

LUNG CANCER PREDICTION USING 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 & Engineering

Submitted by

J. Deepthi (20KD1A0562)

D. Prabhas (20KD1A0534)

B. Teja (20KD1A0516)

G. M Krishnam Naidu (21KD5A0505)

Under the guidance of

Mr. T. Thirupatirao, M. Tech

Assistant Professor

Department of CSE



Department of Computer Science & Engineering

LENDI INSTITUTE OF ENGINEERING & TECHNOLOGY

An Autonomous Institution

Affiliated to JNTU Gurajada, Vizianagaram, Approved by A.I.C.T.E,

Accredited by NAAC with "A" Grade & NBA

Jonnada, Vizianagaram Dist.535005.

2020-2024



LENDI INSTITUTE OF ENGINEERING & TECHNOLOGY (A)

*(Approved by A.I.C.T.E & Affiliated to JNTU Gurajada, Vizianagaram,
Accredited by NBA & Accredited by NAAC with 'A' GRADE)*

JONNADA, DENKADA (M), VIZIANAGARAM - 535005.

BONAFIDE CERTIFICATE

This is to certify that the project entitled “**LUNG CANCER PREDICTION USING CNN**” is a bonafide record of the work done by **J. Deepthi (20KD1A0562), D. Prabhas (20KD1A0534), B. Teja (20KD1A0516), G. M Krishnam Naidu (21KD5A0505)** under the supervision and guidance of **Mr. T.Thirupatirao, M. Tech, Assistant 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, (A), Jonnada, Vizianagaram for the year 2024.

Internal guide

Mr. T. Thirupatirao, M. Tech
Assistant Professor
Department of CSE

Head of the Department

Dr. A. Rama Rao, M. Tech, Ph.D
Professor
Department of CSE

External Examiner

ACKNOWLEDGEMENT

With great solemnity and sincerity, we offer our profuse thanks to our management, for providing all the resources to complete our project successfully. We express our deepest sense of gratitude and pay our sincere thanks to our guide **Mr. T. Thirupatirao M. Tech, Assistant Professor, Department of C.S.E**, who evinced keen interest in our efforts and provided his valuable guidance throughout our project work.

We thank our project coordinator **Mr. P.J.V.G. Prakasa Rao, Mr.A.Yugandhara Rao, Mr. B. Nageswara Rao** who has made his support available in a number of ways and helped us to complete our project work in correct manner.

We thank our **Dr. A. RAMA RAO**, Head of the Department of **Computer Science & Engineering** who 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 good lab facility.

We also thankful to **All Staff Members of Department of Computer Science & Engineering**, for helping us 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 for the success of this project which play a vital role.

J. DEEPTHI	(20KD1A0562)
D. PRABHAS	(20KD1A0534)
B. TEJA	(20KD1A0516)
G. M KRISHNAM NAIDU	(21KD5A0505)

DECLARATION

We hereby declare that the project work entitled “**Lung Cancer Prediction using CNN**” is a record of an original work done by **J. Deepthi (20KD1A0562), D. Prabhas (20KD1A0534), B. Teja (20KD1A0516), G. M Krishnam Naidu (21KD5A0505)** under the esteemed guidance of **Mr. T. Thirupatirao M. Tech, Assistant Professor**, Computer Science & Engineering, Lendi Institute of Engineering & Technology (A). This project work is submitted in the partial fulfillment of the requirements for the award of the degree **Bachelor of Technology in Computer Science & Engineering**. This entire project is done with the best of our knowledge and is not submitted to any university for the award of degree/diploma.

J. DEEPTHI	(20KD1A0562)
D. PRABHAS	(20KD1A0534)
B. TEJA	(20KD1A0516)
G. M KRISHNAM NAIDU	(21KD5A0505)



LENDI INSTITUTE OF ENGINEERING & TECHNOLOGY (A)

*(Approved by A.I.C.T.E & Affiliated to JNTU Gurajada, Vizianagaram,
Accredited by NBA & Accredited by NAAC with 'A' GRADE)*

JONNADA, DENKADA (M), VIZIANAGARAM - 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 behaviour, 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 latest tools and techniques of Computer Engineering so that they can analyze, design and create computing products and solutions for real life problems.

PEO-2: Graduates shall have multidisciplinary approach, professional attitude and ethics, communication and teamwork skills, and an ability to relate and solve social issues through computer engineering.

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 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 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 the 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 modelling to complex engineering activities with an understanding of the limitations.

PO-6 The Engineer and Society: Apply reasoning 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 in 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 own 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

Cancer is the uncontrollable cell division of abnormal cells inside the human body, which can spread to other body organs. Lung cancer is the most common cancer in both men and women and there are two main types of lung cancer: small cell lung cancer (SCLC) and non-small cell lung cancer (NSCLC). Non-small cell lung cancer makes up about 80 percent of lung cancer cases and this type of cancer usually grows and spreads to other parts of the body and there are three types of non-small cell lung cancer they are normal, adenocarcinoma and Squamous cell carcinoma. There are various methods for the diagnosis of lung cancer, such as Xray, CT scan, bronchoscopy and biopsy.

Histopathology refers to the examination by a pathologist of biopsy samples and these images are essential to investigate the status of a certain biological structures and to diagnose the disease. Histopathology image analysis is widely used for cancer grading compared to mammography, CT scan and it provides more comprehensive information for diagnosis and the diseases are analysed by detecting tissue and cells. Hence, to speed up the vital process of diagnosis of lung cancer and reduce the burden on pathologists, Deep learning techniques are used. In our proposed system we are using CNN with a combination of ReLU(Rectified Linear Unit) and BN (Batch Normalization).

Keywords: CNN, Rectified Linear Unit, Batch Normalization.

Outcomes: Our project titled “**Lung Cancer Prediction using 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

S No	Title	Page No
1	INTRODUCTION	1
	1.1 Project Overview	2
	1.2 Project Deliverables	2
	1.3 Project Scope	3
2	LITERATURE SURVEY	4
3	PROBLEM ANALYSIS	6
	3.1 Existing System	6
	3.1.1 Challenges	6
	3.2 Proposed System	7
	3.2.1 Advantages	8
4	SYSTEM ANALYSIS	9
	4.1 System Requirement Specification	9
	4.1.1 Functional Requirements	10
	4.1.2 Non - Functional Requirements	11
	4.2 Feasibility Study	12
	4.3 Use Case Scenarios	12
	4.3.1 Use case Diagrams	15
	4.4 System Requirements	15
	4.4.1 Software Requirements	16
	4.4.2 Hardware Requirements	16
5	SYSTEM DESIGN	17
	5.1 Introduction	17
	5.1.1 Class Diagram	17
	5.1.2 Sequence Diagram	21
	5.1.3 Activity Diagram	22
	5.2 System Architecture	24
	5.3 Algorithm Description	25

6	IMPLEMENTATION	27
	6.1 Technology Description	27
	6.1.1 Google Colab	27
	6.1.2 Python	27
	6.1.3 Python libraries	28
	6.2 Sample Source Code	30
7	TESTING	34
	7.1 Introduction	34
	7.2 Test Cases	39
8	SAMPLE SCREEN SHOTS	40
9	CONCLUSION	46
10	BIBLOGRAPHY	47

LIST OF FIGURES

Fig. No	Figure caption	Page No
4.3.1	Use case diagram	15
5.1.1	Class diagram	20
5.1.2	Sequence diagram	21
5.1.3	Activity diagram	23
5.2	CNN architecture	25
8.1	Importing libraries	40
8.2	Importing tensorflow and imagedatagenerator	40
8.3	Defining image size and folder path	40
8.4	Visualize the columns	41
8.5	Model summary	42
8.6	Training the model	43
8.7	Evaluating on the validation set	43
8.8	Evaluating on the test set	43
8.9	Importing necessary modules for metrics	43
8.10	Making predictions and plotting confusion matrix	44
8.11	Calculating precision, recall, and F1-score	45

LIST OF TABLES

Table No	Table Name	Page No
5.1	Graphical Representation of Sequence Diagram	21
7.2	Test cases	39

1. INTRODUCTION

Lung cancer presents a formidable challenge in the landscape of oncology, characterized by uncontrolled cellular proliferation within the lung tissues, leading to the formation of malignant tumors. Non-small cell lung cancer (NSCLC), comprising the majority of lung cancer cases, demonstrates remarkable heterogeneity, with distinct subtypes that demand nuanced approaches to diagnosis and treatment. Adenocarcinoma and squamous cell carcinoma emerge as predominant subtypes within NSCLC, each with its unique histological and clinical characteristics. Adenocarcinoma, the most common NSCLC subtype, originates from glandular or secretory cells and is frequently associated with smoking, although it can also affect nonsmokers, particularly women and younger individuals. Conversely, squamous cell carcinoma arises from squamous cells lining the airways of the lungs and represents a significant proportion of NSCLC cases. Understanding the molecular underpinnings, environmental factors, and clinical manifestations specific to these subtypes is paramount for accurate diagnosis and the development of tailored treatment strategies, ultimately improving patient outcomes in the battle against lung cancer.

In addition to adenocarcinoma and squamous cell carcinoma, other less common subtypes of NSCLC, such as large cell carcinoma and adenosquamous carcinoma, further contribute to the complex landscape of lung cancer. Large cell carcinoma is a subtype characterized by large, undifferentiated cells that lack the distinctive features of adenocarcinoma or squamous cell carcinoma. Adenosquamous carcinoma, on the other hand, exhibits both glandular (adenocarcinoma) and squamous cell features, presenting unique challenges in diagnosis and treatment planning. While these subtypes may be less prevalent, they underscore the need for comprehensive molecular profiling and individualized therapeutic strategies to address the diverse molecular alterations and clinical behaviors observed in NSCLC. Furthermore, ongoing research efforts aimed at elucidating the molecular mechanisms driving these subtypes hold promise for the development of novel targeted therapies and precision medicine approaches, offering hope for improved outcomes for patients with advanced NSCLC.

Lung cancer, a disease characterized by uncontrolled cell growth in the lungs, encompasses various subtypes, with non-small cell lung cancer (NSCLC) being the most prevalent. NSCLC accounts for the majority of lung cancer cases, approximately 85%. It originates in the lung tissues and has the potential to metastasize, spreading to other parts of the body. While smoking is a common risk factor associated with NSCLC, it also affects non-smokers. Interestingly, NSCLC shows a higher incidence in women and tends to manifest in younger individuals compared to other lung cancer types.

1.1 Project Overview:

The project focuses on the application of Convolutional Neural Networks (CNNs) for lung cancer detection, leveraging Rectified Linear Unit (ReLU) activation functions and batch normalization techniques. Initially, a thorough dataset collection and preprocessing phase are conducted to ensure data quality and suitability for training. The CNN architecture is carefully designed, incorporating multiple convolutional and pooling layers, with ReLU activation functions to introduce non-linearity and enhance feature learning. Batch normalization layers are integrated to stabilize and accelerate the training process. The model is trained using appropriate optimization algorithms and hyperparameters, with evaluation metrics such as accuracy, precision, recall, and F1-score utilized to assess its performance. Results are analyzed comprehensively, discussing both quantitative metrics and qualitative insights into the model's predictions. The project concludes with reflections on limitations, potential improvements, and the broader implications for lung cancer detection using deep learning methodologies.

1.2 Project Deliverables:

The project also aims to address several challenges inherent in lung cancer detection, such as the variability in nodule size, shape, and texture, by leveraging advanced deep learning techniques. The CNN architecture will be meticulously designed to capture intricate features indicative of malignant or benign nodules, enhancing the model's sensitivity and specificity. Furthermore, the integration of ReLU activation functions and Batch Normalization techniques will optimize the network's training dynamics, accelerating convergence and improving overall performance.

In addition to its diagnostic capabilities, the system will prioritize patient privacy and data security by implementing robust encryption and anonymization protocols. This will ensure compliance with regulatory standards such as HIPAA, safeguarding sensitive medical information throughout the diagnostic process.

Collaboration with medical professionals and stakeholders will be integral to the project's success. Regular feedback sessions and user testing will inform iterative improvements to the system's functionality and usability, ensuring alignment with clinical needs and workflow preferences.

