled as no MTrP present are used, totaling as 122 samples. GLCM feature maps are generated for every sample. The elastography image is also scaled to a linear 1-channel image from the 3-channel RGB representation using a reversed OpenCV colormapping that was previously mentioned. An average and standard deviation is calculated for every image for each feature, elastography colormap, and original B-mode image. Of these 122 averages and standard deviations recorded, the averages of these calculations are taken to give an overall representation of MTrP and no MTrP.

3.5 Results

This section describes the results of the GLCM feature map generation and it's variability depending on the different parameters, the comparison between SWE and the feature maps, and lastly, the comparisons of the presence of MTrPS.

3.5.1 GLCM Feature Map Generation

Example feature maps are provided in Figure 3.5.1 to help visualize each feature. These figures show that the muscle belly tissues tend to have a higher homogeneity and correlation in some areas. The dissimilarity and contrast features highlight major changes in structure such as the myofascial tissues separating muscles or some of the more subtle horizontal muscle fibers. The energy feature shows more detailed changes, highlighting differences more within the muscle belly rather than overall structure the way contrast and dissimilarity does.

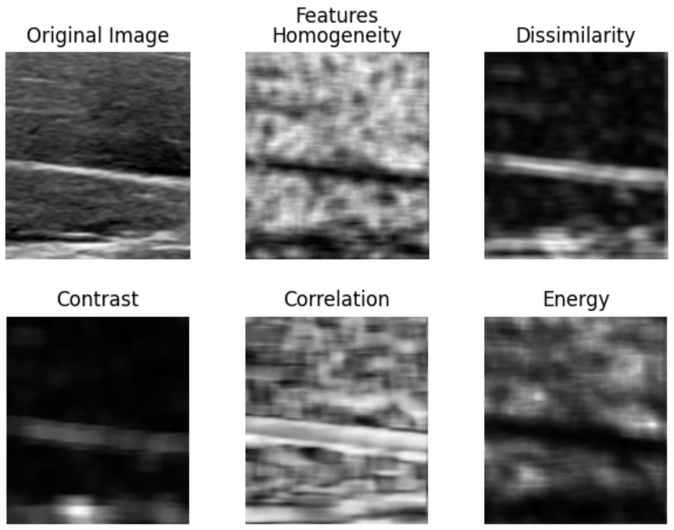
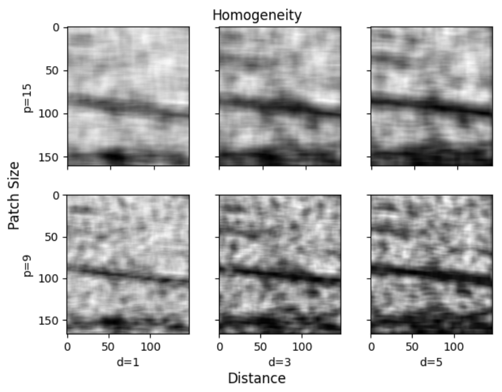
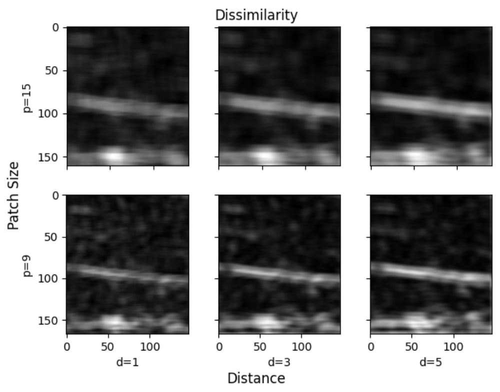


Figure 3.2: Example GLCM feature maps of B-mode ultrasound.

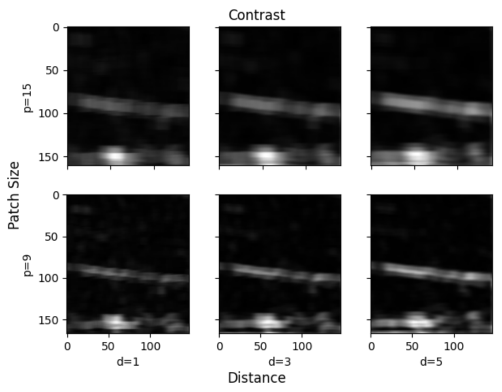
To demonstrate the effects of varying distance d and patch size p, figures of each feature are shown in Figures 3.3a, b, c, d, and e, where the x axis has increasing d and the y axis has increasing p. In every case shown here, smaller distances d tend to result in brighter feature maps. This is more noticeable in the higher intensity maps like homogeneity (Fig. 3.3a), correlation (Fig. 3.3d), and energy (Fig. 3.3e). For the darker maps like dissimilarity and contrast, a smaller d increases brightness of small spots in darker regions of the map, and a larger d actually brightens the already highlighted regions. Increasing patch size p acts almost as a blur for the brighter dominant features (homogeneity Fig. 3.3a, correlation Fig 3.3d, and energy Fig. 3.3e), and as a dilation for the less dominant features like dissimilarity 3.3 and contrast 3.3. The correlation feature maps 3.3 appear to be the most effected by both variables.



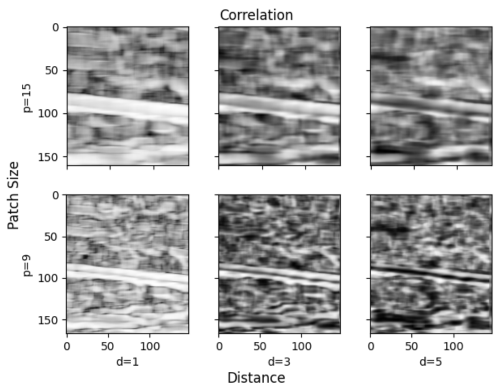
(a) Homogeneity.



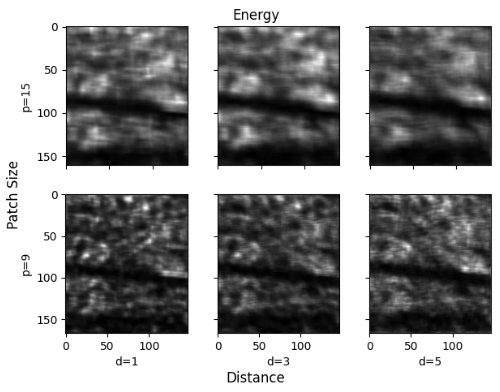
(b) Dissimilarity.



(c) Contrast.



(d) Correlation.



(e) Energy.

Figure 3.3: Varying patch size and distance of each feature map.

3.5.2 SWE and GLCM Feature Maps of B-Mode US Comparison

The image similarity metrics, MSE and SSIM, between the elastography and the feature maps of B-mode ultrasound are represented in Table 3.1 for just one of the tested images. The best MSE score, which is the smallest value and least error, is the correlation feature map. The most error is the contrast feature, but all still better than the fully black image by a fair amount. For the SSIM scores, ranging from -1 (anti-correlated) to 1 (perfectly similar) and 0 being no similarity, all of the features were generally the same scores with homogeneity and dissimilarity having minimally better scores. The same metrics with varying patch size and distance are shown in Table 3.2 and likewise with varying grayscale levels in Table 3.3, again for only one example image that was tested. For patch size p and distance d, the best combination with the least MSE and the best SSIM score was a patch size of 15 and distance of 5. So, the greatest distance and greatest patch size possible yielded the best similarities, although minimally. While these tables represent only one example image, these metrics were calculated with 3 total images, showing a large variance in values because of the varying amounts of masked areas. The results among each image were consistent with with features and parameters were best.

Table 3.1: Example MSE and SSIM metrics between SWE ultrasound image and GLCM feature maps of corresponding B-mode ultrasound image.

Features	**MSE**	**SSIM**
Homogeneity	104.05	0.37
Dissimilarity	135.73	0.37
Contrast	157.24	0.36
Correlation	83.76	0.36
Energy	125.47	0.35
Original	90.68	0.40
Solid Black	232.60	0.26

Table 3.2: Example MSE and SSIM metrics between SWE ultrasound image and GLCM homogeneity feature map of corresponding B-mode ultrasound image: varying patch size and distance.

Patch Size and Distance (p,d)	**MSE**	**SSIM**
7,1	148.68	0.32
7,3	118.42	0.33
7,5	112.02	0.33
15,1	138.20	0.35
15,3	110.28	0.37
15,5	104.05	0.37

Table 3.3: Example MSE and SSIM metrics between SWE ultrasound and GLCM homogeneity feature map of corresponding B-mode ultrasound image: varying grayscale levels.

Grayscale Levels	**MSE**	**SSIM**
8	104.05	0.37
16	90.80	0.37
32	91.41	0.37
64	101.60	0.36
128	114.06	0.35
256	123.05	0.34

3.5.3 MTrPs in SWE and GLCM Feature Maps of B-Mode US

To make comparisons of statistical features regarding the relationship between MTrPs and elastography versus the GLCM feature maps, the averages and standard deviations are shown in Table 3.4. This table shows the averages and standard deviations of each feature maps' average grayscale value with and without a MTrP, as well as the original B-mode for a baseline and the elastography colormap intensity. Box plots with these same statistics for average means and average standard deviations are visualized in Figures 3.5.3, 3.5.3 and 3.5.3. All features are very close when comparing the presence of a MTrP, yet differences do exist. For the presence of MTrPs, a slightly higher homogeneity, dissimilarity, and correlation are found. For no MTrPs present, a higher energy is found. Regardless of these differences, the standard deviations are relatively high, causing much overlap between MTrP and no MTrP.

Table 3.4: Averages and standard deviations of GLCM feature maps of B-mode ultrasound for both MTrP and no MTrP present.

Feature	**Avg $\pm$ StdDev**
Homogeneity MTrP	141 ± 49
Homogeneity No MTrP	149 ± 51
Dissimilarity MTrP	47 ± 36
Dissimilarity No MTrP	43 ± 35
Energy MTrP	47 ± 45
Energy No MTrP	60 ± 51
Contrast MTrP	18 ± 29
Contrast No MTrP	16 ± 26
Correlation MTrP	123 ± 47
Correlation No MTrP	116 ± 49
Original B-mode MTrP	94 ± 37
Original B-mode No MTrP	78 ± 35
Elastography MTrP	145 ± 32
Elastography No MTrP	139 ± 31

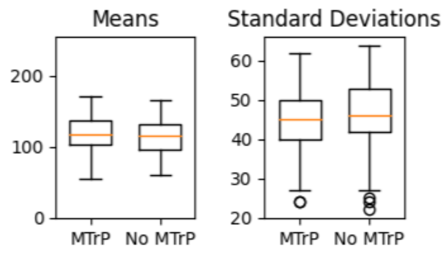


Figure 3.4: Average mean and standard deviation of colormapped SWE with and without MTrPs.

