MACHINE LEARNING SYSTEM TO PREDICT TUBERCULOSIS IN CHEST X-RAY IMAGES

BY

ADEBISI OMOIKHEFE VICTOR

JULY 2024

CONTENTS

LIST OF FIGURES

ABSTRACT

Tuberculosis (TB) remains a significant global health burden, claiming millions of lives annually. Early and accurate diagnosis is critical for effective treatment and preventing disease transmission. While chest X-rays (CXRs) are widely used as a screening tool for TB, their diagnostic accuracy can be limited by factors such as inter-observer variability and the expertise of the radiologist. This study investigates the potential of deep learning to address these challenges and improve TB detection from CXR images. By leveraging large-scale CXR datasets and advanced deep learning architectures, this research aims to develop a highly accurate and efficient model capable of identifying TB-related abnormalities. Successful implementation of such a model could significantly enhance TB diagnosis, particularly in resource-limited settings where access to expert radiologists may be scarce. However, addressing challenges such as data quality, model interpretability, and ethical considerations is essential for the responsible and effective deployment of deep learning technology in clinical practice. Ultimately, this research seeks to contribute to the development of innovative tools for TB control and elimination.

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Tuberculosis (TB) is a major cause of morbidity and mortality worldwide. It is estimated that 25% of the world's population are infected with Mycobacterium tuberculosis, with a 5–10% lifetime risk of progression into TB disease (Gill *et al.*, 2022). Early recognition of TB disease and prompt detection of drug resistance are essential to halting its global burden. Culture, direct microscopy, biomolecular tests and whole genome sequencing are approved methods of diagnosis; however, their widespread use is often curtailed owing to costs, local resources, time constraints and operator efficiency. Methods of optimizing these diagnostics, in addition to developing novel techniques, are under review (Gill *et al.*, 2022).

A cornerstone of radiological imaging for many decades, chest radiography (chest X-ray, CXR) remains the most commonly performed radiological exam in the world with industrialized countries reporting an average 238 erect-view chest X-ray images acquired per 1000 of population annually (Çallı *et al.*, 2021). Chest X-rays may be divided into three principal types, according to the position and orientation of the patient relative to the Xray source and detector panel: posteroanterior, anteroposterior, lateral. The posteroanterior (PA) and anteroposterior (AP) views are both considered as frontal, with the X-ray source positioned to the rear or front of the patient respectively (Çallı *et al.*, 2021). In recent years, AI solutions have shown to be capable of assisting radiologists and clinicians in detecting diseases, assessing severity, automatically localizing and quantifying disease features, or providing an automated assessment of disease prognosis. AI for MI has received extraordinary attention in 2020, as attested by a multitude of interdisciplinary projects attempting to blend AI technologies with knowledge from MI in order to combat COVID-19. A keyword search combining AI and MI revealed 2,563 papers in 2019, while 2020 has seen more than twice the number of such papers (5,401, cf. *Figure* 2).

Machine learning is a branch of computing that studies the design of algorithms with the ability to "learn." A subfield would be deep learning, which is a series of techniques that make use of deep artificial neural networks, that is, with more than one hidden layer, to computationally imitate the structure and functioning of the human organ and related diseases (Alsaffar *et al.*, 2021). Radiologists play a crucial role in interpreting medical images for the diagnosis and prognosis of disease. Although AI technologies have recently demonstrated performance that matches radiologists' accuracy in a number of specific tasks, it remains unclear whether radiologists who adopt AI assistance will replace those who do not. As (Celi *et al.*) put it in 2019, ''the question is not whether computers can outperform human in specific tasks, but how humanity will embrace and adopt these capabilities into the practice of medicine.'' (Born *et al.*, 2021). In recent years, deep learning has become the technique of choice for image analysis tasks and made a tremendous impact in the field of medical imaging (Litjens *et al.*, 2017). Deep learning is notoriously data-hungry and the CXR research community has benefited from the publication of numerous large labeled databases in recent years, predominantly enabled by the generation of labels through automatic parsing of radiology reports. This trend began in 2017 with the release of 112,000 images from the NIH clinical center (Wang *et al.*, 2017b). In 2019 alone, more than 755,000 images were released in 3 labelled databases (CheXpert (Irvin *et al.*, 2019), MIMIC-CXR (Johnson *et al.*, 2019), PadChest (Bustos *et al.*, 2020)). The integration of deep learning models into the diagnostic process of TB from chest X-rays can also address the variability and subjectivity inherent in human interpretation. Studies have shown that deep learning algorithms can match or even surpass the performance of human radiologists in certain diagnostic tasks, providing consistent and reproducible results (Esteva *et al.*, 2017). Despite the potential benefits, the implementation of deep learning models in TB diagnosis from chest X-rays faces several challenges. One significant issue is the need for large and diverse datasets to train robust models that generalize well across different populations and imaging conditions (Wang *et al.*, 2017). Additionally, there are concerns regarding the interpretability of deep learning models, as these "black box" systems can be difficult to understand and trust by medical professionals. Ensuring the ethical use of these technologies and addressing potential biases in the training data are also crucial considerations (Amann *et al.*, 2020).

In conclusion, the diagnosis of tuberculosis from chest X-rays is a critical area where the application of deep learning models offers substantial promise. By leveraging large datasets and advanced computational techniques, these models have the potential to improve the accuracy, consistency, and accessibility of TB diagnosis. However, addressing the challenges associated with data requirements, model interpretability, and ethical considerations will be essential for the successful integration of these technologies into clinical practice. Continued research and collaboration between technologists and healthcare professionals are necessary to realize the full potential of deep learning in combating the global TB epidemic.

1.2 Statement of the Problem

X-ray imaging although known for being --non-invasive, quick, and painless, able to support medical and surgical treatment planning and Guide medical personnel as they insert catheters or stents inside the body to treat tumors, or remove blood clots. (US FDA, 2023)**.** The traditional methods used in the interpretation of an X-ray image rely on domain knowledge on i.e Measuring CTR(cardio thoracic ratio), counting the number of ribs showing, Costophrenic angles, diaphragm and checking if the lungs are clear. The difficulty with this approach is the time it takes to get back results or interpretation from radiologists (Ajiboye *et al.*, personal communication, 2024).

1.3 Aim and Objectives

To simplify the detection of tuberculosis from chest X-ray images compared to conventional approaches, with the following objectives:

i. Gathering of chest X-ray image data (from, Zenodo.org)

ii. Convolutional neural network machine learning model development with the gathered dataset.

iii. Development of a graphical user interface

1.4 Scope of the Study

The system will be designed to enhance the efficiency and effectiveness of radiographers and radiologists in a laboratory setting. It will streamline workflows by automating image processing and analysis, improve diagnostic accuracy and communication.

1.5 Significance and justification of the Study

The implementation of the TB detection system has the potential to enhance patient outcomes, prevent disease transmission and spread. Overall, the study's findings have the potential to significantly impact TB control efforts and improve public health on a global scale.

1.6 Definition of terms

Tuberculosis (TB): An infectious disease caused by the bacterium Mycobacterium tuberculosis, typically affecting the lungs and characterized by coughing, fever, weight loss, and fatigue.

Artificial Intelligence (AI): The simulation of human intelligence processes by machines, especially computer systems, to perform tasks such as learning, problem-solving, and decision-making.

Machine Learning: A subset of AI that enables computers to learn from data and improve their performance on a task without being explicitly programmed.

Deep Learning: A specialized field of machine learning that utilizes artificial neural networks with multiple layers (deep architectures) to extract high-level features from raw data.

Convolutional Neural Networks (CNN): A class of deep neural networks, most commonly applied to analyzing visual imagery, which uses a variation of multilayer perceptrons designed to recognize patterns in pixel images.

Image Classification: The process of categorizing images into predefined classes or categories based on their visual content.

Computer-Aided Diagnosis (CAD): A technology that assists healthcare professionals in interpreting medical images, such as X-rays, by highlighting areas of interest or providing automated diagnoses.

Radiology: The branch of medicine that deals with the study and interpretation of imaging technologies, such as X-rays, CT scans, and MRI scans, for diagnosing and treating diseases.

X-ray Imaging: A medical imaging technique that uses X-rays to create detailed images of the inside of the body, particularly useful for visualizing bones and detecting abnormalities such as lung infections.

Diagnostic Accuracy: The degree to which a diagnostic test or procedure correctly identifies or excludes a disease or condition.

Sensitivity and Specificity: Performance measures of a diagnostic test; sensitivity measures the proportion of true positive results, while specificity measures the proportion of true negative results.

Feature Extraction: The process of selecting or extracting relevant features from raw data, such as images, to represent the underlying patterns or characteristics of interest. **Model Training:** The process of using labeled data to teach a machine learning model to make predictions or classifications by adjusting its internal parameters.

Model Evaluation: The assessment of a trained machine learning model's performance using metrics such as accuracy, precision, recall, and F1-score.

Transfer Learning: A machine learning technique where a model trained on one task is adapted or fine-tuned to perform another related task, often leading to improved performance with limited training data

CHAPTER TWO

LITERATURE REVIEW

2.1 Review of related terms

Artificial Intelligence: Artificial Intelligence (AI) in Healthcare: AI involves using computers to perform tasks that typically require human intelligence. In healthcare, AI can analyze vast amounts of data, like medical images, to assist in diagnosing diseases more accurately and quickly than traditional methods.

Medical Imaging: Medical imaging refers to techniques used to create visual representations of the inside of the body. Chest X-ray imaging is a common method to view the lungs and can help detect conditions like tuberculosis by revealing abnormalities. **Disease Detection and Diagnosis:** This involves identifying diseases from symptoms and medical tests. Automated detection systems use AI to analyze medical images and provide diagnoses, potentially improving accuracy and speed compared to manual analysis. **Tuberculosis (TB):** TB is an infectious disease that primarily affects the lungs and is caused by the bacterium Mycobacterium tuberculosis. Detecting TB early using imaging techniques like chest X-rays is crucial for effective treatment and preventing its spread. **Image Preprocessing:** Before AI can analyze medical images, they often need to be cleaned and enhanced. This process, called image preprocessing, involves improving image quality by reducing noise and highlighting important features.

Model Development: Creating AI models for medical image analysis involves using neural networks, particularly Convolutional Neural Networks (CNNs), which are designed to recognize patterns in images. Techniques like transfer learning and data augmentation help improve these models.

Evaluation Metrics: To determine how well an AI model performs, we use metrics like sensitivity (how well it identifies true cases), specificity (how well it avoids false alarms), and the ROC curve, which helps visualize performance across different thresholds.

Data Quality and Quantity: AI models need lots of high-quality, labeled data to learn effectively. Ensuring that this data is accurate and sufficiently comprehensive is a significant challenge, as is maintaining patient privacy and data security.

Algorithm Performance: AI models must generalize well to new data without being too tailored to the training data (overfitting) or too simplistic (underfitting). Ensuring robust performance across different populations and real-world scenarios is crucial.

Clinical Integration: For AI tools to be useful, they must fit into existing clinical workflows, be accepted by healthcare professionals, and comply with regulations and ethical standards. This includes ensuring the tools are user-friendly and provide clear, actionable insights.

Health Informatics: This field combines healthcare with information technology, focusing on the management and analysis of health data. Electronic Health Records (EHRs) store patient information digitally, which can be integrated with AI tools for comprehensive care.

Public Health Implications: Effective TB detection using AI can support public health initiatives by enabling large-scale screening and early intervention, crucial for controlling and reducing the spread of tuberculosis in populations.

Technological Advancements: Innovations like cloud computing provide the infrastructure to store and process large medical datasets, while edge computing allows for data processing closer to where it is collected, enhancing speed and efficiency in healthcare applications.

