BREAST CANCER DETECTION USING CONVOLUTIONAL NEURAL NETWORKS FOR MRI IMAGES IN TANZANIA A CASE OF MUHIMBILI NATIONAL HOSPITAL

RAMADHANI MRISHO HAMIS

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE (MSc CS) DEPARTMENT OF MATHEMATICS AND INFORMATION AND COMMUNICATION TECHNOLOGY OF THE OPEN UNIVERSITY OF TANZANIA

2024

CERTIFICATION

The undersigned certifies that he has read and here by recommends for acceptance by The Open University of Tanzania a dissertation entitled, Identification of Breast Cancer Using Convolutional Neural Networks. In partial fulfilment of the requirements for the award of Degree of Master of Science in Computer Science (MSc CS).

…………………………….

Dr. Rogers Bhalalusesa

Date …………………………………

COPYRIGHT AND DECLARATION

COPYRIGHT

DECLARATION

I Ramadhani Mrisho Hamis declare that, the work presented in this dissertation is original. It has never been presented to any other University or Institution. Where other people's works have been used, references have been provided. It is in this regard that I declare this work as originally mine. It is hereby presented in partial fulfillment of the requirement for the Degree of Master of Science in Computer Science (MSc CS).

……………………………

Signature

Date …………………………………

DEDICATION

I dedicate this study to my beloved wife Amina Msuya (Ummu Ayman) and my lovely sons (Ayman & Aysar).

ACKNOWLEDGEMENT

I express my gratitude to the all-powerful God, the originator of the cosmos and everything therein, for His guidance throughout my life. His countless acts of grace, compassion, and favor from the time of my birth until this very moment of finalizing this dissertation have consistently provided me with encouragement and fortitude.

I would like to take this opportunity to express my deepest gratitude to the individuals who have contributed immensely to the success of my master's thesis. This research would not have been feasible without the invaluable support, guidance, and expertise they provided.

First and foremost, I would like to express my heartfelt thanks to my supervisor, Dr. Rogers Bhalalusesa, for his unwavering support, helpful guidance, and timely feedback throughout this research. His extensive knowledge and expertise in the field of machine leaning and computer science have been instrumental in shaping the direction of this study. His encouragement and motivation have been a great source of inspiration to me.

Second, I would also like to extend my gratitude to Dr. Lymo, the Head of Radiology at Muhimbili National Hospital, for approving my research proposal and providing the necessary resources to carry out this study. His support has been critical in ensuring the success of this research.

Third, I am grateful to Dr. Agnes Kavishe, Radiologist at Tumbi Hospital, for her valuable insights and inputs in the study. Her expertise in the field of radiology has been immensely helpful in shaping the research questions and identifying the appropriate methodology.

Fourth, I would like to express my appreciation for the guidance and support given by Mr. Quaras, a Radiology Technician, and Dr. Irene, a Radiologist, both working at Muhimbili National

Hospital, in the process of collecting data. Their insights and direction have been vital in guaranteeing the precision and trustworthiness of the gathered data.

Furthermore ,I would also like to thank Mr. Ussule for his support in acquiring permission for data collection. His assistance has been invaluable in obtaining the necessary approvals.

Also, I extend my appreciation to the Ministry of Finance and Planning for providing financial support for my study. Their support has been critical in ensuring that this research is carried out to completion.

Lastly, I would like to acknowledge the management of Eastern Africa Statistical Training Centre (EASTC) for granting me permission to pursue a master's degree in computer science. Their support has been instrumental in providing me with the necessary knowledge and skills to undertake this research.

In conclusion, I am deeply grateful to all the individuals who have contributed to the success of this research. Their support, guidance, and expertise have been invaluable in shaping this study. I am humbled by their generosity and assistance, and I can only hope to repay their kindness in my future endeavors.

ABSTRACT

Breast cancer is a significant global health issue, and early detection is crucial for improving outcomes. However, Tanzania faces challenges in addressing breast cancer, including a lack of locally developed Convolutional Neural Network (CNN) models. This study aims to address this gap by using CNNs with MRI images from Muhimbili National Hospital to identify breast cancer cases.

The research employed an experimental study design using a dataset of 30 MRI images, with 8 malignant and 22 benign cases. To overcome data scarcity and overfitting risks, data augmentation techniques were applied, resulting in an expanded dataset of 1419 images. This augmented dataset provided a stronger foundation for training the CNN model tailored to Tanzania's context.

The CNN model, developed in Python, consisted of multiple layers designed for accurate breast cancer identification. These layers included Conv2D and MaxPooling2D layers for feature capture, Dense layers for classification, and Dropout layers to prevent overfitting. The model achieved an accuracy of 96.4% and an F1 score of 96%, demonstrating its efficacy in identifying breast cancer cases.

Despite the initial dataset's limitations, the research showcases the potential of CNNs and data augmentation techniques for improving breast cancer detection. Further research with larger datasets and diverse populations would be valuable for assessing the model's generalizability. Overall, this study contributes to the field of breast cancer detection by offering an efficient approach for early identification using CNNs for MRI images. Further research and validation using larger datasets and diverse populations would be valuable to assess the generalizability and scalability of the proposed model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANN	:	Artificial Neural Network
AUC	:	Area Under the Curve
BC	:	Breast Cancer
CNN	:	Convolutional Neural Network
Conv2D	:	Convolution layer
DICOM	:	Digital Imaging and Communications in Medicine
HDF5	:	Hierarchical Data Format 5
MaxPooling2D	:	Max pooling layer
MIAS	:	Mammography Image Analysis Society
MNH	:	Muhimbili National Hospital
MRI	:	Magnetic Resonance Imaging
WBCD	:	Wisconsin breast cancer database
WHO	:	World Health Organization
ReLU	:	Rectified Linear Unit
UCI	:	University of California, Irvine

CHAPTER ONE

INTRODUCTION

1.1 Background to the study

Breast cancer is a form of cancer that originates from cells within the breast tissue. It occurs when these cells undergo uncontrolled growth. Typically, breast cancer cells create a tumor that is detectable on an x-ray or through palpation as a lump. The majority of breast cancers commence in the milk-carrying ducts leading to the nipple (ductal cancers), while others may begin in the glands responsible for producing breast milk (lobular cancers),(Al-Haija & Adebanjo, 2020).

Breast cancer is the second most common disease among Tanzanian women in terms of both incidence and mortality, with an estimated 3037 new cases and 1303 deaths in 2018, and is expected to increase by more than 120 percent in terms of both incidence and fatality by 2040(Breast Cancer Initiative, 2017). Based on recent statistical data from the World Health Organization (WHO), approximately 23% of cancer cases and 14% of cancer-related deaths in women are attributed to breast cancer, (Zhao et al., 2018). The timely identification of breast cancer plays a crucial role in influencing the effectiveness of treatment, leading to a substantial reduction in the burden and mortality associated with the disease, (Eroglu et al., 2021).

Breast cancer detection traditionally relied on various methods, including physical examination, mammography, Magnetic Resonance Imaging(MRI) ,ultrasound, and biopsy. These methods have been valuable in diagnosing breast cancer and have saved countless lives. However, they also have limitations such as subjectivity, variability, and the need for expert interpretation,(Lu et al., 2019).

Mammography is the most widely used screening tool for breast cancer detection. It involves taking X-ray images of the breast tissue and analyzing them for abnormalities. While mammography has been effective in detecting breast cancer, it may not be as accurate for women with dense breast tissue or younger women. Ultrasound, on the other hand, utilizes sound waves to create images of the breast and is often used in conjunction with mammography to provide a more comprehensive evaluation.

Magnetic Resonance Imaging (MRI) is a medical imaging method that employs a powerful magnetic field and radio waves to create precise images of the body's internal structures. It offers a non-invasive means to visualize and evaluate different tissues and organs, assisting in the diagnosis and surveillance of a broad spectrum of medical ailments.

In the process of an MRI scan, the patient reclines on a table, which is then positioned inside a sizable cylindrical apparatus. Within this machine, a strong magnet generates a magnetic field around the patient's body. Subsequently, radio waves are sent into the body, prompting the atoms within to emit signals. The MRI machine captures these signals, and with the aid of a computer, creates comprehensive, cross-sectional images that provide detailed visuals of the body's internal structures.

MRI proves especially valuable in visualizing and examining soft tissues like the brain, spinal cord, muscles, joints, and internal organs. It provides high-resolution images that can reveal abnormalities, such as tumors, inflammation, or structural damage. MRI scans are commonly used in various medical specialties, including neurology, orthopedics, cardiology, and oncology, to aid in diagnosis, treatment planning, and monitoring of patients.

MRI is considered a safe procedure without any known adverse effects; however, it may not be appropriate for individuals with specific medical implants or devices that can be affected by magnetic fields. The interpretation of MRI images requires expertise from radiologists or other qualified healthcare professionals who can accurately analyze the images and provide a diagnosis or assessment of the patient's condition,(Yurttakal et al., 2020).

Additionally, Ultrasound and biopsy techniques face several challenges in detecting breast cancer. These challenges include the occurrence of false negatives and false positives, which can lead to misdiagnosis and delayed treatment. The interpretation of ultrasound images and the accuracy of biopsies are operator-dependent, making the expertise and experience of the operator critical. The variability in lesion appearance poses difficulties in accurately distinguishing between benign and malignant lesions based solely on ultrasound imaging. Furthermore, biopsies are invasive procedures associated with potential risks and complications, deterring some patients from undergoing the necessary diagnostic tests. Limited access to resources and the high cost of equipment and specialized personnel further contribute to the challenges in implementing ultrasound and biopsy techniques, particularly in resource-limited settings.

In recent years machine learning techniques have played a significant role in image classification tasks, including breast cancer detection.

Before 2010, traditional approaches to image classification heavily relied on handcrafted feature extraction methods for analysis and classification purposes. These approaches involved manually designing features, such as texture, shape, or intensity, from the images. Subsequently, conventional machine learning algorithms like support vector machines (SVM) or decision trees were utilized to classify the extracted features. However, these approaches heavily relied on domain knowledge and lacked the ability to learn complex patterns directly from raw images.

From 2010 onwards, the rise of deep learning, particularly Convolutional Neural Networks (CNNs), brought about a significant transformation in breast cancer detection. This breakthrough technology enabled the automatic learning of complex patterns and structures from raw image data, leading to remarkable advancements in the field. Unlike traditional approaches, CNNs surpassed expectations by eliminating manual feature engineering requirements, making the detection process more efficient and accurate.

Convolutional Neural Networks (CNNs) are a type of deep learning model that has proven to be remarkably effective in tasks such as image recognition and computer vision, particularly when dealing with visual data. CNNs revolutionized the field by introducing a data-driven approach, allowing automatic feature extraction and end-to-end learning directly from raw images. In the context of breast cancer detection, CNNs have been extensively applied due to their capability to learn hierarchical representations from medical images. Researchers have developed various CNN models for different aspects of breast cancer detection, including distinguishing between benign and malignant lesions, identifying specific features like microcalcifications or masses, and assessing the risk level. Compared to traditional feature-based methods, CNNs have demonstrated significantly superior performance,(Zhao et al., 2018).

A convolutional neural network (CNN) consists of several interconnected layers, including convolutional, pooling, and fully connected layers. The convolutional layers play a crucial role in extracting pertinent features from the input data using a process called convolution. During convolution, a set of learnable filters or kernels is applied to the input image, performing element-wise multiplication and summation to generate feature maps. These feature maps accentuate significant patterns or characteristics in the input image, such as edges, textures, or shapes.

Pooling layers follow the convolutional layers and serve to down sample the feature maps, reducing their spatial dimensionality while retaining the most salient information. This helps in reducing the computational complexity of the model and making it more robust to variations in the input.

The dense layers, also referred to as fully connected layers, have the role of performing the ultimate classification or regression tasks. They receive the features extracted by the convolutional and pooling layers and are trained to associate them with the desired output classes or values. These layers enable the model to grasp intricate relationships within the data and make predictions based on the acquired features.

Convolutional neural networks (CNNs) play a crucial role in breast cancer identification, utilizing labeled data to adjust network weights iteratively through backpropagation. This process, facilitated by optimization algorithms like stochastic gradient descent (SGD), aims to minimize the difference between predicted and actual outputs. CNNs, due to their hierarchical and localized architecture, excel in analyzing visual data. Automatic learning and extraction of relevant features from raw images empower CNNs to perform tasks such as image classification, object detection, image segmentation, and more (Lecun et al., 2015).

The significance of employing CNNs in breast cancer identification lies in their ability to recognize patterns and features within medical images. In this context, the research problem gains relevance as existing models often rely on secondary datasets from foreign sources, lacking the specificity required for local healthcare systems. CNNs, with their capacity to adapt to distinct data distributions and patient demographics, become indispensable tools in developing accurate breast cancer prediction models tailored to the unique context of the target population. Consequently, this research contributes to addressing the gap in locally trained breast cancer prediction models, emphasizing the importance of leveraging CNNs for enhanced diagnostic capabilities.,(Lecun et al., 2015).

1.2 Problem Statement

Various studies have tackled the classification of breast tumors as benign or malignant, yet a significant challenge persists: the absence of a breast cancer prediction model trained on local datasets. Existing research often relies on secondary datasets from sources like the UCI machine learning repository, primarily featuring data from foreign hospitals. This reliance on foreign data poses a notable limitation, especially when implementing models within our country's healthcare system, exacerbated by the disparities in data distribution and patient demographics between foreign and local contexts, (Alanazi et al., 2021).

To bridge this gap, this study aims to develop a breast cancer prediction model tailored specifically to our local datasets. By leveraging locally sourced data, we aim to mitigate the risk of misclassification and enhance the model's relevance to our population's characteristics, ultimately improving the accuracy and effectiveness of breast cancer detection within our healthcare system. Furthermore, we employ data augmentation techniques to bolster our efforts, expanding the size and diversity of our dataset. Through the generation of synthetic data points, we enhance the model's robustness and its ability to generalize across different cases, addressing the challenge of data scarcity and boosting the accuracy and reliability of breast cancer identification within our community, (Shorten et al., 2019).

1.3 General Objective

This study aims to develop a model that identifies breast cancer tumors using convolution neural networks.

1.4 Specific Objectives

i. To extract features in magnetic resonance images (MRI) that are used in detecting breast cancer using CNN.

ii. To develop a Python based CNN model which classifies between benign and malignant breast tissues of our local breast images using a convolutional neural network algorithm.

iii. To evaluate the performance of the developed convolutional neural network (CNN) model by employing performance metrics such as accuracy and F1 score.

1.5 Research Questions

i. How to extract features in magnetic resonance images (MRI) used to detect breast cancer using CNN?

ii. How can a Python-based Convolutional Neural Network (CNN) model be developed to accurately classify between benign and malignant breast tissues in local breast images, employing a convolutional neural network algorithm?

iii. What is the performance of the developed convolutional neural network (CNN) model, as assessed through performance metrics such as accuracy and F1 score?

1.6 Significance of the Research

Early detection of breast cancer leads to effective treatment outcomes. Therefore, the design of a model that can classify benign and malignant tumors in breast cancer would be valuable in the academic world. This is because the study has been conducted using datasets (MRI images) from our local hospital, Muhimbili National Hospital.

Moreover, the model developed in this research holds the promise of enhancing the accuracy of tumor classification (malignant and benign) for pathologists. This was achieved by using data specifically from our local hospital, Muhimbili National Hospital (MNH).

Additionally, this research aims to improve breast cancer detection in Tanzania by using Convolutional Neural Network (CNN) models on local MRI breast images. The study used an experimental design with 30 MRI images, with 8 malignant cases and 22 benign ones. Data augmentation techniques were applied to expand the dataset to 1419 images, providing a more robust foundation for the CNN model. The model, developed in Python, had multiple layers for accurate identification, achieving an accuracy of 96.4% and an F1 score of 96%. The model demonstrated robust feature extraction and classification capabilities, indicating its reliability and potential for clinical application. Further research and validation using larger datasets and diverse populations would be beneficial. This research contributes to the field of breast cancer detection by offering an efficient approach for early identification using CNNs.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

In this chapter, literature review of the study is discussed. Firstly, the concept of breast cancer and its early symptoms is presented. Secondly, Convolutional Neural networks algorithms in breast cancer identification is discussed. Finally, the conceptual framework illustration and knowledge gap has been presented.

2.2 Breast cancer

Breast cancer is a form of cancer that originates in the breast cells. It develops when irregular cells within the breast undergo uncontrolled growth and division, giving rise to a tumor. These cancerous cells have the potential to infiltrate neighboring tissues and metastasize to distant parts of the body through the bloodstream or lymphatic system as shown in Figure 2.1.

Breast cancer is commonly identified by the existence of a new lump or growth, but it's crucial to understand that most breast lumps are not cancerous. These growths can be categorized as either benign or malignant. Benign tumors are non-cancerous, whereas malignant tumors are cancerous. Moreover, the main differentiation between benign and malignant tumors lies in their shape: benign tumors tend to have a round or oval shape, while malignant tumors display a somewhat rounded shape with an irregular boundary. Furthermore, malignant masses appear brighter in colour compared to the surrounding tissue (Ragab et al., 2019).

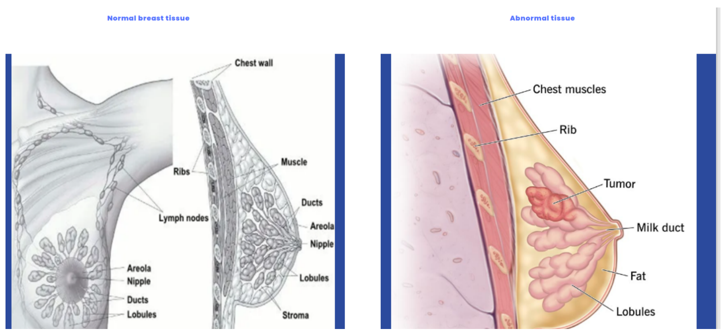


Figure 2.1 Normal breast tissue and abnormal breast tissue (American Cancer Society, 2014)

2.3 Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) belong to the class of deep learning neural networks. In essence, CNN is a machine learning algorithm capable of taking an input image and assigning importance (through learnable weights and biases) to various aspects or objects within the image, allowing it to distinguish between them. CNN achieves this by extracting relevant features from the images. A typical CNN structure comprises several components, including the input layer, which handles grayscale images, and the output layer, responsible for binary or multi-class labels. It also incorporates hidden layers, which consist of convolutional layers, ReLU (rectified linear unit) layers, pooling layers, and a fully connected Neural Network as shown in Figure 2.2, (Ragab et al., 2019).

Convolutional Layer: The convolutional layer applies a set of learnable filters (also known as kernels) to the input image. Each filter convolves over the image, performing element-wise multiplications and summing the results to produce a feature map. This layer helps in detecting local patterns and features by capturing spatial information in the image.

Activation Layer: The activation layer brings non-linearity to the CNN by applying an activation function, such as the Rectified Linear Unit (ReLU), to the feature map derived from the convolutional layer. This introduction of non-linearity allows the model to capture complex relationships between features and enhance its ability to handle more intricate patterns in the data.

Pooling Layer: The pooling layer reduces the spatial dimensions of the feature maps through down sampling. This down sampling is beneficial in controlling computational complexity and mitigating overfitting. The widely used technique in pooling is max pooling, wherein a defined window selects the maximum value as the representative value.

Fully Connected Layer: The role of the fully connected layer is to generate predictions based on the extracted features. It takes the output from the previous layers and applies matrix multiplication with learnable weights. This layer enables the model to learn complex combinations of features and make high-level predictions.

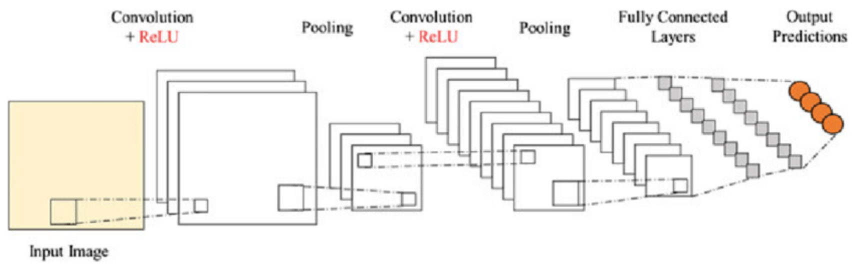


Figure 2.2: The typical CNN architecture(Al-Zuhairi et el,2019)

Convolutional Neural Networks (CNNs) have showed to be highly effective and outperform other classification algorithms. The unique architecture of CNNs makes them particularly suitable for processing images and extracting relevant features.

CNNs are preferred over other classification algorithms in different image classification tasks due to the following; -

Convolutional Neural Networks (CNNs) excel at capturing spatial relationships: In contrast to conventional machine learning algorithms such as Random Forest and Support Vector Machine (SVM), CNNs take advantage of the spatial relationships present in images. They use convolutional layers that apply filters to small portions of the image, allowing them to detect local patterns and features. This capability is crucial for accurately classifying objects in images.

In a study conducted by Krizhevsky et al. (2012), CNNs demonstrated exceptional performance in image classification tasks. The researchers trained a deep CNN architecture called AlexNet on the ImageNet dataset, which consists of millions of labeled images from various categories. The AlexNet achieved a significant reduction in error rate, surpassing traditional machine learning algorithms. This study solidified the dominance of CNNs in image classification tasks (Krizhevsky et al., 2012).

Moreover, CNNs possess the capability to automatically learn hierarchical features. This is a significant advantage, as they can extract low-level features like edges, textures, and shapes through multiple layers of convolution and pooling operations. These extracted low-level features are then combined to create higher-level features that are more discriminative and useful for accurate classification.

The hierarchical and localized nature of CNNs makes them particularly well-suited for analyzing visual data. By automatically learning and extracting pertinent features from raw images, CNNs gain the ability to undertake tasks like image classification, object detection, image segmentation, and other related functions,(Lecun et al., 2015).

Furthermore, CNNs can leverage data augmentation techniques to enhance their performance. Data augmentation entails the application of random transformations, such as rotations, translations, and flips, to the training images. By augmenting the training data, CNNs become more resilient to variations in the input images, resulting in improved generalization and better classification accuracy.

2.4 Magnetic Resonance Imaging (MRI)

Magnetic Resonance Imaging (MRI) is a medical imaging method that employs a powerful magnetic field and radio waves to create detailed images of the body's internal structures. It is extensively utilized in clinical settings to diagnose and monitor various medical conditions, such as brain disorders, musculoskeletal injuries, and breast cancer.(Khooa et al., 1997).

2.5 Data augmentation

Data augmentation is a deep learning technique that involves applying various transformations to the existing training dataset to artificially increase its size and diversity. The primary objective of data augmentation is to enhance the model's generalization ability, robustness, and overall performance by exposing it to a broader range of variations and patterns present in the training data, (Wang & Perez, n.d.).

The most commonly augmentations techniques used are: -

Image flipping and rotation: Images can be horizontally or vertically flipped to create new variations of the original image. Additionally, rotating images at different angles can provide additional training samples. These techniques assist the model to recognize objects from different orientations.

Image scaling and cropping: Scaling an image to different sizes or cropping it at various locations can simulate different viewpoints or zoom levels. This allows the model to handle variations in object size and position.

Image translation: Shifting an image in different directions can create new samples with variations in object location. This helps the model become more invariant to translation and improves its ability to recognize objects in different positions.

Image shearing and skewing: Applying shearing or skewing transformations to images can introduce deformations and changes in perspective. These transformations can enhance the model's ability to handle distorted or perspective-shifted objects, (Shorten & Khoshgoftaar, 2019).

A list of previous works to identify breast cancer using convolutional neural networks conducted by different scholars are summarized in Table 2.1.

Table: 2.1: Related works

RESEARCH TITLE	**DATA ANALYSIS TOOL**	**DATASETS USED**	**EVALUATION**	**RESEARCH GAP**
Yurttakal, A., Akpolat, T., & Arslan, A. (2020). Detection of Breast Cancer via Deep Convolution Neural Networks Using MRI Images. Journal of Medical Imaging and Health Informatics, 10(9), 2160-2166.	MATLAB environment	Breast MRI images of 200 Cases among them, 98 are benign and 102 malignant from (Turkey)	sensitivity, specificity, precision, F1 Score, False Negative Rate, False Discovery Rate, False Positive Rate, Negative Predictive Value and Classification Accuracy,	The research gap in this paper revolves around the need for larger, more diverse datasets and additional validation studies to enhance the generalizability and reliability of the CNN model for breast cancer detection using MRI images.
Zuluaga-Gomez, A., Lopez, J. A., & Ramirez, J. (2021). A CNN-Based Methodology for Breast Cancer Diagnosis Using Thermal Images. Journal of Medical Imaging and Health Informatics, 11(5), 1260-1267.	Python	1120 thermal images from DMR-IR Database (UK)	accuracy, precision, sensitivity, F1-score and ROC-AUC	Breast cancer characteristics and patient populations can vary significantly across different regions and healthcare systems. Therefore, training a CNN model on thermal images from a single database may not adequately capture the variability present in real-world clinical settings. To address this gap, future research could benefit from incorporating thermal images from multiple sources or

				conducting multi-center studies to ensure the robustness and applicability of the developed methodology across diverse populations.
Alanazi, A., AlRubaian, M., & AlShammari, R. (2021). Boosting Breast Cancer Detection Using Convolutional Neural Network. Journal of Medical Imaging and Health Informatics, 11(8), 2350-2356.	Python	275,000, 50 ×50-pixel RGB image patches from Kaggle	accuracy, precision, sensitivity	The research gap in the study lies in the lack of specificity regarding the origin and representativeness of the dataset, making it challenging to assess the extent to which the trained CNN model can be generalized to real-world scenarios, particularly within the context of the local healthcare system where it is intended to be applied. Moreover, potential issues related to data bias, imbalance, and quality within the Kaggle dataset could impact the performance and reliability of the trained model. Thus, future research could address these limitations by utilizing more curated and well-documented datasets specifically tailored to the target population, thereby enhancing the applicability and effectiveness of the developed CNN model for breast cancer detection.
Ragab, Y., Attallah, O., & El-Fishawy, N. (2019). Breast cancer detection using deep convolutional neural networks and Support Vector Machines. IEEE Access, 7, 53168-53175.	MATLAB	The digital database for screening mammography (DDSM) dataset consists of 2,620 cases and the Curated Breast Imaging Subset of DDSM (CBIS-	Accuracy, sensitivity, specificity Area under the curve, F1-score	The research gap in this study lies in the reliance on standardized datasets, specifically the digital database for screening mammography (DDSM) dataset and its subset, the Curated Breast Imaging Subset of DDSM (CBIS-DDSM). While these datasets are widely used in breast cancer research, they may not fully capture the diversity of patient populations and imaging practices

		DDSM) datasets contains 753 micro calcification cases and 891 mass cases		encountered in clinical settings. Therefore, there is a need for research that utilizes more diverse and representative datasets, possibly including data from multiple healthcare institutions or regions, to enhance the generalizability and applicability of deep convolutional neural networks (CNNs) and Support Vector Machines (SVMs) for breast cancer detection across different patient demographics and imaging protocols
Fonseca, J. F., Carneiro, G., Aridas, C. K., & Marcomini, K. D. (2015). Automatic Breast Density Classification Using a Convolutional Neural Network Architecture Search Procedure.	C programming language using the Open MP library	94 mammograms datasets from two medical centers in Lima, Peru	Accuracy	The research gap in the study conducted by Fonseca et al. (2015) lies in the limited scope of the dataset used for training the convolutional neural network (CNN) model. While the study utilizes mammogram datasets from two medical centers in Lima, Peru, totaling 94 cases, the dataset may not fully capture the diversity and variability present in breast cancer cases across different populations. The lack of a more comprehensive and representative dataset from a broader demographic range hinders the generalizability and applicability of the developed CNN model. Consequently, there is a need for research that incorporates larger and more diverse datasets to improve the accuracy and robustness of breast density classification models, ensuring their effectiveness across various patient populations and healthcare settings.
Yue, Z., Fan, Y., Zhang, Y., Wang, S., Li, Y., & Zhang,		WBCD dataset	Classification accuracy	The research gap in Yue et al.'s (2018) study lies in the focus on utilizing the

Z. (2018). Machine Learning with Applications in Breast Cancer Diagnosis and Prognosis. Journal of Healthcare Engineering, 2018, 1-6.				Wisconsin Breast Cancer Dataset (WBCD) for breast cancer diagnosis and prognosis without addressing the potential limitations of using a single dataset for training and evaluation. While classification accuracy is reported as an outcome measure, the study lacks consideration of the dataset's representativeness and generalizability to diverse patient populations. Additionally, there is a need for further investigation into the robustness of the machine learning models developed using the WBCD dataset when applied to real-world clinical settings with varying data distributions and patient demographics. Therefore, future research should aim to validate the findings using multiple datasets from different sources to ensure the reliability and applicability of the developed models in clinical practice.
Eroğlu, O., Karacan, A., Ceylan, H., Kocatürk, T., & Arslan, A. (2021). Convolutional Neural Networks based Classification of Breast Ultrasonography Images by Hybrid Method with respect to Benign, Malignant, and Normal. Journal of Healthcare Engineering, 2021, 1-14.	MATLAB 2019b environment	780 breast ultrasound images () in png format in which 437 are benign, 210 malignant, 133 normal images collected from. This data set was obtained at Behaye hospital (Europe PMC site)	Accuracy, AUC	The research gap in the paper by Eroğlu et al. (2021) lies in the limited scope of the dataset used for training and evaluation. While the study employs breast ultrasound images collected from a single source, Behaye Hospital, the dataset may not fully capture the diversity and variability present in broader patient populations. Additionally, the study does not address potential biases or limitations associated with the specific hospital setting, patient demographics, or imaging protocols. Therefore, there is a need for further research that

				incorporates larger and more diverse datasets, potentially sourced from multiple healthcare institutions, to enhance the generalizability and robustness of the developed convolutional neural network (CNN) models for breast cancer classification using ultrasound images.
Al-Haija, A. M., & Adebanjo, A. T. (2020). Breast Cancer Diagnosis in Histopathological Images using Resnet-50 Convolutional Neural Network. Journal of Medical Imaging and Health Informatics, 10(12), 2875-2882.	Python 3.7	$BreakHis$ dataset composed 9,109 microscopic images of breast tumor tissue collected from 82 patients, it contains 2,480 benign and 5,429 malignant samples (700X460 pixels, 3-channel RGB, 8-bit depth in each channel, PNG format)	accuracy	The research gap in the study by Al-Haija and Adebanjo (2020) lies in the limited diversity and representativeness of the dataset used for training the breast cancer diagnosis model. While the BreastHist dataset consists of a substantial number of histopathological images, all samples are collected from a single source, potentially leading to biases and limitations in the model's generalizability. The dataset's exclusivity to a specific patient population or healthcare institution may not adequately capture the full spectrum of breast tumor tissue variations, histological characteristics, and patient demographics present in broader clinical settings. Consequently, there is a need for research that incorporates more diverse and comprehensive datasets encompassing a wider range of tumor types, tissue structures, and patient populations to enhance the robustness and applicability of breast cancer diagnosis models trained using convolutional neural networks.
Zhao, H., Shi, J., Qi, X., Wang, X., & Jia,	Python	122 digital mammogra	Accuracy	The research gap in Zhao et al.'s (2018) paper lies in the

J. (2018). Classification of Benign and Malignant Breast Mass in Digital Mammograms with Convolutional Neural Networks. ISICDM 2018: Proceedings of the 2nd International Symposium on Image Computing and Digital Medicine, October 2018, pp. 47–50.		m images in which 54 malignant cases and 68 benign cases downloaded from the Mammography Image Analysis Society (MIAS) database		limited size and diversity of the dataset used for training the convolutional neural networks (CNNs) for classifying benign and malignant breast masses in digital mammograms. While the utilization of digital mammogram images from the Mammography Image Analysis Society (MIAS) database is a valid approach, the dataset consists of only 122 images, with 54 classified as malignant and 68 as benign. This small dataset size may not fully capture the variability and complexity of real-world mammogram images, potentially limiting the generalizability and robustness of the CNN models developed. Therefore, there is a need for future research to explore larger and more diverse datasets to enhance the accuracy and reliability of breast cancer classification using CNNs in digital mammograms.
Lu, H., Zhang, Y., Cao, S., & Zhu, Q. (2019). The Classification of Mammogram using Convolutional Neural Network with Specific Image Preprocessing for Breast Cancer Detection. Journal of Medical Systems, 43(8), 234.	MATLAB environment	A total of 2363 examinees with BI-RADS 0, 1, 2, 3, 4, and 5 were collected from a teaching hospital in Taiwan and 9927 images with resolution 2294*1914 were obtained	Accuracy, sensitivity, specificity, and F1 score	The research gap in Lu et al.'s (2019) study lies in the absence of validation and evaluation using diverse datasets from multiple healthcare settings or geographical regions. While the study employs a dataset collected from a teaching hospital in Taiwan, it is essential to assess the generalizability of the developed convolutional neural network (CNN) model across different patient populations and healthcare systems. Validation using datasets from various sources would provide insights into

				the model's robustness and effectiveness in different clinical contexts, thus enhancing its applicability and reliability in real-world breast cancer detection scenarios.