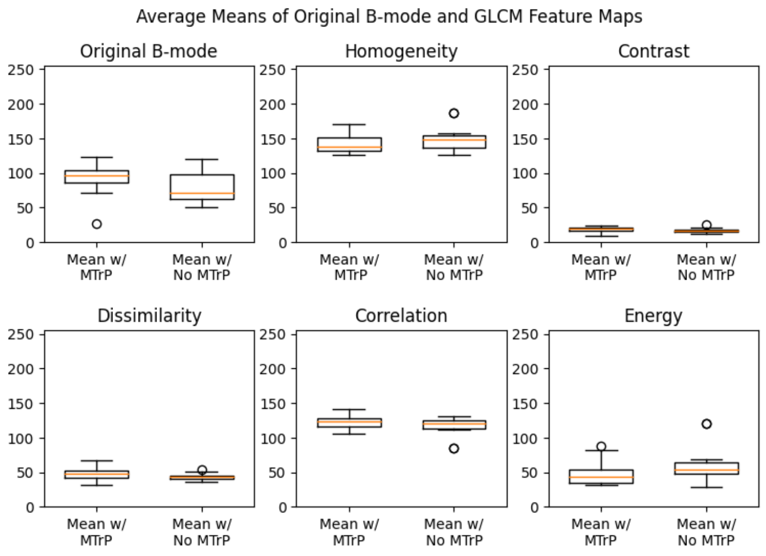


Figure 3.5: Average means of original B-mode and GLCM feature maps with and without MTrPs.

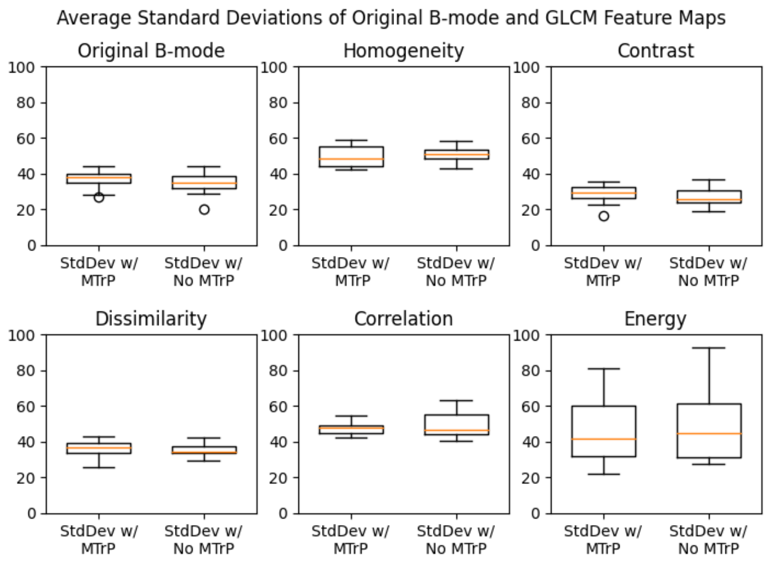


Figure 3.6: Average standard deviations of original B-mode and GLCM feature maps with and without MTrPs.

3.6 Discussion

3.6.1 GLCM Feature Map Generation

From the literature on MTrPs, it is speculated that MTrPs are found within the muscle belly. Based on this speculation and the resulting feature maps, the features homogeneity, correlation, and energy would seem to be the most relevant features for detecting MTrPS since these features tend to highlight patterns found within the muscle belly.

3.6.2 SWE and GLCM Feature Maps of B-Mode US Comparison

For the comparisons made between the shear-wave elastography and the GLCM feature maps of B-mode ultrasound, it was consistent that homogeneity, correlation, and energy had the least error for MSE scores and the better SSIM scores. This was mostly due to the fact that these features highlighted much of the muscle bellies, creating more higher intensity areas than the dissimilarity and contrast features. In terms of the parameters used for the GLCM algorithm, it was consistent that larger patch sizes and distances yielded better scores too. This is because a larger patch size and distance would capture features from a broader level, matching the tendencies of the elastography. The smaller patch sizes created finer-grained highlights, creating almost a noisy effect in the feature maps compared to the elastography. The smaller distances would create too bright and too high contrast, presumably because the texture patterns do not change on a pixel-to-pixel basis at the current resolution. Small distances create a high occurrence of pixel pairs, causing the higher contrasts and brightness. For grayscale levels, it is consistent that lower levels between the values of 16 and 32 gave better scores for the image similarity metrics used. Although the scores may differ a reasonable amount between images, it is consistent that some of the GLCM algorithm parameters scored better than others. These parameters are chosen as a starting point for the machine learning inputs.

3.6.3 MTrPs in SWE and GLCM Feature Maps of B-Mode US

The highest difference in means among the feature maps was energy, however, energy also had one of the highest standard deviations. The original B-mode image had a relatively larger difference, but also had a high standard deviation. Even for the elastography that is designed to measure stiffness, which is one of the speculated characteristics of MTrPS, there was a low difference in means and a high standard deviation. In every case, the standard deviations were higher than the difference of means between MTrP and No MTrP images. Therefore, it is presumed that a simple averaging of pixels in an image is not a sufficient analysis of MTrP presence. A more narrowed analysis on the region of interest is needed to make any confident conclusion about the relationship between MTrPs and any specific image modality or feature map. Furthermore, even though no distinction was determined based on this simple statistical comparison, it further motivates an exploration into the effects of using the feature maps as machine learning inputs.

Chapter 4

GPU Acceleration of GLCM Feature Map Generation

Real-time feature map generation is important for providing a machine learning model with the necessary inputs to give a real-time prediction for a medical specialist performing an ultrasound scan. This chapter describes the process of determining the optimal acceleration of the feature map generation described in Chapter 3. Extensive experimentation demonstrates the effects of many of the tunable parameters that are possible in executing the GLCM feature map generation on a GPU.

4.1 Problem Statement

To generate a single GLCM feature map serially with a CPU takes up to 20 or more minutes. To achieve real-time feature map generation as described in Chapter 3, the algorithm must be parallelized. By distributing the image data to one or multiple GPUs and fine-tuning the hardware utilization, real-time feature map generation becomes possible. This chapter deterministically finds the optimal GPU-based implementation that can tolerate real-time GLCM feature map generation considering a range of GLCM algorithm parameters and GPU implementation parameters.

4.2 Study Design

Image processing lends itself to parallelization given the nature of it's data. Since GPUs are built for performing many tasks simultaneously, they are well-suited for parallelizing this feature map generation process. The GPU acceleration process starts by mapping the input image data to a number of GPUs, then performs the GLCM feature map generation in parallel across the many threads on the GPU, ending with outputting the desired feature maps in real-time. This is done repeatedly to test the limits of the algorithm and the GPU hardware, determining the real-time capability and tolerence of the GLCM feature map generation.

4.3 Methods

The first concept explored in accelerating feature map generation with GPUs is the usage of a single GPU versus multiple GPUs and at what point speedup converges. The second concept explored is the use and design of the hardware on the GPU to make the most of the available resources in terms of memories. These methods are tested with single image inputs. Lastly, the method for processing video input is described to more accurately evaluate an end-to-end real-time processing of an ultrasound scan. These methods seek to find the optimal implementation for acceleration and determine the tolerance of the algorithm under conditions such as downsampling.

4.3.1 GPU Architecture

In general, where CPUs are built for latency, GPUs are built for throughput. Whether designed by NVIDIA, AMD, or Intel, GPUs are divided into streaming multiprocessors or stream processors. These processors have many more transistors dedicated to making computations compared to CPUs, which focus more on latency of memory calls with a number of caches and memories. NVIDIA GPUs are not only more powerful with higher end architectures, but NVIDIA also has developed Compute Unified Device Architecture (CUDA). CUDA provides libraries of C++ code to interact with and program NVIDIA's general-purpose GPUs. Some examples of the methods that CUDA provides include device memory allocation, device memory copying, and thread indexing. CUDA allows for programming GPUs by using what is called a CUDA kernel. This kernel is a block of code that is executed by each thread on the device.

An NVIDIA GPU programmed with CUDA is divided into a grid of thread blocks as specified by the user. For example, if multiplying two matrices together, a two-dimensional block grid may be employed. Within each block, there is a grid of threads also specified by the user. Every thread in a thread block has access to the same shared memory and every thread on the device has access to the global memory. The memory of the device must be allocated for by the CPU before copying data from the host to the device. After the data is copied, the host launches the CUDA kernel, which is code that is run by each individual thread. Upon completion of the kernel, the resulting data is copied back to the host CPU. This workflow is shown below in Figure 4.1.

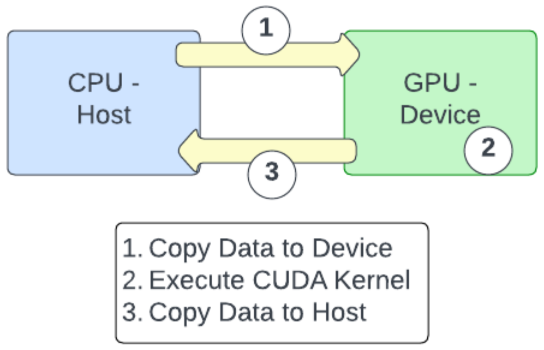


Figure 4.1: GPU dataflow.