2.2 Related Work

According to (Gill *et al.*, 2022)**,** in the journal New developments in tuberculosis diagnosis and treatment, the research focused on Developing an AI algorithm for TB diagnosis using chest X-rays. It Compared the ResNet, VGG, and AlexNet algorithms for TB diagnosis. The U-Net algorithm was used for lung segmentation before TB classification and the ResNet34, VGG, and AlexNet was used for the classification network. The following results were obtained from the research, ResNet model had highest accuracy, sensitivity, and specificity. AI algorithms showed AUC values above 0.99 in testing set. ResNet algorithm recognized TB regions accurately on chest radiographs. AI models provided accurate TB diagnoses across different subgroups. AI algorithm distinguished TB radiographs from non-TB with high accuracy. The major limitations of the research was that Negative sputum culture results may lead to misdiagnoses. Variability in images from two different hospitals. Study population limited to people aged 15 years.

In conclusion I feel the research was able to meet up with its intended purpose despite its limitations. The research done by (Kik *et al.*, 2022) which was documented in the research article,

“Diagnostic accuracy of chest X-ray interpretation for tuberculosis by three 2 artificial intelligence-based software in a screening use-case: an individual 3 patient meta-analysis of global data” aimed to Assess diagnostic accuracy of three CAD solutions for tuberculosis screening, Evaluate performance of CAD systems against microbiological reference standard and Compare CAD products with expert radiologists for sensitivity and specificity. To do this the process of Collecting digital CXRs and demographic data from 6 source studies and analysis of images with CAD4TB, Lunit Insight CXR TB, and qXR was done. The research highlighted that; three CAD products showed reasonable diagnostic accuracy for microbiologically confirmed TB. CAD products had comparable accuracy to experienced radiologists in TB identification and that the WHO recommends CAD as an alternative to human CXR interpretation for TB screening. Although the study lacks pre-validation for CAD systems and also didn’t take to account that Exclusion of healthy individuals may affect sensitivity and specificity rates, the variability in culture results across different studies. The Impact of certain risk factors on CAD accuracy was not confirmed also. In conclusion AI algorithms for TB diagnosis in resource-limited settings can be explored further and Enhanced for more accurate TB diagnosis in diverse populations.

The research by (Qin *et al.*, 2021), on the topic "Tuberculosis detection from chest x-rays for triaging in a high tuberculosis-burden setting: an evaluation of five artificial intelligence algorithms" focused on Evaluating five commercial AI algorithms for triaging tuberculosis detection and to Present a new analytical framework for selecting AI software by using a large dataset from tuberculosis screening centers in Dhaka and comparing AI algorithms with radiologists and WHO's Target Product Profile. It discovered that AI algorithms outperformed radiologists, reducing Xpert tests by 50%, AI algorithms had better specificity compared to radiologists across classifications and that Performance varied with age, history of tuberculosis, and patient source. Though there's a high chance of Limited generalization to other populations due to training strategies and Variability in performance due to different training strategies and datasets.

(Geric *et al.*, 2023), did a study "The rise of artificial intelligence reading of chest X-rays for enhanced TB diagnosis and elimination" The study aimed to evaluate CAD software for TB detection accuracy in different populations. Assess economic implications of CAD as a triage tool for TB. Tools like CAD software was used for TB detection, challenges in standardization, and validation. CAD refinement needed for non-TB lung diseases and subpopulations. From the study it was discovered that CAD software shows high sensitivity for TB detection in adults. CAD products need careful scrutiny and independent performance validation. CAD needs refinement for detecting non-TB lung diseases and in children. Economic evaluations show CAD as a cost-effective triage tool in TB. Certain limitations were discovered from the study; CAD software needs local data calibration due to accuracy heterogeneity. Performance varies with software versions, age, BMI, and disease types. Limited evidence for CAD accuracy in children and non-TB abnormalities. In conclusion, it enhances TB screening but needs refinement for non-TB diseases detection and Validation for TB screening in children is essential for it.

The research (Application of artificial intelligence in digital chest radiography reading for pulmonary tuberculosis screening) done by (Cao *et al.*, 2020) focused on developing CAD systems for TB screening using deep learning technology and to evaluate AI software platforms for TB-associated abnormalities in CXRs. DCNNs were used for TB detection with high accuracy. Training was done on TB-specific dataset, testing was also done on non-TB dataset. Results showed that CAD systems detect abnormalities in CXRs comparable to radiologists. DL systems outperform human radiologists in detecting TBrelated abnormalities. AI-based CAD systems can be used for TB screening and triage. Challenges included generalizability, overdiagnosis, and dataset distribution shifts. CXR has modest specificity and high inter-reader variability. Shortage of qualified radiologists in TB prevalent settings. High cost of CXR hardware limits availability at primary care level. It can be concluded that AI in CXR reading enhances TB screening, overcoming radiologist shortages. CAD systems improve TB detection accuracy, aiding in early diagnosis

According to (Codlin *et al.*, 2021), in the research "Independent evaluation of 12 artifcial intelligence solutions for the detection of tuberculosis" aimed to evaluate 12 AI solutions for tuberculosis detection performance. Assess CAD software performance against human readers in TB screening. Investigate the impact of radiography equipment on CAD software performance. Address the gap between TB treatment and incidence for elimination. Explore the potential of AI in improving TB screening initiatives. Developed test library of CXR images for independent CAD evaluation. Evaluated 12 CAD software solutions using TB clinicians as reference. The results showed that Six CAD systems performed on par with the Expert Reader. Qure.ai, Delft Imaging, and Lunit showed significantly better performance. Xpert positivity was higher with the DRTECH radiography system. CAD software performance can be affected by radiography equipment used. Some of the highlighted limitations were; High CAD score with negative Xpert test affects software performance. False positive Xpert results in patients with past TB treatment. Performance impairment based on radiography system used for CXR capture.

Inter-reader variability in CXR images poses challenges in interpretation. In conclusion, CAD software enhances TB screening, matching human reader performance. AI technology improves TB screening, aiding in early disease detection.

(Nafisah & Muhammad, 2024)**,** performed research on the topic "Tuberculosis detection in chest radiograph using convolutional neural network architecture and explainable artificial intelligence" Develop automatic TB detection system using deep learning models. Compare CNN models for TB detection performance on CXR datasets. Utilized explainable AI to visualize TB-infected lung regions. The methodology involved studying DL methods for TB detection in patient radiography. Propose automatic TB detection system for CXR images. Evaluate system on multiple datasets and cross-dataset scenarios. The results showed that, EfficientNetB3 achieved highest accuracy of 99.1% in TB detection. ResNet50 and EfficientNetB3 showed smooth loss graphs during training. Pretrained CNN models demonstrated high performance in TB classification. It can be concluded that the proposed system achieves 99.1% accuracy using EfficientNetB3 CNN model. Segmented lung CXR images enhance TB detection performance significantly. UNet architecture aids in semantic segmentation for medical image analysis.

(Alsaffar *et al.*, 2021) focused on Detecting tuberculosis in medical X-ray imaging using classification methods and also applying artificial intelligence for automatic classification of chest X-ray images in the study "Detection of Tuberculosis Disease Using Image Processing Technique" the following methods were employed; Three classification methods used: logistic regression, support vector machines, KNN. Preprocessing involved padding, resizing images to 224x224 dimensions. From the research; results include comparison of classification scenarios using different machine learning methods. The best performance was observed with SVM as the learning method. Results show the effectiveness of the method for tuberculosis detection. The limitations of the study comprised; Previous studies required medical exams and trained professionals for input parameters. Proposed approach only uses radiographic images of the lungs. It's concluded that the study could do, Automatic classification of medical images with and without tuberculosis. Deep learning features extraction using RESNET50 neural network

The research "Deep and Hybrid Learning Technique for Early Detection of Tuberculosis Based on X-ray Images Using Feature Fusion" done by (Fati *et al.*, 2022) aimed to Develop AI techniques for early tuberculosis detection using X-ray images. Enhance X-ray images and increase contrast in the region of interest. The various algorithms were used; Hybrid method combines CNN, PCA, SVM for TB detection. ANN uses ResNet-50, GLCM, DWT, LBP for high accuracy diagnosis. The results showed that; Hybrid systems achieved high accuracy in diagnosing tuberculosis from X-ray images. ANN with fused features showed superior results for both datasets. Improved images and contrast in X-ray datasets for early TB detection. One limitation of the research was lack of sufficient images for the TB detection but it was addressed using image augmentation. In conclusion, When Early TB detection is crucial; CNN and ANN models show high accuracy. Proposed methods enhance TB diagnosis, aiding doctors and radiologists.

(Acharya *et al.*, 2022)aimed to develop AI model for TB detection using deep learning techniques, Utilize normalization-free networks for X-ray image classification and Achieve high accuracy and AUC for TB diagnosis in radiology. in the research "AIAssisted Tuberculosis Detection and Classification from Chest X-Rays Using a Deep

Learning Normalization-Free Network Model". It involved Progressive resizing for training models using chest X-ray images. ImageNet fine-tuned Normalization-Free Networks (NFNets) for classification. Score-Cam algorithm highlights regions in chest XRays for detailed inference. The results showed; Proposed method achieved 96.91% accuracy, 99.38% AUC. Score-CAM highlighted TB regions in chest X-rays effectively. Models outperformed other deep learning architectures in TB classification. The limitations of the developed models were due to; Gradient clipping sensitivity and weight factor evaluation challenges. Discrepancies in network behavior during training and testing. It can be concluded from the research that; Deep learning-assisted TB diagnosis with normalization-free network achieved high accuracy and that the Proposed method can be a secondary decision tool in clinical settings.

(Alghamdi *et al.*, 2021)**,** performed a survey "Deep Learning Approaches for Detecting COVID-19 From Chest X-Ray Images: A Survey" The research aims to review and critically assess the use of deep learning methods for diagnosing COVID-19 from chest Xray (CXR) images. It focuses on exploring the potential of convolutional neural networks (CNNs) and other deep learning architectures for automatic diagnosis of COVID-19 using visual information from CXR images. The study also highlights the need for comprehensive and diverse datasets, explainable decisions, and justifiable predictions in this domain. Various deep learning architectures, particularly CNNs, were analyzed for detecting COVID-19 from CXR images. The study reviewed preprint and published reports from March to May 2020, focusing on CNN-based transfer learning and the datasets used by different studies. Challenges such as dataset imbalance, model transparency, and the need for domain knowledge integration were discussed. The Results showed that Most studies utilized CNN-based transfer learning with publicly available datasets, but there is room for improvement due to dataset limitations and imbalance issues. The research highlighted the varying dataset sizes and the need for data augmentation to address the scarcity of COVID-19 CXR images. The main limitation identified was the class imbalance problem in COVID-19 datasets, affecting the robustness of machine learning algorithms. In conclusion, The study emphasizes the significant potential of deep learning methods for automatic COVID-19 diagnosis from CXR images but stresses the importance of collaboration between medical personnel and computer scientists to validate the effectiveness of these techniques

(Jain *et al.*, 2021) performed a research on "Deep learning based detection and analysis of COVID-19 on chest X-ray images" The research aimed to utilize deep learning techniques to detect and analyze COVID-19 on chest X-ray images. The methodology involved collecting 6432 chest X-ray scans from the Kaggle repository, cleaning the data, and using data augmentation techniques to increase the size of the dataset. Three deep learning models, Inception V3, Xception, and ResNeXt, were compared for their performance, with Xception achieving the highest accuracy of 97.97% in detecting COVID-19 cases. The study acknowledged the limitations of a potentially small dataset for training, which could lead to overfitting, emphasizing the need for validation with new data. The research concluded that Xception net showed the best performance in classifying COVID-19 affected patients and suggested the potential automation of diagnosis tasks in the future. It also highlighted the importance of consulting medical professionals for practical applications and emphasized the economic feasibility of using deep learning for disease detection.

(Benmalek *et al.*, 2021) The research paper aimed to evaluate the performance of different convolutional neural networks (CNN) in diagnosing COVID-19 using chest imaging techniques like chest X-ray (CXR) and computed tomography (CT) scan images. The study compares ResNet-18, InceptionV3, and MobileNetV2 models, with ResNet-18 showing the highest precision and sensitivity for COVID-19 cases. The methodology involves analyzing confusion matrices for each model, showcasing their precision, sensitivity, specificity, and F1-score. The results indicate high accuracy in classifying COVID-19 cases, outperforming other diagnostic methods like saliva sample RT-PCR. However, limitations include misclassification of viral pneumonia cases as COVID-19 due to image quality issues and dataset biases. Despite the promising outcomes, challenges in image quality and dataset merging may affect the models' clinical applicability. In conclusion, combining deep learning models with chest imaging offers an efficient method for COVID19 detection, with CT scans demonstrating superior performance over CXR images, although challenges related to image quality and dataset integration need to be addressed for reliable clinical use.

