

An Industrial Oriented Major Project Report on
AI-BASED TUBERCULOSIS CLASSIFICATION IN CHEST X-RAYS

Submitted in Partial fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

BEETOJU MALLESH	20BD1A050E
GINUGA VIKAS REDDY	20BD1A050P
MERADAKONDA ARUN SAI	20BD1A0516
MURIKI AKASH	20BD1A051A

Under the guidance of

Mr. D.Champla, M.Tech(PhD)
Assistant Professor, Department of CSE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH.

Narayanaguda, Hyderabad, Telangana-29

2023-24



KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH.

Narayanaguda, Hyderabad, Telangana-29



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that this is a bonafide record of the project report titled **“AI-BASED TUBERCULOSIS CLASSIFICATION IN CHEST X-RAYS”** which is being presented as the Industrial Oriented Major Project report by

1. BEETOJU MALLESH

20BD1A050E

2. GINUGA VIKAS REDDY

20BD1A050P

3. MERADAKONDA ARUN SAI

20BD1A0516

4. MURIKI AKASH

20BD1A051A

In partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering affiliated to the Jawaharlal Nehru Technological University Hyderabad, Hyderabad

Faculty Supervisor
(Mr. D. Champla)

Head of Department
(Mr. P. Upendar)

Submitted for Viva Voce Examination held on

External Examiner

Vision & Mission of KMIT

Vision of KMIT

- To be the fountainhead in producing highly skilled, globally competent engineers.
- Producing quality graduates trained in the latest software technologies and related tools and striving to make India a world leader in software products and services.

Mission of KMIT

- To provide a learning environment that inculcates problem solving skills, professional, ethical responsibilities, lifelong learning through multi modal platforms and prepares students to become successful professionals.
- To establish an industry institute Interaction to make students ready for the industry.
- To provide exposure to students on the latest hardware and software tools.
- To promote research-based projects/activities in the emerging areas of technology convergence.
- To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
- To induce a spirit of nationalism which will enable the student to develop, understand India's challenges and to encourage them to develop effective solutions.
- To support the faculty to accelerate their learning curve to deliver excellent service to students.

Vision & Mission of CSE

Vision of the CSE

To be among the region's premier teaching and research Computer Science and Engineering departments producing globally competent and socially responsible graduates in the most conducive academic environment.

Mission of the CSE

- To provide faculty with state of the art facilities for continuous professional development and research, both in foundational aspects and of relevance to emerging computing trends.
- To impart skills that transform students to develop technical solutions for societal needs and inculcate entrepreneurial talents.
- To inculcate an ability in students to pursue the advancement of knowledge in various specializations of Computer Science and Engineering and make them industry-ready.
- To engage in collaborative research with academia and industry and generate adequate resources for research activities for seamless transfer of knowledge resulting in sponsored projects and consultancy.
- To cultivate responsibility through sharing of knowledge and innovative computing solutions that benefit the society-at-large.
- To collaborate with academia, industry and community to set high standards in academic excellence and in fulfilling societal responsibilities

PROGRAM OUTCOMES (POs)

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
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.
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.
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.
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.
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.
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.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
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.
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.
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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: An ability to analyze the common business functions to design and develop appropriate Information Technology solutions for social upliftment's.

PSO2: Shall have expertise on the evolving technologies like Python, Machine Learning, Deep learning, IOT, Data Science, Full stack development, Social Networks, Cyber Security, Mobile Apps, CRM, ERP, Big Data, etc.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

- To imbibe analytical and professional skills for successful careers and create enthusiasts to pursue advance education supplementing their career growth.
- Graduates will solve real time problems design, develop and implement innovative ideas by applying their computer engineering principles.
- Graduates will develop necessary skillset for industry by imparting state of art technology in various areas of computer science engineering.
- Graduates will engage in lifelong learning and be able to work collaboratively exhibiting high level of professionalism.

PROJECT OUTCOMES

P1: Accurately detect Tuberculosis and draw bounding boxes accordingly.

P2: Acquire skills in React and Fast API for Application development.

P3: Develop proficiency in Deep Learning for detection model.

P4: Master real-time data integration from FAST API with frontend.

P5: Learn data-driven decision-making by interpreting results and user's X-ray.

MAPPING PROJECT OUTCOMES WITH PROGRAM OUTCOMES

PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
P1	H	H	H	M	L	M	H	H			M	
P2	L	M	M		M			M		L		
P3	M	L	M	M	H			H				
P4	L	L	M	M	M			H			L	
P5	M	M			L	H	L			L	M	

L – LOW

M –MEDIUM

H– HIGH

PROJECT OUTCOMES MAPPING WITH PROGRAM SPECIFIC OUTCOMES

PSO	PSO1	PSO2
P1	H	H
P2	M	M
P3		M
P4	H	M
P5	H	

PROJECT OUTCOMES MAPPING WITH PROGRAM EDUCATIONAL OBJECTIVES

PEO	PEO1	PEO2	PEO3	PEO4
P1	H	H	H	H
P2	M	M	M	M
P3		M		M
P4	H	M	H	M
P5	H	H	H	H

DECLARATION

We hereby declare that the results embodied in the dissertation entitled **“AI-BASED TUBERCULOSIS CLASSIFICATION IN CHEST X-RAYS”** has been carried out by us together during the academic year 2023-24 as a partial fulfillment of the award of the B.Tech degree in Computer Science and Engineering from JNTUH. We have not submitted this report to any other university or organization for the award of any other degree.

Student Name

Rollno.

BEETOJU MALLESH

20BD1A050E

GINUGA VIKAS REDDY

20BD1A050P

MERADAKONDA ARUN SAI

20BD1A0516

MURIKI AKASH

20BD1A051A

ACKNOWLEDGEMENT

We take this opportunity to thank all the people who have rendered their full support to our project work. We render our thanks to **Dr. B L Malleswari**, Principal who encouraged us to do the Project.

We are grateful to **Mr. Neil Gogte**, Founder & Director, **Mr. S. Nitin**, Director for facilitating all the amenities required for carrying out this project.

We express our sincere gratitude to **Ms. Deepa Ganu**, Director Academic for providing an excellent environment in the college.

We are also thankful to **Mr. P Upendar**, Head of the Department for providing us with time to make this project a success within the given schedule.

We are also thankful to our Faculty Supervisor **Mr. D. Champla**, for his valuable guidance and encouragement given to us throughout the project work.

We would like to thank the entire CSE Department faculty, who helped us directly and indirectly in the completion of the project.

We sincerely thank our friends and family for their constant motivation during the project work.

Student Name

Rollno.

BEETOJU MALLESH

20BD1A050E

GINUGA VIKAS REDDY

20BD1A050P

MERADAKONDA ARUN SAI

20BD1A0516

MURIKI AKASH

20BD1A051A

ABSTRACT

Tuberculosis remains a major health threat, demanding accurate and timely diagnosis for effective patient care and the containment of its spread. Leveraging the power of AI, we have devised a robust solution that streamlines the classification of TB in chest X-ray images, offering healthcare professionals an invaluable tool to enhance their diagnostic capabilities.

The project encompasses a multifaceted approach. We commence with a rigorous data collection phase, amassing a diverse and meticulously annotated dataset of chest X-ray images, including both TB-positive and TB-negative cases. Preprocessing techniques are employed to ensure data quality and suitability for AI model training. Subsequently, we delve into deep learning model development, employing state-of-the-art neural network architectures to extract intricate features from chest X-ray images. Our AI model undergoes intensive training and fine-tuning to accurately classify images as TB-positive or TB-negative.

To make this technology accessible to healthcare practitioners, we have developed a user-friendly web-based frontend using React.js. This frontend empowers users to easily upload and view chest X-ray images, which are then processed by our Python-based backend. The backend integrates the AI model, ensuring a seamless flow of data and enabling real-time inference.

In conclusion, our project endeavors to significantly impact TB diagnosis by combining the sophistication of AI with modern web technologies. Through automation, we seek to mitigate diagnostic errors, expedite the diagnostic process, and ultimately contribute to improved patient outcomes, especially in healthcare settings with limited resources.

LIST OF FIGURES

S.No	Name of Screenshot	PageNo.
1.1	Tuberculosis	2
1.2	TB Spots	5
1.3	Manual Process	6
1.4	Architecture Diagram	11
2.1	Existing System	13
3.1	Proposed System	19
4.1	PyCharm	27
4.2	Python	28
4.3	NumPy	28
4.4	TensorFlow	29
5.5	Matplotlib	30
5.1	Visualization of an anchor box	34
5.2	The detection head of YOLOv5	35
5.3	The detection head of YOLOv8	36
5.4	New YOLOv8 C2f module	37
5.5	Mosaic augmentation of chess board photos	38

5.6	YOLOv8 COCO evaluation	39
5.7	Comparison	40
5.8	Dataset	41
5.9	Training Dataset	42
5.10	Installing Library	44
5.11	Configuration	45
5.12	Training Phase	45
5.13	Displaying Confusion matrix	45
5.14	Confusion Matrix	46
5.15	Prediction	47
5.16	Imports	47
5.17	Detection	48
5.18	Modules	48
5.19	Middleware	49
5.20	Saved Model	49
5.21	tbddetection	50
5.22	React	52
5.23	App	54

5.24	Navbar	55
5.25	Routes	56
5.26	Home Page	56
5.27	Contact	57
5.28	Home	62
5.29	Get start	62
5.30	Positive	63
5.31	Detection Box	63
5.32	Enlarged X-rays	64
5.33	Negative	64
5.34	Contact	65
5.35	Log	65
5.36	No file Error	66
5.37	Processing Image Error	66
6.1.1	Use Case Diagram	69
6.1.2	Sequence Diagram	71
6.1.3	Activity Diagram	72
6.1.4	Class Diagram	73

CONTENTS

<u>DESCRIPTION</u>	<u>PAGE</u>
1. Introduction	1-11
1.1 Purpose of the project	4
1.2 Challenges with current systems	6
1.3 Scope of the Project	9
1.4 Architecture Diagram	11
2. Existing System	12-17
2.1 Literature review	15
3. Proposed System	18-20
4. System Requirement Specifications	21-30
4.1 Introduction to SRS	22
4.2 Role of SRS	22
4.3 Requirements Specification Document	25
4.4 Functional Requirements	25
4.5 Non-Functional Requirements	25
4.6 Performance Requirements	26
4.7 Software Requirements	27
4.8 Hardware Requirements	30

<u>DESCRIPTION</u>	<u>PAGE</u>
5. Implementation	31-66
5.1 Application	32
5.2 Testing	58
5.3 Screenshots	62
6. System Design	67-67
6.1 UML Diagrams	68
7. Conclusion	74-76
Future Enhancements	77
References	79

CHAPTER-1

1. INTRODUCTION

Tuberculosis (TB) is a highly contagious bacterial infection caused by *Mycobacterium tuberculosis*. This disease primarily affects the lungs but can also target other organs in the body. The bacteria spread through the air when an infected person expels respiratory droplets through coughing, sneezing, or talking. Upon inhalation by others, the bacteria can lodge in the lungs and initiate an infection.

The symptoms of TB can be insidious and may include a persistent cough, chest pain, unintended weight loss, fatigue, fever, night sweats, and, in advanced cases, coughing up blood. However, some individuals infected with TB may remain asymptomatic, making early detection and treatment challenging.

Diagnosing TB involves a combination of methods. Medical professionals typically consider the patient's medical history, perform a physical examination, and may use imaging techniques such as chest X-rays. Laboratory tests, including sputum tests to identify the presence of the bacteria, are crucial for confirming the diagnosis.



Fig 1.1. Tuberculosis

The cornerstone of TB treatment is a course of antibiotics. Patients are often prescribed a combination of drugs, and completing the entire treatment regimen is crucial to ensure the complete eradication of the bacteria and reduce the risk of developing drug-resistant strains. Drug-resistant TB is a growing concern globally, posing additional challenges to effective treatment. Preventing TB involves a multi-faceted approach. Vaccination, notably with the

Bacille Calmette-Guérin (BCG) vaccine, can offer some protection. Identifying and treating active cases promptly is essential, as is implementing infection control measures to limit the spread of the bacteria in healthcare and community settings.

Certain populations are at higher risk of TB, including those with compromised immune systems, such as individuals with HIV/AIDS, malnutrition, or certain medical conditions. Additionally, close and prolonged contact with an infected person increases the risk of transmission.

Globally, TB remains a significant public health challenge. In 2020, the World Health Organization reported approximately 10 million new cases of TB worldwide. The disease is particularly prevalent in regions with limited access to healthcare resources.

The relationship between TB and HIV is notable, as HIV weakens the immune system, making individuals more susceptible to TB infection. TB is a leading cause of death among people with HIV, emphasizing the importance of integrated strategies to address both diseases.

Public health measures play a critical role in controlling TB. Early detection, contact tracing, and the implementation of infection control practices in healthcare settings are vital components of comprehensive efforts to reduce the incidence and impact of TB on a global scale. While TB is a formidable health challenge, it is a treatable and curable disease, and ongoing research and public health initiatives continue to contribute to the fight against this infectious illness.

1.1 PURPOSE OF THE PROJECT

The purpose of the project is to leverage artificial intelligence (AI) and machine learning (ML) technologies to develop a reliable system for the early detection and classification of tuberculosis (TB) from chest X-ray images.

This initiative serves several important purposes:

Early Detection:

Early identification is crucial in managing tuberculosis effectively. Chest X-ray images are often used in TB diagnosis, and leveraging AI can automate the analysis, leading to faster detection.

Early detection allows for timely intervention and treatment, which is essential in preventing the progression of the disease and reducing its impact on patients.

Improved Accuracy:

Machine learning models excel at pattern recognition, which can enhance the accuracy of TB diagnosis.

By complementing the expertise of healthcare professionals, the system aims to reduce the occurrence of false positives and false negatives, improving the overall reliability of TB screening.

Efficient Screening:

Traditional TB diagnostic methods can be resource-intensive and time-consuming, especially in regions with high prevalence.

An AI-based screening tool can provide a more efficient and scalable solution, enabling the screening of a large number of individuals in a shorter time frame.

Prevention of Spread:

Early identification and isolation of TB cases contribute to preventing the spread of the disease.

By isolating affected individuals early in the disease progression, the project aims to reduce transmission rates and control outbreaks, particularly in densely populated areas.

Resource Optimization:

Automating the initial screening process helps optimize the allocation of healthcare resources. Healthcare professionals can focus on confirmatory diagnostics and treatment planning, streamlining the overall healthcare workflow and potentially improving resource efficiency.

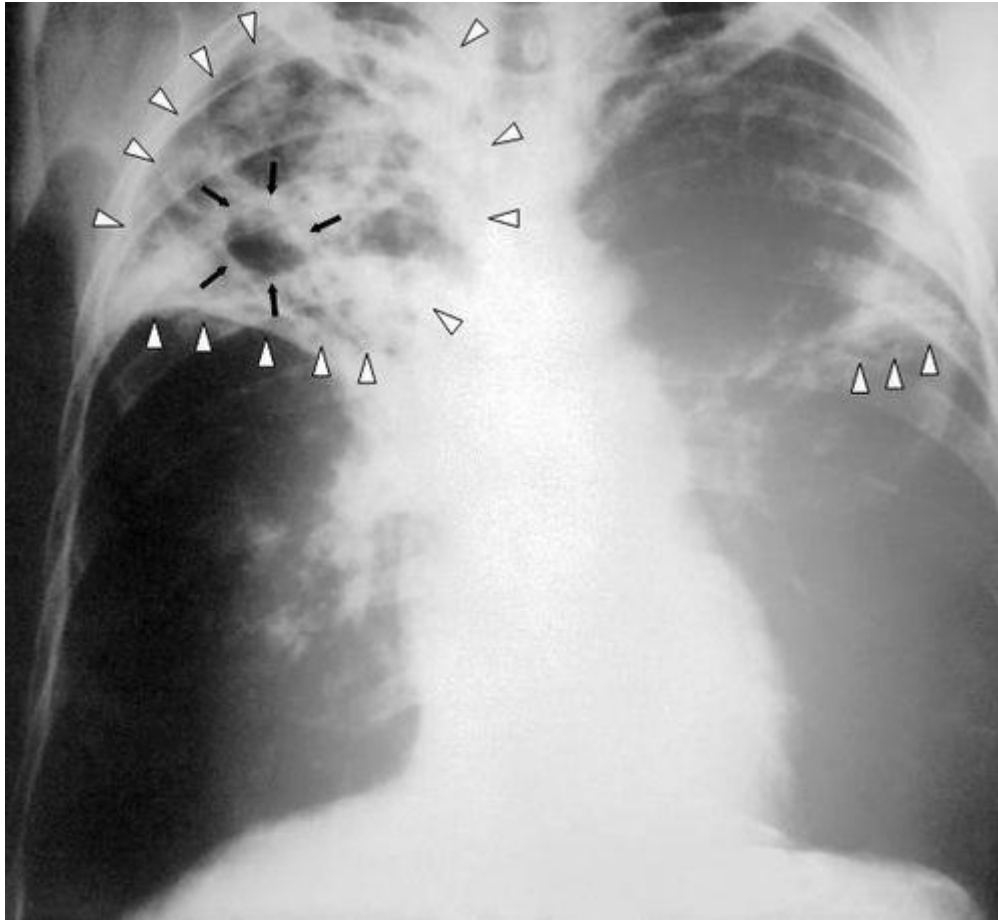


Fig 1.2. TB Spots

Technological Innovation:

The project showcases the application of cutting-edge AI and machine learning technologies in medical imaging and diagnosis.

It demonstrates how innovative solutions can be employed to address significant global health challenges and encourages the adoption of advanced technologies in the healthcare sector.

Global Health Impact:

By addressing a prevalent infectious disease like TB, the project contributes to global health initiatives.

Advances in early detection can have a substantial impact on reducing the burden of TB worldwide, aligning with broader efforts to improve public health on a global scale.

Research and Development:

The project encourages ongoing research in the intersection of AI and healthcare.

Insights and methodologies developed during the project may have applications beyond TB, fostering continuous advancements in diagnostic capabilities for various medical conditions.

Accessibility and Affordability:

The goal is to create a diagnostic tool that is accessible and potentially cost-effective.

This emphasis on affordability makes the technology valuable in diverse healthcare settings, including those with limited resources, thus promoting inclusivity in healthcare.

Patient-Centric Care:

Ultimately, the project aims to improve patient outcomes by facilitating quicker diagnoses and enabling timely interventions.

By supporting healthcare professionals in delivering more effective and patient-centric care, the project contributes to enhancing the overall quality of healthcare services related to TB detection and management.

1.2 CHALLENGES WITH CURRENT SYSTEMS

The existing systems for tuberculosis (TB) detection, particularly those relying solely on traditional methods without the integration of artificial intelligence (AI) and machine learning (ML), face several challenges:

Manual Analysis and Diagnosis:

Challenge:

Traditional methods rely heavily on manual analysis by radiologists and healthcare professionals.