There are two main types of memory on each device: global memory and shared memory. Global memory can be accessed by both the host CPU and any thread on the device. This access allows for read and write operations from both sides. Shared memory is on-chip memory that is only accessible by thread blocks and not the host. This memory is much faster than global memory because of it's proximity to the thread registers on the chip itself. A diagram demonstrating a simple example of a 2x2 grid on the GPU with the various threads and different memories is shown in Figure 4.2

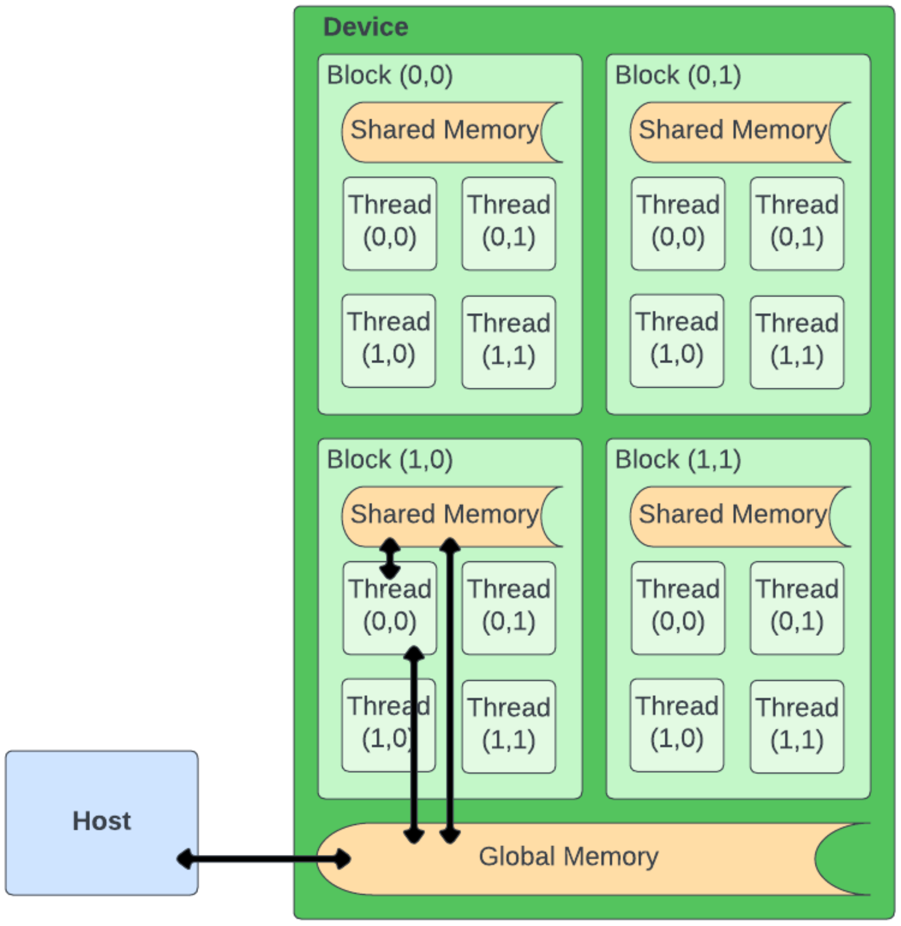


Figure 4.2: GPU architecture with 2x2 grid.

4.3.2 Single GPU Implementation

The first approach utilizes a single GPU to parallelize the feature map generation. The process starts with the CPU reading the input image using the OpenCV library with C++. Using the CUDA libraries, memory is allocated on the GPU and the image data is copied over using *cudaMalloc* and *cudaMemcpy* functions. The CPU then launches the GLCM feature map kernel on a two-dimensional grid to mirror the image data. The GLCM feature map kernel first uses the thread index and block index to determine the respective pixel in the input image (see Equations 4.1,4.2). The thread then executes the typical GLCM algorithm (Algorithm 1) for a surrounding patch about the respective pixel. After processing finishes on the device, the output data is copied back to the host CPU. This parallelization eliminates the outer two for loops of the feature map generation in Algorithm 2. Each thread proceeds to calculate the feature or features necessary, following the equations listed in 3.2,3.3,3.5,3.6, and 3.4. Every thread outputs the feature value for its respective pixel in the image.

$$Row = \texttt{BlockIdx.y} * \texttt{BlockDim.y} + \texttt{ThreadIdx.y} \tag{4.1}$$

$$Col = \texttt{BlockIdx.x} * \texttt{BlockDim.x} + \texttt{ThreadIdx.x} \tag{4.2}$$

4.3.3 Global Memory and Shared Memory

In theory, shared memory is faster since it is on-chip memory and close to the registers, providing shorter read times than reading from global memory. Shared memory really improves performance when there is a lot of data reusage. The data that is stored in shared memory is decided by the user, so speedups are very much determined by design. Control divergence is a key concept to GPU programming since GPUs are made to perform many of the same task in parallel. If tasks begin to differ, say at an important conditional statement, control divergence occurs. Thread blocks are only as fast as their slowest threads.

This method tests one simple approach of using shared memory. A block of the image the size of a thread block is copied to shared memory in a single thread-to-pixel fashion. Pixels that need to be read outside of this range for looping through the surrounding patch are read from global memory.

4.3.4 Multi-GPU Implementation

The second approach utilizes many GPUs for another level of parallelization. In this approach, image data is divided by the number of GPUs available, including any overlapping data needed across boundaries. More specifically, the image data is divided into rows for each GPU, with as many padded rows on each side to account for the patch size of the GLCM. Multi-threading is used on the CPU for configuring the different devices and launching the respective kernels. Multi-threading is necessary for the full utilization of multiple devices since a device is associated with a thread. Multi-processing creates extra overhead for process management and switching between devices. After the image data is distributed to each device and each device is associated with a thread, the CUDA kernel is launched like a single GPU. As each device finishes processing, the CPU thread that launched that device writes the image data to the assigned rows of the output images. Lastly, the threads join and the images are written, resulting in the feature maps.

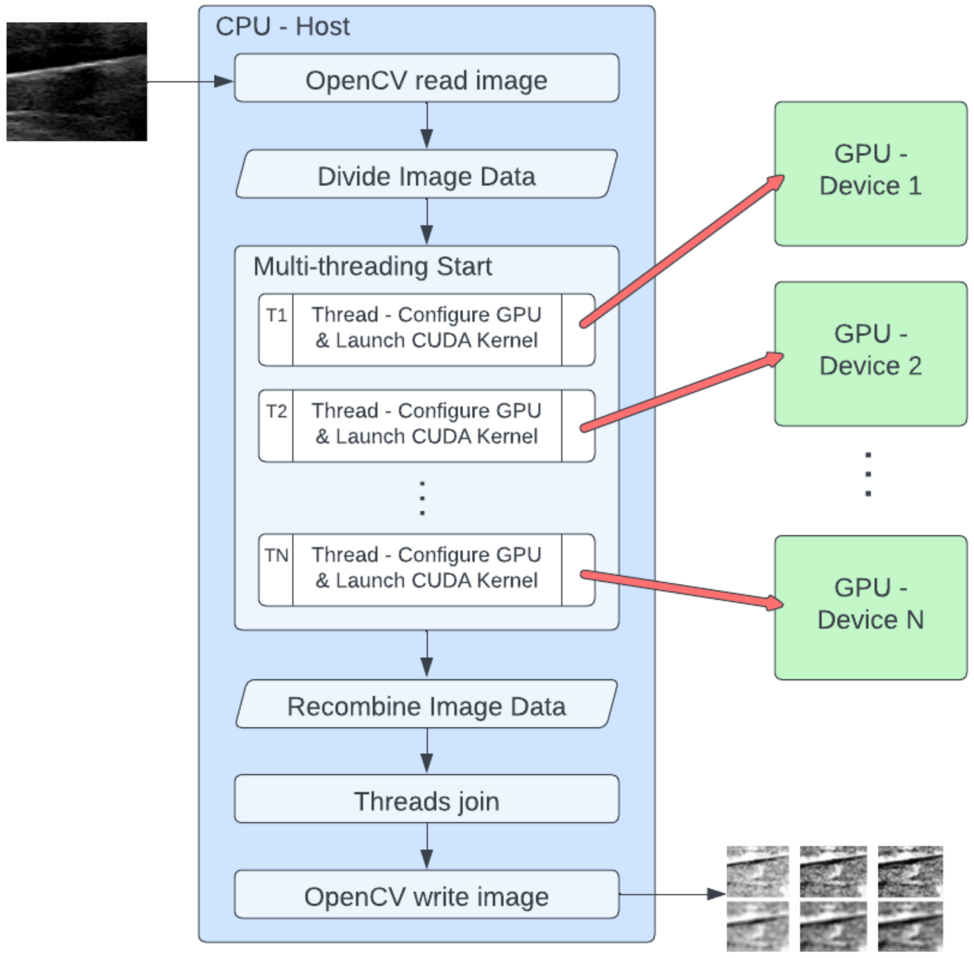


Figure 4.3: Multi-GPU implementation with multi-threading.

4.3.5 Video

Processing video is done by reading frame-by-frame and handling each frame as a single image, as previously described for both the single GPU and multi-GPU implementations. There is a difference for the video processing using multiple GPUs, however. For the multi-GPU implementation of video processing, the multi-threading process starts and ends for each frame. Therefore, memory allocation on the device and data transfer must occur for every iteration through the loop of the video frames. The framerate captured in this experimentation includes the average time spent reading each frame, some basic processing like grayscale conversion of the frame, the multi-threading and CUDA kernel execution, and finally writing each frame.

4.4 Experimentation

This section describes the extensive experiments run to study the behavior and effects of both the hardware specifications and all of the tunable parameters of the software implementations.

4.4.1 Hardware

For both single-GPU and multi-GPU usage, NVIDIA's DGX1 is used, which is comprised of 8 Tesla V100 GPUs. The GPU resources are provided by the Palmetto Cluster, a high performance computing cluster. A few experiments testing device memory usage is used on an NVIDIA Tesla V100, also provided by the Palmetto Cluster. Any serial experimentation is done with an Intel Core i7 processor at 2.90GHz with 16GB Ram. The specifications on memory and CUDA cores and tensor cores of each GPU is found in Table 4.1

Table 4.1: Hardware specifications of GPUs.

	CUDA Cores	**Tensor Cores**	**System Memory**	**High Bandwidth Memory**	**TFLOPS FP32 / FP64**
NVIDIA DGX1 V100 (8 V100s)	40,960	5,120	512GB	16GB per GPU	17.7 / 7.8 per GPU
NVIDIA Tesla V100	5,120	640	32GB	16GB	14 / 7

4.4.2 Software

Parameters of the GLCM algorithm that affect runtime performance include image size, patch size p, and gray levels l. The parameters for the CUDA programming and GPU architecture include block size B, number of GPUs N, and use of shared memory. Experiments are run with almost every possible combination of B, p, l, and N. For video processing, only the number of GPUs N, and grayscale levels l are tested while other parameters are held constant.

First, a serial timing is done using python and the scikit library GLCM method as a baseline. This serial timing is done for a single GLCM feature map of images with sizes 128x128 and 512x512 pixels. Then, timings are done on the GPU for each individual feature as separate kernels, and also one kernel that produces all features. For the rest of the experiments, all features are calculated in the runtimes presented. Next, block size B is compared with increasing image sizes to show which block size is optimal for this algorithm. Block sizes are tested with values 8, 16, and 32 with image sizes of 128x128, 512x512, and 580x760 pixels. These values were chosen since 128x128 would roughly be the size of a smaller region of interest, 512x512 would be roughly the size of an entire muscle belly and surrounding areas, and 580x760 which is closer to a full image size. An experiment is also done with varying grayscale levels used in the feature maps along with varying patch sizes to show which of the GLCM parameters effects runtime the most. This experiment uses patch size p values of 7, 15, and 31 and grayscale values l as 8, 64, 128, and 256. Patch size of 31 for these experiments often produced skewed maps because of a high number of occurrences overflowing the datatype, but was still considered for runtime experiments. For assessment of the multi-GPU based implementation, runtimes and speedup is shown for increasing number of GPUs used. Runtimes and speedups are also shown for multi-GPU use of video input. In both of these experiments, all other parameters were held constant except varying grayscale levels were also recorded. The video used for input has dimensions of 256x256 and a framerate of 30, but openCV processes the frames as fast as possible regardless of the framerate specified in the MP4 format unless programmed to do so.

