MICROSCOPIC BREAST-IMAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK (CNN)

A project report submitted in

The partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science and Engineering

Submitted by

D.V.R SANJAY VARMA	**(18KD1A0542)**
CHENNURU VARSHITH	**(18KD1A0533)**
B.S.R SANDEEP	**(18KD1A0519)**
B.V.V.S. SRILEKHA	**(18KD1A0516)**

Under the guidance of

Mr. D. MADHU BABU, M. Tech (Ph.D.)

Associate Professor

DEPARTMENT OF CSE

DEPARTMENT OF COMPUTER SCIENCE and ENGINEERING

LENDI INSTITUTE OF ENGINEERING & TECHNOLOGY

An Autonomous Institution, Accredited by NBA & NAAC with "A" Grade
Permanently Affiliated to JNTUK, Approved by A.I.C.T.E,
Jonnada, Vizianagaram Dist. – 535005.

2018 – 2022

LENDI INSTITUTE OF ENGINEERING & TECHNOLOGY

AN AUTONOMOUS INSTITUTION

Accredited by NAAC with "A" Grade, Accredited by NBA Permanently Affiliated to JNTUK, Approved by A.I.C.T.E, Jonnada, Vizianagaram Dist. - 535005.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project entitled **"Microscopic Breast-Image Classification using CNN"** is a bonafide record of the work done by **D.V.R Sanjay Varma (18KD1A0542), Chennuru Varshith (18KD1A0533), B.S.R Sandeep(18KD1A0519), B.V.V.S Srilekha (18KD1A0516)** under the supervision and guidance of **Mr. D. MADHU BABU, M. Tech, (Ph.D.), Associate Professor** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** from Lendi Institute of Engineering and Technology (Affiliated to JNTUK), Jonnada, Vizianagaram for the Year 2022.

INTERNAL GUIDE

Mr. D. Madhu Babu, M. Tech, (Ph.D.)
Associate Professor,
Department of C.S.E.

HEAD OF THE DEPATMENT

Dr. A. Rama Rao, M. Tech, Ph.D.
Professor,
Department of C.S.E.

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

With great solemnity and sincerity, we express our deepest sense of gratitude and pay our sincere thanks to our guide **Mr. D. Madhu Babu, Associate Professor,** Department of **Computer Science and Engineering** who evinced keen interest in our efforts and provided her valuable guidance throughout our project work.

We thank our project coordinator, **Dr. Sk. NAGUL, Mr. U. KARTHEEK CHANDRA PATNAIK, & Mrs. R. SUCHITRA** Department of **Computer Science and Engineering** has made their support available in several ways and helped us to complete our project work correctly.

We thank our **Prof. A. RAMA RAO, Head of the Department of Computer Science and Engineering** helped us to complete our project work in a truthful method.

We thank our gratitude to our **Principal, Dr. V. V. RAMA REDDY,** for his kind attention and valuable guidance to us throughout this course in carrying out the project.

We wish to express gratitude to our **Management Members** who supported us in providing a good lab facility.

We are thankful to **All Staff Members of the Department of Computer Science and Engineering,** for helping us directly / indirectly to complete this project work by giving valuable suggestions.

All of the above we great fully acknowledge and express our thanks to our parents who have been instrumental in the success of this project and play a vital role.

D.V.R SANJAY VARMA **(18KD1A0542)**

CHENNURU VARSHITH **(18KD1A0533)**

B.S.R Sandeep **(18KD1A0519)**

B.V.V.S. Srilekha **(18KD1A0516)**

DECLARATION

We hereby declare that the project work entitled **"Microscopic Breast-Image Classification using Convolutional Neural Network (CNN)"** submitted to the JNTU Kakinada is a record of an original work done by **D.V.R Sanjay Varma (18KD1A0542), Chennuru Varshith (18KD1A0533), B.S.R Sandeep (18KD1A0519), B.V.V.S Srilekha (18KD1A0516)** under the esteemed guidance of **Mr. D. MADHU BABU, Associate Professor,** Computer Science and Engineering, Lendi Institute of Engineering & Technology. This project work is submitted in partial fulfillment of the requirements for the award of the degree **Bachelor of Technology** in **Computer Science and Engineering**. This entire project is done to the best of our knowledge and is not submitted to any University for the award of a degree/diploma.

D.V.R SANJAY VARMA **(18KD1A0542)**

CHENNURU VARSHITH **(18KD1A0533)**

B.S.R SANDEEP **(18KD1A0519)**

B.V.V.S SRILEKHA **(18KD1A0516)**

LENDI INSTITUTE OF ENGINEERING & TECHNOLOGY AN AUTONOMOUS INSTITUTION

Accredited by NAAC with "A" Grade, Accredited by NBA Permanently Affiliated to JNTUK, Approved by A.I.C.T.E, Jonnada,

Vizianagaram Dist. - 535005.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To be a frontier in computing technologies to produce globally competent computer science engineering graduates with moral values to build a vibrant society and nation.

MISSION

- Providing a strong theoretical and practical background in computer science engineering with an emphasis on software development.
- Inculcating professional behavior, strong ethical values, innovative research capabilities, and leadership abilities.
- Imparting the technical skills necessary for continued learning towards their professional growth and contribution to society and rural communities.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO-1: Graduates will have strong knowledge and skills to comprehend the latest tools and techniques of Computer Engineering so that they can analyze, sign and create computing products and solutions for real-life problems.

PEO-2: Graduates shall have a multidisciplinary approach, professional attitude and ethics, communication and teamwork skills, and an ability to relate to and solve social issues through their Employment, Higher Studies, and Research.

PEO-3: Graduates will engage in life-long learning and professional development to adapt to rapidly changing technology.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-1: Ability to grasp advanced programming techniques to solve contemporary issues.

PSO-2: Have knowledge and expertise to analyze data and networks using the latest tools and technologies.

PSO-3: Qualify in national and international competitive examinations for successful higher studies and employment.

PROGRAM OUTCOMES (POs)

PO-1 Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-2 Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using the first principles of mathematics, natural sciences, and engineering sciences.

PO-3 Design/development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and the cultural, societal, and environmental considerations.

PO-4 Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data, and synthesis of the information to provide valid conclusions.

PO-5 Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO-6 The Engineer and Society: Apply to reason informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-7 Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-8 Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO-9 Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and multidisciplinary settings.

PO-10 Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO-11 Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12 Life-Long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

Breast Cancer is a serious threat and one of the largest causes of death of women throughout the world. The identification of cancer largely depends on digital biomedical photography analysis such as histopathological images by doctors and physicians.

Microscopic Breast-Image Classification is the ability to Identify the malignancy of tissues from microscopic images. It has always been an issue and time-consuming to identify malignancy of tissues for concerned doctors and radiologists. The Histopathological Breast-Image using Classification Using Local and Frequency Domains by Convolutional Neural Network (CNN) is the solution to this problem which uses the Histopathological image and identifies the malignancy of tissues.

In this project, we are going to implement a computer-aided diagnosis to identify the tissues that are affected by the cancer virus.

Keywords: Classification; Convolutional Neural Network; Fourier Transform.

Outcomes:

Our project titled "**Microscopic Breast-Image Classification Using Convolutional Neural Network (CNN)**" is mapped with the following outcomes.

Program Outcomes: PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO9, PO10, PO11, PO12

Program Specific Outcomes: PSO1, PSO2, PSO3

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1

INTRODUCTION

1. INTRODUCTION

Thousands of females fall victim to breast cancer every year. The human body comprises millions of cells each with its unique function. When there is the unregulated growth of the cells it is termed cancer. In this, cells divide and grow uncontrollably, forming an abnormal swelling tissue part called a tumor. Tumor cells grow and invade digestive, nervous, and circulatory systems disrupting the body's normal functioning. Though every single tumor is not cancerous. Cancer is classified by the type of cell that is affected and more than 200 types of cancers are known. This paper is focused on Breast cancer. Breast cancer is the most common type of cancer among females across the world. As per the National Breast Cancer Foundation, "Breast cancer is the most commonly diagnosed cancer in women". Breast cancer is the second largest cause of cancer death among women. Women generally approach clinics with mild to serious pain in their breasts. After the examination of the breasts, the doctors usually suggest an ultrasound scan. The proceedings after the scan are more painful. Some women feel that the pain is intensely increased only after clinical proceedings. It is because of the painful process that is followed to detect whether the lymph node is malignant or benign. Malignant tumors are harmful or cancerous and benign tumors are harmless and can be removed through surgery. The treatment is then decided after thoroughly examining the state of the tumor. With the help of this paper, the detection of the state of the tumor can be decided with the help of a convolutional neural network.

1.1 PROBLEM OVERVIEW

In today's scenario, we have too much delay and inaccuracy in the diagnosis results provided by clinical centers. There are many cases where patients are given wrong inputs regarding their diagnosis and face false-positive or false-negative results which results in either poring the money unnecessarily or even paying for the death due to delay of treatment. To achieve these disadvantages of traditional methodology, our system proposes an easy approach for clinical examination for breast cancer diagnosis prediction. Using machine learning algorithms the classifier will be training by the dataset (ultrasound scan values) and the classifier will classify the new record values as either benign or malignant tumors. Higher accuracy results will be obtained than traditional methods, which also reduces the patient's waiting time and increases the life rate of people.

1.2 PROJECT DELIVERABLES

We have used our proposed machine learning, based convolutional neural network algorithm for the detection of cancer using the present technology i.e. machine learning, CNN techniques, and the previous dataset record. Using this technology classifies whether the patient is suffering from a benign or malignant tumor. This model can be modified with new training data without having to rebuild the model. This proposed system yields accurate results.

1.3 PROJECT SCOPE

An important part of our information-gathering behavior has always been to find out how the patients and the doctors are benefited. With the growing availability and popularity of different and advanced medical equipment, every organization looks for better quality using the better equipment, and from the patient's point of view, they are always in need of good and advanced medical equipment that helps them in beating the disease. And patients can be treated with a lower advanced medical equipment to the most advanced medical equipment that is in use nowadays, but every patient wants to and should be considered the advanced equipment and at the same time one might not be able to afford such advanced medical equipment which might result in the late identification of the disease he is suffering from which in turn affects their health eventually results to his/her demise. Hence, a machine is built to avoid this and provide all the patients with the equipment to identify and detect the disease in their early stages.

Detection is the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome. Detection is used in various medical fields. Detection analysis is to use the knowledge of what has happened to provide the best valuation of what will happen. Collecting data and analyzing based on requirements by applying accurate techniques will lead to better detection of data with efficient prediction.

CHAPTER – 2

LITERATURE SURVEY

2. LITERATURE SURVEY

Andrea Duggento[1] Breast cancer is one of the most common cancers in women, with more than 1,300,000 cases and 450,000 deaths each year worldwide. In this context, recent studies showed that early breast cancer detection, along with suitable treatment, could significantly reduce breast cancer death rates in the long term. X-ray mammography is still the instrument of choice in breast cancer screening. In this context, the false-positive and false-negative rates commonly achieved by radiologists are extremely arduous to estimate and control although some authors have estimated figures of up to 20% of total diagnoses or more. We explore a total of 260 model architectures in a train-validation-test split to propose a model selection criterion that can pose the emphasis reducing false negatives while still retaining acceptable accuracy. We achieve an area under the receiver operatic characteristics curve of 0.785 (accuracy 71.19%) on the test set.

Jyotismita Talukdar[2] Breast cancer has become the primary reason for death in women in developed countries. Breast cancer is the second most common cause of cancer death in women worldwide. The high incidence of breast cancer in women has increased significantly during the last few decades. In this paper, we have discussed various data mining approaches that have been utilized for the early detection of breast cancer. Breast Cancer Diagnosis is the distinguishing of benign from malignant breast lumps. We have approached the diagnosis of this disease by using the Data mining technique. Data mining is an essential step in the process of knowledge discovery in databases in which intelligent methods are applied to extract patterns. The most effective way to reduce breast cancer deaths is to detect it earlier. This paper discusses the early detection of breast cancer in three major steps of determining breast cancer.