Issue:

This approach is time-consuming, labor-intensive, and can be subject to human error. In regions with a high prevalence of TB, the demand for manual analysis may outstrip available resources.



Fig 1.3. Manual Process

Limited Scalability:**Challenge:**

Current diagnostic methods may lack scalability, especially in densely populated areas or regions with a high incidence of TB.

Issue:

The inability to efficiently screen a large number of individuals hampers early detection efforts, potentially allowing the disease to spread before interventions can take place.

Subjectivity in Interpretation:**Challenge:**

Interpretation of chest X-ray images can be subjective, leading to variations in diagnoses among different healthcare professionals.

Issue:

Inconsistencies in interpretation can result in both false positives and false negatives, impacting the reliability of TB screening and diagnosis.

Dependency on Expertise:**Challenge:**

Accurate TB diagnosis often relies on the expertise of trained radiologists.

Issue:

In regions where there is a shortage of skilled healthcare professionals, the dependency on expertise can hinder the timely and widespread detection of TB cases.

Resource Intensiveness:**Challenge:**

Traditional diagnostic methods, such as sputum microscopy and culture, can be resource-intensive.

Issue:

This resource burden may limit the ability to conduct widespread screening, particularly in resource-constrained settings, delaying the identification and isolation of TB cases.

Delay in Results:**Challenge:**

The time required for obtaining diagnostic results through conventional methods can be significant.

Issue:

Delays in obtaining results may lead to delayed treatment initiation, allowing TB to progress and increasing the risk of transmission to others.

Cost Barriers:**Challenge:**

Access to sophisticated diagnostic equipment and expertise may be limited in some regions due to cost constraints.

Issue:

This limitation can result in disparities in healthcare access, with certain populations facing barriers to early detection and treatment.

Lack of Automation:**Challenge:**

Many existing systems lack automation in the analysis of chest X-ray images.

Issue:

The absence of automated tools can slow down the diagnostic process and may not leverage the full potential of AI and machine learning for accurate and timely detection.

Incomplete Coverage:**Challenge:**

Conventional methods may not cover all at-risk populations comprehensively.

Issue:

This can lead to gaps in the detection and monitoring of TB cases, especially in marginalized or remote communities, allowing the disease to persist and spread.

Resistance to Technological Adoption:**Challenge:**

Some healthcare systems may be resistant to adopting new technologies like AI and machine learning.

Issue:

This resistance can impede the integration of more advanced and efficient diagnostic tools, limiting progress in the field of TB detection and classification.

1.3 SCOPE OF THE PROJECT

Bounding Box Generation:

1. Automated Annotation:

- The system will employ machine learning algorithms for automated annotation of chest X-ray images. These algorithms will be trained on a diverse dataset, learning to identify and outline regions indicative of TB abnormalities.

2. Multi-Class Annotation:

- Bounding boxes will be applied to distinguish between various types and stages of TB. This includes annotating different manifestations such as nodules, cavities, and infiltrates. The multi-class annotation provides a granular understanding of TB-related abnormalities.

Model Training and Validation:

1. Training Dataset Creation:

- The bounding box annotations will contribute to the creation of a comprehensive training dataset. This dataset will include a wide array of TB cases, ensuring that the AI model becomes proficient in recognizing subtle and diverse patterns associated with the disease.

2. Validation and Iterative Improvement:

- During the development phase, the bounding box annotations will be used for model validation. Continuous improvement mechanisms will involve iterative training cycles, incorporating feedback from healthcare professionals and refining the model based on new data and evolving diagnostic standards.

Interpretability and User Interface:

1. Transparent Visualization:

- The bounding boxes will be visualized transparently over chest X-ray images to maintain clarity and transparency. This ensures that healthcare professionals can easily discern between normal and abnormal regions, fostering trust in the AI system's output.

2. User-Adjustable Thresholds:

- The system will allow users to adjust the sensitivity of bounding box annotations. Healthcare professionals can fine-tune the system to be more or less conservative in highlighting potential TB abnormalities, depending on their diagnostic preferences and the clinical context.

Integration with Healthcare Systems:

1. Seamless EHR Integration:

- Bounding box information will seamlessly integrate into electronic health records (EHRs). This ensures that annotated images, along with associated metadata, become part of the patient's comprehensive medical record, supporting longitudinal tracking and historical reference.

2. PACS Compatibility:

- The bounding box data will be compatible with picture archiving and communication systems (PACS), facilitating easy retrieval and collaboration between radiologists and other healthcare professionals involved in the diagnostic workflow.

Ethical Considerations and Data Privacy:

1. Secure Annotation Handling:

- Bounding box annotations will be treated with the same level of confidentiality as the original medical images. Secure protocols will be implemented to protect patient privacy and comply with data protection regulations.

2. Explanatory Documentation:

- The system will include explanatory documentation for healthcare professionals, detailing how bounding boxes are generated, their significance, and the limitations associated with their interpretation. This fosters transparency and ethical usage of AI in healthcare.

Continuous Monitoring and Feedback Loop:

1. Feedback Integration:

- The bounding box data will be part of the feedback loop with healthcare professionals. Insights gained from the practical use of the system, including the accuracy of bounding box annotations, will inform ongoing model improvements.

2. Adaptation to Emerging Standards:

- Bounding box annotations will be adaptable to emerging diagnostic standards and guidelines. The system will be designed to accommodate updates based on advancements in both TB research and medical imaging technologies.

By integrating bounding boxes in these ways, the AI-based tuberculosis detection system aims to provide a comprehensive and user-friendly solution that enhances the diagnostic capabilities of healthcare professionals while maintaining a focus on patient privacy and ethical considerations.

1.4 ARCHITECTURE DIAGRAM

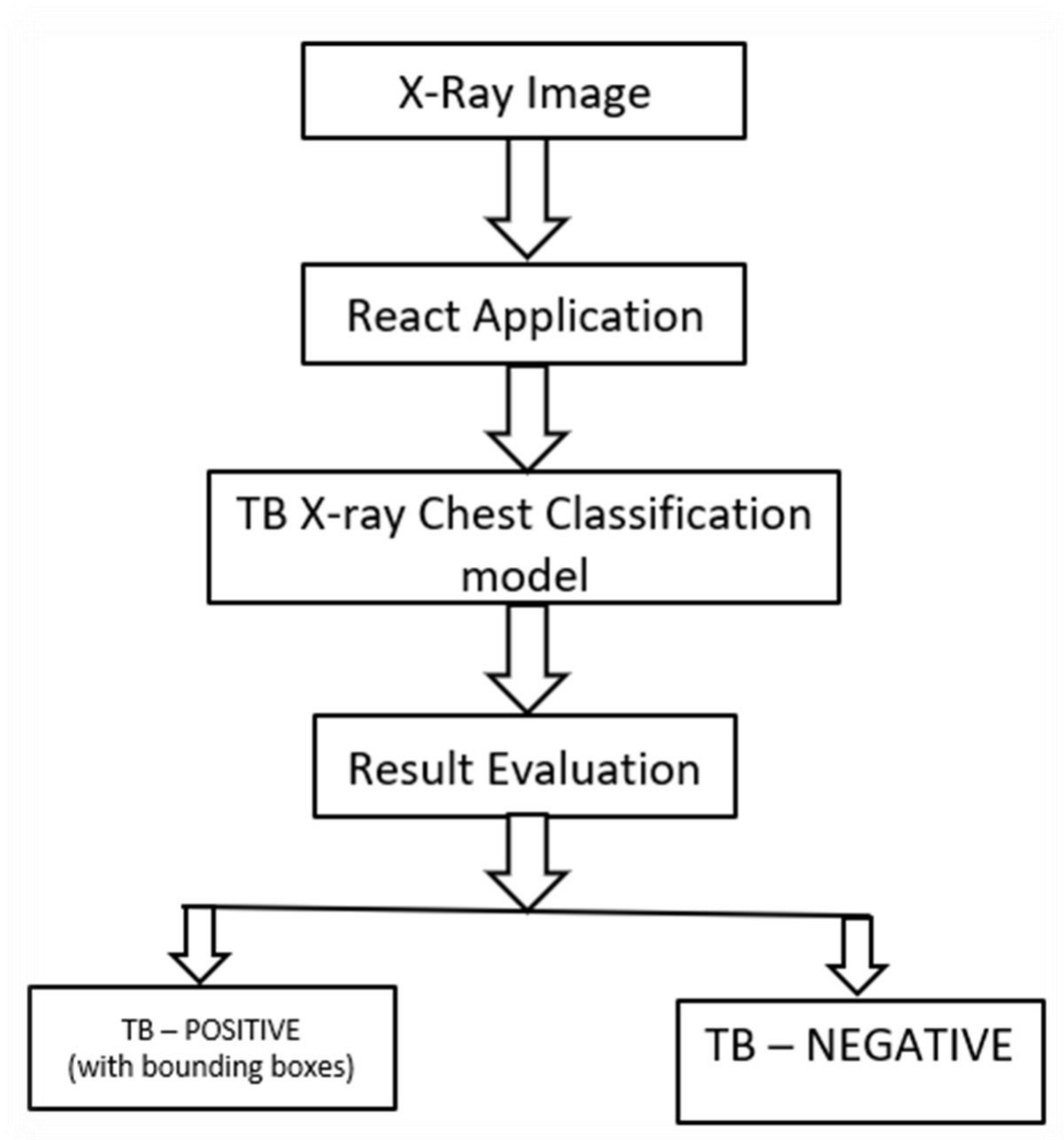


Fig 1.4. Architecture Diagram

CHAPTER-2

2. EXISTING SYSTEM

Manual evaluation of chest X-ray images for tuberculosis (TB) detection involves a series of challenges and considerations that affect its efficacy and efficiency:

- 1. Subjectivity and Variability:** Visual inspection of chest X-rays is inherently subjective and can vary significantly among different healthcare providers. Factors such as experience, expertise, and cognitive biases can influence the interpretation of radiographic findings. This subjectivity introduces inconsistency in diagnosis, leading to variability in patient outcomes.
- 2. Complexity of TB Manifestations:** TB can present with diverse radiographic manifestations, including but not limited to abnormal lung patterns, infiltrates, nodules, cavitations, and pleural effusions. Interpreting these subtle abnormalities accurately requires a high level of proficiency and experience, making it challenging for less-experienced radiologists or healthcare practitioners to detect TB reliably.
- 3. Time-Consumption:** Manual evaluation of chest X-rays is a labor-intensive process that consumes valuable time and resources. Healthcare facilities often experience backlogs in radiology departments due to the sheer volume of X-rays needing interpretation. Delays in diagnosis and treatment initiation can occur, jeopardizing patient care and potentially contributing to disease transmission within communities.

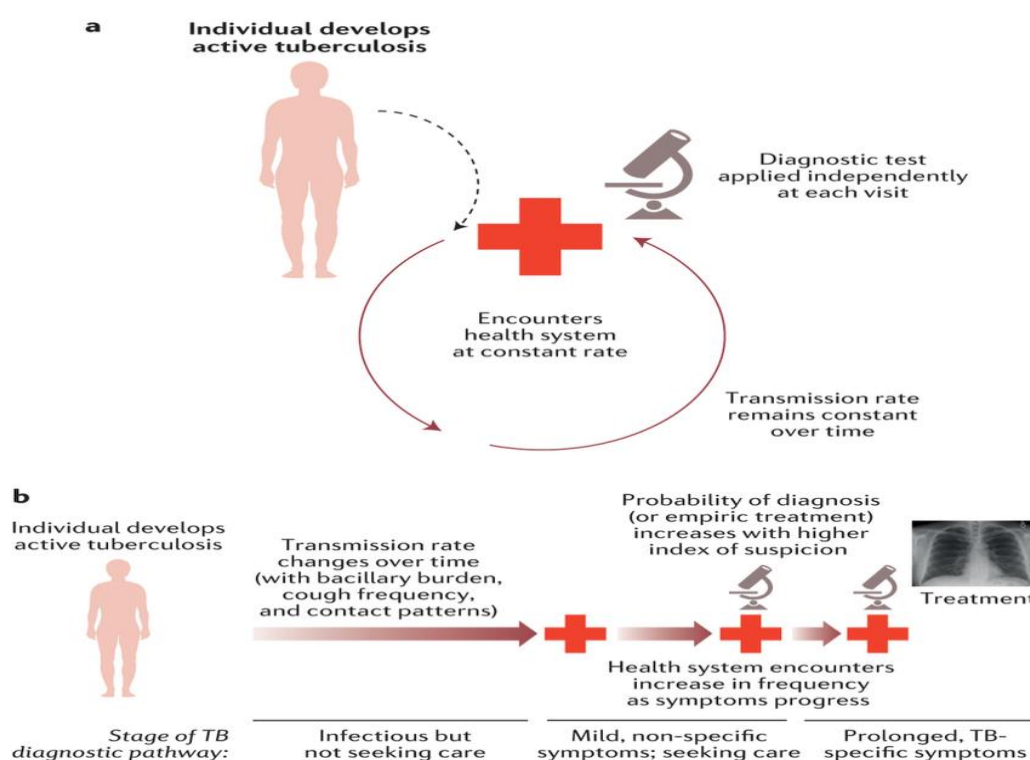


Fig 2.1. Existing System

3. Training and Expertise Requirements: Training radiologists and healthcare professionals to interpret chest X-rays for TB detection requires significant investment in education and skill development. Achieving proficiency in radiographic interpretation demands years of specialized training and clinical experience. Continuous education and refresher courses are necessary to keep practitioners updated on evolving diagnostic criteria and imaging techniques.

5. Risk of False Positives and Negatives: Human-based interpretations of chest X-rays are prone to errors, resulting in false positives or false negatives. False positives may lead to unnecessary investigations, treatments, and psychological distress for patients, while false negatives can delay appropriate management, allowing the disease to progress unchecked and potentially facilitating its spread. These diagnostic inaccuracies can have significant clinical and public health implications, underscoring the need for more reliable and objective diagnostic methods.

6. Quality of Imaging Equipment and Techniques: The quality of chest X-ray images can vary depending on factors such as equipment calibration, positioning of the patient, exposure settings, and image processing techniques. Poor image quality can obscure pathological findings, making it difficult to detect TB accurately. Standardizing imaging protocols and ensuring the availability of high-quality equipment are essential to enhance the reliability of radiographic interpretations.

7. Integration of Technology: While traditional manual evaluation remains the primary method for TB diagnosis through chest X-rays, there's growing interest in integrating technology-driven solutions such as computer-aided detection (CAD) systems and artificial intelligence (AI) algorithms. These tools aim to assist radiologists by providing automated image analysis, reducing interpretation time, and potentially improving diagnostic accuracy. However, the adoption of such technologies requires careful validation, integration into existing workflows, and ongoing evaluation to ensure effectiveness and reliability in real-world clinical settings.

2.1 LITERATURE REVIEW

A number of papers have been published which diagnose TB using deep learning. However, the novelty lies in a variety of factors from data augmentation and regularization to the use of classification network. Chest Visualization is also discussed in this research which will help the medical personnel. Following are our vital contributions to this project.

1. Deep learning for tuberculosis detection from chest X-rays: a comparative study

Year of Publication: 2020 by Computers in Biology and Medicine

The paper explores the usefulness of transfer learning on medical imaging for tuberculosis detection. The authors show an improved method for transfer learning over the regular method of using ImageNet weights. They also discover that the low-level features from ImageNet weights are not useful for imaging tasks for modalities like X-rays and propose a new method for obtaining low-level features by training the models in a multiclass multilabel scenario. This results in an improved performance in the classification of tuberculosis as opposed to training from a randomly initialized setting. In other words, the authors have proposed a better way for training in a data-constrained setting such as the healthcare sector.

The proposed method is useful for detecting and classifying tuberculosis from chest X-ray images. The proposed method can be used in data-constrained settings such as the healthcare sector. The proposed method can be used for both binary and multiclass classification. The proposed method is useful for accurate diagnostics of tuberculosis.

The proposed method is useful for detecting tuberculosis in chest X-ray images. The proposed method is useful for classifying tuberculosis in chest X-ray images. The proposed method is useful for highlighting the regions in the chest X-rays for detailed inference on the diagnosis.

The authors have used a dataset of 662 chest X-ray images for training and testing the model. The proposed model has achieved an accuracy of 96.91% and an AUC of 99.38% on a multiclass classification dataset. The proposed model has achieved a sensitivity of 91.81% and a specificity of 98.42% on a multiclass classification dataset. The authors have used the Score-Cam algorithm to highlight the regions in the chest X-rays for detailed inference on the diagnosis. The paper includes a detailed description of the architecture of the proposed model. The proposed model is engineered to provide accurate diagnostics for both binary and multiclass classification. The authors have shown that the proposed method outperforms the traditional method of using ImageNet weights.

2. Tuberculosis detection on chest X-rays using CNN and traditional machine learning classifiers

Year of Publication: 2019 by Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

The paper presents a framework that helps reduce a radiologist's time in examining a chest X-ray (CXR) by pre-computing some of the more mechanical aspects of the examination while highlighting some of the features, for example, the view position, patient's gender, and critical conditions are observable, using deep-learning (DL), neural networks.

The framework helps radiologists to focus their time and attention more towards the critical areas of the examination rather than the more mechanical aspects. It also reduces the possibility of missing important anomalies in CXRs.

The authors integrated a set of eight Convolutional Neural Networks (CNN) into their framework that operates in a hierarchical order to collaboratively predict the features and metadata of a CXR.

The framework focuses on training lighter convolutional models to predict a single feature each rather than training deep Neural Networks to predict all the features at once, this makes each model lighter and easier to train.

The paper claims that the algorithms developed using deep convolutional neural networks have proven to be quite good at making diagnostic predictions better than the radiologists themselves.

3. A mobile-based deep learning model for automated detection of tuberculosis using chest X-ray images

Year of Publication: 2019 by Computers in Biology and Medicine.

A deep learning-based approach for automatically detecting tuberculosis manifestation from chest X-ray images, the authors used multiple convolutional neural networks to classify chest X-rays into two categories: tuberculosis presence and tuberculosis absence. The dataset used to train the model contained 678 images, having 340 normal chest X-rays and 338 chest X-rays with tuberculosis manifestation. The validation dataset contained 235 images, which observed a sensitivity of 84.91% and a specificity of 93.02%.

The proposed model can classify chest X-rays in real-time. It can help in early detection of tuberculosis and reduce the need for trained personnel for early diagnosis of tuberculosis. The model can be used to diagnose active tuberculosis and distinguish tuberculosis manifestation from chest X-ray images. It can also be used to screen patients at primary health care centers and through mobile X-ray units. The model can diagnose multidrug-resistant tuberculosis and detect tuberculosis in real-time. It can detect tuberculosis in a cost-effective manner.