4.5 Results

This section provides runtime plots, speedups, and timings of the GPU-based acceleration of the GLCM feature map generation. First, the single GPU implementation is shown with comparisons of varying parameters of both the GLCM feature map algorithm and the CUDA programming, including a comparison of the use of device memory. Lastly, the results of the multi-GPU and video experiments are shown.

4.5.1 Single GPU Implementation

Comparing the runtime of generating a single GLCM feature map on the GPU versus the CPU resulted in over 16,000x speedup for an image of size 128x128 pixels, and nearly 70,000x speedup for the image of size 512x512 pixels. This was calculated from the serial CPU implementation taking roughly 83,000 milliseconds whereas the GPU implementation took only 5 milliseconds with the same parameters and an input image of 128x128 pixels. A grayscale level of 8 was only used for the serial CPU implementation because anything greater becomes so slow, feature map generation is rendered useless. The average runtimes of generating each feature map for a 512x512 pixel image are shown in Table 4.2 and in Figure 4.5.1. The averages with their standard deviations are from 3 recorded runs, and all runs showing very consistent results. The quickest feature map to generate is the energy feature at 57 milliseconds, whereas the slowest feature to generate by far is correlation at 262.484 milliseconds. The rest of the features are roughly close in runtime to energy, with homogeneity being the slowest with the exception of correlation. For a thorough experimentation of determining optimal CUDA thread block size, Figure 4.5.1 shows varying block sizes and image sizes. In every case of image size tested, block size B of 16 yielded the fastest runtimes, and was more apparent as image size increased. For the last experiment of the single GPU implementation, Figure 4.5.1 shows runtimes of varying patch sizes as grayscale levels increase. Patch size variations had little to no effect on runtime. For the maximum number of grayscale levels used, patch size p of 31 began to show some decceleration, but as a reminder, patch size of 31 starts to produce skewed feature maps because of too many occurrences causing datatype overflow.

Table 4.2: Runtimes of GLCM feature maps.

Feature	**Runtime (ms)**
Homogeneity	18.656 ± 0.04
Dissimilarity	13.426 ± 0.10
Contrast	13.177 ± 0.07
Correlation	49.615 ± 0.33
Energy	13.186 ± 0.04
All	80.245 ± 0.63

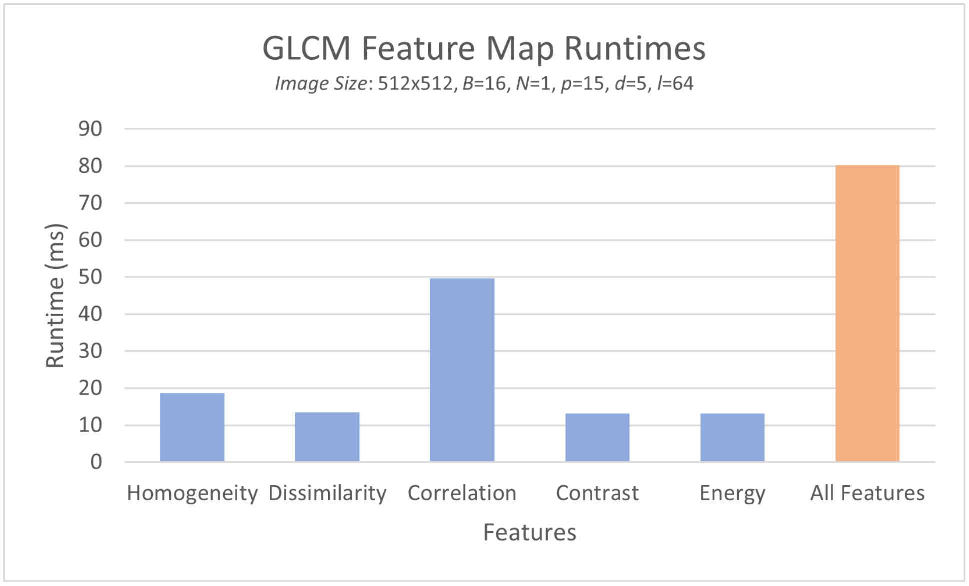


Figure 4.4: Runtimes of GLCM feature maps.

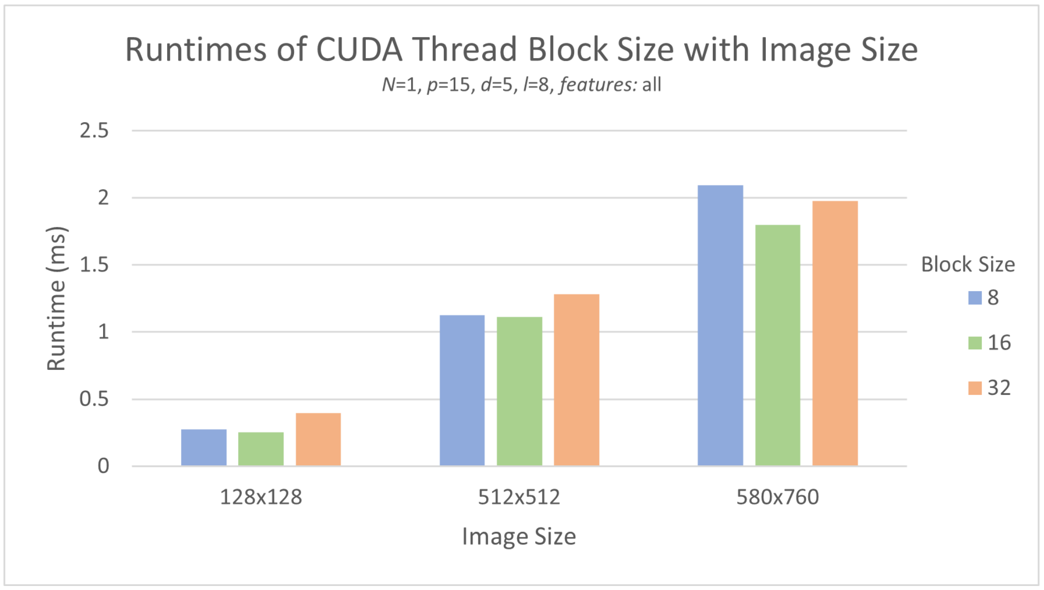


Figure 4.5: Runtimes with varying CUDA thread block size and image size.

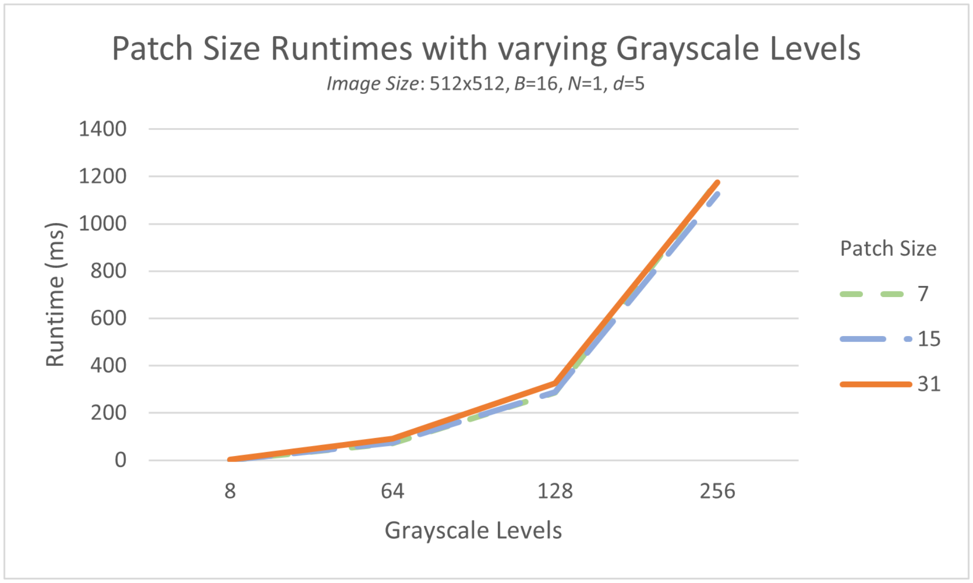
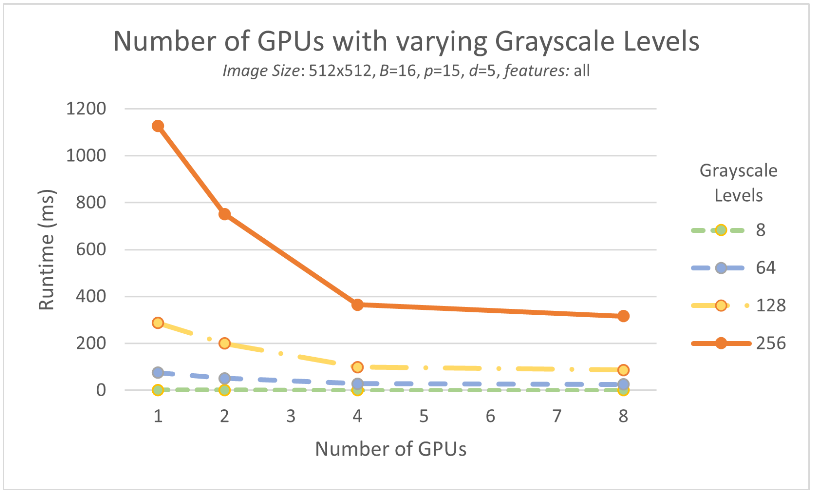


Figure 4.6: Runtimes with varying patch size and grayscale levels.

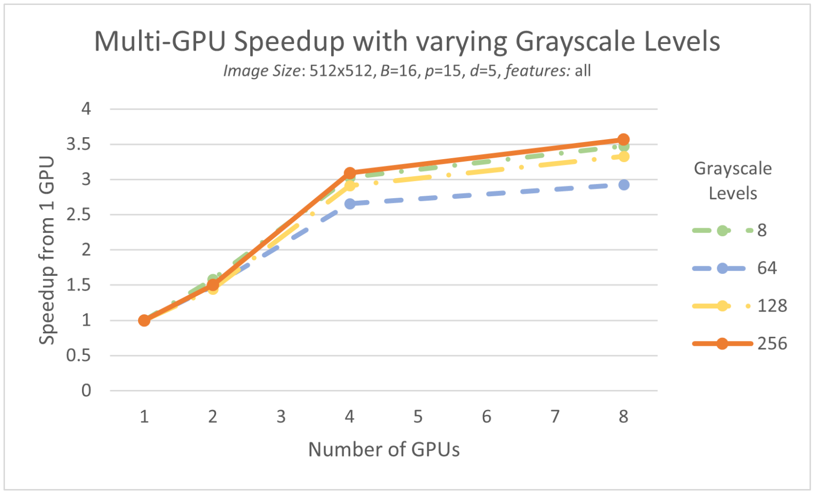
For testing the use of the shared memory on the device, runtimes were nearly the same. Any difference was negligible. These results are further discussed in Section 4.6.