(Zhou *et al.*, 2021) “A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises” The research paper aims to review the application of deep learning in medical imaging, highlighting technological trends, case studies, and future promises in the field. It emphasizes the challenges unique to medical imaging that deep learning must address, such as the need for big data with annotations and advancements in high-performance computing. The paper reviews various technological challenges in medical imaging domains and tasks, focusing on solutions to the data challenge. It discusses the trend of making neural networks deeper, starting from AlexNet to more advanced models like VGGNet, Inception Net, and ResNet. Techniques like skip connections, deep supervision, attention mechanisms, and squeeze and excitation are explored to enhance network performance. The study showcases progress highlights in selected medical imaging cases, including thoracic, neuro, cardiovascular, abdominal, and microscopy imaging. It discusses the advancements made in these areas using deep learning techniques, demonstrating the potential of AI in improving diagnostic accuracy and efficiency in clinical practices. One limitation highlighted is the long-tailed distribution of disease patterns in medical images, where most diseases are infrequent, posing challenges for model training and generalization. Additionally, the "black box" nature of deep learning models is acknowledged, as they lack explicit interpretability based on known physical or mathematical principles. It can be concluded that by emphasizing the transformative potential of deep learning in medical imaging, particularly in identifying complex imaging patterns imperceptible to visual radiologic evaluation. It suggests that AI-based image analysis can redefine diseases on a neurobiological basis, enable early detection, and enhance personalized risk estimates, thereby revolutionizing clinical protocols and contributing to precision medicine (Panayides *et al.*, 2020) The research paper aims to review cutting-edge solutions in medical imaging informatics, discuss clinical translation, and propose future directions for enhancing clinical practice. The methodology involves exploring advances in medical imaging acquisition technologies for various modalities and emphasizing the importance of efficient medical data management strategies. The results highlight the significance of transfer learning approaches in popular frameworks to widen the deep learning research base for new applications. However, the study acknowledges limitations, such as the need for efficient enterprise-wide clinical data repositories to aggregate diverse medical information systematically. In conclusion, the paper underscores the pivotal role of medical imaging informatics in driving clinical research and practice, with AI-based approaches gaining FDA approval for tasks like image enhancement, segmentation, and abnormality detection. The future direction foresees substantial growth in FDA-approved AI-based solutions for radiology and digital pathology images, revolutionizing healthcare practices.

(Ma *et al.*, 2024) The research aimed to evaluate the effectiveness of MedSAM, a foundation model for medical image segmentation, compared to other models like SAM7, U-Net, and DeepLabV3. The methodology involved internal and external validation, where MedSAM showed superior generalization abilities across various medical image segmentation tasks. Results indicated that MedSAM outperformed SAM7 and specialist models in most tasks, showcasing its potential for diverse segmentation challenges. However, SAM7 excelled in specific RGB image segmentation tasks due to distinct target appearances. The study's limitations include the focus on segmentation tasks only and the need for further validation on a broader range of medical imaging tasks. In conclusion, MedSAM's consistent performance on internal and external validation sets highlights its versatility and potential to advance diagnostic and therapeutic tools in medical imaging, ultimately enhancing patient care.

(Willemink *et al.*, 2020) in the research "Preparing Medical Imaging Data for Machine Learning" The research aimed to evaluate the effectiveness of MedSAM, a foundation model for medical image segmentation, compared to other models like SAM7, U-Net, and DeepLabV3. The methodology involved internal and external validation, where MedSAM showed superior generalization abilities across various medical image segmentation tasks. Results indicated that MedSAM outperformed SAM7 and specialist models in most tasks, showcasing its potential for diverse segmentation challenges. However, SAM7 excelled in specific RGB image segmentation tasks due to distinct target appearances. The study's limitations include the focus on segmentation tasks only and the need for further validation on a broader range of medical imaging tasks. In conclusion, MedSAM's consistent performance on internal and external validation sets highlights its versatility and potential to advance diagnostic and therapeutic tools in medical imaging, ultimately enhancing patient care.

(Sarvamangala & Kulkarni, 2022)**,** in the study "Convolutional neural networks in medical image understanding: a survey" The research paper aims to provide a comprehensive survey of the applications of Convolutional Neural Networks (CNNs) in medical image understanding, with the underlying objective of motivating researchers in this field to extensively utilize CNNs in their work. The methodology involves discussing CNNs, their award-winning frameworks, and major medical image understanding tasks like classification, segmentation, localization, and detection. The results highlight the critical and comprehensive survey of CNN applications in medical image understanding for various ailments like brain, breast, and lung conditions. However, the paper acknowledges some limitations, such as not covering all aspects of medical image understanding due to its vast scope. In conclusion, the paper emphasizes the effectiveness of CNNs in surpassing human experts in image understanding tasks and encourages further research in applying CNNs to enhance medical diagnosis and treatment planning.

(Varoquaux & Cheplygina, 2022) in the study “Machine learning for medical imaging: methodological failures and recommendations for the future” aimed to Identify systematic challenges in medical imaging analysis to provide recommendations for future improvements in the field, it highlights the challenges in medical imaging analysis and recommendations for future improvements, The Importance of robust validation methods using multiple datasets. Impact of incorrectly chosen baselines on algorithm evaluation. Best practices for medical machine learning evaluation and benchmarking standards. It also points out the criteria for accepting methods based on significant improvement over existing solutions. The research was able to identify challenges in medical imaging analysis with biases and data limitations. It also discusses efforts to counteract problems and provides recommendations for the future. Some of the limitations of the study were Dataset biases, hidden subgroups, and mislabeled instances. Lack of reproducibility due to omitted details and misinterpreted tests. Overfitting, lack of robust validation, and reuse of datasets, it can be concluded from the research that challenges in clinical ML research impact diagnostic accuracy and model evaluation. Strategies for improvement require procedural, normative, and goal changes.

Singh *et al.*, (2020). “3D Deep Learning on Medical Images: A Review” The research objectives of this paper are to trace the development of three-dimensional convolutional neural networks (3D CNNs) from their machine learning origins, provide a mathematical description, and outline the preprocessing steps required for medical images before they are analyzed using the model. The methodology involves a historical overview of the evolution of 3D CNNs, starting from the foundational advancements in machine learning and CNN architectures, particularly since the breakthrough success of AlexNet in 2012. The paper includes a mathematical exposition and describes the necessary preprocessing techniques for medical imaging data. The results of the research highlighted the advancements and efficacy of 3D CNNs in improving the accuracy and efficiency of medical image analysis across different tasks. The use of 3D CNNs has shown significant potential in enhancing disease diagnosis and assisting clinicians in their work. However, the paper acknowledges several limitations. The challenges associated with the use of the model in medical imaging include computational complexity, the need for large and diverse datasets, potential biases in training data, and the generalizability of models across.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 Data Gathering and Analysis:

This is the systematic process of gathering observations or measurements. Data being the most important factor for accuracy in Machine Learning will be gathered from relevant sources. Medical imaging data, specifically chest X-ray images with and without the tuberculosis condition will be gathered form the sources - Mendeley. Data, the open-source community Kaggle or zenodo.org. These sites are chosen because of their reliability in dataset quality and the recommendations by other machine learning experts.

The present dataset contains both healthy images and tuberculosis affected images, they are from "Tuberculosis (TB) Chest X-Ray Database" collected by researchers from Qatar and Dhaka University, Doha, Qatar, and collaboration with doctors from Hamad Medical Corporation and Bangladesh. All the images are CRX in PNG format and a size 512x512.