Table 2.1 presents literature review on various studies related to detection and diagnosis of breast cancer using deep learning techniques. The studies utilize different data analysis tools, datasets and evaluation metrics.

Yurttakal et al.,(2020) conducted a study with the goal of early breast cancer detection through the application of MRI images and deep convolutional neural networks (CNNs). They achieved successful differentiation between benign and malignant tumors using a dataset of 200 breast MRI images from Turkey.

Zuluaga-Gomez et al., (2021) focused on developing a CNN-based computer-aided diagnosis system for breast cancer using thermal images. The study demonstrated the superiority of CNNs over other techniques using a dataset of 1120 thermal images from the DMR-IR Database (UK). However, the identified gap is the usage of datasets from abroad, suggesting the potential use of local datasets.

Alanazi et al.,(2021) aimed to classify IDC-positive and negative cases and compared performance with other machine learning models using Python. The research used a dataset of 275,000 RGB image patches from Kaggle. The gap identified was the reliance on secondary datasets from Kaggle, suggesting the exploration of local hospital datasets.

Ragab et al., (2019) examined the process of identifying masses and distinguishing between benign and malignant tissues in mammograms by employing a combination of Deep Convolutional Neural

Networks (DCNN) for extracting features and Support Vector Machine (SVM) for classification. The experimentation was carried out using MATLAB. The study utilized data from two sources: the digital database for screening mammography (DDSM) and the Curated Breast Imaging Subset of DDSM (CBIS-DDSM).

Fonseca et al., (2015) focused on automatic breast density classification using a CNN architecture search procedure in C programming language. The research employed 94 mammograms datasets from two medical centers. However, the absence of a more comprehensive and representative dataset from a broader demographic range undermines the generalizability and applicability of the developed CNN model. Therefore, there is a need for further research that incorporates larger and more diverse datasets to enhance the accuracy and robustness of breast density classification models, ensuring their effectiveness across various patient populations and healthcare settings.

Yue et al.(2018) conducted a review of various ML techniques' applications in breast cancer diagnosis and prognosis using the WBCD dataset. Additionally, there is a need for further investigation into the robustness of the machine learning models developed using the WBCD dataset when applied to real-world clinical settings with varying data distributions and patient demographics. Therefore, future research should aim to validate the findings using multiple datasets from different sources to ensure the reliability and applicability of the developed models in clinical practice.

Eroğlu et al. (2021) developed a hybrid-based CNN system to classify breast cancer lesions into three categories: benign, malignant, or normal. using MATLAB. The research utilized 780 breast ultrasound images from Behaye hospital (Europe PMC site). However, the research gap in this study lies in the limited scope of the dataset used for training and evaluation. While the study employs breast ultrasound images collected from a single source, Behaye Hospital, the dataset may not fully capture the diversity and variability present in broader patient populations. Additionally, the study does not address potential biases or limitations associated with the specific hospital setting, patient demographics, or imaging protocols. Therefore, there is a need for further research that incorporates larger and more diverse datasets, potentially sourced from multiple healthcare institutions, to enhance the generalizability and robustness of the developed convolutional neural network (CNN) models for breast cancer classification using ultrasound images.

Al-Haija & Adebanjo (2020) investigated breast cancer analysis in histopathological images using the Resnet-50 CNN model in Python. The research employed the Breast Histology dataset. Additionally, there is a need for research that incorporates more diverse and comprehensive datasets encompassing a wider range of tumor types, tissue structures, and patient populations to enhance the robustness and applicability of breast cancer diagnosis models trained using convolutional neural networks.

Zhao et al., (2018) performed experimental comparisons in Python between CNN-based and SVM-based classifiers to classify benign and malignant cases. The study used 122 digital mammogram images from the MIAS database. The research gap in this study lies in the limited size and diversity of the dataset used for training the convolutional neural networks (CNNs) for classifying benign and malignant breast masses in digital mammograms. While the utilization of digital mammogram images from the Mammography Image Analysis Society (MIAS) database is a valid approach, the dataset consists of only 122 images, with 54 classified as malignant and 68 as benign. This small dataset size may not fully capture the variability and complexity of real-world mammogram images, potentially limiting the generalizability and robustness of the CNN models developed.

Lu et al. (2019) developed an aiding system for breast cancer detection and staging using MATLAB. The study employed a dataset of 2363 examinees from a hospital in Taiwan. The research gap in this study lies in the absence of validation and evaluation using diverse datasets from multiple healthcare settings or geographical regions. While the study employs a dataset collected from a teaching hospital in Taiwan, it is essential to assess the generalizability of the developed convolutional neural network (CNN) model across different patient populations and healthcare systems. Validation using datasets from various sources would provide insights into the model's robustness and effectiveness in different clinical contexts, thus enhancing its applicability and reliability in real-world breast cancer detection scenarios.

2.6 Research Gaps

Several researches have been conducted to predict breast cancer. For example, Yurttakal et al. (2020) developed a CNN model using pixel information in MATLAB, utilizing datasets from the UCI machine learning repository with varying training and testing data sizes. However, many of these studies relied on secondary data from foreign sources, such as the UCI machine learning repository. In order to address this limitation and provide a more suitable model for breast cancer identification, particularly in our country, this study aims to apply a CNN algorithm using Python. The study will utilize secondary data in the form of MRI images from our local hospital, specifically Muhimbili National Hospital (MNH). By using these datasets, the study aims to accurately classify malignant and benign tumors, thereby improving breast cancer detection in our country. Additionally, the implementation of the model has been carried out in Python, a widely used programming language in the field of data science. Python is renowned for its extensive collection of valuable libraries that cater to scientific computing and machine learning tasks.

2.7 The conceptual framework

The conceptual framework illustrated in Figure 2.3 depicts the process of training a developed Convolutional Neural Network (CNN) model to predict whether a tumor is benign or malignant using data from the MNH Breast Cancer dataset. The dataset serves as the primary data source, containing MRI images of breast tumors. These images undergo pre-processing, including standardization, and resizing, before feature extraction. Features extracted from the images, such as radius, perimeter, texture, area, smoothness, edges, and shapes of tumors, serve as independent variables for training the CNN model. The developed CNN model learns from the training datasets and is subsequently evaluated using validation datasets to assess its generalization to unseen data. The dependent variable, the predicted result of whether the tumor is benign or malignant, is determined by the CNN model, which is then used to make predictions on new data. This framework outlines the essential steps in utilizing CNNs for breast cancer prediction, starting from data collection to model development and deployment.

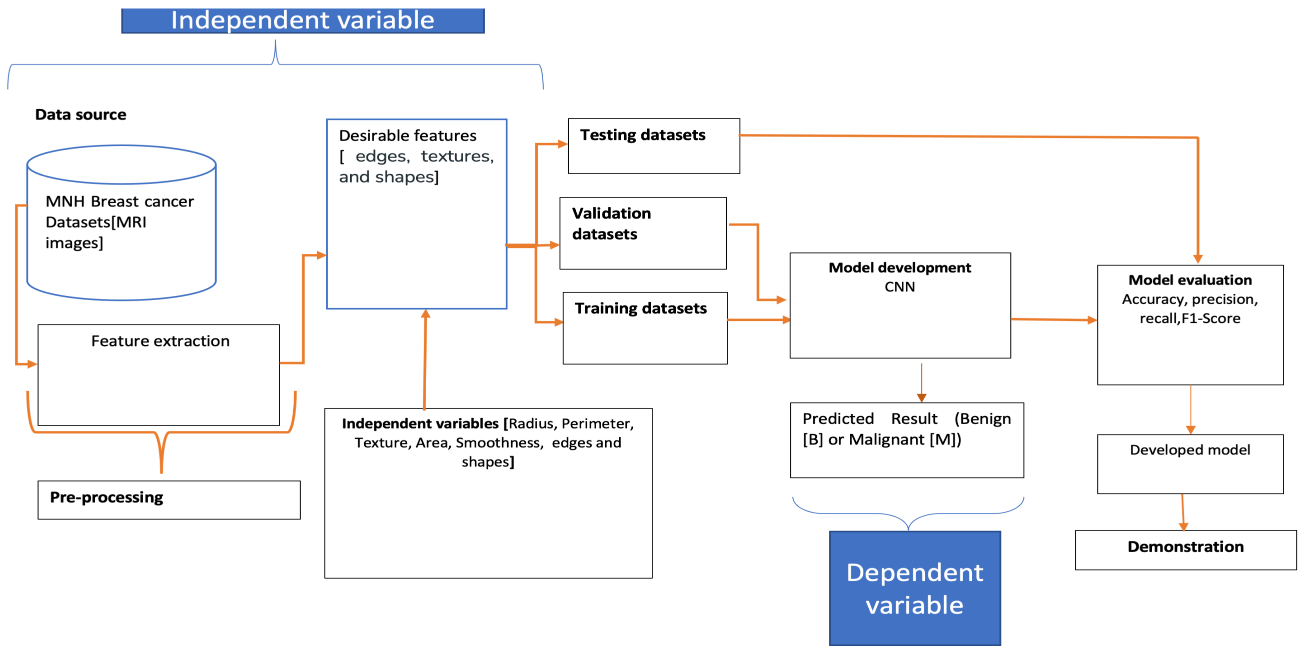

Independent variable
Data source
MNH Breast cancer Datasets[MRI images]
Feature extraction
Pre-processing
Desirable features [edges, textures, and shapes]
Independent variables [Radius, Perimeter, Texture, Area, Smoothness, edges and shapes]
Testing datasets
Validation datasets
Training datasets
Model development CNN
Predicted Result (Benign [B] or Malignant [M])
Dependent variable
Model evaluation Accuracy, precision, recall,F1-Score
Developed model
Demonstration

Figure 2.3: The conceptual framework of the study (Ramadhani,2024).

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Overview

An experimental study design has been adopted. Experimental research design act as a de facto research design in modelling machine learning problems, (Kamiri & Mariga, 2021).

This chapter provides a comprehensive overview of the CNN model developed as part of this research, focusing on image processing and model architecture. The chapter begins with a detailed explanation of the image augmentation methods and algorithm employed, shedding light on the details of segmenting MRI images for accurate analysis. Additionally, the chapter delves into the datasets utilized in the study, explaining the process of dataset selection, training, validation, and testing. By thoroughly explaining the dataset handling procedures, readers gain insight into the rigorous evaluation process employed to ensure the model's accuracy and reliability.

In Figure 3.1, a CNN model for breast cancer classification is illustrated. The model begins with image pre-processing, where medical images of breast tissue undergo resizing, pixel intensity normalization, and format conversion to prepare them for the deep learning model. Subsequently, the pre-processed images are fed into the model for feature extraction, which involves capturing numerical representations of crucial characteristics like shape, texture, and intensity. These extracted features are then used to train a classification model, Once trained, the model can predict the likelihood of breast tissue being benign or malignant when presented with new breast MRI images.

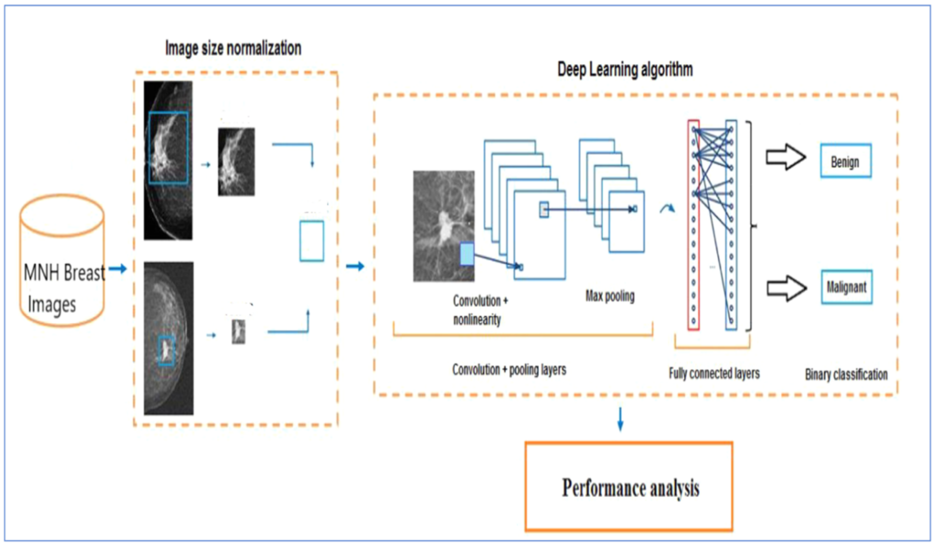


Figure: 3.1 Developed CNN model architecture (Ramadhani,2024)

3.2 Study population and sample size

The research sample comprises 30 MRI images from patients who received breast cancer screenings at Muhimbili National Hospital (MNH).The patient's MRI images included in the study were 8 for malignant and 22 for benign images. Furthermore, the images were augmented to 1419 images includes 719 benign and 700 malignant image using different augmentation techniques, such as shearing, flipping, rotation, cropping, and shifting.

3.3 Area of study

Muhimbili National Hospital (MNH), located in Dar es salaam , serves as the primary research site for this study. The hospital was chosen due to its significance as a leading healthcare institution in the region, providing access to a substantial number of breast cancer cases. Conducting the research in this setting ensures that the findings are relevant to the local context and can contribute to improving breast cancer detection and treatment in the area.

3.4 Hardware and Software

The model was developed using MacBook Pro 2019 which offers powerful specifications including a 16-inch Retina display, 2.3 GHz 8-Core Intel Core i9 processor, AMD Radeon Pro 5500M graphics, and 16 GB of DDR4 memory, provides an ideal platform for developing CNN model with Python, offering efficient processing, graphics rendering, and multitasking capabilities.

3.5 Ethical Consideration

The study observed ethical issues and the images was anonymized and all necessary ethical considerations was taken into considerations to ensure patient privacy and confidentiality. Therefore, before conducting this study the permission was obtained from Research and Publication Centre at Muhimbili National Hospital.

3.6 Research Design

In this study, an experimental research design was employed to address the gap in breast cancer detection in Tanzania by leveraging Convolutional Neural Networks (CNNs) on local MRI breast images sourced from Muhimbili National Hospital. The dataset initially comprised 30 MRI images, with 8 malignant and 22 benign cases, acknowledging the challenge of data scarcity. To mitigate this limitation and the risk of overfitting, data augmentation techniques such as rotation, shifting, flipping, and shearing were applied using Python, resulting in a dataset expansion to 1419 images, encompassing 700 benign and 719 malignant cases. The developed CNN model, implemented in Python, featured multiple layers including Conv2D layers for feature extraction, MaxPooling2D layers for enhancing feature capture, and Dense layers for classification. The dataset was divided into training (50%), validation (40%), and testing (10%) sets. The model demonstrated strong performance metrics, achieving an accuracy of 96.4% and an F1 score of 96%, indicating its efficacy in accurately identifying breast cancer cases.

3.7 Data Preprocessing

The study utilized a dataset consisting of 30 MRI images of breast tumors in DICOM (Digital Imaging and Communications in Medicine) format. DICOM is a standard file format used for storing, exchanging, and transmitting medical images and related information in the healthcare industry. It is widely used in medical imaging modalities such as X-rays, MRIs, CT scans, ultrasounds, and more. The 30 MRI images consisted of 8 malignant and 22 benign cases, then these images were then converted to png format using HOROS software as shown in Figure 3.2. Horos is a no-cost, openly accessible software designed for viewing medical images. It utilizes OsiriXTM and various other open-source medical imaging libraries as its foundation.

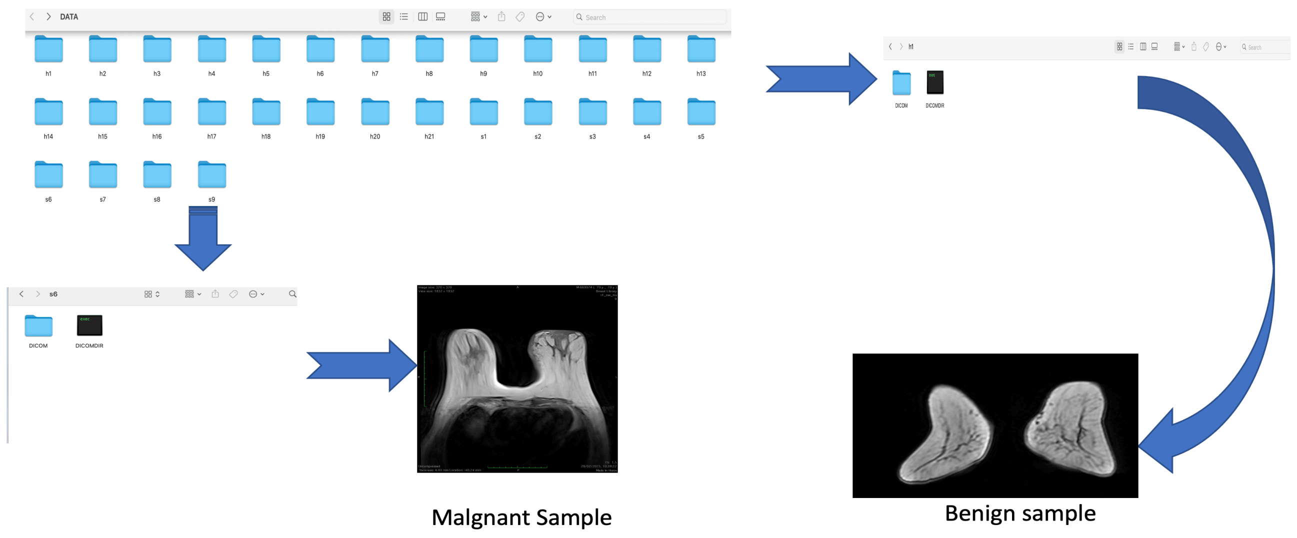

DATA
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h14 h15 h16 h17 h18 h19 h20 h21 s1 s2 s3 s4 s5
s6 s7 s8 s9
s6
DICOM
DICOMDIR
Malgnant Sample
Benign sample

Figure 3.2: DICOM files format conversion to png format.

3.8 Image Augmentation

To enhance the dataset and increase its size, data augmentation techniques were applied. Various augmentation methods such as shearing, cropping, rotation, flipping and shifting were used to generate a total of 1419 MRI images as shown in Figure 3.3. The augmented dataset includes 700 malignant and 719 benign cases, providing a more extensive and diverse dataset for training and evaluation, as shown in Figures 3.5 and 3.6.

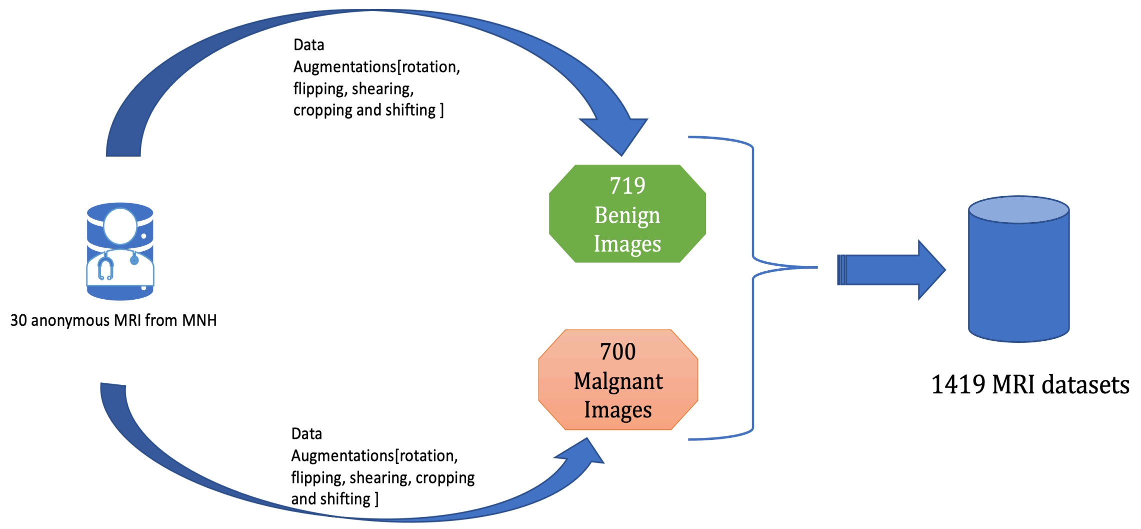

Data Augmentations[rotation, flipping, shearing, cropping and shifting]
719 Benign Images
30 anonymous MRI from MNH
700 Malgnant Images
1419 MRI datasets
Data Augmentations[rotation, flipping, shearing, cropping and shifting]

Figure 3.3: Data augmentation techniques (Ramadhani,2024).

3.8.1 Image Rotation

This technique involves rotating an image by a certain angle. It assists the model become more robust to variations in object orientations.

3.8.2 Image Flipping

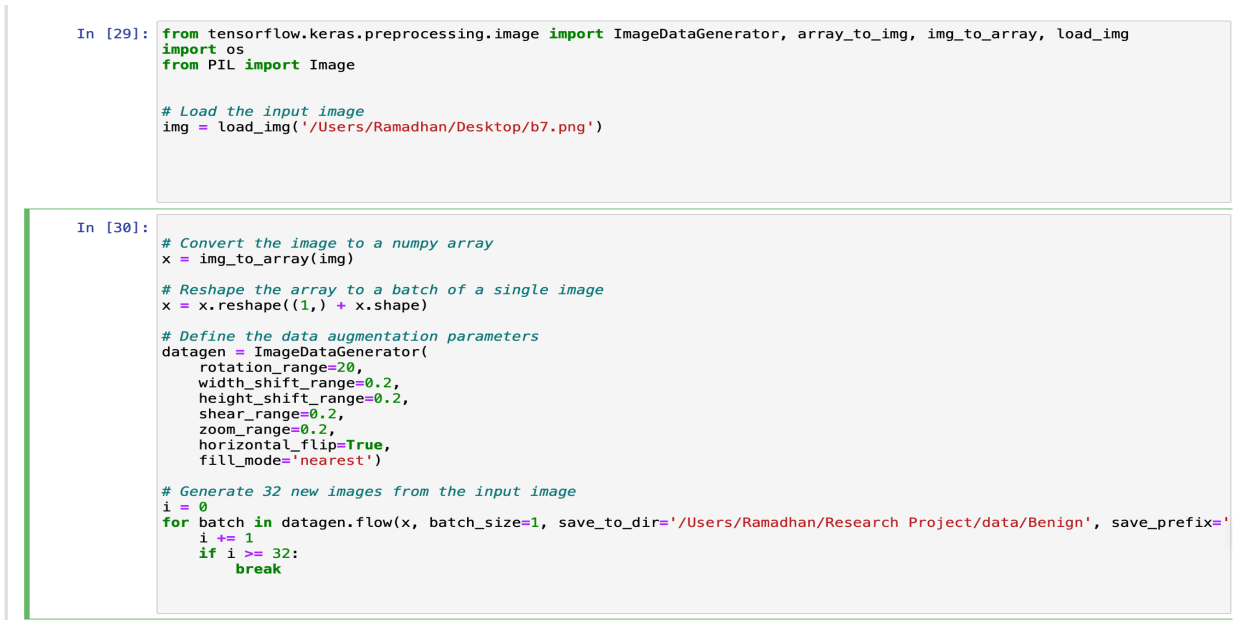
Modifying an image by flipping it either horizontally or vertically is an uncomplicated augmentation method, which effectively enhances the diversity of the dataset.

3.8.3 Image Shearing

Image shearing involves shifting the pixels in a particular direction, giving the image a skewed appearance.

3.8.4 Image Shifting

Shifting an image involves moving the pixels horizontally or vertically by a certain distance. It can help the model become more robust to object translations or changes in position.



```
In [29]: from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
         import os
         from PIL import Image


         # Load the input image
         img = load_img('/Users/Ramadhan/Desktop/b7.png')
```

```
In [30]:
         # Convert the image to a numpy array
         x = img_to_array(img)

         # Reshape the array to a batch of a single image
         x = x.reshape((1,) + x.shape)

         # Define the data augmentation parameters
         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')

         # Generate 32 new images from the input image
         i = 0
         for batch in datagen.flow(x, batch_size=1, save_to_dir='/Users/Ramadhan/Research Project/data/Benign', save_prefix='
             i += 1
             if i >= 32:
                 break
```

Figure 3.4: Code snippet for data augmentation techniques used (Ramadhani,2024).

Figure 3.4 shows the code snippet detailing the data augmentation parameters utilized in this study via the ImageDataGenerator class from the Keras library. The parameters and libraries employed are explained, along with their respective purposes :-

- **TensorFlow**: TensorFlow stands as an open-source platform for machine learning. In this case, we are using the TensorFlow library's submodule tensorflow.keras for deep learning tasks, particularly related to image processing and computer vision.
- **ImageDataGenerator**: This is a class from **tensorflow.keras.preprocessing.image** is used for data augmentation and pre-processing of images. It permits the creation of augmented renditions of images through the application of different changes like rotation, scaling, flipping, and more. This technique is widely employed to enhance the variety and amount of training data available.
- **array_to_img and img_to_array**: These functions, also from **tensorflow.keras.preprocessing.image**, are used to convert between image data formats. array_to_img converts a NumPy array representation of an image to a PIL Image object, while img_to_array converts a PIL Image object to a NumPy array.
- **load_img**: This is another function from **tensorflow.keras.preprocessing.image,** load_img is used to load an image file from a given path as a PIL Image object. It is often used to read image files before further processing or augmentation.
- **os**: The os module is an intrinsic part of Python and serves the purpose of communicating with the operating system. Its primary function is to handle various operations related to files and directories, encompassing actions such as file listing, directory creation, and path manipulation. In this code excerpt, it is probably utilized to interface with the file system and gain access to image files.
- **PIL**: PIL (Python Imaging Library) is a popularly employed library utilized for accessing, editing, and storing a wide range of image file formats. In this code, the PIL module is imported to facilitate working with PIL Image objects.

- **Rotation_range**: It specifies the range of random rotations that can be applied to the images. In this case, the images were rotated up to 20 degrees in either clockwise or counterclockwise direction.
- **Width_shift_range and height_shift_range**: These parameters define the extent of random horizontal and vertical shifts that can be employed on the images. A value of 0.2 implies that the images can undergo horizontal and vertical shifting of up to 20% of the image's width and height, respectively.
- **Shear_range**: It determines the range of random shearing transformations that can be applied to the images. A shear transformation shifts the position of pixels along a certain direction. In this study, the images were sheared by a maximum of 20% in any direction.
- **Zoom_range**: This parameter determines the extent of random zooming applicable to the images. Zooming refers to altering the image's scale, and in this instance, the images were subjected to a maximum zoom of 20%, either zoomed in or out.
- **Horizontal_flip**: It specifies whether random horizontal flips should be applied to the images. Enabling this parameter means that some of the images were horizontally flipped.
- **Fill_mode**: It determines how to fill in the pixels that may appear after applying transformations such as shifting or rotation. The 'nearest' mode fills in any empty areas with the nearest available pixel value.

