Tumor detection by Ultrasound powered by AI

BY

Issara Srirojrattanawadee ID: 63011150

Phuwanarit Asaiphon ID: 63011261

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN BIOMEDICAL ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2023

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	Tumor detection by Ultrasound powered by AI
Student Name	Issara Srirojrattanawadee ID: 63011150 Phuwanarit Asaiphon ID: 63011261
Degree	Bachelor of Engineering in Biomedical Engineering
Project Advisor	Signed: _____ (Assoc. Prof. Wibool Piyawattanametha)
Committee	Signed: _____ (Assoc. Prof. Dr. Chuchart Pintavirooj)
Committee	Signed: _____ (Asst. Prof. Dr. Treesukon Treebupachatsakul)
Committee	Signed: _____ (Asst. Prof. Dr. Kasama Srirussamee)
Head of Department	Signed: _____ (Assoc. Prof Dr.Sarinporn Visitsattapongse)

Project Title	Tumor detection by Ultrasound powered by AI
Student Name	Issara Srirojrattanawadee ID: 63011150 Phuwanarit Asaiphon ID: 63011261
Degree	Bachelor of Engineering in Biomedical Engineering
Project Advisor	Assoc. Prof. Wibool Piyawattanametha
Academic Years	2023

ABSTRACT

In 2020, 685,000 people died worldwide from breast cancer, accounting for 2.3 million new diagnoses. Breast cancer is the most common cancer worldwide, with 7.8 million women alive as of the end of 2020 who had received a diagnosis within the previous five years. All across the world, breast cancer affects women at any age after adolescence, however, its prevalence rises with age, which means the faster the diagnos the higher the is the living rate. Breast cancer start from the development of the aberrant breast cells that after will grow and develop into tumors. Tumors have the potential to grow throughout the body and become lethal if ignored. Since tumor detection often requires a large quantity of data, it is both the most difficult and crucial task in many medical-image applications. Tumors come in a variety of forms and sizes. In modern medicine, computer-aided automatic/semiautomatic detection plays a significant role in diagnosis.

The aim of this research is to use artificial intelligence (AI) and B-scan ultrasound to transform tumor identification to the cutting edge of cancer diagnosis, not just breast cancer but other type of cancer. Our system utilizes ultrasound data to enhance tumor detection efficiency and accuracy by interpreting deep learning methods. Our system, the deep learning algorithm can identify small patterns in ultrasound scans that have the possibility of the existence of a tumor. The algorithm has been trained on a large and variety of datasets to enhance the accuracy and to differentiate the types of tumor. A robotic arm with six degrees of freedom (6DOF) is included to increase the project's impact. This arm allows for precise and guided surgical operations based on AI-processed imaging data. The surgical process will be more efficient, diagnostic accuracy will be increased, and eventually, more precise and effective tumor therapies will result from this integration. The synergistic fusion of robotic accuracy, artificial intelligence, and sophisticated imaging paves the way for a revolutionary approach to guided surgery and tumor identification that might fundamentally alter the field of oncological treatments.

ACKNOWLEDGEMENTS

Our project adviser, Assoc. Prof. Dr. Wibool Piyawattanametha, provided guidance and recommendations that helped us complete the project. In addition, everyone with whom we interact and exchange information each week in the optics lab. Finally, the organizers of the project would like to express their gratitude to King Mongkut's Institute of Technology Ladkrabang's School of Engineering and Biomedical Engineering Department for their cooperation in completing this research.

Issara Srirojrattanawadee

Phuwanarit Asaiphon

TABLE OF CONTENTS

Page

LIST OF TABLES

Tables **Page**

LIST OF FIGURES

Figures **Page**

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	Terms
Mets	Metastasis
US	Ultrasound
CT	Computer Termography
MRI	Magnetic Resonance Imaging
PET	Positron Emission Tomography
NIR	Near-Infrared
DNN	Deep Neural Network
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
DoF	6 Degree of Freedom
CPU	Central Processing Unit
GPU	Graphics Processing Unit
ROS	Robot Operating System
RViz	Robot Visualization
c	The speed of sound in the medium.
∇^2	Laplacian operator describing the spatial variations
$p(r,t)$	The sound pressure generated at position r and time
β	The isobaric thermal expansion coefficient
z	The depth
μ_a	Spatial variation absorbance coefficient

CHAPTER 1

INTRODUCTION

1.1 Breast cancer

According to World Health Organization(WHO) in the topic of breast cancer. In 2020, breast cancer caused 685 000 deaths around the world. Women without any particular risk factors other than age and sex account for almost half of all cases of breast cancer. Breast cancer affects people in every nation, worldwide. In contrast men are affected by breast cancer in a range of 0.5–1%. Breast cancer is caused by aberrant breast cells that grow and develop into tumors. Tumors have the potential to grow throughout the body and become lethal if ignored. The milk ducts and the breast's milk-producing lobules are where breast cancer cells first grow. There is no risk to life from the earliest form. Nearby breast tissue can become infected with cancerous cells. Tumors produced by this result in thickening or lumps. Cancers that invade organs or lymph nodes nearby can spread to those areas[1].

Breast cancers are the most generic form of cancer among girls within the United States, constituting 30% of all newly diagnosed lady cancers annually. According to projections via the American Cancer Society for the year 2023, there are anticipated to be 297,790 new cases of invasive breast cancer, fifty five,720 cases of ductal carcinoma in situ, and forty three,700 breast most cancers-related deaths.

The ailment predominantly affects middle-aged and older girls, with a median age of 62. The lifetime threat of growing breast cancer is expected at 13%, and there may be a 7 in 8 risk of no longer experiencing the disease. Incidence costs have proven a consistent increase of 0.5% according to yr when you consider that 1989, even though death prices have proven a steady decline over the equal duration. This decline is attributed to improvements in early detection, heightened focus, and stepped forward treatment modalities[2].

1.2 Objectives

1. Research aim to build a device that aid the doctor in cancer diagnostic.
2. Aid doctors' decisions-making to reduce waiting times and increase the efficiency of treatment for patients.
3. The study aims to assess the effectiveness of optimized ultrasound image analysis in early detection of breast cancer lesions, evaluating its sensitivity and specificity compared to traditional diagnostic methods.

1.3 Scope of the study

Applying and integrating Deep learning model using breast ultrasound image datasets from www.kaggle.com Collected in 2018, it consists of 780 images, average image size 500*500 pixels, from 600 female patients aged between 25 and 75 years[3].

1.4 Report outline

The rest of this report is organized as follows:

- Chapter 2 reviews of theories that are utilize in this research.
- Chapter 3 describes the methodology and the design in each part of the research.
- Chapter 4 demonstrates how the device and program are working with the result of the research.
- Chapter 5 conclusion of the research, explaining the result.

CHAPTER 2

REVIEW OF THEORY RELATED

2.1 Tumor

A tumor is a group of abnormal cells that occur in the body. Tumors can form throughout the body. It can affect bones, skin, tissues, glands and organs. It doesn't always have to be cancer. Many tumors are benign[4].

Types of tumors

- Cancer: Malignant or cancerous tumors can spread to nearby tissues, glands, and other parts of the body. The new tumor is metastasis (Mets).
- Noncancerous: Benign tumors are not cancerous and are rarely life-threatening. They are localized. This means it generally doesn't affect nearby tissues or spread to other parts of the body.
- Pre-cancerous: These non-cancerous tumors can become cancerous if left untreated.

The technique used in tumor detection that is commonly used today is Imaging Techniques

- Ultrasound (US)
- Computed Tomography (CT)
- Magnetic Resonance Imaging (MRI)
- Positron Emission Tomography (PET)

Disadvantages of Using Imaging Techniques

- Ionizing Radiation: Pregnant women and children are more susceptible to radiation-related risks.
- Limited Sensitivity and Specificity.
- Limited Resolution.

2.2 Ultrasound

Ultrasound, also known as ultrasonography, is a medical imaging technique that uses high-frequency sound waves to create real-time images of the inside of the body. It is a non-invasive and safe imaging modality that has a wide range of applications in various medical fields, including obstetrics, cardiology, gastroenterology, and musculoskeletal imaging.

Ultrasound waves over 20,000 Hz in medicine The principle of ultrasound examination or Ultrasonography is to send high-frequency sound waves from the probe (Transdneer). Sound waves affect various tissues. Which has the ability to pass through and reflect is not the same The probe picks up the sound waves reflected at different levels. which indicates the density and tissue depth Take the received signal to process and create a picture[5].

Working Process of ultrasound

- Generation of Sound Waves:

Ultrasound machines generate high-frequency sound waves (ultrasound waves) using a transducer. The transducer consists of a crystal that vibrates when an electric current is applied, producing sound waves.

- Transmission into the Body:

These sound waves are directed into the body and travel through different tissues. The tissues reflect varying amounts of the sound waves based on their density and composition.

- Reflection and Echo Formation:

When the sound waves encounter a boundary between different tissues (e.g., between fluid and soft tissue), some of the waves are reflected back toward the transducer.

- Reception of Echoes:

The transducer then receives these reflected waves, or echoes. The time it takes for the echoes to return is used to calculate the depth of the tissue or organ that reflected the waves.

- Image Formation:

The ultrasound machine processes the received echoes and converts the information into a visual image. The image represents the distribution of different tissue densities within the body.

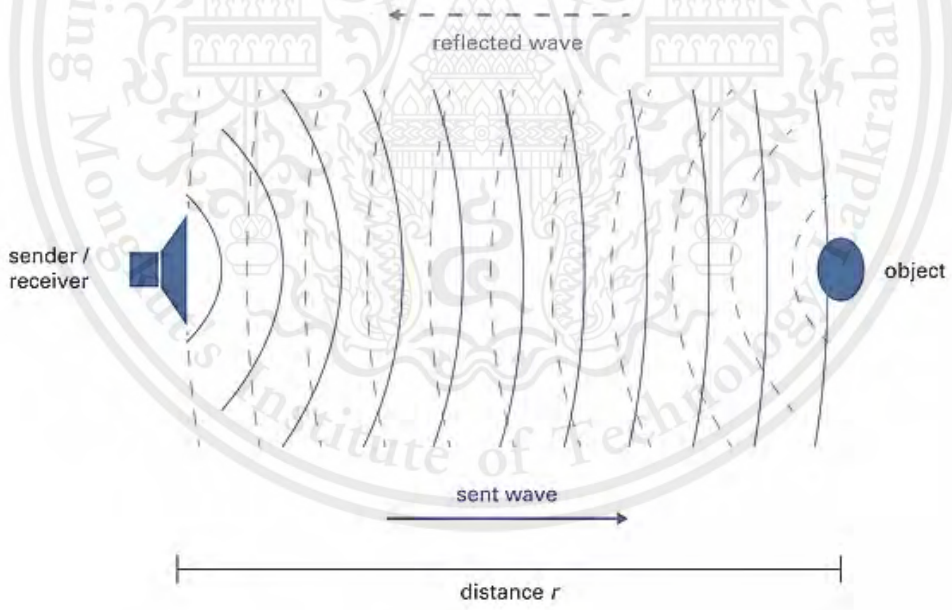

Figure 1. Ultrasound wave Theory[6]

$$c^2 \cdot \nabla^2 p(r,t) - \frac{1}{\beta^2}\frac{\partial^2 p(r,t)}{\partial t^2} = s(r,t)$$

c is the speed of sound in the medium.

∇^2 is a Laplacian operator describing the spatial variations

$p(r,t)$ is the sound pressure generated at position r and time t.

β^2 is isentropic compressibility of the medium

s(r,t) is the source term representing the ultrasound transducer's emitted waveform

Key features of ultrasound

- Non-invasive: No needles or radiation are used, making it a safe procedure for people of all ages, including pregnant women and babies.
- Real-time: The images are produced in real-time, allowing the healthcare professional to see the structures moving and functioning.
- Versatile: Ultrasound can be used to examine a variety of organs and tissues, including the heart, blood vessels, liver, kidneys, uterus, ovaries, and fetus.
- Painless: Most people do not feel any pain during an ultrasound.

Advantages of ultrasound

- Safe: No radiation is used.
- Painless: Most people do not feel any pain during an ultrasound.
- Relatively inexpensive: Ultrasound is a relatively inexpensive imaging technique compared to other modalities such as CT scans and MRIs.
- Portable: Ultrasound machines are portable and can be used in a variety of settings, including hospitals, clinics, and doctors' offices.
- Fast: Ultrasound images can be obtained quickly and easily.

2.3 Image processing

Ultrasound Tumor Detection encompasses several important tiers in photo processing to derive significant statistics from uncooked ultrasound alerts. The method involves Data Acquisition, Preprocessing, Image Reconstruction, Image Enhancement, Image Registration, Quantitative Analysis, Image Visualization, Image Fusion, Artifact Correction, and Post-Processing.

Data Acquisition

The initial step entails taking pictures raw ultrasound indicators generated in the course of ultrasound tumor detection. As the imaging tool scans the goal tissue, these alerts are received over the years and space.

Preprocessing

To beautify sign best, noise reduction strategies are employed. Background noise and electric interference are minimized the use of filtering and signal averaging. Time correction is applied to make sure unique spatial registration of indicators.

Image Enhancement

Techniques like evaluation adjustment, part enhancement, and histogram equalization are hired to improve the readability of structures and enhance evaluation.

Image Registration

Image registration aligns and overlays photographs from numerous modalities or time-lapse imaging for assessment and evaluation.

Quantitative Analysis

Quantitative analysis includes Region of Interest (ROI) analysis, measuring attributes like sign depth or length, and spectroscopic analysis to observe the absorption spectrum for identifying chromophores or assessing tissue traits.

Image Visualization

Images are displayed in various formats, consisting of grayscale, fake color, or parametric maps, to focus on precise facts. Creating three-D renderings or multiplanar perspectives gives a comprehensive perspective.

Image Fusion

Complementary structural and functional records is furnished by using combining ultrasound tumor detection pictures with different modalities like magnetic resonance imaging or computed tomography.

Artifact Correction

Locating and removing artifacts because of factors like motion, auditory litter, or device problems guarantees the accuracy of the imaging.