Ultimately, the project aspires to revolutionize lung cancer detection and management, offering a state-of-the-art solution that enhances early detection rates, facilitates timely intervention, and ultimately improves patient outcomes. Through its multidisciplinary approach and commitment to excellence, the project endeavors to make a meaningful impact in the fight against lung cancer, saving lives.

1.3 Project Scope:

The scope of this project encompasses the design, development, and validation of an automated lung cancer detection system using Convolutional Neural Networks (CNNs), Rectified Linear Units (ReLUs), and Batch Normalization. This involves the collection and preprocessing of a comprehensive dataset of lung images, including X-rays and CT scans, to train and fine-tune the CNN models. The project aims to achieve high accuracy and sensitivity in identifying lung nodules and distinguishing between benign and malignant lesions. A significant aspect of the scope is the integration of the system into clinical workflows, ensuring it supports radiologists in making faster and more accurate diagnoses. Additionally, the project will address the scalability of the model to handle large volumes of data efficiently and its adaptability to accommodate advancements in imaging technology and medical research. Through rigorous testing and validation against established benchmarks, the project seeks to establish a reliable and effective tool for early lung cancer detection, ultimately contributing to better prognosis and treatment outcomes for patients.

To address scalability concerns, the project will explore parallel computing techniques and distributed processing frameworks to handle the growing volume of medical imaging data effectively. This scalability ensures that the system remains efficient and responsive even as the dataset size and computational requirements increase over time.

Additionally, ongoing monitoring and maintenance will be essential to keep the system up-to-date with evolving medical guidelines, technological advancements, and emerging research findings. Regular updates and refinements will ensure the system's continued relevance and effectiveness in the ever-changing landscape of lung cancer diagnosis and treatment.

Ultimately, the successful completion of this project aims to revolutionize early lung cancer detection by providing clinicians with a powerful and reliable tool that enhances diagnostic accuracy, accelerates treatment initiation, and improves patient outcomes.

2. LITERATURE SURVEY

- **Sheeraz Akram, Muhammad Rashid, and Arfan Jaffar, 2023**, Prior studies in lung cancer detection have explored diverse methodologies encompassing machine learning and deep learning approaches. Research has delved into U-Net-based segmentation for accurate lobe identification, while also investigating nodule extraction techniques employing modified architectures. Various classification models, including AlexNet and support vector machines (SVM), have been employed to distinguish between cancerous and non-cancerous nodules. These studies collectively demonstrate advancements in achieving high accuracy, sensitivity, specificity, precision, and F1 scores, notably enhancing the automated identification and classification of lung cancer from CT scans.
- **Tanima Thakur, Isha Batra, Arun Malik, 2023**, Prior studies have extensively explored machine learning and deep learning techniques for cancer prediction based on gene expression data. Research has investigated diverse models, including CNNs (Convolutional Neural Networks), RNNs (Recurrent Neural Networks), and hybrid architectures. Various pre-trained models such as VGG16, VGG19, ResNet50, Inception V3, and MobileNet have been employed for feature extraction and classification. These studies collectively demonstrate advancements in accurately predicting multiple cancer types, showcasing improvements in accuracy, Mean Square Error (MSE), precision, recall, and F1 score metrics using hybrid RNN-CNN classifiers, particularly outperforming standalone models on different datasets.
- **Heru Cahya Rustamaji, Wisnu Ananta Kusuma, Sri Nurdianti, 2023**, Previous research in network analysis has extensively explored traditional centrality measures like degree, closeness, and betweenness centrality in understanding complex systems. However, recent studies have addressed the limitations of these measures in capturing community structures within networks. Novel approaches, such as community-consideration centrality, have emerged to bridge this gap, incorporating a balanced weight (α) to assess both network-wide and community-specific importance. These methodologies have shown enhanced performance in identifying significant genes within networks, especially in the context of cancer-related protein networks, surpassing traditional centrality measures and even outperforming advanced algorithms. This research highlights the pivotal role of community structure in network analysis, offering a more nuanced understanding of centrality in complex systems, notably evident in the study of lung adenocarcinoma proteins.
- **Rabbia Mahum and Abdulmalik, 2023**, Recent research in lung cancer detection has explored various artificial intelligence techniques, notably RetinaNet-based approaches. Prior studies have emphasized multi-scale feature fusion methodologies to enhance semantic information extraction

across network layers. Additionally, the integration of dilated and lightweight algorithms for context modules has been a focal point, aiming to improve feature representation for precise localization of small tumors. These methodologies collectively aim at early tumor detection using CT scans and blood test results. State-of-the-art DL-based methods have been evaluated and compared, showcasing the effectiveness of these techniques in achieving high accuracy, recall, precision, F1-score, and AUC metrics, notably outperforming existing methodologies in detecting and assessing lung tumors accurately.

- **Al-Shouka, Ali Alheet, 2023**, The mortality rate from Lung Cancer is high, and the disease is notoriously difficult to treat. It was the third most frequent mortality reason for women and the leading cause of male mortality, according to data compiled by the International Agency for Research on Cancer (IARC) Cancer Research Facility. Additionally, they provided rates of cancer occurrence and mortality for a total of 36 different cancers occurring in 185 different countries. Typically, early tumor detection is essential for effective therapy. Early cancers can be efficiently examined to help patients recover quickly. Traditional medical imaging techniques, such as X-rays, CT scans, MRIs, etc., offer little promise for lung tumor identification. CNNs are capable of exact image processing. This paper provides a probabilistic deep CNN diagnosis method for lung cancer. This algorithm is considered the best in image classification, which gives better results, making the proposed system more efficient. The accuracy values obtained are 80%, the recall is 92%, and the F1-score is 87%.
- **Swetha Dhamercherla Damodar Reddy Edia, Suresh Dara, 2023**, Cancer is a complex disease caused by aberrant cell growth. Lung cancer is diverse, making genetic profiling essential for targeted treatment. Lung cancer kills 1.76 million people annually. Lung cancer's stealthy nature and nonspecific symptoms delay diagnosis compared to other malignancies. Radiographic imaging and invasive techniques improve diagnosis. Lung cancer is a priority due to its high cancer mortality rate, hence AI and ML have created cancer classification models. Risk assessment is crucial when choosing the appropriate preventative approaches. Effective prognosis and early therapies are key to lowering mortality rates. This paper proposed a nature-inspired optimization technique for cancer classification. First, we collect and preprocess patient data using a low-pass filter (LPF), PCA was used to extract key data properties. Our proposed Improved Support Vector Machine with Gradient Self-Adaptive Sea Lion Optimization improves lung cancer prediction. Reported and compared experimental results by using parametric measures like accuracy, precision, recall, and F1-score to assess the model's ability to accurately forecast lung cancer risk.

3. PROBLEM ANALYSIS

Problem analysis is the systematic process of identifying, understanding, and defining issues or challenges that hinder progress or desired outcomes. It involves breaking down complex problems into manageable components, examining root causes, and assessing their impact. Through careful analysis, individuals or teams can gain clarity on the underlying issues, uncover hidden factors, and identify potential solutions. Effective problem analysis requires critical thinking, data gathering, and often collaboration among stakeholders. By thoroughly understanding the problem, decision-makers can develop informed strategies to address it and achieve meaningful solutions.

3.1 Existing System:

Lung cancer, recognized as one of the deadliest forms of cancer, underscores the crucial significance of early detection for successful treatment and improved survival rates. This existing system presents a comprehensive approach to lung cancer classification, employing advanced computational intelligence techniques. The system comprises three integral phases: lobe segmentation, candidate nodule extraction, and lung cancer classification. Utilizing a modified U-Net architecture, the system accurately segments CT scans to derive lobes, followed by candidate nodule extraction using predicted lobes as input. Further, a modified AlexNet-SVM model is applied to classify candidate nodules as either cancerous or non-cancerous, enhancing the precision of lung cancer diagnosis. Through this integrated approach, the system contributes to timely detection and classification, ultimately aiming to improve patient outcomes and survival rates in lung cancer management.

This holistic approach not only enhances diagnostic accuracy but also fosters a proactive paradigm in lung cancer care, ultimately translating into tangible improvements in patient outcomes and quality of life.

3.1.1 Challenges:

- **Data Imbalance:** The text mentions 6802 patches each for cancer and non-cancer groups. In real-world scenarios, lung cancer cases might be fewer than non-cancer cases. This imbalance can lead the model to be biased towards the majority class (non-cancer) during training.
- **Limited Testing Data:** Limited Testing Data: The model was only tested on 1188 patches, which might not be statistically significant for generalizing the performance on unseen data, raising concerns about its robustness and reliability.

- **Black Box Model:** While the accuracy seems good, using a combination of Alexnet and SVM makes it difficult to interpret why the model predicts a certain class. This can be challenging for trusting the model's decisions in critical cases.
- **Overfitting Potential:** Dividing a relatively small dataset (858 CT scans) 80/20 for training and validation might lead to overfitting, where the model performs well on the training data but not on unseen data. The model trained on a narrow dataset may prioritize certain patterns and struggle to generalize to new lung cancer cases.

3.2 Proposed System:

In addressing the imperative need for efficient lung cancer diagnosis, our proposed system harnesses the power of deep learning techniques, particularly convolutional neural networks (CNNs), for histopathology image analysis. Given the complexities of lung cancer, including its prevalence and diverse subtypes, accurate diagnosis is paramount. Our system begins with the acquisition of histopathology images, obtained through the examination of biopsy samples by pathologists. These images serve as crucial inputs for investigating the status of biological structures and diagnosing the disease. Leveraging CNNs, specifically tailored for image analysis tasks, our proposed system aims to expedite the diagnostic process and alleviate the burden on pathologists.

The proposed CNN model undergoes training on a diverse dataset comprising histopathology slides of lung cancer samples. By utilizing large-scale annotated datasets, the model learns intricate patterns and features indicative of different cancer grades and subtypes. The training process is optimized to achieve high accuracy, with rigorous validation and testing conducted to ensure robustness and generalizability. Through iterative refinement and experimentation with various model architectures and hyperparameters, we strive to enhance the system's performance and reliability in accurately classifying lung cancer histopathology images.

Upon completion of training and validation, the proposed CNN model demonstrates promising results, exhibiting a training accuracy of 99%, validation accuracy of 98.35%, and testing accuracy of 98.72%. These metrics signify the efficacy of our system in accurately diagnosing lung cancer from histopathology images. Furthermore, comparative analyses are conducted to evaluate the performance of different variations of the proposed models, allowing for informed decisions regarding model selection and deployment. With its high accuracy and robust performance, our CNN-based system represents a significant advancement in the fight against lung cancer, offering hope for earlier detection and more effective treatment strategies.

3.2.1 Advantages:

- **Enhanced Efficiency:** By automating histopathology image analysis, the system accelerates the diagnostic process, reducing turnaround times and enabling timely patient management decisions.
- **Improved Accuracy:** Leveraging deep learning techniques, the system achieves high levels of diagnostic accuracy, ensuring reliable identification and classification of lung cancer histopathology images with minimal errors.
- **Reduced Workload:** Automation of labor-intensive tasks alleviates the burden on pathologists, enabling them to focus on more complex diagnostic challenges and enhancing overall productivity.
- **Comprehensive Analysis:** The system's ability to comprehensively analyze histopathology images captures subtle patterns and features indicative of different cancer grades and subtypes, providing valuable diagnostic insights.
- **Cost-Effectiveness:** Streamlining diagnostic processes and improving efficiency contribute to cost savings for healthcare institutions while potentially increasing access to accurate diagnoses and personalized treatment strategies for patients.
- **Enhanced Patient Care:** The system's automation of histopathology image analysis leads to quicker diagnoses and treatment decisions, ultimately improving patient outcomes. With reduced turnaround times, patients can receive timely interventions, leading to better management of their condition and potentially higher survival rates.
- **Advanced Diagnostic Capabilities:** Leveraging deep learning techniques enables the system to detect subtle patterns and features in histopathology images that may not be easily discernible to human pathologists. This enhanced level of analysis can uncover important insights into the nature and progression of lung cancer, facilitating more accurate staging and personalized treatment planning for patients.
- **Timely Patient Management:** The automated histopathology image analysis system accelerates the diagnostic process, significantly reducing turnaround times. This efficiency enables healthcare professionals to make timely patient management decisions, leading to quicker interventions and potentially improving patient outcomes.
- **Reduced Workload and Enhanced Productivity:** The automation of labor-intensive tasks in histopathology image analysis alleviates the burden on pathologists, allowing them to focus on more complex diagnostic challenges. By streamlining routine processes, the system increases overall productivity within healthcare institutions.