4. Tuberculosis detection in chest radiographs using a deep convolutional neural network

Year of Publication: 2017 by Neurocomputing.

The paper proposes an automatic tuberculosis detection system using advanced deep learning models. The proposed system uses sophisticated segmentation networks to extract the region of interest from multimedia chest X-rays. The segmented images are fed into the deep learning models. The study uses different convolutional neural network models in their experiments and compare their classification performance using three publicly available chest X-ray datasets. EfficientNetB3, one of the CNN models, achieves the highest accuracy of 99.1%, with a receiver operating characteristic of 99.9%, and an average accuracy of 98.7%. The experiment results confirm that using segmented lung chest X-ray images produces better performance than does using raw lung chest X-ray images. The paper highlights that tuberculosis is classified as a malignant infectious disease that can be fatal. The paper emphasizes that using advanced tools and technology, automatic analysis and classification of chest X-rays into TB and non-TB can be a reliable alternative to the subjective assessment performed by healthcare professionals.

The paper mentions that the respiratory system is considered one of the major systems of the human body. Humans can breathe using the respiratory system to exchange carbon dioxide for oxygen through inhalation and exhalation. The paper highlights that many diseases endanger the respiratory system, interfering with its function. Tuberculosis is caused by a bacterium called *Mycobacterium tuberculosis*. Normally, the bacterium invades the lungs, reducing the efficiency of lung functions. Tuberculosis can also cause damage to other parts of the body such as the brain or spine. The paper uses explainable artificial intelligence to visualize TB-infected parts of the lung for the subjective assessment.

CHAPTER-3

3. PROPOSED SYSTEM

The proposed system for automated TB detection from chest X-ray images involves several key components and processes:

1. **Image Acquisition and Preprocessing:** The system begins by acquiring chest X-ray images from digital radiography or other imaging modalities. These images may vary in quality and may contain artifacts or noise that could affect the performance of AI algorithms. Preprocessing techniques are applied to standardize and enhance the images, ensuring they are of high quality and suitable for analysis. Common preprocessing steps may include noise reduction, contrast enhancement, image normalization, and resizing to a consistent resolution.
2. **Deep Learning Algorithms:** The core of the system utilizes state-of-the-art deep learning algorithms, particularly convolutional neural networks (CNNs), which have shown remarkable success in image analysis tasks. CNNs are well-suited for learning hierarchical representations of images and extracting relevant features automatically. The system is trained on a large dataset of labeled chest X-ray images, comprising both TB-positive and TB-negative cases. During training, the network learns to identify patterns and features indicative of TB presence from the input images.
3. **Feature Extraction:** The trained deep learning model automatically extracts relevant features from the chest X-ray images. These features may include various visual patterns, textures, shapes, and spatial relationships present in the images, which are associated with TB pathology. The hierarchical layers of the CNN learn to detect increasingly abstract and complex features, allowing the model to capture subtle abnormalities indicative of TB.

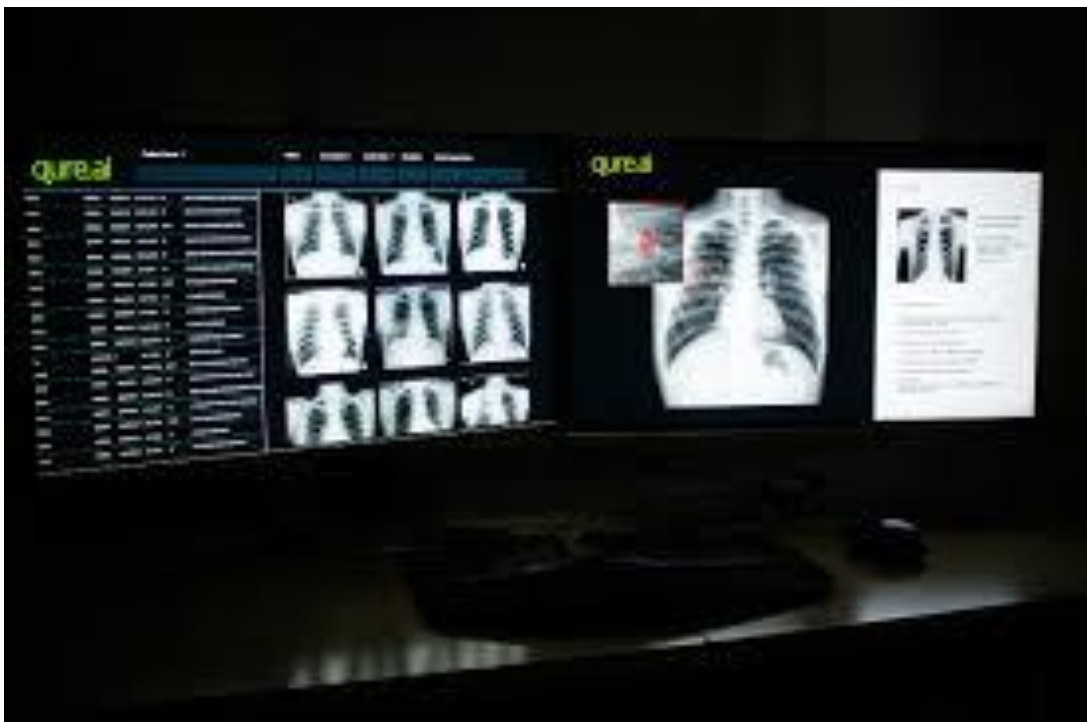


Fig 3.1. Proposed System

4. **Classification:** Once the features are extracted, the AI model performs classification by making predictions about whether a given chest X-ray image is TB-positive or TB-negative. This is typically achieved using a softmax activation function in the output layer of the network, which assigns probabilities to each class (TB-positive and TB-negative). The model's decision is based on the learned patterns and features extracted from the input image during training. The classification result provides a binary outcome, indicating the presence or absence of TB in the analyzed chest X-ray.
5. **Validation and Evaluation:** The performance of the AI model is assessed using a separate validation dataset, which consists of chest X-ray images not seen during training. Evaluation metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic (ROC) curve are commonly used to quantify the model's performance. Rigorous validation ensures that the AI system generalizes well to unseen data and produces reliable predictions in real-world scenarios.
6. **Integration into Clinical Workflow:** Once validated, the automated TB detection system can be integrated into clinical workflows to assist radiologists and healthcare professionals in interpreting chest X-ray images. The system provides rapid, objective, and consistent analysis, aiding in early detection and diagnosis of TB cases. Integration efforts may involve seamless integration with existing picture archiving and communication systems (PACS) or radiology information systems (RIS) to streamline the diagnostic process and facilitate timely patient management.

CHAPTER-4

4. SYSTEM REQUIREMENT SPECIFICATIONS

4.1 INTRODUCTION TO SRS

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

4.2 ROLE OF SRS

The Software Requirements Specification (SRS) plays a crucial role throughout the software development life cycle. Its significance lies in providing a clear and comprehensive understanding of what the software is expected to accomplish.

The key roles of SRS in the software development process:

Communication and Understanding:

Role:

Facilitates effective communication between stakeholders, including clients, users, project managers, and developers.

Importance:

Ensures that everyone involved has a shared understanding of the project goals, features, and functionalities.

Basis for Agreement:

Role:

Serves as a formal agreement or contract between the client and the development team.

Importance:

Establishes a clear understanding of what the software is expected to deliver, reducing the likelihood of misunderstandings.

Foundation for Development:

Role:

Provides a detailed blueprint for the development team, outlining the functional and non-functional requirements of the software.

Importance:

Guides developers in the design, coding, and testing phases, ensuring that the final product aligns with stakeholder expectations.

Basis for Testing:**Role:**

Serves as the foundation for creating test cases and validating the software.

Importance:

Enables the testing team to ensure that the software meets the specified requirements and functions as intended.

Guidance for Design and Implementation:**Role:**

Guides the design and implementation phases by specifying system features, constraints, and performance requirements.

Importance:

Helps developers make informed decisions about system architecture, design choices, and coding practices.

Change Management:**Role:**

Provides a baseline for managing changes to the software requirements.

Importance:

Helps control scope creep by serving as a reference for evaluating the impact of proposed changes and ensuring that changes align with the project's goals.

Reference for Documentation:**Role:**

Serves as a primary reference document throughout the software development life cycle.

Importance:

Provides a single, authoritative source for understanding and validating the software requirements, preventing inconsistencies in documentation.

Basis for Validation:**Role:**

Forms the basis for validating that the delivered software meets the specified requirements.

Importance:

Enables stakeholders to assess whether the final product aligns with their expectations and needs.

Risk Management:**Role:**

Identifies potential risks associated with the project, such as unclear requirements or conflicting expectations.

Importance:

Helps in proactively addressing risks, minimizing the likelihood of project delays or failures.

Collaboration and Feedback:**Role:**

Encourages collaboration and feedback from stakeholders throughout the development process.

Importance:

Enables continuous improvement by incorporating feedback, resolving issues, and adapting to evolving requirements.

The Software Requirements Specification serves as a critical document that guides the entire software development process, from initial discussions with stakeholders to the final validation of the delivered product. Its roles include communication facilitation, agreement establishment, development guidance, testing foundation, change management, and more, contributing to the overall success of the software project.

4.3 REQUIREMENTS SPECIFICATION DOCUMENT

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the names SRS report, software document.

A software document is primarily prepared for a project, software or any kind of application. There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and non-functional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

The purpose of SRS (Software Requirement Specification) document is to describe the external behavior of the application developed or software. It defines the operations, performance and interfaces and quality assurance requirement of the application or software. The complete software requirements for the system are captured by the SRS. This section introduces the requirement specification document for Word Building Game using Alexa which enlists functional as well as non-functional requirements.

4.4 FUNCTIONAL REQUIREMENT

The System after careful analysis has been identified to be present with the modules. A functional requirement defines a function of a system or its components. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that defines what a system is supposed to accomplish the functional requirement specification documents the operation and activities that a system able to perform. Functional requirements include functions performed by specific screens, outlines of work flows performed by the system, and other business compliance requirements the system must meet.

4.5 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements (NFRs) serve a crucial role in software development by defining the overarching qualities and attributes of the resulting system. Unlike functional requirements that specify features and functionalities, NFRs focus on characteristics that contribute to the system's overall success, user satisfaction, and performance. These requirements set constraints on the product being developed, establishing boundaries for critical attributes such as safety, security, usability, reliability, and performance.

In addition to placing restrictions on the product, non-functional requirements also articulate external constraints that the system must adhere to. These constraints may encompass legal and regulatory requirements, industry standards, or interoperability considerations with other systems. By addressing these external factors, NFRs ensure that the system aligns with broader expectations and complies with relevant standards in its operational environment.

Examples of non-functional requirements include safety, ensuring that the system operates without causing harm to users or the environment, and security, which involves safeguarding

the system against unauthorized access and data breaches. Usability requirements define expectations for the user interface and experience, aiming to make the system user-friendly and aligned with user expectations. Reliability requirements ensure consistent and dependable system functionality under varying conditions, while performance requirements specify parameters such as response time, throughput, and resource usage.

Project management issues, such as costs, time, and schedule, are also considered as non-functional requirements. These requirements help in managing the project within defined budget constraints, meeting deadlines, and adhering to the specified schedule. In essence, non-functional requirements play a vital role in guiding the development process, ensuring that the resulting system not only meets functional expectations but also excels in terms of overall quality, reliability, and user satisfaction. Properly defining and managing these requirements is essential for delivering a well-rounded and successful software solution.

4.6 PERFORMANCE REQUIREMENTS

The performance of the system is a critical aspect measured in terms of the output provided by the application. The accuracy and efficiency of the system in detecting and classifying tuberculosis cases from chest X-ray images are paramount to its effectiveness. Requirement specification plays a pivotal role in the analysis and design of such a system. A well-defined set of requirements ensures that the system is tailored to fit into the required environment and can meet the expectations of its users.

The responsibility for providing requirement specifications lies largely with the users of the existing system, particularly those who will be utilizing the AI-based tuberculosis classification system. Users, including healthcare professionals and stakeholders, are the individuals with firsthand knowledge of the nuances and intricacies of tuberculosis detection and diagnosis. Their input is vital during the initial stages of development to ensure that the system is designed to cater to their specific needs.

Given the importance of accurate requirement specifications, it becomes evident that understanding the needs of the end-users is crucial. The requirements must be known early in the development process to guide the design of a system that aligns with the expectations and workflow of healthcare professionals. The challenge lies in the fact that altering the system after it has been designed can be complex and costly. Hence, designing a system that does not adequately address the requirements of the user community renders it ineffective.

The requirement specifications for an AI-based Tuberculosis Classification system in Chest X-rays can be broadly outlined as follows:

Interface Compatibility:

The system should seamlessly interface with existing healthcare systems, ensuring smooth integration into the workflow of healthcare professionals.

Accuracy:

The system should demonstrate high accuracy in detecting and classifying tuberculosis cases from chest X-ray images, minimizing false positives and false negatives.

Improvement Over Existing Systems:

The system should exhibit superior performance compared to traditional methods, offering advancements in terms of speed, accuracy, and efficiency in tuberculosis classification.

Requirements related to resources, response time, transaction rates, throughput, and benchmark specifications are essential considerations in the performance criteria of the system. Additionally, in this project, stakeholders include data publishers (or data holders collecting data from record owners like patients and doctors), data miners or the public (data recipients), and record owners. Clear specification of their roles and requirements ensures a well-rounded and effective AI-based Tuberculosis Classification system in Chest X-rays.

4.7 SOFTWARE REQUIREMENTS

Software requirements are essential specifications that describe the functions, features, constraints, and qualities a software system must possess. They provide a comprehensive outline of what the software is expected to achieve and how it should behave. The software requirements serve as a foundation for the entire software development process, guiding design, implementation, testing, and validation.

4.7.1 Interpreter - PyCharm

It features a lightning-fast source code editor, perfect for day to day use with support of hundreds of languages. It also helps to be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box section, snippets etc. User can run the application either on LINUX or Windows with an internet connection and any internet browser.

Acceptance criteria: Before accepting, the developer must check whether the application is running properly or not and should also check whether the data is correctly sorted or not.



Fig 4.1. PyCharm

4.7.2 Python

Python is a high-level, versatile programming language known for its simplicity and readability. It features dynamic typing and offers a wide range of libraries, making it ideal for various applications, including web development, data analysis, and artificial intelligence. Python's elegant syntax promotes code readability and productivity. It supports both procedural and object-oriented programming. Python's community-driven development has resulted in extensive third-party packages and frameworks, such as Django, NumPy, and TensorFlow. Its cross-platform compatibility allows code to run on different operating systems. Python's use in automation, scripting, and scientific computing has made it one of the most popular programming languages globally. Use of Python 3.6 or later is advised.



Fig 4.2. Python

4.7.3 NUMPY

NumPy, short for "Numerical Python," is a fundamental Python library for scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a vast collection of mathematical functions to operate on these arrays efficiently. NumPy is essential for tasks involving data manipulation, transformation, and analysis in fields like data science, machine learning, and scientific research. Its speed and memory efficiency come from its ability to execute complex mathematical computations without the need for explicit loops. NumPy also facilitates seamless integration with other data analysis libraries in Python, making it a cornerstone of the scientific Python ecosystem.



Fig 4.3. NumPy

4.7.4 TENSORFLOW

TensorFlow is a powerful and widely-used open-source machine learning framework developed by Google. Renowned for its flexibility and versatility, TensorFlow provides a comprehensive ecosystem that caters to various machine learning tasks, from neural networks to natural language processing and image recognition. It features high-level APIs like Keras for simplified model development and integrates seamlessly with tools like TensorBoard for visualizing training processes. TensorFlow's support for distributed computing enables the scaling of training across multiple GPUs and distributed clusters, making it suitable for handling large datasets and complex models. With an active and vibrant community, TensorFlow continues to be at the forefront of advancements in machine learning, offering deployment options for various platforms, including cloud services, mobile devices, and edge computing. Its lightweight version, TensorFlow Lite, extends its reach to mobile and embedded devices, emphasizing its adaptability and impact across diverse applications in the field of artificial intelligence.



Fig 4.4. TensorFlow

4.7.5 MATPLOTLIB

Matplotlib is a powerful and versatile plotting library for Python, widely used in the fields of data science, machine learning, and scientific research. Known for its flexibility and ease of use, Matplotlib provides a comprehensive set of tools for creating static, animated, and interactive visualizations. With a syntax that is intuitive and similar to MATLAB, Matplotlib allows users to generate a variety of plots, charts, and graphs, including line plots, scatter plots, bar charts, histograms, and more. Its extensive customization options enable users to fine-tune the appearance of plots, from adjusting colors and markers to modifying axis scales and adding annotations. Matplotlib plays a crucial role in conveying complex data insights through compelling and visually appealing graphics, making it an essential tool in the Python ecosystem for exploratory data analysis and presentation of results.



Fig 4.5. Matplotlib

4.8 HARDWARE REQUIREMENTS

Hardware requirements refer to the specifications and components necessary for the proper functioning of a software system. These requirements are essential for guiding the selection, configuration, and setup of the physical equipment on which the software will run. The hardware specifications are crucial to ensure optimal performance, compatibility, and scalability of the software.

Key aspects of hardware requirements include:

Processor (CPU):

Specifies the type and speed of the central processing unit (CPU) required for the software. It influences the system's overall processing power and performance.

Memory (RAM):

Indicates the amount of random-access memory (RAM) needed for the software to run efficiently. Sufficient RAM is crucial for handling large datasets and complex computations.

Storage (Hard Disk or SSD):

Specifies the storage capacity and type, whether it is a traditional hard disk drive (HDD) or a solid-state drive (SSD). The storage capacity should accommodate the software, data, and potential future expansion.

Graphics Processing Unit (GPU):

Specifies the type and capabilities of the graphics card, particularly important for applications that involve graphical processing, such as 3D rendering, simulations, or machine learning tasks.

Network Interface Card (NIC):

Defines the type and speed of the network interface card required for network connectivity. It is crucial for software that involves communication over local area networks (LANs) or the internet.

CHAPTER-5

5. IMPLEMENTATION

5.1 APPLICATION:

Introducing a cutting-edge tuberculosis detection system: Powered by React frontend, FastAPI backend, and YOLOv8 model. React ensures an intuitive user interface, while FastAPI delivers high-speed backend processing. YOLOv8, renowned for its accuracy and efficiency, enables robust chest X-ray image analysis. Together, they form a seamless workflow, automating the detection of TB abnormalities with precision. This innovative solution promises rapid diagnosis, reducing manual workload and improving patient outcomes. Welcome to the forefront of medical technology, where advanced AI meets user-friendly design, revolutionizing tuberculosis screening in healthcare.