E. D. Pisano[3] There is now general agreement that screening mammography reduces the rate of death from breast cancer among women who are 40 years of age or older. Meta-analyses of eight large, randomized trials found a reduction in the mortality rate of 16 to 35 percent among women 50 to 69 years of age who were assigned to screening mammography, whereas women who were 40 to 49 years of age at entry had a smaller but significant reduction of 15 to 20 percent. A detailed account of the design of DMIST has been published previously. This trial was conducted by the American College of Radiology Imaging Network. During two years, 49,528 women were recruited for the study at 33 sites. The protocol was approved by the

institutional review boards at all sites. All women gave written informed consent. The study was monitored by a data and safety monitoring board. Women who presented for screening mammography at the study sites were eligible to participate unless they reported symptoms, had breast implants, believed they might be pregnant, had undergone mammography for any purpose within the preceding 11 months, or had a history of breast cancer treated with both lumpectomy and radiation.

R. L. N. Godone[4] Breast Cancer is a complex disease characterized by the occurrence of multiple molecular alterations. Currently, some molecular markers are in use for breast cancer diagnostics, prognostic, and predictive purposes. Thus, genetic signatures are available for improving decision-making. The biomarkers are also essential as therapeutic approaches, but many questions remain due to the lack of efficacy in breast cancer treatment, mainly for the triple-negative breast cancer subtype. Since the genetic profile of breast cancer can also be related to different ethnic groups and geographic areas, the reference populations of the genetic assays and clinical trials need to include a broader population beyond the European and North American patients. In this review, we analyzed the current and potential molecular markers that could help to improve the strategies for breast cancer therapy.

CHAPTER – 3

PROBLEM ANALYSIS

3. PROBLEM ANALYSIS

3.1 EXISTING SYSTEM

An Ad Hoc Random Initialization Deep Neural Network Architecture for Discriminating Malignant Breast Cancer Lesions in Mammographic Images

Breast cancer is one of the most common cancers in women, with more than 1,300,000 cases and 450,000 deaths each year worldwide. In this context, recent studies showed that early breast cancer detection, along with suitable treatment, could significantly reduce breast cancer death rates in the long term. X-ray mammography is still the instrument of choice in breast cancer screening. In this context, the false-positive and false-negative rates commonly achieved by radiologists are extremely arduous to estimate and control although some authors have estimated figures of up to 20% of total diagnoses or more. The introduction of novel artificial intelligence (AI) technologies applied to the diagnosis and, possibly, the prognosis of breast cancer could revolutionize the current status of the management of breast cancer patients by assisting the radiologist in clinical image interpretation. Lately, a breakthrough in the AI field has been brought about by the introduction of deep learning techniques in general and convolutional neural networks in particular. Such techniques require no prior feature space definition from the operator and can achieve classification performances that can even surpass human experts. In this paper, we design and validate an ad hoc CNN architecture specialized in breast lesion classification from imaging data only. We explore a total of 260 model architectures in a train-validation-test split to propose a model selection criterion that can pose the emphasis reducing false negatives while still retaining acceptable accuracy. We achieve an area under the receiver operatic characteristics curve of 0.785 (accuracy 71.19%) on the test set, demonstrating how an ad hoc random initialization architecture can and should be fine-tuned to a specific problem, especially in biomedical applications.

3.1.1 Limitations of the Existing System

1. In this system, the model collects the mammograms from various sources and this may lead to noise data
2. Classes May Become imbalance and the model might lean toward favoring one class only
3. Which in turn leads to False-negative results that can lead to delays in treatment.
4. False-positive results would let the patient go through unwanted painful and expensive procedures.

3.2 PROPOSED SYSTEM

To overcome the drawbacks of the existing method we are proceeding with a Convolutional Neural Network. CNN is one of the most widely used techniques for the processing of images with the utmost accuracy. The goal is to assess whether a lump in a breast could be malignant (cancerous) or benign (non-cancerous). For instance, cancer cells tend to vary in size and shape. So uniformity of cell size/shape, bare nuclei, bland chromatin, and normal nucleoli are signs of benignity. The analysis is probably part of a triple test. If one or both of the other tests suggest malignancy, a biopsy for histological analysis will be necessary. It might be done as a frozen section analysis during the surgical procedure. In this project, the main aim is to detect whether the patient is having benign cells or malignant tissue cells. This detection helps both patients, and radiologists save time and predict the disease at an early stage.

Step – 1: First, we need to collect the data set and required libraries.
Step – 2: Now, we need to identify the total number of images that are present for training the data and organize them into a data frame.
Step – 3: After identifying the total number of images and organizing them into a data frame, we will check cancer and non-cancerous tissues. And now we will balance the class distribution so that the neural network does not lean on favoring only one class.
Step – 4: Now, we will set up the image generators and train the model and evaluate for loss and accuracy metrics and plot the training curves based on the loss and accuracy.
Step – 5: Finally, we will predict the value set and calculate the AUC score and the result.

3.2.1 Advantages of the Proposed System

1) The cost of the learning process is zero. The model can be modified with a new dataset without having to rebuild the model.
2) If datasets are huge, with the help of pandas the entire datasets can be made into one directory and performed on the entire directory without any issues.
3) The most important advantage of the convolutional neural network is it helps to process the image with utmost accuracy by classifying them into one single directory. This will further help in understanding the difference between benign tissue cells and cancer tissue cells.

4) Convolutional Neural Network performs better detection from multiple detections and it can avoid picking/depending on a single predictor.

5) The ability to determine the required data from the noise data and perform the detection.

6) A benefit is if the network is trained with the different images of data, the machine will be able to detect the benign or malignant tissues present in the data apart from the datasets used for training and the response should your goal be a prediction.

7) This proposed system yields better and more efficient accurate results.

CHAPTER – 4

SYSTEM ANALYSIS

4. SYSTEM ANALYSIS

Systems Analysis is a problem-solving technique that decomposes a system into its component pieces studying how well those parts work and interact to accomplish their purpose. System Analysis is the process of studying a procedure to identify its goals and purposes and create systems and procedures that will efficiently achieve them. Analysis and Synthesis, as scientific methods, always go hand, and they complement one another.

An analysis is defined as the procedure by which we break down an intellectual or substantial whole into parts. Synthesis is defined as the procedure by which we combine separate elements or components to form a coherent whole. Analysis can also be defined as a series of components that perform organic functions together. Systems Analysis Researchers apply a methodology to the analysis of systems involved to form an overall picture. System Analysis is used in every field where there is work on developing something.

The development of a computer-based information system includes a systems analysis phase which produces or enhances the data model which itself is a precursor to creating or enhancing a database. There are several different approaches to system analysis. When a computer-based information system is developed, systems analysis would constitute the following steps:

➢ The development of a feasibility study, involves determining whether a project is economically, socially, technologically, and organizationally feasible.

➢Conducting fact-finding measures, designed to ascertain the requirements of the system's end–users. These typically span interviews, questionnaires, or visual observations of work on the existing system.

➢Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for, and so on. An example of System Analysis can be System Engineering. Systems Engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed. A System Requirement Specification is a System Document that captures the complete description of how the system is expected to perform. The System Requirement Specification is usually signed off at the end of the requirement engineering phase. It is the means of translating the ideas in the minds of clients (the input), and into a

formal document (the output of the requirement phase). So, the purpose, the scope, and the objective of our proposed System for the microscopic image dataset can be defined as follows:

1. **Purpose:** The Neural Network's purpose is to deal with different types of datasets to detect benign or malignant cancer tissue cells.

2. **Scope:** Our project is to use different microscopic images in a whole dataset and to train the machine and make it detect the disease them using CNN based approach. Here we have different tissue cell images of affected or unaffected cells details in a dataset and detect the info as per requirement. Our project yields accurate results.

3. **Objective**: The objective is to find out the tissue cells affected by cancerous virus based on different types of affected and unaffected tissue cell images as per their requirement.

4.1 SOFTWARE REQUIREMENTS SPECIFICATIONS

A Software Requirements Specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. Software requirements specifications permit a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Using appropriate software requirements specifications can help prevent software project failure.

4.1.1 Functional Requirements

Functional Requirements describe what the system should do, i.e., the services provided for the users and other systems. The Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks, or functions the system is required to perform. In particular, it is not just the functional requirements of the first product or release that must be supported by the architecture. The Functional Requirements

of early release need to be explicitly taken. The various Functional requirements that are observed from our proposed system can be given as follows:

a. Recall
b. F-1 score
c. Confusion Matrix
d. Prediction of Result

a) Recall:

The recall is the ratio of correctly predicted positive observations to all the observations in the actual class.

b) F-1 Score:

F1-Score is the weighted average of Precision and Recall.

$$F1 = \frac{2 * (Recall * Precision)}{(Recall + Precision)}$$

Fig 4.1.1 F1 Score Formula

The higher the F1-Score, the better the model. For all three metrics, 0 is the worst while 1 is the best.

c) Confusion Matrix:

Confusion Matrix is a very important metric when analyzing misclassification. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. The diagonals represent the classes that have been correctly classified. This helps as we not only know which classes are being misclassified but also what they are being misclassified as.

d) Prediction of Result:

After calculating and training the network with the help of an algorithm we acquire the result of whether the provided tissue image consists of cancer or not.

4.1.2 Non – Functional Requirements

Non – Functional Requirements are the system qualities that capture the required properties of the system, such as performance, security, maintainability, etc., and in other words, how well some behavioral or structural aspect of the system should be accomplished. The various Non – Functional Requirements that are observed from our Proposed System can be given as follows:

a. Reliability
b. Performance
c. Scalability and Availability
d. Accuracy

a) Reliability:

Reliability is the ability of a system to perform its required functions under stated conditions for a specific period, constraints on the run-time behaviour of the system. This proposed system yields accurate results.

b) Performance:

The cost of the learning process is zero. If the image is blurry with the help of the CNN network the pixelated images are clear then the images are used to train the network the model can be modified with new training data without having to rebuild the model.

c) Scalability and Availability:

Availability is the ability to use the model that performs under any specific condition based on the requirement. This approach can be used and predicted based on available data. The model can be applied to any type of dataset that represents the Scalability.

d) Accuracy:

The main purpose of using the Convolutional Neural Network model is to attain accurate results. This technique is highly used and performed for its efficient performance and accurate result.

4.2 FEASIBILITY STUDY

A feasibility Study reports a process of evaluating alternative systems through Cost / Benefit Analysis so that the most feasible and desirable system can be selected for development. Due to the drawback in the existing system which requires a huge database. The main objective of the Feasibility Study is to test the Technical, Operational, and economic feasibility of adding new modules and debugging old running systems. All system is feasible if they are unlimited resources and infinite time. There were '3' main aspects in the Feasibility Study portion of the preliminary investigation:

1. Economic Feasibility.
2. Technical Feasibility.
3. Operational Feasibility.

4.2.1 Economic Feasibility

A system that can be developed technically and that will be used if installed must still be a good investment for the organization. In the Economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. Economic Feasibility includes an evolution of all the international costs and benefits expected if a proposed system is implemented. The financial and economic factors are considered in This stage. The application has been developed using PYTHON and Jupyter notebook concepts which is open-source software. We just require a dataset and directories that are created in a local computer system which is not much costlier. Hence the application of our system is economically feasible.

4.2.2 Technical Feasibility

Technical feasibility is considered with hardware and software. Hence, the proposed system is feasible with the expected computer hardware and software requirements. As application using PYTHON IDLE, back end as Jupyter notebook which utilizes minimum resources of the personal computer. This project is technically feasible.

4.2.3 Operational Feasibility

Considering all the drawbacks of the existing system, the management has given sufficient support to the system. The facilities given/defined in the project are acceptable by the user. The implementation of the system is satisfying all the conditions and norms of the users. Therefore, the project is considered operational and feasible.

4.3 SYSTEM REQUIREMENTS

System Requirements Specification is a detailed statement of the effects that a system is required to achieve. A good specification gives a complete statement of what the system is to do, without making any commitment as to how the system is to do it. A system requirements specification is normally produced in response to user requirements specifications or other expressions of requirements and is then used as the basis for system design. The system requirements specification typically differs from the hardware and software components of a computer system these are required to install and use software efficiently. The software manufacturer will list the system requirements on the software package. If your computer system does not meet the system requirements then the software may not work correctly after installation. System requirements for operating systems will be hardware components, while other application software will list both hardware and operating system requirements. System requirements are most commonly seen listed as minimum and recommended requirements. The minimum system requirements need to be met for the software to run at all on your system, and the recommended system requirements, if met, will offer better software usability

4.3.1 SOFTWARE REQUIREMENTS

Software Requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Operating System: Windows 10.
Coding Language: PYTHON Programming.
Dataset: Mammogram images.
Tool: Jupyter notebook.