4.5.2 Multi-GPU Implementation

For the experimentation of using multiple GPUs, Figure 4.7 shows the runtimes (a.) and speedups (b.) of increasing numbers of GPUs with each grayscale level parameter for processing individual images. The runtimes continue to improve as the number of GPUs increase, but this increase starts to level out between 4 and 8 GPUs. Speedups are mostly similar for all values of grayscale levels except grayscale levels of 64 show the least speedup. Similarly, Figure 4.8 shows the runtimes (a.) and speedups (b.) of increasing numbers of GPUs with each grayscale level parameter, but for video input. Framerate only improves minimally increasing to two GPUs for grayscale levels of 64, 128, and 256. For using 8 grayscale levels, there is a consistent decline in framerate as number of GPUs increase. The speedup in Figure 4.8a, shows the slight increase for all grayscale levels except 8.

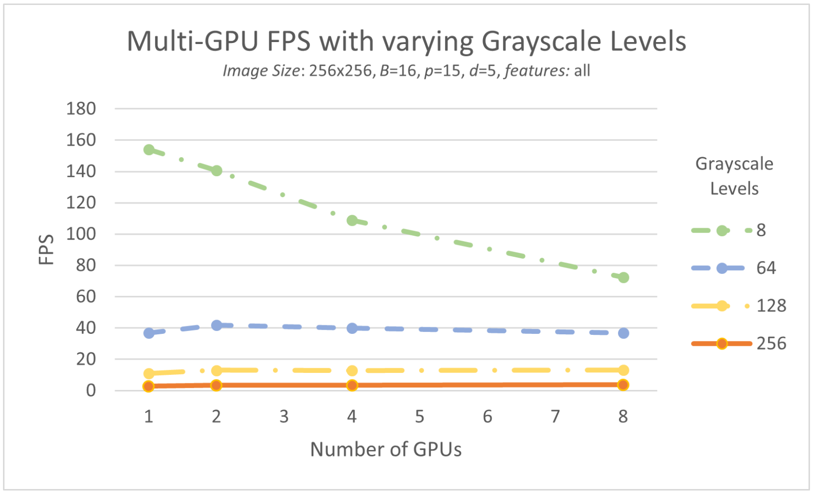


(a) Runtime.

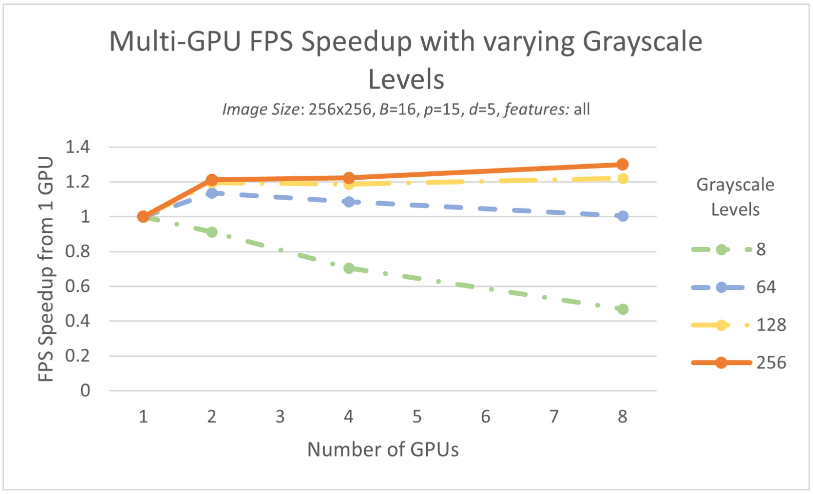


(b) Speedup.

Figure 4.7: (a) Runtimes and (b) speedup of multi-GPU based implementation with varying grayscale levels of single image.



(a) FPS.



(b) FPS Speedup.

Figure 4.8: (a) FPS and (b) FPS speedup of multi-GPU based video implementation with varying grayscale levels.

4.6 Discussion

4.6.1 Single GPU Implementation

For the feature map runtimes, dissimilarity, contrast, and energy were all roughly the same. Homogeneity was slightly higher, which is because of the few extra operations used in calculation. Correlation is the slowest as expected. This is due to the fact that correlation must loop through the GLCM three times to calculate the means, variances, and finally the correlation values. Part of the runtime for each feature map kernel is from the construction of the GLCM itself, which is why calculating all features isn't simply a summation of each feature runtime. As the number of grayscale levels increase, the runtime increases exponentially because each feature loops through the GLCM which is a square matrix of size equal to grayscale levels.

For varying block sizes, a size of 16x16 threads consistently yielded the best performance. This is most likely due to control divergence in blocks along the boundaries of the image data. Since the image is split into the thread blocks starting in the top left of the image, blocks along the bottom and right sides of the grid may have a lot of threads diverging at conditionals if the dimensions are not divisible by the block size. This concept is demonstrated slightly in the results where the image dimensions are not divisible by the block sizes, specifically the 580x760 pixel image in 4.5.1. It is ideal for every thread to be executing the same operations.

For the varying patch sizes and grayscale levels, it is evident that patch size has minimal effect on runtime compared to the number of grayscale levels used. This is because varying patch size only effects the number of pixels that are initially read to build the GLCM. Varying grayscale levels greatly increases the number of calculations since features are calculated from the GLCM where size is determined by grayscale levels, which in most cases will be larger than the patch that is being analyzed. From the findings in Chapter 3, feature maps using grayscale levels ranging from 8 to 32 were minimally better for comparisons to the elastography, therefore, processing up to 32 grayscale levels in real-time is the goal. This was achieved with just a single GPU and single image input, but becomes more difficult as grayscale levels reach higher than 64. However, an image size of 512x512 is relatively large, so smaller images may be able to handle higher numbers of grayscale levels.

4.6.2 Shared Memory and Global Memory

The implementation of shared memory showed no improvement over the use of global memory. This was a surprising result at first, but after closer consideration of the implementation, the results make sense. Because the size of the image chunk stored in shared memory is only as big as the block size, threads are diverging at the condition checking if the pixel is in shared memory. Major control divergence is occurring for when threads are reading outside of the block for looping through the GLCM patch. There are likely too many calls to global memory for there to be any improvement in runtime. Thread blocks are only as fast as their slowest threads, which is why understanding and considering control divergence is crucial in kernel implementation.

4.6.3 Multi-GPU Implementation

For single image processing, speedup from using one GPU is roughly constant up until using 4 GPUs, where speedup begins to converge with using up to 8 GPUs. The convergence of speedup is speculated to be from the overhead of memory allocations and copying on and off each device. If the number of GPUs continued to increase, there would come a point at which image data would be copied multiple times for GPUs since each GPU takes a padding of image data around it's designated boundaries. This would begin to cause serious inefficiency of resources, especially at the point when the designated number of rows to a GPU would become less than the patch size. The decrease in runtime with the multi-GPU implementation made real-time processing possible for using 64 grayscale levels and even almost 128 grayscale levels.

4.6.4 Video Processing

For video processing, the multi-GPU implementation converged much sooner and showed very little improvement, and even a decline for a grayscale level of 8. This is due to the implementation design, which has room for optimization. Specifically, since the device memory allocation occurs after every frame is read, this overhead cannot keep up with the incoming frames. This overhead really only begins to negatively effect the algorithm when using a grayscale level of 8 and even partially 64 because the speed at which the algorithm performs at a grayscale level of 8 is already so much faster comparatively. To avoid this overhead, however, would require a method that keeps the memory on the device allocated and a pointer to this memory is kept between thread lives. With the current implementation, a thread's lifespan is as long as it takes to process a single frame. Within the thread's lifespan, the memory is allocated, utilized in the algorithm, and then freed. The thread would either need to stay alive for the entire video processing, or share the memory allocated to the next thread handling the next frame.

Overall, the FPS would certainly be acceptable with the multi-GPU implementation using up to 64 grayscale levels, but would be too slow for real-time feedback at 128. Again, these capabilities are also slightly dependent on the video dimensions, so a larger frame dimension may be less tolerant, and a smaller dimension would be more tolerant. Also, the definition of real-time can be slightly subjective. Some may consider real-time to be at least 20 FPS, or in some cases much higher. With the goal of real-time inference in the context of a live ultrasound scan, near-real-time may be acceptable, but with the sensitivity of the probe handling during an ultrasound scan, a higher FPS would always be ideal.

Chapter 5

Deep Learning Model

As a step towards pain quantification of MPS, it is imperative to be able to first detect the presence of an MTrP. This chapter describes the initial machine learning methodologies used for detecting MTrPs from B-mode ultrasound. These methodologies include an end-to-end machine learning system that takes B-mode ultrasound as inputs, performs varying preprocessing techniques, and implements different machine learning strategies to make a binary classification of the presence of MTrPs. Combinations of different strategies and techniques including those described in Chapters 3 and 4 are evaluated and compared.

5.1 Problem Statement

Considering the limitations of data collection in this scenario, different machine learning strategies are employed for assessment of MTrP detection. It is anticipated that generating GLCM feature maps of the B-mode ultrasound inputs would assist the model by emphasizing the texture features that are hypothesized to be correlated with MTrPs, given the limited dataset size. This chapter tests this hypothesis by comparing machine learning models with and without the use of GLCM feature map generation for additional inputs.

5.2 System Design

With the objective of this task being MTrP detection and exploring the effects of using GLCM feature maps, two machine learning approaches are taken. The first approach uses a standard deep learning approach with transfer learning and data augmentation to try to detect the presence of MTrPs. The second approach uses the same transfer learning and data augmentation strategies, but rather for inputs, it implements GLCM feature maps of B-mode ultrasound in addition to the normal B-mode ultrasound.

5.3 Methods

This section describes in more detail the machine learning strategies used in the deep learning model, as well as how the GLCM feature maps are implemented in the model.

5.3.1 Transfer Learning

While transfer learning improves models trained on small datasets, it is actually beneficial for most models in making the training faster and more efficient. Even when supplied with a large, well defined dataset, transfer learning helps to reach convergence faster, making the learning more efficient. However, transfer learning is used to address the bigger problem in this scenario, which is a limited dataset size.

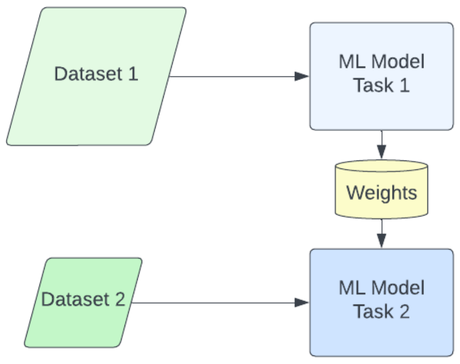


Figure 5.1: Transfer learning general workflow.