4. SYSTEM ANALYSIS

System analysis involves a comprehensive examination of requirements, functionalities, and constraints to design an efficient and effective system. It encompasses understanding the needs of stakeholders, defining objectives, and identifying specific features and functionalities that the system must deliver. System analysts utilize various techniques such as interviews, surveys, and workshops to gather requirements and clarify expectations. They then document these requirements using diagrams, flowcharts, and other visualization tools to ensure clear communication between stakeholders and development teams. System analysis serves as the foundation for the design and development phases, guiding decisions and ensuring alignment with project goals and objectives. Through systematic analysis, projects can mitigate risks, improve efficiency, and deliver solutions that meet user needs and expectations. Additionally, system analysis involves evaluating existing systems, identifying areas for improvement, and recommending enhancements or alternatives to optimize performance and functionality. This iterative process ensures that the final system meets quality standards and effectively addresses the needs of stakeholders.

It is a multifaceted process that involves assessing requirements, functionalities, and constraints to design an efficient system, encompassing stakeholder needs, objective definition, and feature identification. Utilizing techniques like interviews and surveys, analysts gather requirements and clarify expectations, documenting them through visualization tools to ensure clear communication. They evaluate feasibility, manage risks, and create detailed documentation, facilitating collaboration among stakeholders and guiding decision-making throughout the development lifecycle. Additionally, system analysts play a crucial role in translating technical concepts, facilitating iterative refinement, and ensuring alignment with user needs and business objectives, ultimately contributing to the success of software projects.

4.1 System Requirement Specification:

A System Requirements Specification (SRS) is a detailed document outlining the functional and non-functional requirements of a software system. It provides a comprehensive overview of the system's purpose, features, interfaces, constraints, and quality attributes. The SRS serves as a blueprint for the development team, guiding the design, implementation, and testing of the system to ensure it meets the needs and expectations of its users and stakeholders. It includes specific descriptions of system functionality, external interfaces, constraints, and assumptions, along with criteria for verification and validation. The SRS aids in facilitating communication between stakeholders and the development team.

4.1.1 Functional Requirements:

Functional requirements define the specific behaviors and functionalities that a software system must exhibit to satisfy user needs and achieve its intended purpose. These requirements describe what the system should do, such as processing data, performing calculations, or interacting with users. They typically include features, functions, and interactions with external systems or users. Functional requirements serve as the foundation for system design, development, and testing, ensuring that the software meets the expectations of its users and stakeholders. They are documented in detail to provide clear guidelines for implementation and validation, guiding the development team in building a system that aligns with the desired functionality and behavior.

- **Data Preprocessing:** The system begins by preprocessing input data, going beyond basic techniques like resizing and normalization. It incorporates advanced methods such as histogram equalization and geometric transformations to enhance feature representation and improve model robustness. Quality control mechanisms are integrated to detect and filter out corrupted or low-quality images, ensuring data integrity throughout the preprocessing pipeline.
- **Model Definition:** Users define the CNN architecture using modular design principles. They can easily customize and experiment with different network configurations, layer architectures, and hyperparameters. The system facilitates the integration of pre-trained models or model components, enabling transfer learning and leveraging domain-specific features for enhanced performance.
- **Model Training:** The system offers advanced optimization algorithms like stochastic gradient descent variants and adaptive learning rate methods to effectively train the CNN model while minimizing overfitting. Real-time monitoring and visualization of training progress, including metrics such as training loss, validation accuracy, and convergence behavior, facilitate model debugging and optimization.
- **Model Evaluation:** Beyond standard metrics, the system supports comprehensive analysis of model performance across different data subsets. It conducts subgroup analysis based on patient demographics or disease characteristics and enables sensitivity analysis to assess model robustness. Comparison with benchmark models or clinical guidelines evaluates clinical relevance and applicability.
- **Performance Metrics Calculation:** In addition to traditional performance metrics, the system computes uncertainty estimates, calibration curves, and confidence intervals to quantify model uncertainty and assess its impact on decision-making reliability. It supports model interpretability techniques such as feature importance ranking and attribution methods to elucidate the underlying factors driving model predictions, facilitating trust and transparency.

- **Visualization:** The system offers interactive visualization tools such as t-SNE embeddings and attention maps to provide intuitive insights into model behavior and decision boundaries. It enables visualization of model interpretability techniques such as SHAP values and Grad-CAM visualizations, elucidating the rationale behind individual predictions and supporting clinical decision-making.

4.1.2 Non-Functional Requirements:

Non-functional requirements specify the qualities and constraints that define how a system should perform rather than what it should do. These requirements encompass aspects such as performance, reliability, security, usability, and scalability. Performance requirements dictate the system's response times, throughput, and resource utilization under specific conditions. Reliability requirements ensure the system operates consistently and reliably over time, with minimal downtime or errors. Security requirements address data protection, access control, and compliance with regulatory standards to safeguard sensitive information. Usability requirements focus on the system's ease of use, accessibility, and user experience. Scalability requirements outline the system's ability to handle increasing workloads and adapt to changing user demands. Non-functional requirements are crucial for ensuring the overall quality, effectiveness, and user satisfaction of the system.

- **Scalability:** In addition to efficient processing of large-scale datasets, the system should be scalable to accommodate increasing data volumes and model complexities over time. It should support distributed computing frameworks and parallel processing techniques to leverage computational resources effectively and maintain performance scalability as dataset sizes grow.
- **Generalization:** The system should demonstrate robust generalization capabilities, performing consistently across diverse datasets and patient populations. It should undergo rigorous validation on external datasets and real-world clinical scenarios to ensure that the model's performance extends beyond the training environment and is applicable to new cases encountered in practice.
- **Reliability:** The system should exhibit high reliability, with built-in error handling mechanisms and robustness to system failures or interruptions. It should incorporate data integrity checks, model versioning, and automatic recovery procedures to minimize downtime and ensure continuous operation in clinical settings where uninterrupted access is critical.
- **Customizability:** The system should allow for customization and adaptation to specific clinical contexts and user preferences. It should support parameter tuning, model fine-tuning, and the incorporation of domain-specific knowledge or expert feedback to tailor the lung cancer detection model to the unique characteristics of different patient populations or imaging modalities.

- **Integration:** The system should seamlessly integrate with existing healthcare IT infrastructure, electronic health record (EHR) systems, and diagnostic workflows to facilitate adoption and interoperability. It should adhere to industry standards for data exchange, security, and privacy, enabling seamless data flow and communication with other healthcare systems and stakeholders.
- **Scalability:** Moreover, the system should be capable of handling increasing user loads and concurrent requests without compromising performance or responsiveness. It should employ scalable architecture patterns such as microservices, containerization, and auto-scaling to dynamically adapt to changing demand patterns and ensure optimal resource utilization under varying workload conditions.

4.2 Feasibility Study:

The feasibility study of implementing a deep learning-based histopathology image analysis system for the diagnosis of lung cancer appears promising. Given the prevalence of lung cancer as the most common form of cancer and the critical role of histopathology in its diagnosis, there is a clear need for more efficient and accurate diagnostic tools. Deep learning techniques have demonstrated significant improvements in analyzing histopathology slides, offering enhanced efficiency compared to traditional methods. The reported results of the proposed convolutional neural network (CNN) model achieving high training, validation, and testing accuracies further support the feasibility of the project. With training accuracy of 99%, validation accuracy of 98.35%, and testing accuracy of 98.72%, the model shows robust performance across various evaluation metrics. Additionally, the comparison of different variations of the proposed models highlights the potential for optimization and refinement to further enhance performance. Overall, the feasibility study suggests that leveraging deep learning techniques for histopathology image analysis in lung cancer diagnosis is not only viable but also holds significant promise for improving diagnostic accuracy and efficiency, ultimately the feasibility study underscores the potential of deep learning-based histopathology image analysis as a promising avenue for enhancing lung cancer diagnosis, offering both efficiency and accuracy benefits.

4.3 Use Case Scenarios:

Use case scenarios provide concise descriptions of how users interact with a system to achieve specific goals. Each scenario outlines the actor, goal, preconditions, main flow of actions, alternate flows, postconditions, and exceptions. These scenarios offer valuable insights into user requirements, system behavior, and potential error handling. By focusing on user perspectives and real-world interactions, use case scenarios serve as effective communication tools for stakeholders

and development teams, guiding system design and ensuring that the resulting software meets user needs and expectations.

- **Initial Scan and Data Preparation:**

- User selects the option to "Get Scan Images" and uploads CT scan images of a patient's lungs.
- The system processes the uploaded scans and unpacks them into images and NumPy files for further analysis.

- **Data Review and Preprocessing:**

- User accesses the uploaded CT scan images to review them by selecting "View CT Images."
- Radiologist examines the images to identify any abnormalities or suspicious regions.

- **Prediction and Analysis:**

- User selects the "Make a Prediction" option to initiate the lung cancer diagnosis process.
- The system performs preprocessing on the raw scan data, including normalization and feature extraction.
- A convolutional neural network (CNN) model, trained on histopathology images, is applied to predict the likelihood of lung cancer presence.

- **Viewing Predicted Results:**

- After prediction, the user can view the predicted results by selecting "View Predicted Results."
- The system displays the predicted probabilities or classifications for each scan, indicating the presence or absence of lung cancer.

- **Radiologist's Review and Interpretation:**

- Radiologist accesses the predicted results and reference images to validate the predictions.
- The system provides tools for the radiologist to overlay contours or masks on the original CT scan images, highlighting regions identified as potential tumors.

- **Final Diagnosis and Output:**

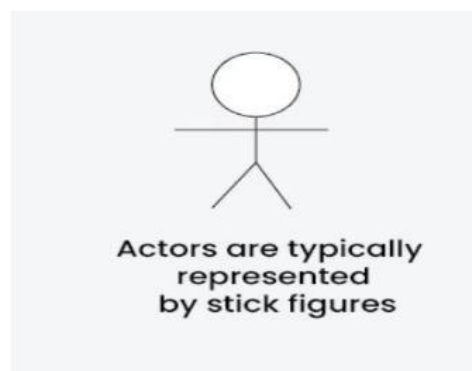
- Based on the radiologist's review and analysis, a final diagnosis is made.
- The system generates output, including diagnostic reports and annotated images, which can be shared with healthcare professionals for further consultation and treatment planning.
- The radiologist reviews the findings and confirms the final diagnosis, ensuring accuracy and reliability in patient management decisions.

Purpose of Use Case Diagrams:

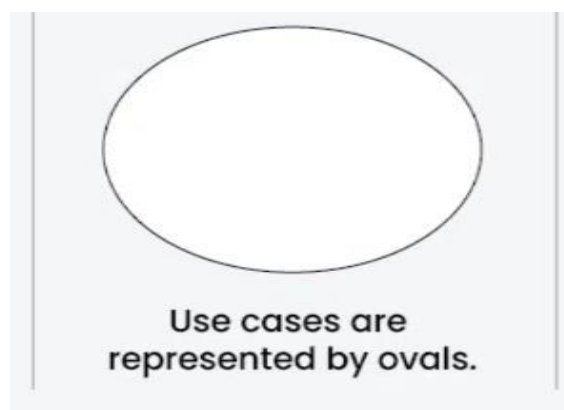
Use case diagrams are fundamental tools in software engineering, serving the primary purpose of illustrating the dynamic behavior of a system. They act as a repository for system requirements, encompassing both internal processes and external influences. By depicting actors, use cases, and their interactions, these diagrams provide a comprehensive view of how various

entities from the external environment engage with different parts of the system. Through visual representation, stakeholders gain insight into the system's functionality and its alignment with user needs. Additionally, use case diagrams aid in feature prioritization, enabling project teams to focus on key functionalities during development. They also facilitate requirement validation by visualizing user scenarios and workflows, allowing for effective testing and verification processes. Furthermore, use case diagrams help in identifying potential risks and dependencies, empowering teams to mitigate issues early in the development lifecycle. Overall, these diagrams play a pivotal role in requirements analysis, design, and validation, ensuring the successful delivery of software systems that meet user expectations.