Post-Processing

Additional evaluation, feature extraction, and facts interpretation operations are carried out on processed images to extract pertinent data or derive quantitative measures in ultrasound tumor detection.

2.4 Deep Neural Network

Deep neural networks (DNNs) represent a machine learning model that mimics the ability of a human to learn from patterns. Known as deep learning, this revolutionary technology allows self-driving cars to recognize and respond to features in their environment. Its remarkable deep learning capabilities have received much attention, and rightly so. This development stands as an unprecedented milestone in the industry, demonstrating the power and impact of this groundbreaking approach[6].

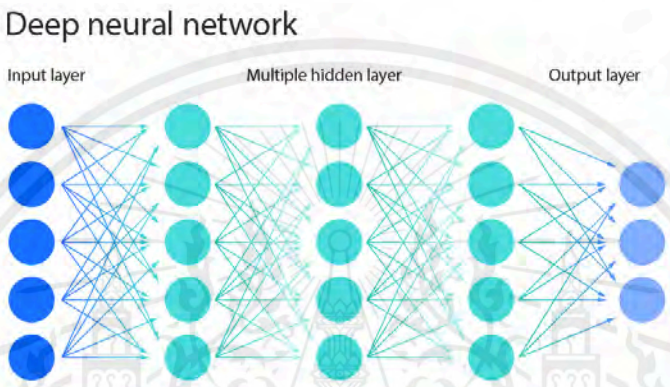

Figure 2. Deep Neural Network [7]

2.4.1 Convolutional Neural Network (CNN)

It is one of the bio-inspired neural networks in which CNN simulates the human vision of space. and combine groups of smaller areas together

When looking at the human subspace Various features will be extracted. of the area, such as lines and color contrast. Humans know whether this area is a straight line or a contrasting color. Because humans look at both the point of interest and the surrounding area at the same time[8,9].

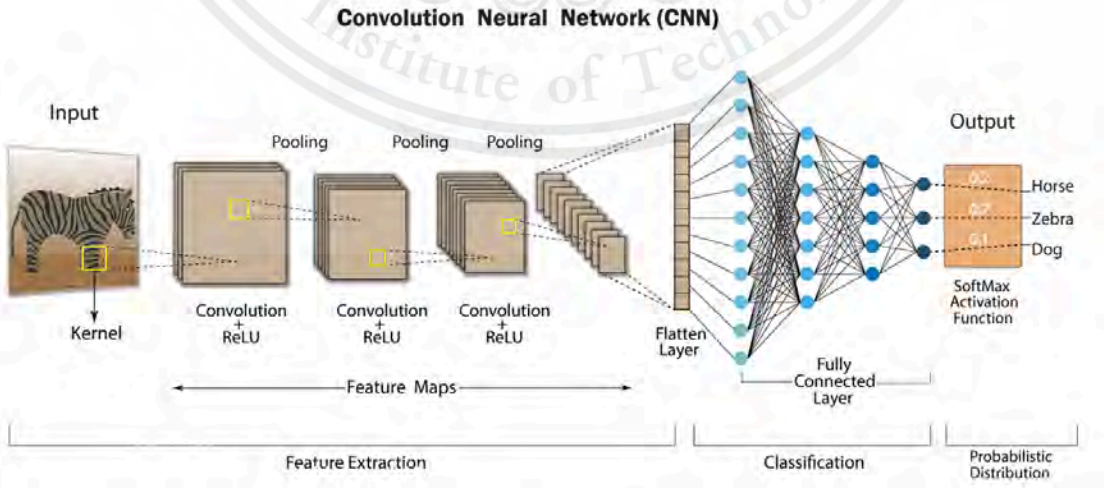

Figure 3. Convolution Neural Network(CNN) [8]

- **Convolution**

Convolution is a mathematical operation for combining information from an input, such as an image, with a convolution filter, often called a kernel. This operation involves applying a filter to the input, performing element-wise matrix multiplication, and stating that the results are combined to form a feature map. A feature map involves features localized to the input, and each point in the feature map corresponds to a specific receiving region in the input. In fact, these convolutions are performed in 3D because images are represented as 3D tensors with height, width, and depth (color channels). Convolution filters also have a 3D structure that corresponds to the depth of the input. Multiple filters are typically applied in a single convolutional layer, each producing a different feature map. These feature maps are assembled using the depth scale to give the final output of the layer. Padding can be used to control the spatial resolution of the output feature maps. This whole process is important for tasks such as image recognition, where CNNs excel in learning and extracting sequences from images[8].

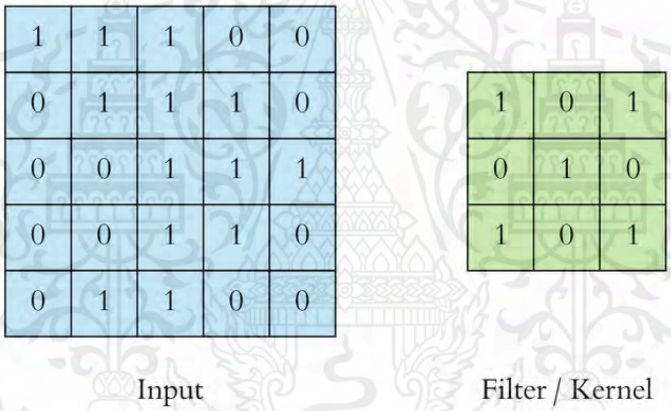

Figure 4. Convolution filter [8]

- **Non-linearity**

Non-linearity stands as a pivotal thing influencing the efficacy and ability of neural networks, and this holds real for Convolutional Neural Networks (CNNs) as well. Within the architecture of CNNs, akin to other styles of neural networks, the infusion of nonlinearity is executed thru the strategic utility of activation features following the convolution operation. Particularly noteworthy is the usage of the Rectified Linear Unit (ReLU) activation feature, a important element in introducing nonlinearity to the community. Post-convolution, wherein the weighted sum of inputs is computed, ReLU replaces bad values with zeros while keeping effective values. This reputedly straight forward step is paramount in instilling a non-linear characteristic into the network. Despite its absence in simplified calculations, it is imperative to grasp that the values in the very last feature maps are a direct result of the ReLU operation carried out to the convolution output. This nuanced non-linearity enhances the community's

capability to determine difficult styles and relationships within the input facts, thereby amplifying its standard gaining knowledge of and choice-making prowess[8].

- **Stride and Padding**

The stride parameter inside convolutional operations is instrumental in determining the volume of movement for the convolution clear out as it slides over the enter. Typically set to one, as illustrated within the supplied context, the default stride allows comprehensive insurance of the enter. However, the adoption of large strides turns into relevant while in search of to decrease the overlap among receptive fields, correctly ensuing in a smaller feature map because of skipped locations for the duration of convolution. In situations in which the upkeep of spatial dimensions in the function map is vital, padding comes into play. Padding involves encircling the enter with zeros or replicating part values, correctly extending the boundaries of the enter. This strategic addition ensures that the convolution filter out can traverse the complete input, permitting the characteristic map to keep the identical dimensionality because the input.Such renovation of spatial statistics proves vital in obligations like image analysis and item detection, wherein preserving spatial records integrity at some point of the network is paramount[8,33].

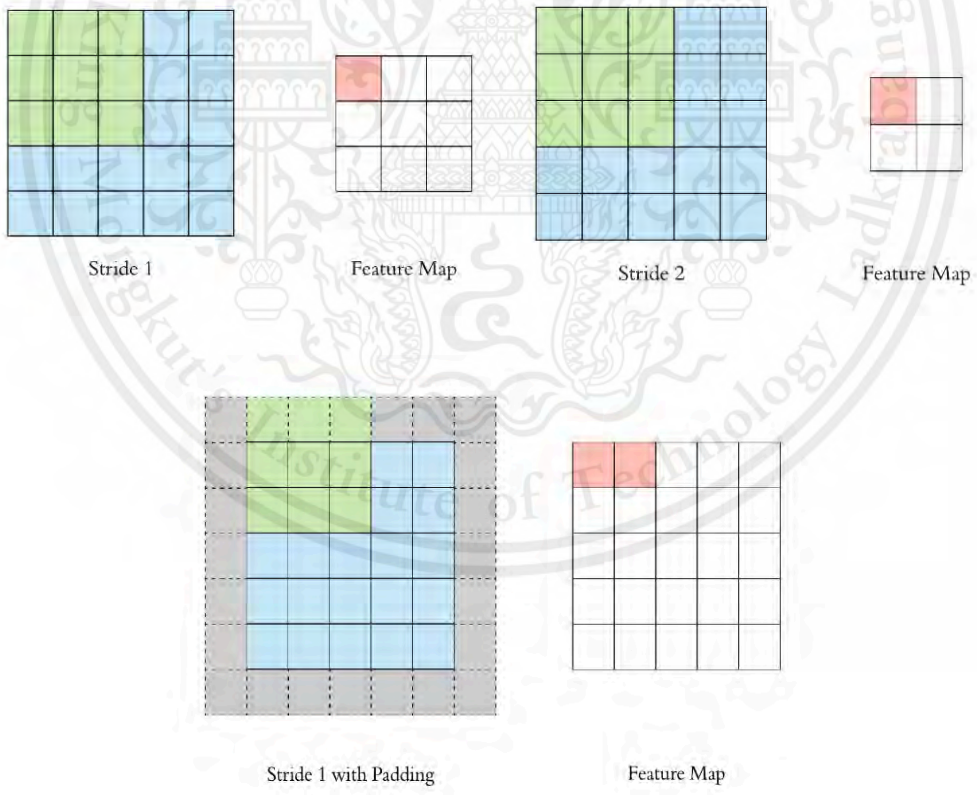

Figure 5. Stride and Padding [8]

- **Pooling**

Pooling layer is an integral part of the design of convolutional neural networks (CNNs) and is used for downsampling of feature maps, and thus gives them a spatial scale commonly referred to as pooling operation, which is often used after convolutional layers in CNN so reduced significantly. The main goal of pooling is to gradually reduce the spatial resolution of the input feature maps, consequently reducing the computational complexity of the mesh One common pooling operation is max pooling, where the layer contains values the maximum from neighboring pixels, and redundant information is discarded. The pooling layer helps focus on the most distinctive features of the network and maintains their essential characteristics. This downsampling is important for controlling computational resources and preventing overfitting. While pooling contributes to semantic flexibility and robust feature extraction, a balance is needed to avoid loss of information. Well-organized pooling layers play an important role in the performance and generalizability of CNNs, leading to their widespread use in tasks such as image recognition and classification[8,10].

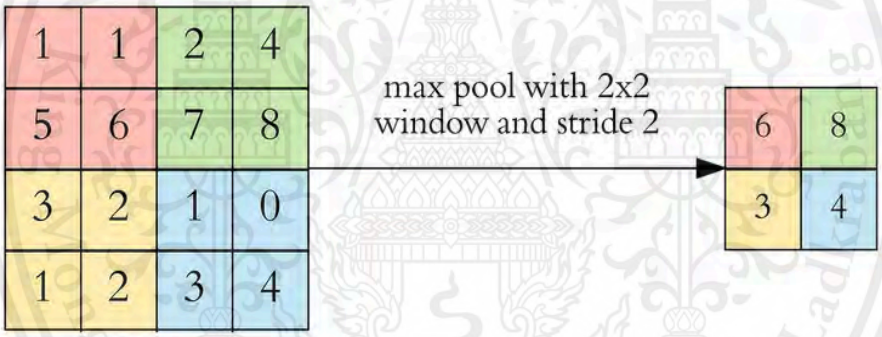

Figure 6. Maxpooling with 2x2 window [8]

- **Hyperparameters**

In a convolutional layer, there are several important hyperparameters that have an effect on the conduct and performance of the network. Firstly, the selection of filter out length is an important selection. While 3x3 filters are commonplace, 5x5 and 7x7 filters are also employed based totally on the particular software requirements. Additionally, there are even 1x1 filters, which can also appear unusual at the start however have unique packages in positive scenarios. It's crucial to be aware that those filters are 3-d, matching the depth of their input, even though this intensity size is often disregarded for simplicity. The second considerable hyperparameter is the filter count, that is highly variable. It commonly levels from powers of , including 32 to 1024. The clear out depend immediately impacts the version's ability, with extra filters permitting the network to capture extra intricate functions. However, an increase in filter out matter additionally escalates the range of model parameters, which can cause overfitting. As a

pleasant exercise, many networks begin with a exceptionally small range of filters inside the preliminary layers and steadily growth the be counted as the network deepens. While the stride hyperparameter exists and can be adjusted, it is often kept at its default fee of 1 to ensure that the convolution clear out slides over the enter with minimum skipping. Finally, the use of padding is a not unusual practice in convolutional layers. Padding entails surrounding the enter with zeros or replicating part values to permit the convolution filter to traverse the input fully. This is essential for keeping the spatial dimensions of function maps for the duration of the community. By using padding, networks can ensure that vital spatial statistics is preserved, that is especially vital for responsibilities like photo analysis and item popularity[11].

- **Fully Connected**

Fully Connected layers, additionally referred to as dense layers, represent a essential factor in neural community architectures, along with Convolutional Neural Networks (CNNs). Unlike convolutional layers specializing in extracting spatial hierarchies, fully linked layers are chargeable for integrating and processing data from the entire enter quantity. In a completely linked layer, each neuron is hooked up to each neuron in the preceding layer, growing a dense internet of connections. This interconnectivity allows these layers to seize complicated relationships and patterns gift inside the discovered capabilities. In the context of CNNs, absolutely related layers regularly follow the convolutional and pooling layers and precede the very last output layer. The neurons in those layers contribute to combining the extracted functions, making high-level abstractions and allowing the network to make state-of-the-art decisions based totally at the learned representations. While necessary to standard neural networks, the inclusion of absolutely connected layers in CNNs contributes to their adaptability in coping with various duties, which include image category and object popularity[12].