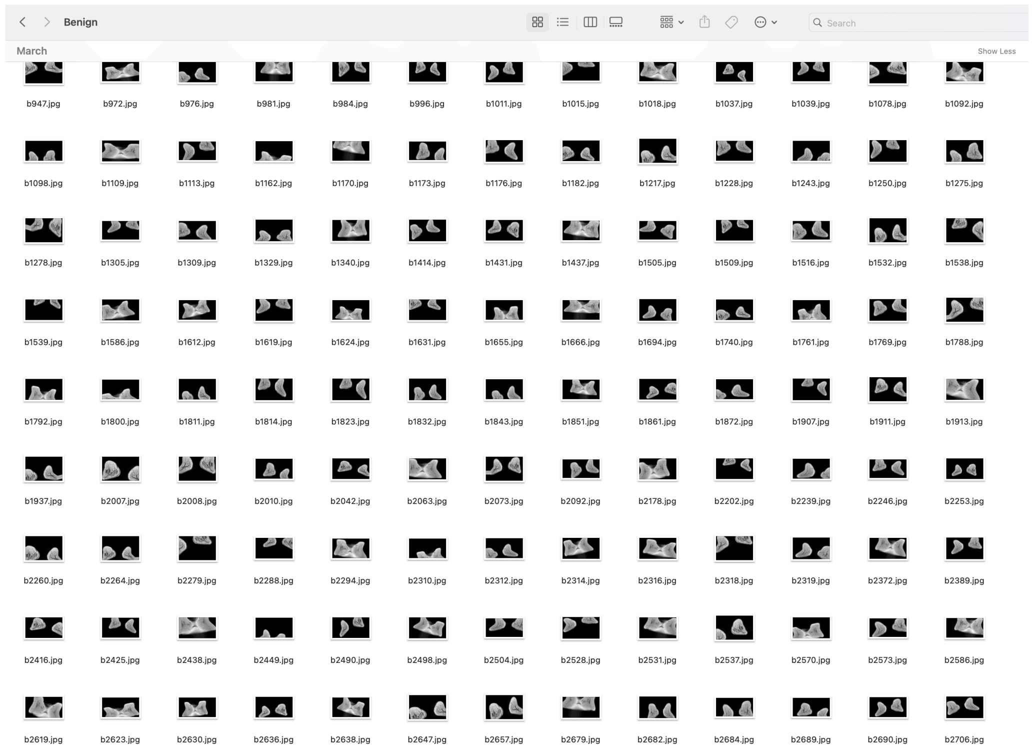

Benign
March
Show Less
Search
b947.jpg b972.jpg b976.jpg b981.jpg b984.jpg b996.jpg b1011.jpg b1015.jpg b1018.jpg b1037.jpg b1039.jpg b1078.jpg b1092.jpg
b1098.jpg b1109.jpg b1113.jpg b1162.jpg b1170.jpg b1173.jpg b1176.jpg b1182.jpg b1217.jpg b1228.jpg b1243.jpg b1250.jpg b1275.jpg
b1278.jpg b1305.jpg b1309.jpg b1329.jpg b1340.jpg b1414.jpg b1431.jpg b1437.jpg b1505.jpg b1509.jpg b1516.jpg b1532.jpg b1538.jpg
b1539.jpg b1586.jpg b1612.jpg b1619.jpg b1624.jpg b1631.jpg b1655.jpg b1666.jpg b1694.jpg b1740.jpg b1761.jpg b1769.jpg b1788.jpg
b1792.jpg b1800.jpg b1811.jpg b1814.jpg b1823.jpg b1832.jpg b1843.jpg b1851.jpg b1861.jpg b1872.jpg b1907.jpg b1911.jpg b1913.jpg
b1937.jpg b2007.jpg b2008.jpg b2010.jpg b2042.jpg b2063.jpg b2073.jpg b2092.jpg b2178.jpg b2202.jpg b2239.jpg b2246.jpg b2253.jpg
b2260.jpg b2264.jpg b2279.jpg b2288.jpg b2294.jpg b2310.jpg b2312.jpg b2314.jpg b2316.jpg b2318.jpg b2319.jpg b2372.jpg b2389.jpg
b2416.jpg b2425.jpg b2438.jpg b2449.jpg b2490.jpg b2498.jpg b2504.jpg b2528.jpg b2531.jpg b2537.jpg b2570.jpg b2573.jpg b2586.jpg
b2619.jpg b2623.jpg b2630.jpg b2636.jpg b2638.jpg b2647.jpg b2657.jpg b2679.jpg b2682.jpg b2684.jpg b2689.jpg b2690.jpg b2706.jpg

Figure 3.5: Benign sample Images after augmentations (Ramadhani,2024).

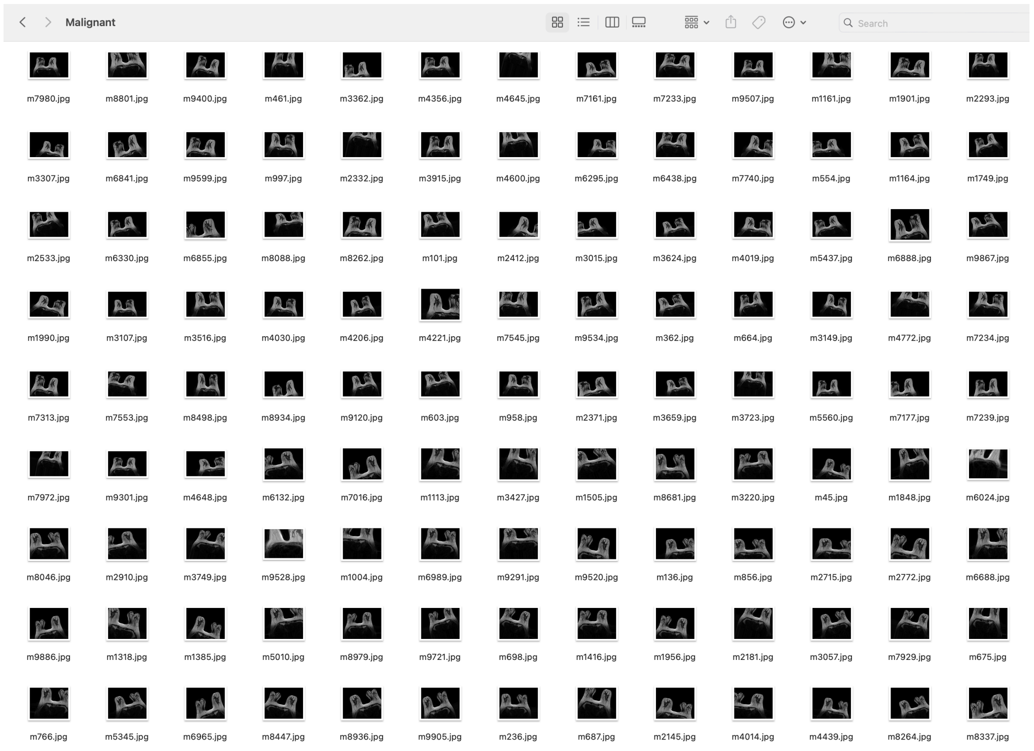
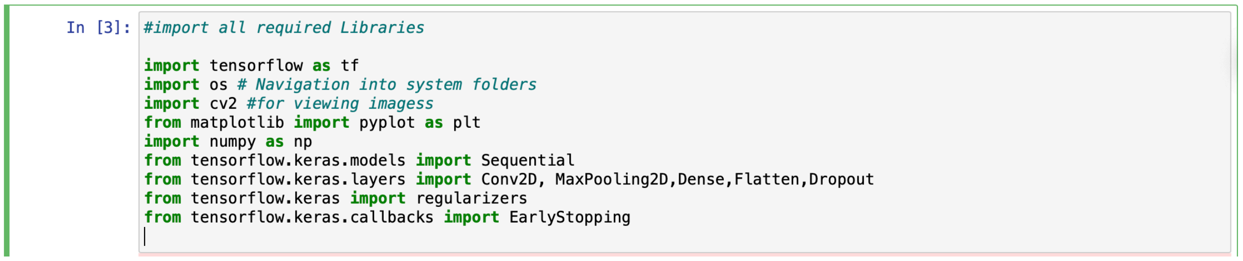

Malignant
Search
m7980.jpg m8801.jpg m9400.jpg m461.jpg m3362.jpg m4356.jpg m4645.jpg m7161.jpg m7233.jpg m9507.jpg m1161.jpg m1901.jpg m2293.jpg
m3307.jpg m6841.jpg m9599.jpg m997.jpg m2332.jpg m3915.jpg m4600.jpg m6295.jpg m6438.jpg m7740.jpg m554.jpg m1164.jpg m1749.jpg
m2533.jpg m6330.jpg m6855.jpg m8088.jpg m8262.jpg m101.jpg m2412.jpg m3015.jpg m3624.jpg m4019.jpg m5437.jpg m6888.jpg m9867.jpg
m1990.jpg m3107.jpg m3516.jpg m4030.jpg m4206.jpg m4221.jpg m7545.jpg m9534.jpg m362.jpg m664.jpg m3149.jpg m4772.jpg m7234.jpg
m7313.jpg m7553.jpg m8498.jpg m8934.jpg m9120.jpg m603.jpg m958.jpg m2371.jpg m3659.jpg m3723.jpg m5560.jpg m7177.jpg m7239.jpg
m7972.jpg m9301.jpg m4648.jpg m6132.jpg m7016.jpg m1113.jpg m3427.jpg m1505.jpg m8681.jpg m3220.jpg m45.jpg m1848.jpg m6024.jpg
m8046.jpg m2910.jpg m3749.jpg m9528.jpg m1004.jpg m6989.jpg m9291.jpg m9520.jpg m136.jpg m856.jpg m2715.jpg m2772.jpg m6688.jpg
m9886.jpg m1318.jpg m1385.jpg m5010.jpg m8979.jpg m9721.jpg m698.jpg m1416.jpg m1956.jpg m2181.jpg m3057.jpg m7929.jpg m675.jpg
m766.jpg m5345.jpg m6965.jpg m8447.jpg m8936.jpg m9905.jpg m236.jpg m687.jpg m2145.jpg m4014.jpg m4439.jpg m8264.jpg m8337.jpg

Figure 3.6: Augmented malignant sample images (Ramadhani,2024).

3.9 Model development process

The model was developed using Python 3.0 in Jupiter notebook 2.7. and the following were the stages involved :-,

3.9.1 First, Importing all required libraries into Jupiter notebook.



```
In [3]: #import all required Libraries

        import tensorflow as tf
        import os # Navigation into system folders
        import cv2 #for viewing imagess
        from matplotlib import pyplot as plt
        import numpy as np
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Conv2D, MaxPooling2D,Dense,Flatten,Dropout
        from tensorflow.keras import regularizers
        from tensorflow.keras.callbacks import EarlyStopping
```

Figure 3.7: Python libraries imported to be used (Ramadhani,2024).

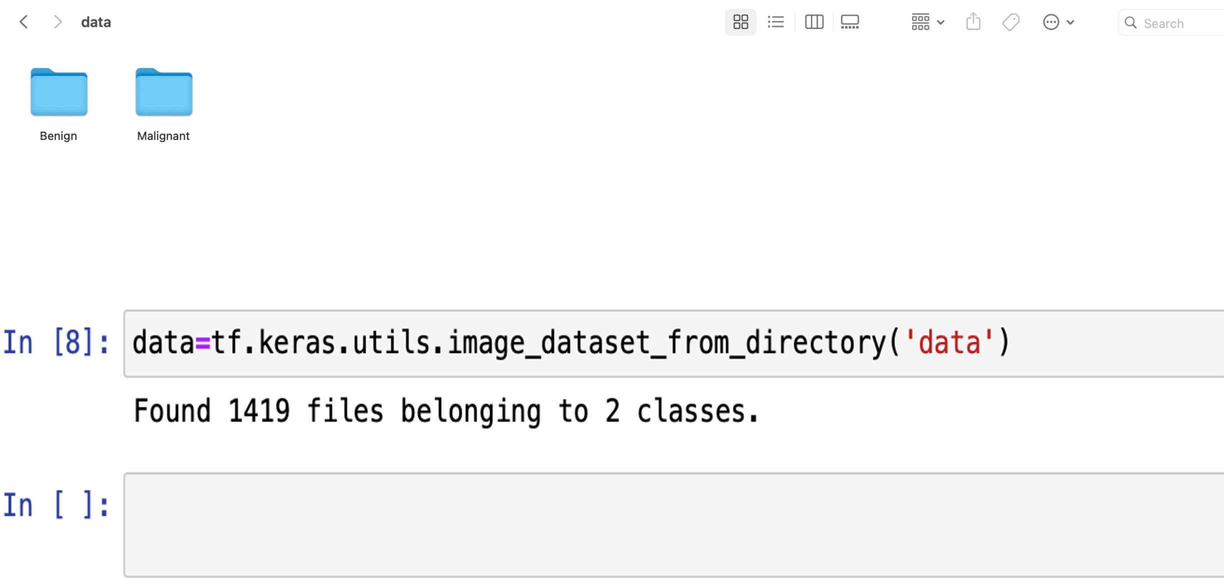
Description of the imported libraries in Figure 3.7 are as follows: -

- cv2: OpenCV, an open-source computer vision library, is employed to perform various tasks related to computer vision and image processing. Within this library, the cv2 module offers functions that enable the reading, manipulation, and visualization of images.
- matplotlib.pyplot (imported as plt): Matplotlib serves as a Python library used for creating plots and visualizations. The pyplot module within Matplotlib offers a straightforward interface to design and customize diverse types of plots and visual elements.
- numpy (imported as np): NumPy is a library for numerical computing in Python. It provides support for large, multi-dimensional arrays and a collection of mathematical functions to operate on these arrays efficiently.

- tensorflow.keras.models.Sequential: The Sequential class in the tensorflow.keras.models module is a linear stack of layers. It is used to build a sequential model, where each layer is added one after the other.
- tensorflow.keras.layers: This contains various types of layers that can be added to a neural network model. The imported layers include Conv2D (convolutional layer), MaxPooling2D (max pooling layer), Dense (fully connected layer), Flatten (flattening layer), and Dropout (dropout layer).
- tensorflow.keras.regularizers: The tensorflow.keras.regularizers module provides regularizers that can be used to apply penalties on layer parameters during model optimization. Regularizers aid in mitigating overfitting by introducing an additional term to the loss function, which serves as a penalty.
- tensorflow.keras.callbacks.EarlyStopping: The EarlyStopping callback is used to stop training the model if a monitored quantity (e.g., validation loss) does not improve for a specified number of epochs. This prevents overfitting and allows early termination of training.

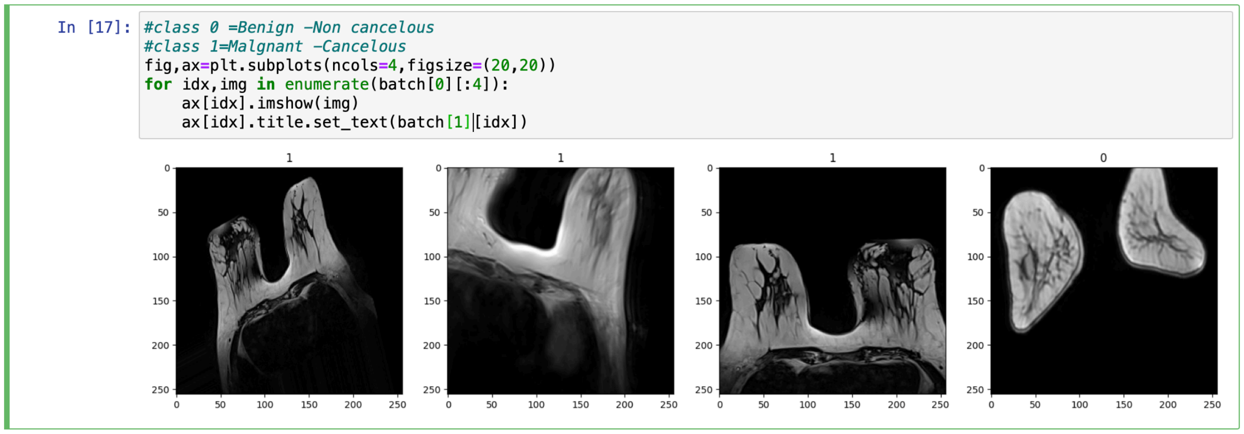
3.9.2 Second, loading image datasets into Jupiter notebook and encoding

This stage was accomplished using a TensorFlow library known as image_datasets_from_directory. This library helps to load image datasets that are in subfolders with two class (in this study was malignant and benign) and provide labels to the existing class where by one class labeled as 0 and another class labeled as 1. Also, the library supports the images in jpeg, png, bmp, gif formats and reshapes them into the size of 256X256 pixels and divide the datasets into 32 batch size as shown in Figure 4.7 and Figure 4.8.



```
In [8]: data=tf.keras.utils.image_dataset_from_directory('data')

Found 1419 files belonging to 2 classes.

In [ ]:
```

Figure 3.8: Image datasets loaded to Jupyter notebook from data directory consists of two subdirectories (Benign and Malignant) (Ramadhani,2024).

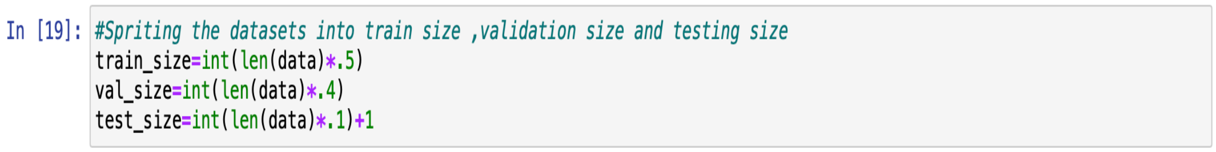


```
In [17]: #class 0 =Benign -Non cancelous
         #class 1=Malgnant -Cancelous
         fig,ax=plt.subplots(ncols=4,figsize=(20,20))
         for idx,img in enumerate(batch[0][:4]):
             ax[idx].imshow(img)
             ax[idx].title.set_text(batch[1][idx])
```

Figure 3.9: Labeled images (0 for Benign images and 1 for Malignant) (Ramadhani,2024).

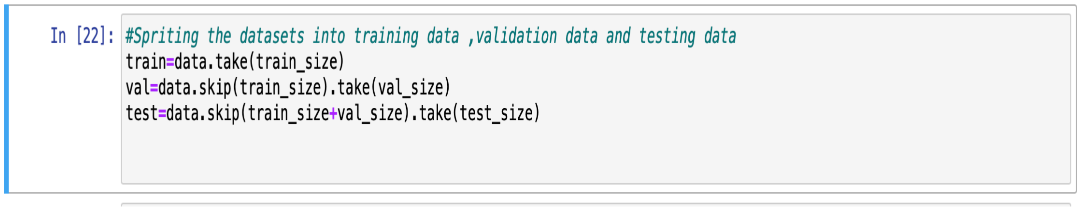
3.9.3 Third, dividing the datasets into training, validation, and test sets.

The datasets were divided into three parts: training, validation, and testing. Specifically, 50% of the images were allocated for training, 40% for validation, and the remaining 10% for testing purposes Additionally, dividing the datasets into training, validation, and test sets serves multiple purposes. Firstly, it helps in assessing the performance of the model during training by providing a separate set of data for validation, which aids in tuning hyperparameters and preventing overfitting. Secondly, the test set allows for the final evaluation of the model's performance on unseen data, providing an unbiased assessment of its generalization ability. By allocating 50% of the images for training, 40% for validation, and the remaining 10% for testing purposes, a balanced distribution ensures robust training, validation, and evaluation processes, as depicted in Figures 3.10 and 3.11.



```
In [19]: #Spriting the datasets into train size ,validation size and testing size
train_size=int(len(data)*.5)
val_size=int(len(data)*.4)
test_size=int(len(data)*.1)+1
```

Figure 3.10: Splitting of the datasets into training, validation and testing variables (Ramadhani,2024).



```
In [22]: #Spriting the datasets into training data ,validation data and testing data
train=data.take(train_size)
val=data.skip(train_size).take(val_size)
test=data.skip(train_size+val_size).take(test_size)
```

Figure 3.11: Splitting of the datasets into training, validation and testing datasets(Ramadhani,2024).

3.9.4 Fourth, creating a Convolutional Neural Network (CNN) model employing diverse layers. The developed CNN model consists of the following layers: -

1. A Conv2D layer comprising 16 filters, each with a kernel size of 3x3, and utilized the ReLU activation function.
    - Conv2D: This layer performs the convolution operation, which involves sliding a small filter (also known as a kernel) over the input image to extract local features. The "2D" in Conv2D refers to the fact that this layer operates on two-dimensional data, such as images. Each filter in the layer learns to detect specific patterns or features within the input.
    - 16 filters: The Conv2D layer comprises 16 individual filters. Each filter is a small matrix of weights that is convolved with the input image. Having multiple filters allows the layer to learn and detect various features simultaneously. In this case, each filter would learn different patterns or feature representations.
    - Kernel size: The size of the kernel dictates the filter's spatial dimensions. In this scenario, the kernel size is 3x3, implying that each filter is represented as a 3x3 matrix. When performing the convolution operation, this filter is employed on a 3x3 section of the input image sequentially. The filter moves across the image, calculating the dot product between the filter's weights and the corresponding values of the input pixels.
    - Rectified Linear Unit (ReLU) activation function: Once the convolution operation is performed, the output of each filter undergoes an element-wise application of an activation function. ReLU is a popular activation function

commonly used in CNNs. It introduces non-linearity to the network by setting all negative values to zero and leaving positive values unchanged. Thus, mathematically it can be written as , ReLU(x) = max (0, x). ReLU activation helps the neural network to learn complex, non-linear relationships between features and enhances the network's ability to generalize and learn more discriminative features.

2. A 2x2 MaxPooling2D layer is used. Another Conv2D layer with 32 filters and a subsequent MaxPooling2D layer were employed, enhancing the network's ability to extract relevant features.
    - The MaxPooling2D layer, utilizing a 2x2 pool size is a common operation used in convolutional neural networks (CNNs) to down sample the input and reduce the spatial dimensions. The pool size determines the size of the pooling window.
    - This layer takes the output from the previous Conv2D layer and performs max pooling with a pool size of 2x2. The layer divides the input into non-overlapping 2x2 regions and picks the maximum value within each region. This operation reduces the spatial dimensions by a factor of two (2), effectively down sampling the feature maps. The purpose of this layer is to capture the most salient features while reducing computational complexity. It enhances the network's ability to generalize and extract relevant features by emphasizing the most prominent features within each region.

- Conv2D (32 filters): After the MaxPooling2D layer, another Conv2D layer with 32 filters is employed. This layer applies 32 filters to the input feature maps, extracting more complex and higher-level qualities from the previous layer's output. The filters in this layer learn to recognize and detect specific patterns or structures within the feature maps. The use of additional filters increases the network's capacity to learn and represent a wider range of features.
- The combination of Conv2D layers and MaxPooling2D layers in this sequence allows the network to learn and extract increasingly complex and meaningful features from the input data. The Conv2D layers identify local patterns, edges, and textures, while the MaxPooling2D layers down sample the feature maps, maintaining the most important information and reducing computational complexity. This architecture helps the network to focus on relevant features, improving its ability to extract and represent the relevant characteristics of the input data.

3. The architecture further includes a Conv2D layer consists of 16 filters, a MaxPooling2D layer, and a Flatten layer that convert the output into a one-dimensional array.
    - Flatten: This layer is used to convert the multidimensional feature maps into a one-dimensional array. It "flattens" the spatial dimensions, resulting in a long vector representation. This layer transforms the output of the previous layer into a format suitable for feeding into the subsequent fully connected layers.

4. The model continued with a Dense layer of 256 units and an activation function(ReLU), followed by a Dropout layer with a dropout rate of 0.2.

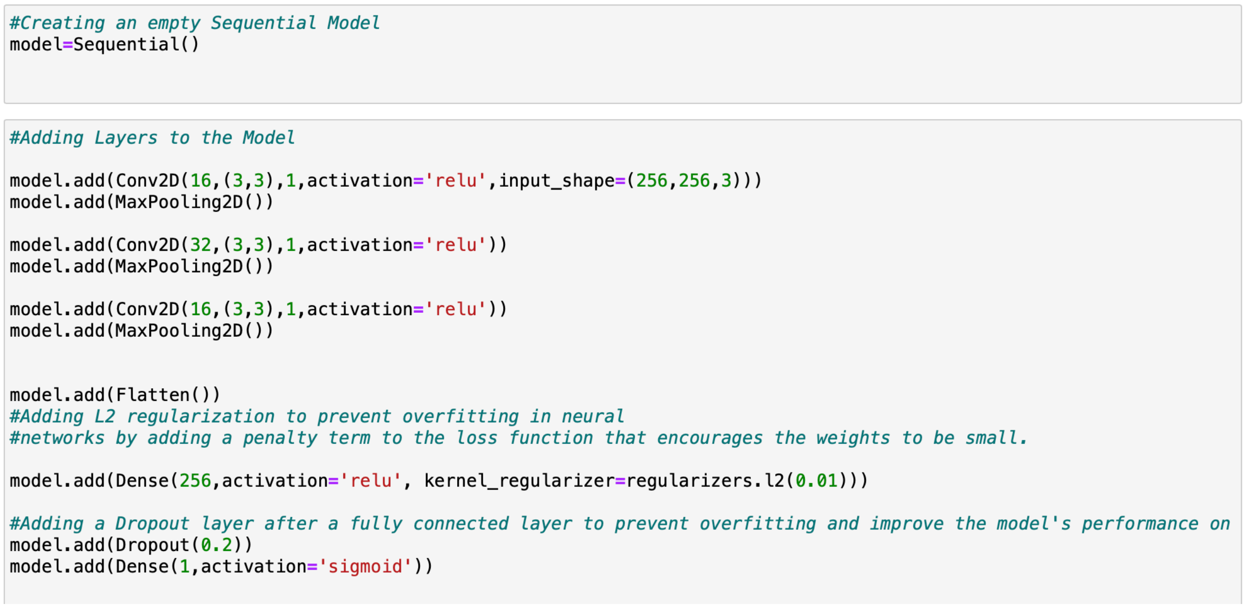
- Dense (256 units, ReLU activation): The Dense layer is a fully connected layer where each neuron is connected to every neuron in the previous layer. The layer has 256 units, which means it produces an output of shape (None, 256). The ReLU activation function is applied to the output of this layer, which introduces non-linearity into the network. ReLU activation sets all negative values to zero and keeps the positive values unchanged. This activation function enables the introduction of non-linear characteristics, enabling the neural network to grasp intricate connections among features.

    - Dropout (dropout rate of 0.2): The Dropout layer was used to mitigate overfitting, which is a common issue in deep learning models. During training, Dropout is a technique that introduces randomness by randomly deactivating a portion of input units during each update in a neural network. By doing so, it effectively simulates the dropout of certain neurons. The primary purpose of this approach is to encourage the network to develop more resilient and general representations by avoiding overreliance on specific sets of features. In this scenario, the Dropout layer is placed after the Dense layer, and it operates with a dropout rate of 0.2, implying that 20% of the input units will be randomly zeroed out while the model is being trained.

    - By adding the Dense layer with ReLU activation and the Dropout layer, the model introduces additional non-linearity and regularization techniques to improve its generalization ability and reduce overfitting. The Dense layer with ReLU activation enables the network to learn more complex representations, and the Dropout layer helps in preventing the model from memorizing the training data too closely, leading to better performance on unseen data.

5. Lastly, a Dense layer with 1 unit and a sigmoid activation function facilitated binary classification.
   - The final layer of the model architecture consists of a Dense layer with 1 unit and a sigmoid activation function. This layer facilitates binary classification, indicating the model's output is a probability value between 0 and 1.

   - Dense (1-unit, sigmoid activation): The Dense layer has 1 unit, representing the modal's final output. In binary classification tasks, this outputs unit typically indicates the probability of fitting to one of the two classes. The sigmoid activation function, also known as the logistic function, is applied to the output of this layer. It squashes the output values between 0 and 1, interpreting them as probabilities. The sigmoid activation is commonly used in binary classification problems as it allows the model to provide a probability estimate for the positive class.
   - By employing a Dense layer with a single unit and applying the sigmoid activation function as the final layer, the model is configured to perform binary classification. The output value, after passing through the sigmoid activation, can be interpreted as the predicted probability of belonging to the positive class. Based on a chosen threshold (e.g., 0.5), the model can make a binary decision by classifying samples with probabilities above the limit as positive and those below as negative.

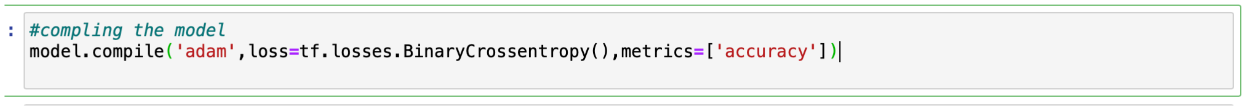


```
#Creating an empty Sequential Model
model=Sequential()
```

```
#Adding Layers to the Model

model.add(Conv2D(16,(3,3),1,activation='relu',input_shape=(256,256,3)))
model.add(MaxPooling2D())

model.add(Conv2D(32,(3,3),1,activation='relu'))
model.add(MaxPooling2D())

model.add(Conv2D(16,(3,3),1,activation='relu'))
model.add(MaxPooling2D())


model.add(Flatten())
#Adding L2 regularization to prevent overfitting in neural
#networks by adding a penalty term to the loss function that encourages the weights to be small.

model.add(Dense(256,activation='relu', kernel_regularizer=regularizers.l2(0.01)))

#Adding a Dropout layer after a fully connected layer to prevent overfitting and improve the model's performance on
model.add(Dropout(0.2))
model.add(Dense(1,activation='sigmoid'))
```

Figure 3.12: Layers of a CNN model (Ramadhani,2024).

After adding layers to a model, the model in then compiled so that it can further be trained as shown in Figure 3.12.



```
#compling the model
model.compile('adam',loss=tf.losses.BinaryCrossentropy(),metrics=['accuracy'])
```

Figure 3.13: Model compilation (Ramadhani,2024).

Figure 3.13 shows the parameters used for compilation of the model and the following is the description of the parameters used.,

- Optimizer: 'adam'

  The Adam optimizer is a popular optimization algorithm commonly used in deep learning. It combines the advantages of two other optimization methods, AdaGrad and RMSProp. Adam adapts the learning rate for each parameter based on their previous gradients, making it effective in training models with large and complex datasets. The 'adam' argument specifies the use of this optimizer.