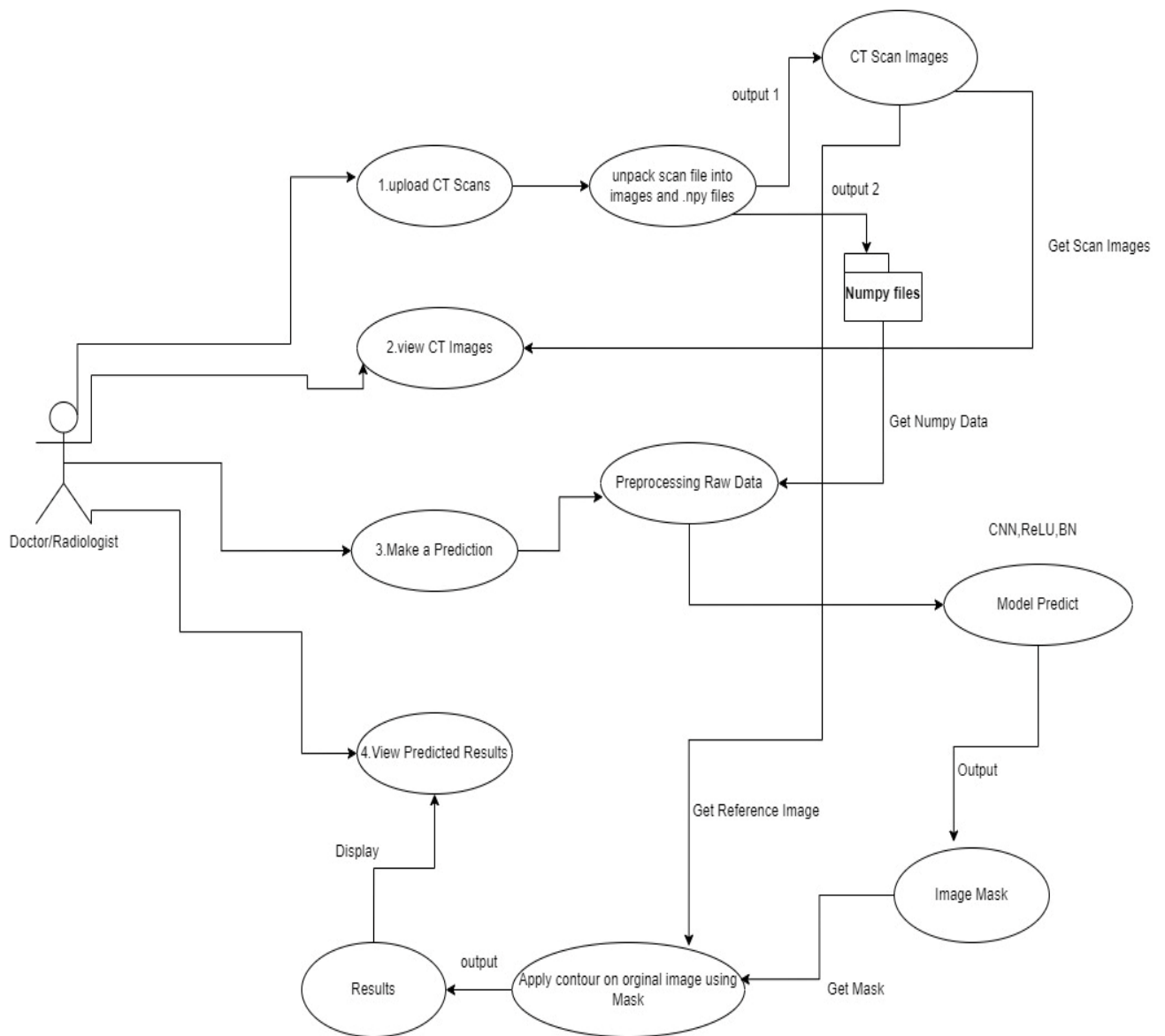
Actor



Use Case



4.3.1 Use case Diagram:



4.3.1 Use case diagram

4.4 System Requirements:

System requirements serve as a foundational framework for both developers and users, ensuring that the system's capabilities align with operational needs and constraints. By clearly defining these parameters, stakeholders can effectively manage expectations, allocate resources, and assess the system's performance against established benchmarks. Additionally, system requirements facilitate communication among multidisciplinary teams, fostering collaboration and clarity throughout the development lifecycle.

4.4.1 Software Requirements:

- Python Libraries - *Tensorflow, Flask, PIL, Numpy, Pickle, Pillow, SciPy*
- Visual Studio Code
- Jupyter Notebook
- Google Cloud Platforms

4.4.2 Hardware Requirements:

- Processor : i3 (or above)
- Speed : 2 GHZ
- RAM : 4 GB
- HDD/SSD : 500 GB

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.

The systems design approach first appeared right before World War ii, when engineers were trying to solve complex control and communications problems. They needed to be able to standardize their work into a formal discipline with proper methods, especially for new fields like information theory, operations research and computer science in general.

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 which can be mapped directly with object-oriented languages.

- **Purpose:**

The purpose of the class diagram is to model the static view of an application. The class diagrams are only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction. The uml diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but 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.
- Describe the responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

- **Active Class:**

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.

- **Visibility:**

Use visibility markers to signify who can access the information which is in a class. There are three visibilities. They are:

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

- **Associations:**

Associations represent 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 roles at the end of an association. Roles represent how the two classes see each other.


Association

- **Multiplicity (Cardinality):**

Multiplicity, also referred to as cardinality, plays a pivotal role in UML diagrams, particularly in class diagrams where it elucidates the relationships between classes. Positioned at the ends of associations, multiplicity notations signify the number of instances of one class linked to instances of another class. These notations can take various forms, such as "1" for one-to-one relationships, "0..1" for optional one-to-one relationships, and "*" for many-to-many relationships. By employing numerical ranges or symbols, like "*", UML diagrams effectively convey the complexity and nature of associations within a system. Understanding multiplicity aids in accurately representing the structure and behavior of the system, ensuring clarity and precision in the visualization of class relationships. Multiplicity is essential for accurately

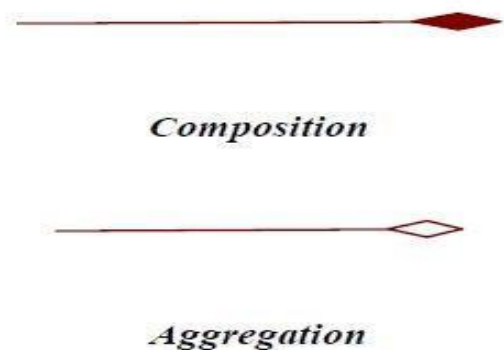
modeling the relationships between classes in a system. Mastering multiplicity is fundamental for creating precise and understandable visualizations that effectively represent the structure and behavior of the system's class relationships.

- **Constraint:**

Constraints serve as crucial elements for specifying conditions or rules that must be satisfied within the system. These constraints are typically represented using curly braces {} and can encompass various aspects of system behavior, data validation, or business logic requirements. Placed within the diagram's structure, constraints provide additional context and specificity to the design, ensuring that the system functions correctly under specified conditions. By encapsulating specific rules or conditions, constraints help in clarifying the system's intended behavior and constraints imposed on its elements, aiding both developers and stakeholders in understanding and validating the system's requirements.

- **Composition and Aggregation:**

Composition and Aggregation in UML diagrams establish relationships between classes, but with nuanced differences. Composition, depicted by a filled diamond, signifies a strong "whole-part" relationship, indicating ownership. Aggregation, represented by an unfilled diamond, implies a looser association, denoting that one class is part of or collaborates with another. They are used in class diagram. They both differ in their symbols.



- **Generalization:**

Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher-level entity if they have some attributes in common. In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher-level entity. Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach. In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.



Generalization

- **Class Diagram:**

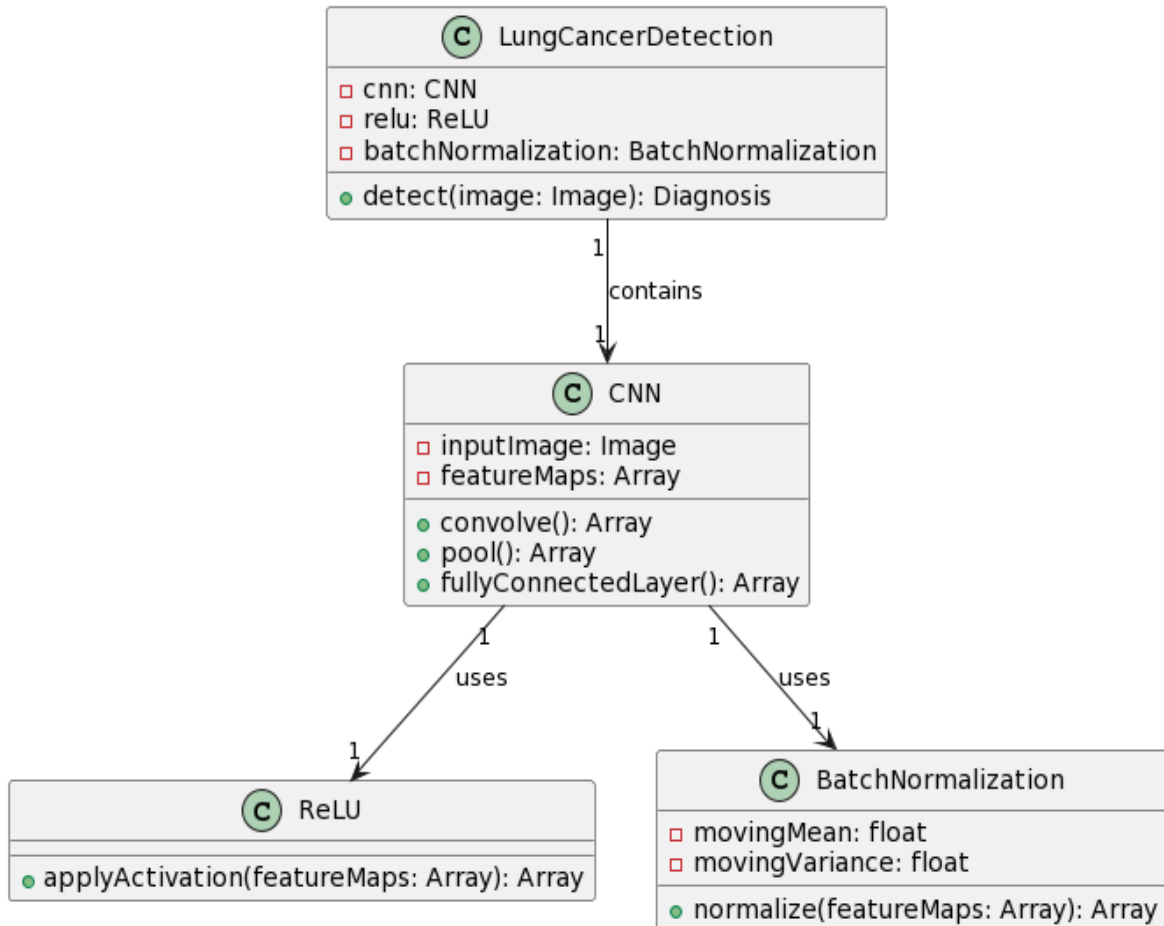


Fig 5.1.1 Class Diagram for Lung cancer detection

- **Class Diagram Scenario:**

In this class scenario, we have a "Lung Disease Prediction System" comprising several interconnected components for predicting lung disease based on input data. The system involves the utilization of a "CNN Module" for processing raw data, which is received as a string input. This module applies specific algorithms defined by the "model" parameter to generate predictions. Additionally, there are preprocessing steps involved, handled by the "Parameters" class, which preprocesses the raw data and produces normalized data. The "ReLU Activation" and "Batch Normalization" components are used for activation and normalization, respectively, enhancing the effectiveness of the prediction process. Furthermore, the system

incorporates "Activated Data" and "Normalized Data" as intermediary steps in the prediction pipeline. Overall, the system's architecture is designed to efficiently process input data, apply appropriate algorithms, and produce accurate predictions regarding lung disease.