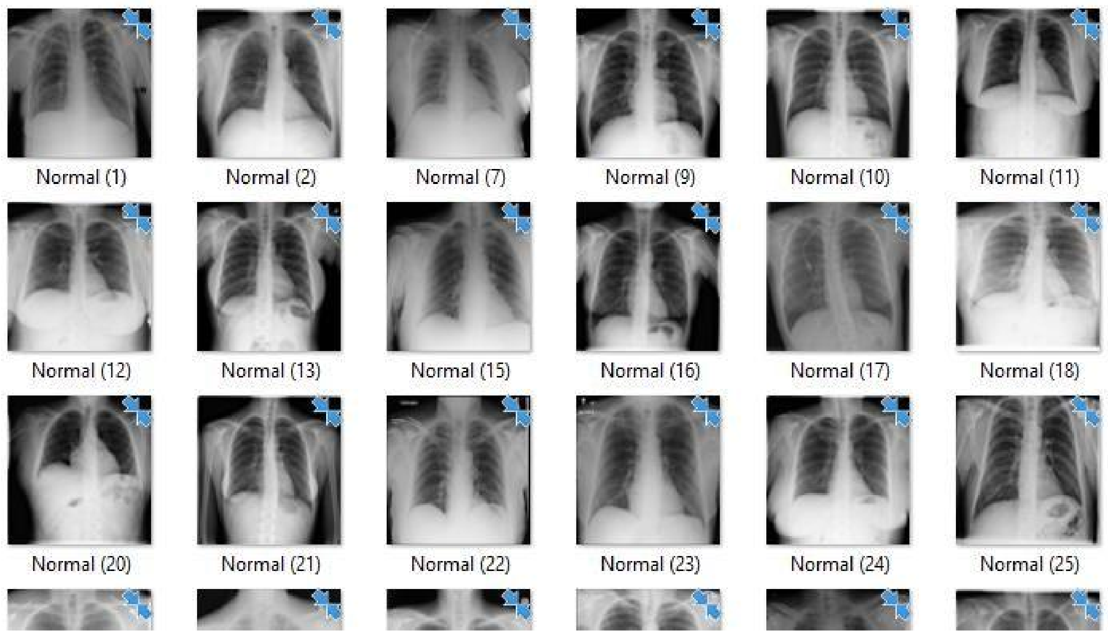


Figure 3.1: *Chest x-ray images without tuberculosis.*

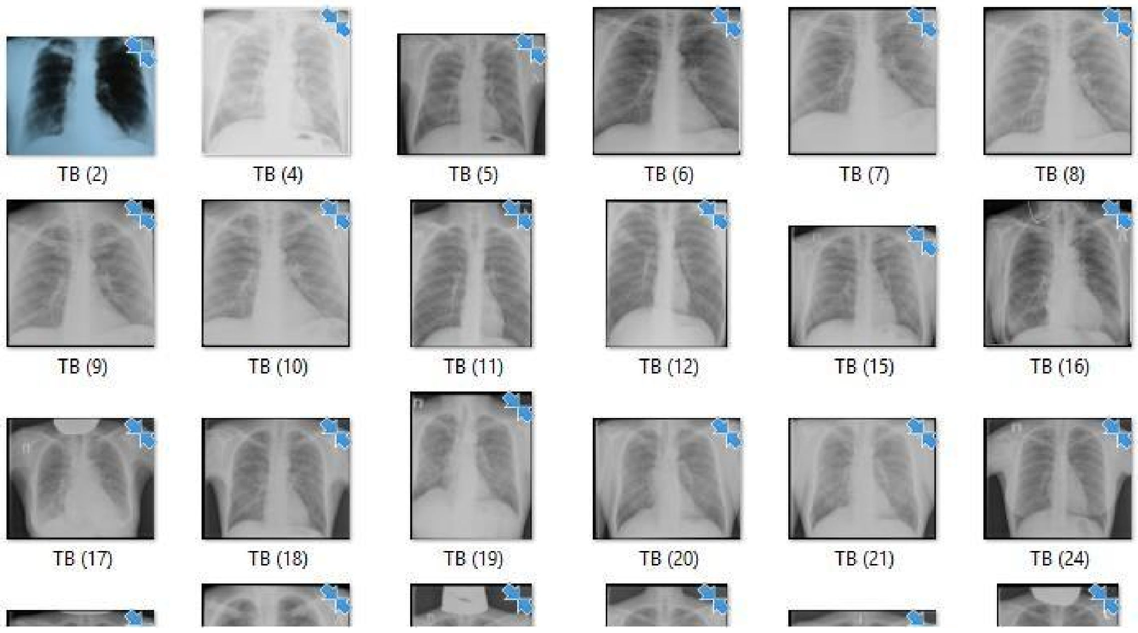


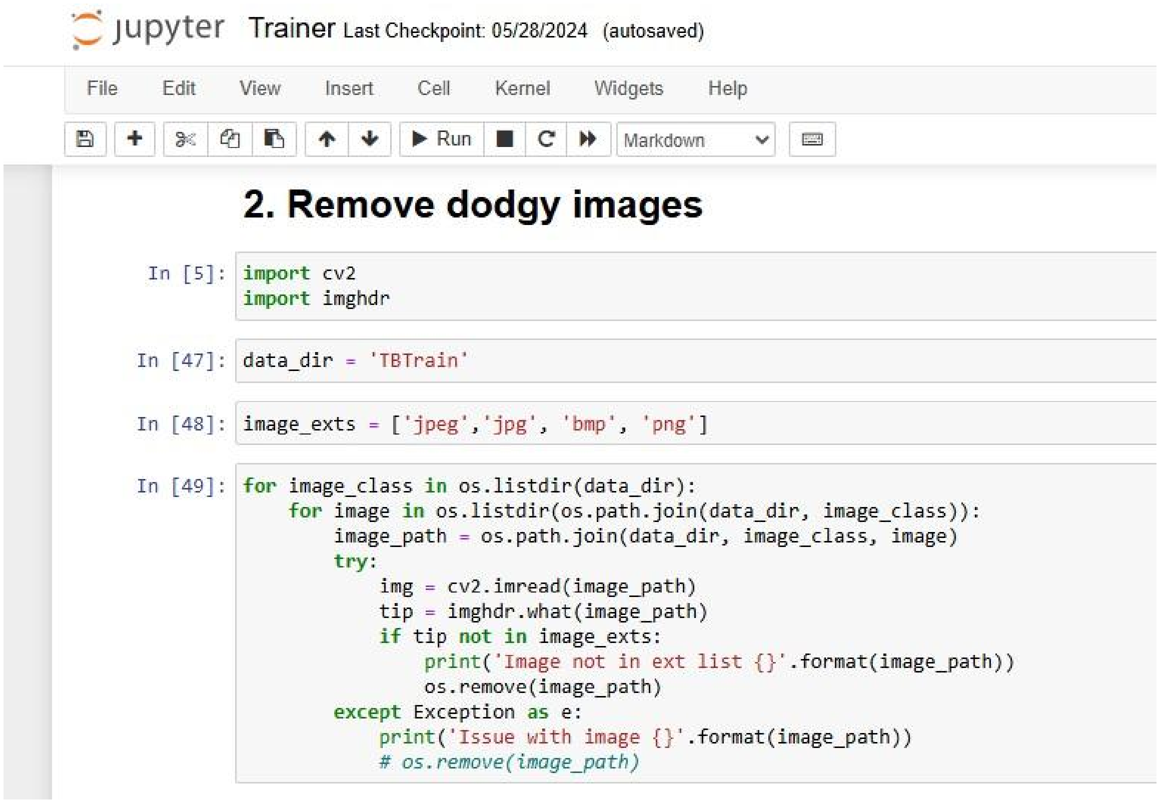
Figure 3.2: *Chest x-ray images with tuberculosis.*

CITE: Visuña, L. (2022). Computer-Aided diagnostic for classifying Chest X-Ray Images Using Deep Ensemble Learning [Data set]. Zenodo. https://doi.org/10.5281/zenodo.6637854

3.1.1 Data preprocessing:

Image preprocessing is the steps taken to format images before they are used by model training and inference. This includes, but is not limited to, resizing, orienting, and color corrections. Preprocessing is required to clean image data for model input. For example, fully connected layers in convolutional neural networks required that all images are the same sized arrays. Image preprocessing may also decrease model training time and increase model inference speed. If input images are particularly large, reducing the size of these images will dramatically improve model training time without significantly reducing model performance.

The images were preprocessed in jupyter notebook (from the anaconda software) see *figure* 3.3 below.



2. Remove dodgy images

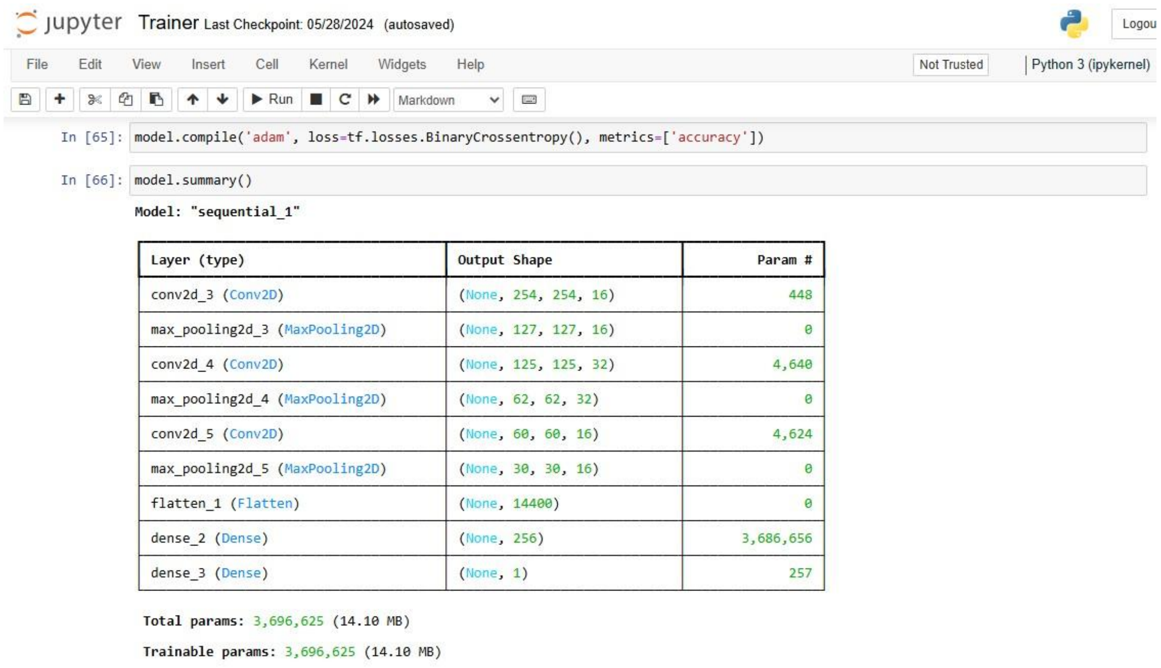
```
In [5]: import cv2
        import imghdr

In [47]: data_dir = 'TBTrain'

In [48]: image_exts = ['jpeg','jpg', 'bmp', 'png']

In [49]: for image_class in os.listdir(data_dir):
             for image in os.listdir(os.path.join(data_dir, image_class)):
                 image_path = os.path.join(data_dir, image_class, image)
                 try:
                     img = cv2.imread(image_path)
                     tip = imghdr.what(image_path)
                     if tip not in image_exts:
                         print('Image not in ext list {}'.format(image_path))
                         os.remove(image_path)
                 except Exception as e:
                     print('Issue with image {}'.format(image_path))
                     # os.remove(image_path)
```

Figure 3.3 image preprocessing in Jupyter notebook

3.1.2 Feature Extraction:

Features are characteristics of the objects of interest, which represent the maximum relevant information that the image has to offer for the complete characterization of a tumor. Feature extraction methodologies analyze objects and images to extract the most prominent features that are representative of the various classes of objects. Features are used as inputs to classifiers that assign them to the class that they represent. In this research, the CNN algorithm is proposed to be used.

Features used for tuberculosis detection are widely divided into two main categories: global features and local features.


jupyter Trainer Last Checkpoint: 05/28/2024 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [65]: model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])

In [66]: model.summary()
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 254, 254, 16)	448
max_pooling2d_3 (MaxPooling2D)	(None, 127, 127, 16)	0
conv2d_4 (Conv2D)	(None, 125, 125, 32)	4,640
max_pooling2d_4 (MaxPooling2D)	(None, 62, 62, 32)	0
conv2d_5 (Conv2D)	(None, 60, 60, 16)	4,624
max_pooling2d_5 (MaxPooling2D)	(None, 30, 30, 16)	0
flatten_1 (Flatten)	(None, 14400)	0
dense_2 (Dense)	(None, 256)	3,686,656
dense_3 (Dense)	(None, 1)	257

```
Total params: 3,696,625 (14.10 MB)
Trainable params: 3,696,625 (14.10 MB)
```

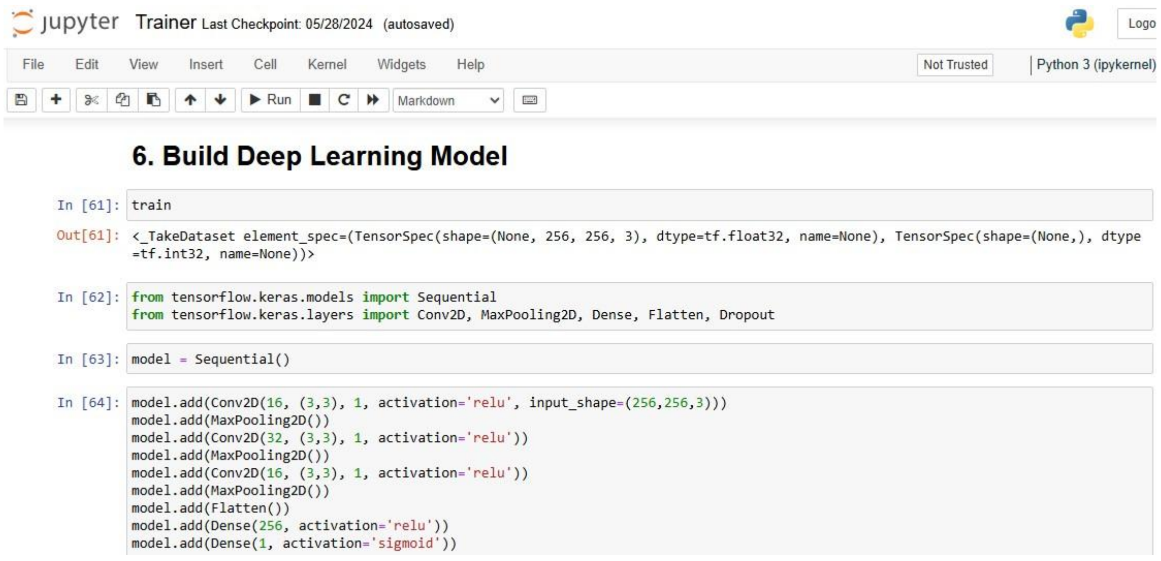
Figure 3.4 feature extraction parameters using Jupyter notebook

Global features are feasible features that are classified as general features and domainspecific features. This leads to the case of images where general features are considered for the detection process (Gemescu *et al.*, 2019). Local features are features extracted from a different section of the chest image which depends on image partitioning, there is a need for features to be at different levels, for instance at the chest level (Eweje *et al.* 2021).