- Loss Function: tf.losses.BinaryCrossentropy()

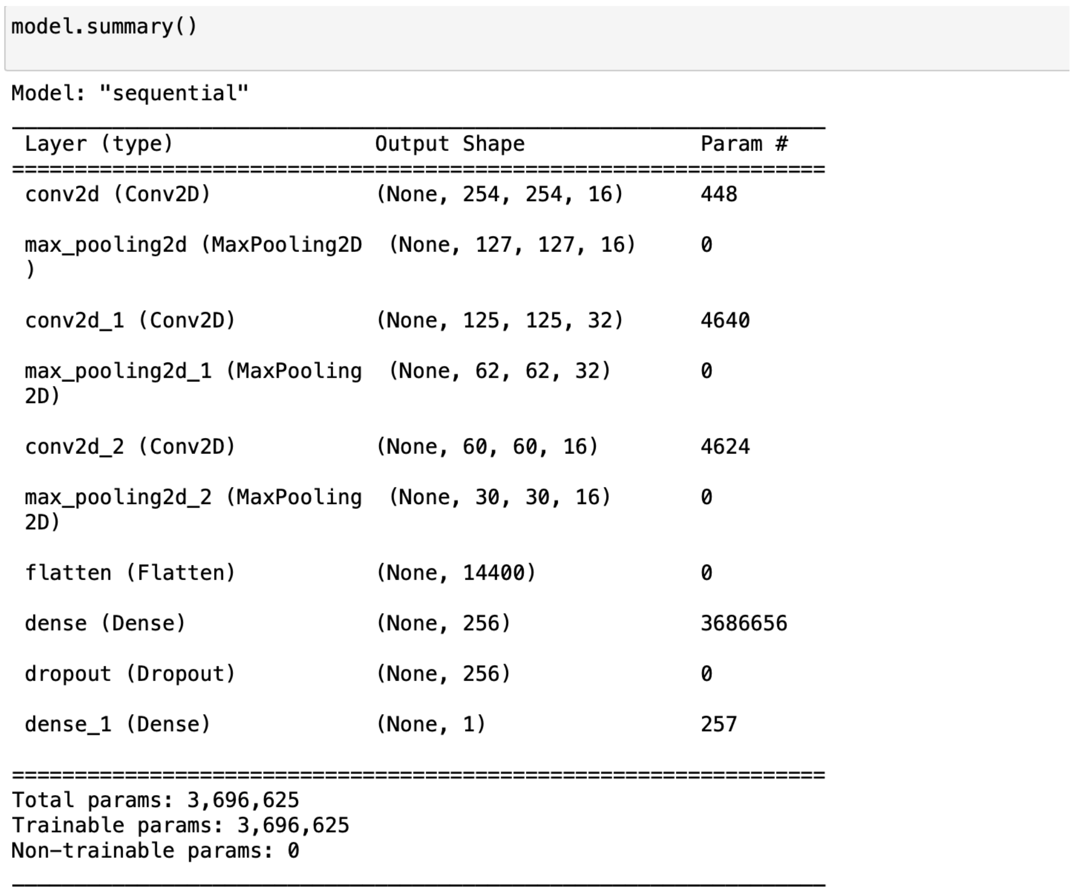
The BinaryCrossentropy loss function is suitable for binary classification problems, where the task is to predict between two classes. It measures the difference between the predicted probabilities and the true labels. The tf.losses.BinaryCrossentropy() function calculates the loss by comparing the predicted values with the true labels. The goal of training the model is to minimize this loss function (errors).

- Metrics: ['accuracy']

The 'accuracy' metric is commonly used to evaluate the performance of classification models. It calculates the accuracy of the model's predictions, which is the ratio of correctly predicted samples to the total number of samples. By specifying ['accuracy'], the model will track and report the accuracy metric during training and evaluation.

When the model is compiled with these configurations, it is ready for training using the specified optimizer, loss function, and metrics.

Table 3.1: The model architecture summary



```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 16)      448

 max_pooling2d (MaxPooling2D  (None, 127, 127, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 16)        4624

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 16)       0
 2D)

 flatten (Flatten)           (None, 14400)             0

 dense (Dense)               (None, 256)               3686656

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 3,696,625
Trainable params: 3,696,625
Non-trainable params: 0
_________________________________________________________________
```

Table 3.1 provides architecture summary of a Convolutional Neural Network (CNN) model with specific layer types, output shapes, and the number of parameters for each layer. The model consists of several convolutional and max-pooling layers, followed by a flatten layer and two dense layers with dropout.

The first convolutional layer (Conv2D) outputs feature maps of size (None, 254, 254, 16) and has 448 trainable parameters. Subsequently, a max-pooling layer (MaxPooling2D) reduces the spatial dimensions to (None, 127, 127, 16) with no additional parameters.

The second convolutional layer (Conv2D_1) generates feature maps of size (None, 127, 125, 32) with 4640 trainable parameters. Another max-pooling layer (max_pooling2d_1) reduces the dimensions to (None, 62, 62, 32) without any parameters.

The third convolutional layer (conv2d_2) generates feature maps of size (None, 60, 60, 16) with 4624 trainable parameters. A max-pooling layer (max_pooling2d_2) further reduces the dimensions to (None, 30, 30, 16) with no parameters.

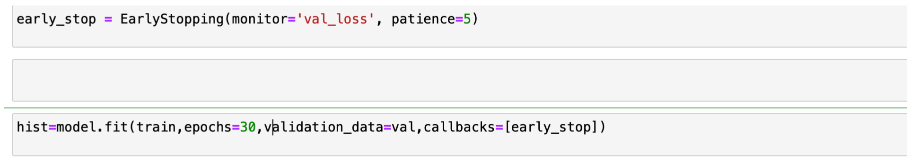
Then, a flatten layer converts the 3D feature maps into a 1D vector of size (None, 14400) before passing the data to the first dense layer (Dense). The dense layer outputs (None, 256) neurons with 3,686,656 trainable parameters. A dropout layer follows the dense layer, which acts as a regularization technique and has no additional parameters.

Finally, the last dense layer (dense_1) produces a single output neuron, aiming for binary classification (None, 1) with 257 trainable parameters.

The total number of model's parameters is 3,696,625, all of which are trainable, as there are no non-trainable parameters in this architecture.

3.9.5 Fifth, Training and validation of the model

The model was trained using fit() function, with some additional parameters and a callback for early stopping as shown in Figure 3.14.



```
early_stop = EarlyStopping(monitor='val_loss', patience=5)
```

```
hist=model.fit(train,epochs=30,validation_data=val,callbacks=[early_stop])
```

Figure 3.14: Model training and validation process (Ramadhani,2024).

Figure 3.14 shows the code snippet of the model's training process. The following is an explanation of what each component does:-

- Early Stopping callback:

The Early Stopping callback is used to monitor a specified metric during training and stop the training process if the metric does not improve after a certain number of epochs. In this case, the monitored metric is 'val_loss', which refers to the validation loss. The patience parameter was set to 5, indicating that training has to stop if the validation loss does not improve for 5 consecutive epochs.

- Training data: train

The 'train' variable represents the training data that has been used to train the model. It has been in the form of input features and corresponding target labels.

- Number of epochs: epochs=30

The 'epochs' parameter specifies the number of times the model has iterate over the entire training dataset during training. In this case, the model has been trained for 30 epochs. This value was chosen after several testing of different numbers of epochs.

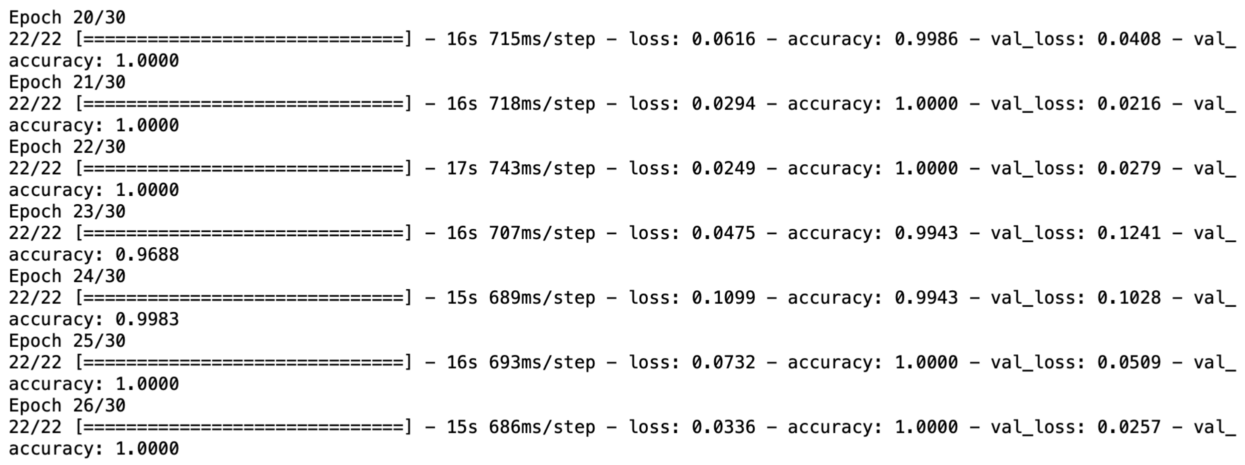
- Validation data: validation_data=val

The 'val' variable represents the validation data that has been used to evaluate the model's performance during training. It should be in the same format as the training data, with input features and corresponding target labels.

- Callbacks: callbacks=[early_stop]

The 'callbacks' parameter allows to specify a list of callbacks to apply during training. In this case, the early_stop is included to monitor the validation loss and stop training if the metric does not improve for five (5) successive epochs.

By calling model.fit () with these parameters and the specified data, the model will be trained using the training data, evaluated on the validation data, and the training will stop early if the model's validation loss does not improve for 5 successive epochs. The training history will be stored in the 'hist' variable, which can be used for further analysis or visualization of the training progress as shown in Figure 3.15.



```
Epoch 20/30
22/22 [==============================] - 16s 715ms/step - loss: 0.0616 - accuracy: 0.9986 - val_loss: 0.0408 - val_
accuracy: 1.0000
Epoch 21/30
22/22 [==============================] - 16s 718ms/step - loss: 0.0294 - accuracy: 1.0000 - val_loss: 0.0216 - val_
accuracy: 1.0000
Epoch 22/30
22/22 [==============================] - 17s 743ms/step - loss: 0.0249 - accuracy: 1.0000 - val_loss: 0.0279 - val_
accuracy: 1.0000
Epoch 23/30
22/22 [==============================] - 16s 707ms/step - loss: 0.0475 - accuracy: 0.9943 - val_loss: 0.1241 - val_
accuracy: 0.9688
Epoch 24/30
22/22 [==============================] - 15s 689ms/step - loss: 0.1099 - accuracy: 0.9943 - val_loss: 0.1028 - val_
accuracy: 0.9983
Epoch 25/30
22/22 [==============================] - 16s 693ms/step - loss: 0.0732 - accuracy: 1.0000 - val_loss: 0.0509 - val_
accuracy: 1.0000
Epoch 26/30
22/22 [==============================] - 15s 686ms/step - loss: 0.0336 - accuracy: 1.0000 - val_loss: 0.0257 - val_
accuracy: 1.0000
```

Figure 3.15: The last seven training progress of the model for each epoch (Ramadhani,2024).

3.10 Evaluation Metrics

Using Accuracy and F1 score as evaluation metrics in a breast cancer prediction model is essential for a comprehensive assessment of its performance. Accuracy offers an overall measure of correct predictions, providing insight into the model's correctness in classifying both malignant and benign cases. However, in the context of breast cancer prediction where class imbalances can occur, F1 score assumes critical importance by considering both precision and recall. It ensures that the model not only accurately classifies cases but also effectively captures positive instances, mitigating the risks associated with false negatives and false positives. By combining Accuracy and F1 score, the evaluation process gains a more holistic view of the model's ability to balance precise identification of cancer cases while minimizing misclassifications, thereby enhancing the model's clinical relevance and aiding informed medical decisions.

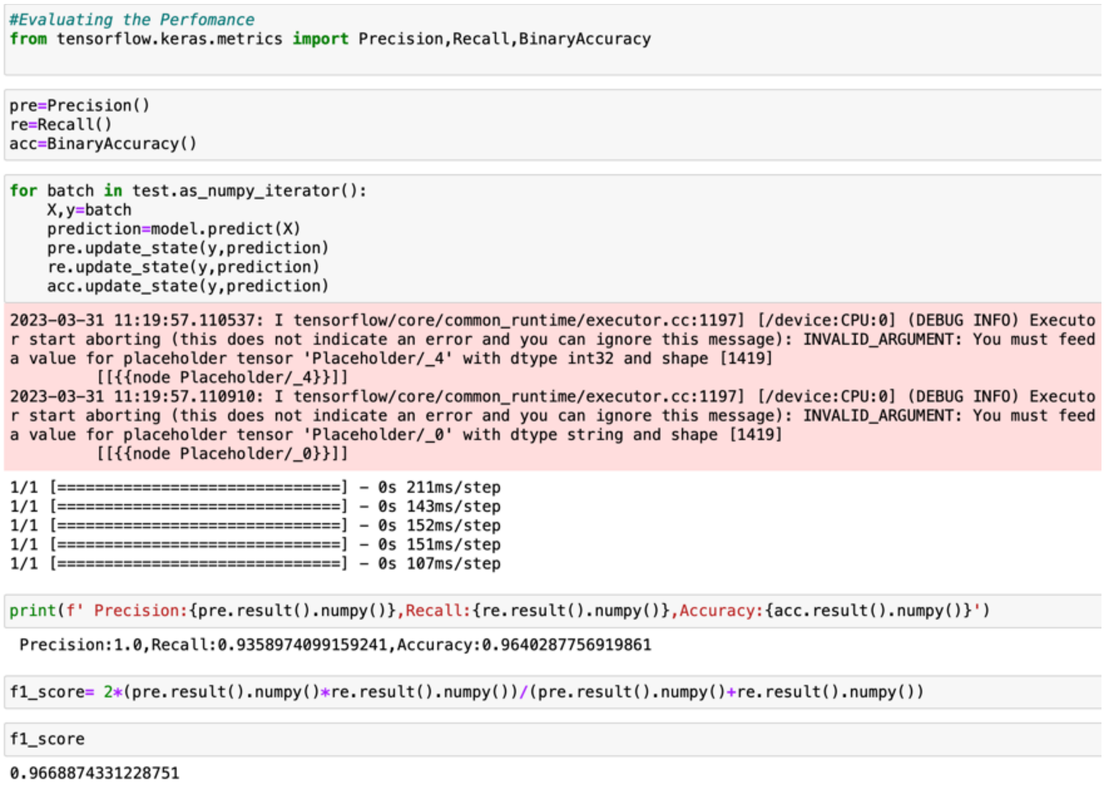
The model was evaluated using the following performance metrics:

- **Accuracy**: Accuracy measures the overall accuracy of a model's predictions by determining the percentage of correctly classified instances among all instances. In this study the developed CNN model achieved an accuracy of 96.4%, indicating that it correctly classified 96.40% of the MRI images. This high accuracy score suggests that the model performs well in distinguishing between different classes.

$$\boldsymbol{Accuracy} = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} x100 \qquad (3.1)$$

- **F1 score**: The F1 score is a measure that combines precision and recall into a single metric, providing a balanced assessment of the performance of the model. The F1 score takes into account both false positives and false negatives. In this study, the developed model achieved an F1 score of 96.69 %, indicating a high balance between precision and recall. This score suggests the model has a good balance between correctly identifying positive cases and minimizing false positives and false negatives.

$$\boldsymbol{F1 - Score} = \frac{2*(Precision*Recall)}{Precision+Recall} \tag{3.2}$$



```
#Evaluating the Perfomance
from tensorflow.keras.metrics import Precision,Recall,BinaryAccuracy
```

```
pre=Precision()
re=Recall()
acc=BinaryAccuracy()
```

```
for batch in test.as_numpy_iterator():
    X,y=batch
    prediction=model.predict(X)
    pre.update_state(y,prediction)
    re.update_state(y,prediction)
    acc.update_state(y,prediction)
```

```
2023-03-31 11:19:57.110537: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executo
r start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed
a value for placeholder tensor 'Placeholder/_4' with dtype int32 and shape [1419]
	 [[{{node Placeholder/_4}}]]
2023-03-31 11:19:57.110910: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executo
r start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed
a value for placeholder tensor 'Placeholder/_0' with dtype string and shape [1419]
	 [[{{node Placeholder/_0}}]]
1/1 [==============================] - 0s 211ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 107ms/step
```

```
print(f' Precision:{pre.result().numpy()},Recall:{re.result().numpy()},Accuracy:{acc.result().numpy()}')
```

```
 Precision:1.0,Recall:0.9358974099159241,Accuracy:0.9640287756919861
```

```
f1_score= 2*(pre.result().numpy()*re.result().numpy())/(pre.result().numpy()+re.result().numpy())
```

```
f1_score
```

```
0.9668874331228751
```

Figure 3.16: Code snippet of how the performance metrics was calculated (Ramadhani,2024)

Figure 3.16 shows the code snippet that has been used to imports three performance metrics from TensorFlow's Keras library: Precision, Recall, and BinaryAccuracy. The following is a breakdown of the code and what it does:

from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy: This line imports the required performance metrics from the tensorflow.keras.metrics module. These metrics are used to evaluate the performance of classification models.

pre = precision(): This line creates an instance of the Precision metric. This measures the proportion of correctly predicted positive cases out of the total cases predicted as positive.

re = recall(): This line creates an instance of the Recall metric. Recall measures the proportion of correctly predicted positive instances out of the total actual positive instances.

acc = BinaryAccuracy(): This line creates an instance of the BinaryAccuracy metric. BinaryAccuracy calculates the accuracy of binary classification models. It measures the proportion of correctly predicted instances (both true positives and true negatives) out of the total number of instances.

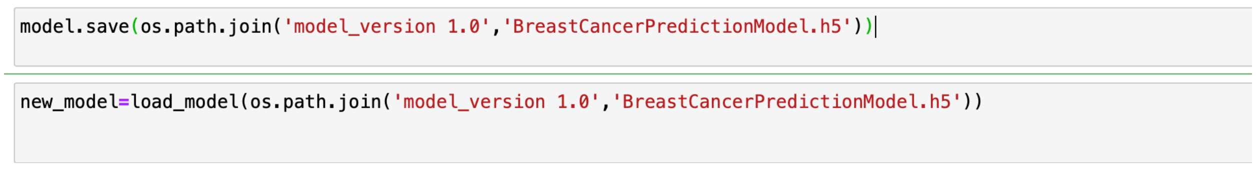
print(f'Precision:{pre.result().numpy()},Recall:{re.result().numpy()},Accuracy:{acc. result().numpy()}'): This line prints the current results of the metrics. By calling result() on each metric and converting it to a NumPy array using numpy(), so as to obtain the current value of each metric.

3.10.1 Additionally, Saving the model in HDF5 format using Kera's save function.

The trained model was saved as an HDF5 file using the .h5 file extension. The HDF5 file format is commonly used to store and manage large amounts of numerical data, including machine learning models.

By saving the model as an HDF5 file, it can be easily loaded and reused later without having to retrain it from scratch. The HDF5 file contains all the necessary information about the model architecture, weights, optimizer configuration, and training parameters, allowing you to restore the model and make predictions on new supplied data.

To load the saved model from the HDF5 file, the load_model function from the Keras library, as shown in the code snippet in Figure 3.17.



```
model.save(os.path.join('model_version 1.0','BreastCancerPredictionModel.h5'))
```

```
new_model=load_model(os.path.join('model_version 1.0','BreastCancerPredictionModel.h5'))
```

Figure 3.17: How the model was saved as HDF5 file (Ramadhani,2024).

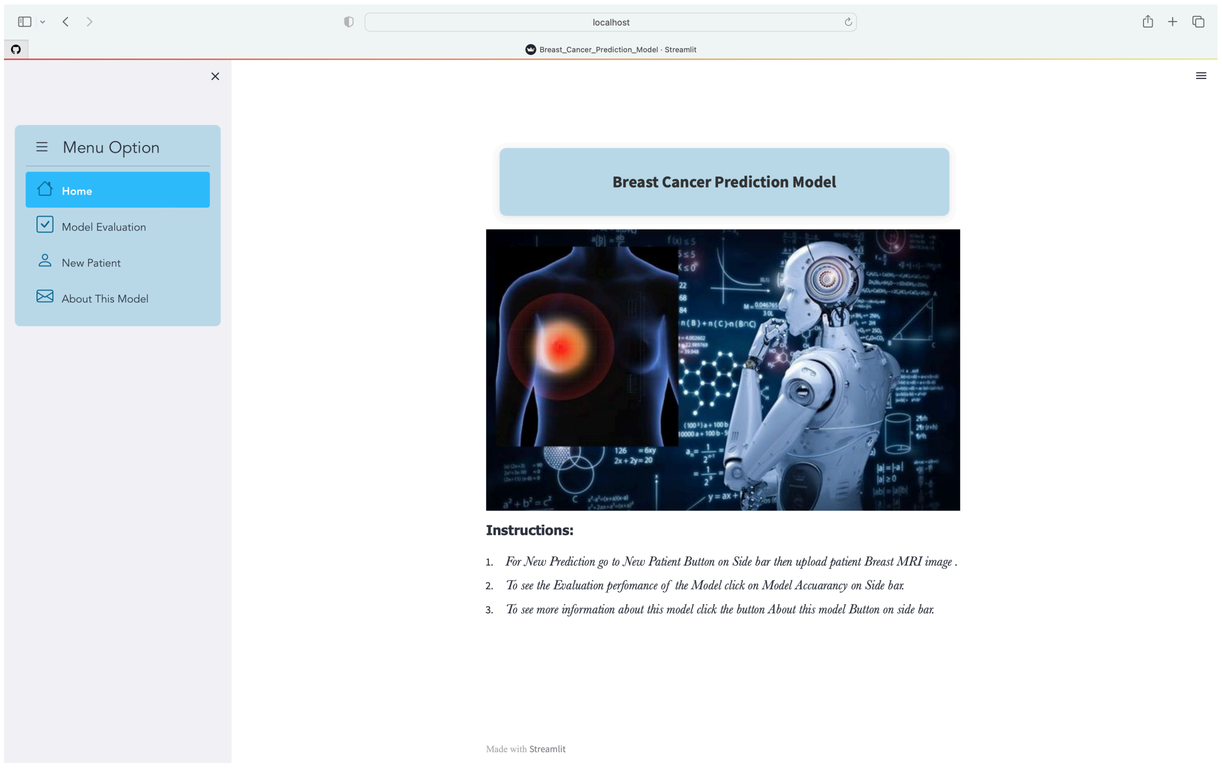
By calling `model.save()`, the developed model is saved as an HDF5 file (`BreastCancerPredictionModel.h5`) in the specified directory (`model_version 1.0`). Later, the `load_model` function is used to load the saved model from the same directory and assign it to the `new_model` variable for further use, as shown in Figure 4.15

3.10.2 Lastly, Integrating the model with a streamlit dashboard.

Furthermore, the developed model was integrated with a dashboard made using streamlit and the source code file named to **Breast_Cancer_Prediction_Model.py**. Streamlit is an open-source Python library that allows to create and deploy custom web applications for machine learning and data science projects. It simplifies the process of building interactive web-based interfaces by providing an intuitive and user-friendly way to design and develop applications.

Streamlit, can quickly convert Python scripts, models, and data visualizations into interactive web apps. It provides a high-level API that allows user to create and customize UI elements such as buttons, sliders, dropdowns, and plots. Streamlit enables writing of code that can handle user inputs, process data, and update the app dynamically.

Streamlit is mostly used in the data science and machine learning community because of its simplicity and ease of use. It promotes a smooth workflow for quickly prototyping and sharing ideas, making it a popular choice for building interactive data applications and creating demos for ML models. To run the model's dashboard, we run the command **streamlit run Breast_Cancer_Prediction_Model.py** on the terminal after navigate to the directory where the Breast_Cancer_Prediction_Model.py file is located.



Breast Cancer Prediction Model

Instructions:

1. *For New Prediction go to New Patient Button on Side bar then upload patient Breast MRI image .*
2. *To see the Evaluation perfomance of the Model click on Model Accuarancy on Side bar.*
3. *To see more information about this model click the button About this model Button on side bar.*

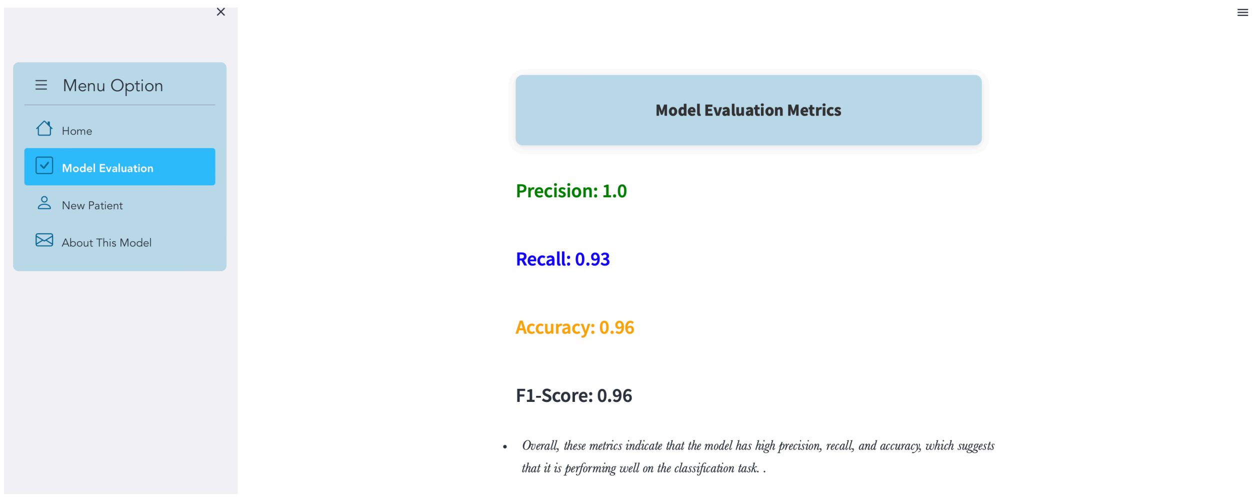
Made with Streamlit

Figure 3.18: A home page of an integrated model with streamlit (Ramadhani,2024).

Figure 3.18 shows the dashboard of the model developed using Streamlit. It has the following menu options:

Home: This option displays the home page, which serves as an introduction to the application. It provides users with an overview of the model and giving them a context of what the model does and how to use it.

Model Evaluation: Selecting this option presents users with the values of evaluation metrics such as accuracy, recall, precision, and F1 score. These metrics provide an assessment of the performance of the developed model and can help users understand its effectiveness as shown in Figure 3.19.



Model Evaluation Metrics

Precision: 1.0

Recall: 0.93

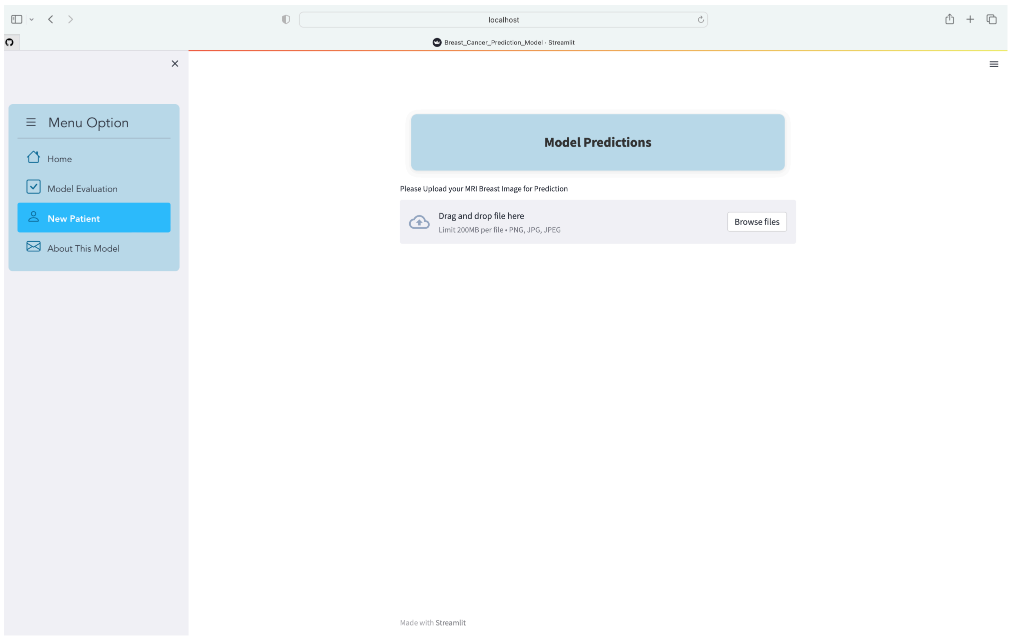
Accuracy: 0.96

F1-Score: 0.96

- *Overall, these metrics indicate that the model has high precision, recall, and accuracy, which suggests that it is performing well on the classification task. .*

Figure 3.19: Model's evaluation metrics (Ramadhani,2024).

New Patient: This button give user a window where user can upload a new MRI image for prediction. By clicking on this option, users can access the place to select a new MRI image file, which will be processed by the model and make predictions related to either the image provided belong to benign or malignant class . It provides a means for users to interact with the model and receive predictions for new patients as shown in Figure 3.20 , Figure 3.21 and Figure 3.22.



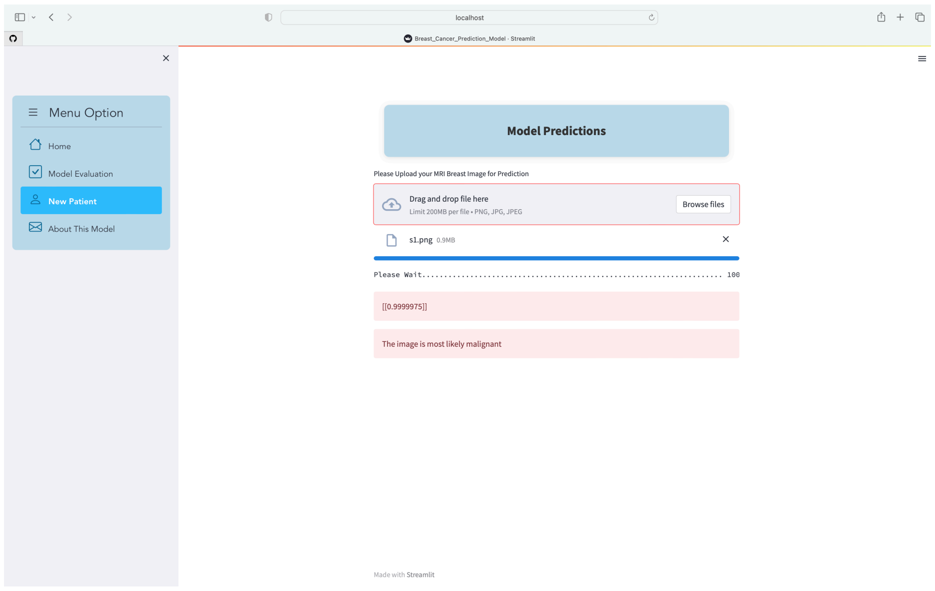
Model Predictions

Please Upload your MRI Breast Image for Prediction

Drag and drop file here

Limit 200MB per file • PNG, JPG, JPEG

Figure 3.20: Page for new prediction of a new MRI image (Ramadhani,2024).