4.4.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, a hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in the case of operating systems. An HCL lists tested compatible, and sometimes incompatible hardware devices for a particular operating system or applications. The following sub-sections discuss the various aspects of hardware requirements for our proposed system are:

System Processor: Intel Core i5, RYZEN-5.
Hard Disk: 1 TB.
Input Devices: Keyboard, and Mouse.
RAM: 8 GB.

CHAPTER – 5

SYSTEM DESIGN

5. SYSTEM DESIGN

5.1 INTRODUCTION

System Design is the process of defining the elements of a system such as the architecture modules and components, the different interfaces of those components, and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well–running system. Systems Design mainly concentrates on defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. Systems Design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way, the process is systematic wherein it takes into account all related variables of the system that needs to be created — from the architecture to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering, and systems architecture. We typically start with the candidate objects defined during analysis but add much more rigor to their definitions. The following benefits like:

1. **Maintainability:** Maintainability through simplified mapping to the problem domain, which provides for less analysis effort, less complexity in system design, and easier verification by the users.
2. **Reusability**: Reusability of design artifacts, which saves time and costs.
3. **Productivity:** Productivity gains through the direct mapping of the features of various python programming

5.1.1 CLASS DIAGRAM

The Class Diagram is a Static Diagram. It represents the static view of an application. Class Diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. It describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams that can be mapped directly with object-oriented languages.

The Class Diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a “Structural Diagram”.

PURPOSE

The purpose of the class diagram is to model the static view of an application. The class diagrams are only diagrams that can be directly mapped with object-oriented languages and thus are widely used at the time of construction.

The UML diagrams like activity diagrams and sequence diagrams can only give the sequence flow of the application but the class diagram is a bit different. So, it is the most popular UML diagram in the coder community. So, the purpose of the class diagram can be summarized as:

- Analysis and Design of the static view of an application,
- To describe the Responsibilities of a System,
- Base for Component and Deployment Diagrams, and
- Forward and Reverse Engineering.

VISIBILITY

Use visibility markers to signify who can access the information which is in a class. There are four visibilities and can be summarized as:

- Private visibility hides information from anything outside the class partition,
- Public visibility allows all other classes to view the marked information, and
- Protected visibility allows child classes to access information that is inherited from a parent class.

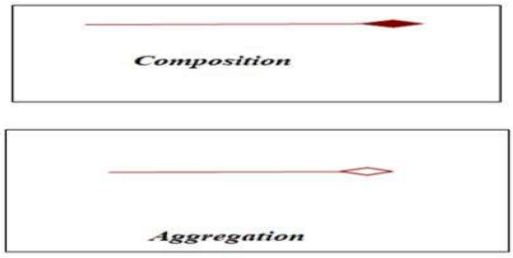
ASSOCIATIONS

Associations represent a static relationship between the classes. Place the association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place rolls at the end of an association. Roles represent how the two classes see each other.

COMPOSITION AND AGGREGATION

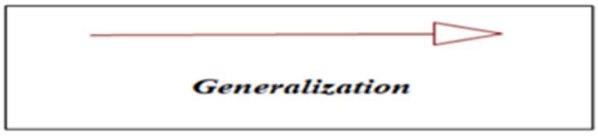
Composition and Aggregation link a semantic association between two classes in the UML diagram.

They are used in the class diagram. They both differ in their symbols.



GENERALIZATION

It is a specification relationship in which objects of the specialized element (the child) are suitable for objects of the generalization element (the parent). It is used in the class diagram.



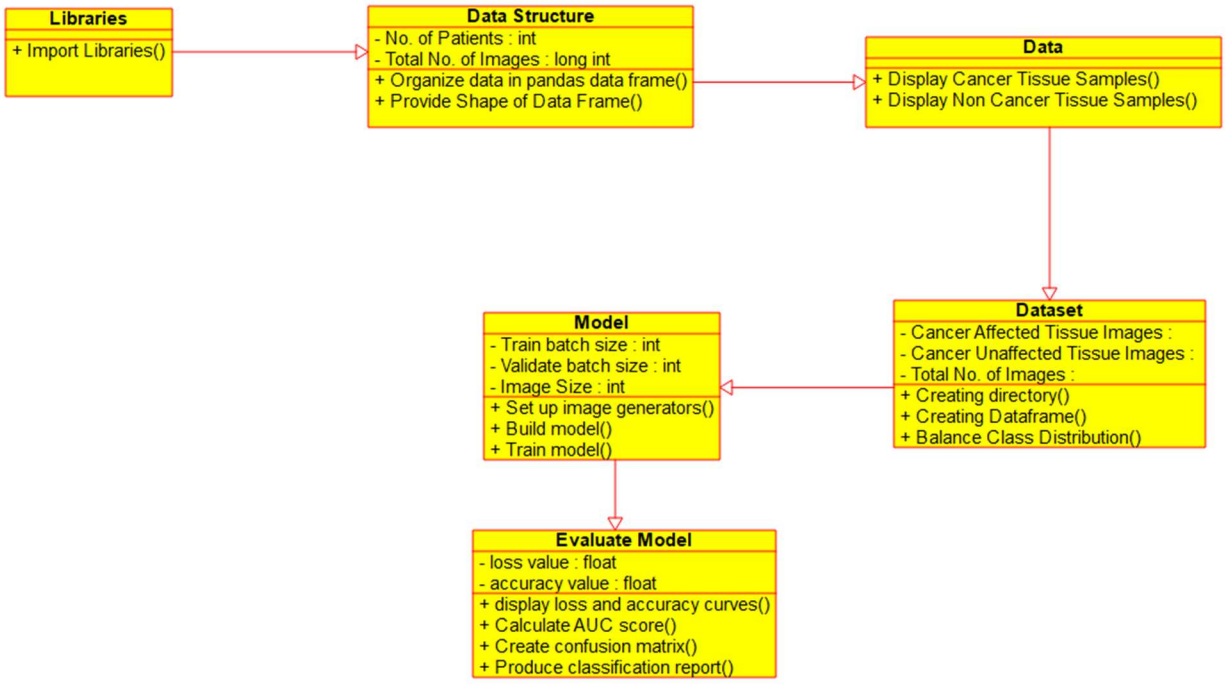


Fig 5.1.1 Class Diagram for Breast Cancer Detection

5.1.2 ACTIVITY DIAGRAM

Activity Diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modelling Language (UML), Activity Diagrams are intended to model both computational and organizational processes (i.e., Workflows). Activity Diagrams show the overall flow of control. Activity Diagrams can be used to model higher-level business processes at the business unit level, or to model low–level internal class actions. Activity Diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types of Activity Diagrams are:

• Rounded rectangles represent actions,

• Diamonds represent decisions,

A black circle represents the start (initial state) of the workflow, and

• An encircled black circle represents the end (final state).

Activity Diagrams may be regarded as a form of a flowchart. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

Action State	It represents the execution of atomic action. An action state is a simple state with an entry action whose only exit transition is triggered by the implicit event of completing the execution of the entry action.	**Action Name**
Initial State	An initial node is a control node at which flow starts when the activity is invoked. An activity may have more than one initial state.	
Transition	Transition is used to show the flow of action from one state to another.	
Final State	A final note is control at which flow starts when the activity is invoked.	

Table 5.2 Graphical Representation of Activity Diagram

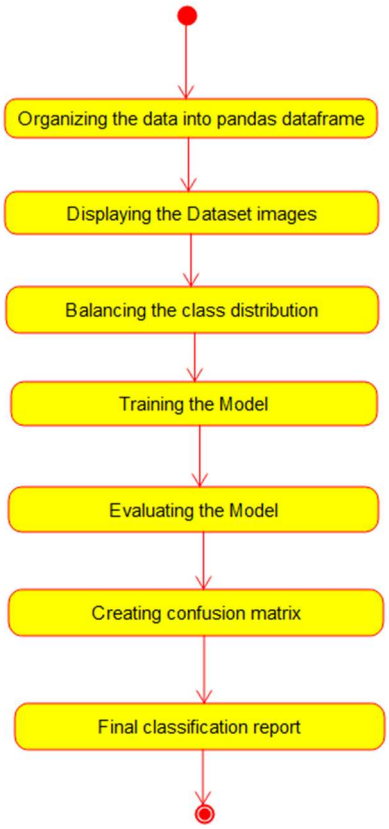


Fig 5.1.2 Activity Diagram for Breast Cancer Detection

5.2 SYSTEM ARCHITECTURE

The System Architecture provides the core design of the proposed system. In this system, datasets consisting of microscopic images of cancer-affected and unaffected tissues are taken as input, and process the data at preprocessing stage, then this data will be fed to the ROI segmentation phase, which does the image retrieval process, and in the next phase, the feature extraction of images is done and then it is provided to the CNN model where convolutional layers are made into pooling layers and then theses are again made into a fully connected layer. And finally, the report is diagnosed and the classification is done.

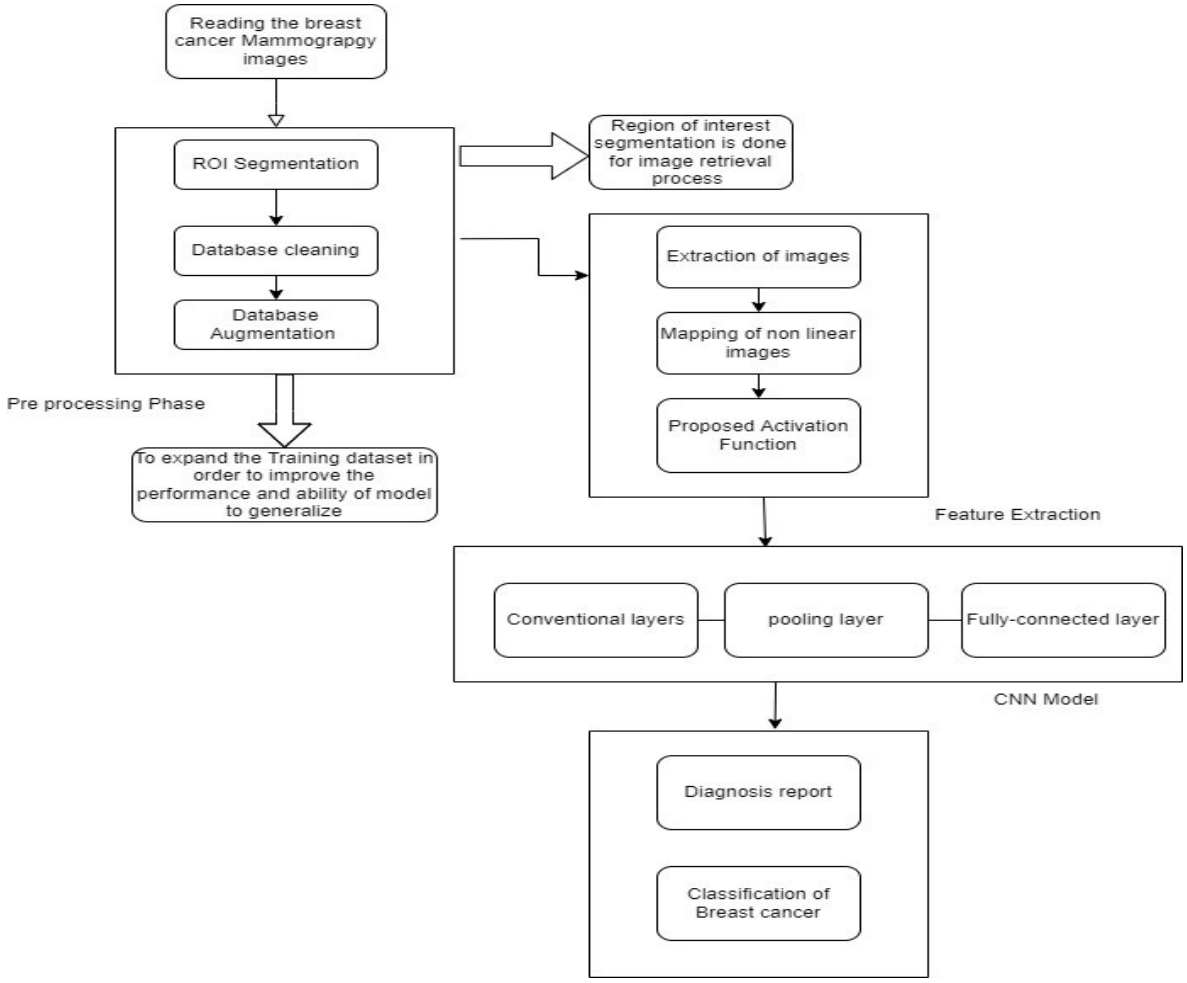


Fig 5.2 2 System Architecture for Breast Cancer Detection

5.2.1 DATA FLOW DIAGRAM

In computers, the path of data from the source document to data entry to processing to final reports. Data changes format and sequence (within a file) as it moves from program to program.