Two main datasets for pretraining are considered in this work, but only one is experimented with. The first dataset that is actually employed is ImageNet [13]. It's been clearly demonstrated in other works of it's ability to be used as a basis for transfer learning. The second dataset that was considered is RadImageNet [30]. Somewhat surprisingly, many works still have shown better performance using ImageNet rather than RadImageNet, despite the more similar nature of the data. Because of the consistent performance provided by ImageNet, ImageNet is the decided dataset for pretraining in the transfer learning step.

A deep convolutional neural network is used since CNNs perform exceptionally well with learning features of image data. A few of the most powerful and most efficient architectures used for transfer learning include ResNet, VGG, and Inception. For this work, the ResNet architecture is used, specifically ResNet-50 given it's balance between efficiency and performance.

5.3.2 Data Augmentation

To increase the size of the training data, data augmentation is used. Only two types of augmentations are used which include random rotation up to 45 degrees and random horizontal flipping. Vertical flipping or any other more complex augmentations are not used since it is yet to be determined the important properties of the MTrPs. It is anticipated that the samples need to maintain a mostly horizontal orientation given that the muscle fibers and fascia tissues appear horizontal in the scan. A significant change in the nature of the data like a 90 degree turn of the muscle fibers, may hinder the model's ability to learn patterns of a MTrP. Additionally, vertical flipping is not used since the top of the images may contain layers of fat tissue close to the surface of the body. These are merely assumptions for this work and still a facet to explore, hence the use of only basic augmentations.

5.3.3 GLCM Feature Maps

To evaluate the contributions of using GLCM feature maps of B-mode ultrasound to detect MTrPs, they are generated as a part of the preprocessing phase of the machine learning system. Along with the B-mode ultrasound as input, feature maps are generated for every feature listed in Chapter 3. The combination of these features is used as additional inputs to the machine learning model in an effort to emphasize the texture patterns. Similar to how an RGB input image would have three channels as inputs to a model, this approach would have a channel for every feature, which is represented as a grayscale heatmap. Additionally, another channel is used for the original B-mode image, totalling as a six channel input.

5.4 Experimentation

For determining the effects of using the GLCM feature maps of B-mode ultrasound as machine learning inputs, performance of two models are compared. The first model uses the original B-mode ultrasound image as a single channel grayscale input. The second model takes the original B-mode image in addition to it's respective five GLCM feature maps as a six channel input. Both models implement data augmentation and transfer learning approaches to help with the challenge of limited data.

5.4.1 Preliminaries and Data Collection

As described in Chapter 3, volunteer patients come to a lab with a Philips Epiq Elite 7 ultrasound machine to be scanned. A physical examination is done by a medical specialist or examiner to identify painful areas in the upper trapezius muscles, which are then labeled appropriately. Scans are taken of these areas and labeled, as well as scans of normal areas with no pain are taken and labeled. The dataset size includes 40 labeled samples from 5 different subjects. There are 22 samples being positive for MTrP present and 18 samples being no MTrP.

5.4.2 Model Architecture and Setup

The model is programmed in python using the PyTorch libraries. For the transfer learning, the ResNet50 architecture is used for the model and is pretrained on ImageNet. The input layer is changed from a 3-channel input to a single channel and six channel input for the original B-mode image and GLCM feature map implementations, respectively. To adjust for a binary classification problem, the fully connected output classification layer is changed from 1000 to 1 and a sigmoid activation function is added. The code for this is shown in the code snippet below 5.4.2. For data augmentation, only two transforms are used, which are a random rotation of up to 45 degrees and a horizontal flip. These augmentations are applied to both training datasets, increasing the training dataset sizes by a factor of 5. The model overview is provided in Figure 5.4.2.

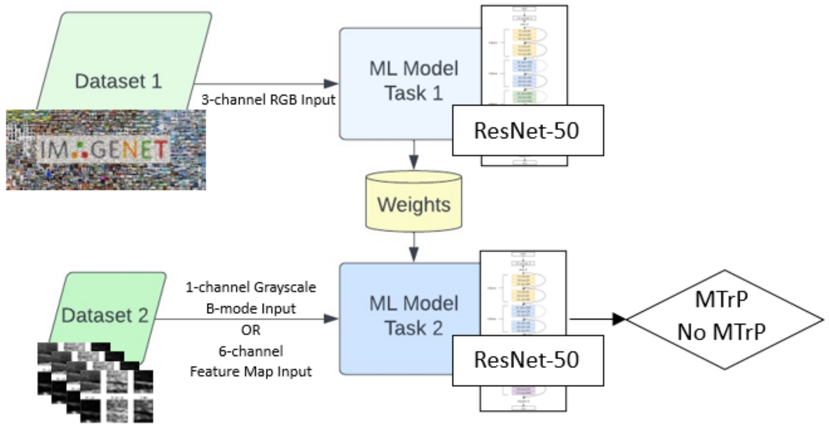


Figure 5.2: Machine learning model implementation with ResNet-50 architecture and ImageNet transfer learning.

As shown in Algorithm 3, the code loops through the number of folds with each fold having a different validation set, and the remaining samples being the training set. The training set then undergoes data augmentation, increasing the dataset by a factor of 5. The mode then proceeds looping through each epoch, and then through each training batch provided by the dataloader. Every batch goes through the learning process and then the validation set is predicted, saving the best model.

Algorithm 3 ML training algorithm with cross-validation

```
for $Fold$ in $KFolds$ do
    Assign training and validation data
    Augment training data
    for $Epoch$ in $NumEpochs$ do
        for $phase$ in $Training, Validation$ do
            for $batch$ in $Dataloader$ do
                Predict
                Calculate Loss
                if $training$ then
                    Back Propogate
                    Optimize
                    ...
                end if
            end for
        end for
    end for
end for
```

```python
# Loading pre-trained ResNet-50 model on ImageNet
resnet50_model = models.resnet50(weights='IMAGENET1K_V1')
# Change number of channels for input layer
if use_glcm:
    # 6 channel input for GLCM
    resnet50_model.conv1 = nn.Sequential(
        nn.Conv2d(6,64,kernel_size=(7,7),stride=(2,2),padding=(3,3),bias=False),
    )
else:
    # Single channel input for grayscale B-mode Image
    resnet50_model.conv1 = nn.Sequential(
        nn.Conv2d(1,64,kernel_size=(7,7),stride=(2,2),padding=(3,3),bias=False),
    )
# Change classification layer for binary classification
resnet50_model.fc = nn.Sequential(
    nn.Linear(in_features=2048, out_features=1, bias=True),
    nn.Sigmoid()
)
```

Listing 5.1: Code Snippet of loading Pre-Trained ResNet-50 with ImageNet weights and changing the input and output layers.

5.4.3 Hyperparameters

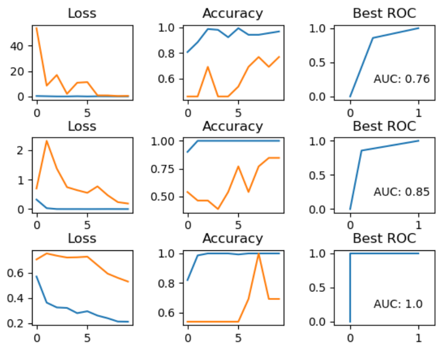
The loss function used is binary cross entropy loss. A learning rate of 0.001 with an Adam optimizer is used. A learning rate scheduler is also used to decay the learning rate by a factor of 0.1 after 5 epochs. An L2 regularization or weight decay with the optimizer is set to 0.01. Varying batch sizes were used in the dataloaders ranging from 10 to 100. Both models train for only 10 epochs since overfitting begins to occur with such little data. For cross validation, 3 folds are used since anything greater than 4 starts to create too small of a validation set.

5.4.4 Performance Analysis

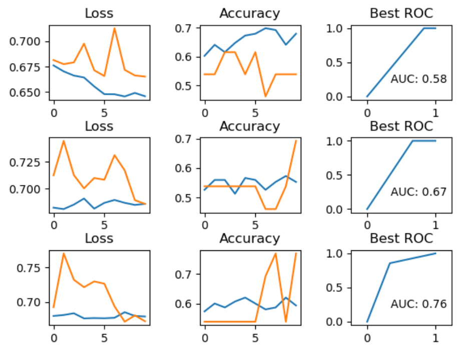
For performance analysis, cross validation is employed to gain a more accurate understanding of the model given such a low amount of data to test with. Data augmentation is not used on the validation set. Confusion matrices are created for each validation fold which are then summed to create an overall confusion matrix representation of each model. Precision, recall, and average accuracies are calculated from this confusion matrix. Loss curves and accuracies of throughout training are visualized in figures for example models. ROC curves are generated for the best models of every fold, as well as the area under the ROC curve. 3-Fold cross validation is run 5 times for each of the two proposed model implementations.

5.5 Results

As shown in Figures 5.3 and 5.4, the models have a mostly consistent decline in loss with a few spikes and increases in the validation loss. The training accuracies generally reach 100% in most cases, but fluctuate just a little in some validation folds. The validation accuracies fluctaute a lot more, but still reach around 70-80% on average. The ROC curves for nearly every model was above the diagonal, showing that the True Positive Rate to False Positive Rate ratio was positive.

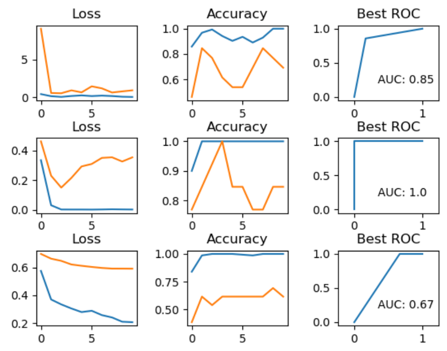


(a) Example.

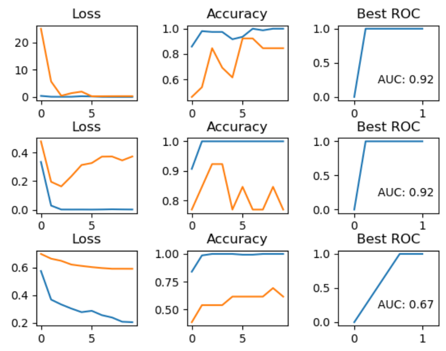


(b) Example.

Figure 5.3: Loss, accuracy, and ROC curve of example 3-fold cross-validated models using only B-mode images as input (training in blue, validation in orange).



(a) Example



(b) Example

Figure 5.4: Loss, accuracy, and ROC curve of example 3-fold cross-validated models using GLCM feature maps of B-mode images as input (training in blue, validation in orange).