The class scenario outlines a structured system for predicting lung disease, leveraging a CNN module, preprocessing steps, and activation and normalization components to enhance prediction accuracy and efficiency. The scenario describes a systematic approach utilizing CNN-based algorithms and preprocessing techniques to predict lung disease accurately.

5.1.2 Sequence Diagram:

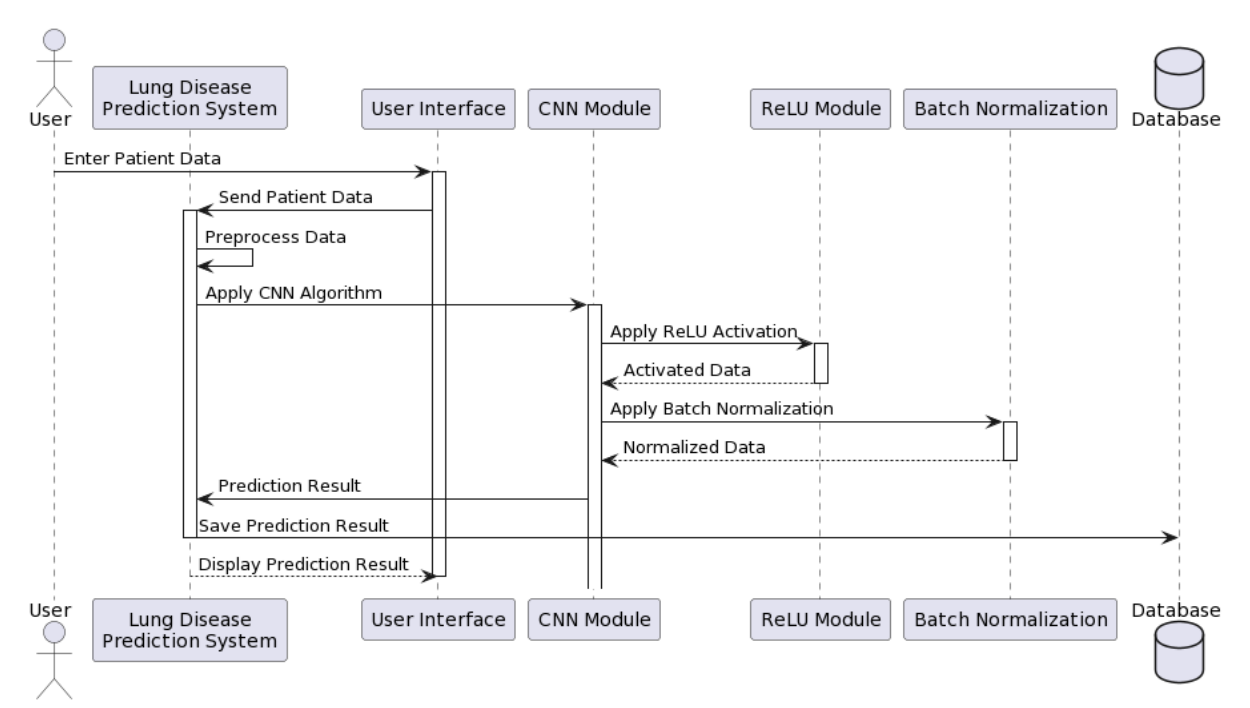


Fig 5.1.2 Sequence Diagram of Lung cancer detection

Object	Objects are instances of classes and are arranged horizontally. The pictorial representation for an Object is class (a rectangle) with the name prefixed by the object name (optional).	<div>Object1</div>
--------	---	--------------------




Actor	Actor can also communicate with objects so they too can be listed as a column. An Actor is modeled using the stick figure.	
Lifeline	The Lifeline identifies the existence of the object over time. The notation for a lifetime is a vertical dotted line extending from an object.	
Activation	Activation modeled as rectangular boxes on the lifeline indicate when the object is performing an action.	

Table 5.1 Graphical Representation of Sequence Diagram

5.1.3 ACTIVITY DIAGRAM

Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. It is a type of behavioral diagram and we can depict both sequential processing and concurrent processing of activities using an activity diagram i.e. An activity diagram focuses on the condition of flow and the sequence in which it happens.

Purpose of Activity Diagram:

Activity diagrams serve as powerful visual tools in the field of software engineering and business process modeling. Their primary purpose is to depict the flow of activities or actions within a system, process, or workflow. These diagrams provide a clear and concise representation of the sequence of activities, decisions, and transitions that occur during the execution of a particular process.

One of the main objectives of activity diagrams is to facilitate communication and understanding among stakeholders involved in the development or analysis of a system or process. By presenting complex workflows in a graphical format, activity diagrams help stakeholders

visualize the various steps involved, the order in which they occur, and the decision points that influence the flow of activities. Activity diagrams are instrumental in software engineering and business process modeling, offering a visual depiction of activity flows and decision points. They aid stakeholders in comprehending process sequences and facilitating effective communication during system development or analysis.

Notations Of Activity Diagram:

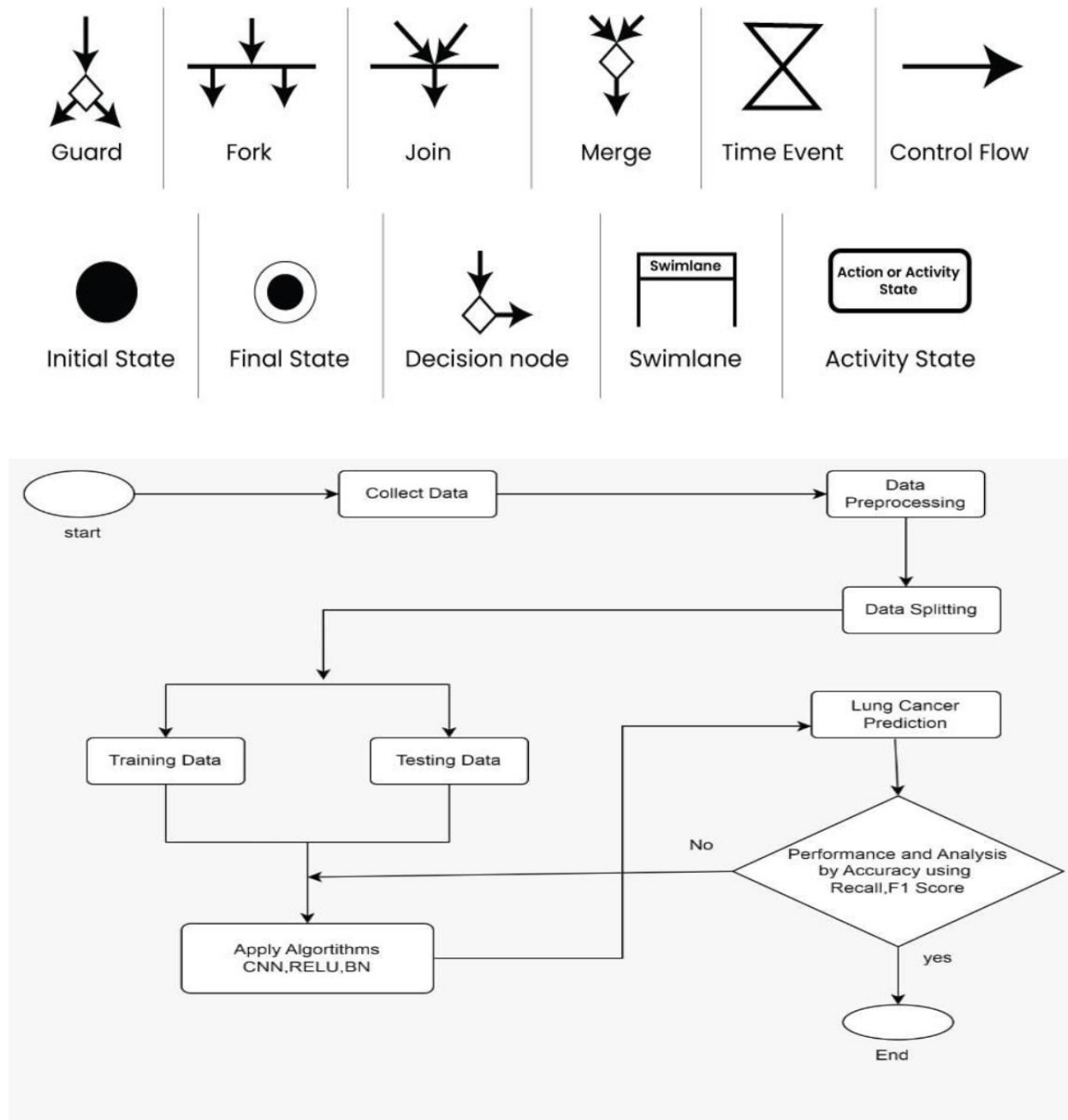


FIG 5.1.3 Activity diagram

5.2 System Architecture:

System architecture refers to the design and structure of a complex system, encompassing both software and hardware components, as well as their interactions and relationships. It serves as the blueprint for building and maintaining the system, providing guidelines for development, deployment, and scalability. A well-designed system architecture ensures that various components work together seamlessly to achieve the desired functionality while meeting performance, reliability, and security requirements.

At its core, system architecture involves defining the components of the system, their functionalities, and how they communicate with each other. This includes identifying the different layers of the system, such as presentation, application logic, and data storage layers, and determining how they interact. Decisions about technology stack, data flow, and communication protocols are crucial aspects of system architecture, influencing the system's performance, flexibility, and maintainability.

Moreover, system architecture plays a pivotal role in ensuring the scalability and adaptability of the system over time. A well-designed architecture anticipates future growth and technological advancements, allowing for seamless integration of new features and functionalities. Scalability considerations encompass both horizontal and vertical scaling, enabling the system to handle increasing loads and user demands without compromising performance or stability. Additionally, system architecture addresses concerns related to security and privacy, implementing robust measures to safeguard sensitive data and mitigate potential risks. Through careful planning and iteration, system architects strive to create a robust and resilient architecture that can evolve in tandem with changing business needs and technological landscapes.

System architecture also encompasses non-functional requirements such as usability, maintainability, and reliability. Usability considerations involve designing interfaces and interactions that are intuitive and user-friendly, enhancing user adoption and satisfaction. Maintainability aspects focus on facilitating system maintenance and updates, ensuring that changes can be implemented efficiently without disrupting system functionality. Reliability entails designing the system to minimize downtime and errors, implementing fault tolerance mechanisms and redundancy where necessary. By addressing these non-functional requirements alongside functional ones, system architecture aims to deliver a holistic solution that not only meets business objectives but also delivers a positive user experience and can be effectively managed and sustained over its lifecycle.

In summary, a meticulously crafted system architecture lays the groundwork for a robust, adaptable, and secure system that meets both current and future demands.

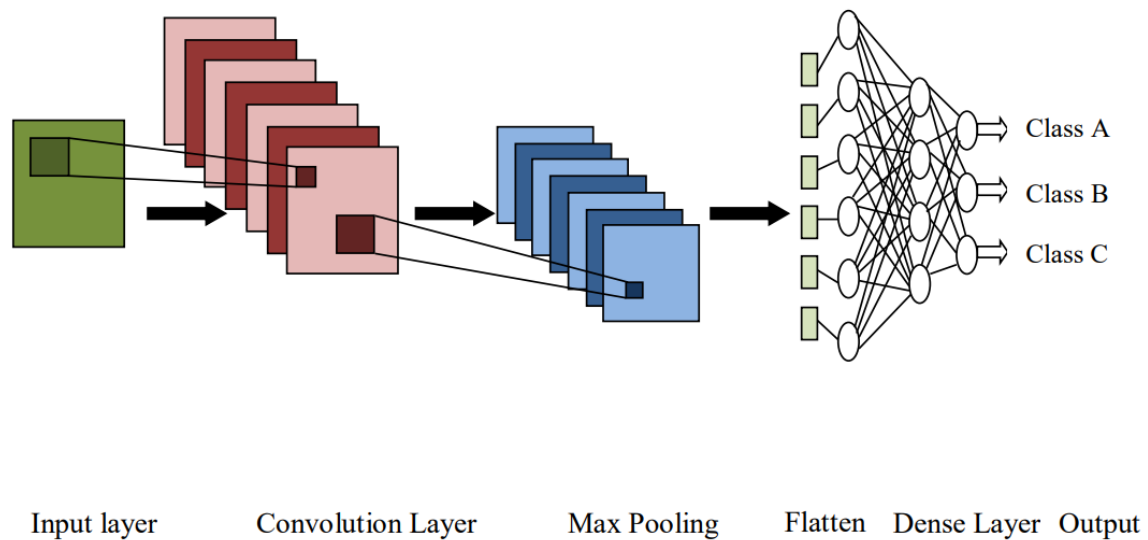


Fig 5.2 CNN architecture

5.3 Algorithm Description:

An algorithm is a step-by-step procedure designed to solve a problem or perform a specific task. It comprises a set of well-defined instructions executed in a precise sequence to achieve a desired outcome. Algorithms are fundamental to computer science and programming, underpinning various applications across industries, from data analysis and machine learning to cryptography and optimization. Their efficiency, accuracy, and scalability are essential considerations in algorithm design, impacting the performance and effectiveness of software systems. Through continuous refinement and innovation, algorithms drive advancements in technology, enabling the development of increasingly sophisticated solutions to complex problems. Algorithms are precise sequences of instructions, fundamental to solving problems in computer science and driving technological advancements.