5.1.1 YOLO:

The YOLO (You Only Look Once) series of models has become famous in the computer vision world. YOLO's fame is attributable to its considerable accuracy while maintaining a small model size. YOLO models can be trained on a single GPU, which makes it accessible to a wide range of developers. Machine learning practitioners can deploy it for low cost on edge hardware or in the cloud.

YOLO has been nurtured by the computer vision community since its first launch in 2015 by Joseph Redmond. In the early days (versions 1-4), YOLO was maintained in C code in a custom deep learning framework written by Redmond called Darknet. YOLOv5 inferring on a bicycle

YOLOv8 author, Glenn Jocher at Ultralytics, shadowed the YOLOv3 repo in PyTorch (a deep learning framework from Facebook). As the training in the shadow repo got better, Ultralytics eventually launched its own model: YOLOv5.

YOLOv5 quickly became the world's SOTA repo given its flexible Pythonic structure. This structure allowed the community to invent new modeling improvements and quickly share them across repository with similar PyTorch methods.

Along with strong model fundamentals, the YOLOv5 maintainers have been committed to supporting a healthy software ecosystem around the model. They actively fix issues and push the capabilities of the repository as the community demands.

In the last two years, various models branched off of the YOLOv5 PyTorch repository, including Scaled-YOLOv4, YOLOR, and YOLOv7.

Other models emerged around the world out of their own PyTorch based implementations, such as YOLOX and YOLOv6. Along the way, each YOLO model has brought new SOTA techniques that continue to push the model's accuracy and efficiency.

Over the last six months, Ultralytics worked on researching the newest SOTA version of YOLO, YOLOv8. YOLOv8 was launched on January 10th, 2023.

YOLOv8:

Here are a few main reasons why you should consider using YOLOv8 for your next computer vision project:

1. YOLOv8 has a high rate of accuracy measured by Microsoft COCO and Roboflow 100.
2. YOLOv8 comes with a lot of developer-convenience features, from an easy-to-use CLI to a well-structured Python package.
3. There is a large community around YOLO and a growing community around the YOLOv8 model, meaning there are many people in computer vision circles who may be able to assist you when you need guidance.

YOLOv8 achieves strong accuracy on COCO. For example, the YOLOv8m model -- the medium model -- achieves a 50.2% map when measured on COCO. When evaluated against Roboflow 100, a dataset that specifically evaluates model performance on various task-specific domains, YOLOv8 scored substantially better than YOLOv5. More information on this is provided in our performance analysis later in the article.

Furthermore, the developer-convenience features in YOLOv8 are significant. As opposed to other models where tasks are split across many different Python files that you can execute, YOLOv8 comes with a CLI that makes training a model more intuitive. This is in addition to a Python package that provides a more seamless coding experience than prior models.

The community around YOLO is notable when you are considering a model to use. Many computer vision experts know about YOLO and how it works, and there is plenty of guidance online about using YOLO in practice. Although YOLOv8 is new as of writing this piece, there are many guides online that can help.

YOLOv8 ARCHITECTURE:

YOLOv8 does not yet have a published paper, so we lack direct insight into the direct research methodology and ablation studies done during its creation. With that said, we analyzed the repository and information available about the model to start documenting what's new in YOLOv8.

If you want to peer into the code yourself, check out the YOLOv8 repository and you view this code differential to see how some of the research was done.

Here we provide a quick summary of impactful modeling updates and then we will look at the model's evaluation, which speaks for itself.

Anchor Free Detection

YOLOv8 is an anchor-free model. This means it predicts directly the center of an object instead of the offset from a known anchor box.

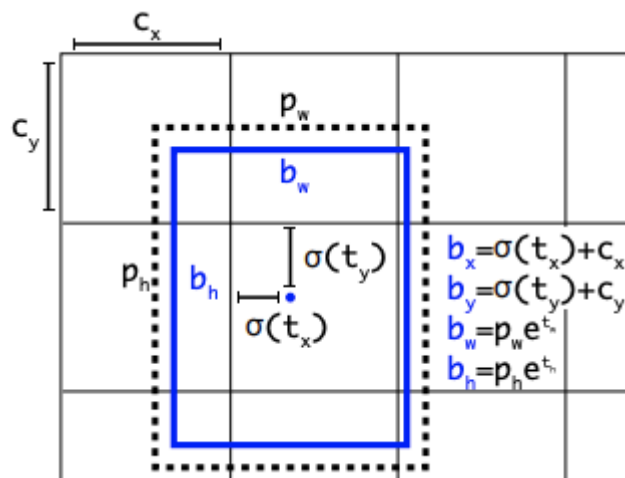


Fig 5.1. Visualization of an anchor box

Anchor boxes were a notoriously tricky part of earlier YOLO models, since they may represent the distribution of the target benchmark's boxes but not the distribution of the custom dataset.

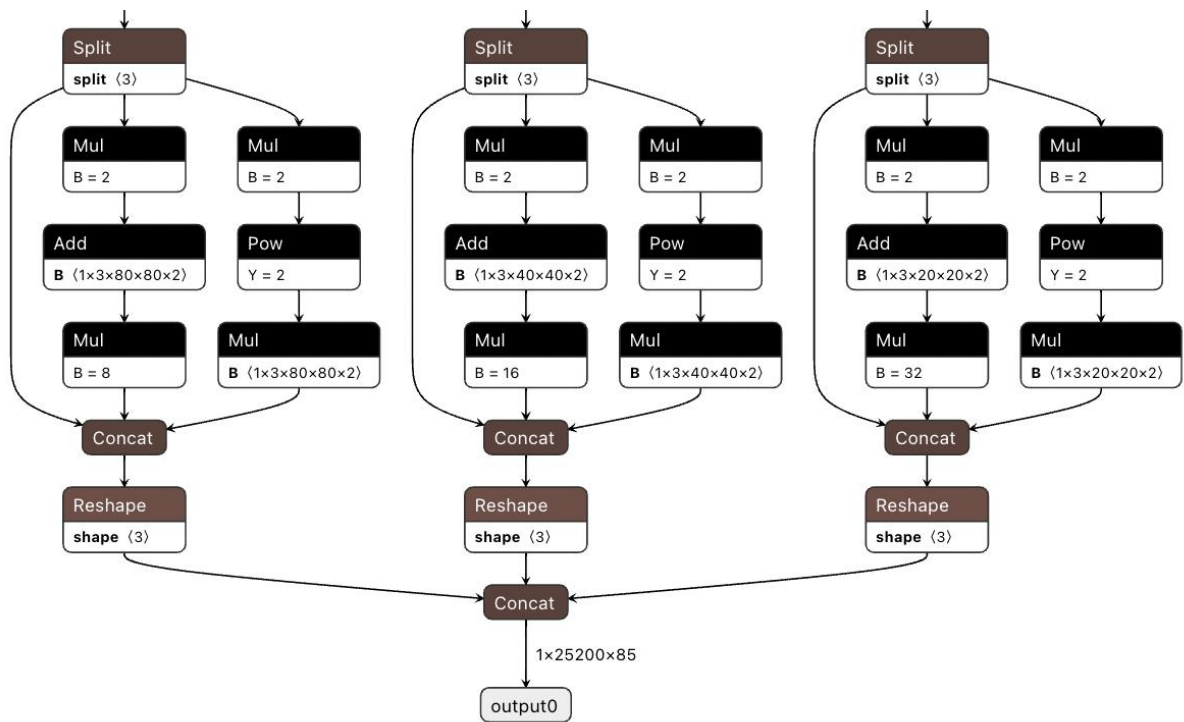


Fig 5.2. The detection head of YOLOv5

Anchor free detection reduces the number of box predictions, which speeds up Non-Maximum Suppression (NMS), a complicated post processing step that sifts through candidate detections after inference.

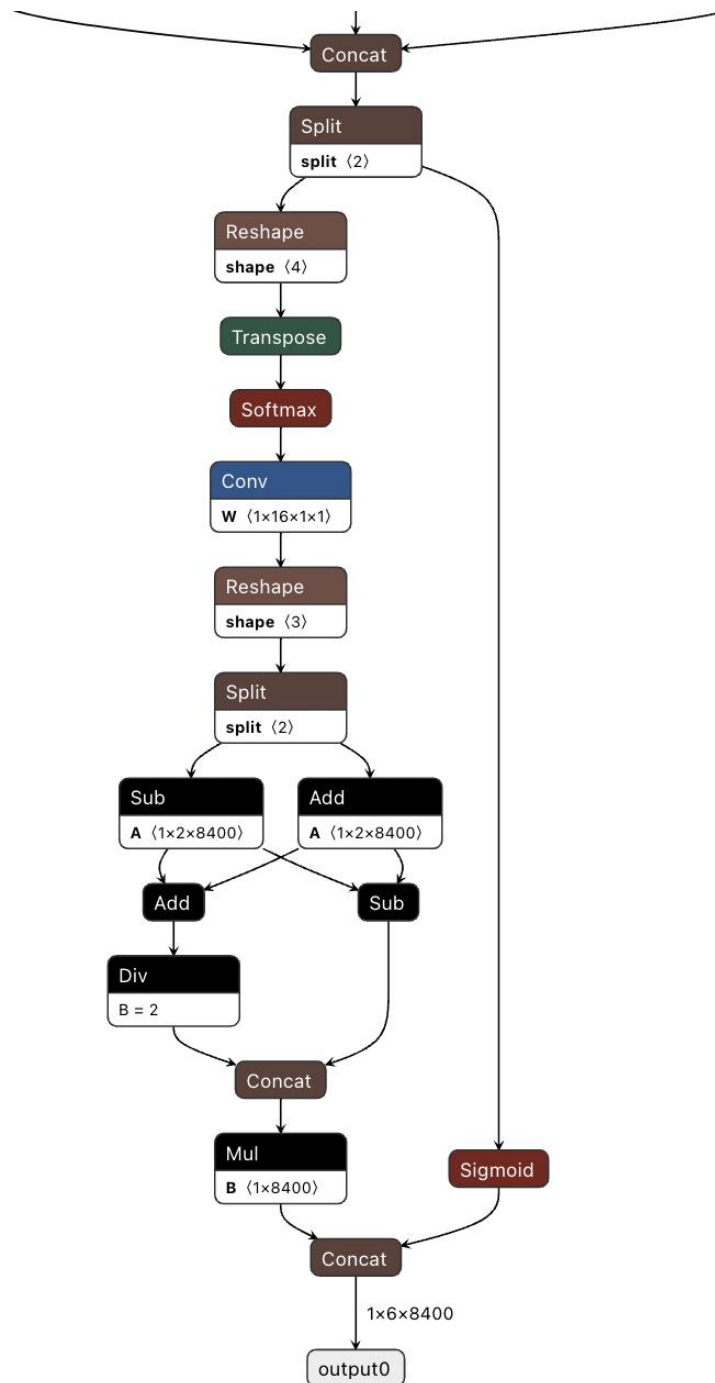


Fig 5.3. The detection head of YOLOv8

New Convolutions

The stem's first 6x6 conv is replaced by a 3x3 the main building block was changed, and C2f replaced C3 . The module is summarized in the picture below, where "f" is the number of features, "e" is the expansion rate and CBS is a block composed of a Conv a BatchNorm and a siLu later.

In C2f, all the outputs from the Bottleneck (fancy name for two 3x3 convs with residual connections) are concatenated. While in C3 only the output of the last Bottleneck was used.

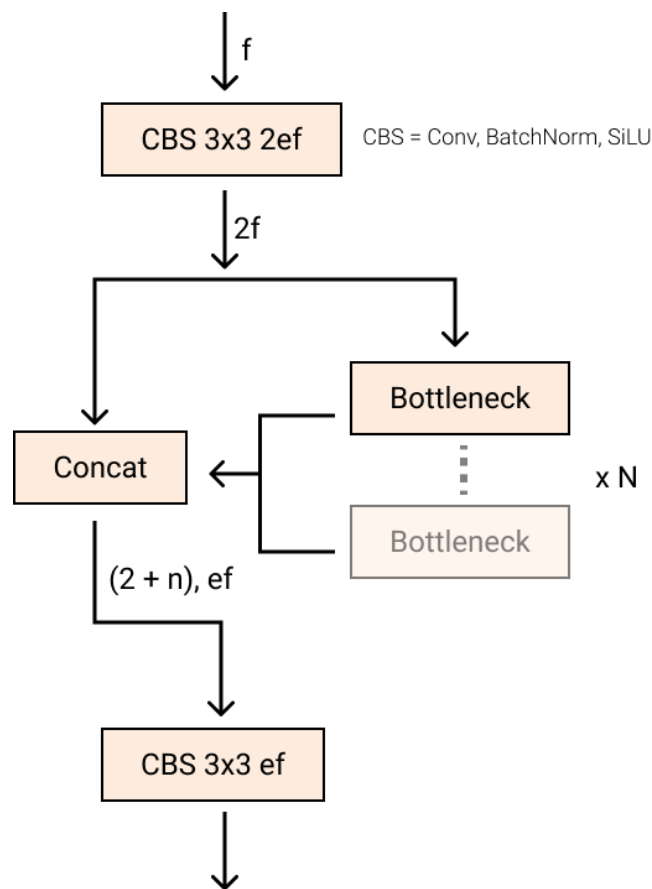


Fig 5.4. New YOLOv8 C2f module

The Bottleneck is the same as in YOLOv5 but the first conv's kernel size was changed from 1x1 to 3x3. From this information, we can see that YOLOv8 is starting to revert to the ResNet block defined in 2015.

In the neck, features are concatenated directly without forcing the same channel dimensions. This reduces the parameters count and the overall size of the tensors.

Closing the Mosaic Augmentation

Deep learning research tends to focus on model architecture, but the training routine in YOLOv5 and YOLOv8 is an essential part of their success.

YOLOv8 augments images during training online. At each epoch, the model sees a slightly different variation of the images it has been provided.

One of those augmentations is called mosaic augmentation. This involves stitching four images together, forcing the model to learn objects in new locations, in partial occlusion, and against different surrounding pixels.

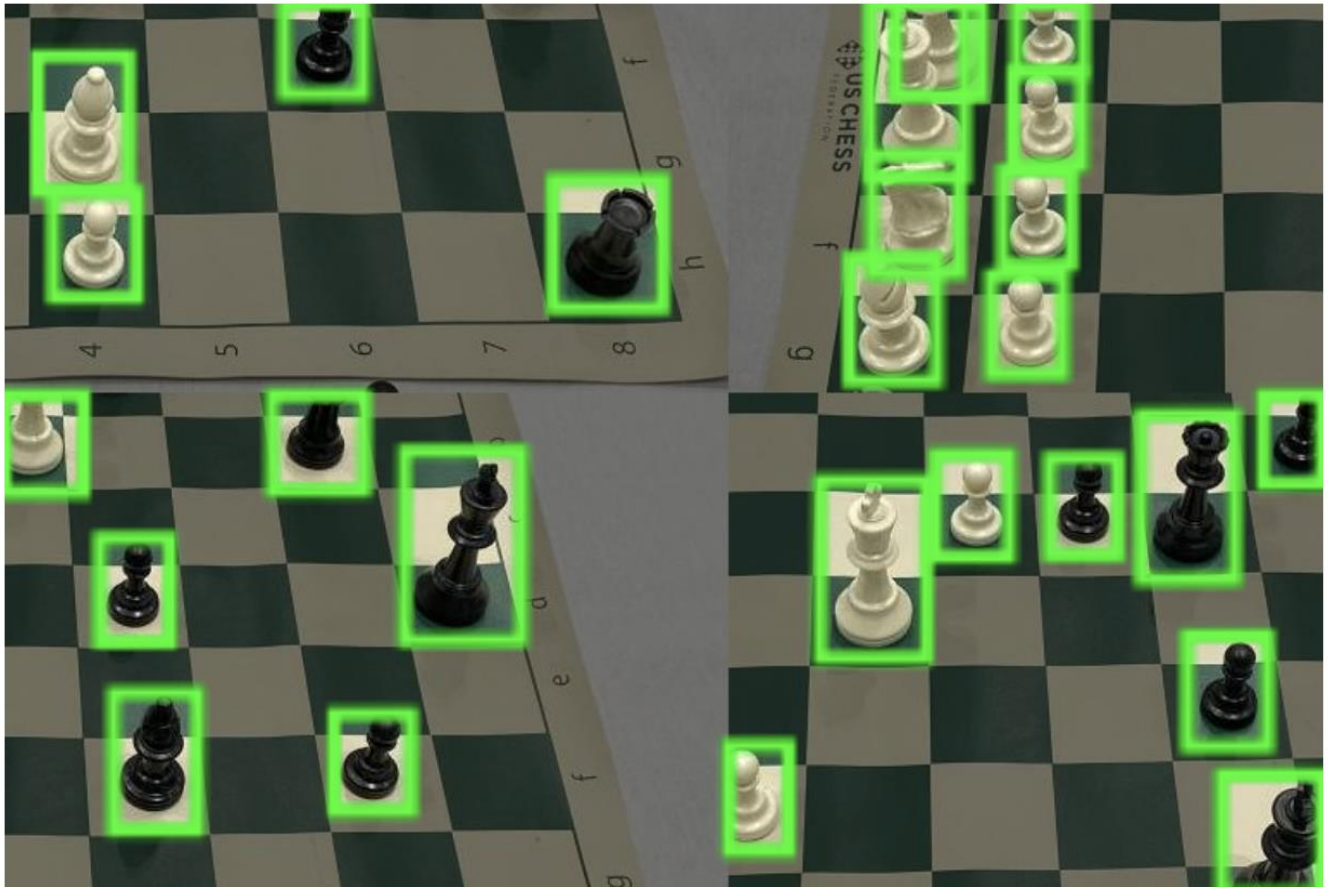


Fig 5.5. Mosaic augmentation of chess board photos

However, this augmentation is empirically shown to degrade performance if performed through the whole training routine. It is advantageous to turn it off for the last ten training epochs.

This sort of change is exemplary of the careful attention YOLO modeling has been given in overtime in the YOLOv5 repo and in the YOLOv8 research.

YOLOv8 Accuracy Improvements

YOLOv8 research was primarily motivated by empirical evaluation on the COCO benchmark. As each piece of the network and training routine are tweaked, new experiments are run to validate the changes effect on COCO modeling.