- **Training**

Training a Convolutional Neural Network (CNN) follows the equal essential concepts as education a well known Artificial Neural Network (ANN). It relies at the backpropagation algorithm with gradient descent or its versions for updating the model's parameters. However, because of the tricky nature of the convolutional operations concerned in CNNs, the mathematical details of education are greater complicated. The convolutional layers introduce additional issues which include weight sharing, characteristic map dimensions, and the interaction among convolution, pooling, and fully connected layers. While those complexities are crucial for the community's capacity to learn hierarchical functions from facts like pics, they're beyond the scope of this newsletter. If you are interested by delving into the intricate mathematical underpinnings of CNN schooling, you could talk over with more specialised sources dedicated to the problem[13].

2.4.2 Implementation of deep neural networks

- **Conv2D**

The Conv2D layer is an important building block in deep neural network applications, especially in computer vision applications. "Conv2D" stands for two-dimensional convolution, referring to its main function of converting two-dimensional input data, such as an image. This layer runs on convolutional neural networks (CNNs), where it plays a key role in capturing spatial hierarchy and learning features from image input The Conv2D layer uses a series of filters or kernels to input data, . convolving those inputs to find patterns and features. Each filter focuses on identifying specific attributes, and the layer parameters are optimized during training to identify these features. In addition, the layer includes activation functions, usually Rectified Linear Units (ReLU), which introduce nonlinearity into the model. The deep convolutions performed by Conv2D enable the network to learn complexity and spatial relationships among the inputs, making it more efficient for tasks such as image classification, object recognition, and classification.

- **MaxPooling2D**

Layer is liable for developing max-pooling layers, wherein the best required argument is the window size. A not unusual preference is a 2x2 window. By default, the stride duration fits the window length, that's 2 in this situation, and it's far frequently left unchanged.

- **Flatten**

Layer is used after the convolution and pooling layers to transform their output into a 1D vector. This flattened vector is then fed into the fully related layers for added processing, that's normally used for classification tasks. These fundamental layers are the building blocks that allow the design of powerful CNN architectures, mainly well-ideal for responsibilities like photo recognition and function extraction.

- **Dropout**

Dropout has emerged as one of the maximum widely used regularization strategies for deep neural networks, even in ultra-modern models that gain high accuracy. Its primary purpose is to combat overfitting by using a simple but powerful mechanism. During schooling, at each new release, dropout randomly disables or "drops out" neurons with a possibility denoted as p. This manner that for a given new release, a subset of neurons, typically round 50%, can be briefly deactivated along side all their incoming and outgoing connections. The vital point is that the precise set of dropped-out neurons modifications with every training step. This dynamic procedure encourages the network to become less reliant on any character neuron and promotes the independence of all neurons in making predictions. The sudden effectiveness of dropout lies in its capacity to prevent the network from becoming overly dependent on a small

subset of neurons, thereby encouraging the entire network to examine sturdy and numerous representations. It's relatively analogous to spreading information and abilities in a place of work. If one man or woman holds all the know-how, the group becomes depending on them. However, if, through risk, that person is sometimes absent (like within the dropout analogy), others must adapt and study, main to a greater informed and resilient crew. Dropout may be implemented to enter or hidden layers but no longer to output nodes. During testing, after the community is educated, dropout isn't applied; it is exclusively used at some point of education to encourage model generalization. It's really worth noting that dropout is just considered one of numerous regularization strategies utilized in deep learning. Another famous method is batch normalization, which enables stabilize education and boost up convergence. Together with different techniques, dropout contributes to the progressed overall performance and generalization skills of modern deep neural networks[14].

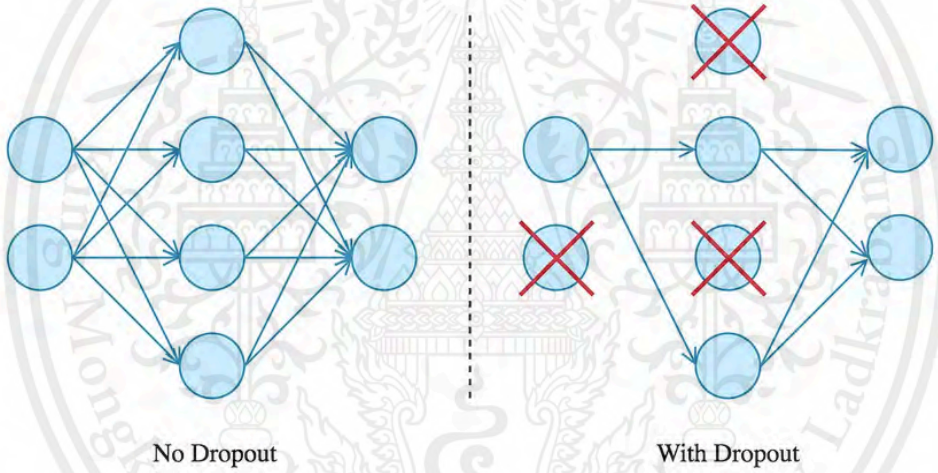

Figure 7. Dropout [11]

- **Model Performance**

Model Performance is an essential aspect of device gaining knowledge of and deep studying, representing the capability of a model to make accurate predictions or classifications on new, unseen information. It encompasses diverse assessment metrics tailor-made to specific tasks, including accuracy, precision, don't forget, F1-Score, AUC-ROC, and more. These metrics provide insights into the model's potential to stabilize factors like correctness, false positives, and fake negatives, relying on the trouble at hand. Model overall performance assessment is critical for gauging the effectiveness of system studying algorithms and their real-global applicability. Continuously tracking and great-tuning model performance are critical steps in ensuring that machine gaining knowledge of models meet their intended objectives and supply dependable and actionable consequences in various domains, from healthcare to finance and beyond[15,16].

- **Data Augmentation**

Data Augmentation is a crucial technique in gadget gaining knowledge of and deep gaining knowledge of, mainly for duties like picture classification and object detection. It involves artificially growing the size of a schooling dataset by way of making use of diverse adjustments to the authentic information. These differences can encompass random rotations, flips, translations, scaling, and shade adjustments, among others. The objective of record augmentation is to introduce variety into the schooling information, which improves the model's capability to generalize to unseen examples and emerge as extra robust to versions inside the input facts.[17]

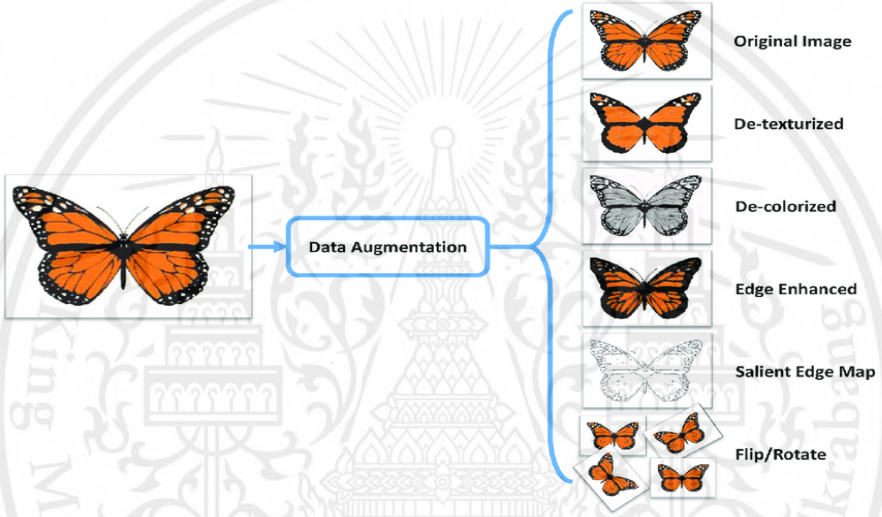

Figure 8. Data Augmentation [17]

- **Updated Model**

The context of device getting to know and deep getting to know refers to a sophisticated or more advantageous version of a previously educated version. This refinement can be carried out via various way, consisting of adjusting hyperparameters, incorporating extra numerous and extensive schooling information, or enforcing advanced architectural changes. The objective of updating a model is to enhance its performance by means of enhancing its capacity to make correct predictions or classifications on new facts. It often involves a rigorous system of version evaluation, fine-tuning, and optimization. By constantly refining and updating models, machine mastering practitioners and researchers purpose to conform to changing facts distributions, enhance robustness, and ensure that the models remain powerful in addressing evolving demanding situations and goals in fields ranging from herbal language processing to laptop imaginative and prescient and past.

- **DICOM**

These DICOM documents generated by using the Logiq Book XP ultrasound machine serve as a valuable aid no longer only for medical practitioners but also for the development of artificial intelligence and deep learning in healthcare. These files can be harnessed to teach AI and deep gaining knowledge of models, enabling the improvement of shrewd systems able to supporting in clinical picture evaluation, ailment detection, and clinical decision-making. By utilizing DICOM documents from ultrasound scans, AI algorithms can learn how to recognize patterns, anomalies, and crucial functions within medical images, ultimately improving diagnostic accuracy, treatment planning, and patient care. This intersection of clinical imaging and AI showcases the capacity for technology to revolutionize healthcare practices and enhance affected person outcomes[18].

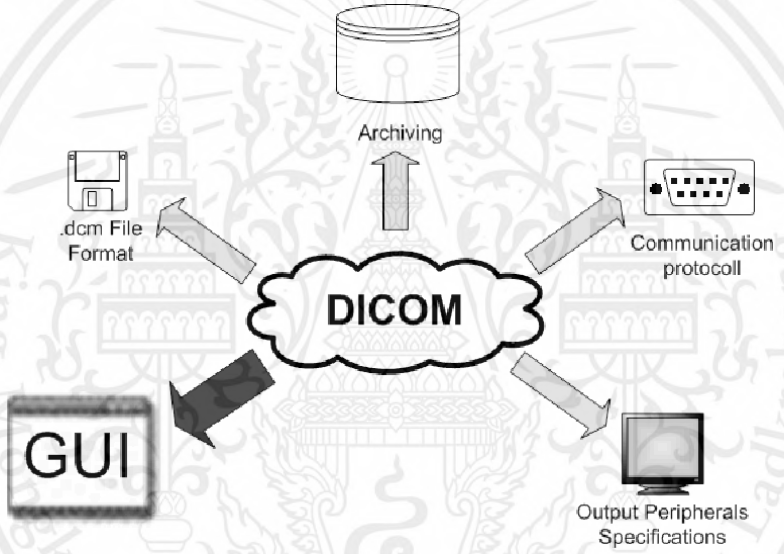

Figure 9. DICOM documents [19]

2.4 6 DoF Robot Arm

6-DoF robot or 6-axis robot, is a type of robotic system that has the ability to move and manipulate objects in six different directions or along six different axes. Each DoF corresponds to a specific type of motion, allowing the robot to achieve a wide range of tasks and positions.

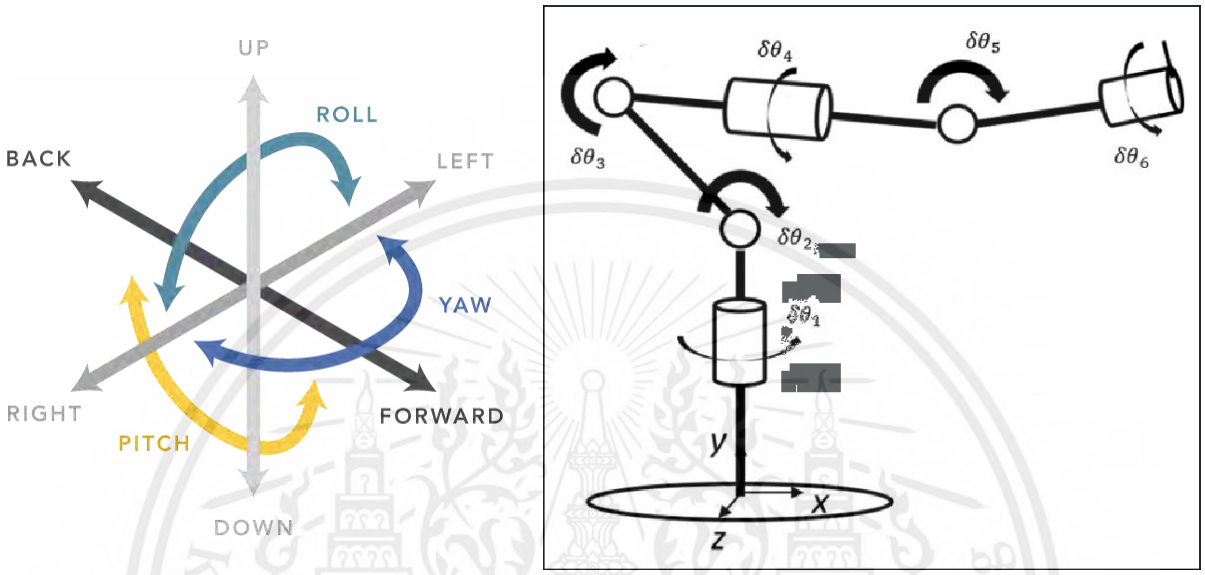

Figure 10. Left, 6 DoF. Right, 6 DoF robot joint. [20,21]

Here's a breakdown of the six DoF:

- Translation along the X-axis: This allows the robot to move back and forth along a linear path in the horizontal direction.
- Translation along the Y-axis: This enables the robot to move left and right along a linear path in the horizontal direction.
- Translation along the Z-axis: This permits the robot to move up and down along a linear path in the vertical direction.
- Rotation around the X-axis: This allows the robot to rotate its end-effector or tool in a pitch motion, similar to nodding the head.
- Rotation around the Y-axis: This enables the robot to rotate its end-effector or tool in a yaw motion, similar to shaking the head.
- Rotation around the Z-axis: This permits the robot to rotate its end-effector or tool in a roll motion, similar to tilting the head from side to side.