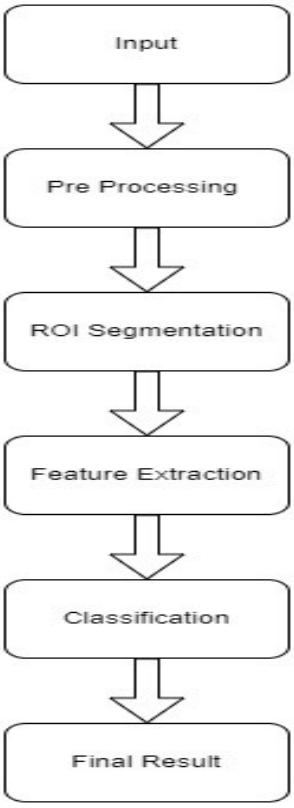


Fig 5.2.1 Data Flow Diagram for Breast Cancer Detection

5.3 ALGORITHM SPECIFICATION

The concept of an algorithm is fundamental to computer science. Algorithms exist for many common problems, and designing efficient algorithms plays a crucial role in developing large–scale computer systems. Therefore, before we proceed further we need to discuss this concept more fully. So, we begin with the Definition of an Algorithm as "it is a finite set of instructions that, if followed, accomplishes a particular task".

Our proposed system will classify the patients suffering from breast cancer disease to those who are healthy based on the training given to the system network with the help of different datasets. The CNN algorithm takes the microscopic images of cancer-affected and unaffected tissues as input and processes the data and then it is provided to the CNN model where convolutional layers are made into pooling layers and then these are again made into a fully connected layer.

Procedure

Step-1: First, we need to import the necessary libraries.

Step-2: In this stage, we need to identify the total number of images that are present for training the data and organize them into a Pandas data frame.

Step-3: After identifying the total number of images and organizing them into a data frame, we will look into the data and find out the percentage of IDC patches on the tissue images, as the IDC patches are the ones that determine whether the tissues are affected by cancer virus or not and come to the notion that the classes of IDC and non-IDC are imbalanced.

Step-4: check cancer and non-cancerous tissue images by displaying the sample tissue. We are provided with the notion that cancer-affected tissues are more violet in color than healthy tissues.

Step-5: Prepare a dataset so that the entire images in different datasets are transferred to a single data set separated by the directories '0' and '1', where 0 represents the directory for benign tissue samples and 1 represents the directory for malignant tissue samples.

Step-6: In this step, balance the class distribution so that the neural network does not lean on favoring only one class.

Step-7: Create new directories for training datasets and validation datasets.

Step-8: Now, we will set up the image generators and build the model.

Step-9: Train the model, with the help of the training dataset and evaluate for loss and accuracy metrics and plot the training curves based on the loss and accuracy.

Step-10: Now, after the model is trained validation of the data is performed with the help of the validation directory and make predictions to calculate the AUC score, print the confusion matrix, and the F1 score.

Step-11: Finally, calculate the AUC score and create a confusion matrix and provide the classification report result.

CHAPTER – 6

IMPLEMENTATION

6. IMPLEMENTATION

6.1 TECHNOLOGY DESCRIPTION

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage is achieving a new successful system and giving confidence in the new system to the users that it will work efficiently and effectively.

The system can be implemented only after thorough testing is done and if it is found to work according to the specification. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change, over and an evaluation of change over methods apart from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system.

The more complex the system being implemented, the more involved will be the systems analysis and design effort required just for implementation. The implementation phase comprises several activities. The required hardware and software acquisitions are carried out. The System may require some hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

A project is a series of activities that aims at solving particular problems within a given time frame and in a particular location. The activity includes time, money, and human and material resources. Before achieving the objectivities, a project goes through several stages. Implementation should take place at and be integrated into all stages of the project cycle. Implementation is the process of having systems personnel check out and put new equipment in to use, train users, install the new application, and construct any files of data needed for it.

Depending on the size of the organization that will be involved in using the application and the risk associated with its use, system developers may choose to test the operation in only one area of the firm, say in one department or with only one or two persons. Sometimes they will run the old and new systems together to compare the results. In still other situations, developers will stop using the old system one-day and begin using the new one the next.

6.2 ABOUT THE PYTHON PROGRAMMING

Python programming language and environment are designed to solve several problems in modern programming practice. Python started as a part of a larger project to develop advanced software for consumer electronics. These devices are small, reliable, portable, distributed, real-time embedded systems.

Fig 6.2 About the Python Programming

Python is a high–level, general-purpose, and very popular programming language. Python programming language (latest Python 3) is being used in Web Development, Machine Learning Applications, along with all cutting–edge technology in Software Industry. Python Programming Language is very well suited for Beginners, also for experienced programmers with other programming languages like C, C++, and JAVA.

6.2.1 THE PYTHON JUPYTER

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Its uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The “notebook” term can colloquially refer to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context.

According to the official website of Jupyter, Project Jupyter exists to develop open-source software, open standards, and services for interactive computing across dozens of programming languages.

Jupyter Book is an open-source project for building books and documents from computational material. It allows the user to construct the content in a mixture of Markdown, an extended version of Markdown called MyST, Maths & Equations using MathJax, Jupyter Notebooks, and reStructuredText, the output of running Jupyter Notebooks at build time. Multiple output formats can be produced (currently single files, multipage HTML web pages, and PDF files).

6.2.2 JUPYTER INSTALLATION

One of the requirements here is Python, either Python 3.3 or greater or Python 2.7. You want to simply install the Jupyter Notebook the Pythonic way, make sure you have downloaded the latest Python Version (which is 3.9 at the time of writing of this article), and then proceed to issue the following command in the command prompt or PowerShell.

```
pip install jupyter
```

Fig 6.2.2 jupyter install through PIP

6.2.3 JUPYTER NOTEBOOK ENVIRONMENT DETAILS

After launching the Jupyter Notebook on your localhost web browser, you should be directed to a page that looks as follows:

After launching the jupyter notebook on your localhost web browser, you should be directed to a page look as below

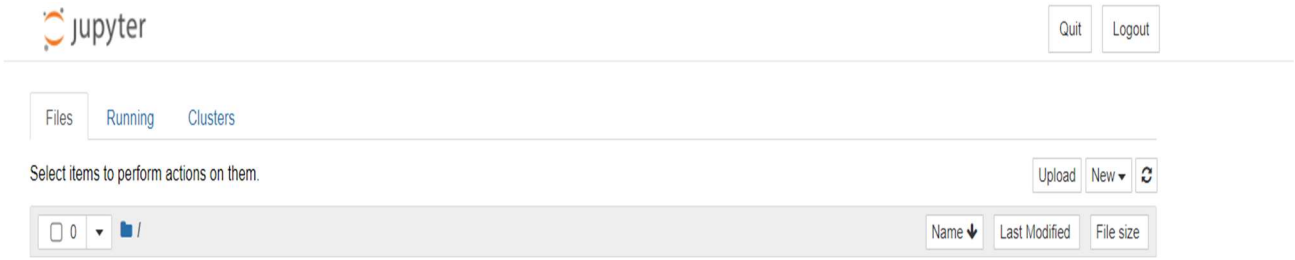


Fig 6.2.3 Jupyter notebook environment

6.2.4 TEST:

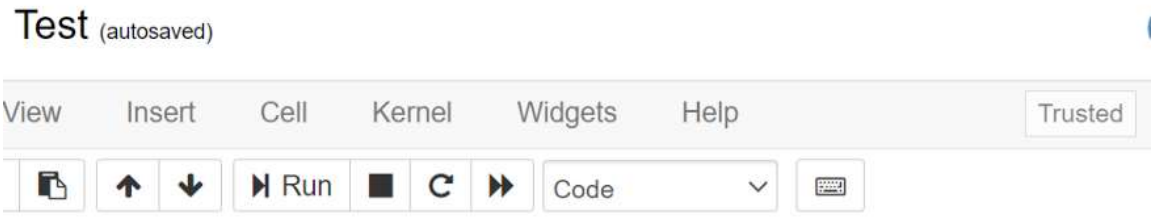


Fig 6.2.4 Jupyter Testing interface

Upon cross-checking the localhost URL that was launched, you will now notice that there is a new Jupyter Notebook file that is created in the specified directory.

Fig 6.2.4 Jupyter Testing interface

6.2.5 RUNNING CODE CELL:

Fig 6.2.5 Jupyter Run Interface

You can run the notebook document step-by-step (one cell at a time) by pressing *ctrlS + enter* for running the particular cell or *shift + enter* to run the current cell as well as create a new cell below it. You can run the whole notebook in a single step by clicking on the menu *Cell -> Run All*.

6.2.6 JUPYTER PROS:

- Best Platform for getting started with data science.
- Easy to share notebooks and visualizations.
- Availability of markdowns and other additional functionalities.

6.2.7 JUPYTER CONS:

- Lack of powerful features which are included in some IDEs.

6.3 DATA SET:

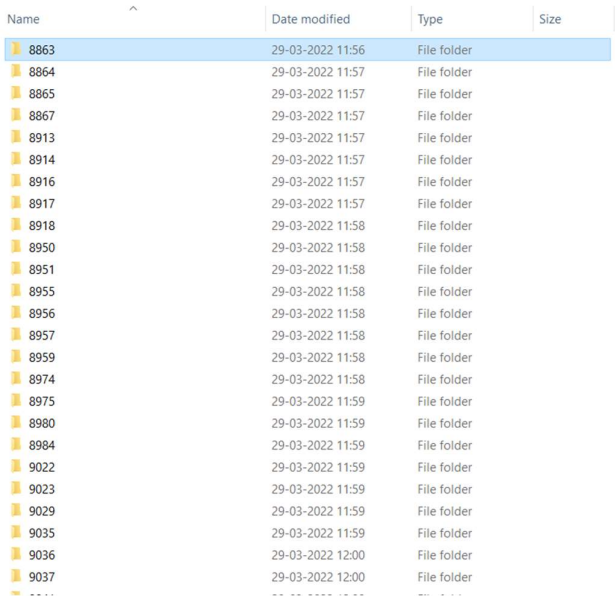


Fig 6.3 Data Set used to be used

- These are the previous dataset records used to train the network

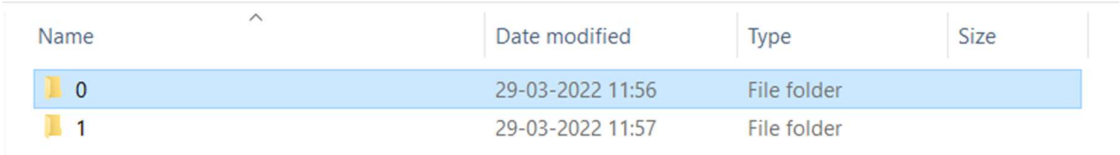
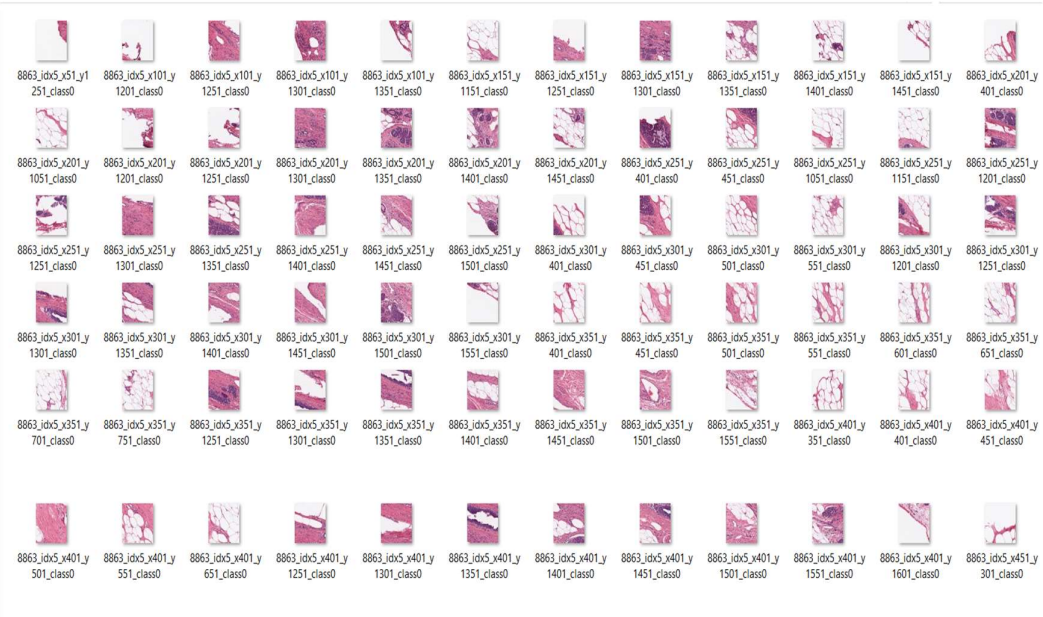


Fig 6.3.1 Inner Directories of the data sets

- This is the image of the benign and malignant datasets that are part of the dataset records.