3.1.3 Algorithm -CNN (Convolutional Neural Network)- Development:

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

The Convolutional Neural Network will be used because they are efficient in automatically extracting meaningful spatial features from images, enabling accurate object recognition. Can learn hierarchical features like edges, textures, and shapes, which are important for image data. Can reduce the number of weights and computational cost compared to fully connected layers.


jupyter
Trainer Last Checkpoint: 05/28/2024 (autosaved)
Logo
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

6. Build Deep Learning Model

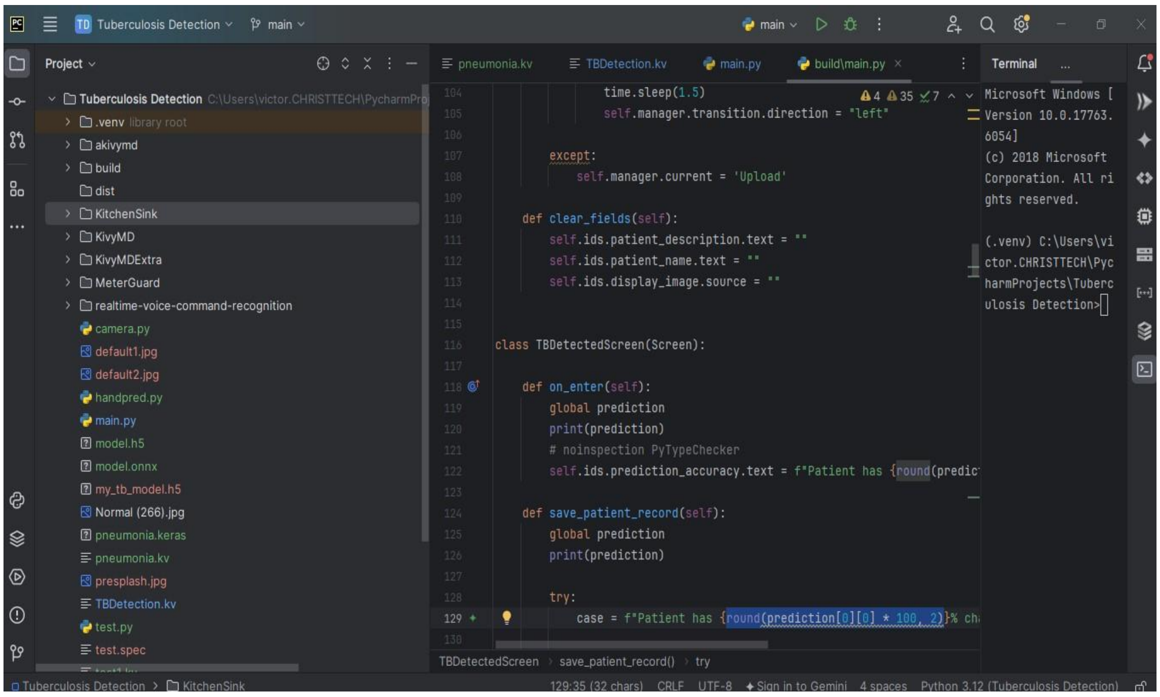
```
In [61]: train
Out[61]: <_TakeDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None,), dtype
         =tf.int32, name=None))>

In [62]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout

In [63]: model = Sequential()

In [64]: model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
         model.add(MaxPooling2D())
         model.add(Conv2D(32, (3,3), 1, activation='relu'))
         model.add(MaxPooling2D())
         model.add(Conv2D(16, (3,3), 1, activation='relu'))
         model.add(MaxPooling2D())
         model.add(Flatten())
         model.add(Dense(256, activation='relu'))
         model.add(Dense(1, activation='sigmoid'))
```

Figure 3.5 CNN algorithm setup

Furthermore, they reduce overfitting (Overfitting is when a model fits too closely to its training data and cannot generalize well to new data) by using pooling and dropout layers the algorithm can effectively handle translation invariant structures in the data, meaning they can recognize objects regardless of their position or scale.

3.2 Desktop application development:

For ease of use and access to the software, an offline application will be developed. An offline application has some advantages compared to web-based application; 1. Faster than web apps 2. Greater functionality as they have access to system resources 3. Can work offline 3. Easier to build due to the availability of developer tools, interface elements, and SDKs. The graphical user interface was developed in using the pycharm IDE. See *figure* 3.5 below



```
Tuberculosis Detection C:\Users\victor.CHRISTTECH\PycharmPro
  .venv library root
  akivymd
  build
  dist
  KitchenSink
  KivyMD
  KivyMDExtra
  MeterGuard
  realtime-voice-command-recognition
  camera.py
  default1.jpg
  default2.jpg
  handpred.py
  main.py
  model.h5
  model.onnx
  my_tb_model.h5
  Normal (266).jpg
  pneumonia.keras
  pneumonia.kv
  presplash.jpg
  TBDetection.kv
  test.py
  test.spec
```

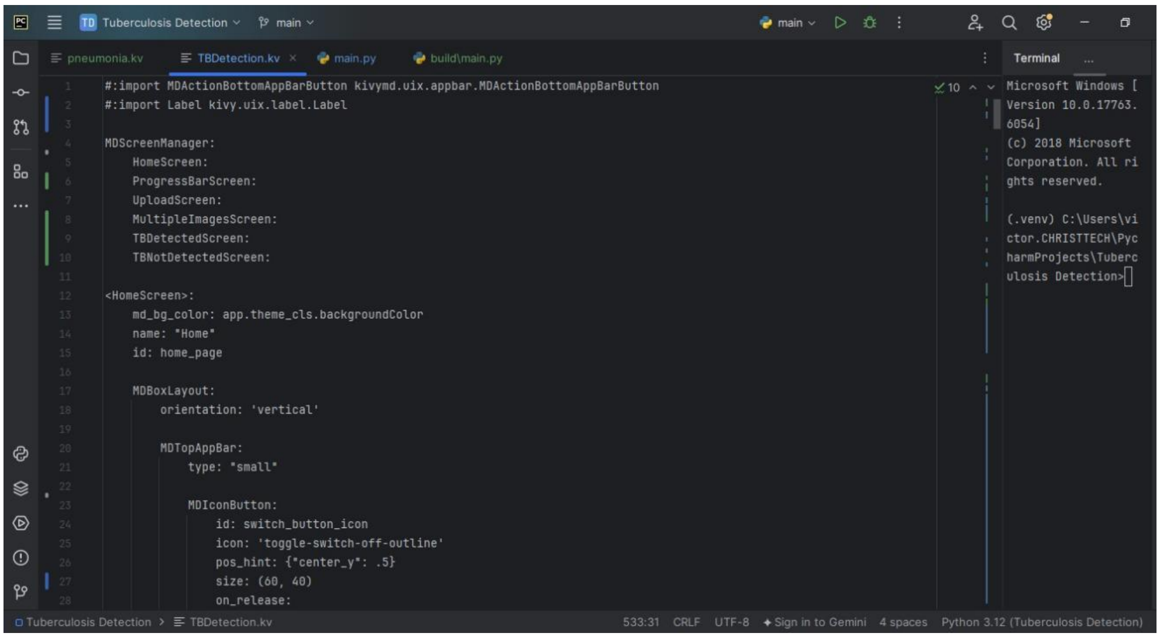
```python
                time.sleep(1.5)
                self.manager.transition.direction = "left"

        except:
            self.manager.current = 'Upload'

    def clear_fields(self):
        self.ids.patient_description.text = ""
        self.ids.patient_name.text = ""
        self.ids.display_image.source = ""


class TBDetectedScreen(Screen):

    def on_enter(self):
        global prediction
        print(prediction)
        # noinspection PyTypeChecker
        self.ids.prediction_accuracy.text = f"Patient has {round(predic

    def save_patient_record(self):
        global prediction
        print(prediction)

        try:
            case = f"Patient has {round(prediction[0][0] * 100, 2)}% ch
```

```
Microsoft Windows [
Version 10.0.17763.
6054]
(c) 2018 Microsoft
Corporation. All ri
ghts reserved.

(.venv) C:\Users\vi
ctor.CHRISTTECH\Pyc
harmProjects\Tuberc
ulosis Detection>
```

```
Tuberculosis Detection > KitchenSink
TBDetectedScreen > save_patient_record() > try
129:35 (32 chars)  CRLF  UTF-8  Sign in to Gemini  4 spaces  Python 3.12 (Tuberculosis Detection)
```

Figure 3.6 Pycharm integrated development environment

The Python Kivy/KivyMD framework will be used to create the user interface. KivyMD is an extension of the Kivy framework. KivyMD is a collection of Material Design widgets for use with Kivy, a GUI framework for making mobile applications. It is similar to the Kivy framework but provides a more attractive GUI.



```
#:import MDActionBottomAppBarButton kivymd.uix.appbar.MDActionBottomAppBarButton
#:import Label kivy.uix.label.Label

MDScreenManager:
    HomeScreen:
    ProgressBarScreen:
    UploadScreen:
    MultipleImagesScreen:
    TBDetectedScreen:
    TBNotDetectedScreen:

<HomeScreen>:
    md_bg_color: app.theme_cls.backgroundColor
    name: "Home"
    id: home_page

    MDBoxLayout:
        orientation: 'vertical'

        MDTopAppBar:
            type: "small"

            MDIconButton:
                id: switch_button_icon
                icon: 'toggle-switch-off-outline'
                pos_hint: {"center_y": .5}
                size: (60, 40)
                on_release:
```

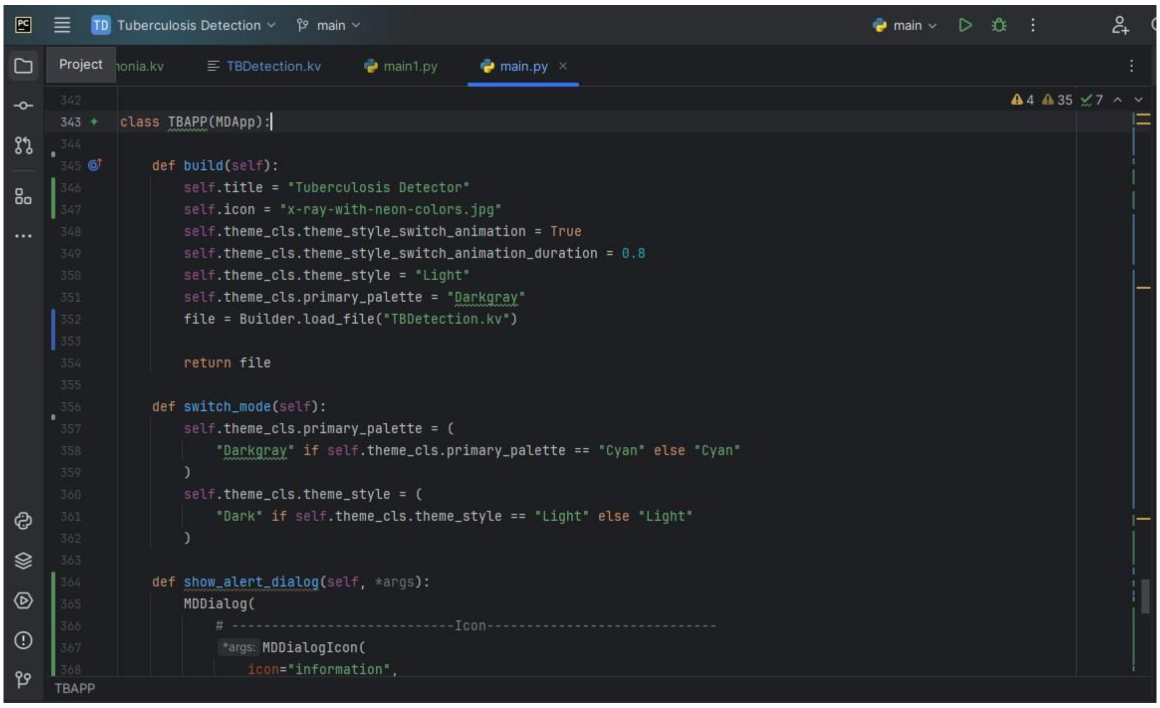
```
Microsoft Windows [
Version 10.0.17763.
6054]
(c) 2018 Microsoft
Corporation. All ri
ghts reserved.

(.venv) C:\Users\vi
ctor.CHRISTTECH\Pyc
harmProjects\Tuberc
ulosis Detection>
```

Figure 3.7 Kivy file snippet used for the user interface development

Python is a high-level, general-purpose programming language renowned for its readability and simplicity. Its syntax, characterized by significant indentation, enhances code clarity and maintainability. Unlike many languages, Python is dynamically typed, meaning variable types are determined at runtime, providing flexibility in development. As a garbage-collected language, Python automatically manages memory allocation and deallocation, relieving programmers from manual memory management.