Please Upload your MRI Breast Image for Prediction

Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG

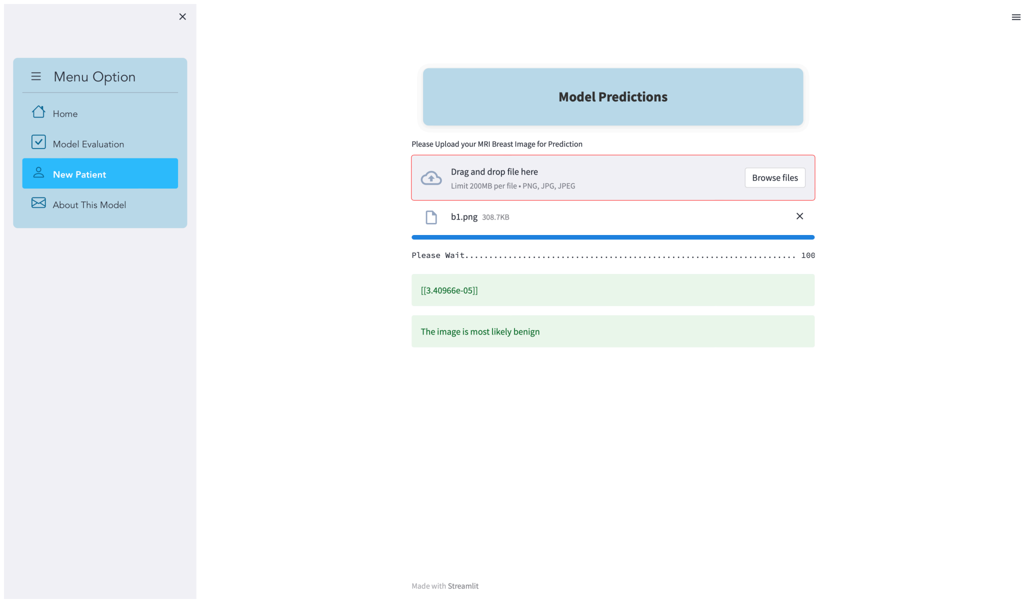
s1.png 0.9MB

```
Please Wait................................................................ 100
```

[[0.9999975]]

The image is most likely malignant

Figure 3.21: Predicted outcome from a new MRI image, i.e., the probability outcome (0.999) and the class of an MRI image (Malignant) (Ramadhani,2024).



Please Upload your MRI Breast Image for Prediction

Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG

b1.png 308.7KB

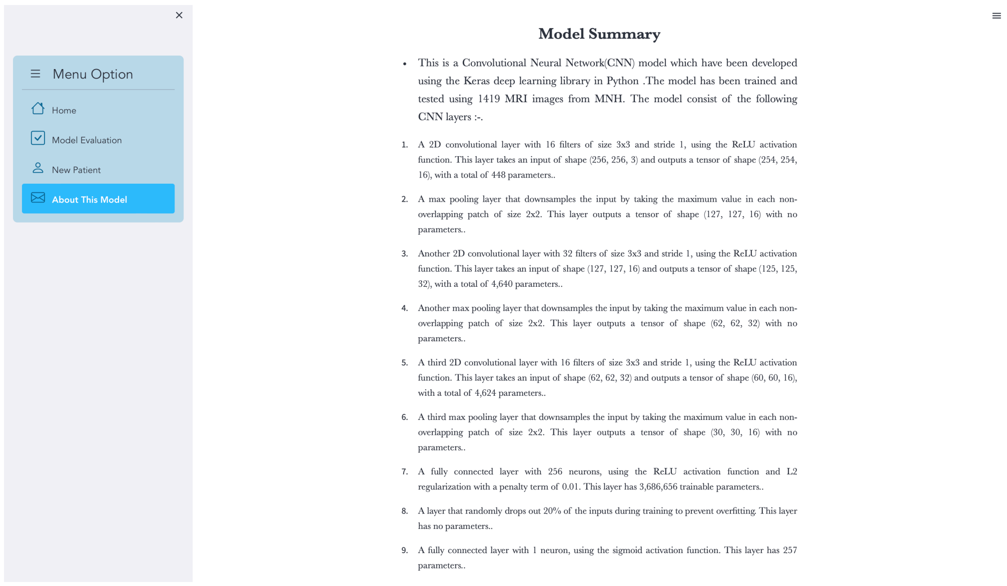
```
Please Wait................................................................ 100
```

[[3.40966e-05]]

The image is most likely benign

Figure 3.22: Predicted outcome from a new MRI image, i.e., the probability outcome (0.00034) and the class of an MRI image (Benign) (Ramadhani,2024).

About This Model: This button provides information about how the model was built. It could include details about the architecture, training data, preprocessing steps, and any other relevant information that would help users understand the underlying model and its development process as shown in Figure 3.23.



Model Summary

- This is a Convolutional Neural Network(CNN) model which have been developed using the Keras deep learning library in Python .The model has been trained and tested using 1419 MRI images from MNH. The model consist of the following CNN layers :-.

1. A 2D convolutional layer with 16 filters of size 3x3 and stride 1, using the ReLU activation function. This layer takes an input of shape (256, 256, 3) and outputs a tensor of shape (254, 254, 16), with a total of 448 parameters..
2. A max pooling layer that downsamples the input by taking the maximum value in each non-overlapping patch of size 2x2. This layer outputs a tensor of shape (127, 127, 16) with no parameters..
3. Another 2D convolutional layer with 32 filters of size 3x3 and stride 1, using the ReLU activation function. This layer takes an input of shape (127, 127, 16) and outputs a tensor of shape (125, 125, 32), with a total of 4,640 parameters..
4. Another max pooling layer that downsamples the input by taking the maximum value in each non-overlapping patch of size 2x2. This layer outputs a tensor of shape (62, 62, 32) with no parameters..
5. A third 2D convolutional layer with 16 filters of size 3x3 and stride 1, using the ReLU activation function. This layer takes an input of shape (62, 62, 32) and outputs a tensor of shape (60, 60, 16), with a total of 4,624 parameters..
6. A third max pooling layer that downsamples the input by taking the maximum value in each non-overlapping patch of size 2x2. This layer outputs a tensor of shape (30, 30, 16) with no parameters..
7. A fully connected layer with 256 neurons, using the ReLU activation function and L2 regularization with a penalty term of 0.01. This layer has 3,686,656 trainable parameters..
8. A layer that randomly drops out 20% of the inputs during training to prevent overfitting. This layer has no parameters..
9. A fully connected layer with 1 neuron, using the sigmoid activation function. This layer has 257 parameters..

Figure 3.23: About the model(Ramadhani,2024).

Overall, the developed streamlit dashboard offers a user-friendly interface with distinct menu options. It provides an introduction, showcases model evaluation metrics, allows users to upload new MRI images for prediction, and shares information about the model's construction. This way, users can navigate through the different sections, understand the model's performance, interact with it, and gain insights into its development.

CHAPTER FOUR

RESEARCH FINDINDS AND DISCUSSION

4.1 Overview

This chapter presents and discusses the research findings. The evaluation metrics used to assess the performance of the model include accuracy and F1 score. The purpose of this chapter is to provide an in-depth analysis of the outcomes and implications of this study.

4.2 Discussion

The achieved evaluation metrics demonstrate the efficacy of the CNN model in classifying Breast MRI images.

The high accuracy score of 96.4% indicates that the model made correct predictions for a significant portion of the MRI images. This demonstrates the model's capability to effectively classify between different classes and make accurate classifications.

The F1 score of 96.69 % suggests that the model strikes a good balance between precision and recall. This balance is important to avoid overly biased predictions towards either false positives or false negatives.

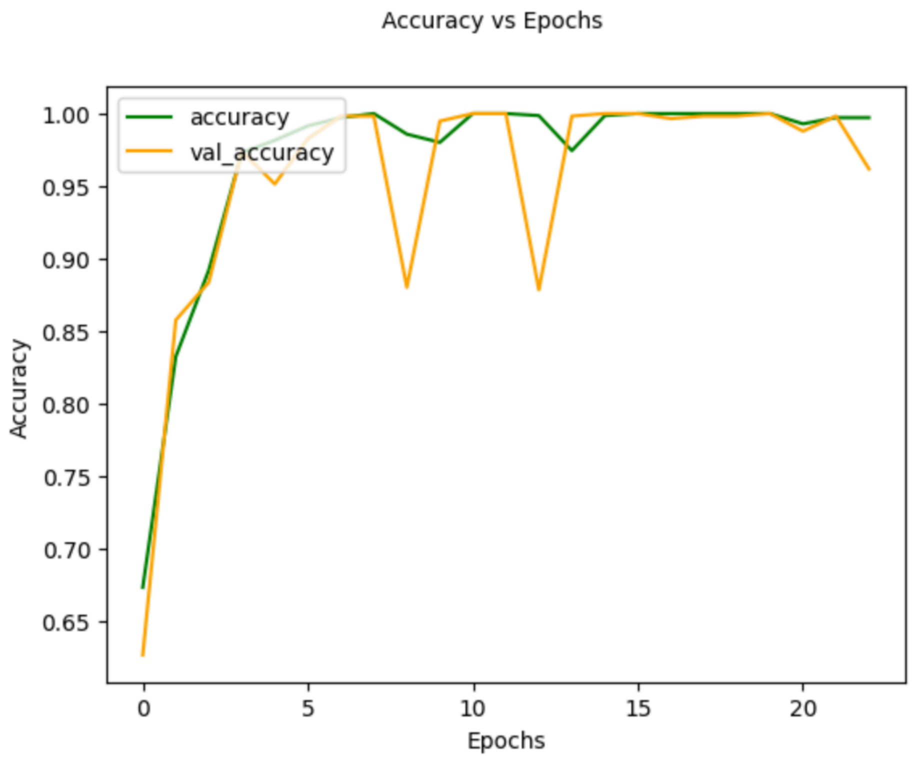


Figure 4.1: A plot of accuracy against epochs (Ramadhani,2024).

In Figure 4.1, the accuracy of the model is represented on the y-axis, while the number of epochs is depicted on the x-axis. The plot shows how the accuracy of the model changes as the number of training epochs increases.

Initially, at the beginning of training, the accuracy was low as the model was still learning and adjusting its parameters. However, as the epochs progress, the accuracy tends to improve. This improvement indicates that the model was getting better at correctly classifying breast cancer cases based on the MRI images,(Zhao et al., 2018).

The plot demonstrates an increasing trend in accuracy with each epoch, indicating that the model's performance was improving over time. The rate at which the accuracy increases vary, with larger improvements in the earlier epochs and possibly slower progress as training continues.

The increasing trend of accuracy with epoch was encouraging sign, suggesting that the model is effectively learning the patterns and features in the MRI images that are indicative of breast cancer. The model becomes more confident in its predictions as it is exposed to more training data and learns from it.

The plot of accuracy vs epoch provides insights into the model's learning progress and can help determine when to stop training to avoid overfitting. It also serves as a visual representation of the performance of the model over time, showcasing the increasing assurance and reliability of the breast cancer detection model as the number of epochs increases (Zhao et al., 2018).

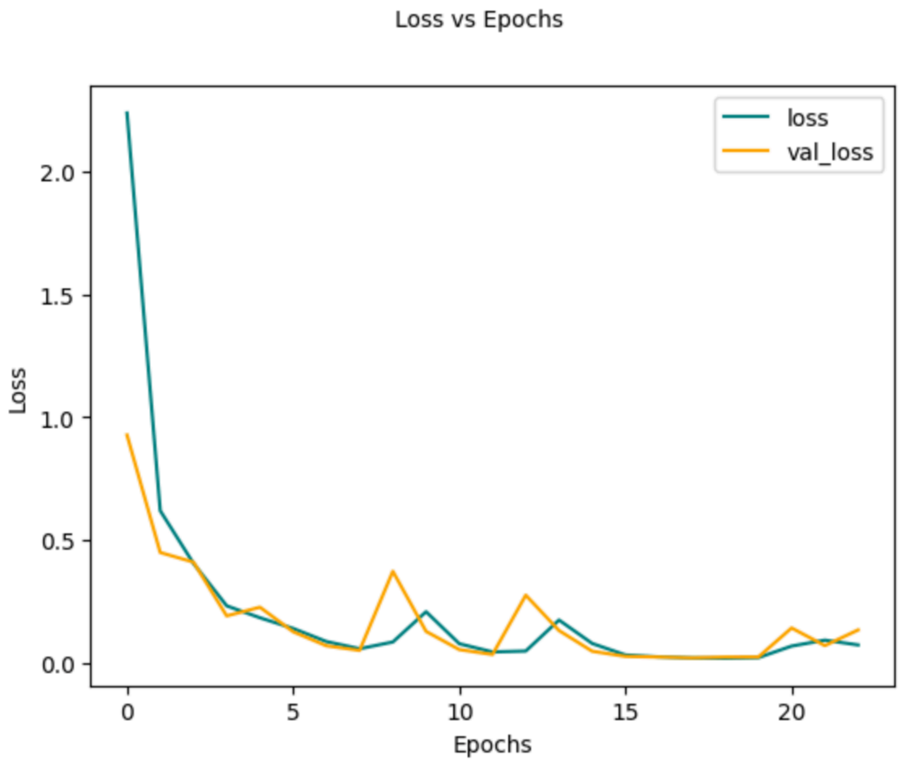


Figure 4.2: Plot of a loss against epochs (Ramadhani,2024).

Figure 4.2,shows a plot of loss vs. epoch for the CNN model used in breast cancer detection using MRI images shows that the loss of the model is represented on the y-axis, while the number of epochs is depicted on the x-axis. The plot illustrates how the loss, which is a measure of the model's prediction error, changes as the number of training epochs increases.

At the beginning of training, the value of the loss was relatively high as the model's initial predictions are likely to be far from the true labels. However, as the epochs progress and the model learn from the training data, the loss gradually decreases.

The decreasing trend in the loss indicates that the model was becoming more accurate in its predictions over time. As the model adjusts its parameters through optimization algorithms, it minimizes the difference between its predictions and the true labels. Consequently, the loss value decreases, signifying improved performance in capturing the patterns and features associated with breast cancer in the MRI images, (Zhao et al., 2018).

Ideally, the loss should decrease steadily with each epoch. However, it is important to monitor the plot closely for any irregularities or fluctuations in the loss curve. Sharp spikes or sudden increases in the loss may indicate that the model is overfitting to the training data or encountering other issues. On the other hand, a loss curve that plateaus or levels off may suggest that the model has reached its optimal performance and further training might not significantly improve the results.

The plot of loss vs epoch provides valuable insights into the learning progress of the CNN model. It helps assess the effectiveness of the model in minimizing errors and refining its predictions over time. By observing the decreasing trend in the loss, we gain confidence in the model's ability to accurately detect breast cancer based on the MRI images ,(Zhao et al., 2018).

Overall, a decreasing loss value with increasing epochs signifies the model's learning capability and its improved performance in detecting breast cancer using MRI images.

Table 5.1: Comparison of the accuracy of this research in relation to prior studies.

RESEARCH	DATASETS USED	ACCURANCY
Yurttakal et al., (2020)	Breast MRI images of 200 Cases among them, 98 are benign and 102 malignant from (Turkey)	97.5%
Zuluaga-Gomez et al., (2021)	1120 thermal images from DMR-IR Database (UK)	92%
Alanazi et al., (2021)	275,000, 50 ×50-pixel RGB image patches from Kaggle	87%
Ragab et al., (2019)	The digital database for screening mammography (DDSM) dataset comprises 2,620 cases, while the Curated Breast Imaging Subset of DDSM (CBIS-DDSM) dataset includes 753 cases of microcalcifications and 891 cases of masses.	73.6%
Zhao et al.,(2018)	122 digital mammogram images in which 54 malignant cases and 68 benign cases obtained from the Mammography Image Analysis Society (MIAS) database	97%
This study (Ramadhani Mrisho,2024)	Uses a total of 1419 MRI images. The datasets consist of 700 malignant and 719 benign.	96.4%

In Table 5.1, Yurttakal et al. (2020) achieved an accuracy of 97.57% using breast MRI images from Turkey, analyzed in the MATLAB environment.

Zhao et al. (2018) obtained an accuracy of 97% using digital mammogram images from the MIAS database.

Zuluaga-Gomez et al. (2021) obtained an accuracy of 92% by employing thermal images from the DMR-IR Database.

Alanazi et al. (2021) obtained an accuracy of 87% using 275,000 RGB image patches from Kaggle.

Lastly Ragab et al. (2019) achieved the lowest accuracy of 73.6% using the DDSM and CBIS-DDSM datasets in the MATLAB environment.

This study achieved an accuracy of 96.4% using a dataset of 1,419 MRI images, including both malignant and benign cases, indicating that the developed model can perform better in breast cancer prediction in our local health systems. Simply the developed model has also achieved high accuracy precision, recall as well as F1-score .

4.3 Summary

4.3.1 Objective 1

To extract features in magnetic resonance images (MRI) that are used in detecting breast cancer using CNN.

This objective was accomplished by employing CNN layers, as shown in Chapter 3 in Section 3.9. Thus, Convolutional Neural Networks (CNNs) are capable of extracting crucial features from breast images for predicting breast cancer and differentiating between benign and malignant cases. These features encompass various aspects of the images. Firstly, CNNs capture texture patterns within the breast tissue, detecting irregularities, fine structures, and areas with distinct texture variations. Secondly, they learn to analyze shape and contour information, identifying irregular or asymmetrical shapes typically associated with malignancy, as opposed to regular and symmetrical shapes indicative of benign structures. CNNs also capture spatial relationships between different regions within the breast image, considering the distribution of features like microcalcifications or masses. Additionally, they focus on tumor margins, detecting irregular and spiculated edges associated with malignancy and smoother, better-defined margins typically found in benign masses. Lastly, CNNs learn to distinguish between different tissue densities, recognizing that dense breast tissue is connected to a higher risk of obtaining breast cancer. By extracting these features automatically, CNNs enable accurate breast cancer prediction.

4.3.2 Objective 2

To develop a model which classifies between benign and malignant breast tissues of our local breast images at an early stage using a convolutional neural network algorithm.

In achieving this objective. Python 3.7 was used to build a CNN model using different Python libraries like TensorFlow, Keras, OS, CV2, and others, as shown in Chapter 3 in Section 3.9. The developed model is capable of classifying MRI images between benign and malignant tumors. Additionally, this achievement underscores the importance of thoughtful dataset augmentation and the utilization of cutting-edge technologies in the development of effective neural network algorithms for medical image classification.

4.3.3 Objective 3

To evaluate the performance of the developed model by employing performance metrics such as accuracy and F1 score.

To achieve this objective, The model was assessed using accuracy and the F1 score. The model achieved high accuracy and F1-scores of 96.4%, and 96.6%, respectively, as shown in Chapter 3 in Section 3.10. This metric indicates the model's potential as a valuable tool for breast cancer prognosis.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Overview

This chapter comprises three main sections, namely, the conclusion, recommendations, and limitations of the study. In the conclusion section, the key findings and outcomes of the research are summarized and discussed. The recommendations section provides suggestions and guidance based on the study's results for future actions or improvements. Lastly, the limitations of the study are acknowledged and discussed, highlighting any constraints or challenges encountered during the research process.

5.2 Conclusion

The research was conducted at Muhimbili National Hospital. The main objective of the study was to develop a CNN breast cancer prediction model using Python that could classify between Benign and Malignant tumors based on MRI images.

The study uses 30 MRI images from the Muhimbili National Hospital, consisting of 8 malignant and 22 benign cases. To address the data scarcity, data augmentation techniques such as rotation, shifting, flipping, and shearing were applied using Python, resulting in a dataset of 1419 images, containing 700 benign and 719 malignant cases.

The CNN model architecture included multiple layers such as Conv2D, MaxPooling2D, and Dense layers with appropriate activation functions. The dataset was divided into training, validation, and testing sets. The model achieved impressive evaluation metrics, including high accuracy, recall, precision, and F1 score, indicating its effectiveness in accurately identifying breast cancer cases.

The research findings highlight the potential of CNNs and data augmentation techniques in improving breast cancer identification. Despite the initial limited dataset, data augmentation increased the number of samples and improved model performance. The proposed CNN architecture demonstrated robust feature extraction and classification capabilities.

Further research with larger datasets and diverse populations is recommended to validate and generalize the proposed model. Nevertheless, this study contributes significantly to the field of breast cancer detection by offering an efficient approach using CNNs for early identification.

5.3 Limitation of the study

The study has limitations that should be considered. Firstly, the dataset used in this research is relatively small, consisting of only 30 original MRI images. This limited sample size may not fully represent the variability and complexity of real-world breast cancer cases. Therefore, the generalizability of the performance of the model to larger and more diverse datasets needs to be further investigated.

Additionally, the study focused on using MRI images as the sole modality for breast cancer detection. In real-world clinical settings, multiple imaging modalities includes mammography, and ultrasound results are often used in conjunction for accurate diagnosis. The exclusion of these complementary imaging modalities may limit the model's ability to perform at its full potential.

Furthermore, the research solely focused on binary classification of breast cancer cases into malignant and benign categories. While this provides valuable insights into early detection, the model's performance in distinguishing different subtypes or stages of breast cancer was not explored. Future studies could consider incorporating more detailed classification tasks.

Lastly, the study was conducted using a specific dataset from Muhimbili National Hospital, which may have specific demographic and regional characteristics. The applicability of the model to other populations and healthcare settings should be investigated to assess its generalizability.

5.4 Recommendations

The study has shown promising results in the early identification of breast cancer using CNN. However, there are still several chances for further research that can be explored to improve the accuracy and reliability of the model.

One potential area of future research could be to expand the dataset used in this study to include more diverse images of breast tissue from different populations and demographics[multi-center]. This could help to ensure that the model is effective in identifying breast cancer in a wider range of patients.

Finally, it may be useful to explore the integration of this model into clinical practice and to investigate the potential benefits and limitations of using such a model in real-world settings.

REFERENCES

Al-Haija, Q. A., & Adebanjo, A. (2020). Breast cancer diagnosis in histopathological images using ResNet-50 convolutional neural network. *IEMTRONICS 2020 - International IOT, Electronics and Mechatronics Conference, Proceedings*, *50*. https://doi.org/10.1109/IEMTRONICS51293.2020.9216455

Alanazi, S. A., Kamruzzaman, M. M., Islam Sarker, M. N., Alruwaili, M., Alhwaiti, Y., Alshammari, N., & Siddiqi, M. H. (2021). Boosting Breast Cancer Detection Using Convolutional Neural Network. *Journal of Healthcare Engineering*, *2021*. https://doi.org/10.1155/2021/5528622

A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/ 4824-imagenet-classification-with-deep-convolutional-neural-networks. pdf

Eroğlu, Y., Yildirim, M., & Çinar, A. (2021). Convolutional Neural Networks based classification of breast ultrasonography images by hybrid method with respect to benign, malignant, and normal using mRMR. *Computers in Biology and Medicine*, *133*(April). https://doi.org/10.1016/j.compbiomed.2021.104407

Fonseca, P., Mendoza, J., Wainer, J., Ferrer, J., Pinto, J., Guerrero, J., & Castaneda, B. (2015). Automatic breast density classification using a convolutional neural network architecture search procedure. *Medical Imaging 2015: Computer-Aided Diagnosis*, *9414*, 941428. https://doi.org/10.1117/12.2081576

Khooa, V. S., Dearnaley, D. P., Finniganb, D. J., Padhani, A., Tannerd, S. F., & Leachd, M. (1997). *Magnetic resonance imaging (MRI): considerations and applications in radiotherapy treatment planning*. *42*, 1–15.

Lu, H. C., Loh, E. W., & Huang, S. C. (2019). The Classification of Mammogram Using Convolutional Neural Network with Specific Image Preprocessing for Breast Cancer Detection. *2019 2nd International Conference on Artificial Intelligence and Big Data, ICAIBD 2019*, 9–12. https://doi.org/10.1109/ICAIBD.2019.8837000

Ragab, D. A., Sharkas, M., Marshall, S., & Ren, J. (2019). Breast cancer detection using deep convolutional neural networks and support vector machines. *PeerJ*, *2019*(1), 1–23. https://doi.org/10.7717/peerj.6201

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*. https://doi.org/10.1186/s40537-019-0197-0

Wang, J., & Perez, L. (n.d.). *Learning*.

Yue, W., Wang, Z., Chen, H., Payne, A., & Liu, X. (2018). Machine learning with applications in breast cancer diagnosis and prognosis. *Designs*, *2*(2), 1–17.

https://doi.org/10.3390/designs2020013

Yurttakal, A. H., Erbay, H., İkizceli, T., & Karaçavuş, S. (2020). Detection of breast cancer via deep convolution neural networks using MRI images. *Multimedia Tools and Applications*, *79*(21–22), 15555–15573. https://doi.org/10.1007/s11042-019-7479-6

Zhao, X., Wang, X., & Wang, H. (2018). Classification of Benign and Malignant Breast Mass in Digital Mammograms with Convolutional Neural Networks. *ACM International Conference Proceeding Series*, 47–50. https://doi.org/10.1145/3285996.3286006

Zuluaga-Gomez, J., Al Masry, Z., Benaggoune, K., Meraghni, S., & Zerhouni, N. (2021). A CNN-based methodology for breast cancer diagnosis using thermal images. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging and Visualization*, *9*(2), 131–145. https://doi.org/10.1080/21681163.2020.1824685

APPENDICES

APPENDIX I : Published Paper

International Journal of Computer Applications Technology and Research
Volume 12–Issue 12, 11 - 16, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1212.1003

AI-Driven Mobile Application for Breast Cancer Detection Using CNN for MRI Images in Tanzania

Ramadhani Mrisho Hamis
Eastern Africa Statistical Training Centre
Dar es salaam , Tanzania

Dr. Rogers Bhalalusesa
The Open University of Tanzania
Dar es salaam , Tanzania

Abstract: Breast cancer remains a pressing health concern in Tanzania, characterized by rising incidence rates and consequential mortality. Traditional diagnostic methodologies, encompassing mammography, ultrasound, and biopsy, exhibit limitations concerning accuracy, subjectivity, and accessibility. In recent years, the advent of machine learning techniques, notably Convolutional Neural Networks (CNNs), has offered a promising avenue for breast cancer detection, addressing the deficiencies inherent in conventional approaches. This paper centers on the development of an AI-driven mobile application (Mobile App) integrated with breast cancer prediction model tailored for breast cancer identification in Tanzania, leveraging the capabilities of CNN models. The app will allow radiologists to capture breast MRI images through a mobile phone camera and receive predictions categorizing the captured images as Malignant or Benign, aiding in prompt diagnosis and improve the accuracy of the diagnosis. The developed model uses 30 MRI breast images from Muhimbili National Hospital (MNH). Subsequently, data augmentation techniques were implemented, bolstering the dataset to 1419 images, inclusive of 700 benign and 719 malignant cases. The resultant CNN model developed demonstrated an exceptional accuracy of 96.69%, underscoring its potential effectiveness in discerning breast cancer and its prospects for facilitating early detection within the Tanzanian context.

Keywords: Breast Cancer; Artificial Intelligence(AI); Convolutional Neural Networks (CNNs); Mobile App; Data Augmentation

1. INTRODUCTION

Breast cancer poses a significant and escalating health challenge in Tanzania, manifesting a surge in both the occurrence and fatality rates. It stands as the second most prevalent ailment among Tanzanian women, recording an estimated 3037 new cases and 1303 fatalities in 2018. Alarming projections anticipate an over 120 percent escalation in both incidence and fatality rates by 2040. The World Health Organization (WHO) reports that breast cancer contributes to 23% of cancer-related cases and 14% of cancer-related deaths in women (Breast Cancer Initiative, 2017; Zhao et al., 2018). Conventional diagnostic techniques, encompassing mammography, ultrasound, and biopsy, suffer from limitations in accuracy, subjectivity, and accessibility (Lu et al., 2019). However, the advent of machine learning, notably Convolutional Neural Networks (CNNs), has emerged as a potent alternative for breast cancer detection, transcending the deficiencies of traditional methods (Yue et al., 2018).

Artificial Intelligence (AI) is a branch of computer science that focuses on systems and devices capable of analyzing human intelligence in general It includes technologies that enable computers to simulate intelligent behavior, learn from data, recognize patterns and based on decision information A variety of techniques including machine learning, natural language processing, computer vision, and neural networks are used to simulate cognitive processes. These systems can analyze big data, gain insights, and automatically adapt to new data, enabling them to solve complex problems, predict, and act across industries, and change services ranging from healthcare finance to transportation and beyond.

Convolutional Neural Networks (CNNs) stand as specialized deep learning models tailored for scrutinizing visual data, presenting an ideal framework for tasks involving image classification. Diverging from traditional methodologies, CNNs alleviate the necessity for manual feature crafting by autonomously assimilating intricate patterns and configurations from unprocessed image data. Their efficacy in breast cancer detection surpasses that of traditional feature-centric approaches, showcasing substantial potential. CNNs excel in discriminating between benign and malignant lesions, identifying nuanced characteristics such as microcalcifications or masses, and gauging risk levels with commendable precision (Alanazi et al., 2021).