6.3.2 Benign Dataset

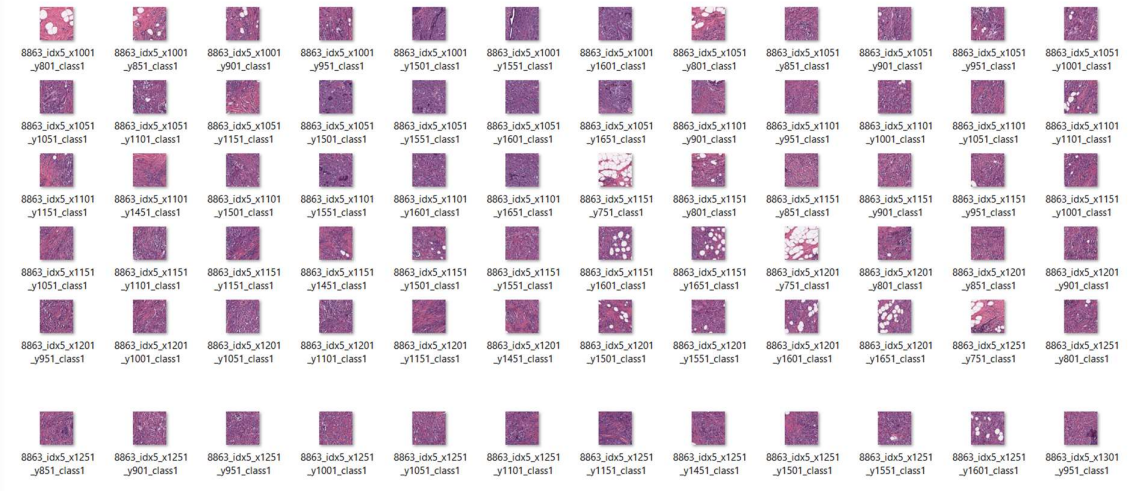


Fig 6.3.3 Malignant Dataset

6.4 SYSTEM MODULES

1. **Sklearn Module:**

   Scikit – learn is probably the most useful library for machine learning in **Python**. The **sklearn library** contains a lot of efficient tools for

machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction.

2. **Matplotlib Module:**

   matplotlib. **pyplot** is a collection of functions that make **matplotlib** work like **MATLAB**. Each **pyplot** function makes some change to a figure: E.g. creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.,

3. **Pandas Module:**

   Pandas are mainly **used** for data analysis. **Pandas** allow importing data from various file formats such as comma-separated-values, JSON, SQL, and Microsoft Excel. **Pandas** allow various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

4. **Numpy Module:**

   NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python.

5. **Tensorflow:**

   TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on the training and inference of deep neural networks.

6. **Cv2:**

   OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage and then Itseez. The library is cross-

platform and free for use under the open-source Apache 2 License. OpenCV has a function to read video, which is cv2.

7. **Skimage:**

scikit-image is an open-source image processing library for the Python programming language. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more.

8. **Os**:

The OS module in Python **provides functions for interacting with the operating system**. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.

9. **Imageio:**

Imageio is a Python library that provides an easy interface to read and write a wide range of image data, including animated images, volumetric data, and scientific formats.

10. **Itertools:**

Itertool is **a module that provides various functions that work on iterators to produce complex iterators**. This module works as a fast, memory-efficient tool that is used either by itself or in combination to form iterator algebra.

11. **Shutil:**

This method is used **to replicate the complete directory**. It copies an entire directory tree rooted at the source to the destination directory. The destination directory must not already be present.

12. **Seaborn:**

Seaborn is an open-source Python library built on top of matplotlib. It is **used for data visualization and exploratory data analysis**. Seaborn

works easily with data frames and the Pandas library. The graphs created can also be customized easily.

```
#importing necissary libraries

from numpy.random import seed
seed(101)

import pandas as pd
import numpy as np

import tensorflow

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

import os
from os import listdir
import cv2

import imageio
import skimage
import skimage.io
import skimage.transform

from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from skimage.io import imread
import itertools
import shutil
import matplotlib.pyplot as plt
import seaborn as sns
```

Fig 6.4 Libraries used in code

6.5 SOURCE CODE

```
#importing necessary libraries
from numpy.random import seed
seed(101)
import pandas as pd
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau,
ModelCheckpoint
import os
from os import listdir
import cv2
import imageio
import skimage
import skimage.io
import skimage.transform
from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from skimage.io import imread
import itertools
import shutil
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Removing duplicate folders to save space before running the notebook
```

```
shutil.rmtree('/kaggle/working/all_images_dir', ignore_errors=True)
shutil.rmtree('/kaggle/working/base_dir', ignore_errors=True)
#Data Structure
files = listdir("../../input")
print(len(files))
#looking at the first 10 folders
files[0:10]
base_path = "../../input"
folder = listdir(base_path)
print("No. of Patients:",len(folder))
#numberoftotalimages
total_images = 0
for n in range(len(folder)):
    patient_id = folder[n]
    for c in [0, 1]:
        patient_path = base_path + '/'+patient_id
        class_path = patient_path + '/' + str(c) + '/'
        subfiles = listdir(class_path)
        total_images += len(subfiles)
print("Total Images in dataset: ", total_images )
#dataintopandasdataframe
data = pd.DataFrame(index=np.arange(0, total_images), columns=["patient_id", "path",
"target"])
k = 0
for n in range(len(folder)):
    patient_id = folder[n]
    patient_path = base_path + '/'+patient_id
    for c in [0,1]:
        class_path = patient_path + "/" + str(c) + "/"
        subfiles = listdir(class_path)
        for m in range(len(subfiles)):
            image_path = subfiles[m]
            data.iloc[k]["path"] = class_path + image_path
            data.iloc[k]["target"] = c
```

```
        data.iloc[k]["patient_id"] = patient_id
        k += 1
data.head()
#shapeofdataframe
data.shape
#data
cancer_perc = data.groupby("patient_id").target.value_counts() /
data.groupby("patient_id").target.size()
canxer_perc = cancer_perc.unstack()
fig, ax = plt.subplots(1, 3,figsize = (20,5))
sns.distplot(data.groupby('patient_id').size(), ax=ax[0], color='Orange', kde=False, bins=30)
ax[0].set_xlabel('Number of patches')
ax[0].set_ylabel('Frequency')
ax[0].set_title('how may patches do we have per patient?')
sns.distplot(cancer_perc.loc[:, 1]*100, ax=ax[1], color="Tomato", kde=False, bins=30)
ax[1].set_title("How much percentage of an image is covered by IDC?")
ax[1].set_ylabel("Frequency")
ax[1].set_xlabel("% of patches with IDC");
sns.countplot(data.target, palette="Set2", ax=ax[2]);
ax[2].set_xlabel("no(0) versus yes(1)")
ax[2].set_title("How many patches show IDC?");
# coverting target to int
data.target = data.target.astype(np.int)
#displayingcancertissuesamples
cancer_selection = np.random.choice(data[data.target == 1].index.values, size=50,
replace=False)
fig, ax = plt.subplots(5, 10, figsize=(20, 10))
for n in range(5):
   for m in range(10):
      idx = cancer_selection[m + 10*n]
      image = imread(data.loc[idx, "path"])
      ax[n,m].imshow(image)
      ax[n,m].grid(False)
#displaynon-cancertissue
```

```
non_cancer_selection = np.random.choice(data[data.target == 0].index.values, size=50,
replace=False)
fig, ax = plt.subplots(5, 10, figsize=(20, 10))
for n in range(5):
    for m in range(10):
        idx = non_cancer_selection[m + 10*n]
        image = imread(data.loc[idx, "path"])
        ax[n,m].imshow(image)
        ax[n,m].grid(False)
# Creating diresctory to store all images
all_images_dir = 'all_images_dir'
if os.path.isdir(all_images_dir):
    pass
else:
    os.mkdir(all_images_dir)
# This code copies all images from their separate folders into the same
# folder called all_images_dir.
'''
The directory structure is like this:
    patient_id:
                0
                1
'''
patient_list = folder
for patient in patient_list:
    path_0 = "../../input/" + str(patient) + '/0'
    path_1 = "../../input/" + str(patient) + '/1'
    # create list of all files in folder 0
    file_list_0 = listdir(path_0)
    #create a list of all files in folder 1
    file_list_1 = listdir(path_1)
    # moving the 0 class images to all_images_dir
    for fname in file_list_0:
        src = os.path.join(path_0, fname)
```

```
        dst = os.path.join(all_images_dir, fname)
        shutil.copyfile(src, dst)
    # moving the 1 class images to all_images_dir
    for fname in file_list_1:
        src = os.path.join(path_1, fname)
        dst = os.path.join(all_images_dir, fname)
        shutil.copyfile(src, dst)
# Total number of images
len(listdir(all_images_dir))
#createdataframeofallimages
image_list = os.listdir('all_images_dir')
df_data = pd.DataFrame(image_list, columns=['image_id'])
df_data.head()
# Defining helper functions
def extract_patient_id(x):
    a = x.split('_')
    patient_id = a[0]
    return patient_id
def extract_target(x):
    a = x.split('_')
    b = a[4]
    target = b[5]
    return target
# creating new column named patient_id
df_data['patient_id'] = df_data['image_id'].apply(extract_patient_id)
#creating new column named target
df_data['target'] = df_data['image_id'].apply(extract_target)
df_data.head(10)
# class distribution of the images
df_data['target'].value_counts()
#balancetheclassdistribution
SAMPLE_SIZE = 78786
# take a sample of the majority class 0 (total = 198738)
df_0 = df_data[df_data['target'] == '0'].sample(SAMPLE_SIZE, random_state=101)
```

```
# take a sample of class 1 (total = 78786)
df_1 = df_data[df_data['target'] == '1'].sample(SAMPLE_SIZE, random_state=101)
# concat the two dataframes
df_data = pd.concat([df_0, df_1], axis=0).reset_index(drop=True)
# Check the new class distribution
df_data['target'].value_counts()
#creatingtrainandtestsets
y = df_data['target']
df_train, df_val = train_test_split(df_data, test_size=0.10, random_state=101, stratify=y)
print(df_train.shape)
print(df_val.shape)
# Creating new base directory
base_dir ='base_dir2'
os.mkdir(base_dir)
# Creating train directory inside base directory
train_dir = os.path.join(base_dir, 'train_dir')
os.mkdir(train_dir)
# Creating validation directory inside base directory
val_dir = os.path.join(base_dir, 'val_dir')
os.mkdir(val_dir)
# create new folders inside train_dir
a_no_idc = os.path.join(train_dir, 'a_no_idc')
os.mkdir(a_no_idc)
b_has_idc = os.path.join(train_dir, 'b_has_idc')
os.mkdir(b_has_idc)
# create new folders inside val_dir
a_no_idc = os.path.join(val_dir, 'a_no_idc')
os.mkdir(a_no_idc)
b_has_idc = os.path.join(val_dir, 'b_has_idc')
os.mkdir(b_has_idc)
# check that the folders have been created
os.listdir('base_dir/train_dir')
# Set the id as the index in df_data
df_data.set_index('image_id', inplace=True)
```

```
train_list = list(df_train['image_id'])
val_list = list(df_val['image_id'])
# Transfering the train images
for image in train_list:
    try:
        fname = image
        target = df_data.loc[image, 'target']
        if target == '0':
            label = 'a_no_idc'
        if target == '1':
            label = 'b_has_idc'
        # source path to image
        src = os.path.join(all_images_dir, fname)
        # destination path to image
        dst = os.path.join(train_dir, label, fname)
        # move the image from the source to the destination
        shutil.move(src, dst)
    except:
        continue
for image in val_list:
    try:
        fname = image
        target = df_data.loc[image,'target']
        if target == '0':
            label = 'a_no_idc'
        if target == '1':
            label = 'b_has_idc'
        # source path to image
        src = os.path.join(all_images_dir, fname)
        # destination path to image
        dst = os.path.join(val_dir, label, fname)
        # move the image from the source to the destination
        shutil.move(src, dst)
```

```
    except:
        continue
# check how many val images we have in each folder
print(len(os.listdir('base_dir/train_dir/a_no_idc')))
print(len(os.listdir('base_dir/train_dir/b_has_idc')))
#settingimagegenerators
train_path = 'base_dir/train_dir'
valid_path = 'base_dir/val_dir'
num_train_samples = len(df_train)
num_val_samples = len(df_val)
train_batch_size = 10
val_batch_size = 10
train_steps = np.ceil(num_train_samples / train_batch_size)
val_steps = np.ceil(num_val_samples / val_batch_size)
IMAGE_SIZE = 50
datagen = ImageDataGenerator(rescale = 1.0 / 255,
                             rotation_range = 90,
                             zoom_range = 0.2,
                             horizontal_flip=True,
                             vertical_flip=True)
train_gen = datagen.flow_from_directory(train_path,
                                        target_size=(IMAGE_SIZE,IMAGE_SIZE),
                                        batch_size=train_batch_size,
                                        class_mode='categorical')
val_gen = datagen.flow_from_directory(valid_path,
                                      target_size=(IMAGE_SIZE,IMAGE_SIZE),
                                      batch_size=val_batch_size,
                                      class_mode='categorical')
# Note: shuffle=False causes the test dataset to not be shuffled
test_gen = datagen.flow_from_directory(valid_path,
                                       target_size=(IMAGE_SIZE,IMAGE_SIZE),
                                       batch_size=1,
                                       class_mode='categorical',
                                       shuffle=False)
```

```
# Building the model
kernel_size = (3,3)
pool_size= (2,2)
first_filters = 32
second_filters = 64
third_filters = 128
dropout_conv = 0.3
dropout_dense = 0.3
model = Sequential()
model.add(Conv2D(first_filters, kernel_size, activation = 'relu',
                 input_shape = (IMAGE_SIZE, IMAGE_SIZE, 3)))
model.add(Conv2D(first_filters, kernel_size, activation = 'relu'))
model.add(Conv2D(first_filters, kernel_size, activation = 'relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))
model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))
model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))
model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(dropout_dense))
model.add(Dense(2, activation = "softmax"))
model.summary()
#trainingthemodel
model.compile(Adam(lr=0.0001), loss='binary_crossentropy',
              metrics=['accuracy'])
filepath = "model.h5"
```

```
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
                             save_best_only=True, mode='max')
reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.5, patience=3,
                              verbose=1, mode='max', min_lr=0.00001)
callbacks_list = [checkpoint, reduce_lr]
history = model.fit_generator(train_gen, steps_per_epoch=train_steps,
                    validation_data=val_gen,
                    validation_steps=val_steps,
                    epochs=50, verbose=1,
                   callbacks=callbacks_list)
try:
   model.save('/kaggle/working/model.h5')
except:
   pass
try:
   model.save('model.h5')
except:
   pass
model.metrics_names
# Here the best epoch will be used.
model.load_weights('model.h5')
val_loss, val_acc = \
model.evaluate_generator(test_gen,
                         steps=len(df_val))
print('val_loss:', val_loss)
print('val_acc:', val_acc)
# display the loss and accuracy curves
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, loss, 'bo', label='Training loss')
```

```
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.figure()
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
# make a prediction
predictions = model.predict_generator(test_gen, steps=len(df_val), verbose=1)
predictions.shape
# This is how to check what index Keras has internally assigned to each class.
test_gen.class_indices
# Put the predictions into a data frame.
# The columns need to be ordered to match the output of the previous cell
df_preds = pd.DataFrame(predictions, columns=['no_idc', 'has_idc'])
df_preds.head()
# Get the true labels
y_true = test_gen.classes
# Get the predicted labels as probabilities
y_pred = df_preds['has_idc']
#calculatingAUCscore
from sklearn.metrics import roc_auc_score
roc_auc_score(y_true, y_pred)
#creatingconfusionmatrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion Matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
```

```
        print('Confusion matrix, without normalization')
    print(cm)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
test_labels = test_gen.classes
test_labels.shape
# argmax returns the index of the max value in a row
cm = confusion_matrix(test_labels, predictions.argmax(axis=1))
# Print the label associated with each class
test_gen.class_indices
# Define the labels of the class indices. These need to match the
# order shown above.
cm_plot_labels = ['a_no_idc', 'b_has_idc']
plot_confusion_matrix(cm, cm_plot_labels, title='Confusion Matrix')
#createclassificationreport
from sklearn.metrics import classification_report
# Generate a classification report
# For this to work we need y_pred as binary labels not as probabilities
y_pred_binary = predictions.argmax(axis=1)
report = classification_report(y_true, y_pred_binary, target_names=cm_plot_labels)
print(report)
```