YOLOv8 COCO Accuracy

COCO (Common Objects in Context) is the industry standard benchmark for evaluating object detection models. When comparing models on COCO, we look at the mAP value and FPS measurement for inference speed. Models should be compared at similar inference speeds. The image below shows the accuracy of YOLOv8 on COCO, using data collected by the Ultralytics team and published in their YOLOv8 README:

▼ Detection

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU (ms)	Speed T4 GPU (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	-	-	3.2	8.7
YOLOv8s	640	44.9	-	-	11.2	28.6
YOLOv8m	640	50.2	-	-	25.9	78.9
YOLOv8l	640	52.9	-	-	43.7	165.2
YOLOv8x	640	53.9	-	-	68.2	257.8

- **mAP^{val}** values are for single-model single-scale on [COCO val2017](#) dataset.
Reproduce by `yolo mode=val task=detect data=coco.yaml device=0`
- **Speed** averaged over COCO val images using an [Amazon EC2 P4d](#) instance.
Reproduce by `yolo mode=val task=detect data=coco128.yaml batch=1 device=0/cpu`

Fig 5.6. YOLOv8 COCO evaluation

YOLOv8 COCO accuracy is state of the art for models at comparable inference latencies as of writing this post.

RF100 Accuracy

At Roboflow, we have drawn 100 sample datasets from Roboflow Universe, a repository of over 100,000 datasets, to evaluate how well models generalize to new domains. Our benchmark, developed with support from Intel, is a benchmark for computer vision practitioners designed to provide a better answer to the question: "how well will this model work on my custom dataset?"

The box plot below tells us YOLOv8 had fewer outliers and an overall better mAP when measured against the Roboflow 100 benchmark.

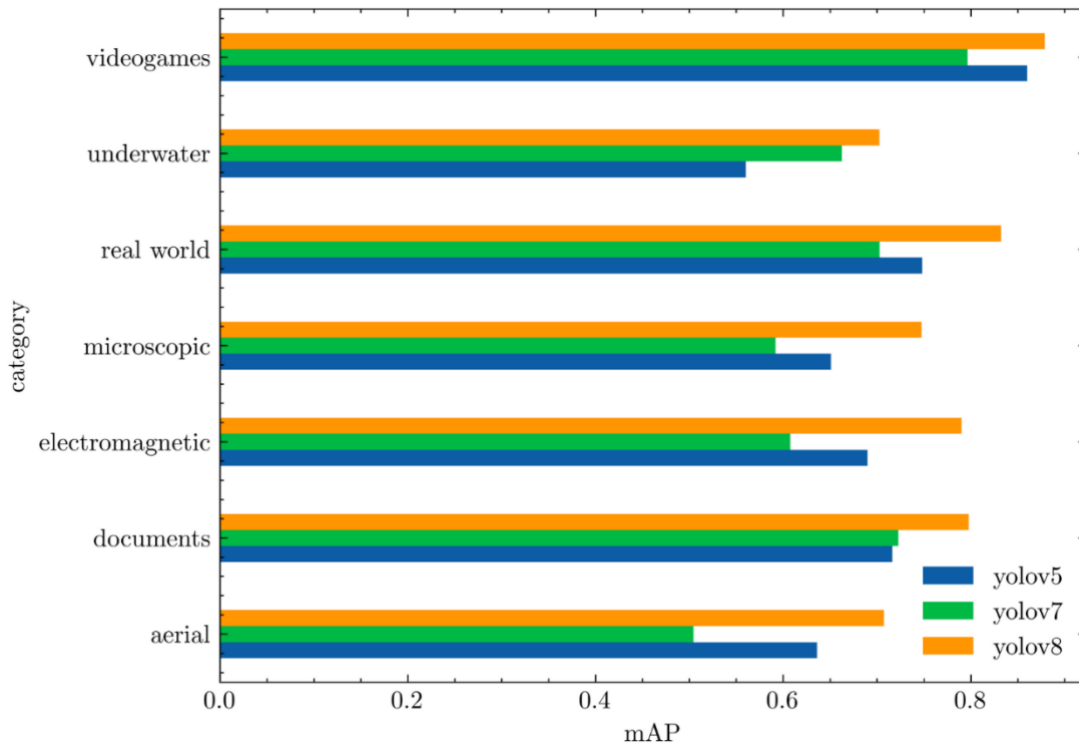


Fig 5.7. Comparison

Relative to the YOLOv5 evaluation, the YOLOv8 model produces a similar result on each dataset, or improves the result significantly.

5.1.2 DATASET:

The TBX11K dataset is a large dataset containing 11000 chest x-ray images. It's the only TB dataset that I know of that includes TB bounding boxes. This allows both classification and detection models to be trained.

However, it can be mentally tiring to get started with this dataset. It includes many xml, json and txt files that you need to sift through to try to understand what everything means, how it all fits together and how to extract the bounding box coordinates.

Here I've simplified the dataset. Now there's just one csv file, one folder containing the training images and one folder containing the test images.

fname	image_height	image_width	source	bbox	target	tb_type	image_type
h0001.png	512	512	train	none	no_tb	none	healthy
h0003.png	512	512	train	none	no_tb	none	healthy
h0005.png	512	512	train	none	no_tb	none	healthy
h0008.png	512	512	train	none	no_tb	none	healthy
h0009.png	512	512	train	none	no_tb	none	healthy
h0010.png	512	512	train	none	no_tb	none	healthy
h0012.png	512	512	train	none	no_tb	none	healthy
h0013.png	512	512	train	none	no_tb	none	healthy
h0016.png	512	512	train	none	no_tb	none	healthy
h0017.png	512	512	train	none	no_tb	none	healthy
h0019.png	512	512	train	none	no_tb	none	healthy
h0020.png	512	512	train	none	no_tb	none	healthy
h0021.png	512	512	train	none	no_tb	none	healthy
h0022.png	512	512	train	none	no_tb	none	healthy
h0023.png	512	512	train	none	no_tb	none	healthy
h0025.png	512	512	train	none	no_tb	none	healthy
h0026.png	512	512	train	none	no_tb	none	healthy
h0027.png	512	512	train	none	no_tb	none	healthy
h0028.png	512	512	train	none	no_tb	none	healthy
h0030.png	512	512	train	none	no_tb	none	healthy

Fig 5.8. Dataset

Files and Folders:

1. CSV File:

File Name: tbx11k_dataset.csv

Content: This CSV file serves as the primary source of information and metadata for the dataset. It likely includes columns for image file names, labels, and bounding box coordinates.

2. Training Images Folder:

Folder Name: train_images

Content: This folder contains the training set of chest X-ray images in a standardized format (e.g., JPEG or PNG). Each image corresponds to a row in the CSV file.

tb0192.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0192.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0193.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0194.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0197.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0199.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0199.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0200.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0200.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0202.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0203.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0203.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0205.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0209.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0209.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0212.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0212.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0216.png	512	512	train	{'xmin':	tb	active_tb	tb
tb0217.png	512	512	train	{'xmin':	tb	active_tb	tb

Fig 5.9. Training Dataset

1. Test Images Folder:

Folder Name: test_images

Content: This folder contains the test set of chest X-ray images in the same standardized format as the training images. These images are distinct from the training set.

CSV File Details:

1. Image File Column:

Column Name: image_filename

Description: Contains the names or paths of the chest X-ray images. Each entry corresponds to an image in the dataset.

2. Labels Column:

Column Name: labels

Description: Specifies the class labels associated with each image. In the context of TB detection, this might indicate whether an image contains TB-related abnormalities.

3. Bounding Box Coordinates Column:

Column Name: box_coordinates

Description: Provides the bounding box coordinates for TB-related abnormalities in the corresponding image. The coordinates are likely in a standardized format, such as (x_min, y_min, x_max, y_max).

Dataset Usage:

1. Training:

The images in the train_images folder along with the information in the CSV file to train classification and detection models for TB detection.

2. Testing:

The test_images folder can be used to evaluate the performance of the trained models on unseen data.

3. Data Exploration:

The CSV file simplifies the process of understanding and navigating the dataset. It provides a comprehensive overview of the images, their labels, and bounding box information.

Dataset Advantages:

1. Simplicity:

The simplified structure reduces the complexity of working with the TBX11K dataset, making it more accessible for researchers and practitioners.

2. Unified Information:

- All relevant information is consolidated in the CSV file, streamlining the data exploration and model development processes.

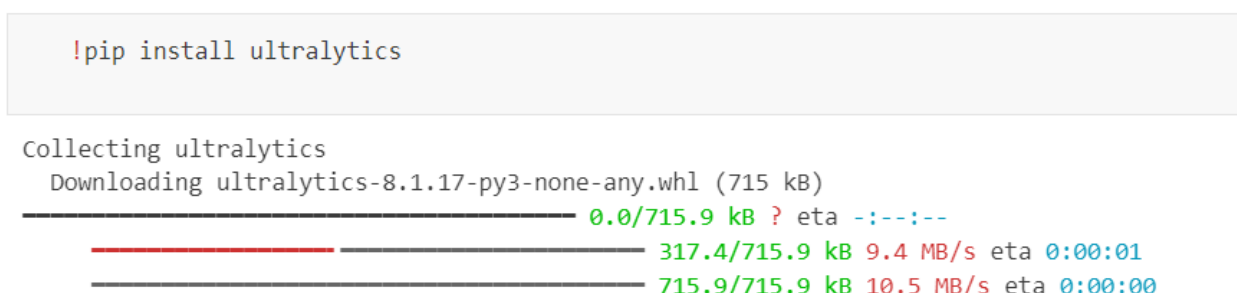
By consolidating the information into a single CSV file and organizing the images into specific folders, you've made it more convenient for users to comprehend and utilize the TBX11K dataset for classification and detection tasks related to tuberculosis in chest X-ray images.

5.1.3 Code Snippets:

1. Model (YoloV8.ipynb):

- The notebook starts by checking the GPU information using “!nvidia-smi”.
- Installs the ultralytics library and then uses it to predict bounding boxes on an example image (bus.jpg) using a pre-trained YOLOv8n model (yolov8n.pt).

```
!pip install ultralytics
```



```
Collecting ultralytics
  Downloading ultralytics-8.1.17-py3-none-any.whl (715 kB)
  0.0/715.9 kB ? eta -:--:--
  317.4/715.9 kB 9.4 MB/s eta 0:00:01
  715.9/715.9 kB 10.5 MB/s eta 0:00:00
```

Fig 5.10. Installing Library

- Mounts Google Drive to access and save files.
- Changes the working directory to where the YOLO project is stored in Google Drive (/content/drive/MyDrive/Colab Notebooks/YOLO).

- Runs YOLO training using the yolo command, specifying the model (yolov8m.pt), dataset configuration file (config.yaml), training mode, and other parameters. This is performed for 50 epochs with an image size of 512.

	from	n	params	module	arguments
0	-1	1	1392	ultralytics.nn.modules.conv.Conv	[3, 48, 3, 2]
1	-1	1	41664	ultralytics.nn.modules.conv.Conv	[48, 96, 3, 2]
2	-1	2	111360	ultralytics.nn.modules.block.C2f	[96, 96, 2, True]
3	-1	1	166272	ultralytics.nn.modules.conv.Conv	[96, 192, 3, 2]
4	-1	4	813312	ultralytics.nn.modules.block.C2f	[192, 192, 4, True]
5	-1	1	664320	ultralytics.nn.modules.conv.Conv	[192, 384, 3, 2]
6	-1	4	3248640	ultralytics.nn.modules.block.C2f	[384, 384, 4, True]
7	-1	1	1991808	ultralytics.nn.modules.conv.Conv	[384, 576, 3, 2]
8	-1	2	3985920	ultralytics.nn.modules.block.C2f	[576, 576, 2, True]
9	-1	1	831168	ultralytics.nn.modules.block.SPPF	[576, 576, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']

Fig 5.11. Configuration

```
50 epochs completed in 0.320 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 52.0MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.0.229 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25840339 parameters, 0 gradients, 78.7 GFLOPs

```

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	9	15	0.84	0.867	0.93	0.514

```
Speed: 0.2ms preprocess, 9.6ms inference, 0.0ms loss, 1.8ms postprocess per image
Results saved to runs/detect/train2
```

Fig 5.12. Training Phase

- Displays a confusion matrix image generated during the training process.

```
from IPython.display import display, Image
Image(filename='/content/drive/MyDrive/Colab Notebooks/YOLO/runs/detect/train2/confusion_matrix.png',width=600)
```

Fig 5.13. Displaying Confusion matrix

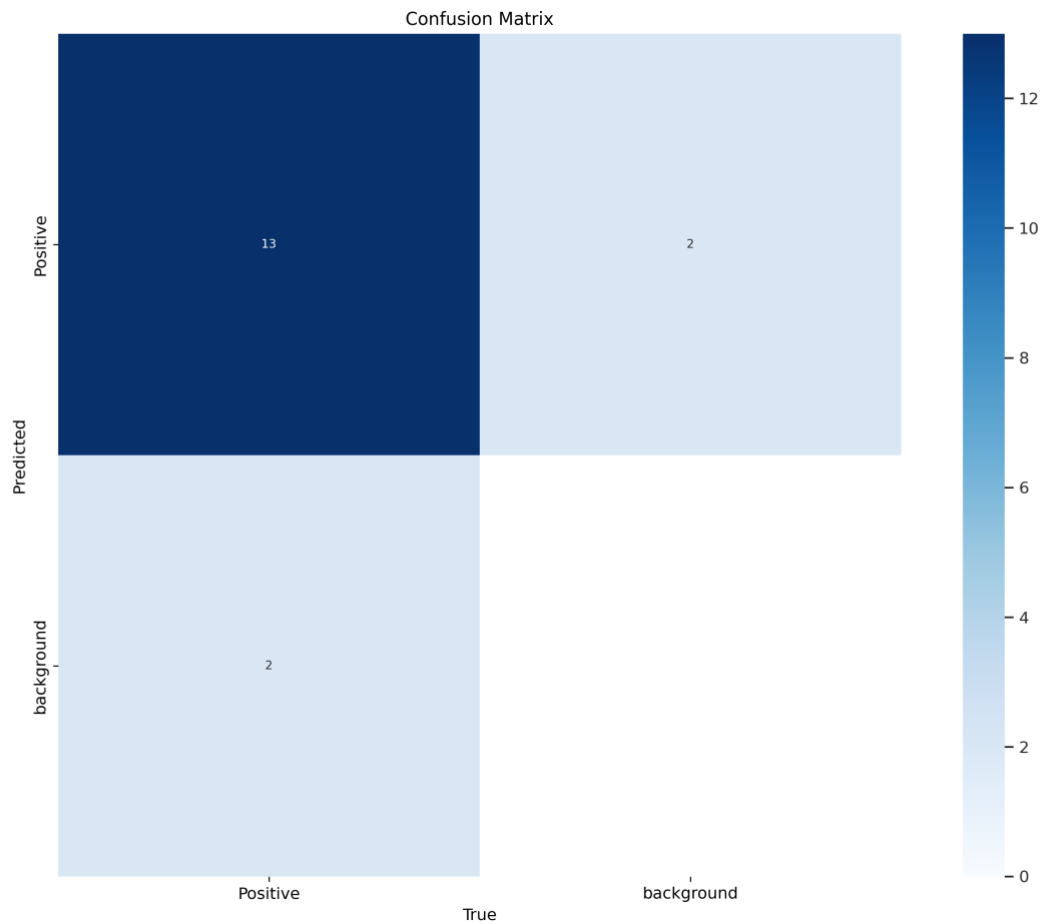


Fig 5.14. Confusion Matrix

- Performs detection on the validation set and makes predictions on the test set using the trained model.

```
!yolo task=detect mode=predict model=runs/detect/train2/weights/best.pt conf=0.25 source=data/test/images
```

Ultralytics YOLOv8.1.17 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25840339 parameters, 0 gradients, 78.7 GFLOPs

```
image 1/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0001.png: 512x512 (no detections), 25.4ms
image 2/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0003.png: 512x512 (no detections), 25.4ms
image 3/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0004.png: 512x512 (no detections), 25.4ms
image 4/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0005.png: 512x512 (no detections), 25.5ms
image 5/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0006.png: 512x512 (no detections), 25.5ms
image 6/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0008.png: 512x512 (no detections), 25.5ms
image 7/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0009.png: 512x512 (no detections), 25.5ms
image 8/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0010.png: 512x512 (no detections), 25.5ms
image 9/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0012.png: 512x512 (no detections), 25.4ms
image 10/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/h0013.png: 512x512 (no detections), 25.5ms
image 11/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1170.png: 512x512 2 Positives, 25.5ms
image 12/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1173.png: 512x512 2 Positives, 25.4ms
image 13/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1174.png: 512x512 1 Positive, 25.4ms
image 14/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1175.png: 512x512 2 Positives, 25.4ms
image 15/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1177.png: 512x512 2 Positives, 25.4ms
image 16/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1178.png: 512x512 2 Positives, 25.5ms
image 17/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1179.png: 512x512 2 Positives, 25.5ms
image 18/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1180.png: 512x512 1 Positive, 25.4ms
image 19/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1181.png: 512x512 1 Positive, 25.4ms
image 20/20 /content/drive/MyDrive/Colab Notebooks/YOLO/data/test/images/tb1183.png: 512x512 2 Positives, 25.4ms
Speed: 1.4ms preprocess, 25.5ms inference, 36.3ms postprocess per image at shape (1, 3, 512, 512)
Results saved to runs/detect/predict2
```

Fig 5.15. Prediction

2. prediction:

- Imports necessary modules including FastAPI, YOLO from ultralytics, and other utilities.

```
from ultralytics import YOLO
import os
import shutil
```

Fig 5.16. Imports

- Initializes a FastAPI application (app).
- Configures Cross-Origin Resource Sharing (CORS) middleware to allow requests from specified origins.
- Initializes a YOLO model using the pre-trained weights (best.pt).
- Defines utility functions like get_new_filename and process_image for handling file names and processing images.

```

MODEL_PATH = 'best.pt'
# Load the YOLOv8 model
model = YOLO(MODEL_PATH)
# Make prediction
image_path = 'C:/Users/vikas/mig/fastapi/example.png'
results_list = model.predict(image_path, save=True, imgsz=512, conf=0.25)
results = results_list[0]

output_dir = results.save_dir
filename = os.path.basename(image_path)
shutil.move(output_dir, 'C:/Users/vikas/mig/fastapi/')
output_image = 'C:/Users/vikas/mig/fastapi/predict/'+filename
# shutil.rmtree('C:/Users/vikas/mig/fastapi/predict')
print(output_image)

```

Fig 5.17. Detection

- Implements two main endpoints:
 - **/ (Root Endpoint):** Returns a simple message confirming the successful running of the TB Detection API.
 - **/tbdetection/ (TB Detection Endpoint):** Accepts X-ray image uploads, saves the image, processes it using the YOLO model, and returns the processed image with bounding boxes as a streaming response.

Uses uvicorn to run the FastAPI application on 0.0.0.0:8000.