Step 1: Import necessary libraries such as NumPy, Matplotlib, Keras, TensorFlow, and scikit-learn.

Step 2: Define the size of images and the path to the dataset folder.

Step 3: Create data generators for the training, validation, and test sets using Image Data Generator.

Step 4: Construct a CNN model using the Sequential API of TensorFlow/Keras.

Step 5: Include convolutional layers, batch normalization, max-pooling, flatten layer, and dense layers with appropriate activation functions.

Step 6: Train the defined model using the training set.

Step 7: Validate the model's performance on the validation set for a 20 number of epochs.

Step 8: Evaluate the trained model on the validation set and calculate validation loss and accuracy

Step 9: Evaluate the trained model on the test set to assess its performance on unseen data.

Step 10: Calculate test loss and accuracy.

Step 11: Import necessary modules for calculating performance metrics such as confusion matrix, precision, recall, and F1-score.

Step 12: Make predictions on the test set using the trained model.

Step 13: Calculate the confusion matrix and plot it to visualize the model's performance.

Step 14: Compute precision, recall, and F1-score based on the predicted and true labels of the test set.

Step 15: Print out the calculated metrics and display the validation loss and accuracy.

Step 16: Show the test loss, accuracy, and performance metrics including the confusion matrix, precision, recall, and F1-score.

6. IMPLEMENTATION

6.1 Technology Description:

6.1.1 Google Colab:

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs. Graphics Processing Unit (GPU) is extensively used for large data with mathematical and graphical computations and having more logical cores leads to simultaneous computation of multiple tasks which enhances the speed of computations. One of the key benefits of using Google Colab for a final report is that it provides a collaborative workspace that allows multiple users to work on the same document simultaneously. This means that team members can work together to create a polished, professional final report, even if they are located in different parts of the world. Colab provides an online platform to students, a researcher working around Artificial Intelligence, Machine Learning and Deep Learning to train their complex 38 model on the large dataset for free, on high-end remote machines with the easy sharing option. Google Colab is that it offers built-in support for a wide range of Python libraries and frameworks, including NumPy, Pandas, TensorFlow, and PyTorch. This means that users can leverage the power of these tools to analyze and visualize data, build machine learning models, and more, all within the same platform.

6.1.2 Python:

Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to software development and scientific applications. It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

- **Features of Python**

A simple language which is easier to learn, Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

Free and Open-Source: Python is freely available for both personal and commercial use, allowing users to distribute, modify, and even sell software written in Python. Its open-source

nature fosters a large and active community of developers continually improving the language with each iteration.

Portability: Python programs can be easily moved from one platform to another without any modifications, running seamlessly on various operating systems including Windows, Mac OS X, and Linux.

Extensible and Embeddable: Python allows for easy integration with other languages like C/C++, enabling developers to combine high-performance code with Python's scripting capabilities, providing both efficiency and flexibility in application development.

High-Level, Interpreted Language: Python abstracts away complex tasks such as memory management and garbage collection, making it more user-friendly compared to languages like C/C++. Additionally, Python's interpreter automatically translates code into machine-readable language, eliminating the need for low-level operations.

Large Standard Libraries:

Python boasts a plethora of standard libraries that streamline common programming tasks. These libraries, such as MySQLdb for connecting to MySQL databases, are well-tested and widely used, ensuring reliability and efficiency in application development.

Object-oriented:

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively. With OOP, you are able to divide these complex problems into smaller sets by creating object.

6.1.3 Python libraries:

- **NumPy:**

NumPy is the fundamental package for scientific computing in Python. NumPy support for multidimensional arrays, which can be used to represent vectors, matrices, and other complex data structures. NumPy provides a wide range of functions for performing common operations on these arrays, including basic arithmetic operations, matrix operations, and statistical functions.

- **Seaborn:**

Seaborn is a Python data visualization library used for statistical graphics plotting. It provides a high-level interface for creating informative and attractive statistical graphics, and is particularly useful for working with complex datasets and for creating visualizations that emphasize relationships between variables. Seaborn is used for beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures. Making it a versatile choice for data

visualization tasks in Python. Its emphasis on aesthetics and informative graphics makes it a popular choice among data scientists and analysts seeking to convey insights effectively.

- **Matplotlib:**

Matplotlib is a Python data visualization library that provides a wide range of tools for creating static, animated, and interactive visualizations in Python. Matplotlib is also highly customizable, allowing users to create visualizations that meet their specific needs. It provides a range of tools for controlling the layout of plots, including subplots and grids, and for adjusting the appearance of individual elements of a plot, such as ticks and labels.

- **TensorFlow:**

TensorFlow is an open-source machine learning framework developed by Google. It was first released in 2015 and has since become one of the most widely used and popular machine learning libraries in the world. TensorFlow is designed to handle the computation of large-scale neural networks, which are used for various tasks such as image classification, speech recognition, natural language processing, and many others. It provides a flexible and powerful programming interface for building and training machine learning models. One of the key features of TensorFlow is its use of data flow graphs, which represent the computation of a machine learning model as a directed graph. This allows TensorFlow to efficiently distribute computation across multiple processors or even across multiple machines in a cluster, making it suitable for large-scale distributed training of deep learning models.

- **Keras:**

Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation. Keras provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity. Keras provides a range of building blocks for constructing CNN models, including layers for convolution, pooling, normalization, and activation. These building blocks can be combined to create a wide variety of CNN architectures, and Keras provides a number of pre-built models that can be easily adapted to specific use cases.

- **OS module:**

OS module is a built-in Python module that provides a way to interact with the operating system. It provides a range of functions for working with files and directories, managing processes, and accessing system information. Some of the commonly used functions provided by the OS module include getting the current working directory, creating and deleting directories, renaming and deleting files, executing shell commands, and accessing

environment variables. The OS module is platform-independent and provides platform-specific functionality as well, making it an essential tool for any Python programmer working with files, directories, system administration, or process management.

- **Pandas:**

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. It provides tools for working with structured data such as spreadsheets, databases, and CSV files. Some of the key features of Pandas include. Some of the key features of Pandas also include data cleaning and preparation, Data exploration and data visualization.

6.2 Sample Source Code:

These are the sample packages we used during the implementation:

```
!pip install kaggle
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d srinivasbece/lung-cancer-preprocessed-dataset
!unzip -q lung-cancer-preprocessed-dataset.zip
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import keras.utils
import os
import pandas as pd
from sklearn.model_selection import train_test_split
%matplotlib inline
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
picture_size = 224
folder_path = '/content/lung'
batch_size = 128
datagen_train = ImageDataGenerator()
datagen_val = ImageDataGenerator()
```

```

datagen_test = ImageDataGenerator()
train_set = datagen_train.flow_from_directory(folder_path+"/Train",
target_size=(picture_size, picture_size),
color_mode="rgb",
batch_size=batch_size,
class_mode='categorical',
shuffle=True)
test_set = datagen_test.flow_from_directory(folder_path+"/Test",
target_size=(picture_size, picture_size),
color_mode="rgb",
batch_size=batch_size,
class_mode='categorical',
shuffle=False)
val_set = datagen_val.flow_from_directory(folder_path+"/Val",
target_size=(picture_size, picture_size),
color_mode="rgb",
batch_size=batch_size,
class_mode='categorical',
shuffle=False)
model = tf.keras.models.Sequential([
tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(224, 224, 3), padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D((2,2)),
tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D((2,2)),
tf.keras.layers.Conv2D(128, (3,3), activation='relu', padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D((2,2)),
tf.keras.layers.Conv2D(256, (3,3), activation='relu', padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D((2,2)),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(512, activation='relu'),
tf.keras.layers.Dense(3, activation='softmax')

```

```

])
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_set, validation_data=val_set, epochs=30)
model.save("lung_cancer_model.h5")
# Part 7: Evaluating on the validation set
loss, accuracy = model.evaluate(val_set)
print('Validation loss:', loss)
print('Validation accuracy:', accuracy)
# Part 8: Evaluating on the test set
loss, accuracy = model.evaluate(test_set)
print('Test loss:', loss)
print('Test accuracy:', accuracy)
# Part 9: Importing necessary modules for metrics
import numpy as np
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support
# Part 10: Making predictions and plotting confusion matrix
predictions = model.predict(test_set)
y_pred = [np.argmax(probas) for probas in predictions]
y_test = test_set.classes
class_names = test_set.class_indices.keys()
from sklearn.metrics import confusion_matrix, classification_report
import itertools
# Function to plot confusion matrix
def plot_confusion_matrix(cm, classes, title='Confusion matrix', cmap=plt.cm.Reds):
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(7, 5))
    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 = '.0f'
    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[j, i] > thresh
    else "black")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.tight_layout()
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, y_pred)
print(cnf_matrix)
# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_names, title='Confusion matrix')
plt.show()
# Part 11: Calculating precision, recall, and F1-score
precision, recall, f1_score, _ = precision_recall_fscore_support(y_test, y_pred, average='macro')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1_score: {f1_score}')

```

7. TESTING

7.1 Introduction:

Software testing is an investigation conducted to provide clients 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 have 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 sssss
- Achieves the general result 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 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 programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

Testing Types:

A software engineering product can be tested in one of two ways:

- Black box testing
- White box testing

Black box testing:

Knowing the specified function that a product has been designed to perform, determine whether each function is fully operational.

White box testing:

Knowing the internal workings of a software product determine whether the internal operation implementing the functions perform according to the specification, and all the internal components have been adequately exercised.

Testing Strategies:

Four Testing Strategies that are often adopted by the software development team include:

- Unit Testing
- Integration Testing
- Validation Testing
- System Testing

Unit Testing:

We adopt white box testing when using this testing technique. This testing was carried out on individual components of the software that were designed. Each individual module was tested using this technique during the coding phase. Every component was checked to make sure that they adhere strictly to the specifications spelt out in the data flow diagram and ensure that they perform the purpose intended for them. All the names of the variables are scrutinized to make sure that they are truly reflected of the element they represent. All the looping mechanisms were verified to ensure that they were as decided. Beside these, we trace through the code manually to capture syntax errors and logical errors.

Integration Testing:

After finishing the Unit Testing process, next is the integration testing process. In this testing process we put our focus on identifying the interfaces between components and their functionality as dictated by the DFD diagram. The Bottom-up incremental approach was adopted during this testing. Low level modules are integrated and combined as a cluster before testing.

The Black box testing technique was employed here. The interfaces between the components were tested first. This allowed identifying any wrong linkages or parameters passing early in the development process as it just can be passed in a set of data and checked if the result returned is an accepted one.

Validation Testing:

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.