Summing the confusion matrices for each fold in the cross validation results in a confusion matrix like Tables 5.1 and 5.2 for each model run. Tables 5.1 and 5.2 are the sums of all 5 model runs with 3-fold cross validation. Each model was able to achieve near-perfect results and perfect recall at least once, with only a few false positives. The model using only B-mode images as input was a bit more sporadic across runs, however, resulting in the confusion matrix shown in Table 5.1. The model using GLCM feature maps of the B-mode image had a few less false positives, and far less false negatives as shown in the confusion matrix in Table 5.2. The model using the GLCM feature maps as input achieved perfect recall twice out of the 5 runs, having no false negatives. Averaging the precision, recall, f-score, and accuracy from these matrices results in the scores found in Table 5.3. The precision, recall, f-scores, and accuracy calculated from 5 different trained models, were somewhat consistent with a small variation for the model using only B-mode images. The model using the GLCM feature maps with the B-mode image not only had much more consistent scores, but also scored better in every category. Both models showed a relatively high recall, especially the model using GLCM feature maps.

Table 5.1: Summed confusion matrix and normalized matrix for 5 runs of 3-fold cross-validation model using only B-mode inputs.

	Actual MTrP	**Actual No MTrP**
Predicted MTrP	85	28
Predicted No MTrP	20	62

	Actual MTrP	**Actual No MTrP**
Predicted MTrP	0.44	0.14
Predicted No MTrP	0.10	0.32

Table 5.2: Summed confusion matrix and normalized matrix for 5 runs of 3-fold cross-validation model using GLCM feature map inputs.

	Actual MTrP	**Actual No MTrP**
Predicted MTrP	102	21
Predicted No MTrP	3	69

	Actual MTrP	**Actual No MTrP**
Predicted MTrP	0.52	0.11
Predicted No MTrP	0.02	0.35

Table 5.3: Average precision, recall, F-scores, and accuracy with standard deviations for 5 runs of 3-fold cross-validation for both models.

	Model 1: Original B-Mode Inputs	**Model 2: GLCM Feature Map Inputs**
Precision	0.77 ± 0.10	0.83 ± 0.05
Recall	0.81 ± 0.16	0.97 ± 0.03
F-Score	0.78 ± 0.09	0.90 ± 0.03
Accuracy	$75.4 \pm 9.7\%$	$87.7 \pm 4.2\%$

5.6 Discussion

From the loss and accuracies shown in Figures 5.3 and 5.4, it is apparent that the model is overfitting from very early on. The training accuracy almost immediately reaches 100% while the validation accuracy slowly increases and fluctuates a lot. This is believed to be due to too much repetition in the data augmentation. This is a difficult challenge to overcome however, because the data augmentation is a necessary step in achieving the performance that is demonstrated here. There may be room for some optimization with further hyperparameter tuning and creating more variability in the data augmentations, but this is left for a future work. From looking at some of the validation loss curves, there are spikes, which is most likely due to unlucky batch distributions. Again, since the dataset size is so small, smaller batch sizes work better for the model, causing the spread of some batches to not be very equal. A batch containing many negatives and few positives could cause the loss to spike as seen in some of these loss curves. In most cases, for both model implementations, the ROC curve was above the diagonal and the area under the curve was greater than 0.5. This demonstrates that generally, both models performed better than a complete guess. In some cases, the ROC curve goes straight to the point $(0, 1)$, which shows that fold was 100% accurate and the area under the curve was 1. This happened occasionally for both models. This was mainly only possible because of the small dataset size and is the reason for cross validation, and even averaging multiple cross validation scores. The confusion matrices and metrics such as precision, recall, and f-score provide a better understanding of the model performances in this case.

During a select few validations, both models were able to achieve near-perfect performance with the best for each reaching perfect recall. Given that the model performances seem rather sporadic based on the loss curves and accuracies, Table 5.3 gives much more insight into the overall performance. First, considering the precision of both implementations, the models using GLCM feature maps as input are more precise than those using only B-mode. This means that using the GLCM feature maps allowed for a higher percentage of MTrPs detected among all images that contained MTrPs. Similarly, using the GLCM feature maps yielded a much higher, near-perfect, recall. This means that in many validations, using the GLCM feature maps enabled detection of nearly all, and in some cases all, samples where MTrPs were present. If precision and recall are equally important, the F-score is a representation of both of these metrics. The F-score is still considerably better for the implementation using GLCM feature maps. This is presumably because of the much higher recall and the precision scores are very close considering the standard deviations of the scores.

It is necessary to determine which metric, either precision, recall, or F-score, is more important for this application of detecting MTrPs. For cases where serious disease like cancer is the application, a false positive would be preferred over a false negative. However, for the application in this work, a false negative may be preferred over a false positive. The detection of MTrPs is a step towards overall pain measurement of the MTrPs, where treatment and diagnosis is a challenge, but not life-threatening, and drug misuse is common. This motivation gives reason to precision being more important than recall, since it would be preferable to have a false negative and not treat it, rather than having a false positive and contribute to drug misuse. By this reasoning, the model using the GLCM feature maps is still preferable over the model using only B-mode ultrasound, although by a small margin because of the inconsistency in the model using only B-mode ultrasound. Additionally, the model would not be used for diagnosis, but only as an aid in diagnosis and is eventually intended to monitor the severity of the MTrP.

Chapter 6

Conclusion

This chapter summarizes the overall goals and findings of this work. The main objective of detecting Myofascial Trigger Points (MTrPs) in B-mode ultrasound was approached in three steps. The first step used the GLCM texture recognition algorithm as a preprocessing step to create feature maps that were anticipated to be useful as machine learning inputs, emphasizing certain features of an image to improve the efficiency and accuracy of a machine learning model. The second step was to accelerate this feature map generation process to provide real-time processing and prediction. Finally, the last step was to implement a model that uses GLCM feature maps of B-mode ultrasound as inputs to detect MTrPs and compare with a model that takes the original B-mode image as input. Furthermore, this chapter describes the future work that was inspired from these findings. Deeper explorations in each of these three main topics are desired.

6.1 Summary of Conclusions

6.1.1 GLCM Feature Map Generation

The GLCM algorithm is used to create feature maps of standard B-mode ultrasound images that emphasize certain statistical features such as homogeneity, dissimilarity, contrast, correlation, and energy. These feature maps are anticipated to be helpful in improving the efficiency and accuracy of a machine learning model for detecting the presence of MTrPs. Knowing that elastography ultrasound is used to measure stiffness in certain tissues, GLCM feature map values were compared to shear-wave elastography to determine if any correlations exist between GLCM features and stiffness. This evaluation was also used as a starting point for what GLCM algorithm parameters to use for the machine learning map inputs. The comparison to elastography showed minimal similarity, but certain parameters scored better on the similarity metrics, which these parameters were then chosen for the machine learning inputs. Next, GLCM feature maps were compared with and without the presence of MTrPs. This comparison was inconclusive because of the large standard deviations of feature values. Even among the elastography images, the comparison was inconclusive. However, based on the comparison techniques used, which involved averages and standard deviations of whole images, it was decided that a narrowed analysis is needed.

6.1.2 GPU Acceleration

A study is done to determine an optimal method of generating the GLCM feature maps in real-time. Real-time generation is needed for real-time inference and feedback for a medical specialist performing an ultrasound scan. This study examines the use of a single GPU and multiple GPUs for both single image processing and video processing. An extensive experimentation is done to find the best set of parameters for both the GLCM algorithm and variables used in the GPU architecture design and programming. The use of a GPU provided exceptional speedup, making real-time processing feasible depending on a few of the parameters used, with the main contributor being number of grayscale levels. Implementing a multi-GPU based method showed further improvement for single image processing, however, not as much for video processing. Because of a design choice made in the multi-threading process for video processing, multi-GPU usage began to hinder the framerate of feature maps generated with a lower amount of grayscale values and showed very little speedup for greater grayscale values. Overall, compared to a serial approach, the GPU acceleration of the GLCM feature map generation made real-time processing feasible and made it possible to use as inputs for the machine learning model.

6.1.3 Machine Learning

A machine learning approach is employed to detect MTrPs using B-mode ultrasound imaging. Under the constrictions of a small dataset, machine learning strategies like transfer learning and data augmentation are used in an effort to improve accuracy. It is also anticipated that using feature maps generated from the GLCM texture recognition algorithm would also improve accuracy by emphasizing certain features that are believed to be present in the image data not seen by the naked eye. Various models are run to find whether MTrPs are detectable in B-mode ultrasound and if using GLCM feature maps aid in the task.

Because of the small dataset, validation scores are somewhat inconsistent, but based on multiple runs using cross validation, the use of GLCM feature maps of the B-mode ultrasound images showed better precision, recall, and accuracy. It is assumed that precision is of slightly more importance because of the overall pain treatment motivation and unfortunately, the precision score was considerably lower compared to recall. In nearly every case, both model implementations showed performance better than that of guessing, which is significant considering the difficulty of identifying MTrPs in the B-mode ultrasound. Considering the performance achieved under the limitations of dataset size, these results motivate continued exploration.

6.2 Future Work

6.2.1 GLCM Feature Map Generation

A deeper look into correlating the GLCM features to both elastography measurements and the presence MTrPs would be a very beneficial analysis. To correlate GLCM features to elastography would provide a means for stiffness measurement without the elastography mode on an ultrasound machine, increasing accessibility. To make this correlation would require a more in-depth analysis and narrower comparisons between the GLCM feature values and the elastography measurements, say in smaller regions of interest, whereas the current analysis simply compared entire image averages and standard deviations. To statistically demonstrate a relationship between GLCM features of B-mode utlrasound and either of elastography or MTrP presence would further motivate the use of GLCM feature maps and contribute greatly to the detection and measurement process of MTrPs. Knowing the correlations would enable a better implementation with the machine learning model, leveraging the features that are known to be related to the MTrPs.

6.2.2 GPU Acceleration

While the implementations of GPUs demonstrated great performance, there is still room for optimization, especially for the video processing. It would be beneficial to eliminate the overhead of allocating device memory for each frame. This should be possible by either keeping the threads alive for the duration of the video and redesign how the data is shared among threads, or determine a way to keep the memory allocated on device and share the pointer between threads. This would make almost any variation of the algorithm possible to process in real-time, specifically using all 256 grayscale levels. Also, designing a better implementation for the use of shared memory is expected to increase speedup. This would be done by increasing the boundaries of the block of data that is copied to shared memory so that threads can access pixel data in the GLCM patch that would otherwise be outside of the thread block boundaries. It is unknown what the effect of increasing the size of shared memory used would have and to what extent it could be increased without causing too much overhead or reaching memory limitations.