3. FastAPI

- Imports necessary modules including FastAPI, File, UploadFile, HTTPException, StreamingResponse, and CORS middleware.

```

from fastapi import FastAPI, File, UploadFile, HTTPException
from fastapi.responses import StreamingResponse
from fastapi.middleware.cors import CORSMiddleware
import os
import shutil
from ultralytics import YOLO
import uvicorn

```

Fig 5.18. Modules

- Initializes a FastAPI application (app).
- Configures CORS middleware to handle cross-origin resource sharing.

```
origins = ["http://localhost", "http://localhost:3000", "*"]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

Fig 5.19. Middleware

- Initializes a YOLO model using the pre-trained weights (best.pt).
- Defines utility functions like get_new_filename and process_image for handling file names and processing images.

```
# Initialize YOLO model
MODEL_PATH = 'best.pt'
model = YOLO(MODEL_PATH)

def get_new_filename(output_dir, filename):
    new_filename = filename
    i = 1
    while os.path.exists(os.path.join(output_dir, new_filename)):
        name, ext = os.path.splitext(filename)
        new_filename = f"{name}({i}){ext}"
        i += 1
    return new_filename

def process_image(file_path):
    results_list = model.predict(file_path, save=True, imgsz=512, conf=0.25)
    results = results_list[0]
    os.remove(file_path)
    output_dir = results.save_dir
    # print(results)
    filename = os.path.basename(file_path)
    extracted_folder = os.path.basename(output_dir)
    new_filename = get_new_filename('C:/Users/vikas/mig/fastapi/output/', filename)
    shutil.move(output_dir + '\\\\' + filename, os.path.join('C:/Users/vikas/mig/fastapi/output/', new_filename))
    output_image = 'C:/Users/vikas/mig/fastapi/output/' + new_filename

    return output_image
```

Fig 5.20. Saved Model

```

@app.get("/")
async def root():
    return {"message": "TB Detection"}

@app.post("/tbdetection/")
async def tb_detection(file: UploadFile = File(...)):
    try:
        # Save the uploaded file
        file_path = f'C:/Users/vikas/mig/fastapi/{file.filename}'
        with open(file_path, "wb") as buffer:
            shutil.copyfileobj(file.file, buffer)
        # Process the image using the YOLO model
        output_image_path = process_image(file_path)
        # Return the processed image to the frontend
        return StreamingResponse(open(output_image_path, "rb"), media_type="image/jpeg")

    except Exception as e:
        # Log the error
        print(f"Error processing image: {str(e)}")

        # Raise HTTP exception with a meaningful error message
        raise HTTPException(status_code=500, detail="Error processing image")

```

Fig 5.21. tbdetection

- Uses uvicorn to run the FastAPI application on 0.0.0.0:8000.

Execution Flow:

1. Model Training:

The YOLOv8 model is trained using the Jupyter Notebook (YoloV8.ipynb). The best weights are saved as best.pt.

2. FastAPI Initialization:

The FastAPI application is initialized in both the Jupyter Notebook (YoloV8.ipynb) and the Python script (predict.py or app.py).

3. YOLO Model Loading:

The YOLO model is loaded in both the Python script (predict.py and app.py).

4. API Endpoint Configuration:

The FastAPI application provides an endpoint (/tbdetection/) to receive X-ray images.

5. Image Processing:

When an image is uploaded to the /tbdetection/ endpoint, it is saved and processed by the YOLO model.

6. Response:

The processed image with bounding boxes is returned as a streaming response.

4. FRONTEND:

What Is React:



Fig 5.22. React

React is a framework that employs Webpack to automatically compile React, JSX, and ES6 code while handling CSS file prefixes. React is a JavaScript-based UI development library. Although React is a library rather than a language, it is widely used in web development. The library first appeared in May 2013 and is now one of the most commonly used frontend libraries for web development.

React offers various extensions for entire application architectural support, such as Flux and React Native, beyond mere UI.

When compared to other technologies on the market, React is a new technology. Jordan Walke, a software engineer at Facebook, founded the library in 2011, giving it life. The likes of XHP, a straightforward HTML component framework for PHP, have an influence on React. React's newsfeed was its debut application in 2011. Later, Instagram picks it up and incorporates it into their platform.

Why React?

React's popularity today has eclipsed that of all other front-end development frameworks. Here is why:

- **Easy creation of dynamic applications:** React makes it easier to create dynamic web applications because it requires less coding and offers more functionality, as opposed to JavaScript, where coding often gets complex very quickly.

- Improved performance: React uses Virtual DOM, thereby creating web applications faster. Virtual DOM compares the components' previous states and updates only the items in the Real DOM that were changed, instead of updating all of the components again, as conventional web applications do.
- Reusable components: Components are the building blocks of any React application, and a single app usually consists of multiple components. These components have their logic and controls, and they can be reused throughout the application, which in turn dramatically reduces the application's development time.
- Unidirectional data flow: React follows a unidirectional data flow. This means that when designing a React app, developers often nest child components within parent components. Since the data flows in a single direction, it becomes easier to debug errors and know where a problem occurs in an application at the moment in question.
- Small learning curve: React is easy to learn, as it mostly combines basic HTML and JavaScript concepts with some beneficial additions. Still, as is the case with other tools and frameworks, you have to spend some time to get a proper understanding of React's library.
- It can be used for the development of both web and mobile apps: We already know that React is used for the development of web applications, but that's not all it can do. There is a framework called React Native, derived from React itself, that is hugely popular and is used for creating beautiful mobile applications. So, in reality, React can be used for making both web and mobile applications.
- Dedicated tools for easy debugging: Facebook has released a Chrome extension that can be used to debug React applications. This makes the process of debugging React web applications faster and easier.
- The above reasons more than justify the popularity of the React library and why it is being adopted by a large number of organizations and businesses. Now let's familiarize ourselves with React's features.

ReactJS Keys

While dealing with components that are produced periodically in React, keys are essential. Your component will continue to be uniquely identifiable after the modification if the key value is set. They aid React in determining which elements have changed, been eliminated, or been added.

When making lists of components in React, you must use a special string personality factor "key". React uses keys to indicate which list items have been modified, destroyed, or altered. Or, to put it another way, we may say that keys are utilized to identify the components in lists.

ReactJS Advantages

1. React.js builds a customized virtual DOM. Because the JavaScript virtual DOM is quicker than the conventional DOM, this will enhance the performance of apps.
2. ReactJS makes an amazing UI possible.
3. Search - engine friendly ReactJS.
4. Modules and valid data make larger apps easier to manage by increasing readability.
5. React integrates various architectures.
6. React makes the entire scripting environment process simpler.
7. It makes advanced maintenance easier and boosts output.
8. Guarantees quicker rendering
9. The availability of a script for developing mobile apps is the best feature of React.
10. ReactJS is supported by a large community.

1. App.js:

- Dependencies:
 - Imports the necessary styles from './App.css'.
 - Imports the Navbar component, and Routs component for handling routes.
 - Imports the BrowserRouter component from react-router-dom for enabling client-side routing.

```
import './App.css';
import Navbar from './components/Navbar';
import { BrowserRouter as Router } from "react-router-dom";
import Routs from './routes/Routs';

function App() {
  return (
    <div className="App">
      <Router>
        <Navbar />
        <Routs />
      </Router>
    </div>
  );
}

export default App;
```

Fig 5.23. App

- **Component Structure:**
 - Contains the main App component, which serves as the entry point of the application.
 - Uses the Router to wrap the entire app, providing navigation capabilities.
 - Renders the Navbar and Routs components within the Router.

2. Navbar.js:

- **Dependencies:**
 - Imports styles from './styles/Navbar.module.css'.
 - Imports the Link component from react-router-dom for creating navigation links.
 - Imports the logo image from './assets/logo/1.png'.
- **Component Structure:**
 - Defines the Navbar component, containing a logo and navigation links.
 - Maps through navOptions array to dynamically generate navigation links using the Link component.

```
const navOptions = [  
  {name: 'Home', link: '/'},  
  {name: 'Get started', link: '/get-started'},  
  {name: 'Contact', link: '/team'}  
];
```

Fig 5.24. Navbar

- Utilizes CSS modules for styling to avoid class name conflicts.

3. Routs.js:

- **Dependencies:**
 - Imports Route and Routes from react-router-dom.
 - Imports individual components (HomePage, CheckerPage, Members) to be rendered for specific routes.

```

import { Route, Routes } from "react-router-dom";
import HomePage from "../components/HomePage";
import CheckerPage from "../components/CheckerPage";
import Members from "../components/Members";

const Routs = () => {
  return (
    <Routes>
      <Route path="/" element={ <HomePage /> } />
      <Route path="/get-started" element={ <CheckerPage /> } />
      <Route path="/team" element={ <Members /> } />
    </Routes>
  );
}
export default Routs;

```

Fig 5.25. Routes

- Component Structure:
 - Defines the Routs component to handle the app's routes.
 - Uses the Routes component to specify different routes and their corresponding components.

4. HomePage.js:

- Dependencies:
 - Imports styles from './styles/Homepage.module.css'.
 - Imports the Link component for creating navigation links.

```

import React from 'react';
import classes from './styles/Homepage.module.css';
import { Link } from 'react-router-dom';

const HomePage = () => {
  return (
    <div className={classes.pageContainer}>
      <div className={classes.container}>
        <div className={classes['inner-container']}>
          <h3 className={classes.quote}>Its Time to Prevent TB to End TB!!</h3>
          <Link to="/get-started">
            <button className={classes.primary}>
              Get Started
            </button>
          </Link>
        </div>
      </div>
    </div>
  );
}

export default HomePage;

```

Fig 5.26. Home Page

- **Component Structure:**
 - Defines the HomePage component with a quote and a "Get Started" button.
 - Utilizes the Link component to navigate to the '/get-started' route when the button is clicked.

5. CheckerPage.js:

- **Dependencies:**
 - Imports styles from './styles/Homepage.module.css'.
 - Imports the Link component for creating navigation links.
- **Component Structure:**
 - Defines the CheckerPage component with a heading and a "Reach Us" button.
 - Utilizes the Link component to create a link for contacting via email when the button is clicked.

6. Members.js:

- **Dependencies:**
 - Imports styles from './styles/Members.css'.

```
import React from 'react';
import '../components/styles/Members.css';

const Members = () => {
  return (
    <div>
      <div className="become-member-box">
        <h1>Contact Us</h1>
        <div className="become-member-content">
          <p>Please feel free to contact us with any questions or concerns.</p>
          <a href="mailto:tbai@gmail.com" className="join-button">Reach Us</a>
        </div>
      </div>
    </div>
  );
};