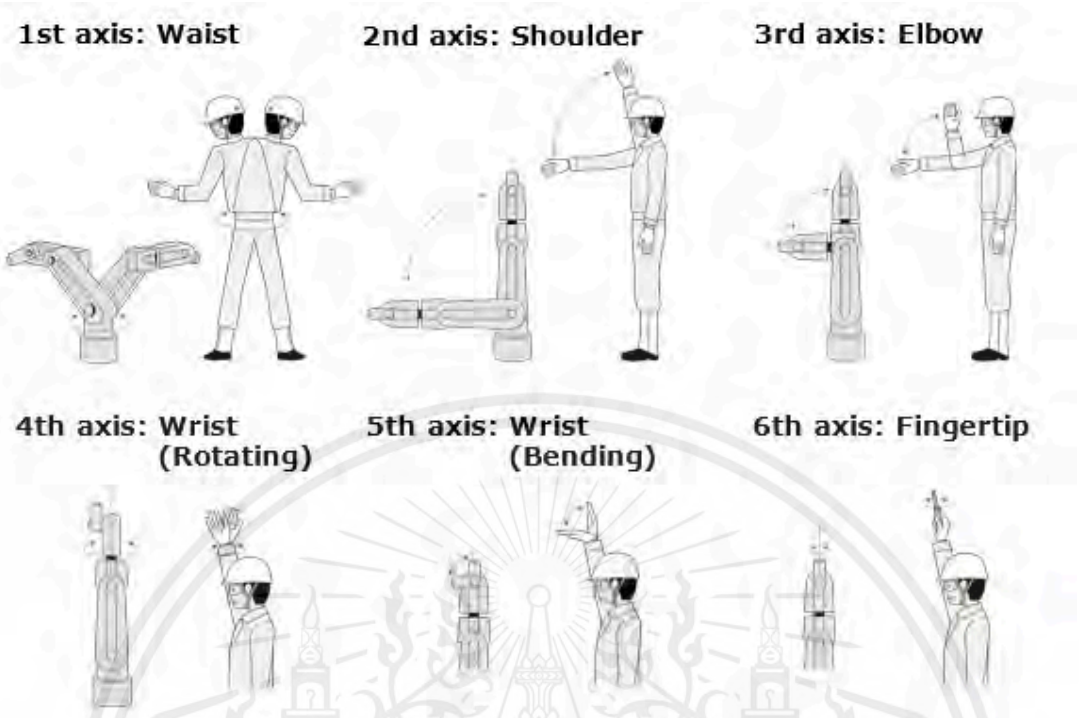

Figure 11. The six DoFs with human represent. [22]

Combining 6 Degrees of Freedom (DoF) robots with ultrasound imaging can enhance guided surgery in medical applications, especially when the aim is to visualize and navigate specific tissues or structures within the body.

- Tissue Visualization: combining a 6-DoF robot with ultrasound imaging, surgeons can obtain real-time, high-resolution images of tissues, blood vessels, and tumors within the body. This enhanced visualization helps identify critical structures and precisely navigate surgical instruments.
- Real-Time Guidance: Ultrasound imaging offers real-time imaging capabilities, allowing surgeons to see and monitor changes in tissue characteristics during the surgical procedure. This enables dynamic navigation and adaptation of the surgical approach based on the evolving tissue conditions.
- Precise Biopsy and Intervention: When performing biopsies or other interventional procedures, the combination of robotics and ultrasound imaging can guide the insertion and positioning of needles or other instruments with high accuracy, reducing the risk of complications.

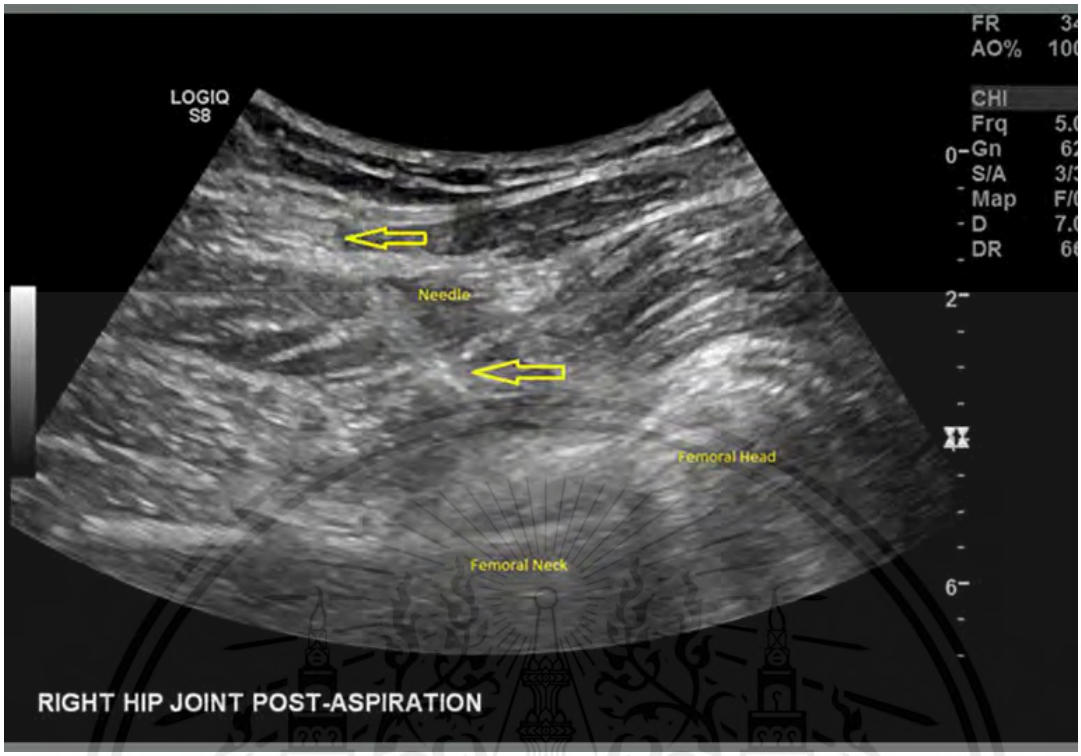

Figure 12. Ultrasound image of the guided surgery [23]

CHAPTER 3
MATERIAL AND METHODOLOGY

3.1 Material

3.1.1 Hardware

Ultrasound system

Using the portable ultrasound system from GE Logiq Book XP.

The GE LogiqBook was GE's first portable ultrasound machine in a laptop format with battery power. This low-cost machine provides high performance and portability with the same power as any mid-tier, full-size device.The Logic Book is best used for endocrinology and OB-GYN but could be used in a variety of applications, including Urological, Vascular, Small parts imaging and more[24].

Logiqbook features and Specificities:

- Color Flow Doppler
- PDI and Directional PDI
- Pulsed Wave Doppler
- M-Mode
- Vascular
- Ob-Gyn
- Small Parts
- Integrated Hard Drive (20GB)
- CD Burner, High Fidelity Speakers
- Highly portable weights only 10 Lbs
- 100-240 International voltage

Figure 13. GE Logiqbook XP Ultrasound [Own image]

Video connection system

A video capturing card, also known as a capture card or video capture device, is a hardware component that allows you to capture and record video and audio signals from external sources, such as cameras, camcorders, or other video playback devices, on a computer. These cards are commonly used in various applications, including video production, live streaming, content creation, and gaming. A video capturing card are used to convert the video output signal form ultrasound to input signal for connecting to computer. The system will recognice the device as a USB camera.

Figure 14. Video capturing card[25]

MyCobot 280 Jetson Nano 6 DOF

Using MyCobot 280 Jetson Nano 6 DOF from ElephantRobotics. The myCobot 280 series of robotic arms are 6-DOF collaborative robots developed by Elephant Robot specifically for research and education, science and technology applications, and commercial exhibitions[26].

MyCobot is the high precision 6 DOF robotic arm which provide a high specification as follow:

Table 1. Speciation of MyCobot 280 Jetson Nano 6 DOF

Payload	250g
Weight	1030g
Working Radius	280mm
Positioning Accuracy	±0.5mm
Working Temperature	-5℃~45℃
Working Lifespan	500h
Power Input	DC 8.4-14V

Figure 15. MyCobot 280 Jetson Nano 6 DOF [26]

MyCobot 280 JN 2023 is built-in with NVIDIA Jetson Nano, providing powerful image processing capabilities. It comes with a standard 2D camera module at the end, making it easy and convenient for eye-in-hand application development.

Figure 16. Jetson Nano developer kit [27]

Table 2. Specification of Jetson Nano developer kit

CPU	Quad-core ARM®A57 @ 1.43 GHz
GPU	128-core NVIDIA Maxwell™
RAM	2 GB 64-bit LPDDR4 25.6 GB/s
Bluetooth	Dual mode Bluetooth 2.4G/5G
Wifi	802.11AC,Dual model WiFi 2.4G/5G

Built-in with the myCobot series universal Ubuntu Mate 20.04 operating system, integrated with multiple development control environments including myBlockly, Python, ROS 1/2, and supports the extension of dozens of end control accessories, greatly expanding the application capabilities of the robot scene.

3D printed part

Using 3D printer to print B-scan probe holder to attach to the robotic arm.

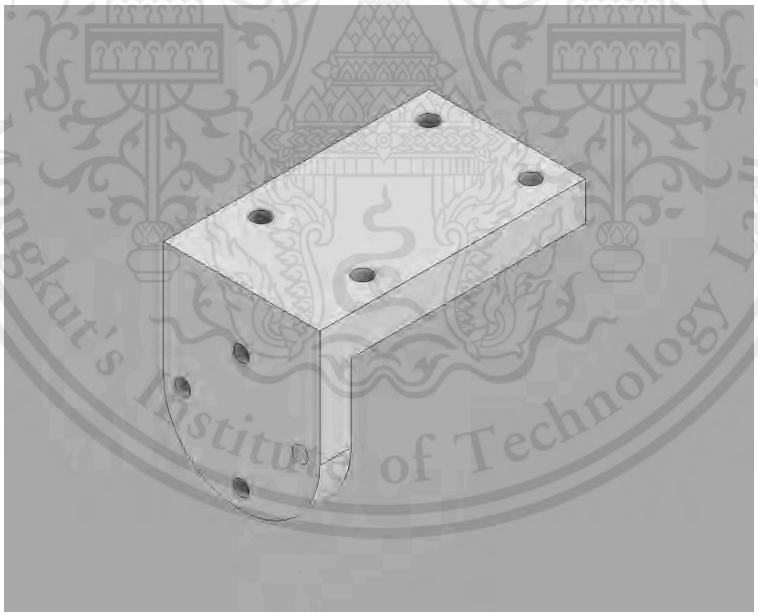

Figure 17. 6DOF Adapter

Note: The image was taken form Autodesk inventor. [Own work]

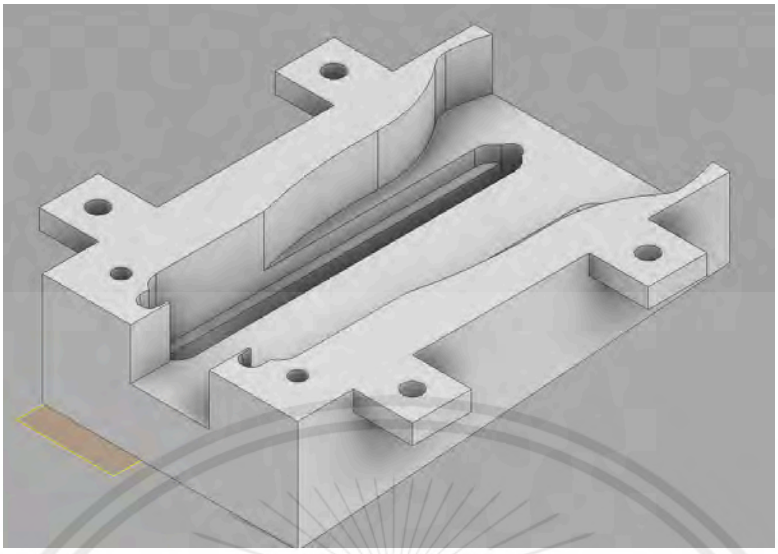

Figure 18. Probe holder

Note: The image was taken form Autodesk inventor. [Own work]

Sample

Using the artificial breast to simulate the diagnostic of breast cancer. The artificial breast are made up of soft silicone which were very similar to the actual breas. The downside with the artificial breast is that the silicone does not have layer of difference density it have the same density of smooth material, result in the ultrasound image will appear as blank space. To simulate the tumor we use hot glue from in a shape like tumor and insert it into the artificial breast. This create the difference in density of the material result in the difference gradient form in the ultrasound image showing the shape of the tumor.

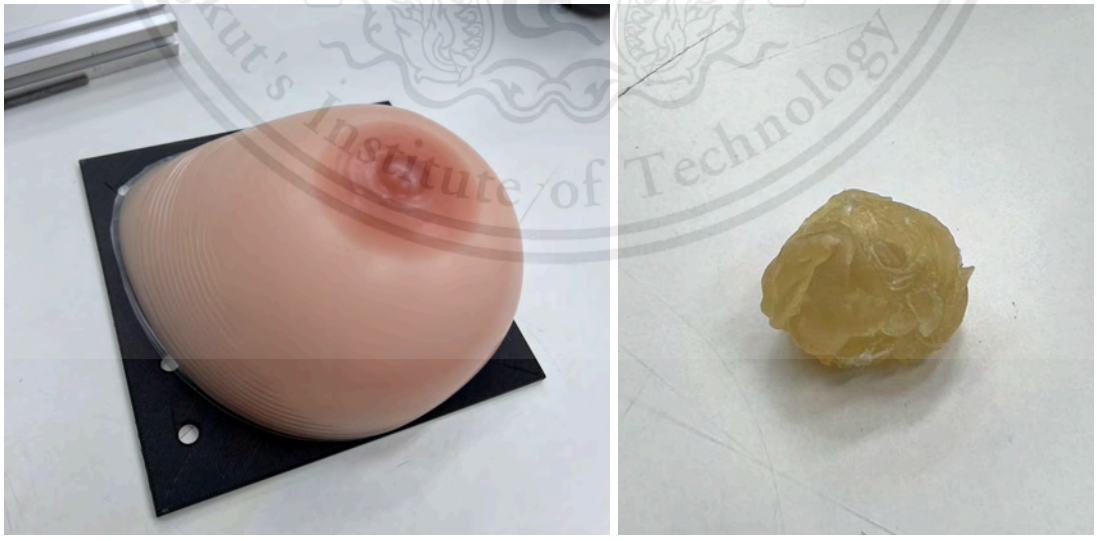

Figure 19. Artificial breast and hot glue ball as a tumor [Own work]

3.1.2 Software

Python

Python, outstanding as an interpreted, item-oriented, and excessive-level programming language with dynamic semantics, boasts a flexible set of attributes that make it a desired preference across diverse programs. Its talent in coping with excessive-level integrated statistics structures, coupled with dynamic typing and dynamic binding, positions it as an extremely appealing option for Rapid Application Development (RAD). Python excels no longer simplest as a strong scripting language however also as a flexible glue language, seamlessly connecting disparate additives. What sets Python aside is its fashionable and without problems graspable syntax, emphasizing code readability, which in flip interprets to decreased program preservation charges. Python's aid for modules and applications fosters program modularity and code reuse, facilitating the creation of scalable and maintainable software program answers. Moreover, Python's interpreter and good sized popular library are accessible without fee, serving all important platforms and enabling free distribution.