The structure of a Convolutional Neural Network (CNN) consists of three key components: convolutional layers for extracting features, pooling layers for reducing spatial dimensionality while retaining crucial details, and fully connected layers for classification. By using filters, convolutional layers identify important patterns in input images, while pooling layers preserve vital information. Fully connected layers learn to associate these features with intended output, enabling predictions. CNNs are trained using labeled data and iterative weight adjustments to minimize prediction differences. This training, called backpropagation, refines network parameters using optimization algorithms. Integrating CNNs has revolutionized breast cancer detection by automating feature extraction, enhancing accuracy and efficiency. Utilizing CNN capabilities in Tanzania shows promise for more accurate and accessible breast cancer diagnostic tools, enabling early detection and improved treatment outcomes, potentially reducing mortality rates (Buda et al., 2018; Masud, 2020; Lecun et al., 2015).

This paper centers on the development of a mobile application that integrates a Convolutional Neural Network (CNN) breast cancer prediction model. The app's creation involves the utilization of Flask framework for API development and React Native for mobile application development. Specifically, Flask, a Python-based web framework, serves as the backend technology, enabling the construction of robust APIs. It facilitates the integration of the CNN breast cancer prediction model within the app's architecture, allowing seamless communication between the mobile interface and the predictive model. On the other hand, React Native is a JavaScript-based framework used for front-end mobile app development. It uses a single codebase for both iOS and Android platforms to facilitate the creation of responsive, user-friendly interfaces. It facilitates the creation of a responsive, user-friendly interface by leveraging a single codebase for both iOS and Android platforms. The app, thus, encompasses a harmonious synergy between Flask for backend API development and React Native for the mobile app's frontend, culminating in a comprehensive and efficient solution for breast cancer prediction accessible via mobile devices.

2. METHODOLOGY

The app development commences by employing Flask version 3.0.0 for API development, configuring endpoints, and integrating the pre-existing CNN breast cancer prediction model within the Flask framework. This process involves adapting the model to align with Flask's architecture and establishing seamless data exchange mechanisms between the mobile app and the predictive model. Concurrently, React Native version 0.72 is utilized for frontend mobile application development, focusing on crafting an intuitive user interface (UI) design ensuring consistent user experience (UX) across iOS and Android platforms. The methodology encompasses an iterative development approach, encompassing testing and refinement cycles to validate functionality, usability, and performance. Rigorous testing, evaluation, and optimization phases are undertaken to ensure the integrated mobile application's efficacy, accuracy, and reliability in facilitating breast cancer prediction via mobile devices.

3. MOBILE APP AND API INTEGRATION

3.1 Mobile App development

The mobile application (Mobile App) was developed using React Native version 0.72 within a development environment optimized on a powerful MacBook Pro using VS Code IDE version 1.84.

The development process utilized the React Native framework. Extensive testing procedures were carried out on an Android device operating on OS version 12 to guarantee smooth functionality across various Android platforms. For testing purposes, Expo Go was employed, known for its user-friendly interface and simplified setup, which streamlined the testing phase, hastened feature evaluations, and simplified quality assurance processes. This holistic approach aimed to create a top-tier, cross-platform mobile application with consistent and reliable performance across diverse Android devices.

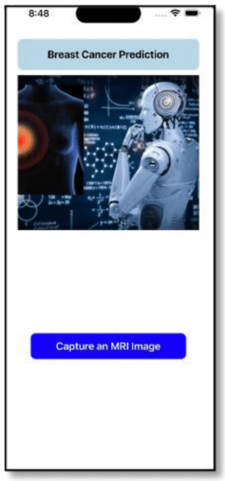
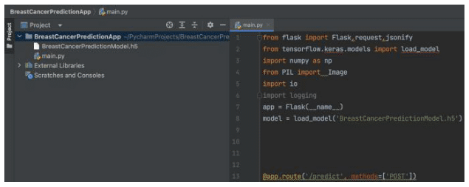


Figure 1. A developed Mobile App user Interface

3.2 API Integration

The pre-trained model, saved as a BreastCancerPredictionModel.h5 file, underwent integration into an Application Programming Interface (API) leveraging the Flask framework version 3.0.0. This process was executed within the PyCharm development environment. Flask, a widely used Python web framework renowned for its simplicity and flexibility, served as the foundation for constructing the API. With Flask's capabilities, the pre-trained model was seamlessly incorporated to facilitate its accessibility and utilization for predictive tasks.


BreastCancerPredictionApp
main.py
Project
BreastCancerPredictionApp ~/PycharmProjects/BreastCancerPre
BreastCancerPredictionModel.h5
main.py
External Libraries
Scratches and Consoles

```python
from flask import Flask,request,jsonify
from tensorflow.keras.models import load_model
import numpy as np
from PIL import Image
import io
import logging
app = Flask(__name__)
model = load_model('BreastCancerPredictionModel.h5')



@app.route('/predict', methods=['POST'])
```

Figure 2. Code snippet and folder structures for the developed API

In the Figure 2 above, the developed model, referred to as "BreastCancerPredictionModel.h5," has been loaded and assigned as "model." The API utilizes this loaded model for conducting predictions.

International Journal of Computer Applications Technology and Research
Volume 12–Issue 12, 11 - 16, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1212.1003

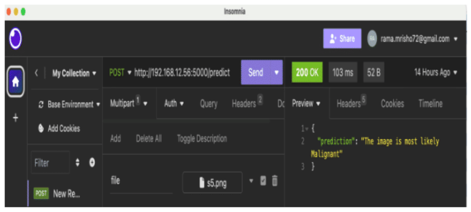
The loaded CNN model, was developed using Keras in Python, underwent training and testing using 1419 MRI images sourced from MNH. It comprises several layers:

i. An initial 2D convolutional layer with 16 filters (3x3 size, stride 1), employing the ReLU activation function. This layer transforms inputs of shape (256, 256, 3) to an output tensor of shape (254, 254, 16), featuring 448 parameters.
ii. Subsequent max pooling layers that down sample data.
iii. Following the initial layer, additional 2D convolutional layers (with 32 and 16 filters respectively, also 3x3 size, stride 1) were incorporated, each employing ReLU activation. These layers alter the tensor shapes and contain 4,640 and 4,624 parameters respectively.
iv. Fully connected layers, including one with 256 neurons and ReLU activation (with L2 regularization), amounting to 3,686,656 trainable parameters.
v. Dropout layers to prevent overfitting during training.
vi. Finally, a fully connected layer with 1 neuron using the sigmoid activation function, featuring 257 parameters.

The integration process involved configuring routes and endpoints within Flask to establish a structured interface for interacting with the pre-trained model. PyCharm, an integrated development environment (IDE) known for its robust features and support for Python, provided a conducive environment for coding, testing, and deploying the Flask-based API. This combination of Flask and PyCharm offered a conducive ecosystem for handling the integration intricacies, enabling efficient development and management of the API.

The Flask framework, being Python-based, ensured compatibility and ease of integration with the pre-trained model, streamlining the API development process. Leveraging Flask's capabilities within PyCharm, developers were empowered to create a functional and accessible API, encapsulating the pre-trained model's functionalities for consumption by various applications and systems. This approach aimed to provide a scalable, efficient, and user-friendly interface to access the predictive capabilities encapsulated within the pre-trained model.

3.3 Testing a Model API using Insomnia



POST http://192.168.12.56:5000/predict
200 OK 103 ms 52 B 14 Hours Ago

file s5.png

```
{
  "prediction": "The image is most likely Malignant"
}
```

Figure 3 shows a result of an API end point after submit an MRI image file.

3.4 AI – Mobile App after integration

The integration process involved the development of a Flask API hosting a pre-trained breast cancer prediction model. Within the Flask framework, endpoints were set up to handle requests from a React Native Mobile App, acting as a bridge for data flow between the app and the predictive model. The React Native app was tailored to capture breast MRI images via the phone's camera and initiate HTTP requests to the Flask API for prediction processing. Upon receiving image data, the Flask API processed it through the pre-trained model and relayed the prediction results (benign or malignant) back to the React Native app. This integration aimed for a seamless connection between the frontend React Native Mobile App and the backend Flask API, ensuring accurate predictions for breast cancer using captured MRI images.

Following the integration, rigorous testing verified the functionalities, including image capturing, data transmission, model prediction accuracy, and user interface responsiveness. The React Native app's user-friendly design facilitated image capture and displayed prediction outcomes while maintaining a smooth user experience. With optimizations made to enhance performance and stability, the complete Mobile App for breast cancer predictions, combining the React Native frontend and Flask backend, was readied for deployment. This integration offered a reliable solution for users, enabling them to obtain accurate breast cancer predictions conveniently through their mobile devices, ensuring accessibility and efficacy in breast cancer diagnosis.

4. MOBILE APP PREDICTIONS

Once the application is initiated, radiologists can capture an MRI image by tapping the "Capture MRI Image" button. This action triggers the mobile camera, enabling users to take an MRI image. Refer to the figure below for visual guidance.

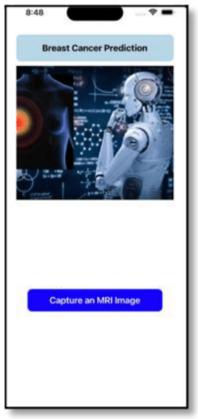


Figure 4 . A user interface for capturing an image

Next, the app will display the captured image, offering users(radiologists) the option to submit it for predictions. See Figure 5 below for a visual representation.

International Journal of Computer Applications Technology and Research
Volume 12–Issue 12, 11 - 16, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1212.1003

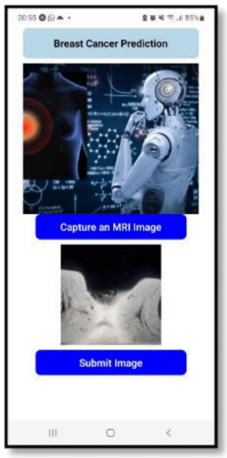
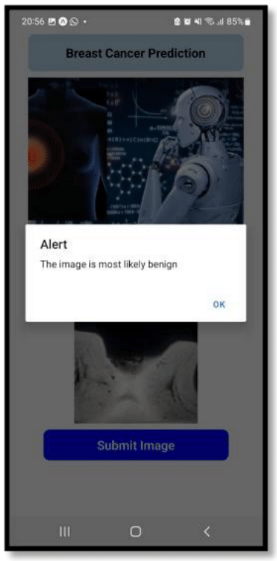


Figure 5 . Shows the captured MRI image through the camera

Then, in order for an App to predict , radiologist must click the button Submit Image, then the results will be displayed on the screen as either the captured image is Malignant or Benign. See Figure 6 below.



Alert
The image is most likely benign
OK

Figure 6. Shows a result of the captured image

Figure 6 depicted above displays the prediction output of the captured image as "Benign," as determined by the model's predictions.

Additionally, the model is also capable of predicting whether the captured image is Malignant. Refer to the figure 7 below for more details.

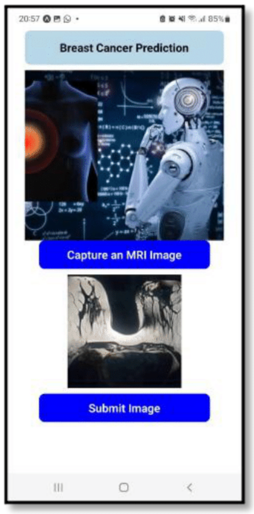
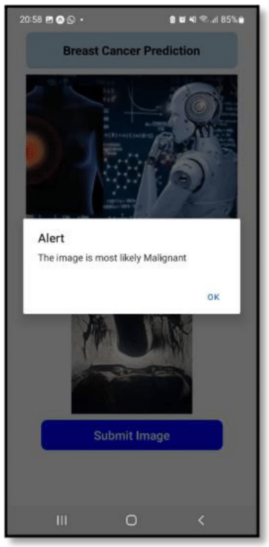


Figure 7. Another captured image for prediction

Also again, upon clicking the "**Submit Image**" button, the model will process the image and generate the output results. This process is demonstrated below.



Alert
The image is most likely Malignant
OK

Figure 8. Results of a captured image

International Journal of Computer Applications Technology and Research
Volume 12–Issue 12, 11 - 16, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1212.1003

The figure depicted above displays the prediction output of the captured image as "Malignant" as determined by the model's predictions.

5. CONCLUSION AND FUTURE WORKS

The study focuses on the integration of a Flask API model with a mobile app, establishing an AI-driven platform for predicting MRI images in breast cancer detection. Using Python, the research aimed to craft a CNN breast cancer prediction model capable of categorizing MRI images as either benign or malignant. Employing 30 MRI images sourced from Muhimbili National Hospital and expanding the dataset to 1419 images through data augmentation, the developed CNN model comprised multiple layers and exhibited remarkable evaluation metrics, affirming its efficacy in accurately discerning breast cancer cases. This underscores the potential of CNNs and data augmentation in bolstering breast cancer detection, underscoring the necessity for expanded research with more comprehensive and diverse datasets.

The application of deep learning, particularly CNNs, holds significant promise in enhancing breast cancer detection in Tanzania as it will helps radiologist to improve the accuracy of diagnosis, offering a potential solution to the limitations of traditional detection methods. The paper explores the application of deep learning techniques, delves into the effectiveness of data augmentation, addresses the constraints of conventional methodologies, and proposes a neural network model aimed at augmenting breast cancer detection rates in Tanzania. Future research avenues include broadening the dataset to encompass images from diverse demographics, thereby enhancing the model's adaptability across various patient cohorts. Moreover, integrating the model into clinical settings and assessing its real-world advantages and constraints would provide invaluable insights. This study contributes to the breast cancer detection domain by presenting an efficient CNN-based approach and sets the stage for further advancements in precision and dependability.

6. REFERENCES

[1] Al-Haija, Q. A., & Adebanjo, A. (2020). Breast cancer diagnosis in histopathological images using ResNet-50 convolutional neural network. IEMTRONICS 2020 - International IOT, Electronics and Mechatronics Conference, Proceedings, 50. https://doi.org/10.1109/IEMTRONICS51293.2020.9216455

[2] Alanazi, S. A., Kamruzzaman, M. M., Islam Sarker, M. N., Alruwaili, M., Alhwaiti, Y., Alshammari, N., & Siddiqi, M. H. (2021). Boosting Breast Cancer Detection Using Convolutional Neural Network. Journal of Healthcare Engineering, 2021. https://doi.org/10.1155/2021/5528622

[3] Bakkouri, I., & Afdel, K. (2017). Breast tumor classification based on deep convolutional neural networks. In Proceedings - 3rd International Conference on Advanced Technologies for Signal and Image Processing, ATSIP 2017. https://doi.org/10.1109/ATSIP.2017.8075562

[4] Breast Cancer Initiative. (2017). Tanzania Breast Health Care Assessment 2017: An assessment of breast cancer early detection, diagnosis and treatment in Tanzania. Available Https://Ww5.Komen.Org/Breastcancertanzania, 1–62. https://ww5.komen.org/uploadedFiles/_Komen/Content/Grants_Central/International_Grants/Grantee_Resources/Full_Tanzania_Assessment_report.pdf

[5] Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259. https://doi.org/10.1016/j.neunet.2018.07.011

[6] Dani, H. (2022). Review on Frameworks Used for Deployment of Machine Learning Model. *International Journal for Research in Applied Science and Engineering Technology*, *10*(2), 211–215. https://doi.org/10.22214/ijraset.2022.40222

[7] Eroğlu, Y., Yildirim, M., & Çinar, A. (2021). Convolutional Neural Networks based classification of breast ultrasonography images by hybrid method with respect to benign, malignant, and normal using mRMR. *Computers in Biology and Medicine*, *133*(April). https://doi.org/10.1016/j.compbiomed.2021.104407

[8] Fonseca, P., Mendoza, J., Wainer, J., Ferrer, J., Pinto, J., Guerrero, J., & Castaneda, B. (2015). Automatic breast density classification using a convolutional neural network architecture search procedure. *Medical Imaging 2015: Computer-Aided Diagnosis*, *9414*, 941428. https://doi.org/10.1117/12.2081576

[9] Gonzalez, T. F. (2007). Handbook of approximation algorithms and metaheuristics. *Handbook of Approximation Algorithms and Metaheuristics*, 1–1432. https://doi.org/10.1201/9781420010749

[10] Kamiri, J., & Mariga, G. (2021). Research Methods in Machine Learning: A Content Analysis. *International Journal of Computer and Information Technology(2279-0764)*, *10*(2), 78–91. https://doi.org/10.24203/ijcit.v10i2.79

[11] Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. In *Nature* (Vol. 521, Issue 7553, pp. 436–444). Nature Publishing Group. https://doi.org/10.1038/nature14539.

[12] Lu, H. C., Loh, E. W., & Huang, S. C. (2019). The Classification of Mammogram Using Convolutional Neural Network with Specific Image Preprocessing for Breast Cancer Detection. *2019 2nd International Conference on Artificial Intelligence and Big Data, ICAIBD 2019*, 9–12. https://doi.org/10.1109/ICAIBD.2019.8837000

[13] Masud, M. (2020). Convolutional neural network-based models for diagnosis of breast cancer. *Neural Computing and Applications*, *5*. https://doi.org/10.1007/s00521-020-05394-5

[14] Ragab, D. A., Sharkas, M., Marshall, S., & Ren, J. (2019). Breast cancer detection using deep convolutional neural networks and support vector machines. *PeerJ*, *2019*(1), 1–23. https://doi.org/10.7717/peerj.6201

[15] Yue, W., Wang, Z., Chen, H., Payne, A., & Liu, X. (2018). Machine learning with applications in breast

International Journal of Computer Applications Technology and Research
Volume 12–Issue 12, 11 - 16, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1212.1003

cancer diagnosis and prognosis. *Designs*, *2*(2), 1–17. https://doi.org/10.3390/designs2020013

[16] Yurttakal, A. H., Erbay, H., İkizceli, T., & Karaçavuş, S. (2020). Detection of breast cancer via deep convolution neural networks using MRI images. *Multimedia Tools and Applications*, *79*(21–22), 15555–15573. https://doi.org/10.1007/s11042-019-7479-6

[17] Zhao, X., Wang, X., & Wang, H. (2018). Classification of Benign and Malignant Breast Mass in Digital Mammograms with Convolutional Neural Networks. *ACM International Conference Proceeding Series*, 47–50. https://doi.org/10.1145/3285996.3286006

[18] Zuluaga-Gomez, J., Al Masry, Z., Benaggoune, K., Meraghni, S., & Zerhouni, N. (2021). A CNN-based methodology for breast cancer diagnosis using thermal images. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging and Visualization*, *9*(2), 131–145. https://doi.org/10.1080/21681163.2020.1824685

APPENDIX II : Source code of the developed model

```
#import all required Libraries

import tensorflow as tf
import os # Navigation into system folders
import cv2 #for viewing imagess
from matplotlib import pyplot as
plt
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D,Dense,Flatten,Dr
from tensorflow.keras import regularizers
from tensorflow.keras.callbacks import EarlyStopping
```

```
os.listdir('data')
```

```
['.DS_Store', 'Malignant','Benign']
```

```
os.listdir(data_dir)
```

```
data_dir='data'
```

```
Out[6]: ['.DS_Store', 'Malignant', 'Benign']
```

```
os.listdir(os.path.join(data_dir,'Malignant'))
```

```
Out[7]: ['malignant_0_2802.jpg',
'malignant_0_8491.jpg',
 'malignant_0_652.jpg',
 'malignant_0_4283.jpg',
 'malignant_0_3275.jpg',
 'malignant_0_691.jpg',
 'malignant_0_9599.jpg',
 'malignant_0_8877.jpg',
 'malignant_0_9228.jpg',
 'malignant_0_5412.jpg',
 'malignant_0_7239.jpg',
 'malignant_0_1072.jpg',
 'malignant_0_3665.jpg',
 'malignant_0_5572.jpg',
 'malignant_0_3671.jpg',
 'malignant_0_3659.jpg',
 'malignant_0_8297.jpg',
 'malignant_0_5957.jpg',
 'malignant_0_732.jpg',
 'malignant_0_3467.jpg',
 'malignant_0_929.jpg',
```

```
 'malignant_0_8269.jpg',
 'malignant_0_4490.jpg',
 'malignant_0_4645.jpg',
 'malignant_0_6640.jpg',
 'malignant_0_2432.jpg',
 'malignant_0_6132.jpg',
 'malignant_0_7562.jpg',
 'malignant_0_1113.jpg',
 'malignant_0_4094.jpg',
 'malignant_0_3923.jpg',
 'malignant_0_9598.jpg',
 'malignant_0_6330.jpg',
 'malignant_0_2630.jpg',
 'malignant_0_9765.jpg',
 'malignant_0_3506.jpg',
 'malignant_0_8447.jpg',
 'malignant_0_2181.jpg',
 'malignant_0_8490.jpg',
 'malignant_0_2803.jpg',
 'malignant_0_9968.jpg',
 'malignant_0_5820.jpg',
 'malignant_0_8337.jpg',
 'malignant_0_9997.jpg',
 'malignant_0_7992.jpg',
 'malignant_0_6865.jpg',
 'malignant_0_5377.jpg',
 'malignant_0_3060.jpg',
 'malignant_0_5439.jpg',
 'malignant_0_2430.jpg',
 'malignant_0_9565.jpg',
```

```
#img=cv2.imread(os.path.join(data_dir,'Benign','h2.jpg'))
```

```
 'malignant_0_4874.jpg',]
```

```
#img.shape
```

```
#plt.imshow(img)
```

```
data=tf.keras.utils.image_dataset_from_directory('data')
```

```
Found 1419 files belonging to 2 classes.
```

```
#Badili 0-255 kwenda 0 and 1
my_data=data.map(lambda x,y:(x/255,y))
```

```
data_iterator= my_data.as_numpy_iterator()
```

```
batch=data_iterator.next()
```

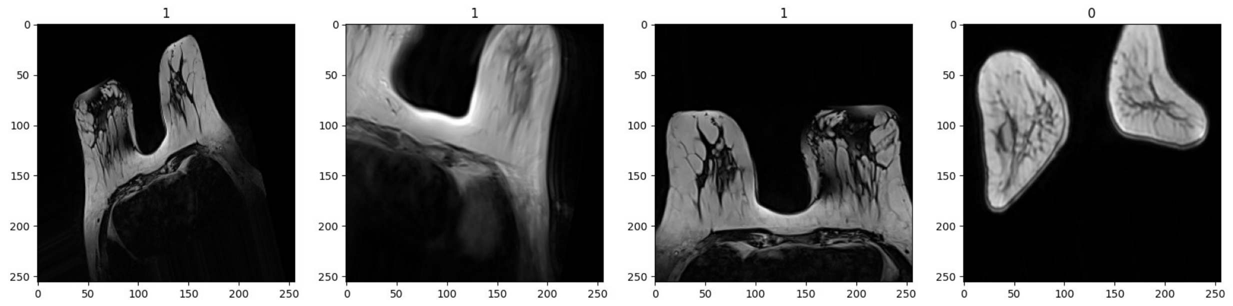
```
batch[1]
```

```
Out[12]: array([1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0,
1, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 1], dtype=int32)
```

```
batch
```

```
Out[13]: (array([[[[0.00392157, 0.00392157, 0.00392157],
[0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],
         [[0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
[0.00392157, 0.00392157, 0.00392157],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],
         [[0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],
...,
         [[0.03921569, 0.03921569, 0.03921569],
           [0.03921569, 0.03921569, 0.03921569],
      [0.03921569, 0.03921569, 0.03921569],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],

         [[0.03921569, 0.03921569, 0.03921569],
          [0.03374694, 0.03374694, 0.03374694],
          [0.03374694, 0.03374694, 0.03374694],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],

         [[0.03556985, 0.03556985, 0.03556985],
          [0.0292739 , 0.0292739 , 0.0292739 ],
          [0.02883241, 0.02883241, 0.02883241],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
             [0.        , 0.        , 0.        ]]],


        [[[0.4046243 , 0.4046243 , 0.4046243 ],
          [0.52548444, 0.52548444, 0.52548444],
          [0.5739181 , 0.5739181 , 0.5739181 ],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
             [0.        , 0.        , 0.        ]],

         [[0.43476754, 0.43476754, 0.43476754],
           [0.5008138 , 0.5008138 , 0.5008138 ],
      [0.56170535, 0.56170535, 0.56170535],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
             [0.        , 0.        , 0.        ]],

         [[0.43676472, 0.43676472, 0.43676472],
          [0.4693321 , 0.4693321 , 0.4693321 ],
          [0.55607957, 0.55607957, 0.55607957],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],

        ...,

         [[0.01568628, 0.01568628, 0.01568628],
          [0.01519608, 0.01519608, 0.01519608],
          [0.01568628, 0.01568628, 0.01568628],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],

         [[0.01568628, 0.01568628, 0.01568628],
          [0.01519608, 0.01519608, 0.01519608],
          [0.01568628, 0.01568628, 0.01568628],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],

         [[0.01568628, 0.01568628, 0.01568628],
          [0.01568628, 0.01568628, 0.01568628],
          [0.01568628, 0.01568628, 0.01568628],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
 [0.        , 0.        , 0.        ]]],


        [[[0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          ...,
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157]],

         [[0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          ...,
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157]],

         [[0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          ...,
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157]],

         ...,

         [[0.07581571, 0.07581571, 0.07581571],
          [0.08995481, 0.08995481, 0.08995481],           [0.10526961,
0.10526961, 0.10526961],
          ...,
          [0.07328431, 0.07328431, 0.07328431],
          [0.03259421, 0.03259421, 0.03259421],
          [0.02887944, 0.02887944, 0.02887944]],

         [[0.07438725, 0.07438725, 0.07438725],
          [0.08002451, 0.08002451, 0.08002451],
          [0.0914254 , 0.0914254 , 0.0914254 ],
          ...,
          [0.07328431, 0.07328431, 0.07328431],
          [0.04546569, 0.04546569, 0.04546569],
          [0.04546569, 0.04546569, 0.04546569]],

         [[0.0942402 , 0.0942402 , 0.0942402 ],
          [0.08811274, 0.08811274, 0.08811274],
          [0.08598728, 0.08598728, 0.08598728],
          ...,
          [0.06037454, 0.06037454, 0.06037454],
          [0.05281863, 0.05281863, 0.05281863],
          [0.04093137, 0.04093137, 0.04093137]]],


        ...,


        [[[0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          ...,
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
[0.00392157, 0.00392157, 0.00392157]],

         [[0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          ...,
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157]],

         [[0.        , 0.        , 0.        ],
           [0.        , 0.        , 0.        ],
      [0.        , 0.        , 0.        ],
          ...,
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157]],

         ...,

         [[0.00392157, 0.00392157, 0.00392157],          [0.00392157,
0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          ...,
          [0.0066636 , 0.0066636 , 0.0066636 ],
          [0.00392157, 0.00392157, 0.00392157],

          [0.07671569, 0.07671569, 0.07671569],          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],

         [[0.03529412, 0.03529412, 0.03529412],
          [0.05416667, 0.05416667, 0.05416667],
          [0.08317823, 0.08317823, 0.08317823],
          ...,
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ],
          [0.        , 0.        , 0.        ]],

         [[0.03529412, 0.03529412, 0.03529412],
          [0.05833333, 0.05833333, 0.05833333],
          [0.08848039, 0.08848039, 0.08848039],
          ...,
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157],
          [0.00392157, 0.00392157, 0.00392157]]],

          [0.0292739  ,  0.0292739  ,  0.0292739  ]]]],  dtype=float32),
array([1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0 ,
      1, 1, 1, 0, 0, 1, 0, 0, 1, 1], dtype=int32))
```

```
#class 0 = Benign -Non Cancelous
#class 1=Malgnant -Cancelous
fig,ar=plt.subplots(ncols=4,figsize=(20,20))
for cs,img in enumerate(batch[0][:4]):
ar[cs].imshow(img)
    ar[cs].title.set_text(batch[1][cs])
```



```
#Spriting the datasets into train size ,validation size and testing
size
train_size=int(len(mydata)*.5) val_size=int(len(mydata)*.4)
test_size=int(len(mydata)*.1)+1
```

```
len(test)
```

```
#Bulding  a CNN deep leaning Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D,Dense,Flatten,Dr
```

```
 model=Sequential()
```

```
#layer 1

model.add(Conv2D(16,(3,3),1,activation='relu',input_shape=(256,256,3)))

#layer2

model.add(MaxPooling2D())

#Layer 3

model.add(Conv2D(32,(3,3),1,activation='relu'))

#Layer 4

model.add(MaxPooling2D())

#Layer 5

model.add(Conv2D(16,(3,3),1,activation='relu'))


#Layer 6


model.add(MaxPooling2D())


#Layer 7


model.add(Flatten())


#Adding L2 regularization to prevent overfitting in neural
#networks by adding a penalty term to the loss function that encourages t

#Layer 8




model.add(Dense(256,activation='relu',kernel_regularizer=regularizers.l2

#Adding a Dropout layer after a fully connected layer to prevent
overfitt
```

```
#Layer 9


model.add(Dropout(0.2))
#Layer 10


model.add(Dense(1,activation='sigmoid'))
```

```
#compling the model
model.compile('adam',loss=tf.losses.BinaryCrossentropy(),metrics=['accura
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 254, 254, 16)      448
max_pooling2d (MaxPooling2D  (None, 127, 127, 16)     0         )
conv2d_1 (Conv2D)           (None, 125, 125, 32)      4640
max_pooling2d_1 (MaxPooling  (None, 62, 62, 32)       0
2D)
conv2d_2 (Conv2D)           (None, 60, 60, 16)        4624
max_pooling2d_2 (MaxPooling  (None, 30, 30, 16)       0
2D)
flatten (Flatten)           (None, 14400)             0
dense (Dense)               (None, 256)               3686656
dropout (Dropout)           (None, 256)               0
dense_1 (Dense)             (None, 1)                 257
=================================================================
Total params: 3,696,625
Trainable params: 3,696,625
Non-trainable params: 0
_________________________________________________________________
```

```
#Stop training when a monitored metric has stopped improving.
#For stopping the training process if validation loss  does not change
fo early_stop = EarlyStopping(monitor='val_loss', patience=5)
```

```
hist=model.fit(train,epochs=30,validation_data=val,callbacks=[early_stop]
```

```
Epoch 1/30
22/22 [==============================] - ETA: 0s - loss: 2.2250 - accura
cy: 0.7230
22/22 [==============================] - 17s 697ms/step - loss: 2.2250 -
accuracy: 0.7230 - val_loss: 0.7458 - val_accuracy: 0.7552 Epoch 2/30
22/22 [==============================] - 16s 719ms/step - loss: 0.5419 -
accuracy: 0.8679 - val_loss: 0.3395 - val_accuracy: 0.9392 Epoch 3/30
22/22 [==============================] - 16s 715ms/step - loss: 0.2875 -
accuracy: 0.9616 - val_loss: 0.2345 - val_accuracy: 0.9774 Epoch 4/30
22/22 [==============================] - 16s 707ms/step - loss: 0.1866 -
accuracy: 0.9830 - val_loss: 0.1537 - val_accuracy: 0.9948 Epoch 5/30
22/22 [==============================] - 16s 711ms/step - loss: 0.1358 -
accuracy: 0.9929 - val_loss: 0.1654 - val_accuracy: 0.9792 Epoch 6/30
22/22 [==============================] - 16s 706ms/step - loss: 0.1443 -
accuracy: 0.9830 - val_loss: 0.1387 - val_accuracy: 0.9948 Epoch 7/30
22/22 [==============================] - 16s 705ms/step - loss: 0.1482 -
accuracy: 0.9929 - val_loss: 0.1124 - val_accuracy: 1.0000 Epoch 8/30
22/22 [==============================] - 16s 718ms/step - loss: 0.1223 -
accuracy: 0.9915 - val_loss: 0.1987 - val_accuracy: 0.9566 Epoch 9/30
22/22 [==============================] - 16s 713ms/step - loss: 0.1544 -
accuracy: 0.9901 - val_loss: 0.1262 - val_accuracy: 0.9931 Epoch 10/30
22/22 [==============================] - 16s 707ms/step - loss: 0.1636 -
accuracy: 0.9773 - val_loss: 0.1933 - val_accuracy: 0.9878 Epoch 11/30
22/22 [==============================] - 16s 729ms/step - loss: 0.1483 -
accuracy: 0.9972 - val_loss: 0.1007 - val_accuracy: 0.9965 Epoch 12/30
22/22 [==============================] - 17s 747ms/step - loss: 0.0825 -
accuracy: 0.9957 - val_loss: 0.0980 - val_accuracy: 0.9931 Epoch 13/30
22/22 [==============================] - 17s 739ms/step - loss: 0.1148 -
accuracy: 0.9915 - val_loss: 0.1292 - val_accuracy: 0.9896 Epoch 14/30
22/22 [==============================] - 16s 718ms/step - loss: 0.1307 -
accuracy: 0.9886 - val_loss: 0.1555 - val_accuracy: 0.9809 Epoch 15/30
22/22 [==============================] - 16s 715ms/step - loss: 0.1222 -
accuracy: 0.9957 - val_loss: 0.0811 - val_accuracy: 1.0000 Epoch 16/30
22/22 [==============================] - 16s 736ms/step - loss: 0.0718 -
accuracy: 0.9986 - val_loss: 0.0536 - val_accuracy: 0.9983 Epoch 17/30
```