CHAPTER – 7
TESTING

7. TESTING

7.1 INTRODUCTION

Software Testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test has the following:

- ➢ Meets the requirements that guided its design and development,
- ➢ Responds correctly to all kinds of inputs,
- ➢ Performs its functions within an acceptable time,
- ➢ Is sufficiently usable,
- ➢ Can be installed and run in its intended environments, and
- ➢ Achieves the general result of its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). Software Testing can provide objective, independent information about the quality of software and the risk of its failure to users and/or sponsors. Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing

occurs after system requirements have been defined and then implemented in testable 49 programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

7.2 TESTING APPROACH

In Testing Approach, the Testing performs a critical role in quality assurance and in ensuring the reliability of software. To accomplish the task of testing software, this is done under two categories of test case design techniques

- White Box Testing, and
- Black Box Testing.

White Box Testing

White Box Testing is also known as "Structural Testing". This test focuses on the program control structure. Test cases are derived to ensure that all the statements in the program have been executed at least once during testing and that all logical conditions have been exercised. White Box Testing has many advantages: They are easy to automate and provide traceability of tests from the source allowing future changes to the software to be easily captured in changes to the tests. The whole point of the White Box Testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

Black Box Testing

Black Box Testing is also known as "Functional Testing". These tests are designed to validate functional requirements without regard to the working of the program. The Black Box Testing technique focuses on the information domain of the software, deriving test cases by partitioning input and output in a manner that provides thorough test coverage.

Black-box testers do not have access to the source code and are oblivious of the system architecture. In Black Box Testing, the target software is exercised over a range of inputs and the outputs are observed for correctness.

Levels of Testing

The basic levels of testing are Unit Testing, Integrated Testing, System Testing, and Acceptance Testing. These different levels of testing attempt to detect different types of faults, and they can be defined as

Unit Testing

Unit Testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration.

Unit Testing is the first level of testing. In this, the different modules are tested against the specifications of the system during the design for the modules. It is essential for the verification of code developed during the coding phase.

Integration Testing

Integration Tests are designed to test integrated software components to determine if they run as one program. Testing is event-driven and is more concerned with the basic outcome of screens or fields.

Integration testing is specifically aimed at exposing the problems that arise from the combination of components. This testing is a systematic technique for constructing the program structure while at the same time conducting the test to uncover errors associated with the interfacing. There are two approaches to this testing:

- Bottom-Up Integration, and
- Top-Down Integration.

Bottom-Up Integration

Testing can be performed starting from the smallest and lowest level modules and proceeding one at a time. For each module in bottom-up testing, a short program executes the module and provides the needed data so that the module

is asked to perform the way it will when embedded within the larger system. When bottom-level modules are tested attention turns to those on the next level that use the lower–level ones they are tested individually and then linked with the previously examined lower-level modules.

Top-Down Integration

This type of testing starts from upper–level modules. Since the detailed activities usually performed in the lower–level routines are not provided stubs are written. A stub is a module shell called by upper – a level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower-level module.

Functional Testing

Functional Tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

➢ Valid Input: identified classes of valid input must be accepted,

➢ Invalid Input: identified classes of invalid input must be rejected,

➢ Function: identified functions must be exercised,

➢ Output: identified classes of application outputs must be exercised, and

➢ Systems / Procedures: interfacing systems or procedures must be invoked.

The organization and preparation of functional tests are focused on requirements, key functions, or special test cases. In addition, systematic coverage about identifying Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System Testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points. System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results.

Here, the entire system is tested. The reference document for this purpose is the requirements document and the goal is to see if the software meets its requirements. They are:

1. Recovery Testing

This testing technique forces the software to fail. If the software recovers back automatically (or) it recovers back with minimum human intention, then the software is said to be within acceptable limits.

2. Security Testing

This testing technique verifies the protection mechanisms of the software and decides whether the software developed can protect from proper access by hackers.

3. Stress Testing

This testing technique is to determine how sturdy and reliable the software is in extreme conditions.

4. Acceptance Testing

This testing technique is sometimes performed on the realistic data of the client to demonstrate that the software is working satisfactorily. Testing here focuses on the external behavior of the system.

5. Validation Testing

In this testing technique, the system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input,

corresponding error messages are displayed. For a System, the Validation Testing can be validated under:

Configuration Review

This is to inspect whether the software is satisfying all the requirements of all the users and also to inspect the documentation.

Alpha Testing

This testing can be validated and done under the controlled environment of the client on the developer's side.

Beta Testing

The clients in the absence of the developer do it at all parallel sides. Once the beta testing is complete the developer announces an official release of the software.

Test Planning

Test Planning can be defined as a strategic document that describes the procedure of how to perform various testing on the total application in the most efficient way like:

➢ The document involves the Scope of testing,

➢ The objective of testing,

➢ Areas that need to be tested,

➢ Areas that should not be tested, and

➢ Scheduling resource planning.

Test Development

➢ Test case development (checklist), and

➢ Test procedure preparation (description of test cases).

Test Execution

➢ Implementation of Test Cases, and

➢ Observing the Result.

7.3 TEST CASES

In general, a Test Case is a set of test data and test programs and their expected results. A Test Case in Software Engineering normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, and output and it validates one or more system requirements and generates a pass or fail.

The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. It may take many test cases to determine that a software program or system is considered sufficiently scrutinized to be released.

Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites. If a requirement has sub-requirements, each sub-requirement must have at least two test cases. Written test cases should include a description of the functionality to be tested, and the preparation required to ensure that the test can be conducted.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies, and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail unacceptably. There are various types of tests. Each test type addresses a specific testing requirement.

7.3.1 TEST CASE FORMAT

Generally, the Test Cases usually have the following components:

➢ Test Case Summary,

➢ Initial Condition,

➢ Steps to run the test case, and,

➢ Expected behavior/outcome.

1. Positive Test Cases:

➢ The positive flow of the functionality must be considered,

➢ Valid inputs must be used for testing, and

➢ Must have a positive perception to verify whether the requirements are justified.

S.no. of Test Case	**TC1**
Test Design/Name of Test	**"Whether the system can identify malignant tissue"**
Test Description & Sample Input / Test Data	Provide dataset for process with the valid dataset of malignant tissue.
Expected Output	Display Message as "Malignant tissue is found"
Actual Output	Same as expected.
Test Status / Remarks	Passed.