Programmers frequently broaden a deep affection for Python, often due to the great increase in productiveness it grants. With no compilation step to sluggish down the improvement cycle, the technique of enhancing, trying out, and debugging Python programs turns into quite fast. Debugging in Python is substantially sincere; errors trigger exceptions in place of catastrophic segmentation faults. The interpreter gives clear insights via stack lines whilst exceptions occur, whilst a built-in supply-degree debugger empowers programmers with capabilities like variable inspection, expression evaluation, breakpoints, and line-by using-line code traversal. Notably, the debugger itself is written in Python, underscoring Python's introspective skills. In practice, a simple but effective approach often entails augmenting the supply code with strategically positioned print statements to expedite debugging, capitalizing on Python's fast edit-take a look at-debug cycle. This flexible and efficient programming language unearths its application in various domains, consisting of person interfaces, in which it is able to be harnessed to control complicated systems like a 6-diploma-of-freedom (6DOF) robotic arm, supplying a combination of robustness and simplicity of development for such complicated obligations[28].

Google Colab

Google Colab is a cloud-based platform furnished by using Google that is especially valuable for schooling AI deep gaining knowledge of models, specifically on complicated information like ultrasound DICOM (Digital Imaging and Communications in Medicine) images. With its loose access to GPUs and TPUs, collaborative features, and no setup necessities, we can leverage Google Colab to efficiently expand and train deep learning algorithms for responsibilities like clinical photo evaluation. This includes duties along with photo segmentation, class, and anomaly detection, which are critical in healthcare and diagnostics. By

combining the electricity of Google Colab's assets with specialized deep gaining knowledge of frameworks like TensorFlow and PyTorch, we will work on contemporary AI tasks, decorate scientific imaging packages, and make contributions to improvements within the subject of healthcare technology[29].

```
+ Code  + Text

# Convert testing data to Numpy arrays
X_test_array = np.array([np.array(Image.open(img).resize((150, 150)).convert("RGB")) for img in X_test])
y_test_array = np.array(y_test)

# Initialize the model
model = Sequential()

# Add convolutional layers
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Add more convolutional layers
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten the output
model.add(Flatten())

# Add dense layers
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Data augmentation using ImageDataGenerator
```

Figure 20. Google colab code example [Own work]

3.1.3 Firmware

Robot Operating System(ROS)

An open-source middleware system called Robot Operating System (ROS) was created specifically for robotics. It provides a networked and adaptable architecture that makes it easier to create robotic software. Originally created by Willow Garage and now maintained by the Open Source Robotics Foundation, ROS offers a publish-subscribe style of data sharing for nodes over a communication infrastructure. Code reusability across various robotic systems is made possible by ROS's support for hardware abstraction and package-based software structure and distribution system. By managing node-to-node communication, middleware—like DDS or XML-RPC—enables modular and scalable robotic systems. A variety of programming,

visualization, and debugging tools and libraries are included in ROS, and the platform's vibrant community encourages cooperation, code sharing, and problem-solving[30].

RViz

Rviz (Robot Visualization) is a powerful and necessary tool inside the discipline of robotics and automation. It serves as a 3-D visualization and configuration platform that permits engineers, researchers, and roboticists to have interaction with and understand complex robotic records and environments successfully. Rviz offers an actual-time, 3-dimensional view of the robotic sensor records, its environment, and its deliberate trajectories, making it a treasured asset for robot improvement, simulation, and debugging. With its consumer-pleasant interface and widespread set of functions, Rviz aids in obligations inclusive of motion planning, sensor statistics evaluation, and gadget testing, facilitating the improvement of strong and efficient robot systems. Whether it's used in studies and improvement or for academic functions, Rviz plays a pivotal role in enhancing the knowledge, evaluation, and nice-tuning of robotic systems and is a key aspect in advancing the abilities of robots across various domains[31].

Moveit

MoveIt is an open-source software tool and library built on the Robot Operating System (ROS) to create motion planning, control, and manipulation for robotic systems. MoveIt is designed to simplify the development of motion planning capabilities for robotic arms and manipulators. It provides a framework that includes motion planning algorithms, collision checking, kinematics, control, and visualization tools[32].

3.2 Methodology

3.2.1 6DoF robotic arm

Mycobot 280 has the ROS Neotic, Rviz, Moveit and many more firmware already installed in the processor. Using the ROS Shell firmware to run the Rviz in order to check each joint of the robot.

The code for running the Rviz, by opening the ROS shell terminal and running this code to open the Rviz program:

```
Unset
# The default serial port name of mycobot 280-JetsonNano version is
"/dev/ttyTHS1", and the baud rate is 1000000.
roslaunch mycobot_280jn slider_control.launch port:=/dev/ttyTHS1
baud:=1000000
```

Rviz and a slider component will be opened, and you will see the following interface:

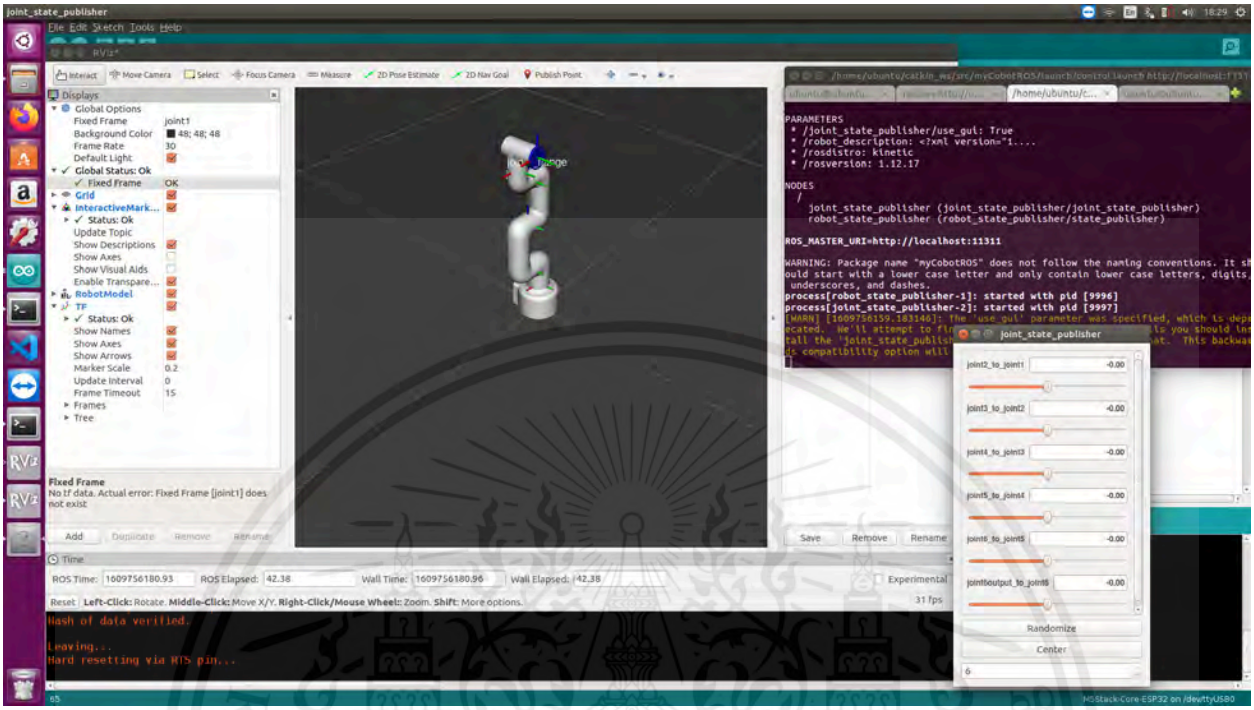

Figure 21. Rviz and a slider component

Note: the Image was taken from the linux operation system in the Invidia Jetson nano that come with the myCobot 280.

Then we can control the model in Rviz to make it move by dragging the slider. To move the real Mycobot with the model, open another command line and run:

```
Unset
# The default serial port name of mycobot 280-JetsonNano version is
"/dev/ttyTHS1", and the baud rate is 1000000.
rosrun mycobot_280jn slider_control.py _port:=/dev/ttyTHS1 _baud:=1000000
```

To run Moveit, Close previous terminal and open the new terminal from ROS Shell and use this code:

```
Unset
roslaunch mycobot_280jn_moveit mycobot_moveit.launch
```

The program will show as follow:

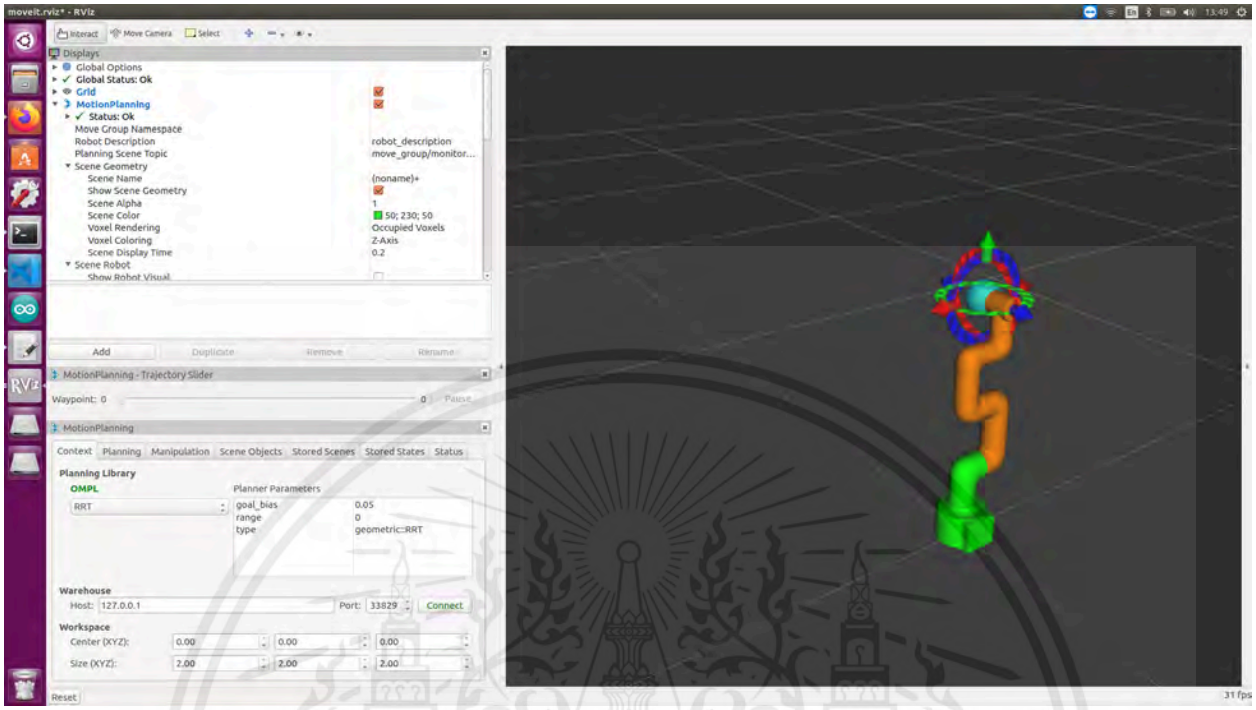

Figure 22. Moveit program

Note: the Image was taken from the linux operation system in the Invidia Jetson nano that come with the myCobot 280.

To move the real Mycobot with the model, open another command line and run:

```
Unset
# The default serial port name of mycobot 280-JetsonNano version is "/dev/ttyTHS1", and the baud rate is 1000000.
rosrun mycobot_280jn_moveit sync_plan.py _port:=/dev/ttyTHS1 _baud:=1000000
```

Using Moveit Setup Assistant to create the post of the model and to plan the action of the robot. To start the MoveIt Setup Assistant:

```
Unset
roslaunch moveit_setup_assistant setup_assistant.launch
```

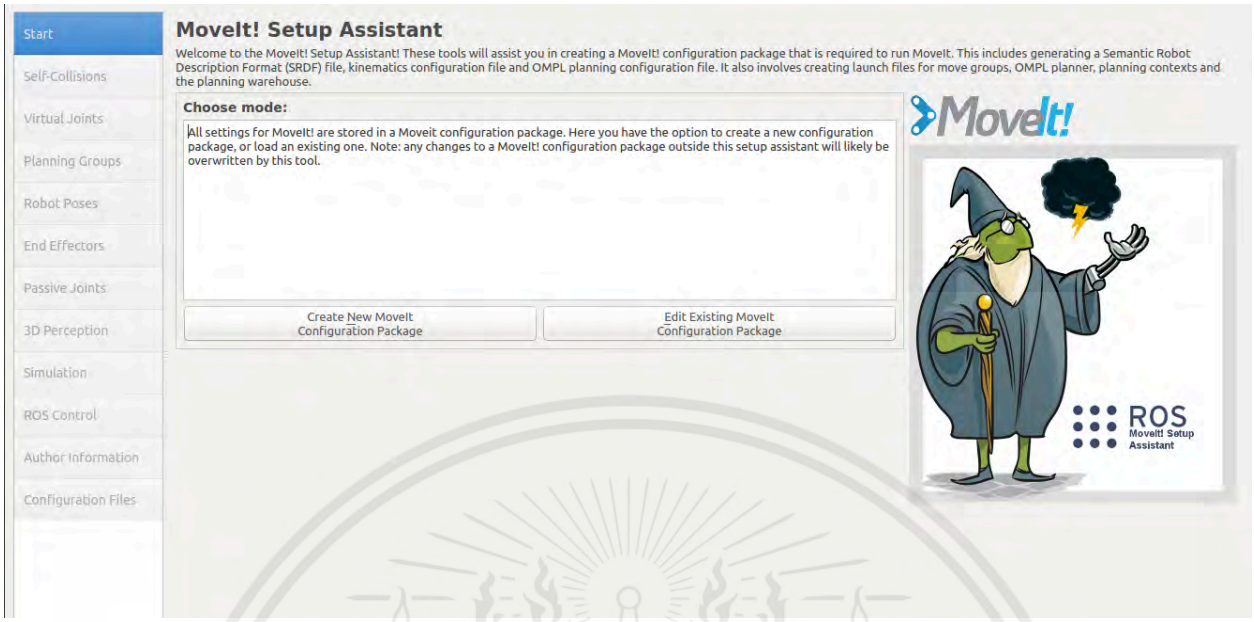

Figure 23. MoveIt Setup Assistant

Note: the Image was taken from the linux operation system in the Invidia Jetson nano that come with the myCobot 280.