export default Members;
```

Fig 5.27. Contact

- **Component Structure:**
 - Defines the Members component, which provides information and a "Reach Us" button for contacting via email

5.2 TESTING

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification

1.SOFTWARE VALIDATION

Validation is process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated.

Validation ensures the product under development is as per the user requirements. Validation emphasizes on user requirements.

2.SOFTWARE VERIFICATION

Verification is the process of confirming if the software is meeting the business requirements and is developed adhering to the proper specifications and methodologies. Verification ensures the product being developed is according to design specifications.

Verification answers the question– "Are we developing this product by firmly following Verifications concentrate on the design and system specifications.

3.TARGET OF THE TEST ARE

- Errors -These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output, considered as an error.
- Fault - When error exists fault occurs. A fault, also known as a bug, is a result of an error which can cause system to fail.
- Failure - failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

1.BLACK-BOX TESTING

It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise.

Black-Box Testing Techniques

- Equivalence class - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- Boundary values - The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.

- Cause-effect graphing - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
- Pair-wise Testing - The behavior of software depends on multiple parameters. In pair wise testing, the multiple parameters are tested pair-wise for their different values.
- State-based testing - The system changes state on provision of input. These systems are tested based on their states and input.

WHITE-BOX TESTING

It is conducted to test program and its implementation, in order to improve code efficiency or structure. It is also known as ‘Structural’ testing

In this testing method, the design and structure of the code are known to the tester. The following are some White box testing techniques

- Control-flow testing - The purpose of the control-flow testing to set up a test case which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- Data-flow testing - This testing technique emphasis to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

TESTING LEVELS

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated and verified. Testing separately is done just to make sure that there are no hidden bugs or issues left in the software.

Unit Testing

Unit testing is a crucial aspect of software development, allowing programmers to ensure the functionality and reliability of individual units of code. Under the white-box testing approach, developers have visibility into the internal logic and structure of the code, enabling them to design tests that thoroughly exercise all paths and conditions within the unit.

These tests are crafted to verify that each unit behaves as expected, adhering to the specified requirements and producing the correct output for various inputs. By meticulously testing each unit in isolation, developers can identify and rectify errors early in the development process, preventing potential issues from propagating to higher levels of integration.

Unit testing provides several benefits to developers and the overall software development lifecycle. It helps in detecting defects early, reducing the cost and effort

required for debugging and maintenance later on. Additionally, unit tests serve as documentation for the code, clarifying its intended behavior and facilitating future modifications or enhancements.

Moreover, unit testing fosters confidence in the codebase, allowing developers to make changes confidently, knowing that regressions can be quickly identified and addressed. This iterative approach to testing promotes code quality, enhances reliability, and ultimately contributes to the delivery of robust, error-free software systems.

UNIT	PURPOSE	STATUS
React App	When user uploads the image and clicks submit button, API accepts and sends the response.	Success
React App	When user uploads the image and clicks submit button. Then the App shows image processing error.	Fail
API	When user uploads the image and clicks submit button, API doesn't accept the incoming requests.	Fail
Application	When users clicks submit without selecting x-ray.	Fail
API	When user uploads the x-ray and clicks submit, API accepts the request but fails to send response.	Partial Success/Partial Fail
Application	When user tries to upload the x-ray but it doesn't due to low latency.	Fail
React App	When user uploads the x-ray and clicks submit button. Timeout error for response.	Partial Success/Partial Fail

Table: Summary of Unit tests

Integration Testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passes and data updating etc.

System Testing

The software is compiled as product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- **Functionality testing** - Tests all functionalities of the software against requirement.
- **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.
- **Security & Portability** - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

Acceptance Testing

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it or works, it may be rejected.

- **Alpha testing** - The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.
- **Beta testing** - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not as yet the delivered product. Developers expect that users at this stage will bring minute 65 problems, which were skipped to attend.

Regression Testing

Whenever a software product is updated with new code, feature or functionality, it is tested thoroughly to detect if there is any negative impact of the added code.

End-to-End Testing

End-to-End testing is a type of Software testing that not only validates the software system under test but also check its integration with external interfaces. It uses actual production like data and test environment to simulate real-time settings. End-to-End testing is also called Chain testing. End-to-End design framework consists of three parts: Build User functions, Build conditions, Build test cases.

5.3. SCREENSHOTS

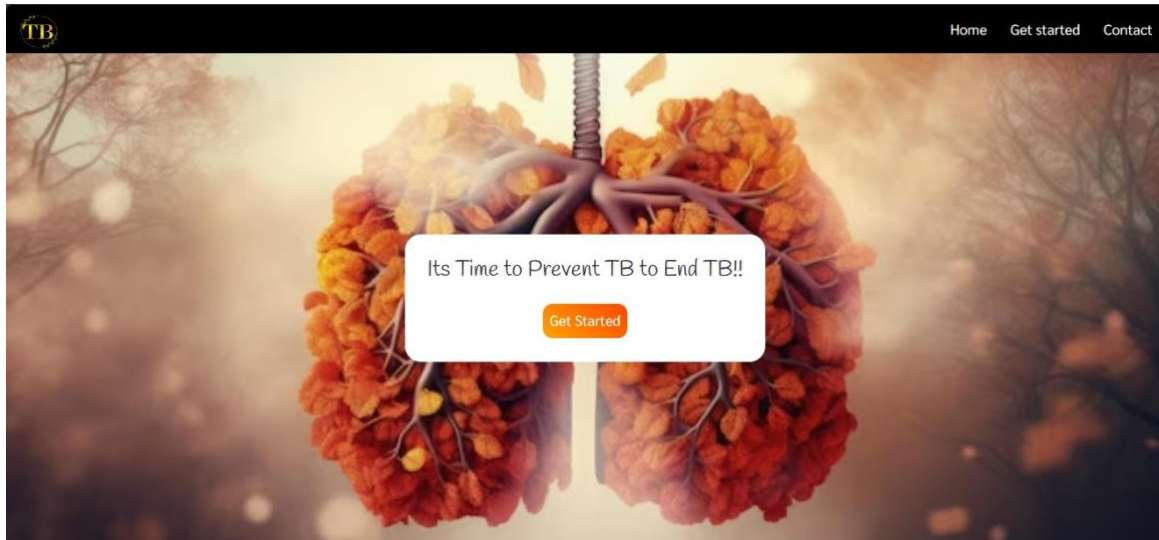


Fig 5.28. Home

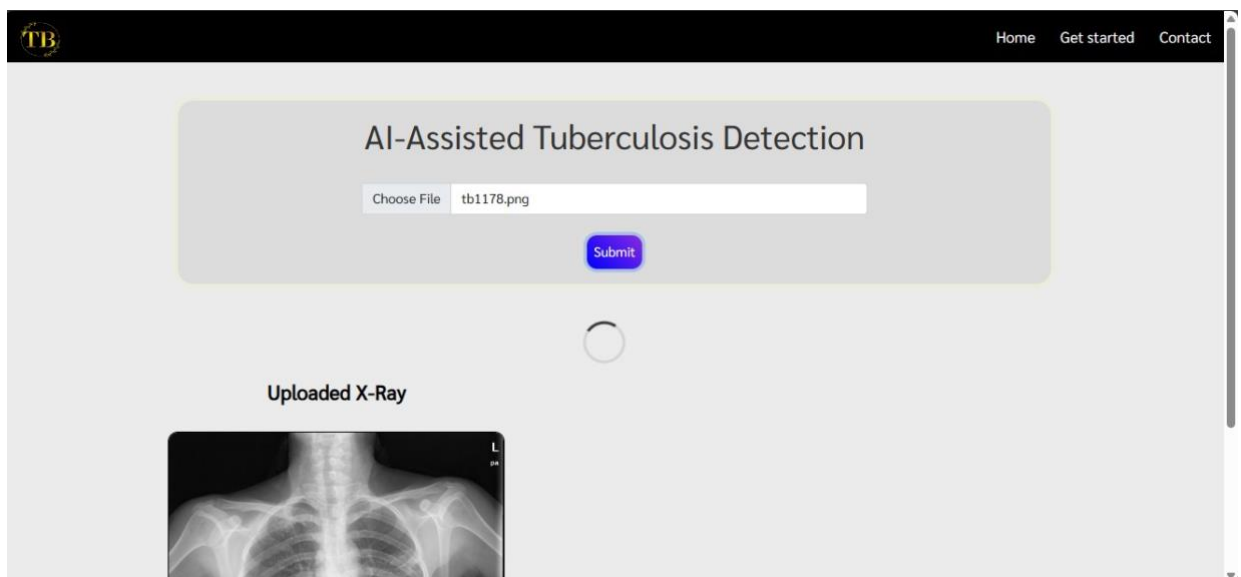


Fig 5.29. Get start

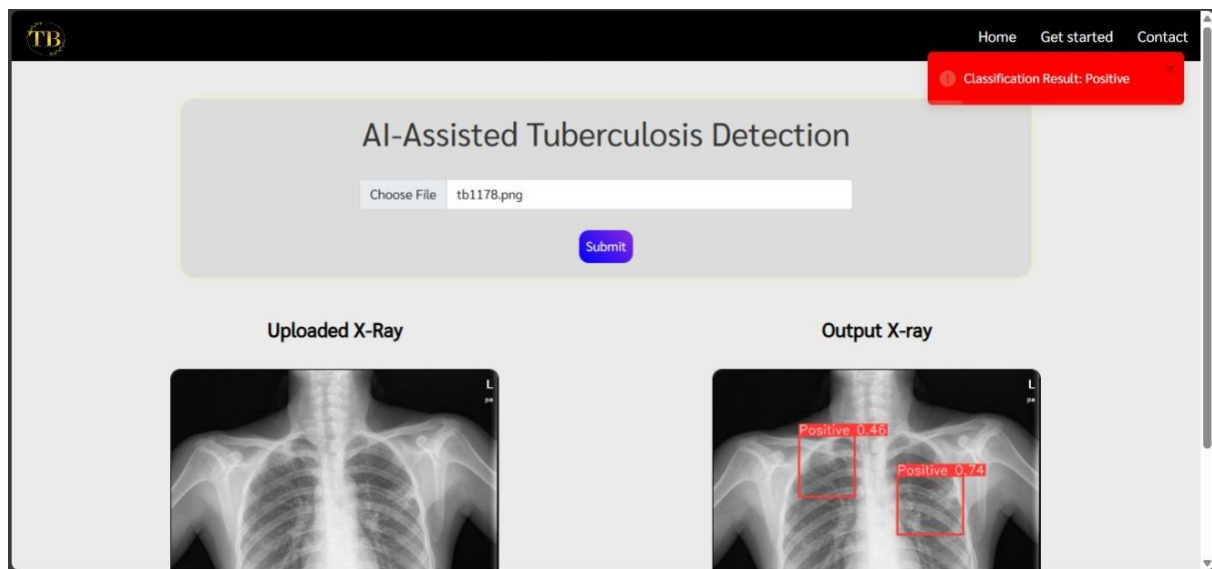


Fig 5.30. Positive

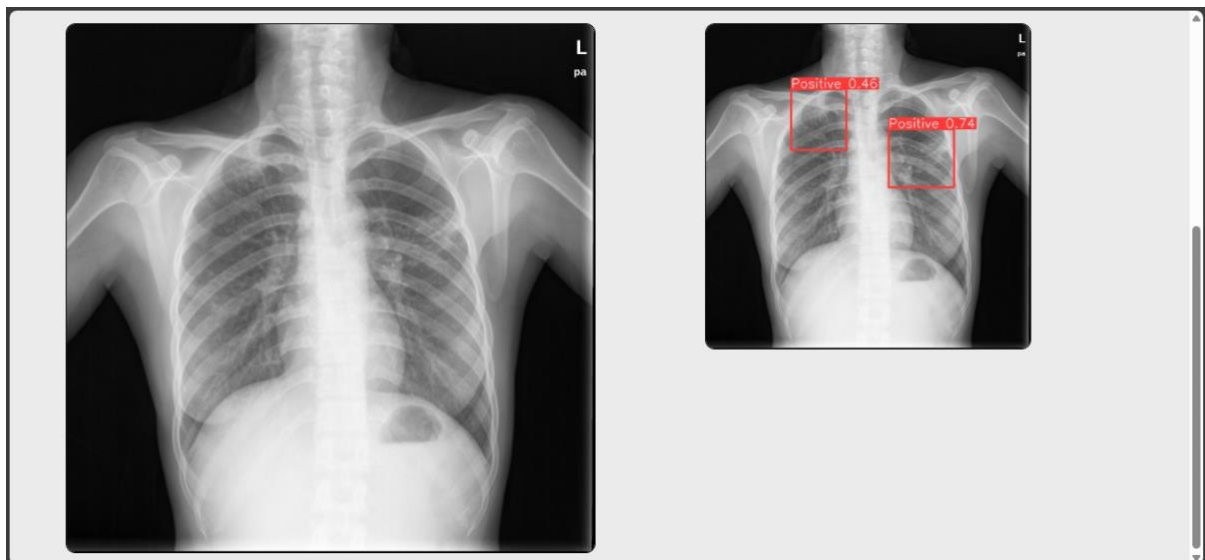


Fig 5.31. Detection Box

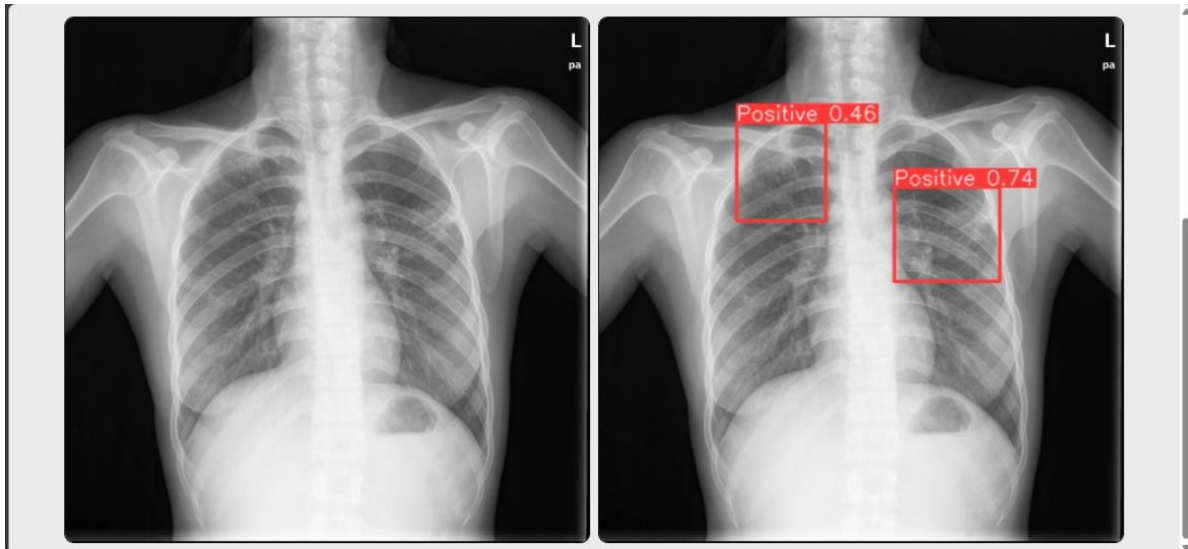


Fig 5.32. Enlarged X-rays

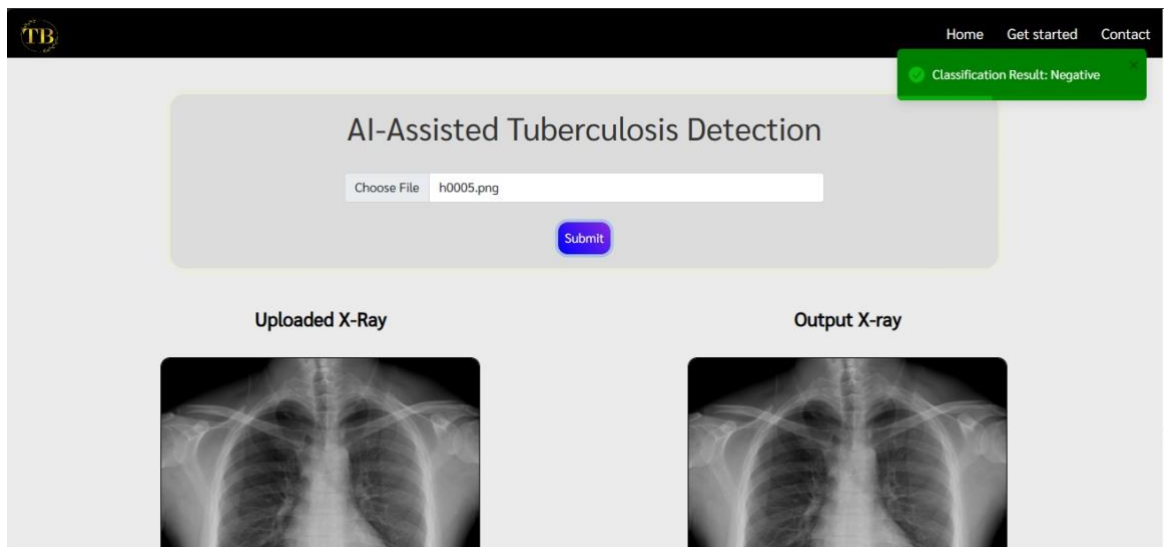


Fig 5.33. Negative

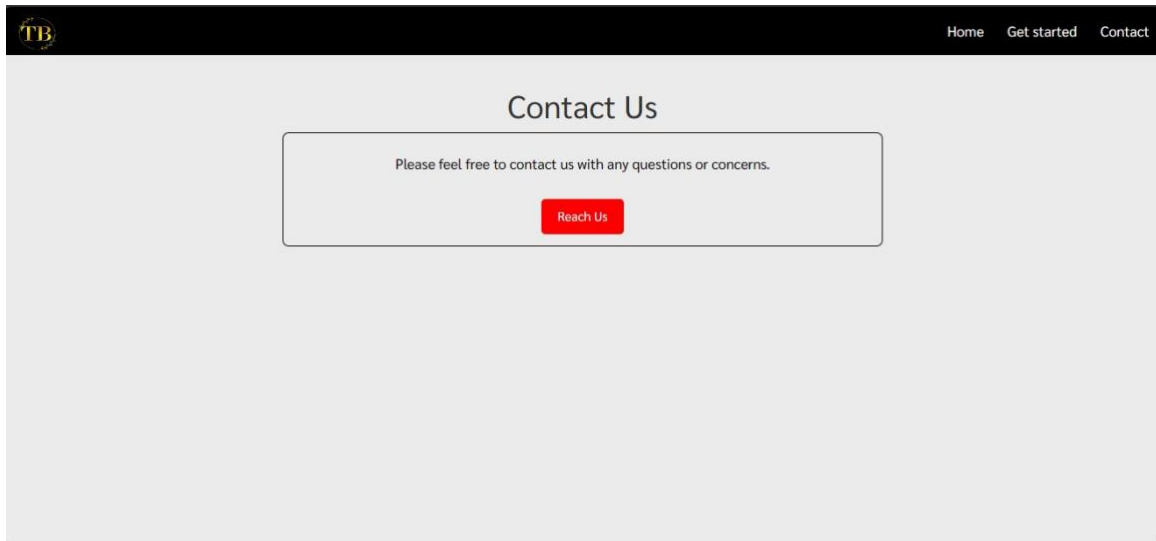


Fig 5.34. Contact

```
PS C:\Users\vikas\mig\fastapi> python api.py
INFO: Started server process [11652]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)

image 1/1 C:\Users\vikas\mig\fastapi\tb1178.png: 512x512 2 Positives, 584.0ms
Speed: 20.8ms preprocess, 584.0ms inference, 15.6ms postprocess per image at shape (1, 3, 512, 512)
Results saved to C:\Users\vikas\runs\detect\predict19
INFO: 127.0.0.1:50629 - "POST /tbdetection/ HTTP/1.1" 200 OK

image 1/1 C:\Users\vikas\mig\fastapi\tb1178.png: 512x512 2 Positives, 873.2ms
Speed: 0.0ms preprocess, 873.2ms inference, 0.0ms postprocess per image at shape (1, 3, 512, 512)
Results saved to C:\Users\vikas\runs\detect\predict19
negative
INFO: 127.0.0.1:50630 - "POST /tbdetection/class HTTP/1.1" 200 OK

image 1/1 C:\Users\vikas\mig\fastapi\h0005.png: 512x512 (no detections), 1216.6ms
Speed: 0.7ms preprocess, 1216.6ms inference, 3.1ms postprocess per image at shape (1, 3, 512, 512)
Results saved to C:\Users\vikas\runs\detect\predict19
INFO: 127.0.0.1:50662 - "POST /tbdetection/ HTTP/1.1" 200 OK