Python's versatility is evident in its support for multiple programming paradigms. It excels in structured programming, offering a clear and sequential approach to problem-solving. Additionally, Python's object-oriented capabilities enable the creation of complex and modular applications through classes and objects. Functional programming concepts are also embraced, allowing for elegant and concise code through functions as first-class citizens. This adaptability makes Python suitable for a wide range of applications, from web development and data analysis to scientific computing and machine learning.

Beyond its core features, Python's extensive standard library provides a rich set of modules and functions for various tasks, often described as a "batteries included" philosophy. This comprehensive library, combined with its readability and ease of use, has contributed to Python's widespread adoption across industries and among programmers of all levels.



```
class TBAPP(MDApp):

    def build(self):
        self.title = "Tuberculosis Detector"
        self.icon = "x-ray-with-neon-colors.jpg"
        self.theme_cls.theme_style_switch_animation = True
        self.theme_cls.theme_style_switch_animation_duration = 0.8
        self.theme_cls.theme_style = "Light"
        self.theme_cls.primary_palette = "Darkgray"
        file = Builder.load_file("TBDetection.kv")

        return file

    def switch_mode(self):
        self.theme_cls.primary_palette = (
            "Darkgray" if self.theme_cls.primary_palette == "Cyan" else "Cyan"
        )
        self.theme_cls.theme_style = (
            "Dark" if self.theme_cls.theme_style == "Light" else "Light"
        )

    def show_alert_dialog(self, *args):
        MDDialog(
            # ---------------------------Icon----------------------------
            MDDialogIcon(
                icon="information",
```

Figure 3.8 Python code snippet used for the user interface development

CHAPTER FOUR

RESULTS AND DISCUSSIONS

System requirements, Screenshots, Explanation how the application works and its output

Minimum requirements

Intel core duo

2 GB of RAM

Pycharm IDE

Required Python Libraries Windows
operating system ***Best requirements:***

2.6 GHz Processor

A graphics processing unit

8 GB or more of RAM

Anaconda IDE

Required Python Libraries

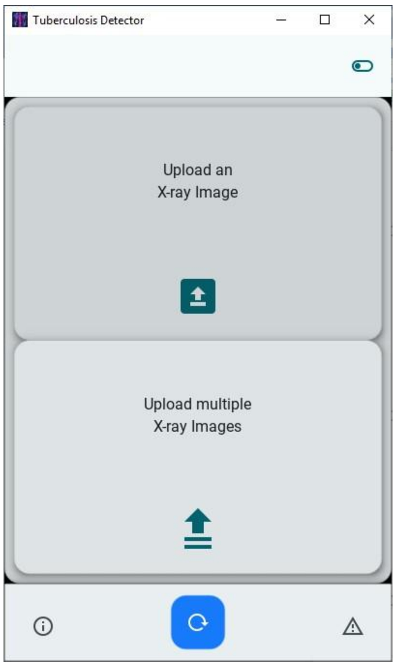
Windows operating system

4.1 Results

This section shows the resulting system's functionality and usage;

Interfaces

Opening the application will give you an interface that allows you to either upload a single image for prediction or multiple images coupled with a toggle button that makes the interface more appealing depending on the time of day (light mode or dark mode), two alert buttons (the one at the left of the reload button shows the information of the project's owner and the one at the right side of the screen shows a warning sign signifying that the software is not 100% accurate). *See figure 4.1* below



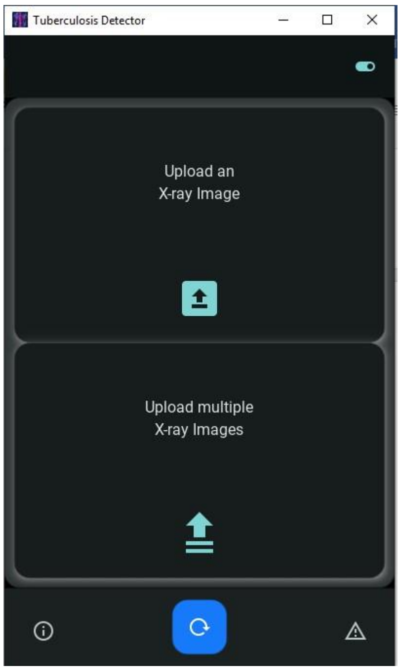


Figure 4.1: Starting of the system, light mode to the left and dark mode to the right.

Figure 4.2 below is the interface designed for user interaction, featuring two separate text boxes where users can enter the patient's information, such as name and relevant medical details. This organization helps in accurately capturing and managing patient data, ensuring that all necessary information is recorded for effective processing.

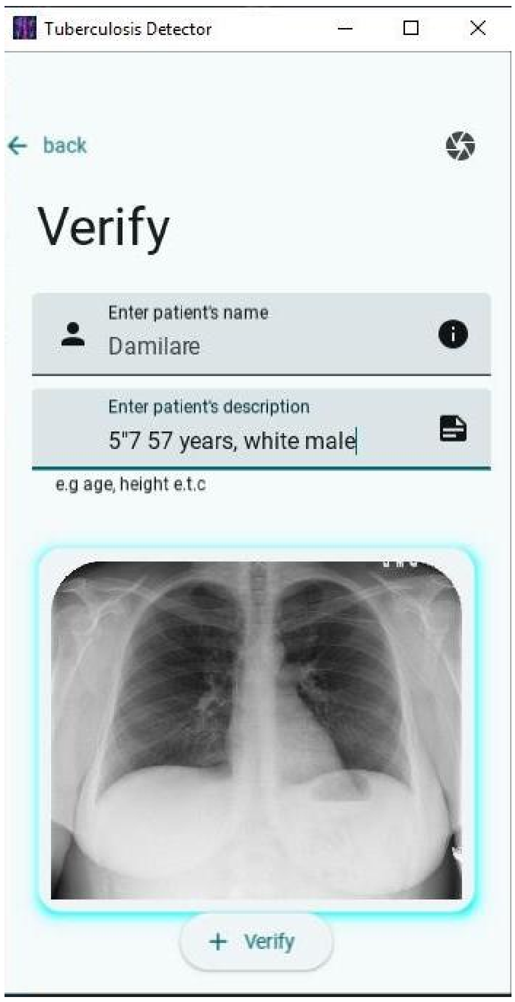
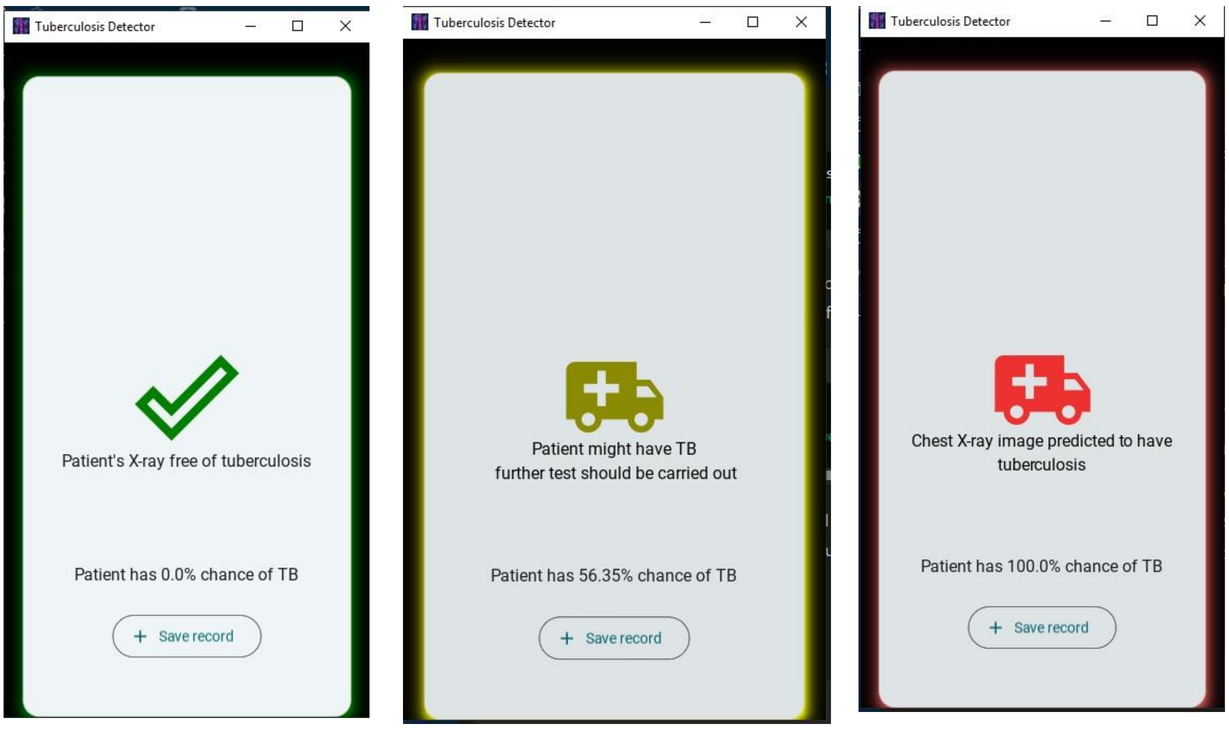


Figure 4.2: Image verification interface

The interface also includes a back button positioned at the top left of the screen, which allows users to easily navigate back and make changes to image uploads or other details if needed. The verify button is used to submit the uploaded image to a trained machine learning model for analysis and prediction. Additionally, the image box prominently displays the uploaded image, enabling users to visually confirm its accuracy and completeness before proceeding with the verification process.



Patient's X-ray free of tuberculosis

Patient has 0.0% chance of TB

Patient might have TB
further test should be carried out

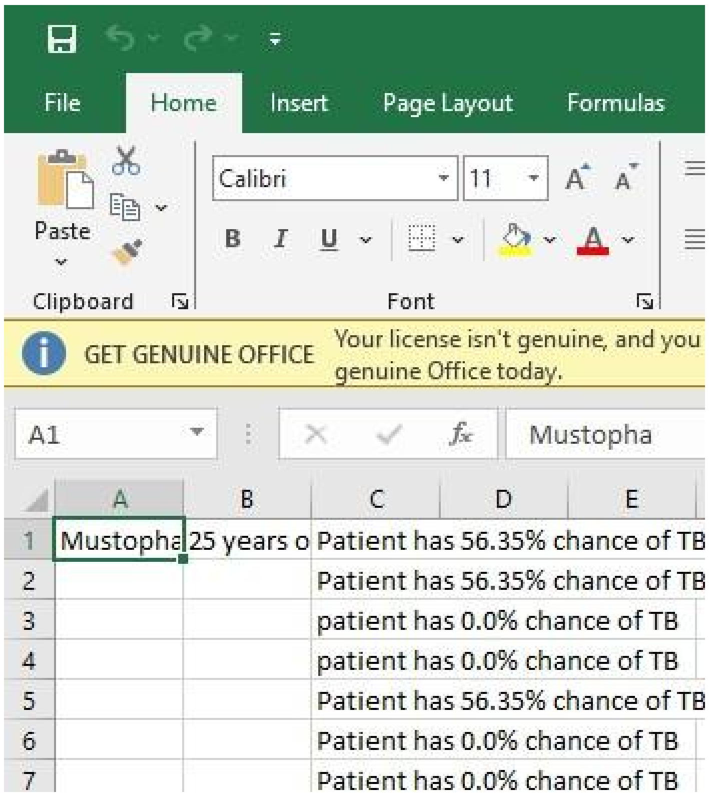
Patient has 56.35% chance of TB

Chest X-ray image predicted to have tuberculosis

Patient has 100.0% chance of TB

Figure 4.3: Without tuberculosis, not certain, with tuberculosis

The interfaces shown in *figure* 4.3 shows the prediction gotten from the machine learning model, the predicted state of the is listed in the order left to right (not infected, may be infected and infected). From this screen when the save record button is clicked the details entered previously and the prediction will be saved to an excel file for retrieval in a later time. See *figure* 4.4 below



	A	B	C	D	E
1	Mustopha	25 years o	Patient has 56.35% chance of TB		
2			Patient has 56.35% chance of TB		
3			patient has 0.0% chance of TB		
4			patient has 0.0% chance of TB		
5			Patient has 56.35% chance of TB		
6			Patient has 0.0% chance of TB		
7			Patient has 0.0% chance of TB		

Figure 4.4 excel sheet used for storing patient's information

From the main screen when the upload multiple image button is clicked the user is directed to select up to 10 different images which will be given to the predictive machine learning model and the infected images will be displayed in red while the non-infected images will be displayed in green (see *figure* 4.4 below).