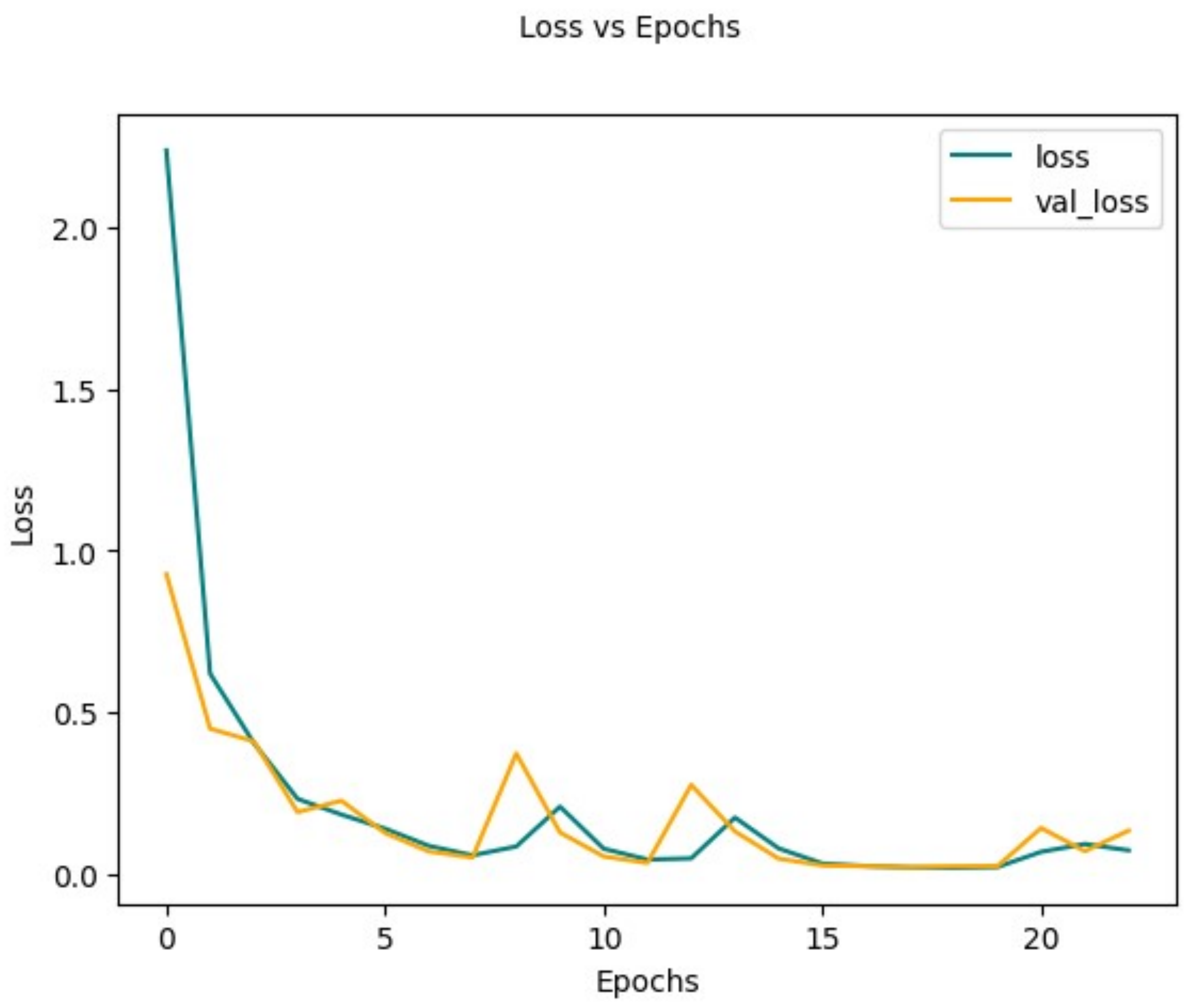
```
22/22 [==============================] - 16s 733ms/step - loss: 0.0727 -
accuracy: 0.9886 - val_loss: 0.2055 - val_accuracy: 0.9427 Epoch 18/30
22/22 [==============================] - 16s 725ms/step - loss: 0.1989 -
accuracy: 0.9716 - val_loss: 0.1868 - val_accuracy: 0.9931 Epoch 19/30
22/22 [==============================] - 16s 728ms/step - loss: 0.1437 -
accuracy: 0.9943 - val_loss: 0.0854 - val_accuracy: 0.9983 Epoch 20/30
22/22 [==============================] - 16s 715ms/step - loss: 0.0616 -
accuracy: 0.9986 - val_loss: 0.0408 - val_accuracy: 1.0000 Epoch 21/30
22/22 [==============================] - 16s 718ms/step - loss: 0.0294 -
accuracy: 1.0000 - val_loss: 0.0216 - val_accuracy: 1.0000 Epoch 22/30
22/22 [==============================] - 17s 743ms/step - loss: 0.0249 -
accuracy: 1.0000 - val_loss: 0.0279 - val_accuracy: 1.0000 Epoch 23/30
22/22 [==============================] - 16s 707ms/step - loss: 0.0475 -
accuracy: 0.9943 - val_loss: 0.1241 - val_accuracy: 0.9688 Epoch 24/30
22/22 [==============================] - 15s 689ms/step - loss: 0.1099 -
accuracy: 0.9943 - val_loss: 0.1028 - val_accuracy: 0.9983 Epoch 25/30
22/22 [==============================] - 16s 693ms/step - loss: 0.0732 -
accuracy: 1.0000 - val_loss: 0.0509 - val_accuracy: 1.0000 Epoch 26/30
22/22 [==============================] - 15s 686ms/step - loss: 0.0336 -
accuracy: 1.0000 - val_loss: 0.0257 - val_accuracy: 1.0000
```

```
#hist.history
```

In
[30]

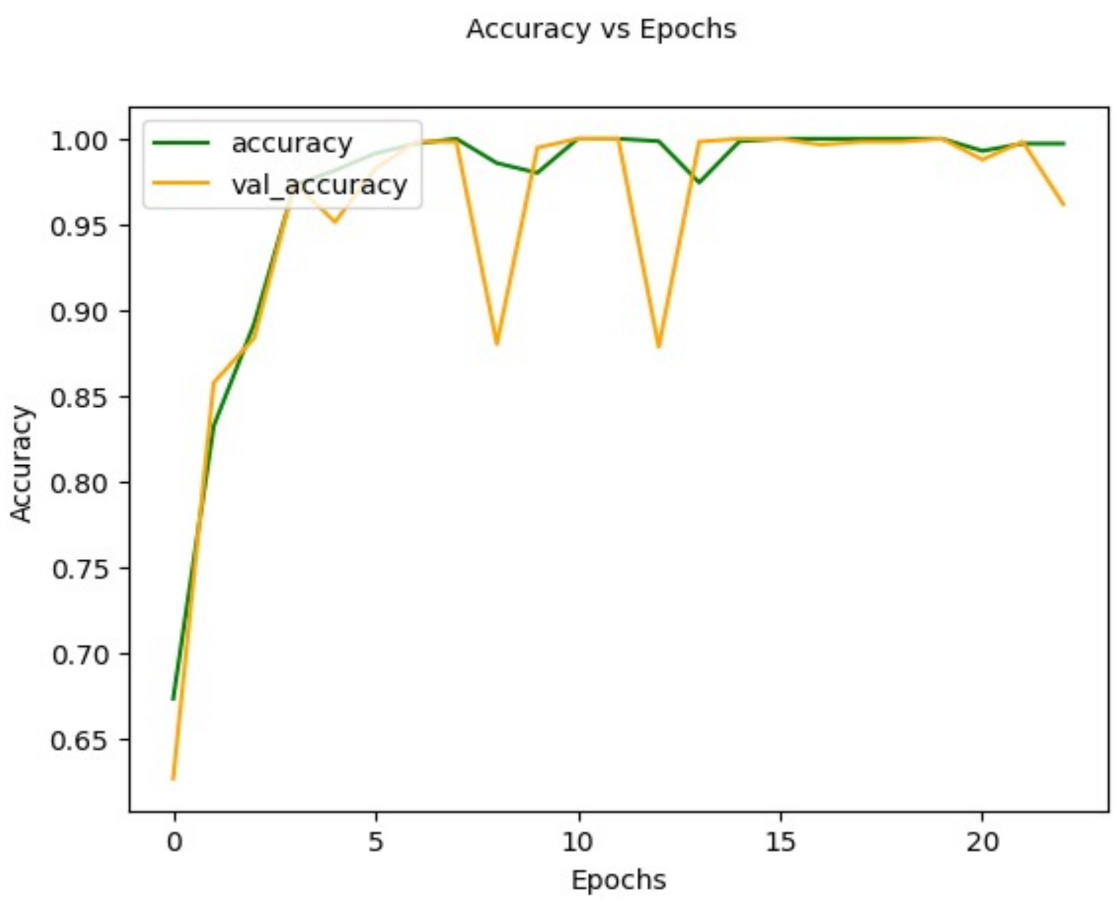
```
#Plotting the Loss  Perfomance
fig_1=plt.figure()
plt.plot(hist.history['loss'],color='teal',label='loss')
plt.plot(hist.history['val_loss'],color='orange',label='val_loss')
fig_1.suptitle('Loss vs Epochs',fontsize=10)
plt.legend(loc="upper right")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```

:



```
#Plotting the accuracy  Perfomance
 fig_2=plt.figure()
plt.plot(hist.history['accuracy'],color='green',label='accuracy')
plt.plot(hist.history['val_accuracy'],color='orange',label='val_accuracy'
fig_2.suptitle('Accuracy vs Epochs',fontsize=10)
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc="upper left")
plt.show()
```

In
[354
…



```
import Precision,Recall,BinaryAccuracy
```

```
pre=Precision() re=Recall()
acc=BinaryAccuracy()
```

```
In [342… for batch in test.as_numpy_iterator():
X,y=batch
prediction=model.predict(X)
pre.update_state(y,prediction)
re.update_state(y,prediction)
acc.update_state(y,prediction)
```

```
1/1 [==============================] - 0s 211ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 107ms/step
```

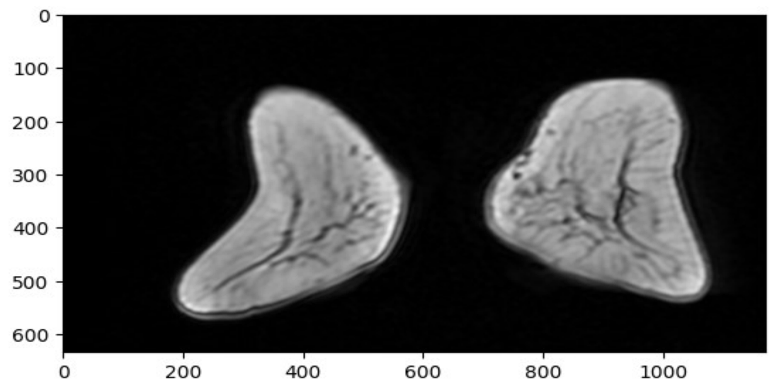
```
In [343… print(f'

Precision:{pre.result().numpy()},Recall:{re.result().numpy()},Ac

Precision:1.0,Recall:0.9358974099159241,Accuracy:0.9640287756919861
```

```
In [344… f1_score=
2*(pre.result().numpy()*re.result().numpy())/(pre.result().nump
```

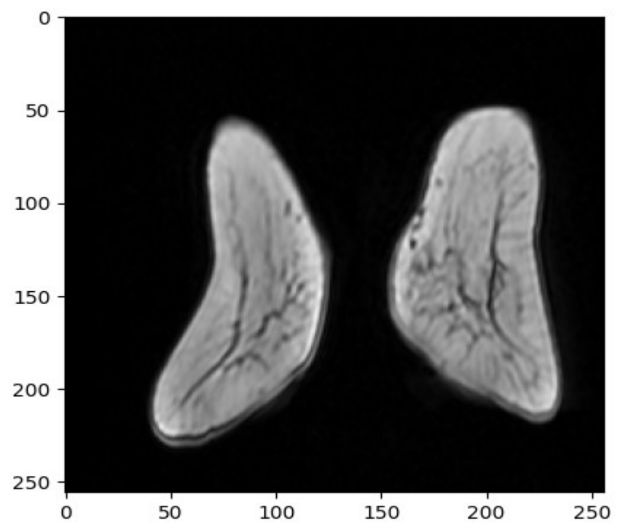
```
In [345… f1_score

Out[345]: 0.9668874331228751
```

```
import cv2
```

```
img=cv2.imread('benign.png')
plt.imshow(img)
plt.show()
```



```
In [347… resize=tf.image.resize(img,(256,256))
plt.imshow(resize.numpy().astype(int))
plt.show()
```



```
In [348… prediction=model.predict(np.expand_dims(resize/255,0))

1/1 [==============================] - 0s 21ms/step

In [349… prediction

Out[349]: array([[0.00012774]], dtype=float32)

In [350… if prediction > 0.5:      print('This
image is likely to be Malignant') else:
print('This image is likely to be Benign')

This image is likely to be  Benign

In [377… model.save(os.path.join('model_version
1.0','BreastCancerPredictionModel.

In [378… new_model=load_model(os.path.join('model_version
1.0','BreastCancerPredic
```

APPENDIX III : Source code of the model’s dashboard made using streamlit

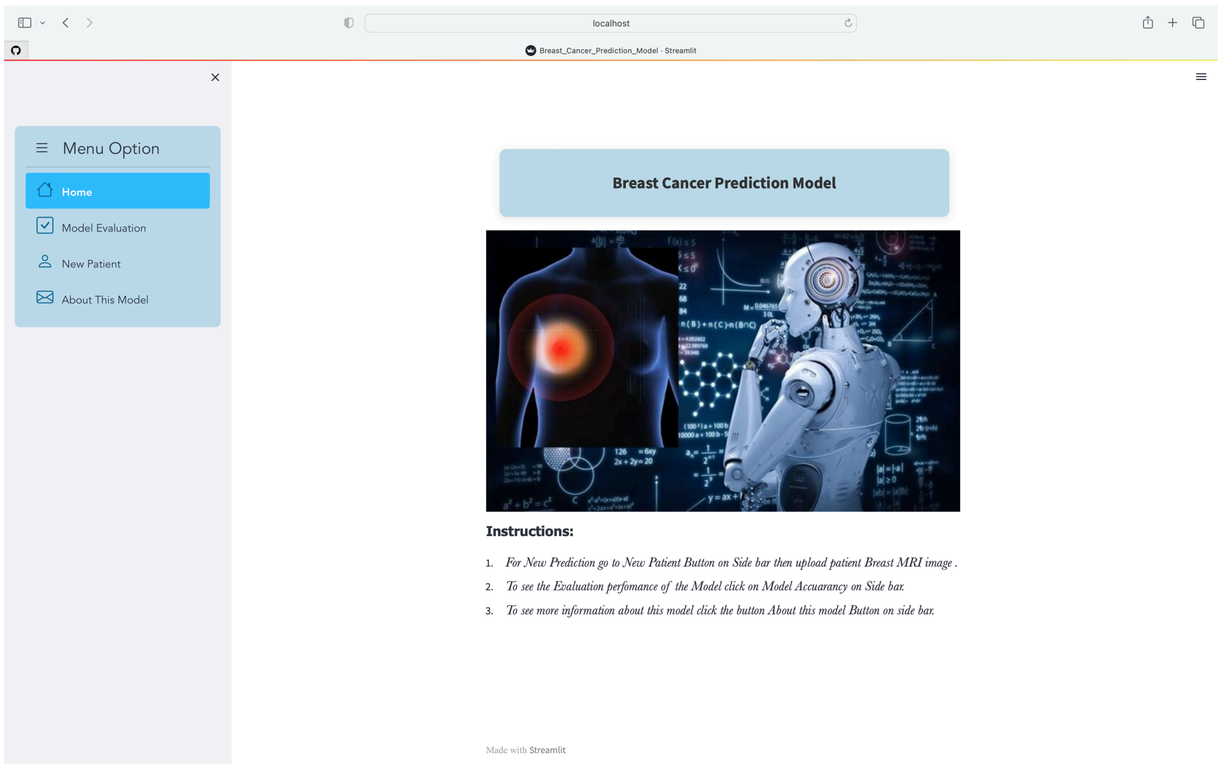
```
import streamlit as st
from streamlit_option_menu import option_menu
from PIL import Image, ImageOps
import time
import h5py
import tables
import random
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np
import tensorflow as tf



with st.sidebar:
     selected =option_menu(
     menu_title="Menu Option",
     options=["Home","Model Evaluation","New Patient","About This Model"],
     icons=["house","check-square","person","envelope"],
     menu_icon="list",
     default_index=0,
         styles={
             "container": {"padding": "5!important", "background-color":
"#B8D8E8"},
             "icon": {"color": "#0D6B99", "font-size": "25px"},
             "nav-link": {"font-size": "16px", "text-align": "left",
"margin": "0px", "--hover-color": "#eee"},
             "nav-link-selected": {"background-color": "#33BBFF"},
         }
      )

    # Define CSS style
     style = """
    <style>
        .metrics {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100px;
            background-color: #B8D8E8;
            border-radius: 10px;
            box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.1);
            margin: 20px;
            padding: 20px;
            font-size: 24px;
            font-weight: bold;
            color: #333333;
            text-align: center;
        }
        .metric-item {
            margin: 20px;
        }
        .precision {
            color: green;
        }
        .recall {
            color: blue;
        }
        .accuracy {
            color: orange;
        }
    </style>
    """

    # Display the CSS style
     st.markdown(style, unsafe_allow_html=True)


#Home button
if selected=="Home":
    times_new_roman_style = """
       <style>
           body {
               font-family: 'Times New Roman', Times, serif;
           }
       </style>
   """

    # Use the css method to apply the CSS style to the page
    st.write(times_new_roman_style, unsafe_allow_html=True)

    # Display some text using the Times New Roman font
    #st.write("This text is in Times New Roman.")

    #st.success('BREAST CANCER PREDICTION MODEL')
    st.write("<div class='metrics'>Breast Cancer Prediction Model</div>",
unsafe_allow_html=True)
    #st.write("<h3 style='text-align: center;font-family:Baskerville'>Breast
Cancer Prediction Model</h2>", unsafe_allow_html=True)
    #st.image("ai.jpeg",use_column_width='center',width=700)

    with st.empty():
     st.image("ai.jpeg",use_column_width='center',width=700)

    #instrctions
    st.write("<h5 style='text-align: left;font-
family:Tahoma'>Instructions:</h2>", unsafe_allow_html=True)
    st.write("<ol>"
             "<li>""<i style='text-align: justify;font-
family:Baskerville;font-size:20px'>For New Prediction go to New Patient
Button on Side bar then upload patient Breast MRI image " ".</i>""</li>"
             "<li><i style='text-align: justify;font-family:Baskerville;font-
size:20px'>To see the Evaluation perfomance of the Model click on Model
Accuarancy on Side bar" ".</i></li>"
             "<li><i style='text-align: justify;font-family:Baskerville;font-
size:20px'>To see more information about this model click the button  About
this model Button on side bar" ".</i></li>"
             "</ol>", unsafe_allow_html=True)
```

```
#Model Accuracy button
if selected=="Model Evaluation":


    # Define the values
    precision = 1.0
    recall = 0.93
    accuracy = 0.96
    f1_score=0.96

    st.write("<div class='metrics'> Model Evaluation
Metrics</div>",unsafe_allow_html=True)
    # Display the values using the CSS style
    #st.write("<div class='metrics'>", unsafe_allow_html=True)
    st.write(f"<h3 class='metric-item precision'>Precision:
{precision}</h3>", unsafe_allow_html=True)
    st.write(f"<h3 class='metric-item recall'>Recall: {recall}</h3>",
unsafe_allow_html=True)
    st.write(f"<h3 class='metric-item accuracy'>Accuracy: {accuracy}</h3>",
unsafe_allow_html=True)
    st.write(f"<h3 class='metric-item '>F1-Score: {f1_score}</h3>",
unsafe_allow_html=True)

    st.write("<ul>"
             "<li>""<i style='text-align: justify;font-
family:Baskerville;font-size:20px'>Overall, these metrics indicate that the
model has high precision, recall, and accuracy, which suggests that it is
performing well on the classification task. " ".</i>""</li>"
             "</ul>", unsafe_allow_html=True)

    #st.image("loss.png", use_column_width='left', width=400)
    #st.image("accuracy.png", use_column_width='right', width=400)

    #st.write("</div>", unsafe_allow_html=True)




  # st.write("This is Model Accuracy")


#Model Prediction button
if selected=="New Patient":
    # Loading the Model
   st.write("<div class='metrics'> Model Predictions</div>",
unsafe_allow_html=True)
   model = load_model('BreastCancerPredictionModel.h5')
   uploaded_image = st.file_uploader("Please Upload your MRI Breast Image for
Prediction",type=["png", "jpg", "jpeg"])
   if uploaded_image is not None:

       progress_bar = st.progress(0)
       status_text = st.empty()
       for i in range(100):
           time.sleep(0.1)
           progress_bar.progress(i + 1)
```

```
            status_text.text(f"Please
Wait........................................................................ {i
+ 1}%")
        image = Image.open(uploaded_image)

        #CHECKING IF AN IMAGE HAVE 3 OR 4 CHANNELS

        num_channels = len(image.getbands())
        if(num_channels==4):
            image = Image.open(uploaded_image).convert('RGB')

        resize_image= tf.image.resize(image, (256, 256))

        prediction =model.predict(np.expand_dims(resize_image/255,0))

        if(prediction<0.5):
         st.success(prediction)
         st.success("The image is most likely benign")
        else:
         st.error(prediction)
         st.error("The image is most likely malignant")

        # st.image(image, caption='Uploaded image', use_column_width=True)



if selected=="About This Model":

    st.write("<h3 style='text-align: center;font-family:Baskerville'>Model
Summary</h3>", unsafe_allow_html=True)

    st.write("<ul><li><p style='text-align: justify;font-
family:Baskerville;font-size:20px'>This is a Convolutional Neural
Network(CNN) model which have been  developed using the Keras deep learning
library in Python .The model has been trained and tested  using 1419 MRI
images from MNH. The model  consist of the following CNN layers :-"
             ".</p></li></ul>", unsafe_allow_html=True)

    st.write("<ol>"
             "<li>""<p style='text-align: justify;font-
family:Baskerville;font-size:17px'>A 2D convolutional layer with 16 filters
of size 3x3 and stride 1, using the ReLU activation function. This layer
takes an input of shape (256, 256, 3) and outputs a tensor of shape (254,
254, 16), with a total of 448 parameters." ".</p>""</li>"
             "<li><p style='text-align: justify;font-family:Baskerville;font-
size:17px'>A max pooling layer that downsamples the input by taking the
maximum value in each non-overlapping patch of size 2x2. This layer outputs a
tensor of shape (127, 127, 16) with no parameters." ".</p></li>"
             "<li><p style='text-align: justify;font-family:Baskerville;font-
size:17px'>Another 2D convolutional layer with 32 filters of size 3x3 and
stride 1, using the ReLU activation function. This layer takes an input of
shape (127, 127, 16) and outputs a tensor of shape (125, 125, 32), with a
total of 4,640 parameters." ".</p></li>"
             "<li><p style='text-align: justify;font-family:Baskerville;font-
size:17px'>Another max pooling layer that downsamples the input by taking the
maximum value in each non-overlapping patch of size 2x2. This layer outputs a
tensor of shape (62, 62, 32) with no parameters." ".</i></li>"
```