image 1/1 C:\Users\vikas\mig\fastapi\h0005.png: 512x512 (no detections), 1188.3ms
Speed: 11.5ms preprocess, 1188.3ms inference, 0.0ms postprocess per image at shape (1, 3, 512, 512)
Results saved to C:\Users\vikas\runs\detect\predict19
positive
INFO: 127.0.0.1:50663 - "POST /tbdetection/class HTTP/1.1" 200 OK
```

Fig 5.35. Log

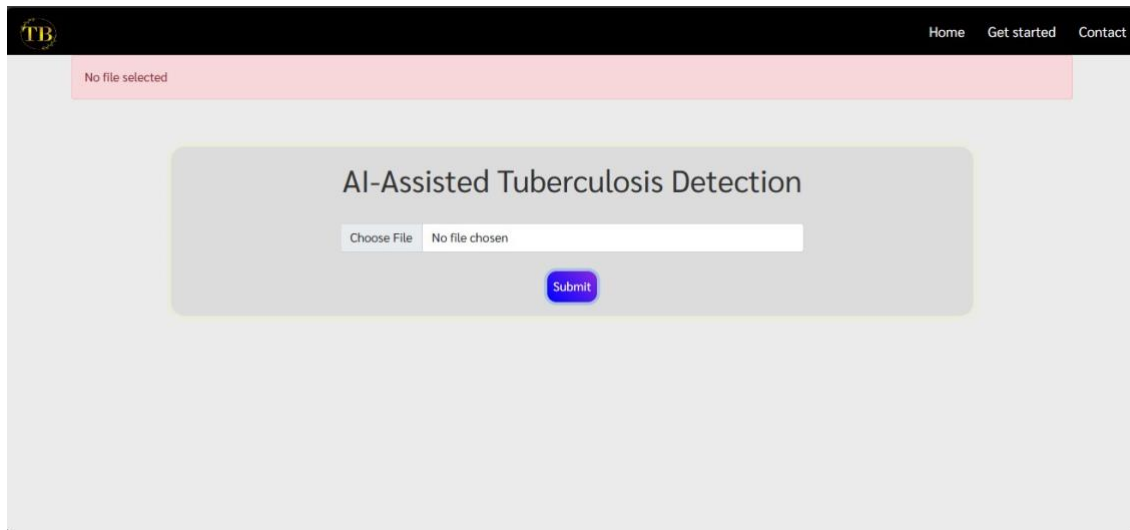


Fig 5.36. No file Error

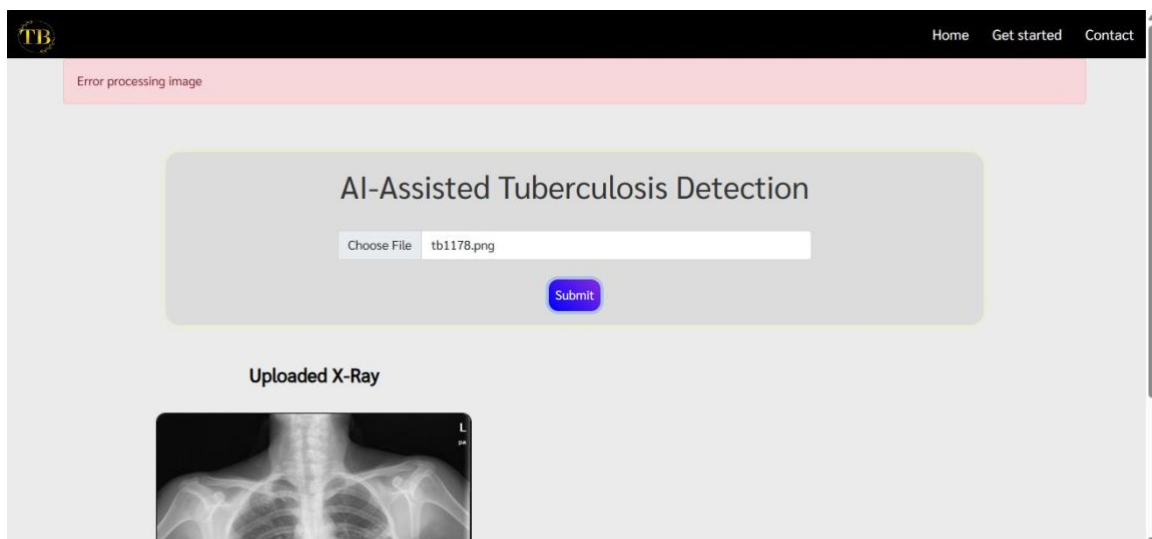


Fig 5.37. Processing Image Error

CHAPTER-6

6. SYSTEM DESIGN

Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from a distinctly different perspective.

Each view is defined by a set of diagram, which is as follows:

1. **User Model View:**
This view represents the system from the users' perspective. The analysis representation describes a usage scenario from the end-users' perspective.
2. **Structural Model View:**
In this model, the data and functionality arrive from inside the system. This model view models the static structures.
3. **Behavioural Model View:**
It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.
4. **Implementation Model View:**
In this view, the structural and behavioral as parts of the system are represented as they are to be built.
5. **Environmental Model View:**
In this view, the structural and behavioral aspects of the environment in which the systems to be implemented are represented.

6.1 UML DIAGRAMS

USE CASE DIAGRAM

To model a system, the most important aspect is to capture the dynamic behavior. To clarify a bit in details, dynamic behavior means the behavior of the system when it is running/operating.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and use case diagrams is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams are consisting of actors, use cases and their relationships. The diagram is used to model the system/ subsystem of an application. A single use case diagram captures a particular functionality of a system. So, to model the entire system numbers of use case diagrams

are used.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

In brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Showing the interacting among the requirements are actors.

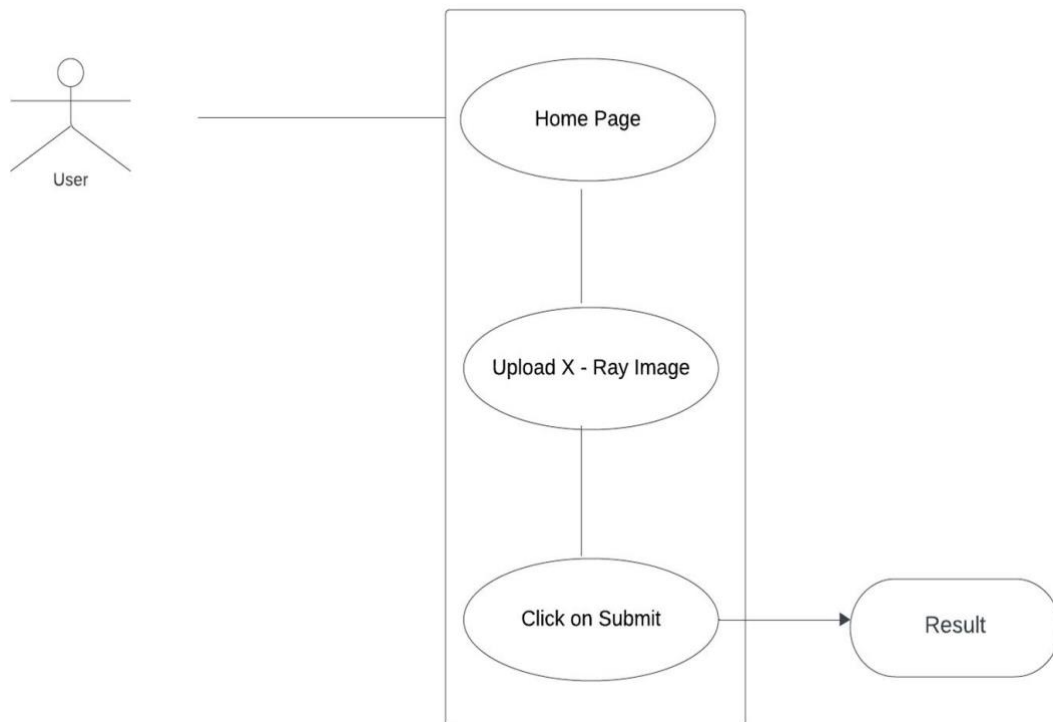


Figure 6.1.1 - Use Case Diagram

SEQUENCE DIAGRAM

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.

The aim of a sequence diagram is to define event sequences, which would have a desired outcome. The focus is more on the order in which messages occur than on the message per se. However, the majority of sequence diagrams will communicate what messages are sent and the order in which they tend to occur. Basic Sequence Diagram Notations

Class Roles or Participants:

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Activation or Execution Occurrence:

Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin grey rectangle placed vertically on its lifeline.

Messages:

Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

Lifelines:

Lifelines are vertical dashed lines that indicate the object's presence over time.

Destroying Objects:

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an

X. This object is removed from memory. When that object's lifeline ends, you can place an X at the end of its lifeline to denote a destruction occurrence.

Loops:

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets []. When modeling object interactions, there will be times when a condition must be met for a message to be sent to an object. Guards are conditions that need to be used throughout UML diagrams to control flow.

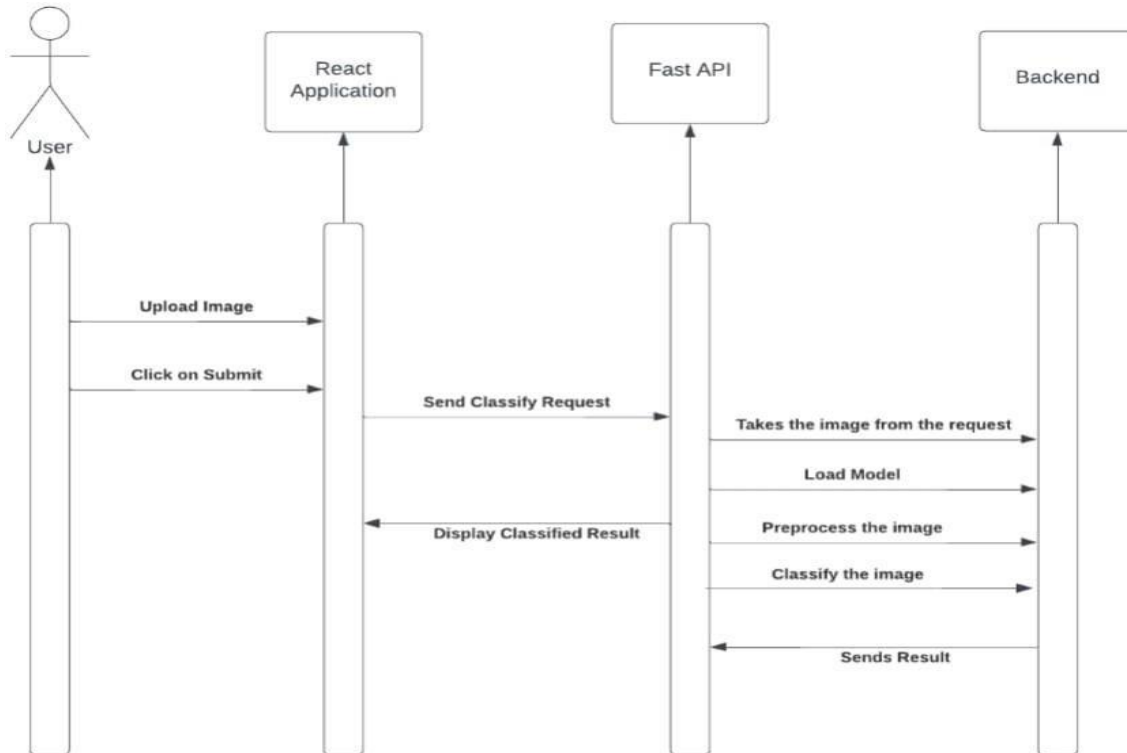


Figure 6.1.2 - Sequence Diagram

ACTIVITY DIAGRAM

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another,

in particular, use cases where activities may overlap and require coordination. It is also suitable for modeling how a collection of use cases coordinates to represent business workflows.

- Identify candidate use cases, through the examination of business workflows
- Identify pre- and post-conditions (the context) for use cases
- Model workflows between/within use cases
- Model complex workflows in operations on objects

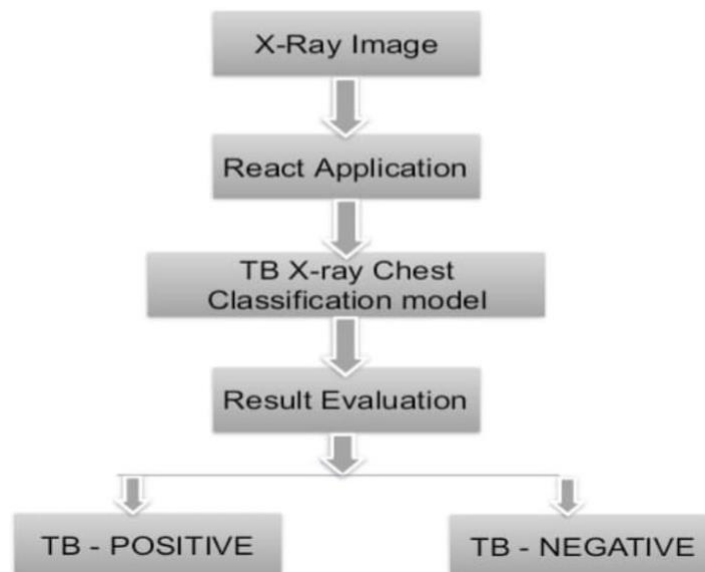


Figure 6.1.3 - Activity Diagram

CLASS DIAGRAM

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. Class diagram 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. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Purpose of Class Diagrams:

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

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however the class diagram is a bit different. It is the most popular UML diagram in the coder community.

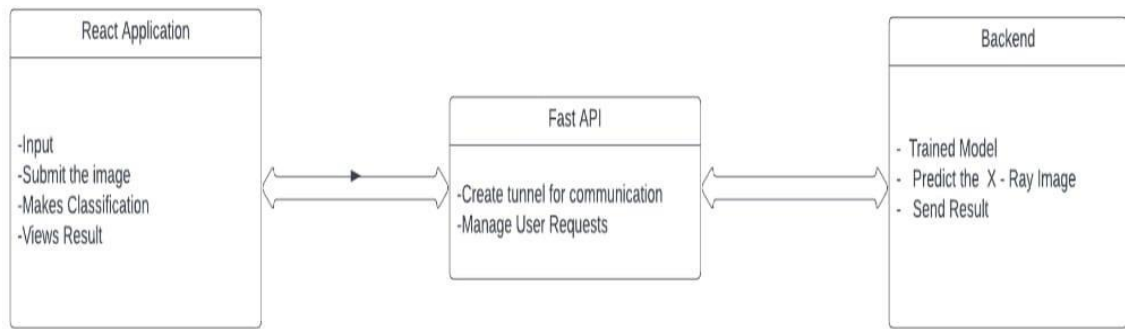


Figure 6.1.4 - Class Diagram

CHAPTER-7

7. CONCLUSION

The progress achieved in Tuberculosis (TB) classification using X-ray images represents a significant leap forward in healthcare technology and public health efforts. Through the development of deep learning models, meticulous data annotation, and seamless integration with healthcare systems, these advancements have not only enhanced diagnostic accuracy but also facilitated early detection and treatment initiation. Moreover, the deployment of real-time classification systems on edge devices and cloud platforms has enabled remote diagnosis, particularly beneficial for underserved areas. This comprehensive approach, coupled with rigorous validation, educational initiatives, and impactful research contributions, underscores the collaborative efforts of multidisciplinary teams in revolutionizing TB diagnosis and healthcare delivery.

Technological advancements in TB classification systems have ushered in a new era of healthcare innovation, with tangible applications across various domains. From early detection and remote diagnosis to population screening and clinical decision support, these advancements have transformed the landscape of TB management and control. Moreover, by supporting public health interventions, epidemiological studies, and capacity-building initiatives, TB classification technologies not only contribute to improved patient outcomes but also drive cross-disciplinary collaboration and innovation in healthcare. As such, they represent a pivotal step forward in the global effort to combat tuberculosis and advance healthcare equity and innovation.

The strides made in Tuberculosis (TB) classification through the utilization of X-ray imaging and deep learning represent a watershed moment in healthcare technology. By harnessing the capabilities of advanced algorithms and meticulous data annotation, these innovations have significantly elevated the precision and efficiency of TB diagnosis. Moreover, their seamless integration into healthcare systems has facilitated prompt identification and treatment initiation, thus bolstering efforts to curb the spread of this infectious disease. The deployment of real-time classification systems on diverse platforms, from edge devices to cloud networks, has extended the reach of TB diagnosis, particularly benefiting marginalized communities with limited access to healthcare resources. This comprehensive approach, marked by collaborative endeavors across disciplines, underscores the transformative potential of technology in revolutionizing TB management and advancing global health equity.

FUTURE ENHANCEMENTS

While significant progress has been made in TB classification using X-ray images, there are several avenues for future work and recommendations to further enhance the efficacy, accessibility, and impact of these technologies. Here are some key areas to focus on:

1. Enhanced Model Performance:

- Continuously improve the performance of TB classification models by exploring advanced deep learning architectures, optimization algorithms, and ensemble techniques.
- Investigate the integration of multimodal data, such as clinical metadata and laboratory results, to enhance the discriminative power of the models.

2. Addressing Data Imbalance:

- Develop strategies to mitigate data imbalance issues, particularly in datasets with skewed class distributions, to ensure robust performance across diverse patient populations.

3. Federated Learning and Privacy Preservation:

- Explore federated learning approaches to train TB classification models collaboratively across multiple healthcare institutions while preserving patient privacy and data confidentiality.
- Investigate the integration of privacy-preserving techniques, such as secure aggregation and differential privacy, to enhance the security and confidentiality of patient data during model training and inference.

4. Explainable AI (XAI):

- Develop interpretable and explainable AI methods to provide insights into the decision-making process of TB classification models, enhancing transparency, trust, and clinical acceptance.
- Investigate techniques such as attention mechanisms, saliency maps, and feature importance analysis to elucidate the rationale behind model predictions and facilitate clinical validation.

5. Robust Quality Control Measures:

- Implement robust quality control measures to ensure the reliability, consistency, and safety of TB classification systems in real-world clinical settings.
- Incorporate automated quality checks, human-in-the-loop validation mechanisms, and regular performance monitoring to detect and mitigate errors and anomalies.

6. Domain Adaptation and Transfer Learning:

- Investigate domain adaptation techniques to improve the generalization of TB classification models across diverse patient populations, imaging protocols, and clinical settings.
- Explore transfer learning approaches to leverage pre-trained models and domain-specific knowledge to accelerate model convergence and enhance performance in resource-constrained environments.

7. Unsupervised Learning and Anomaly Detection:

- Explore unsupervised learning approaches for anomaly detection in TB classification, enabling proactive identification of abnormal patterns indicative of TB infection or other lung pathologies.
- Investigate clustering, autoencoders, and anomaly detection algorithms to identify outliers and deviations from normal X-ray patterns, flagging suspicious cases for further evaluation.

8. User-Friendly Interfaces:

- Develop intuitive and user-friendly interfaces for TB classification systems, incorporating interactive visualization tools, customizable preferences, and context-sensitive help features.
- Conduct usability testing and user feedback sessions to ensure that the interfaces meet the needs, expectations, and workflow constraints of healthcare professionals in diverse clinical settings.

9. Integration of Feedback Mechanisms:

- Establish feedback mechanisms to collect user feedback, bug reports, and feature requests for continuous improvement of TB classification systems.
- Integrate feedback loops into the classification pipeline to automatically aggregate and analyze user feedback, prioritizing updates and enhancements based on user needs and preferences.

10. Ethical Considerations and Algorithm Bias:

- Address ethical considerations such as algorithm bias, fairness, transparency, and accountability to ensure equitable access to TB classification technologies and mitigate potential harms.
- Implement bias detection, mitigation, and fairness-aware training techniques to promote unbiased and equitable outcomes in TB diagnosis and patient care.

11. Collaboration with Healthcare Stakeholders:

- Collaborate with healthcare providers, policymakers, regulatory authorities, and patient advocacy groups to advocate for the adoption, regulation, and reimbursement of TB classification technologies.
- Engage in transparent communication, stakeholder engagement, and evidence-based advocacy efforts to build trust, promote acceptance, and foster sustainable adoption of AI-driven innovations in healthcare.

12. Longitudinal Studies and Outcome Evaluation:

- Conduct longitudinal studies and outcome evaluations to assess the long-term impact of TB classification technologies on patient outcomes, healthcare delivery, and public health outcomes.
- Monitor key performance indicators, such as diagnostic accuracy, treatment initiation rates, and patient outcomes, to evaluate the effectiveness and sustainability of TB classification systems in real-world clinical practice.

REFERENCES

- [1] A. Sharma and P. K. Mishra, “Performance analysis of machine learning based optimized feature selection approaches for breast cancer diagnosis,” *International Journal of Information Technology*, vol. 14, no. 4, pp. 1949–1960, 2022.
- [2] S. Abdul Gafoor, N. Sampathila, S. Ks, and S. K S, “Deep learning model for detection of covid-19 utilizing the chest x-ray images,” *Cogent Engineering*, vol. 9, no. 1, Article ID 2079221, 2022.
- [3] K. Chadaga, S. Prabhu, K. V. Bhat, S. Umakanth, and N. Sampathila, “Medical diagnosis of covid-19 using blood tests and machine learning,” *Journal of Physics: Conference Series*, IOP Publishing, vol. 2161, Article ID 012017, 2022.
- [4] C. Yan, L. Wang, J. Lin et al., “A fully automatic artificial intelligence–based ct image analysis system for accurate detection, diagnosis, and quantitative severity evaluation of pulmonary tuberculosis,” *European Radiology*, vol. 32, no. 4, pp. 2188–2199, 2022.
- [5] Y. Zaizen, Y. Kanahori, S. Ishijima et al., “Deep-learning-aided detection of mycobacteria in pathology specimens increases the sensitivity in early diagnosis of pulmonary tuberculosis compared with bacteriology tests,” *Diagnostics*, vol. 12, no. 3, p. 709, 2022.
- [6] P. Podder, S. Bharati, M. Mondal, and A. Khamparia, “Rethinking the transfer learning architecture for respiratory diseases and covid-19 diagnosis,” *Biomedical Data Analysis and Processing Using Explainable (XAI) and Responsive Artificial Intelligence (RAI)*, Springer, Berlin, Germany, pp. 105–121, 2022.
- [7] M. R. H. Mondal, S. Bharati, and P. Podder, “Co-irv2: optimized inceptionresnetv2 for covid-19 detection from chest ct images,” *PLoS One*, vol. 16, no. 10, Article ID e0259179, 2021.
- [8] M. R. H. Mondal, S. Bharati, and P. Podder, “Diagnosis of covid-19 using machine learning and deep learning: a review,” *Current Medical Imaging Formerly Current Medical Imaging Reviews*, vol. 17, no. 12, pp. 1403–1418, 2021.
- [9] S. Bharati, P. Podder, M. R. H. Mondal, and V. S. Prasath, “Co-resnet: optimized resnet model for covid-19 diagnosis from x-ray images,” *International Journal of Hybrid Intelligent Systems*, vol. 17, no. 1-2, pp. 71–85, 2021.

- [10] A. Sharma and P. K. Mishra, “Deep learning approaches for automated diagnosis of covid-19 using imbalanced training cxr data,” in Proceedings of the International Conference on Advanced Network Technologies and Intelligent Computing, pp. 453–472, Springer, Varanasi, India, December 2021.
- [11] B. P. Health, “Belarus tuberculosis portal,” 2021
- [12] A. Brock, S. De, S. L. Smith, and K. Simonyan, “High-performance large-scale image recognition without normalization,” 2021
- [13] T. Chen, Z. Zhang, X. Ouyang, Z. Liu, Z. Shen, and Z. Wang, “Training binary neural networks without batch normalization,” 2021
- [14] A. Brock, S. De, and S. L. Smith, “Characterizing signal propagation to close the performance gap in unnormalized resnets, CoRR abs/2101,” Article ID 08692, 2021
- [15] Niaid, “TB portal program dataset,” Article ID 45502, 2021
- [16] T. Rahman, A. Khandakar, M. A. Kadir et al., “Reliable tuberculosis detection using chest x-ray with deep learning, segmentation and visualization,” IEEE Access, vol. 8, pp. 191586–191601, 2020.
- [17] S. S. Han, I. J. Moon, W. Lim et al., “Keratinocytic skin cancer detection on the face using region-based convolutional neural network,” JAMA Dermatology, vol. 156, no. 1, pp. 29–37, 2020, arXiv:.
- [18] M. Yang, “Painful conversations: therapeutic chatbots and public capacities,” Communication and the Public, vol. 5, no. 1-2, pp. 35–44, 2020.
- [19] M. A. Noyan, M. Durdu, and A. H. Eskiocak, “TzanckNet: a convolutional neural network to identify cells in the cytology of erosive-vesiculobullous diseases,” Scientific Reports, vol. 10, no. 1, Article ID 18314, 2020.
- [20] S. Sathitrataneewin, P. Sunanta, and K. Pongpirul, “Deep learning for automated classification of tuberculosis-related chest x-ray: dataset distribution shift limits diagnostic performance generalizability,” Heliyon, vol. 6, no. 8, Article ID e04614, 2020.