S.no. of Test Case	**TC2**
Test Design/Name of Test	**"Whether the system can identify malignant tissue"**
Test Description & Sample Input / Test Data	Provide dataset for process with the invalid dataset
Expected Output	Display Message as "Dataset Invalid"
Actual Output	Same as expected.
Test Status / Remarks	Passed.

Table 7.3 Test cases for the system user to perform Unit Testing and Validation Testing

S.no. of Test Case	TC3
Test Design/Name of Test	**"Whether the system can identify benign tissue"**
Test Description & Sample Input / Test Data	Checks for the selected item.
Expected Output	Display message as "benign is found
Actual Output	Same as expected
Test Status / Remarks	Passed

S.no. of Test Case	TC4
Test Design/Name of Test	**"Whether the system can display benign tissue"**
Test Description & Sample Input / Test Data	Checks for the selected item.
Expected Output	Display message as "benign tissues are displayed"
Actual Output	Same as expected
Test Status / Remarks	Passed

S.no. of Test Case	TC5
Test Design/Name of Test	**"Whether the system can display malignant tissue or not"**
Test Description & Sample Input / Test Data	Checks for the selected item.
Expected Output	Display message as "malignant tissues displayed"
Actual Output	Same as expected
Test Status / Remarks	Passed

S.no. of Test Case	TC3
Test Design/Name of Test	**"Whether the system can identify both malignant and benign tissue.**
Test Description & Sample Input / Test Data	Checks for the selected item.
Expected Output	Classifies whether the tissues are either malignant or benign.
Actual Output	Same as expected
Test Status / Remarks	Passed

Table 7.3 Test cases for the model to perform Unit Testing and Validation Testing

7.3.2 TEST RESULTS:

S.No.	USER INPUT Dataset(File) (Cancerless Tissues)	ANALYZATION
1.		The system identifies the tissue is not affected by cancer

Table 7.3 Analyzing the dataset

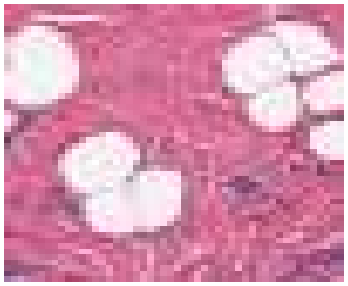
S.No.	USER INPUT Dataset (File) (Cancer-less Tissues)	ANALYZATION
1.		The system identifies the tissue is affected by cancer

Table 7.3 Analyzing the dataset

7.3.3 GUIDELINES FOR DEVELOPING TEST CASES

1. Describe which feature or service your test attempts to cover.

2. If the test case is based on a use case it is a good idea to refer to the use case name.

3. Remember that the use cases are the source of test cases. In theory, the software is supposed to match the use cases, not the reverse. As soon as you have enough use cases, go ahead and write the test plan for that piece.

4. Specify what you are testing and which particular feature. Then specify what you are going to do to test the feature and what you expect to happen.

5. Test the normal use of the object's methods. Test the abnormal but reasonable use of the object's methods.

6. Test the abnormal but unreasonable use of the object's methods.

7. Test the boundary conditions. Also, specify when you expect error dialog boxes when you expect some default event, and when functionality till is being defined.

8. Test object's interactions and the messages sent among them. If you have developed sequence diagrams, they can assist you in this process.

9. When the revisions have been made, document the cases so they become the starting bases for the follow–up test, and finally, attempt to reach an agreement on answers generally will raise others– if. Add thereto the list and answer them, repeat the process until the lit is stabilized, then you need not add any more questions.

CHAPTER – 8

RESULTS

8. RESULTS

1. Explore The Data

a) No. of Patients and Total number of images in the dataset

```
base_path = "../../input"
folder = listdir(base_path)
print("No. of Patients:",len(folder))
```

```
No. of Patients: 279
```

```
total_images = 0
for n in range(len(folder)):
    patient_id = folder[n]
    for c in [0, 1]:
        patient_path = base_path + '/'+patient_id
        class_path = patient_path + '/' + str(c) + '/'
        subfiles = listdir(class_path)
        total_images += len(subfiles)

print("Total Images in dataset: ", total_images )
```

```
Total Images in dataset:  277524
```

Fig 8.1 Screenshot of the total number of patients and images

b) Organizing data into a pandas data frame

```
data = pd.DataFrame(index=np.arange(0, total_images), columns=["patient_id", "path", "target"])

k = 0
for n in range(len(folder)):
    patient_id = folder[n]
    patient_path = base_path + '/'+patient_id
    for c in [0,1]:
        class_path = patient_path + "/" + str(c) + "/"
        subfiles = listdir(class_path)
        for m in range(len(subfiles)):
            image_path = subfiles[m]
            data.iloc[k]["path"] = class_path + image_path
            data.iloc[k]["target"] = c
            data.iloc[k]["patient_id"] = patient_id
            k += 1

data.head()
```

	patient_id	path	target
0	10253	../../input/10253/0/10253_idx5_x1001_y1001_cla...	0
1	10253	../../input/10253/0/10253_idx5_x1001_y1051_cla...	0
2	10253	../../input/10253/0/10253_idx5_x1001_y1101_cla...	0
3	10253	../../input/10253/0/10253_idx5_x1001_y1151_cla...	0
4	10253	../../input/10253/0/10253_idx5_x1001_y1201_cla...	0

Fig 8.2 Screenshot for organizing the dataset into a pandas data frame

c) Exploring the data

```
cancer_perc = data.groupby("patient_id").target.value_counts() / data.groupby("patient_id").target.size()
canxer_perc = cancer_perc.unstack()

fig, ax = plt.subplots(1, 3,figsize = (20,5))
sns.distplot(data.groupby('patient_id').size(), ax=ax[0], color='Orange', kde=False, bins=30)
ax[0].set_xlabel('Number of patches')
ax[0].set_ylabel('Frequency')
ax[0].set_title('how may patches do we have per patient?')
sns.distplot(cancer_perc.loc[:, 1]*100, ax=ax[1], color="Tomato", kde=False, bins=30)
ax[1].set_title("How much percentage of an image is covered by IDC?")
ax[1].set_ylabel("Frequency")
ax[1].set_xlabel("% of patches with IDC");
sns.countplot(data.target, palette="Set2", ax=ax[2]);
ax[2].set_xlabel("no(0) versus yes(1)")
ax[2].set_title("How many patches show IDC?");
```

Fig 8.3 Screenshot for determining the patches on a single tissue cell of a patient

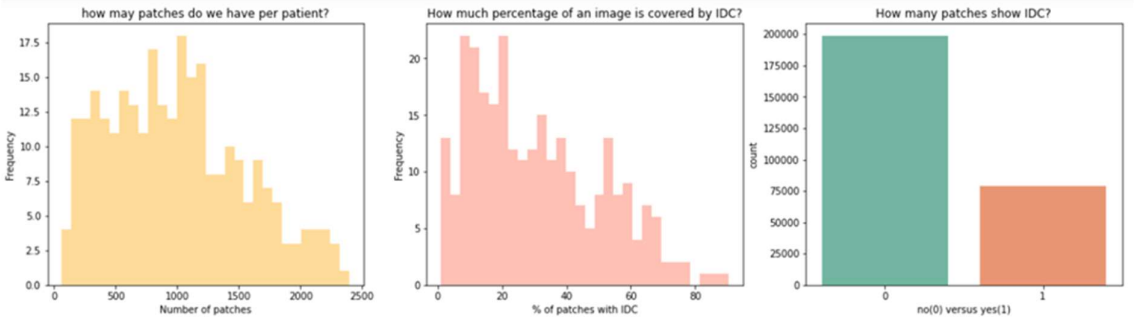


Fig 8.4 Screenshot for Graphical representation of the percentage of patches on the tissues

d) Displaying Cancer Tissue Samples

```
cancer_selection = np.random.choice(data[data.target == 1].index.values, size=50, replace=False)

fig, ax = plt.subplots(5, 10, figsize=(20, 10))

for n in range(5):
    for m in range(10):
        idx = cancer_selection[m + 10*n]
        image = imread(data.loc[idx, "path"])
        ax[n,m].imshow(image)
        ax[n,m].grid(False)
```

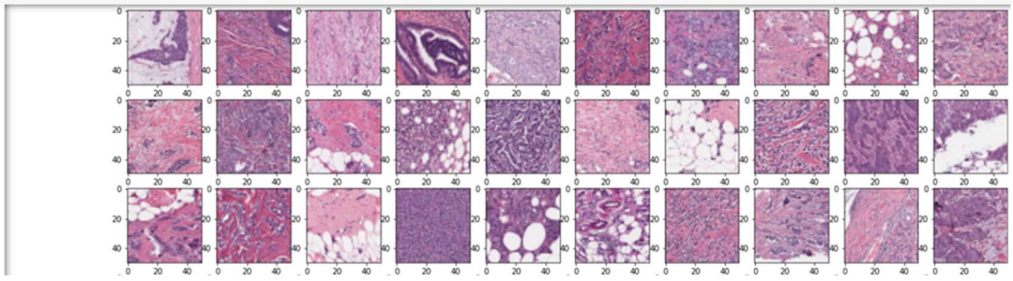


Fig 8.5 Screenshot to Display the cancer-affected tissue cells

e) Displaying Non-Cancer Tissue Samples

```
non_cancer_selection = np.random.choice(data[data.target == 0].index.values, size=50, replace=False)

fig, ax = plt.subplots(5, 10, figsize=(20, 10))

for n in range(5):
    for m in range(10):
        idx = non_cancer_selection[m + 10*n]
        image = imread(data.loc[idx, "path"])
        ax[n,m].imshow(image)
        ax[n,m].grid(False)
```

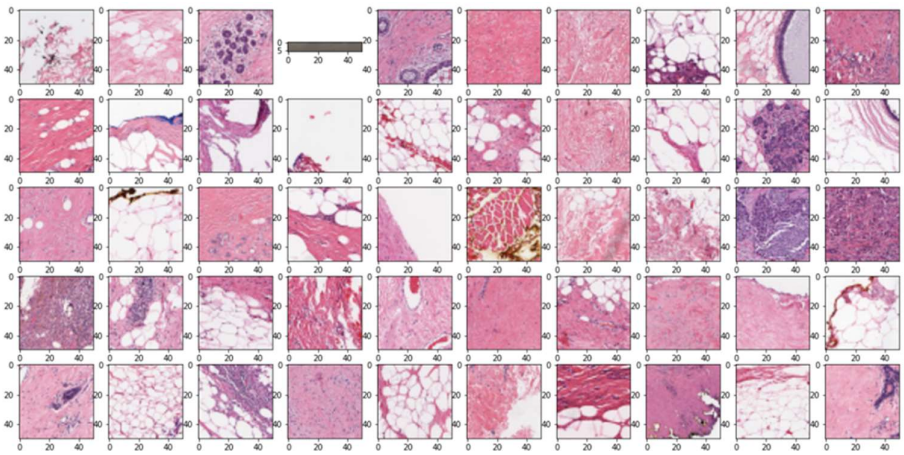


Fig 8.6 Screenshot to Display the cancer unaffected tissue cells

f) Preparing the Data Set

```
# Creating diresctory to store all images
all_images_dir = 'all_images_dir'

if os.path.isdir(all_images_dir):
    pass
else:
    os.mkdir(all_images_dir)
```

```
# This code copies all images from their seperate folders into the same
# folder called all_images_dir.

'''
The directory structure is like:
    patient_id:
                0
                1
'''

patient_list = folder

for patient in patient_list:

    path_0 = "../../input/" + str(patient) + '/0'
    path_1 = "../../input/" + str(patient) + '/1'

    # create list of all files in folder 0
    file_list_0 = listdir(path_0)

    #create a list of all files in folder 1
    file_list_1 = listdir(path_1)

    # moving the 0 class images to all_images_dir
    for fname in file_list_0:

        src = os.path.join(path_0, fname)
        dst = os.path.join(all_images_dir, fname)
        shutil.copyfile(src, dst)

    # moving the 1 class images to all_images_dir
    for fname in file_list_1:

        src = os.path.join(path_1, fname)
        dst = os.path.join(all_images_dir, fname)
```

```
        shutil.copyfile(src, dst)
```

```
# Total number of images
len(listdir(all_images_dir))
```

```
Out[21]: 277524
```