System Testing:

System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all the work should verify that all system elements have been properly integrated and perform allocated functions. System testing also ensures that the project works well in the environment. It traps the errors and allows convenient processing of errors without coming out of the program abruptly. Recovery testing is done in such a way that failure is forced to a software system and checked whether the recovery is proper and accurate. The performance of the system is highly effective. Software testing is critical element of software quality assurance and represents ultimate review of specification, design and coding. Test case design focuses on a set of technique for the creation of test cases that meet overall testing objectives. Planning and testing of a programming system involve formulating a set of test cases, which are similar to the real data that the system is intended to manipulate. Test cases consist of input specifications, a description of the system functions exercised by the input and a statement of the extended output. Through testing involves producing cases to ensure that the program responds, as expected, to both valid and invalid inputs, that the program perform to specification and that it does not corrupt other programs or data in the system. In principle, testing of a program must be extensive. Every statement in the

program should be exercised and every possible path combination through the program should be executed at least once. Thus, it is necessary to select a subset of the possible test cases and conjecture that this subset will adequately test the program.

Guidelines for developing test cases:

- Describe which feature or service your test attempts to cover
- If the test case is based on a use case it is a good idea to refer to the use case name.
- 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.
- 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.

- Test the normal use of the object's methods. Test the abnormal but reasonable use of the object's methods.
- 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.
- Test object 's interactions and the messages sent among them. If you have developed sequence diagrams, they can assist you in this process.
- when the revisions have been made, document the cases so they become the starting bases for the follow- up test.

Attempting to reach agreement on answers generally will raise other what-if questions. Add these to the list and answer them, repeat the process until the list is stabilized, then you need not add any more questions.

Functionality Testing:

Functionality testing is performed to verify that a software application performs and functions correctly according to design specifications. During functionality testing we check the core application functions, text input, menu functions and installation and setup on localized machines, etc.

The following is needed to be checked during the functionality testing:

- Installation and setup on localized machines running localized operating systems and local code pages.
- Text input, including the use of extended characters or non-Latin scripts.
- Core application functions.
- String handling, text, and data, especially when interfacing with non-Unicode applications or modules.
- Regional settings defaults.
- Text handling (such as copying, pasting, and editing) of extended characters, special fonts, and non-Latin scripts.
- Accurate hot-key shortcuts without any duplication.

Functionality testing verifies that an application is still fully functional after localization. Even applications which are professionally internationalized according to world-readiness guidelines require functionality testing. Functionality testing ensures applications remain fully operational post-localization, even after professional internationalization.

Performance Testing:

Performance testing plays a pivotal role in assessing the speed and efficiency of various computing systems, networks, software programs, or devices. It encompasses quantitative tests conducted in controlled environments, such as measuring response times or computing speeds in terms of MIPS (millions of instructions per second). Additionally, qualitative attributes like reliability, scalability, and interoperability are also evaluated during performance testing. Often conducted alongside stress testing, performance testing aims to ensure that systems meet the specifications claimed by manufacturers or vendors. It involves comparing multiple devices or programs based on parameters such as speed, data transfer rate, bandwidth, throughput efficiency, or reliability.

Moreover, performance testing serves as a diagnostic tool for identifying communication bottlenecks within systems. By pinpointing areas of inefficiency, such as slow data transfer rates or bandwidth limitations, performance testing enables organizations to optimize system performance and enhance overall efficiency. For instance, diagnosing and resolving a bottleneck in network communication can significantly improve system functionality, even on high-performance computers. Ultimately, performance testing not only validates system specifications but also aids in optimizing system components, leading to improved performance and user experience across various computing environments.

Performance testing evaluates the speed and effectiveness of computing systems, networks, software programs, or devices through quantitative tests measuring factors like response times and computing speeds. It ensures systems meet claimed specifications, bolstering stakeholder confidence. When conducted with stress testing, it identifies and resolves potential issues before impacting real-world performance.

Furthermore, performance testing serves as a valuable diagnostic tool for pinpointing communication bottlenecks within systems. By identifying areas of inefficiency, such as slow data transfer rates or limited bandwidth, organizations can take proactive measures to optimize system performance and enhance overall efficiency. For example, diagnosing and resolving network communication bottlenecks can significantly improve system functionality, even on high-performance computers. Ultimately, performance testing plays a crucial role in validating system specifications, optimizing system components, and ensuring a seamless user experience across diverse computing environments. Performance testing assesses computing systems' speed and effectiveness, ensuring they meet specified criteria and enhancing stakeholder confidence. It also

identifies and resolves potential issues preemptively, optimizing system performance and enhancing efficiency.

7.2 Test Cases:

T ID	Description	Input	Expected value	Actual value	Result
1.	Train CNN Algorithms	Dataset	Accuracy	Accuracy	Pass
2.	Preprocessing the Input Data	Image	Process Image to array	Image processed to array	Pass
3.	Predict cancer	Image	Predicted cancer	Predicted cancer	Pass
4.	Adding new data	Image	Change from old dataset	Changed from old dataset	Pass

8. SAMPLE SCREENSHOTS

```
# Part 1: Importing libraries
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import keras.utils
import os
import pandas as pd
from sklearn.model_selection import train_test_split
%matplotlib inline
```

Fig. 8.1 Importing libraries

```
# Part 2: Importing TensorFlow and ImageDataGenerator
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

Fig. 8.2 Importing TensorFlow and ImageDataGenerator

```
# Part 3: Defining image size and folder path
picture_size = 224
from google.colab import drive
drive.mount('/content/drive')
folder_path = '/content/drive/MyDrive/lung'
```

Fig. 8.3 Defining image size and folder path

```
# Part 4: Setting batch size and creating data generators
batch_size = 128
datagen_train = ImageDataGenerator()
datagen_val = ImageDataGenerator()
datagen_test = ImageDataGenerator()
train_set = datagen_train.flow_from_directory(folder_path+"/Train",
target_size=(picture_size, picture_size),
color_mode="rgb",
batch_size=batch_size,
class_mode='categorical',
shuffle=True)
test_set = datagen_test.flow_from_directory(folder_path+"/Test",
target_size=(picture_size, picture_size),
color_mode="rgb",
batch_size=batch_size,
class_mode='categorical',
shuffle=False)
```

```
val_set = datagen_val.flow_from_directory(folder_path+"/Val",
target_size=(picture_size, picture_size),
color_mode="rgb",
batch_size=batch_size,
class_mode='categorical',
shuffle=False)
```

Fig. 8.4 Visualize the columns

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(224, 224, 3), padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D((2,2)),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D((2,2)),

    tf.keras.layers.Conv2D(256, (3,3), activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D((2,2)),

    tf.keras.layers.Conv2D(512, (3,3), activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D((2,2)),

    tf.keras.layers.Conv2D(512, (3,3), activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D((2,2)),


    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
conv2d_5 (Conv2D)	(None, 224, 224, 64)	1792
batch_normalization_5 (Batch Normalization)	(None, 224, 224, 64)	256
max_pooling2d_5 (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_6 (Conv2D)	(None, 112, 112, 128)	73856
batch_normalization_6 (Batch Normalization)	(None, 112, 112, 128)	512
max_pooling2d_6 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_7 (Conv2D)	(None, 56, 56, 256)	295168
batch_normalization_7 (Batch Normalization)	(None, 56, 56, 256)	1024
max_pooling2d_7 (MaxPooling2D)	(None, 28, 28, 256)	0

conv2d_8 (Conv2D)	(None, 28, 28, 512)	1180160
batch_normalization_8 (Batch Normalization)	(None, 28, 28, 512)	2048
max_pooling2d_8 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_9 (Conv2D)	(None, 14, 14, 512)	2359808
batch_normalization_9 (Batch Normalization)	(None, 14, 14, 512)	2048
max_pooling2d_9 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_2 (Dense)	(None, 512)	12845568
dense_3 (Dense)	(None, 3)	1539
=====		
Total params: 16763779 (63.95 MB)		
Trainable params: 16760835 (63.94 MB)		
Non-trainable params: 2944 (11.50 KB)		

Fig. 8.5 Model summary

 # Part 6: Training the model opt = tf.keras.optimizers.SGD(learning_rate=0.001) model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy']) history = model.fit(train_set, validation_data=val_set, epochs=30) model.save("lung_cancer_model.h5")	
Epoch 1/30	
88/88 [=====]	- 83s 794ms/step - loss: 4.3239 - accuracy: 0.8492 - val_loss: 4.2217 - val_accuracy: 0.3872
Epoch 2/30	
88/88 [=====]	- 65s 739ms/step - loss: 0.1567 - accuracy: 0.9433 - val_loss: 1.3133 - val_accuracy: 0.6576
Epoch 3/30	
88/88 [=====]	- 63s 720ms/step - loss: 0.1119 - accuracy: 0.9588 - val_loss: 0.3392 - val_accuracy: 0.8731
Epoch 4/30	
88/88 [=====]	- 63s 719ms/step - loss: 0.0543 - accuracy: 0.9811 - val_loss: 0.1616 - val_accuracy: 0.9456
Epoch 5/30	
88/88 [=====]	- 63s 711ms/step - loss: 0.0415 - accuracy: 0.9864 - val_loss: 0.1784 - val_accuracy: 0.9552
Epoch 6/30	
88/88 [=====]	- 63s 714ms/step - loss: 0.0364 - accuracy: 0.9880 - val_loss: 0.1086 - val_accuracy: 0.9691
Epoch 7/30	
88/88 [=====]	- 63s 715ms/step - loss: 0.0410 - accuracy: 0.9875 - val_loss: 0.2277 - val_accuracy: 0.9429
Epoch 8/30	
88/88 [=====]	- 62s 705ms/step - loss: 0.0191 - accuracy: 0.9937 - val_loss: 0.1136 - val_accuracy: 0.9701
Epoch 9/30	
88/88 [=====]	- 62s 708ms/step - loss: 0.0071 - accuracy: 0.9987 - val_loss: 0.0821 - val_accuracy: 0.9771
Epoch 10/30	
88/88 [=====]	- 65s 742ms/step - loss: 0.0051 - accuracy: 0.9990 - val_loss: 0.0951 - val_accuracy: 0.9744
Epoch 11/30	
88/88 [=====]	- 64s 724ms/step - loss: 0.0029 - accuracy: 0.9996 - val_loss: 0.1161 - val_accuracy: 0.9728
Epoch 12/30	
88/88 [=====]	- 64s 723ms/step - loss: 0.0050 - accuracy: 0.9986 - val_loss: 0.1074 - val_accuracy: 0.9723
Epoch 13/30	
88/88 [=====]	- 62s 708ms/step - loss: 0.0116 - accuracy: 0.9964 - val_loss: 0.1368 - val_accuracy: 0.9701
Epoch 14/30	
88/88 [=====]	- 63s 714ms/step - loss: 0.0243 - accuracy: 0.9929 - val_loss: 0.1381 - val_accuracy: 0.9605
Epoch 15/30	
88/88 [=====]	- 64s 722ms/step - loss: 0.2277 - accuracy: 0.9622 - val_loss: 0.3630 - val_accuracy: 0.9125
Epoch 16/30	
88/88 [=====]	- 62s 706ms/step - loss: 0.0504 - accuracy: 0.9853 - val_loss: 0.1126 - val_accuracy: 0.9616
Epoch 17/30	
88/88 [=====]	- 63s 718ms/step - loss: 0.0227 - accuracy: 0.9929 - val_loss: 0.2234 - val_accuracy: 0.9440