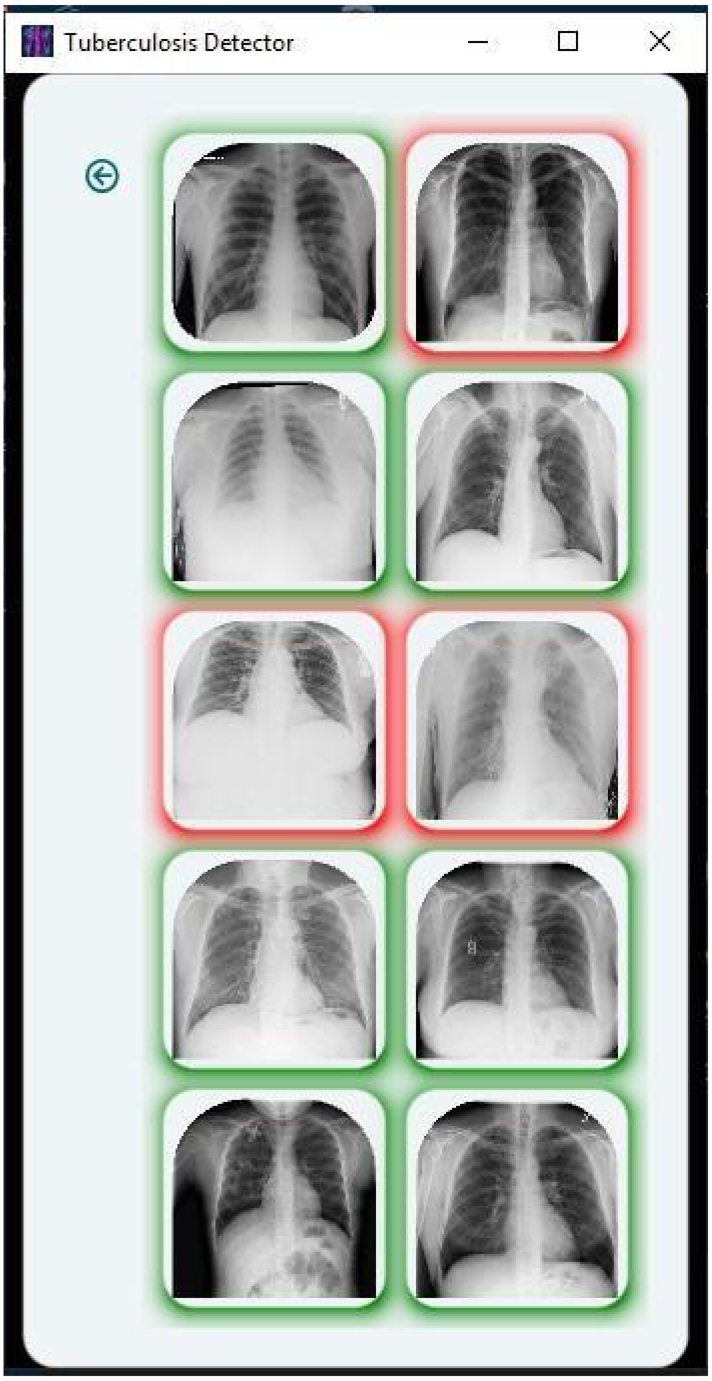


Figure 4.4 multiple image prediction screen

4.2 Discussion

The application interface is designed with user-friendliness and functionality in mind. The option to upload either a single image or multiple images, combined with a toggle for light and dark modes, ensures that the application is adaptable to various user preferences and environmental conditions, making it more accessible and comfortable for prolonged use. The clear layout, featuring separate text boxes for patient information and a visually distinct image box for confirmation, helps users maintain organized data entry and verification processes. The inclusion of alert buttons for project ownership and accuracy warnings adds transparency, fostering user trust and ensuring they are aware of the software's limitations.

Despite these strengths, the design offers opportunities for further enhancement. The manual image upload feature and interaction with various interface elements are beneficial for providing detailed control and customization, catering to users who appreciate a handson approach. The application’s capability to handle batch processing is a significant advantage, enabling efficient analysis of multiple images. To further optimize user experience, especially for those who may be less experienced, simplifying batch processing and refining the interface could make the application even more intuitive and responsive. These improvements would enhance user satisfaction and ensure the system performs efficiently, even with increased data volume.

CHAPTER FIVE

SUMMARY, CONCLUSION, RECOMMENDATION AND REFERENCES

This chapter presents the summary of the whole research study and the conclusion to the research. This section also presents related topics or areas for future researches.

5.1 Summary

In this project on tuberculosis prediction, a tuberculosis prediction system was developed using a chest x-ray image dataset for predicting the disease. The work focused on the development of machine learning enhancements for accurately predicting tuberculosis. In the later stages, discussion on various tools and techniques to bring the idea into practical application was carried out. After developing the predictive model, it was rigorously tested, and the results were analyzed, highlighting a few deficiencies. Following the testing phase, the advantages of the system were described and suggestions were offered for further enhancements and improvements.

5.2 Advantages of the system

i. Able to predict tuberculosis with a high accuracy

ii. Runs fast and consumes low power

iii. Provide significant help for the people with low orientation about prediction. iv. Lower operational costs.

v. Doesn’t require machine learning experts to use

5.3 Disadvantages

i. It is limited only to checking if an image has tuberculosis or not
ii. It is only available for desktop at the moment
iii. Might mistake some other diseases for tuberculosis i.e pneumonia

5.4 Conclusion

This project detects ML-based framework for intrusion detection uptake classification and correlates its underlying factors. The whole research has been succeeded based on the newly proposed dataset gathered from zenodo.org. The recommended framework reveals that a weighted ensemble of ML models successfully enhances the prediction results, as it weighted aggregates the output probabilities of the ensemble candidates' model.

5.5 Recommendation

In future works, this work can be taken into more detail and more work can be done on the project in order to bring modifications and additional features. The current system can only accurately predict tuberculosis infected and non-infected images, and also a larger dataset can be used to retrain the model so as to improve the efficiency of the software. The current version of the software supports only few areas of the system application software which is only on machine learning environment.

REFERENCE

Acharya, V., Dhiman, G., Prakasha, K., Bahadur, P., Choraria, A., M, S., J, S., Prabhu, S., Chadaga, K., Viriyasitavat, W., & Kautish, S. (2022). AI-Assisted Tuberculosis Detection and Classification from Chest X-Rays Using a Deep Learning Normalization-Free Network Model. *Computational Intelligence and Neuroscience*, *2022*, 1–19. https://doi.org/10.1155/2022/2399428

Alghamdi, H. S., Amoudi, G., Elhag, S., Saeedi, K., & Nasser, J. (2021). Deep Learning Approaches for Detecting COVID-19 From Chest X-Ray Images: A Survey. *IEEE Access*, *9*, 20235–20254. IEEE Access. https://doi.org/10.1109/ACCESS.2021.3054484

Alsaffar, M., Alshammari, G., Alshammari, A., Aljaloud, S., Almurayziq, T. S., Hamad, A. A., Kumar, V., & Belay, A. (2021). Detection of Tuberculosis Disease Using Image Processing Technique. *Mobile Information Systems*, *2021*, e7424836. https://doi.org/10.1155/2021/7424836

Benmalek, E., Elmhamdi, J., & Jilbab, A. (2021). Comparing CT scan and chest X-ray imaging for COVID-19 diagnosis. *Biomedical Engineering Advances*, *1*, 100003. https://doi.org/10.1016/j.bea.2021.100003

Born, J., Beymer, D., Rajan, D., Coy, A., Mukherjee, V. V., Manica, M., Prasanna, P., Ballah, D., Guindy, M., Shaham, D., Shah, P. L., Karteris, E., Robertus, J. L., Gabrani, M., & Rosen-Zvi, M. (2021). On the role of artificial intelligence in medical imaging of COVID-19. *Patterns*, *2*(6), 100269. https://doi.org/10.1016/j.patter.2021.100269

Çallı, E., Sogancioglu, E., van Ginneken, B., van Leeuwen, K. G., & Murphy, K. (2021). Deep learning for chest X-ray analysis: A survey. *Medical Image Analysis*, *72*, 102125. https://doi.org/10.1016/j.media.2021.102125

Codlin, A. J., Dao, T. P., Vo, L. N. Q., Forse, R. J., Van Truong, V., Dang, H. M., Nguyen, L. H., Nguyen, H. B., Nguyen, N. V., Sidney-Annerstedt, K., Squire, B., Lönnroth, K., & Caws, M. (2021). Independent evaluation of 12 artificial intelligence solutions for the detection of tuberculosis. *Scientific Reports*, *11*(1), 23895. https://doi.org/10.1038/s41598-021-03265-0

Fati, S. M., Senan, E. M., & ElHakim, N. (2022). Deep and hybrid learning technique for early detection of tuberculosis based on X-ray images using feature fusion. *Applied Sciences*, *12*(14), 7092.

Gemescu, I. N., Thierfelder, K. M., Rehnitz, C., & Weber, M.-A. (2019). Imaging Features of Bone Tumors. *Magnetic Resonance Imaging Clinics of North America*, *27*(4), 753–767. https://doi.org/10.1016/j.mric.2019.07.008

Geric, C., Qin, Z. Z., Denkinger, C. M., Kik, S. V., Marais, B., Anjos, A., David, P.-M., Ahmad Khan, F., & Trajman, A. (2023). The rise of artificial intelligence reading of chest X-rays for enhanced TB diagnosis and elimination. *The International Journal of Tuberculosis and Lung Disease*, *27*(5), 367–372. https://doi.org/10.5588/ijtld.22.0687

Gill, C. M., Dolan, L., Piggott, L. M., & McLaughlin, A. M. (2022). New developments in tuberculosis diagnosis and treatment. *Breathe*, *18*(1), 210149. https://doi.org/10.1183/20734735.0149-2021

Jain, R., Gupta, M., Taneja, S., & Hemanth, D. J. (2021). Deep learning based detection and analysis of COVID-19 on chest X-ray images. *Applied Intelligence*, *51*(3), 1690–1700. https://doi.org/10.1007/s10489-020-01902-1

Kasban, H., El-Bendary, M. A. M., & Salama, D. H. (2015). *A Comparative Study of Medical Imaging Techniques*.