Fig 8.7 Screenshot for storing the datasets into new directories and displaying total images

g) Class distribution of images

```
# class distribution of the images

df_data['target'].value_counts()
```

```
0    198738
1     78786
Name: target, dtype: int64
```

Fig 8.8 Screenshot check the class distribution of images

h) Balance the class distribution

```
SAMPLE_SIZE = 78786

# take a sample of the majority class 0 (total = 198738)
df_0 = df_data[df_data['target'] == '0'].sample(SAMPLE_SIZE, random_state=101)
# take a sample of class 1 (total = 78786)
df_1 = df_data[df_data['target'] == '1'].sample(SAMPLE_SIZE, random_state=101)

# concat the two dataframes
df_data = pd.concat([df_0, df_1], axis=0).reset_index(drop=True)

# Check the new class distribution
df_data['target'].value_counts()
```

```
0    78786
1    78786
Name: target, dtype: int64
```

Fig 8.9 Screenshot to balance the distribution

i) Training the model

```
model.compile(Adam(lr=0.0001), loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
filepath = "model.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
                             save_best_only=True, mode='max')

reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.5, patience=3,
                                  verbose=1, mode='max', min_lr=0.00001)


callbacks_list = [checkpoint, reduce_lr]

history = model.fit_generator(train_gen, steps_per_epoch=train_steps,
                    validation_data=val_gen,
                    validation_steps=val_steps,
                    epochs=50, verbose=1,
                   callbacks=callbacks_list)

try:
    model.save('/kaggle/working/model.h5')
except:
    pass

try:
    model.save('model.h5')
except:
    pass
```

```
Epoch 1/50
14182/14182 [==============================] - ETA: 0s - loss: 0.4669 - accuracy: 0.7898WARNING:tensorflow:Can save best mode
l only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
14182/14182 [==============================] - 1347s 95ms/step - loss: 0.4669 - accuracy: 0.7898 - val_loss: 0.3989 - val_acc
uracy: 0.8313 - lr: 1.0000e-04
Epoch 2/50
14182/14182 [==============================] - ETA: 0s - loss: 0.4094 - accuracy: 0.8216WARNING:tensorflow:Can save best mode
l only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
14182/14182 [==============================] - 1325s 93ms/step - loss: 0.4094 - accuracy: 0.8216 - val_loss: 0.3718 - val_acc
uracy: 0.8417 - lr: 1.0000e-04
Epoch 3/50
14182/14182 [==============================] - ETA: 0s - loss: 0.3890 - accuracy: 0.8330WARNING:tensorflow:Can save best mode
l only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
```

```
Epoch 4/50
14182/14182 [==============================] - ETA: 0s - loss: 0.3779 - accuracy: 0.8388WARNING:tensorflow:Can save best mode
l only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
14182/14182 [==============================] - 1338s 94ms/step - loss: 0.3779 - accuracy: 0.8388 - val_loss: 0.3618 - val_acc
uracy: 0.8480 - lr: 1.0000e-04
Epoch 5/50
14182/14182 [==============================] - ETA: 0s - loss: 0.3697 - accuracy: 0.8431WARNING:tensorflow:Can save best mode
l only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
14182/14182 [==============================] - 1338s 94ms/step - loss: 0.3697 - accuracy: 0.8431 - val_loss: 0.3519 - val_acc
uracy: 0.8520 - lr: 1.0000e-04
Epoch 6/50
14182/14182 [==============================] - ETA: 0s - loss: 0.3627 - accuracy: 0.8471WARNING:tensorflow:Can save best mode
l only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
```

Fig 8.10 Screenshot to Train the Data

j) Evaluating the model for loss and accuracy

```
# get the metric names so we can use evaulate_generator
model.metrics_names
```

```
['loss', 'accuracy']
```

```
# Here the best epoch will be used.

model.load_weights('model.h5')

val_loss, val_acc = \
model.evaluate_generator(test_gen,
                         steps=len(df_val))

print('val_loss:', val_loss)
print('val_acc:', val_acc)
```

```
val_loss: 0.28590962290763855
val_acc: 0.880378246307373
```

Fig 8.11 Screenshot for loss and accuracy with the validation dataset

k) Plotting the training curves

```
# display the loss and accuracy curves

import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.figure()

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
```

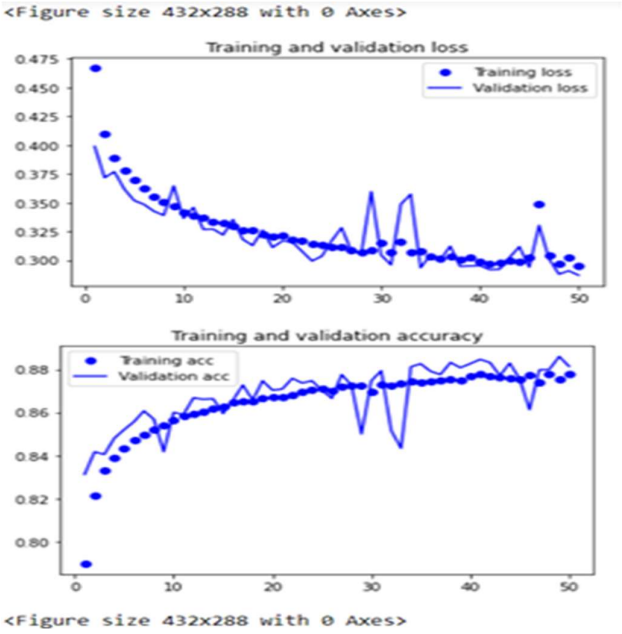


Fig 8.12 Screenshot to plot the training curves

l) predicting on the val set

```
# make a prediction
predictions = model.predict_generator(test_gen, steps=len(df_val), verbose=1)
```

```
15758/15758 [==============================] - 86s 5ms/step
```

Fig 8.13 Screenshot to predict using the validation set

m) Put the predictions into a data frame

```
# Put the predictions into a dataframe.
# The columns need to be oredered to match the output of the previous cell

df_preds = pd.DataFrame(predictions, columns=['no_idc', 'has_idc'])

df_preds.head()
```

	no_idc	has_idc
0	0.560014	0.439986
1	0.061548	0.938452
2	0.998882	0.001118
3	0.998087	0.001913
4	0.653806	0.346194

Fig 8.14 Screenshot to put the prediction into a data frame

n) Calculating the AUC score

```
In [52]: from sklearn.metrics import roc_auc_score

         roc_auc_score(y_true, y_pred)

Out[52]: 0.9511169147533367
```

Fig 8.15 Screenshot for calculating the AUC score

o) Creating the confusion matrix

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion Matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```

```
# Get the labels of the test images.

test_labels = test_gen.classes
```

```
In [56]: test_labels.shape

Out[56]: (15758,)

In [57]: # argmax returns the index of the max value in a row
         cm = confusion_matrix(test_labels, predictions.argmax(axis=1))

In [58]: # Print the label associated with each class
         test_gen.class_indices

Out[58]: {'a_no_idc': 0, 'b_has_idc': 1}

In [59]: # Define the labels of the class indices. These need to match the
         # order shown above.
         cm_plot_labels = ['a_no_idc', 'b_has_idc']

         plot_confusion_matrix(cm, cm_plot_labels, title='Confusion Matrix')

         Confusion matrix, without normalization
         [[6665 1214]
          [ 667 7212]]
```

Fig 8.16 Screenshot to build a confusion matrix

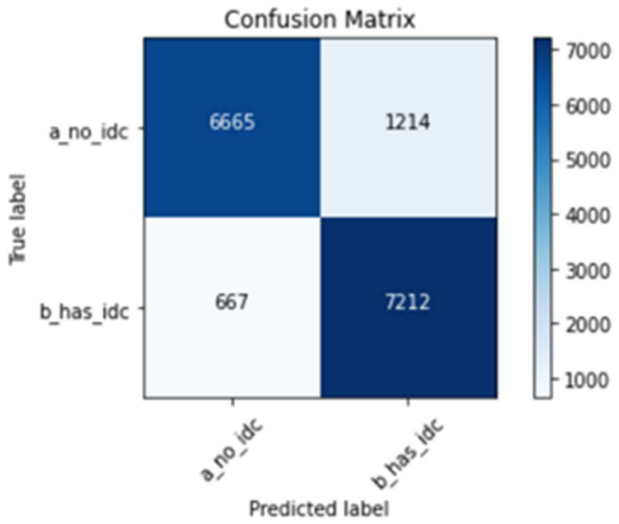


Fig 8.17 Screenshot of the confusion matrix

p) Creating a classification report

In [60]:
```
from sklearn.metrics import classification_report

# Generate a classification report

# For this to work we need y_pred as binary labels not as probabilities
y_pred_binary = predictions.argmax(axis=1)

report = classification_report(y_true, y_pred_binary, target_names=cm_plot_labels)

print(report)
```

```
              precision    recall  f1-score   support

    a_no_idc       0.91      0.85      0.88      7879
   b_has_idc       0.86      0.92      0.88      7879

    accuracy                           0.88     15758
   macro avg       0.88      0.88      0.88     15758
weighted avg       0.88      0.88      0.88     15758
```

Fig 8.18 Screenshot of the report consisting of the accuracy of the model

CHAPTER – 9

CONCLUSION

9. CONCLUSION

In our project, we have presented the capability of classifying the malignant and benign cancer tissues using the classification techniques with the help of microscopic image datasets. Convolutional Neural Network is one of the most emerging disruptive technologies that is used in many sectors like e-commerce, the agricultural sector, the health sector, the industrial sector, etc. The main aim of this model is to provide an earlier warning to the users and is also a cost and time-saving benefit to all users. The mortality rate of breast cancer is the highest among all other types of cancer, it can be detected early by detecting the breast nodules. In this system, image preprocessing and image segmentation are implemented to obtain the diagnosis result. By using these steps, the nodules are detected and some features are extracted. The extracted features can be used for the classification of disease stages. Determining the nodule features can provide to know more information about the condition of breast cancer at the early stages. In our project we have focused on this issue, we have developed a machine with improved accuracy which could assist radiologists in reviewing each mammogram, flagging potentially cancerous results that may have otherwise been missed for human review, and take correct decisions for breast cancer patient in short time with accuracy. Therefore, this method is less costly, less time-consuming, and easy to implement.

- Our project has attained the following levels of **POs** and **PSOs**

PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
3	3	2	2	3	2	1	2	3	2	3	2

PSO1	PSO2	PSO3
3	2	1

CHAPTER – 10

BIBLIOGRAPHY

10. BIBLIOGRAPHY

[1] “An Ad Hoc Random Initialization Deep Neural Network Architecture for Discriminating Malignant Breast Cancer Lesions in Mammographic Images”- Andrea Duggento, Marco Aiello, and Carlo Cavaliere, In WILEY (Hindawi), (AUG 2018).

[2] Jyotismita Talukdar, Dr. Sanjib Kr. Kalita: Detection of Breast Cancer using Data Mining (WEKA) November 2015.

[3] E. D. Pisano, C. Gatsonis, E. Hendrick et al., “Diagnostic performance of digital versus film mammography for breast-cancer screening,” New England Journal of Medicine, vol. 353, no. 17, pp. 1773–1783, 2005.

[4] R. L. N. Godone, G. M. Leitão, N. B. Arau´jo, C. H. M. Castelletti, J. L. Lima-Filho, and D. B. G. Martins, “Clinical and molecular aspects of breast cancer: targets and therapies,” Biomedicine & Pharmacotherapy, vol. 106, pp. 14–34, 2018.

[5] S. Hofvind, A. Ponti, J. Patnick et al., “False-positive results in mammographic screening for breast cancer in Europe: a literature review and survey programs screening programmes,” Journal of Medical Screening, vol. 19, no. 1, pp. 57-66, 2012.

[6] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in Proceedings of the European Conference on Computer Vision -ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8689, Lecture Notes in Computer Science, Zurich, Switzerland, September 2014.

[7] H.Chougrad, H. Zouaki, and O. Alheyane, “Deep convolutional neural networks for breast cancer screening,” Computer MethodsandProgramsinBiomedicine, vol.157, pp.19–30,2018.