The UI of the program will appear, create the new Moveit and follow each step carefully to add joint and creator posts.

3.2.2 Dataset for Ultrasound images

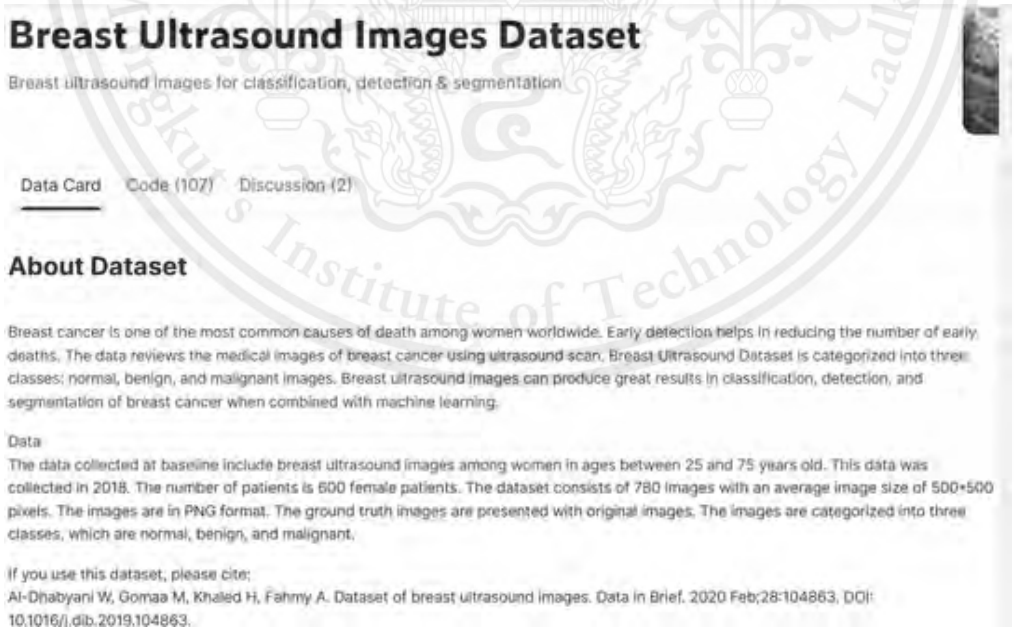

Figure 24. Breast Ultrasoung Image Dataset Note: The image was taken from www.kaggle.com form the topic ultrasound brest canner image dataset.

Breast ultrasound datasets, which are divided into normal, benign, and malignant categories, serve as a valuable resource for classification, detection, and segmentation applications when used in conjunction with machine learning techniques. The dataset was collected in 2018 and contains a total of 780 images obtained from 600 female patients between the ages of 25 and 75. The average image size within the dataset is 500x500 pixels. Images are presented in their original format. It is a comprehensive and versatile suite for medical imaging tasks involving breast ultrasound analysis[3].

Contain: 1578 picture

- Benign class: 891 picture
- Malignant: 421 picture
- Normal: 266 picture

3.2.3 Google colab code [33](For full code see Appendix A.)

Mount Google Drive

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Define the path to the dataset directory in Google Drive
dataset_path = "/content/drive/MyDrive/Test/Dataset_BUSI_with_GT"
```

This code installs your Google Drive on the Colab notebook in the specified path /content/drive. This allows Google Colab to access files stored in Google Drive and sets the path to the directory where your dataset is in Google Drive. Adjust this option based on dataset location.

Get the list of image files and labels

```
# Get the list of image files
image_files = []
labels = []

# Iterate over the directories
for class_name in ['benign', 'malignant', 'normal']:
    class_path = os.path.join(dataset_path, class_name)
    for filename in os.listdir(class_path):
        image_path = os.path.join(class_path, filename)
        image_files.append(image_path)
        labels.append(class_name)
```

This loop returns the three classes ('benign', 'malignant', 'normal') of your dataset, reads the file paths, and adds them to the image_files with their corresponding labels in the labels.

Convert labels to numerical values and one-hot encoded format

```
# Convert class labels to numerical format
label_encoder = LabelEncoder()
y_numeric = label_encoder.fit_transform(labels)

# Convert labels to one-hot encoded format
y_onehot = to_categorical(y_numeric)
```

Using Label Encoding

LabelEncoder is a utility class from scikit-learn that encodes categorical labels with numeric values. In this case, it assigns numerical values to three groups (‘benign’, ‘malignant’, and ‘normal’).
For example, if 'benign' is assigned 0, then 'malignant' should be assigned 1 and 'normal' 2.

One-Hot Encoding

to_categorical is a function from Keras (part of TensorFlow) that converts integer-encoded category labels to warm-encoded format. One hot encoding is a binary matrix representation of labels. Each row corresponds to a label, the column corresponding to the numeric value of the label is set to 1, and all other columns are set to 0 . For example, if 'benign' is encoded as 0, it can have one hot encoding [1, 0, 0]. Similarly, ‘death’ must be [0, 1, 0] and ‘normal’ must be [0, 0, 0].

Split Data into Training and Testing Sets

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(image_files, y_onehot, test_size=0.2, random_state=42)
```

The train_test_split function from scikit-learn is used to split the data set into training and testing sets. image_files: List of image file paths. labels: Matching labels (after label encoding and one-hot encoding). test_size=0.2: Specifies that 20% of the data should be used for testing, and the remaining 80% for training. random_state=42: Provides the result of a random number generator for return.

Convert Image Data to NumPy Arrays

```
# Convert training data to Numpy arrays
X_train_array = np.array([np.array(Image.open(img).resize((150, 150)).convert("RGB")) for img in X_train])
y_train_array = np.array(y_train)

# Convert testing data to Numpy arrays
X_test_array = np.array([np.array(Image.open(img).resize((150, 150)).convert("RGB")) for img in X_test])
y_test_array = np.array(y_test)
```

Image data is loaded, resized to (150, 150) pixels, and converted to RGB format. The resulting arrays are stored as X_train_array, y_train_array, X_test_array, and y_test_array.

Build the Convolutional Neural Network (CNN) Model

```
# Initialize the model
model = Sequential()

# Add convolutional layers
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Add more convolutional layers
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten the output
model.add(Flatten())

# Add dense layers
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
```

The CNN model is described using the Keras Sequential API, which includes convolutional layers, max-pooling layers, and dense layers. Three nodes with softmax activation for multiclass classification on the output layer.

Define and compile the model

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Adaptive Moment Estimation (Adam) is an optimizer that can adjust learning rates for each parameter at a time and also solve the decaying of gradients in each previous step, just like AdaDelta. It also explains the occurrence of the decays of the average of the previous M(t) gradient, as well as the momentum.

Adam is the most popular optimizer because it incorporates the highlights of each Optimizer and removes the weaknesses of both the decaying learning rate of adagrad, allowing the model not to stop learning, as well as being more vulnerable to Gradient Descent and reducing the problem of scaling parameters.

Data Augmentation using ImageDataGenerator

```
# Data augmentation using ImageDataGenerator
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

The ImageDataGenerator is configured using various enhancement techniques, including rotation, shifting, shearing, zooming, and horizontal flipping.

Set up Early Stopping and Model Checkpoint Callbacks

```
# Set up early stopping and model checkpoint callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model_checkpoint = ModelCheckpoint('/content/drive/MyDrive/Test/your_model.h5', save_best_only=True)
```

EarlyStopping is a callback in deep getting to know frameworks that video display units a specific metric at some stage in the training system and forestalls the schooling as soon as a sure situation is met. In this situation: monitor='val_loss': The callback monitors the validation loss for the duration of education. Staying power=5: It waits for 5 consecutive epochs in which the validation loss does no longer improve before triggering the early stopping. Restore_best_weights=True: After early stopping, it restores the model's weights to those that resulted within the excellent performance at the validation set.

ModelCheckpoint is another callback that saves the version's weights in the course of schooling. In this example: '/content material/power/MyDrive/Test/your_model.H5': Specifies the file course where the model weights can be saved. Adjust this path primarily based for your favored area. Save_best_only=True: It saves best the weights that bring about the fine performance at the validation set. If False, it'd keep the weights on the stop of each epoch.

These callbacks are typically used when training neural networks to enhance the training process. The early stopping helps prevent overfitting by stopping training when the model's performance on the validation set stops improving. The model checkpoint saves the best weights, allowing you to later load the model with the best performance on the validation set.

Train the model

```
# Train the model with data augmentation
history = model.fit(
    datagen.flow(X_train_array, y_train_array, batch_size=32),
    steps_per_epoch=len(X_train_array) // 32,
    epochs=20,
    validation_data=(X_test_array, y_test_array),
    callbacks=[early_stopping, model_checkpoint]
)
```

- Batch Size (32): The data are divided into groups of 32 samples.
- Epochs (20): This model has been trained for 20 time
- Validation Split (0.1):10% of the training data is used as validation set during training. This set checks the performance of the model, and helps detect overfitting.

Save the Final Model

```
# Save the final model
model.save('/content/drive/MyDrive/Test/final_model.h5')
```

3.2.4 Ultrasound detection(For full code see Appendix B)

Function for Real-Time White Object Detection

```
# Function for real-time white object detection with adjusted parameters
def detect_white_objects_in_middle(frame, middle_width=215, middle_height=200, width_increase=50, height_increase=50):
    # Convert the frame to grayscale
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Get the frame dimensions
    frame_height, frame_width = frame_gray.shape

    # Calculate the new coordinates for the detection region
    middle_x = frame_width // 2
    middle_y = frame_height // 2
    start_x = max(0, middle_x - middle_width // 2)
    end_x = min(frame_width, middle_x + middle_width // 2 + width_increase)
    start_y = max(0, middle_y - middle_height // 2)
    end_y = min(frame_height, middle_y + middle_height // 2 + height_increase)
```

A function is defined for real-time white object detection within a specified region of interest in the frame. The function takes a frame as input and returns a frame with rectangles drawn around detected white objects.

Camera Initialization and Configuration

```python
camera_index = 1
cap = cv2.VideoCapture(camera_index)

# Check if the camera opened successfully
if not cap.isOpened():
    print(f"Error: Could not open camera with index {camera_index}.")
else:
    while True:
        ret, frame = cap.read()
        if not ret or frame is None:
            print("Error: Could not read frame from the camera.")
```

The Camera Initialization and Configuration aspect of the script is essential for organising a connection with the webcam and configuring its parameters. In Python, the OpenCV library is hired to interface with the digicam through the cv2.VideoCapture magnificence. The user has the ability to pick the camera index, allowing the script to work seamlessly with exclusive cameras, along with built-in webcams or outside gadgets. A vital mistakes-checking mechanism is carried out to handle situations wherein the camera initialization is unsuccessful, offering informative mistakes messages to customers. The isOpened approach verifies whether or not the digital camera is efficiently opened, and if not, an mistakes message is displayed, making sure a robust and consumer-pleasant experience. This initialization and configuration step lays the inspiration for subsequent real-time object detection operations, allowing the script to access and system stay video feeds from the chosen digital camera supply.

Real-Time Object Detection

```python
    # Perform real-time detection
    frame_with_white_objects = detect_white_objects_in_middle(frame, middle_width=225, middle_height=200, width_increase=100,

    # Display the frame with detection or perform other operations
    cv2.imshow("Real-Time Detection", frame_with_white_objects)

    # Capture image when 'p' key is pressed
    key = cv2.waitKey(1) & 0xFF
    if key == ord('p'):
        # Use datetime to generate a unique filename
        timestamp = datetime.now().strftime("%Y%m%d%H%M%S")
        filename = f"captured_image_{timestamp}.png"

        # Save the captured image
        cv2.imwrite(filename, frame)
        print(f"Captured image saved as {filename}")

    # Break the loop if 'q' key is pressed
    elif key == ord('q'):
        break
```

The Real-Time Object Detection script utilizes computer imaginative and prescient techniques for instant identification and visualization of unique items in a stay video feed. Built the usage of the Python programming language and leveraging libraries together with OpenCV and NumPy, the script is designed to work seamlessly with a webcam. The center of the functionality lies inside the detect_white_objects_in_middle feature, which, given a video frame, isolates a defined vicinity of hobby, detects white objects within this area, attracts bounding

rectangles round them, and optionally annotates large items. This permits users to benefit real-time insights into the presence and location of white items in the video.

The script establishes a connection with the webcam, allowing users to pick the camera index for flexibility. Error handling mechanisms are incorporated to gracefully control situations wherein the camera fails to open or frames can not be study. A non-stop loop captures video frames from the webcam, applies the object detection feature, and presentations the processed body in real-time. Moreover, users have the capability to capture and save frames by pressing the 'p' key, and every saved picture is uniquely timestamped for smooth reference. The loop can be exited at any time via urgent the 'q' key, ensuring a person-friendly and interactive enjoy. In summary, the Real-Time Object Detection script gives a practical and handy answer for promptly figuring out and visualizing white objects within a live video feed.

3.2.5 AI Prediction(For full code see Appendix C)

Image Preprocessing Function