```
            "<li><p style='text-align: justify;font-family:Baskerville;font-size:17px'>A third 2D convolutional layer with 16 filters of size 3x3 and stride 1, using the ReLU activation function. This layer takes an input of shape (62, 62, 32) and outputs a tensor of shape (60, 60, 16), with a total of 4,624 parameters." ".</p></li>"
            "<li><p style='text-align: justify;font-family:Baskerville;font-size:17px'>A third max pooling layer that downsamples the input by taking the maximum value in each non-overlapping patch of size 2x2. This layer outputs a tensor of shape (30, 30, 16) with no parameters." ".</p></li>"
            "<li><p style='text-align: justify;font-family:Baskerville;font-size:17px'>A fully connected layer with 256 neurons, using the ReLU activation function and L2 regularization with a penalty term of 0.01. This layer has 3,686,656 trainable parameters." ".</p></li>"
            "<li><p style='text-align: justify;font-family:Baskerville;font-size:17px'>A layer that randomly drops out 20% of the inputs during training to prevent overfitting. This layer has no parameters." ".</p></li>"
            "<li><p style='text-align: justify;font-family:Baskerville;font-size:17px'>A fully connected layer with 1 neuron, using the sigmoid activation function. This layer has 257 parameters." ".</p></li>"
            "</ol>", unsafe_allow_html=True)
```

APPENDIX IV : Model's dashboard

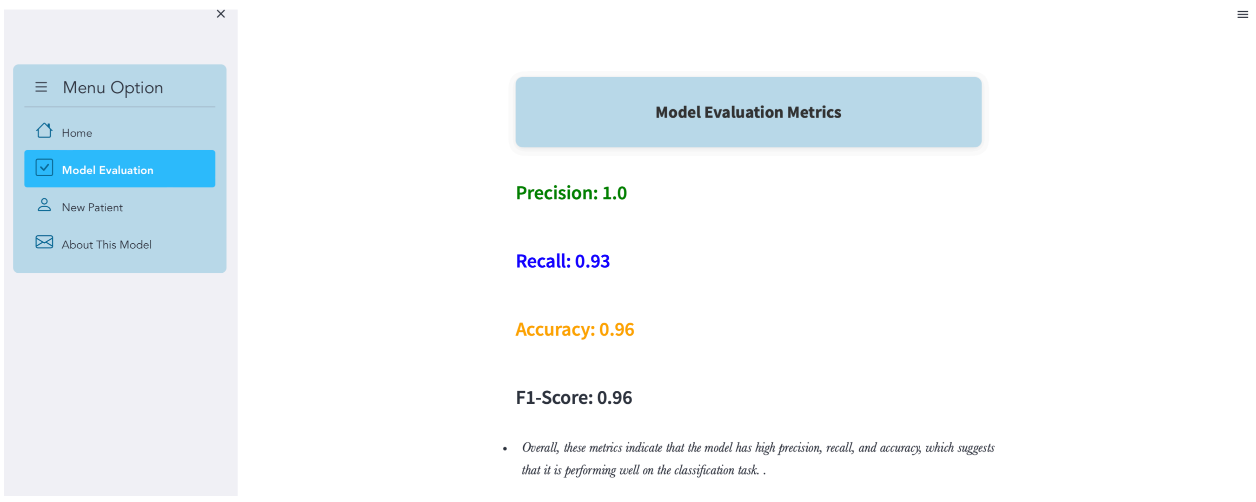


Breast Cancer Prediction Model

Instructions:

1. *For New Prediction go to New Patient Button on Side bar then upload patient Breast MRI image .*
2. *To see the Evaluation perfomance of the Model click on Model Accuarancy on Side bar.*
3. *To see more information about this model click the button About this model Button on side bar.*

Made with Streamlit



Model Evaluation Metrics

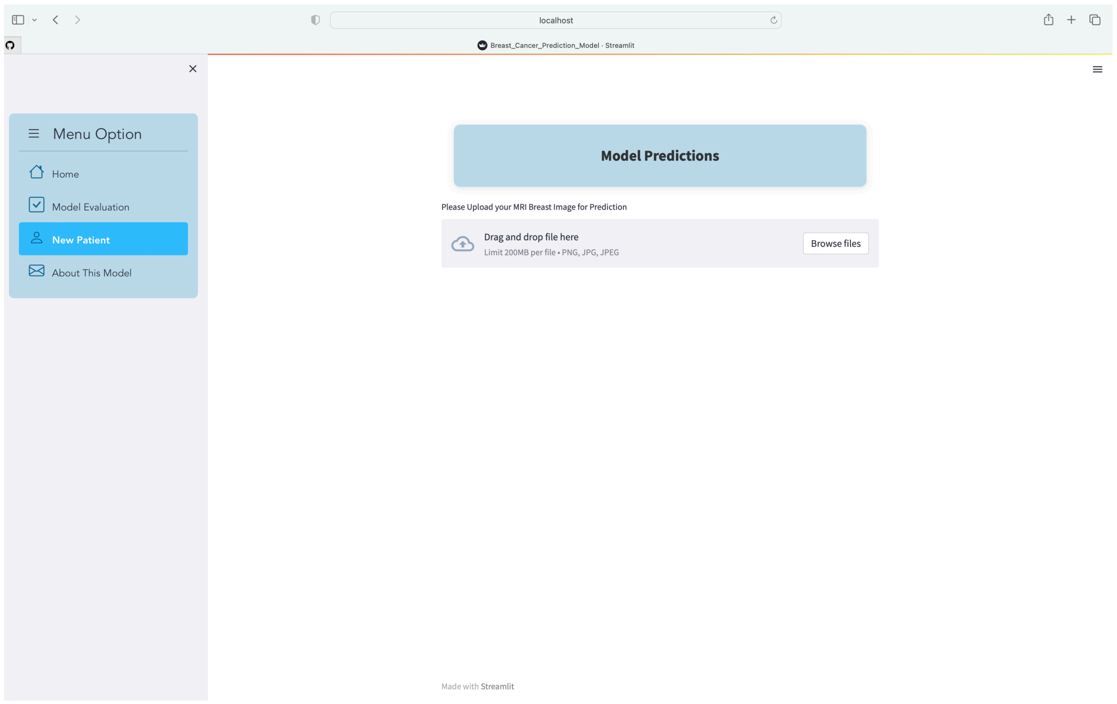
Precision: 1.0

Recall: 0.93

Accuracy: 0.96

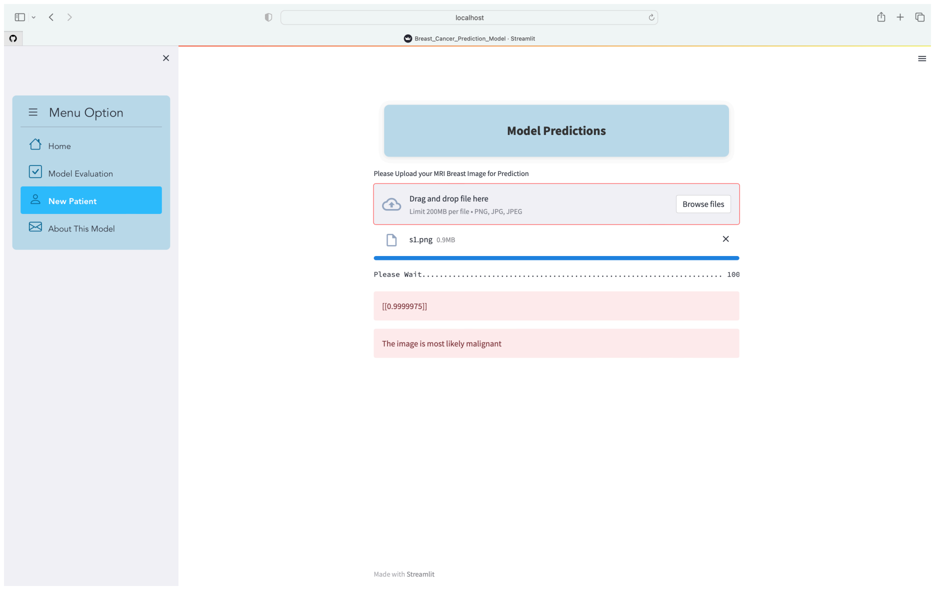
F1-Score: 0.96

- *Overall, these metrics indicate that the model has high precision, recall, and accuracy, which suggests that it is performing well on the classification task. .*



Model Predictions

Please Upload your MRI Breast Image for Prediction



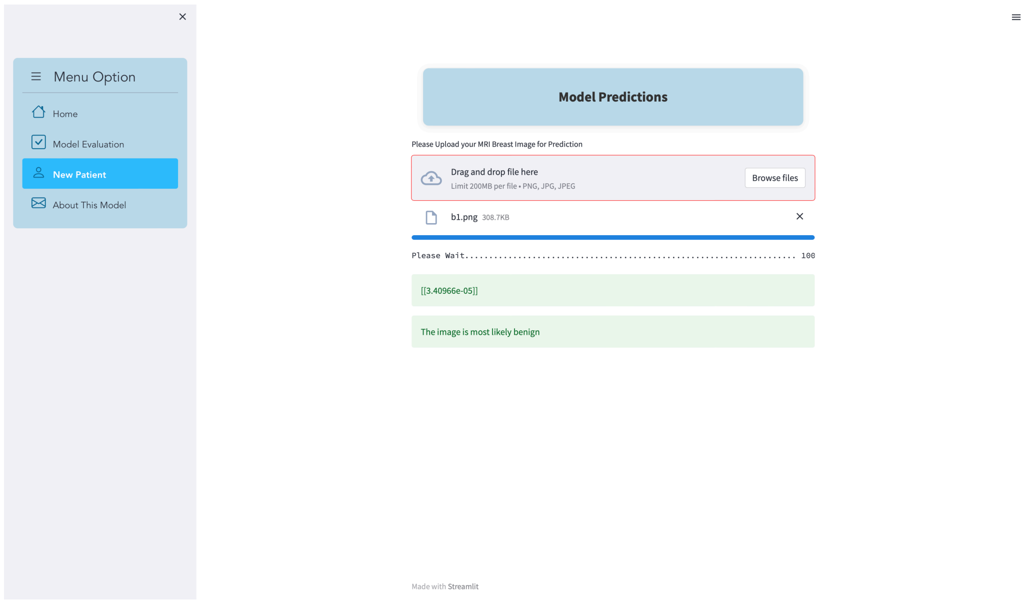
Model Predictions

Please Upload your MRI Breast Image for Prediction

```
Please Wait................................................................... 100
```

[[0.9999975]]

The image is most likely malignant



Model Predictions

Please Upload your MRI Breast Image for Prediction

Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG
Browse files

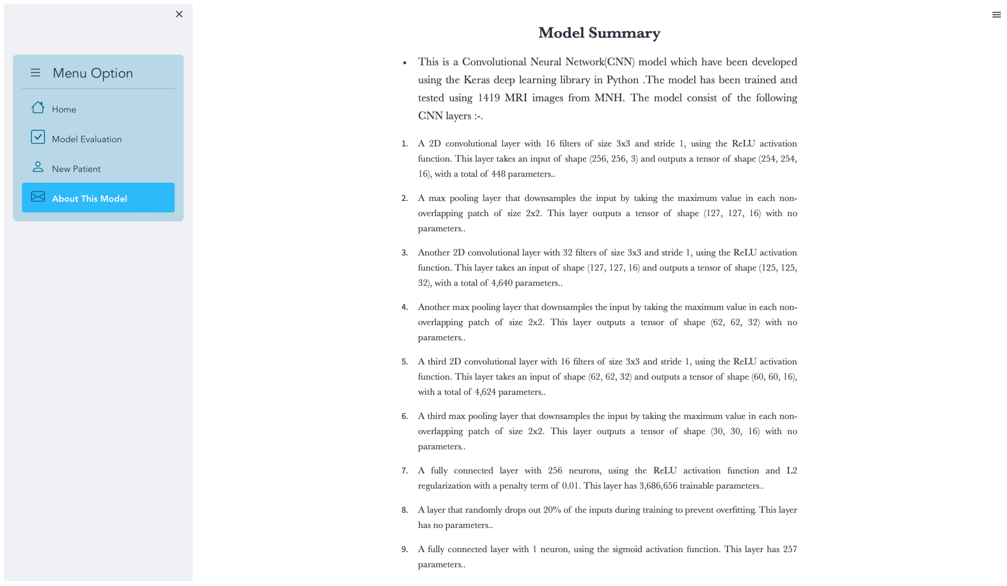
b1.png 308.7KB

```
Please Wait...................................................................... 100
```

[[3.40966e-05]]

The image is most likely benign

Made with Streamlit

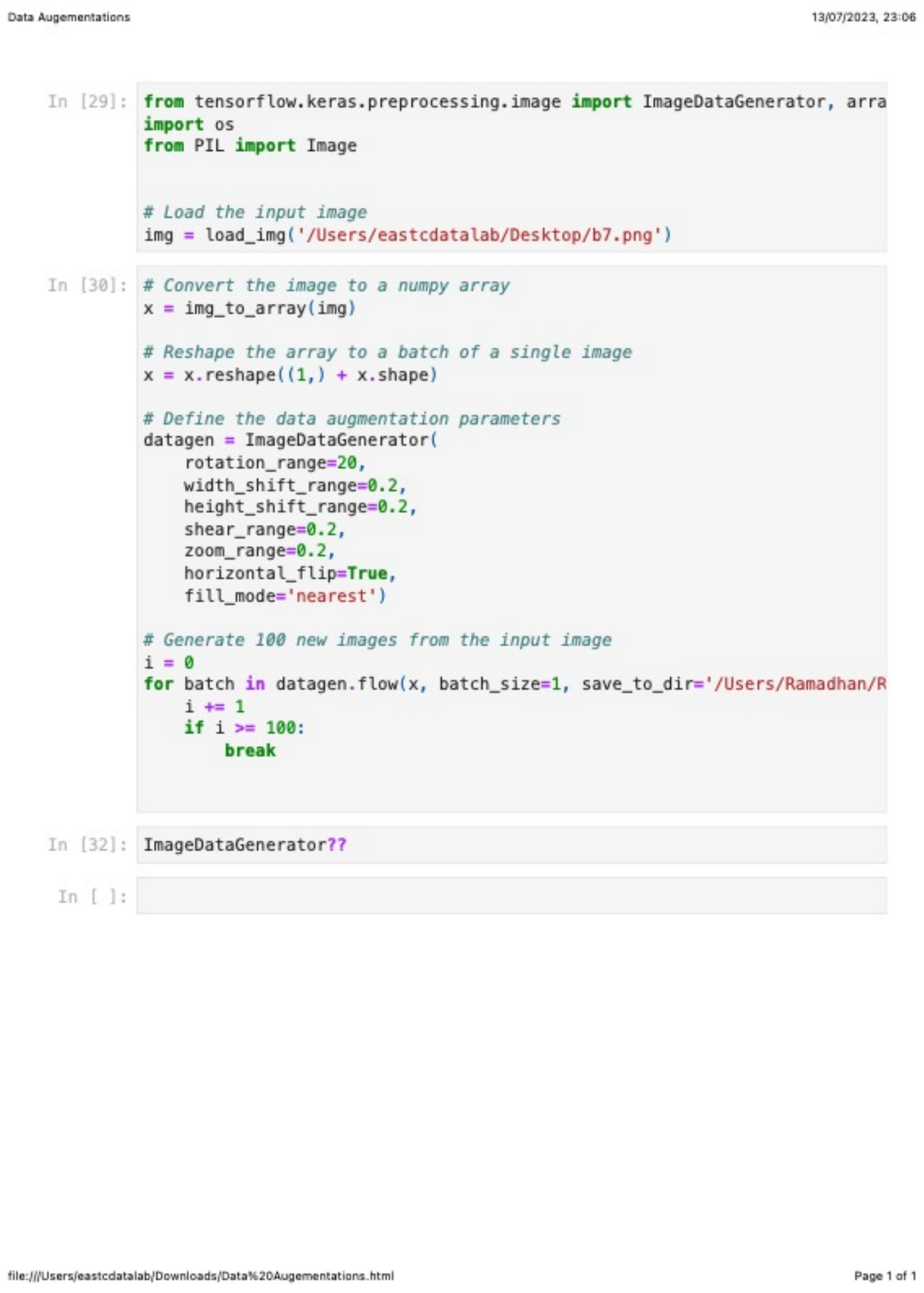


Model Summary

- This is a Convolutional Neural Network(CNN) model which have been developed using the Keras deep learning library in Python .The model has been trained and tested using 1419 MRI images from MNH. The model consist of the following CNN layers :-.

1. A 2D convolutional layer with 16 filters of size 3x3 and stride 1, using the ReLU activation function. This layer takes an input of shape (256, 256, 3) and outputs a tensor of shape (254, 254, 16), with a total of 448 parameters..
2. A max pooling layer that downsamples the input by taking the maximum value in each non-overlapping patch of size 2x2. This layer outputs a tensor of shape (127, 127, 16) with no parameters..
3. Another 2D convolutional layer with 32 filters of size 3x3 and stride 1, using the ReLU activation function. This layer takes an input of shape (127, 127, 16) and outputs a tensor of shape (125, 125, 32), with a total of 4,640 parameters..
4. Another max pooling layer that downsamples the input by taking the maximum value in each non-overlapping patch of size 2x2. This layer outputs a tensor of shape (62, 62, 32) with no parameters..
5. A third 2D convolutional layer with 16 filters of size 3x3 and stride 1, using the ReLU activation function. This layer takes an input of shape (62, 62, 32) and outputs a tensor of shape (60, 60, 16), with a total of 4,624 parameters..
6. A third max pooling layer that downsamples the input by taking the maximum value in each non-overlapping patch of size 2x2. This layer outputs a tensor of shape (30, 30, 16) with no parameters..
7. A fully connected layer with 256 neurons, using the ReLU activation function and L2 regularization with a penalty term of 0.01. This layer has 3,686,656 trainable parameters..
8. A layer that randomly drops out 20% of the inputs during training to prevent overfitting. This layer has no parameters..
9. A fully connected layer with 1 neuron, using the sigmoid activation function. This layer has 257 parameters..

APPENDIX V : Source code of data augmentation techniques used



Data Augementations 13/07/2023, 23:06

```
In [29]: from tensorflow.keras.preprocessing.image import ImageDataGenerator, arra
         import os
         from PIL import Image


         # Load the input image
         img = load_img('/Users/eastcdatalab/Desktop/b7.png')

In [30]: # Convert the image to a numpy array
         x = img_to_array(img)

         # Reshape the array to a batch of a single image
         x = x.reshape((1,) + x.shape)

         # Define the data augmentation parameters
         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')

         # Generate 100 new images from the input image
         i = 0
         for batch in datagen.flow(x, batch_size=1, save_to_dir='/Users/Ramadhan/R
             i += 1
             if i >= 100:
                 break

In [32]: ImageDataGenerator??

In [ ]:
```

file:///Users/eastcdatalab/Downloads/Data%20Augementations.html Page 1 of 1

APPENDIX VI : Research clearance letter

THE UNITED REPUBLIC OF TANZANIA

MINISTRY OF EDUCATION, SCIENCE AND TECHNOLOGY

THE OPEN UNIVERSITY OF TANZANIA

Ref. No OUT/ PG202000184 **10th January 2023**

Regional Administrative Secretary,
Dar es salaam Region,
P.O Box 5429,
DAR ES SALAAM.

Dear Regional Administrative Secretary,

RE: RESEARCH CLEARANCE FOR MR.RAMADHANI MRISHO HAMIS, REG NO: PG202000184

2. The Open University of Tanzania was established by an Act of Parliament No. 17 of 1992, which became operational on the 1stMarch 1993 by public notice No.55 in the official Gazette. The Act was however replaced by the Open University of Tanzania Charter of 2005, which became operational on 1stJanuary 2007.In line with the Charter, the Open University of Tanzania mission is to generate and apply knowledge through research.

3. To facilitate and to simplify research process therefore, the act empowers the Vice Chancellor of the Open University of Tanzania to issue research clearance, on behalf of the Government of Tanzania and Tanzania Commission for Science and Technology, to both its staff and students who are doing research in Tanzania. With this brief background, the purpose of this letter is to introduce to you **Mr. Ramadhani Mrisho Hamis, Reg. No: PG202000184)** pursuing **Master of Science in Computer Science (Msc computer science).** We here by grant this clearance to conduct a research titled **"Early Identification of Breast Cancer Using Convolutional Neural Networks".** He will collect his data at Muhimbili National Hospital in Dar es salaam Region from 11th January to 11th February 2023.

4. In case you need any further information, kindly do not hesitate to contact the Deputy Vice Chancellor (Academic) of the Open University of Tanzania, P.O.Box 23409, Dar es Salaam. Tel: 022-2-2668820.We lastly thank you in advance for your assumed cooperation and facilitation of this research academic activity.

Yours sincerely,

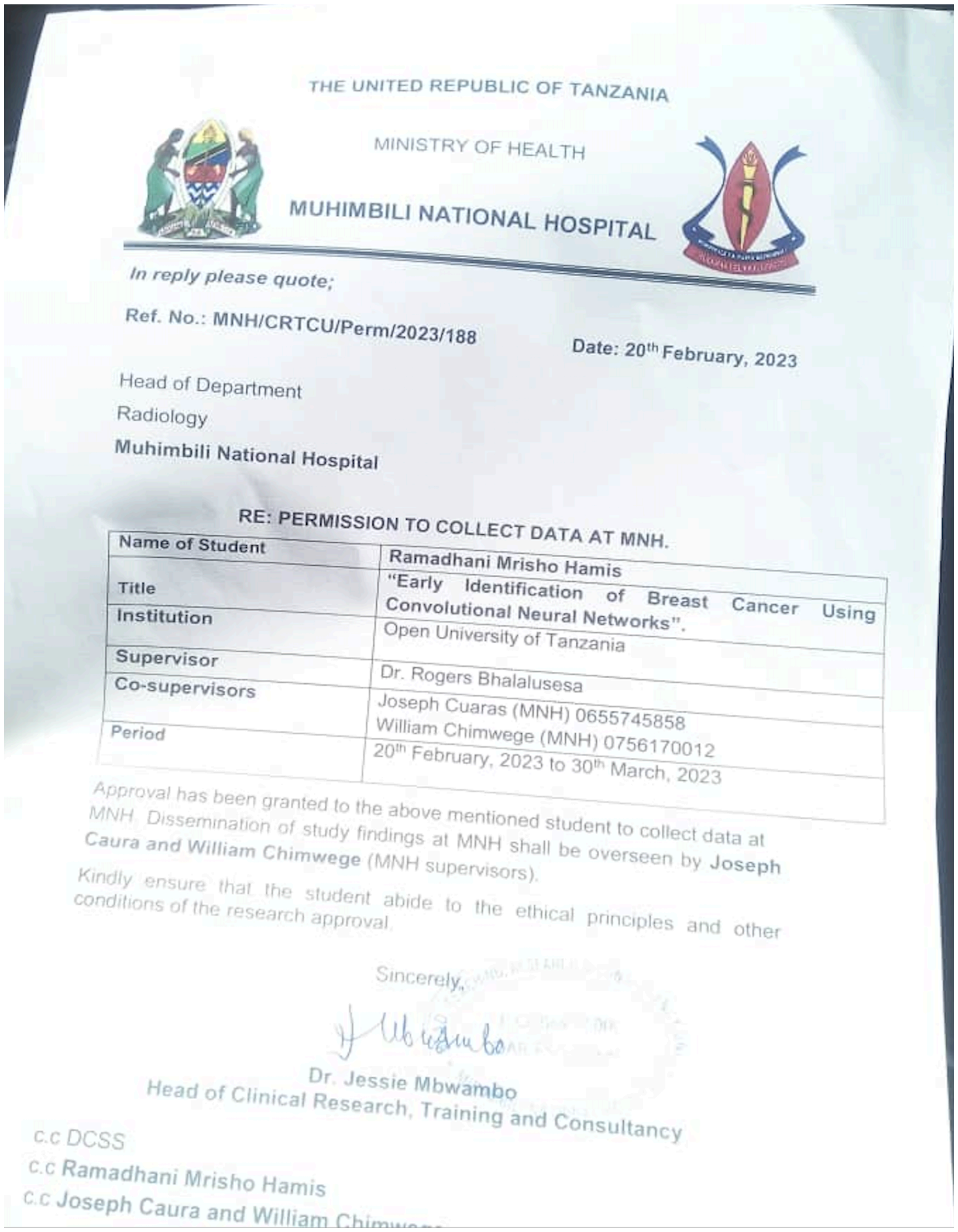
<u>THE OPEN UNIVERSITY OF TANZANIA</u>

[illegible]

Prof. Magreth S.Bushesha

For*:* <u>VICE CHANCELLOR</u>

APPENDIX VII : Data collection permit at MNH



THE UNITED REPUBLIC OF TANZANIA

MINISTRY OF HEALTH

MUHIMBILI NATIONAL HOSPITAL

In reply please quote;

Ref. No.: MNH/CRTCU/Perm/2023/188 **Date: 20th February, 2023**

Head of Department
Radiology
Muhimbili National Hospital

RE: PERMISSION TO COLLECT DATA AT MNH.

Name of Student	**Ramadhani Mrisho Hamis**
Title	**"Early Identification of Breast Cancer Using Convolutional Neural Networks".**
Institution	Open University of Tanzania
Supervisor	Dr. Rogers Bhalalusesa
Co-supervisors	Joseph Cuaras (MNH) 0655745858 William Chimwege (MNH) 0756170012
Period	20th February, 2023 to 30th March, 2023

Approval has been granted to the above mentioned student to collect data at MNH. Dissemination of study findings at MNH shall be overseen by **Joseph Caura and William Chimwege** (MNH supervisors).

Kindly ensure that the student abide to the ethical principles and other conditions of the research approval.

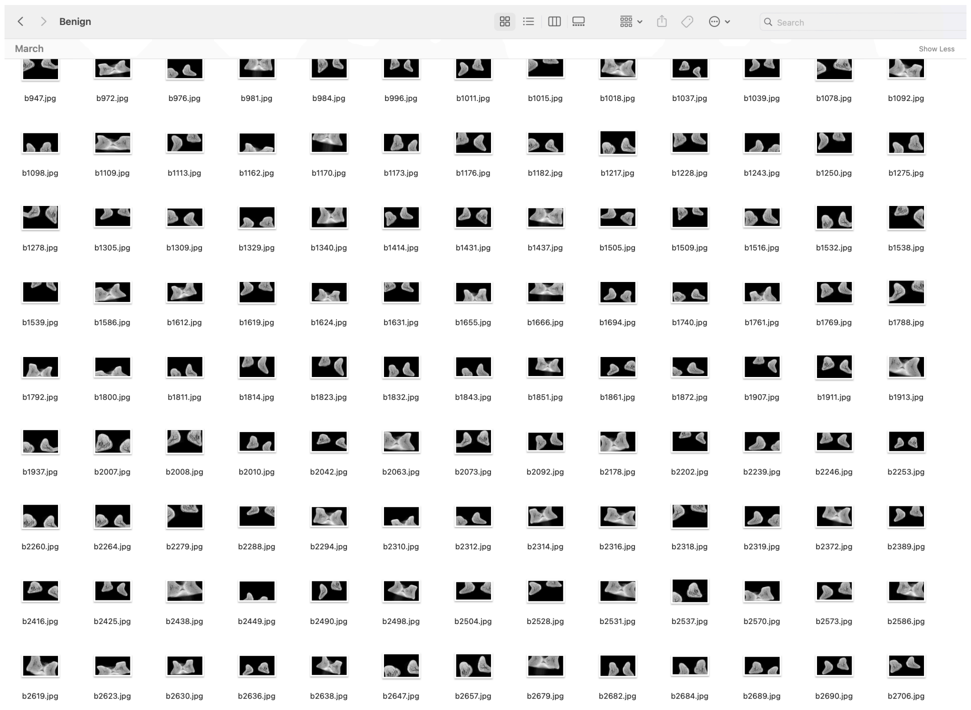
Sincerely,

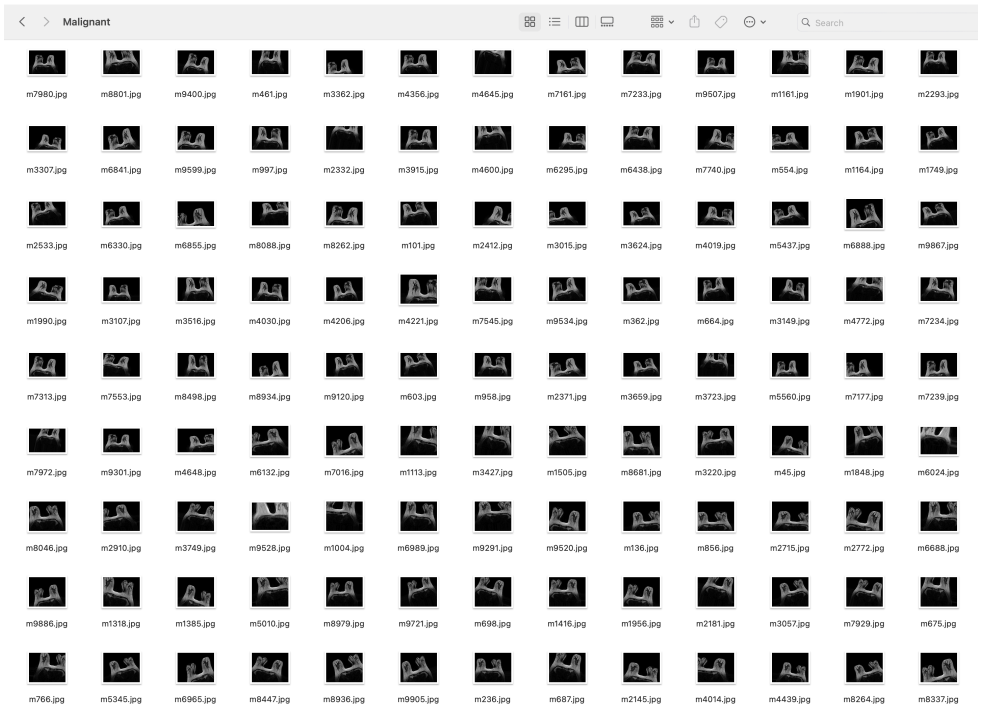
[illegible]

Dr. Jessie Mbwambo
Head of Clinical Research, Training and Consultancy

c.c DCSS
c.c Ramadhani Mrisho Hamis
c.c Joseph Caura and William Chimw

APPENDIX VIII : Image datasets used to build the model


Benign
Search
March
Show Less
b947.jpg b972.jpg b976.jpg b981.jpg b984.jpg b996.jpg b1011.jpg b1015.jpg b1018.jpg b1037.jpg b1039.jpg b1078.jpg b1092.jpg
b1098.jpg b1109.jpg b1113.jpg b1162.jpg b1170.jpg b1173.jpg b1176.jpg b1182.jpg b1217.jpg b1228.jpg b1243.jpg b1250.jpg b1275.jpg
b1278.jpg b1305.jpg b1309.jpg b1329.jpg b1340.jpg b1414.jpg b1431.jpg b1437.jpg b1505.jpg b1509.jpg b1516.jpg b1532.jpg b1538.jpg
b1539.jpg b1586.jpg b1612.jpg b1619.jpg b1624.jpg b1631.jpg b1655.jpg b1666.jpg b1694.jpg b1740.jpg b1761.jpg b1769.jpg b1788.jpg
b1792.jpg b1800.jpg b1811.jpg b1814.jpg b1823.jpg b1832.jpg b1843.jpg b1851.jpg b1861.jpg b1872.jpg b1907.jpg b1911.jpg b1913.jpg
b1937.jpg b2007.jpg b2008.jpg b2010.jpg b2042.jpg b2063.jpg b2073.jpg b2092.jpg b2178.jpg b2202.jpg b2239.jpg b2246.jpg b2253.jpg
b2260.jpg b2264.jpg b2279.jpg b2288.jpg b2294.jpg b2310.jpg b2312.jpg b2314.jpg b2316.jpg b2318.jpg b2319.jpg b2372.jpg b2389.jpg
b2416.jpg b2425.jpg b2438.jpg b2449.jpg b2490.jpg b2498.jpg b2504.jpg b2528.jpg b2531.jpg b2537.jpg b2570.jpg b2573.jpg b2586.jpg
b2619.jpg b2623.jpg b2630.jpg b2636.jpg b2638.jpg b2647.jpg b2657.jpg b2679.jpg b2682.jpg b2684.jpg b2689.jpg b2690.jpg b2706.jpg


Malignant
Search
m7980.jpg m8801.jpg m9400.jpg m461.jpg m3362.jpg m4356.jpg m4645.jpg m7161.jpg m7233.jpg m9507.jpg m1161.jpg m1901.jpg m2293.jpg
m3307.jpg m6841.jpg m9599.jpg m997.jpg m2332.jpg m3915.jpg m4600.jpg m6295.jpg m6438.jpg m7740.jpg m554.jpg m1164.jpg m1749.jpg
m2533.jpg m6330.jpg m6855.jpg m8088.jpg m8262.jpg m101.jpg m2412.jpg m3015.jpg m3624.jpg m4019.jpg m5437.jpg m6888.jpg m9867.jpg
m1990.jpg m3107.jpg m3516.jpg m4030.jpg m4206.jpg m4221.jpg m7545.jpg m9534.jpg m362.jpg m664.jpg m3149.jpg m4772.jpg m7234.jpg
m7313.jpg m7553.jpg m8498.jpg m8934.jpg m9120.jpg m603.jpg m958.jpg m2371.jpg m3659.jpg m3723.jpg m5560.jpg m7177.jpg m7239.jpg
m7972.jpg m9301.jpg m4648.jpg m6132.jpg m7016.jpg m1113.jpg m3427.jpg m1505.jpg m8681.jpg m3220.jpg m45.jpg m1848.jpg m6024.jpg
m8046.jpg m2910.jpg m3749.jpg m9528.jpg m1004.jpg m6989.jpg m9291.jpg m9520.jpg m136.jpg m856.jpg m2715.jpg m2772.jpg m6688.jpg
m9886.jpg m1318.jpg m1385.jpg m5010.jpg m8979.jpg m9721.jpg m698.jpg m1416.jpg m1956.jpg m2181.jpg m3057.jpg m7929.jpg m675.jpg
m766.jpg m5345.jpg m6965.jpg m8447.jpg m8936.jpg m9905.jpg m236.jpg m687.jpg m2145.jpg m4014.jpg m4439.jpg m8264.jpg m8337.jpg