```

Epoch 18/30
88/88 [=====] - 63s 719ms/step - loss: 0.0076 - accuracy: 0.9972 - val_loss: 0.1029 - val_accuracy: 0.9712
Epoch 19/30
88/88 [=====] - 64s 721ms/step - loss: 0.0024 - accuracy: 0.9996 - val_loss: 0.0854 - val_accuracy: 0.9776
Epoch 20/30
88/88 [=====] - 63s 720ms/step - loss: 0.0010 - accuracy: 0.9999 - val_loss: 0.0818 - val_accuracy: 0.9787
Epoch 21/30
88/88 [=====] - 63s 716ms/step - loss: 5.0149e-04 - accuracy: 1.0000 - val_loss: 0.0792 - val_accuracy: 0.9797
Epoch 22/30
88/88 [=====] - 62s 702ms/step - loss: 5.1159e-04 - accuracy: 1.0000 - val_loss: 0.0794 - val_accuracy: 0.9771
Epoch 23/30
88/88 [=====] - 63s 719ms/step - loss: 5.4280e-04 - accuracy: 0.9999 - val_loss: 0.0906 - val_accuracy: 0.9755
Epoch 24/30
88/88 [=====] - 63s 709ms/step - loss: 3.8138e-04 - accuracy: 1.0000 - val_loss: 0.0838 - val_accuracy: 0.9771
Epoch 25/30
88/88 [=====] - 62s 703ms/step - loss: 1.9626e-04 - accuracy: 1.0000 - val_loss: 0.0840 - val_accuracy: 0.9787
Epoch 26/30
88/88 [=====] - 65s 733ms/step - loss: 1.7793e-04 - accuracy: 1.0000 - val_loss: 0.0841 - val_accuracy: 0.9797
Epoch 27/30
88/88 [=====] - 63s 718ms/step - loss: 1.3571e-04 - accuracy: 1.0000 - val_loss: 0.0812 - val_accuracy: 0.9787
Epoch 28/30
88/88 [=====] - 65s 733ms/step - loss: 1.0528e-04 - accuracy: 1.0000 - val_loss: 0.0826 - val_accuracy: 0.9792
Epoch 29/30
88/88 [=====] - 62s 708ms/step - loss: 1.1044e-04 - accuracy: 1.0000 - val_loss: 0.0834 - val_accuracy: 0.9787
Epoch 30/30
88/88 [=====] - 64s 728ms/step - loss: 1.6986e-04 - accuracy: 1.0000 - val_loss: 0.0866 - val_accuracy: 0.9792

```

Fig. 8.6 Training the model

```

# Part 7: Evaluating on the validation set
loss, accuracy = model.evaluate(val_set)
print('Validation loss:', loss)
print('Validation accuracy:', accuracy)

15/15 [=====] - 9s 564ms/step - loss: 0.0866 - accuracy: 0.9792
Validation loss: 0.08660732954740524
Validation accuracy: 0.979200005531311

```

Fig. 8.7 Evaluating on the validation set

```

# Part 8: Evaluating on the test set
loss, accuracy = model.evaluate(test_set)
print('Test loss:', loss)
print('Test accuracy:', accuracy)

15/15 [=====] - 8s 549ms/step - loss: 0.0551 - accuracy: 0.9861
Test loss: 0.05508797988295555
Test accuracy: 0.986133337020874

```

Fig. 8.8 Evaluating on the test set

```

# Part 9: Importing necessary modules for metrics
import numpy as np
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support

```

Fig. 8.9 Importing necessary modules for metrics

```

# Part 10: Making predictions and plotting confusion matrix
predictions = model.predict(test_set)
y_pred = [np.argmax(probas) for probas in predictions]
y_test = test_set.classes
class_names = test_set.class_indices.keys()
from sklearn.metrics import confusion_matrix, classification_report
import itertools

# Function to plot confusion matrix
def plot_confusion_matrix(cm, classes, title='Confusion matrix', cmap=plt.cm.Reds):
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(7, 5))
    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 = '.0f'
    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[j, i] > thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()

# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, y_pred)
print(cnf_matrix)

# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_names, title='Confusion matrix')
plt.show()

```

15/15 [=====] - 8s 544ms/step

```

[[612  0  13]
 [  0 625  0]
 [ 13  0 612]]

```

<Figure size 640x480 with 0 Axes>

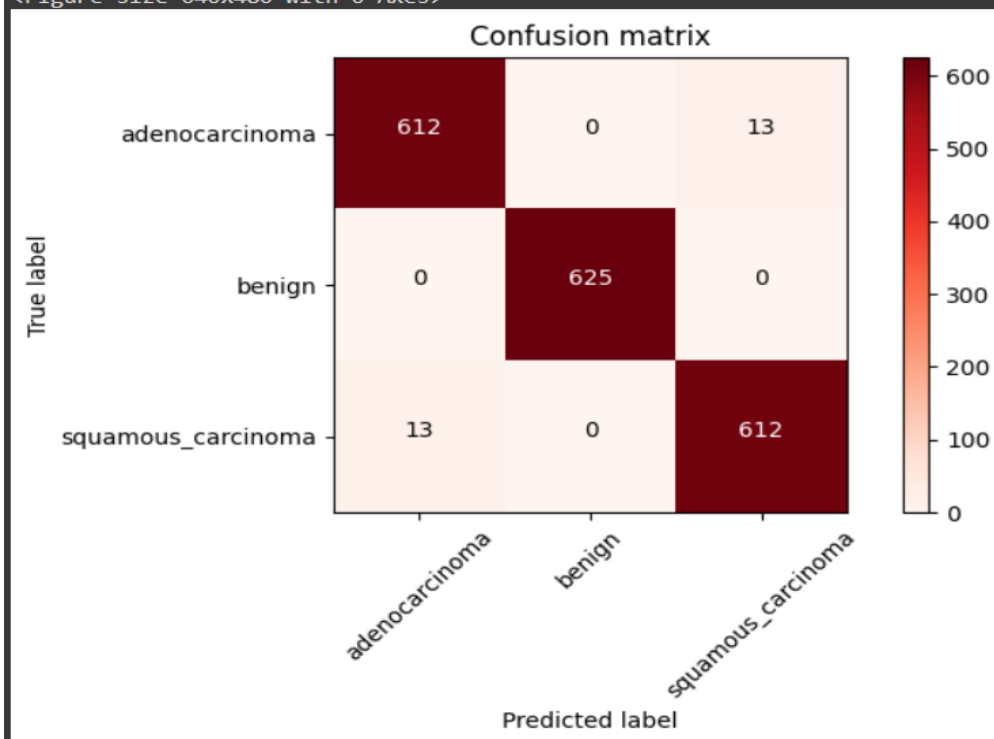


Fig. 8.10 Making predictions and plotting confusion matrix

```
# Part 11: Calculating precision, recall, and F1-score
precision, recall, f1_score, _ = precision_recall_fscore_support(y_test, y_pred, average='macro')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1_score: {f1_score}')
```

```
Precision: 0.9861333333333334
```

```
Recall: 0.9861333333333334
```

```
F1_score: 0.9861333333333334
```

Fig. 8.11 Calculating precision, recall, and F1-score

9. CONCLUSION

In our study, we introduced a custom CNN-based model tailored for the classification of five common rice leaf diseases in Bangladesh. Achieving an impressive accuracy of 97.82% on independent test images, our model demonstrates robust performance across diverse backgrounds and lighting conditions while maintaining efficiency in memory storage.

Moving forward, our research aims to enhance the reliability and robustness of our model on datasets from different regions. We plan to address challenges such as complex backgrounds and varied illumination conditions, leveraging interpretable CNN-based models and data augmentation techniques to optimize performance and contribute to the advancement of rice disease detection methodologies.

In future, Developing a real-time system for non-small cell lung cancer detection from histopathology images promises efficient classification and subtype analysis. This advancement would alleviate the burden on pathologists, offering rapid and accurate results while enabling them to focus on complex cases. With automated classification, timely treatment decisions can be made, potentially improving patient outcomes. The system's real-time capability ensures faster results, facilitating early detection and intervention for better prognosis.

In conclusion, our research endeavors aim to continually improve disease detection methodologies, leveraging advanced CNN-based models and real-time systems. By addressing challenges and enhancing efficiency, we strive to contribute to better patient outcomes and alleviate burdens on healthcare professionals. Through ongoing innovation and technological advancements, we aspire to make significant strides in disease detection and diagnosis. Our research focuses on advancing disease detection using CNN models and real-time systems, aiming to improve patient outcomes and alleviate burdens on healthcare professionals through innovation and efficiency enhancements.

Our project with the Course Outcomes CO1 to CO5 has attained Program Outcomes PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO9, PO10, PO11, PO12 and Program Specific Outcomes PSO1, PSO2, PSO3.

PO	PO	PO	PO	PO	PO	PO	PO	PO	PO	PO	PO	PSO	PSO	PSO
1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
3	3	2	2	3	2	2	3	3	3	3	3	2	3	3

10. BIBLIOGRAPHY

BIBLIOGRAPHY:

- [1] M. Wang and D. Li, "An automatic segmentation method for lung tumor based on improved region growing algorithm", *Diagnostics*, vol. 12, no. 12, pp. 2971, Nov. 2022.
- [2] G. Zhang, Z. Yang, L. Gong, S. Jiang and L. Wang, "Classification of benign and malignant lung nodules from CT images based on hybrid features", *Phys. Med. Biol.*, vol. 64, no. 12, pp. 1-12, 2019.
- [3] J. Zhao, M. Dang, Z. Chen and L. Wan, "DSU-Net: Distraction-sensitive U-Net for 3D lung tumor segmentation", *Eng. Appl. Artif. Intell.*, vol. 109, Mar. 2022.
- [4] T. Meraj, H. T. Rauf, S. Zahoor, A. Hassan, M. I. Lali, L. Ali, et al., "Lung nodules detection using semantic segmentation and classification with optimal features", *Neural Comput. Appl.*, vol. 33, no. 17, pp. 10737-10750, Sep. 2021.
- [5] Z. Ali, A. Irtaza and M. Maqsood, "An efficient U-Net framework for lung nodule detection using densely connected dilated convolutions", *J. Supercomput.*, vol. 78, no. 2, pp. 1602-1623, Feb. 2022.
- [6] Y. Lan, N. Xu, X. Ma and X. Jia, "Segmentation of pulmonary nodules in lung CT images based on active contour model", *Proc. 14th Int. Conf. Intell. Hum.-Mach. Syst. Cybern. (IHMSC)*, pp. 132-135, Aug. 2022.
- [7] S. Dlamini, Y. H. Chen and C. F. Jeffrey Kuo, "Complete fully automatic detection segmentation and 3D reconstruction of tumor volume for non-small cell lung cancer using YOLOv4 and region-based active contour model", *Expert Syst. Appl.*, vol. 212, pp. 1-9, Feb. 2023.
- [8] A. Halder, S. Chatterjee and D. Dey, "Adaptive morphology aided 2-pathway convolutional neural network for lung nodule classification", *Biomed. Signal Process. Control*, vol. 72, Feb. 2022.
- [9] M. Kanipriya, C. Hemalatha, N. Sridevi, S. R. SriVidhya and S. L. J. Shabu, "An improved capuchin search algorithm optimized hybrid CNN-LSTM architecture for malignant lung nodule detection", *Biomed. Signal Process. Control*, vol. 78, pp. 1-39, Sep. 2022.
- [10] J. Yang, B. Wu, L. Li, P. Cao and O. Zaiane, "MSDS-UNet: A multi-scale deeply supervised 3D U-Net for automatic segmentation of lung tumor in CT", *Comput. Med. Imag. Graph.*, vol. 92, pp. 1-10, Sep. 2021.

- [11] J. Park, S. K. Kang, D. Hwang, H. Choi, S. Ha, J. M. Seo, et al., "Automatic lung cancer segmentation in [18F]FDG PET/CT using a two-stage deep learning approach", *Nucl. Med. Mol. Imag.*, vol. 57, no. 2, pp. 86-93, Apr. 2023.
- [12] S. Suresh and S. Mohan, "ROI-based feature learning for efficient true positive prediction using convolutional neural network for lung cancer diagnosis", *Neural Comput. Appl.*, vol. 32, no. 20, pp. 15989-16009, Oct. 2020.
- [13] S. Nageswaran, G. Arunkumar, A. K. Bisht, S. Mewada, J. N. V. R. S. Kumar, M. Jawarneh, et al., "Lung cancer classification and prediction using machine learning and image processing", *BioMed Res. Int.*, vol. 2022, pp. 1-8, Aug. 2022.
- [14] D. Zhao, Y. Liu, H. Yin and Z. Wang, "An attentive and adaptive 3D CNN for automatic pulmonary nodule detection in CT image", *Expert Syst. Appl.*, vol. 211, no. 1, pp. 1-24, 2023.
- [15] N. Bhaskar and T. S. Ganashree, "Pulmonary nodule detection using Laplacian of Gaussian and deep convolutional neural network", *Smart Innov. Syst. Technol.*, vol. 282, pp. 633-648, Jan. 2022.
- [16] S. Dodia, B. Annappa and M. A. Padukudru, "A novel artificial intelligence-based lung nodule segmentation and classification system on CT scans", *Commun. Comput. Inf. Sci.*, vol. 1568, pp. 552-564, Jan. 2022.