```
# Function to preprocess the selected image
def preprocess_image(image_path):
    img = Image.open(image_path).resize((150, 150)).convert("RGB")
    img_array = np.array(img)
    img_array = img_array / 255.0  # Normalize pixel values
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    return img_array
```

The preprocess image characteristic serves a critical role in standardizing the format of a photo selected for prediction in the interactive photograph classification gadget. Upon receiving the file path to the image, the function employs the Python Imaging Library (PIL) to open the picture. Subsequently, it resizes the photo to a consistent size of 150x150 pixels, making sure uniformity in enter dimensions for the neural network model. The conversion to the RGB coloration area is executed to guarantee that the photograph is represented within the 3-channel format expected with the aid of the model. Normalization of pixel values is then executed, scaling them to a variety between zero and 1. This normalization is vital for developing a standardized enter, as neural networks generally perform optimally when working with values inside this normalized variety. Finally, the characteristic expands the picture array and add batch dimension.

Image Upload Function

```
# Function to upload an image from PC to Google Colab
def upload_image():
    uploaded = files.upload()
    if uploaded:
        file_path = list(uploaded.keys())[0]
        display_image(file_path)
        predict_image(file_path)
```

The upload_image function is accountable for allowing customers to upload an image from their neighborhood gadget to the Google Colab surroundings, facilitating real-time predictions the usage of the pre-skilled neural network version. Leveraging the files.Upload() characteristic from the google.Colab library, this characteristic prompts the user to select an photograph file for upload. Once the person uploads an picture, the feature obtains the report route of the uploaded photo, displaying it the use of the display_image characteristic. Additionally, the characteristic triggers the predict_image feature, initiating the version prediction procedure for the uploaded image. This user-pleasant functionality complements the interactive nature of the image class device, permitting users to without difficulty engage with the version and receive predictions on custom pictures of interest.

Image Display Function

```
# Function to display the selected image
def display_image(file_path):
    img = Image.open(file_path)
    img = img.resize((300, 300))
    display(img)
```

The display_image characteristic is designed to showcase the selected photograph within the Google Colab environment. Leveraging the Image class from the PIL (Python Imaging Library) module, the feature opens the photo file exact through the provided report direction. Subsequently, it resizes the photograph to dimensions suitable for show, typically 300x300 pixels, the usage of the resize approach. Finally, the characteristic employs the display characteristic from IPython.Display to showcase the resized image immediately in the Colab pocket book. This visible representation enhances person interplay, permitting them to visually verify the photograph they have got decided on for the subsequent prediction method.

Prediction Function

```
# Function to make a prediction on the selected image
def predict_image(file_path):
    # Preprocess the image
    img_array = preprocess_image(file_path)

    # Make a prediction
    predictions = model.predict(img_array)
    class_names = ['benign', 'malignant', 'normal']
    predicted_class = class_names[np.argmax(predictions)]

    # Display the prediction result
    print(f"Prediction: {predicted_class}")
```

The Predict_image function checks if it makes a prediction on the selected image using the previously trained model. First, it uses the preprocess_image function to prepare the image for prediction by resizing it to 150x150 pixels and converting it to an RGB color space. The processed image is then passed to the pre-trained neural network using a prediction algorithm, which generates a list of probabilities for each class The function determines the class with the highest probability, and returns to the corresponding alphabet meet it ('benign', 'malignant ', or 'normal'). The predicted rectangles are printed, providing the user with an estimate of the details of the selected medical image. This predictive capability is important for research applications, as it allows users to quickly diagnose possible malignancies or other conditions based on the analysis of the AI model.

User Interaction

```
# Create a button to upload images
print("Click the button below to upload an image:")
upload_button = files.upload()
upload_image()
```

The user interface is the front-end of the google colab for AI prediction to create a upload button. The upload button will connect with the file.upload function (open file browser) then run upload_image function to export the selected file from computer to the google colab.

CHAPTER 4
EXPERIMENTAL RESULT AND DISCUSSION

Introduction

In this chapter, we analyze in detail the experimental results obtained from training an AI model for breast cancer detection including three categories: normal, benign, and normal dangerous. The model was implemented in Google Collab using a single onboard GPU and trained on a data set stored in Google Drive. The duration of RAM training is approximately 1367 seconds.

Result

Image classification system

```
Epoch 12/20
36/36 [==============================] - 66s 2s/step - loss: 0.1417 - accuracy: 0.9463 - val_loss: 0.4872 - val_accuracy: 0.8425
Epoch 13/20
36/36 [==============================] - 70s 2s/step - loss: 0.1016 - accuracy: 0.9692 - val_loss: 0.5254 - val_accuracy: 0.8268
Epoch 14/20
36/36 [==============================] - 66s 2s/step - loss: 0.0597 - accuracy: 0.9771 - val_loss: 0.8036 - val_accuracy: 0.8268
Epoch 15/20
36/36 [==============================] - 67s 2s/step - loss: 0.1001 - accuracy: 0.9656 - val_loss: 1.1165 - val_accuracy: 0.7795
Epoch 16/20
36/36 [==============================] - 66s 2s/step - loss: 0.1419 - accuracy: 0.9533 - val_loss: 0.5426 - val_accuracy: 0.8110
Epoch 17/20
36/36 [==============================] - 70s 2s/step - loss: 0.1045 - accuracy: 0.9612 - val_loss: 0.6537 - val_accuracy: 0.8110
Epoch 18/20
36/36 [==============================] - 66s 2s/step - loss: 0.0778 - accuracy: 0.9780 - val_loss: 0.5241 - val_accuracy: 0.8504
Epoch 19/20
36/36 [==============================] - 69s 2s/step - loss: 0.0530 - accuracy: 0.9824 - val_loss: 0.5512 - val_accuracy: 0.8504
Epoch 20/20
36/36 [==============================] - 65s 2s/step - loss: 0.0924 - accuracy: 0.9744 - val_loss: 0.4879 - val_accuracy: 0.8425
```

Figure 25. Result [Own Work]

In Training Progress The model underwent training for 20 epochs, with the following key observations:

Epoch 1-5: The accuracy steadily increased, reaching 77.17% on the validation set by the 5th epoch.

Epoch 6-10: A decrease in training loss and an improvement in validation accuracy to 85.04%.

Epoch 11-15: Further enhancements with training accuracy reaching 96.92% and validation accuracy at 85.83%.

Epoch 16-20: The model continued to improve, achieving a final training accuracy of 98.33% and validation accuracy of 87.40%.

Model Performance The model architecture consisted of convolutional layers followed by max-pooling layers and dense layers. Key metrics from the final epoch include: Training Loss: 0.0730 Training Accuracy: 98.33% Validation Loss: 0.4682 Validation Accuracy: 87.40%

Realtime detecting system

Figure 26 Ultrasound image with realtime tumor detection.

Note: The image is taken from computer link to the Logiq Book XP. Operating with the breast sample.[Own Work.]

Form figure. 26, The system is tested on the artificial sample of silicone breast. It show the result of the ultrasound that have difference density material. The dark spot on the image are surrounded by the white box indicating suspicious of tumor and mark it on the screen by looking at the legion size.

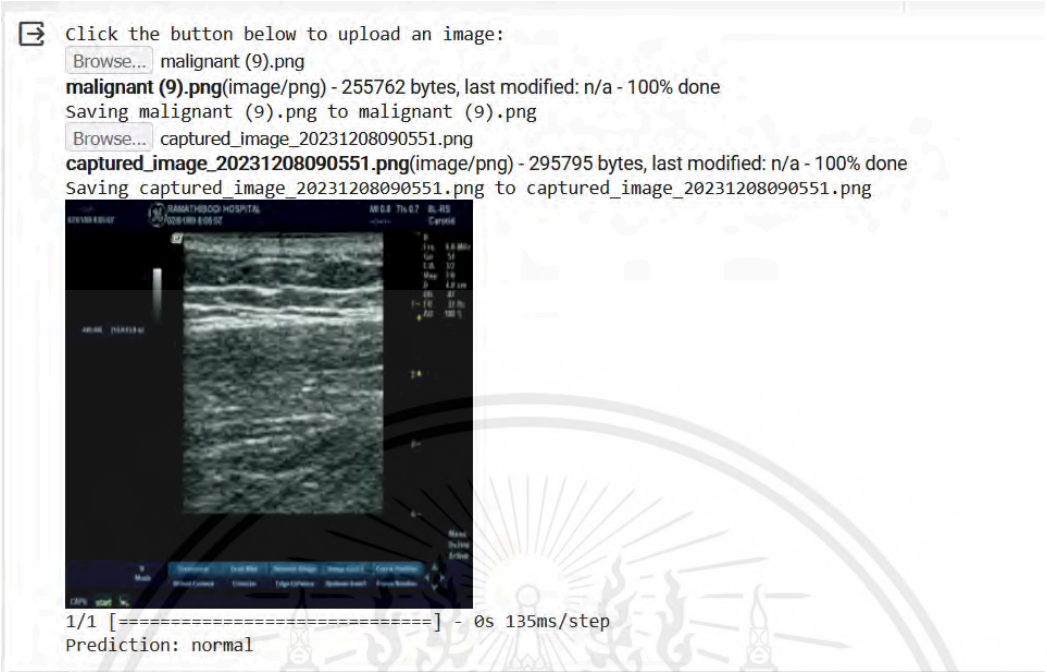

Figure 27. Ultrasound Image classification [Own Work]

After real time detection of the tumor, the image will be capture and upload to an image classification that will classify the class of tumor by testing the model that had been train. From the uploaded image process, the image that not came from the real time tumor detection will not be able to upload because the file type and size are not the same as the model.

Full system setup

In the hardware, we print the probe adapter attached to the 6DoF robotic arm. The probe holders are specifically created for this probe model only, this give benefits in the strong and firm grip of the probe.

Figure 28. Probe case and the 6DOF robotic arm adapter [Own Work]

The full system setup are composed of three screen: an ultrasound, a six-degree robot, and a computer running deep learning software. When all of them are operating simultaneously, we can identify tumors in real time on the computer screen and also operate the robot arm.

Figure. 29 Full system setup

Note: Thw system were setup at the Light Lab, Biomedical Department, King Mongkut' Institude of Technology Ladkrabang. Own Work

CHAPTER 5
CONCLUSION

5.1 Summary

Chapter 1 give the knowledge about Tumors or strange cells that grows within the body, it can affect bones, skin, tissues, glands, and organs. It may be cancerous, benign, or pre-cancerous. In this chapter will introduce to the tumor, types and state of it. Tumor are usually lead to cancer which can cause sevier illness. Imaging strategies such as ultrasound, computed tomography, MRI, and PET are commonly used to detect tumors. However, with the aid of image detection and deep learning AI can helps doctor with diagnostic to be more easier and faster.

Chapter 2 contains all the theories of each topic that get interpred into the project. Giving the knowledge of the biological field as well as the deep learning construction. This will give you the first interaction about the topic, get you to know the principle of Ultrasound. a medical imaging technique that uses high-frequency sound waves to create real-time images of the inside of the body. It is a non-invasive and safe diagnostic tool widely used in various medical fields. By emitting sound waves into the body and detecting their echoes, ultrasound produces images that help assess and diagnose conditions in organs, tissues, and blood vessels. In addition, the knowledge of deep learning process, convolution network, deep neural networrk are some of the AI part that came to interpret with ultrasoung to enhance the diagnostic.

Chapter 3, in this chapter contain the material used and the methodology of the project. We are using GE LogiqBook portable as for now, mycobot 280 6DoF robotic arm, and we are also printing the part to attach the probe to the robotic arm. In the firmware we are using Ubuntu to control the robotic arm. We are using the ultrasound tumor database to train the deep learning model. We using the artificial breast to simulate the diagnostic of the tumor.

Chapter 4 the result is that we have the accuracy of 87.40% acquired from the training of the AI model. The image detection can detect the suspicious area of tumourin real time of scanning. After detected we need to press the button to capture and it will automatically save the image and send to the deep learning algorithms to the process of image classification to classify the class of tumor. The 3D printed parts are attached to the robotics arm to hold the ultrasound probe.

5.2 Discussion

The experimental results show a remarkable class imbalance in the dataset, especially in terms of the number of images per class. The data available for the studies are as follows.

- Benign class: 891 pictures
- Malignant class: 421 pictures
- Normal class: 266 pictures

This difference in the number of observations across the classes may affect the model's ability to generalize and accurately classify instances of each class, especially as the number of images in the Normal class decreases.

Class imbalance

To reduce the impact of class imbalances and increase the accuracy of the model, it is desirable to explore strategies that focus on increasing data sets for unrepresented classes—which take all factors around. Here are ways to address this imbalance:

Data Augmentation: Create new training samples for common classes by applying rotation, flipping, and scaling transformations to existing images. This allows the size of the Normal class data set to be increased artificially.

Dataset Expansion: Get additional real-world models for generic category to complement existing datasets. This may require additional imagery from different sources to ensure a representative and comprehensive dataset.

Possible effects on accuracy

Using these techniques, the data set can be extended for one category, potentially leading to a balanced representation of all groups. This can also improve the model's ability to accurately allocate instances from the Normal class. As a result, the overall accuracy of the model is expected to increase, as it becomes more adept at handling each class in the same manner.

Real-time detection future project

The future improvement to enhance its capabilities by means of implementing real-time detection and incorporating a greater appropriate model structure to offer percent-based predictions. Real-time detection is a vital advancement which could provide immediately insights in the course of ultrasound examinations, enabling healthcare experts to assess tumors dynamically and make quicker selections.

To obtain real-time detection, optimization of the prevailing version structure is important. Transitioning to a version structure this is nicely-ideal for real-time processing which can

substantially enhance the system's responsiveness. This adjustment is vital for ensuring that the model can analyze ultrasound pix in close to real-time, retaining pace with the imaging method all through examinations.

Furthermore, introducing a percentage-primarily based prediction device provides a layer of transparency and interpretability to the consequences. Instead of presenting a discrete class label, the model can output the chance or self belief rating for every magnificence, indicating the proportion reality for benign, malignant, or ordinary conditions. This data can be treasured for healthcare experts to better recognize the model's self belief in its predictions, helping within the choice-making system.

The implementation of actual-time detection and percent-primarily based predictions entails a thoughtful consideration of the computational assets to be had and the trade-off among velocity and accuracy. Continuous refinement of the model structure, doubtlessly exploring actual-time optimized architectures can strike a stability between performance and performance.

Ultimately, these improvements in actual-time detection and end result interpretation can make contributions to the assignment's sensible application in scientific settings, providing a greater dynamic and informative tool for healthcare specialists undertaking ultrasound examinations for tumor detection

REFERENCES

[1] “Breast Cancer”, World Health Organization(WHO), 12 July 2023 [URL:https://www.who.int/news-room/fact-sheets/detail/breast-cancer#:~:text=Scope%20of%20the%20problem,the%20world's%20most%20prevalent%20cancer.]

[2] “Key Statistics for Breast Cancer”, The American Cancer Society medical and editorial contentteam[URL:https://www.cancer.org/cancer/types/breast-cancer/about/how-common-is-breast-cancer.html]

[3] Al-Dhabyani W, Gomaa M, Khaled H, Fahmy A. Dataset of breast ultrasound images. Data in Brief.2020 Feb;28:104863.DOI:10.1016/j.dib.2019.104863. [URL:https://www.kaggle.com/datasets/aryashah2k/breast-ultrasound-images-dataset/data]

[4] Tumor , Cleveland Clinic, [URL https://my.clevelandclinic.org/health/diseases/21881-tumor]

[5] “Ultrasound”, Science Topic, Science Education, National Institude of Biomedical Imaging and Bioengineering, Review on 2023, [URL:https://www.nibib.nih.gov/science-education/science-topics/ultrasound]

[6] “Generating Ultrasound with Piezo Components”, piezo-technology, physikinstrumente.com, [URL:https://www.physikinstrumente.com/en/expertise/technology/piezo-technology/generating-ultrasound-with-piezo-components]

[7] “Neural Network”, International Business Machines(IBM), [URL:https://www.ibm.com/topics/neural-networks]

[8] Swapna K E, “Convolution Neural Network”, developersbreach.com, [URL:https://developersbreach.com/convolution-neural-network-deep-learning/]

[9] Yu Han Liu 2018, J. Phys.: Conf. Ser. 1087 062032,Feature Extraction and Image Recognition with Convolutional Neural Networks[URL: https://iopscience.iop.org/article/10.1088/1742-6596/1087/6/062032/pdf]

[10] A Gentle Introduction to Pooling Layers for Convolutional Neural Networks by Jason Brownlee: [URL: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/]

[11] Samvardhan Singh, Types of neural networks: Convolutional Neural Networks, Medium.com, Review on Dec 24, 2023,

[URL:https://medium.com/@shekhawatsamvardhan/types-of-neural-networks-convolutional-neural-networks-bd973e4fe78c]

[12] Fully Connected Layer by Vaibhav Rastogi Sep 8, 2023 [URL: https://medium.com/@vaibhav1403/fully-connected-layer-f13275337c7c]

[13] Training convolutional neural networks June 27, 2023 by Ole Dreessen [URL: https://www.embedded.com/training-convolutional-neural-networks/]

[14] Rodrigo Polo-Mendoza, Jose Duque, David Mašín, Emilio Turbay & Carlos Acosta (2023) Implementation of deep neural networks and statistical methods to predict the resilient modulus of soils, International Journal of Pavement Engineering, 24:1,DOI: 10.1080/10298436.2023.2257852 [URL: https://www.tandfonline.com/doi/full/10.1080/10298436.2023.2257852]

[15] Modeling exploration strategies to predict student performance within a learning environment and beyond March 2017 DOI:10.1145/3027385.3027422 Conference: the Seventh International Learning Analytics & Knowledge Conference [URL: https://www.researchgate.net/publication/314105851_Modeling_exploration_strategies_to_predict_student_performance_within_a_learning_environment_and_beyond]

[16] A Gentle Introduction to Imbalanced Classification by Jason Brownlee: [URL: https://machinelearningmastery.com/what-is-imbalanced-classification/]

[17] How to Configure Image Data Augmentation in Keras by Jason Brownlee: [URL: https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/]

[18] DICOM (Digital Imaging and Communications in Medicine) By Megan Charles [URL: https://www.techtarget.com/searchhealthit/definition/DICOM-Digital-Imaging-and-Communications-in-Medicine]

[19] Cannella, Vincenzo & Gambino, Orazio & Pirrone, Roberto & Vitabile, Salvatore. (2009). GUI Usability in Medical Imaging. Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2009. 778-782. 10.1109/CISIS.2009.87.

[20] Aiden Samuel, 6 DoF, University of Toronto, posted on march 21st, 2022, [URL: https://robotics.utoronto.ca/history-of-robotics/1974-canadarm/6dof-in-colors-v1/]

[21] Mayyas, Mohammad & Mellish, Rochelle. (2016). A method for the automatic generation of inverse kinematic maps in modular robotic systems. International Journal of Advanced Robotic Systems. 13. 10.1177/1729881416662790.

[22] Kawasaki Robotics, explaining the work of the 6DoF industrial robot. [URL:https://kawasakirobotics.com/blog/the-ins-outs-of-industrial-robot-arms/]

[23] Optical surgery, Mayoclinic.org. [URL:https://www.mayoclinic.org/medical-professionals/orthopedic-surgery/news/success-with-ultrasound-guided-aspirations-of-hip-arthroplasties/mac-20527343]

[24] GE Logiq Book XP Portable Ultrasound Machine, ameultrasounds.com, [URL:https://ameultrasounds.com/products/ge-logiqbook-portable-ultrasound-machine]

[25] Video Capturing card, Amazon shopping website, [URL:https://www.amazon.in/Microware-Capture-Camcorder-Broadcast-Streaming/dp/B0882XY5Q4?th=1]

[26] My cobot 280, Elephante Robotic website, [URL:https://www.elephantrobotics.com/en/mycobot-en/]

[27] Jetson Nano developer kit, Invidia Developer website. [URL:https://developer.nvidia.com/embedded/jetson-nano-developer-kit]

[28] Programming language: Python by Yuseph ABD'ALLAH.I Vertical Fellah Published Sep 16, 2023 [URL:https://www.linkedin.com/pulse/programming-language-python-yuseph-abd-allah-i]

[29] A Comprehensive Guide to Google Colab: Features, Usage, and Best Practices by Abhishek Sharma 23 Jan, 2024 [URL:https://www.analyticsvidhya.com/blog/2020/03/google-colab-machine-learning-deep-learning/]

[30] Robot Operating System(ROS), Ros.org, [URL:http://wiki.ros.org/ROS/Introduction]

[31] Rviz, Ros.org, [URL:https://www.ros.org/]

[32] Moveit, Moveit.ros.org, [URL:https://moveit.ros.org/]

[33] Y. Wang, Z. Xiao, and G. Cao, “A convolutional neural network method based on Adam optimizer with power-exponential learning rate for bearing fault diagnosis,” Journal of

Vibroengineering, Vol. 24, No. 4, pp. 666–678, Jun. 2022, [URL: https://doi.org/10.21595/jve.2022.22271]

APPENDIX A

Google Colab Code

Unset