6.2.3 Machine Learning

The models would benefit greatly from more data being collected. To determine how much data is needed for a more confident result would require either a statistical power analysis or doing an analysis with confidence intervals of varying models. An analysis like this could even use a different dataset and mirror some of the setup to determine confidence at varying sample sizes. It was also mentioned that a deeper exploration in the data augmentation methods used could benefit the model. More complex transformations could be beneficial, but also determining to what extent the ultrasound images can be rotated or scaled and still act as a new original sample. Lastly, since results from the comparisons made in Chapter 3 were largely inconclusive, comparisons could be made from model performances with different GLCM parameters being used. Overall, more confident conclusions and more findings could be made with a larger dataset.

Bibliography

[1] C. A. Ancy and Lekha S. Nair. An efficient cad for detection of tumour in mammograms using svm — ieee conference publication — ieee xplore. 2017.

[2] Ali Abbasian Ardakani, Afshin Mohammadi, Fariborz Faeghi, and U. Rajendra Acharya. Performance evaluation of 67 denoising filters in ultrasound images: A systematic comparison analysis. *International Journal of Imaging Systems and Technology*, 33(2):445–464, 2023.

[3] MA Ben Atitallah, Rostom Kachouri, Manel Kammoun, and Hassène Mnif. An efficient implementation of glcm algorithm in fpga. In *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, pages 147–152. IEEE, 2018.

[4] Saeful Bahri and E. Juliastuti. Texture analysis of ultrasound images to differentiate pneumonia and covid-19. In *2021 IEEE International Biomedical Instrumentation and Technology Conference (IBITeC)*, pages 24–28. IEEE, 2021.

[5] C. Balleyguier, L. Ciolovan, S. Ammari, S. Canale, S. Sethom, R. Al Rouhbane, P. Vielh, and C. Dromain. Breast elastography: the technical process and its applications. *Diagnostic and interventional imaging*, 94(5):503–513, 2013.

[6] Saida Sarra Boudouh and Mustapha Bouakkaz. Breast cancer: Using deep transfer learning techniques alexnet convolutional neural network for breast tumor detection in mammography images. pages 1–7, Piscataway, May 08, 2022. IEEE.

[7] Sachintha R. Brandigampala, Abdullah F. Al-Battal, and Truong Q. Nguyen. Data augmentation methods for object detection and segmentation in ultrasound scans: An empirical comparative study. In *2022 IEEE 35th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 288–291. IEEE, 2022.

[8] Aladin Carovac, Fahrudin Smajlovic, and Dzelaludin Junuzovic. Application of ultrasound in medicine. *Acta Informatica Medica*, 19(3):168, 2011. pmid:23408755.

[9] Junghwan Cho, Kyewook Lee, Ellie Shin, Garry Choy, and Synho Do. How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? *arXiv preprint arXiv:1511.06348*, 2015.

[10] Nayeema Chowdhury and Leonard Goldstein. Diagnosis and management of myofascial pain syndrome: Investigating—and eliminating—the underlying causes of active trigger points is key for clinicians to effectively manage patients with myofascial pain syndrome, a common cause of chronic musculoskeletal pain. *Prac Pain Manage*, 12(2), 2012.

[11] D. Ll Cochlin, R. H. Ganatra, and DFR Griffiths. Elastography in the detection of prostatic cancer. *Clinical radiology*, 57(11):1014–1020, 2002.

[12] Daniel M. Cushman, Ziva Petrin, Sarah Eby, Nathan D. Clements, Peter Haight, Brian Snitily, and Masaru Teramoto. Ultrasound evaluation of the patellar tendon and achilles tendon and its association with future pain in distance runners. *The Physician and sportsmedicine*, 49(4):410–419, 2021.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] N. Frulio and Herve Trillaud. Ultrasound elastography in liver. *Diagnostic and interventional imaging*, 94(5):515–534, 2013.

[15] V. Gaike, N. Akhter, K. V. Kale, and P. Deshmukh. Application of higher order glcm features on mammograms. pages 1–3, 2015.

[16] Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 241–246. IEEE, 2016.

[17] Robert M. Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE transactions on systems, man, and cybernetics*, (6):610–621, 1973.

[18] Christopher J. Harvey, James M. Pilcher, Robert J. Eckersley, Martin JK Blomley, and David O. Cosgrove. Advances in ultrasound. *Clinical radiology*, 57(3):157–177, 2002. pmid:11952309.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Qinghua Huang, Yonghao Huang, Yaozhong Luo, Feiniu Yuan, and Xuelong Li. Segmentation of breast ultrasound image with semantic classification of superpixels. *Medical image analysis*, 61:101657, 2020.

[21] Ashfaq Hussain and Ajay Khunteta. Semantic segmentation of brain tumor from mri images and svm classification using glcm features. In *2020 second international conference on inventive research in computing applications (ICIRCA)*, pages 38–43. IEEE, 2020.

[22] Barys Ihnatsenka and André Pierre Boezaart. Ultrasound: Basic understanding and learning the language. *International journal of shoulder surgery*, 4(3):55, 2010. pmid:21472065.

[23] Prathiba Jonnala and G. Sitaramanjaneya Reddy. Image de-noising of ultrasound carotid artery images using various filters. In *2023 4th International Conference for Emerging Technology (INCET)*, pages 1–4. IEEE, 2023.

[24] Justin Ker, Lipo Wang, Jai Rao, and Tchoyoson Lim. Deep learning applications in medical image analysis. *Ieee Access*, 6:9375–9389, 2017.

[25] Hui Liang Khor, Siau-Chuin Liew, and Jasni Mohd Zain. A review on parallel medical image processing on gpu. In *2015 4th International Conference on Software Engineering and Computer Systems (ICSECS)*, pages 45–48. IEEE, 2015.

[26] Dinesh A. Kumbhare, Alyaa H. Elzibak, and Michael D. Noseworthy. Assessment of myofascial trigger points using ultrasound. *American journal of physical medicine rehabilitation*, 95(1):72–80, 2016.

[27] Sarah Leclerc, Erik Smistad, Joao Pedrosa, Andreas Østvik, Frederic Cervenansky, Florian Espinosa, Torvald Espeland, Erik Andreas Rye Berg, Pierre-Marc Jodoin, and Thomas Grenier. Deep learning for segmentation using an open large-scale dataset in 2d echocardiography. *IEEE Transactions on Medical Imaging*, 38(9):2198–2210, 2019.

[28] Tommy Löfstedt, Patrik Brynolfsson, Thomas Asklund, Tufve Nyholm, and Anders Garpebring. Gray-level invariant haralick texture features. *PloS one*, 14(2):e0212110, 2019.

[29] Alexia L. McKnight, Jennifer L. Kugel, Phillip J. Rossman, Armando Manduca, Lynn C. Hartmann, and Richard L. Ehman. Mr elastography of breast cancer: preliminary results. *American Journal of Roentgenology*, 178(6):1411–1417, 2002.

[30] Xueyan Mei, Zelong Liu, Philip M. Robson, Brett Marinelli, Mingqian Huang, Amish Doshi, Adam Jacobi, Chendi Cao, Katherine E. Link, and Thomas Yang. Radimagenet: an open radiologic deep learning research dataset for effective transfer learning. *Radiology: Artificial Intelligence*, 4(5):e210315, 2022.

[31] Nontokozo Mpofu and Michael Sears. Binary view classification of echocardiograms of the heart using transfer learning. In *2020 IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS)*, pages 46–52. IEEE, 2020.

[32] E. A. Nehary, Sreeraman Rajan, and Carlos Rossa. Comparison of covid-19 classification via imagenet-based and radimagenet-based transfer learning models with random frame selection. In *2023 IEEE Sensors Applications Symposium (SAS)*, pages 1–6. IEEE, 2023.

[33] Timo Ojala, Matti Pietikainen, and David Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of 12th international conference on pattern recognition*, volume 1, pages 582–585. IEEE, 1994.

[34] Andreas S. Panayides, Amir Amini, Nenad D. Filipovic, Ashish Sharma, Sotirios A. Tsaftaris, Alistair Young, David Foran, Nhan Do, Spyretta Golemati, and Tahsin Kurc. Ai in medical imaging informatics: current challenges and future directions. *IEEE journal of biomedical and health informatics*, 24(7):1837–1857, 2020.

[35] Asad Parvez and Anuradha C. Phadke. Efficient implementation of glcm based texture feature computation using cuda platform. In *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, pages 296–300. IEEE, 2017.

[36] Kay M. Pepin, Richard L. Ehman, and Kiaran P. McGee. Magnetic resonance elastography (mre) in cancer: Technique, analysis, and applications. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 90:32–48, 2015.

[37] Baidaa Mutasher Rashed and Nirvana Popescu. Machine learning techniques for medical image processing. In *2021 International Conference on e-Health and Bioengineering (EHB)*, pages 1–4. IEEE, 2021.

[38] Ahmad M. Saladin, Licheng Jiao, and Xiangrong Zhang. Gpu-accelerated computation for texture features using opencl framework. In *2014 11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–6. IEEE, 2014.

[39] G. Sanjeevi, Rahul Krishnan Pathinarupothi, G. Uma, and Thushara Madathil. Deep learning pipeline for echocardiogram noise reduction. In *2022 IEEE 7th International conference for Convergence in Technology (I2CT)*, pages 1–6. IEEE, 2022.

[40] Umme Sara, Morium Akter, and Mohammad Shorif Uddin. Image quality assessment through fsim, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 7(3):8–18, 2019.

[41] Hariharan Shankar and Sapna Reddy. Two-and three-dimensional ultrasound imaging to facilitate detection and targeting of taut bands in myofascial pain syndrome. *Pain medicine*, 13(7):971–975, 2012.

[42] Saeed Shurrab and Rehab Duwairi. Self-supervised learning methods and applications in medical imaging analysis: A survey. *PeerJ Computer Science*, 8:e1045, 2022.

[43] S. Sikdar, R. Ortiz, T. Gebreab, L. H. Gerber, and J. P. Shah. Understanding the vascular environment of myofascial trigger points using ultrasonic imaging and computational modeling, -08 2010.

[44] Siddhartha Sikdar, Jay P. Shah, Tadesse Gebreab, Ru-Huey Yen, Elizabeth Gilliams, Jerome Danoff, and Lynn H. Gerber. Novel applications of ultrasound technology to visualize and characterize myofascial trigger points and surrounding soft tissue. *Archives of Physical Medicine and Rehabilitation*, 90(11):1829–1838, 2009.

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[46] Prerna Singh, Ramakrishnan Mukundan, and Rex de Ryke. Feature enhancement in medical ultrasound videos using multifractal and contrast adaptive histogram equalization techniques. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 240–245. IEEE, 2019.

[47] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[48] Alexander A. Tishin and Sergey N. Kuznetsov. Basic principles and methods ultrasound elastography. overview elastography known methods depending on the method of creating strain. In *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 2438–2441. IEEE, 2020.