```
import numpy as np
from PIL import Image
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.utils import to_categorical
import os

# Define the path to the dataset directory
dataset_path = "/content/drive/MyDrive/Test/Dataset_BUSI_with_GT"

# Get the list of image files
image_files = []
labels = []

# Iterate over the directories
for class_name in ['benign', 'malignant', 'normal']:
        class_path = os.path.join(dataset_path, class_name)
        for filename in os.listdir(class_path):
        image_path = os.path.join(class_path, filename)
        image_files.append(image_path)
        labels.append(class_name)

# Convert class labels to numerical format
label_encoder = LabelEncoder()
y_numeric = label_encoder.fit_transform(labels)

# Convert labels to one-hot encoded format
y_onehot = to_categorical(y_numeric)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(image_files, y_onehot,
test_size=0.2, random_state=42)

# Convert training data to Numpy arrays
```

```
X_train_array = np.array([np.array(Image.open(img).resize((150,
150)).convert("RGB")) for img in X_train])
y_train_array = np.array(y_train)

# Convert testing data to Numpy arrays
X_test_array = np.array([np.array(Image.open(img).resize((150,
150)).convert("RGB")) for img in X_test])
y_test_array = np.array(y_test)

# Initialize the model
model = Sequential()

# Add convolutional layers
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Add more convolutional layers
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten the output
model.add(Flatten())

# Add dense layers
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Data augmentation using ImageDataGenerator
datagen = ImageDataGenerator(
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
```

```
        horizontal_flip=True,
        fill_mode='nearest'
)

# Set up early stopping and model checkpoint callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)
model_checkpoint = ModelCheckpoint('/content/drive/MyDrive/Test/your_model.h5',
save_best_only=True)
# Train the model with data augmentation
history = model.fit(
        datagen.flow(X_train_array, y_train_array, batch_size=32),
        steps_per_epoch=len(X_train_array) // 32,
        epochs=20,
        validation_data=(X_test_array, y_test_array),
        callbacks=[early_stopping, model_checkpoint]
)
# Save the final model
model.save('/content/drive/MyDrive/Test/final_model.h5')
```

APPENDIX B

Ultrasound Dection Code

Unset

```
import cv2
import numpy as np
from datetime import datetime  # Import datetime module for unique filenames

# Function for real-time white object detection with adjusted parameters
def detect_white_objects_in_middle(frame, middle_width=215, middle_height=200,
width_increase=50, height_increase=50):
    # Convert the frame to grayscale
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Get the frame dimensions
    frame_height, frame_width = frame_gray.shape

    # Calculate the new coordinates for the detection region
    middle_x = frame_width // 2
    middle_y = frame_height // 2
    start_x = max(0, middle_x - middle_width // 2)
    end_x = min(frame_width, middle_x + middle_width // 2 + width_increase)
    start_y = max(0, middle_y - middle_height // 2)
    end_y = min(frame_height, middle_y + middle_height // 2 + height_increase)

    # Extract the region of interest (ROI) from the grayscale frame
    roi_gray = frame_gray[start_y:end_y, start_x:end_x]

    # Threshold the grayscale image to find white objects
    _, thresh = cv2.threshold(roi_gray, 30, 255, cv2.THRESH_BINARY_INV) # Invert
the threshold

    # Find contours in the inverted thresholded image
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # Draw rectangles around the white objects in the frame
    frame_with_detection = frame.copy()
    for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    x += start_x
    y += start_y

    # Filter out small detection boxes
```

```
        if w > 20 and h > 20:
        # Draw white rectangles
        cv2.rectangle(frame_with_detection, (x, y), (x + w, y + h), (255, 255, 255), 2)

        # Add text if the box is large enough
        if w > 40 and h > 40:
        cv2.putText(frame_with_detection,    "Malignant",    (x,    y    -    10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

        return frame_with_detection

# Try different camera indices (0, 1, 2, etc.)
camera_index = 1
cap = cv2.VideoCapture(camera_index)

# Check if the camera opened successfully
if not cap.isOpened():
        print(f"Error: Could not open camera with index {camera_index}.")
else:
        while True:
        ret, frame = cap.read()
        if not ret or frame is None:
        print("Error: Could not read frame from the camera.")
        break

        # Perform real-time detection
        frame_with_white_objects      =      detect_white_objects_in_middle(frame,
middle_width=225, middle_height=200, width_increase=100, height_increase=50)

        # Display the frame with detection or perform other operations
        cv2.imshow("Real-Time Detection", frame_with_white_objects)

        # Capture image when 'p' key is pressed
        key = cv2.waitKey(1) & 0xFF
        if key == ord('p'):
        # Use datetime to generate a unique filename
        timestamp = datetime.now().strftime("%Y%m%d%H%M%S")
        filename = f"captured_image_{timestamp}.png"

        # Save the captured image
        cv2.imwrite(filename, frame)
        print(f"Captured image saved as {filename}")

        # Break the loop if 'q' key is pressed
        elif key == ord('q'):
```

```
        break

    # Release the camera and close the OpenCV window
    cap.release()
    cv2.destroyAllWindows()
```

APPENDIX C

Prediction By AI

```
Unset
from google.colab import files
from PIL import Image
import numpy as np
import tensorflow as tf
from IPython.display import display

# Load the trained model
model = tf.keras.models.load_model('/content/drive/MyDrive/Test/final_model.h5')

# Function to preprocess the selected image
def preprocess_image(image_path):
	img = Image.open(image_path).resize((150, 150)).convert("RGB")
	img_array = np.array(img)
	img_array = img_array / 255.0 # Normalize pixel values
	img_array = np.expand_dims(img_array, axis=0) # Add batch dimension
	return img_array

# Function to upload an image from PC to Google Colab
def upload_image():
	uploaded = files.upload()
	if uploaded:
	file_path = list(uploaded.keys())[0]
	display_image(file_path)
	predict_image(file_path)

# Function to display the selected image
def display_image(file_path):
	img = Image.open(file_path)
	img = img.resize((300, 300))
	display(img)

# Function to make a prediction on the selected image
def predict_image(file_path):
	# Preprocess the image
	img_array = preprocess_image(file_path)

	# Make a prediction
	predictions = model.predict(img_array)
	class_names = ['benign', 'malignant', 'normal']
```

```
        predicted_class = class_names[np.argmax(predictions)]

        # Display the prediction result
        print(f"Prediction: {predicted_class}")

# Create a button to upload images
print("Click the button below to upload an image:")
upload_button = files.upload()
upload_image()
```