Kik, S. V., Gelaw, S. M., Ruhwald, M., Song, R., Khan, F. A., Van Hest, R., Chihota, V., Nhung, N. V., Esmail, A., Celina Garfin, A. M., Marks, G. B., Gorbacheva, O., Akkerman, O. W., Moropane, K., Ngoc Anh, L. T., Dheda, K., Fox, G. J., Marano,

N., Lönnroth, K., … Denkinger, C. M. (2022). *Diagnostic accuracy of chest X-ray interpretation for tuberculosis by three artificial intelligence-based software in a screening use-case: An individual patient meta-analysis of global data*. https://doi.org/10.1101/2022.01.24.22269730

Ma, J., He, Y., Li, F., Han, L., You, C., & Wang, B. (2024). Segment anything in medical images. *Nature Communications*, *15*(1), 654. https://doi.org/10.1038/s41467-02444824-z

Nafisah, S. I., & Muhammad, G. (2024). Tuberculosis detection in chest radiograph using convolutional neural network architecture and explainable artificial intelligence. *Neural Computing and Applications*, *36*(1), 111–131. https://doi.org/10.1007/s00521-022-07258-6

Panayides, A. S., Amini, A., Filipovic, N. D., Sharma, A., Tsaftaris, S. A., Young, A., Foran, D., Do, N., Golemati, S., Kurc, T., Huang, K., Nikita, K. S., Veasey, B. P., Zervakis, M., Saltz, J. H., & Pattichis, C. S. (2020). AI in Medical Imaging Informatics: Current Challenges and Future Directions. *IEEE Journal of Biomedical and Health Informatics*, *24*(7), 1837–1857. IEEE Journal of Biomedical and Health Informatics. https://doi.org/10.1109/JBHI.2020.2991043

Qin, Z. Z., Ahmed, S., Sarker, M. S., Paul, K., Adel, A. S. S., Naheyan, T., Barrett, R., Banu, S., & Creswell, J. (2021). Tuberculosis detection from chest x-rays for triaging in a high tuberculosis-burden setting: An evaluation of five artificial intelligence algorithms. *The Lancet Digital Health*, *3*(9), e543–e554. https://doi.org/10.1016/S2589-7500(21)00116-3

Sarvamangala, D. R., & Kulkarni, R. V. (2022). Convolutional neural networks in medical image understanding: A survey. *Evolutionary Intelligence*, *15*(1), 1–22. https://doi.org/10.1007/s12065-020-00540-3

Singh, S. P., Wang, L., Gupta, S., Goli, H., Padmanabhan, P., & Gulyás, B. (2020). 3D Deep Learning on Medical Images: A Review. *Sensors*, *20*(18), Article 18. https://doi.org/10.3390/s20185097

Varoquaux, G., & Cheplygina, V. (2022). Machine learning for medical imaging: Methodological failures and recommendations for the future. *Npj Digital Medicine*, *5*(1), 1–8. https://doi.org/10.1038/s41746-022-00592-y

Willemink, M. J., Koszek, W. A., Hardell, C., Wu, J., Fleischmann, D., Harvey, H., Folio, L. R., Summers, R. M., Rubin, D. L., & Lungren, M. P. (2020). Preparing Medical Imaging Data for Machine Learning. *Radiology*, *295*(1), 4–15. https://doi.org/10.1148/radiol.2020192224

Zhou, S. K., Greenspan, H., Davatzikos, C., Duncan, J. S., Van Ginneken, B., Madabhushi, A., Prince, J. L., Rueckert, D., & Summers, R. M. (2021). A Review of Deep Learning in Medical Imaging: Imaging Traits, Technology Trends, Case Studies With Progress Highlights, and Future Promises. *Proceedings of the IEEE*, *109*(5), 820–838. Proceedings of the IEEE. https://doi.org/10.1109/JPROC.2021.3054390

APPENDIX

```
import os


import time
from kivy.properties import ListProperty, NumericProperty, StringProperty, Clock
from kivy.uix.anchorlayout import AnchorLayout
import kivy
from kivymd.app import MDApp
from kivy.core.window import Window
from kivy.lang.builder import Builder
from kivy.uix.screenmanager import ScreenManager, Screen
from plyer import filechooser
import tensorflow as tf
import numpy as np
import cv2
from openpyxl import Workbook, load_workbook
from kivy.clock import Clock
from kivymd.uix.dialog import (
    MDDialog,
    MDDialogIcon,
    MDDialogHeadlineText,
    MDDialogSupportingText,
)


if os.name == "posix":
    pass
else:
    Window.size = (400, 650)



prediction = ""


# this holds the image to be verified
picture = ""


# loads the pre-trained model
loaded_model = tf.keras.models.load_model("my_tb_model.h5")



class HomeScreen(Screen):


    def file_chooser(self):
        filechooser.open_file(on_selection=self.selected,
                              filters=["*.*", "*.png", "*.jpg", "*.bmp"], )


    def selected(self, selection):
        # this assigns the image_path to the picture variable
        global picture
        if selection:
            picture = selection[0]
    # changes the icon of the toggle button
    def change_toggle_button(self):
        button = self.ids.switch_button_icon
        button.icon = 'toggle-switch-off-outline' if button.icon == 'toggle-switch' else 'toggle-switch'

    def on_button_press(self):
        load_multiple_image = self.manager.get_screen('Multiple')
        load_multiple_image.load_multiple_files()

    def reload(self):
        self.manager.current = "Progress"
        self.manager.current = "Home"

    def splash(self, *args):
        ScreenManager.current = "Home"


class ProgressScreen(Screen):
    pass


class UploadScreen(Screen):

    def change_image(self):
        global picture
        self.ids.display_image.source = f"{picture}"


    def verify_image(self):
        try:
            global picture, prediction
            img = cv2.imread(f"{picture}")
            resize = tf.image.resize(img, (256, 256))
            input_img = np.expand_dims(resize / 255, 0)


            prediction = loaded_model.predict(input_img)


            if prediction > 0.75:
                self.manager.current = "Detected"
                #time.sleep(1.5)
                self.manager.transition.direction = "left"


            elif prediction > 0.4:
                self.manager.current = "Detected"
                self.manager.direction = "left"
                self.manager.get_screen("Detected").ids.unsafe_state.shadow_color = "yellow"
                self.manager.get_screen("Detected").ids.verified_check.icon_color = "#959500"
                self.manager.get_screen("Detected").ids.successful_sign.text = ("Patient might have TB\n
"
                                                                             "further test should be carried out")
            else:
                self.manager.current = "NotDetected"
                time.sleep(1.5)
                self.manager.transition.direction = "left"

        except:
            self.manager.current = 'Upload'

    def clear_fields(self):
        self.ids.patient_description.text = ""
        self.ids.patient_name.text = ""
        self.ids.display_image.source = ""


class TBDetectedScreen(Screen):

    def on_enter(self):
        global prediction
        print(prediction)
        # noinspection PyTypeChecker
        self.ids.prediction_accuracy.text = f"Patient has {round(prediction[0][0] * 100, 2)}% chance of
TB"

    def save_patient_record(self):
        global prediction
```

```
print(prediction)

try:
    case = f"Patient has {round(prediction[0][0] * 100, 2)}% chance of TB"

    """this section adds the patient information to an excel sheet"""
    excel_file_name = "TB_Diagnosis.xlsx"
    excel_path = os.getcwd() + "\\" + excel_file_name

    if not os.path.exists(excel_path):
        workbook = Workbook()
        sheet = workbook.active

        sheet["A1"] = f"{self.manager.get_screen('Upload').ids.patient_name.text}"
        sheet["B1"] = f"{self.manager.get_screen('Upload').ids.patient_description.text}"
        sheet["C1"] = f"{case}"

        filename = excel_file_name
        workbook.save(filename=filename)

    else:
        work_book = excel_file_name
        file = load_workbook(work_book)
        page = file.active
        new_patient_data = [self.manager.get_screen('Upload').ids.patient_name.text,
                            self.manager.get_screen('Upload').ids.patient_description.text, case]


        row = page.max_row + 1
        column = 1
        for i, info in enumerate(new_patient_data, start=row):
            page.cell(row=row, column=column, value=info)


            column += 1


        file.save(filename=work_book)


    self.manager.current = "Home"
    self.manager.transition.direction = "right"


# this is triggered when the patient information can't be saved to the excelsheet
except PermissionError:
    self.ids.prediction_accuracy.text = "Can't save: close excel sheet and try again..."
    self.ids.prediction_accuracy.color = (1, 0, 0, 1)
    if self.manager.current == "Detected":
        self.manager.current = "Detected"
    else:
        self.manager.current = "NotDetected"
```

```
class MultipleImagesScreen(Screen):


    def load_multiple_files(self):
        """"A Function that allows the user to select multiple image files"""


        filechooser.open_file(
            multiple=True,
            filters=["*.*", "*.png", "*.jpg", "*.bmp"],
            on_selection=self.on_file_selection
        )


    def on_file_selection(self, selection):
        """A callback function that is called when a file is selected."""
        global prediction
        max_files = 10  # Set your desired maximum number of files
        selected_files = len(selection)


        if selected_files > max_files:
            use_files = selection[:max_files]
        else:
            use_files = selection
        self.selected_files = use_files


        def multiple_image_verify():
```

```
try:
    count = 1

    for files in self.selected_files:
        img = cv2.imread(f"{files}")

        if img is None:
            print(f"Error: Unable to load image {files}. Check file path and file integrity.")
            continue

        resize = tf.image.resize(img, (256, 256))
        input_img = np.expand_dims(resize / 255, 0)

        prediction = loaded_model.predict(input_img)

        if prediction > 0.5:
            print("image predicted to have TB")
            print(files)

            if count == 1:
                self.ids.space_1.shadow_color = "red"
                self.ids.display_image_1.source = f"{files}"

            elif count == 2:
                self.ids.space_2.shadow_color = "red"
    self.ids.display_image_2.source = f"{files}"
elif count == 3:
    self.ids.space_3.shadow_color = "red"
    self.ids.display_image_3.source = f"{files}"
elif count == 4:
    self.ids.space_4.shadow_color = "red"
    self.ids.display_image_4.source = f"{files}"
elif count == 5:
    self.ids.space_5.shadow_color = "red"
    self.ids.display_image_5.source = f"{files}"
elif count == 6:
    self.ids.space_6.shadow_color = "red"
    self.ids.display_image_6.source = f"{files}"
elif count == 7:
    self.ids.space_7.shadow_color = "red"
    self.ids.display_image_7.source = f"{files}"
elif count == 8:
    self.ids.space_8.shadow_color = "red"
    self.ids.display_image_8.source = f"{files}"
elif count == 9:
    self.ids.space_9.shadow_color = "red"
    self.ids.display_image_9.source = f"{files}"
else:
    self.ids.space_10.shadow_color = "red"
    self.ids.display_image_10.source = f"{files}"
    count += 1


else:
    print(f"image predicted to not have TB")
    if count == 1:
        self.ids.space_1.shadow_color = "green"
        self.ids.display_image_1.source = f"{files}"
    elif count == 2:
        self.ids.space_2.shadow_color = "green"
        self.ids.display_image_2.source = f"{files}"
    elif count == 3:
        self.ids.space_3.shadow_color = "green"
        self.ids.display_image_3.source = f"{files}"
    elif count == 4:
        self.ids.space_4.shadow_color = "green"
        self.ids.display_image_4.source = f"{files}"
    elif count == 5:
        self.ids.space_5.shadow_color = "green"
        self.ids.display_image_5.source = f"{files}"
    elif count == 6:
        self.ids.space_6.shadow_color = "green"
        self.ids.display_image_6.source = f"{files}"
    elif count == 7:
        self.ids.space_7.shadow_color = "green"
        self.ids.display_image_7.source = f"{files}"
```

```
                elif count == 8:
                    self.ids.space_8.shadow_color = "green"
                    self.ids.display_image_8.source = f"{files}"
                elif count == 9:
                    self.ids.space_9.shadow_color = "green"
                    self.ids.display_image_9.source = f"{files}"
                else:
                    self.ids.space_10.shadow_color = "green"
                    self.ids.display_image_10.source = f"{files}"
                count += 1

                continue

            except Exception as e:
                print(f"there was an error: {e}")

        multiple_image_verify()


class TBNotDetectedScreen(Screen):
    def on_enter(self):
        global prediction
        self.ids.prediction_accuracy.text = f"Patient has {round(prediction[0][0] * 100, 2)}% chance of
TB"
```

```
    def save_patient_record(self):
        TBDetectedScreen.save_patient_record(self)



class ProgressBarScreen(Screen):
    pass



class CircularProgressBar(AnchorLayout):
    set_value = NumericProperty(0)
    value = NumericProperty(0)
    bar_color = ListProperty([20 / 255, 115 / 255, 233 / 255])
    bar_width = NumericProperty(10)
    text = StringProperty("0%")
    duration = NumericProperty(1.5)
    counter = 0

    def __init__(self, **kwargs):
        super(CircularProgressBar, self).__init__(**kwargs)
        Clock.schedule_once(self.animate, 0)

    def animate(self, *args):
        Clock.schedule_interval(self.percent_counter, self.duration / self.value)

    def percent_counter(self, *args):
        if self.counter < self.value:
            self.counter += 1
            self.text = f"{self.counter}%"
            self.set_value = self.counter
        else:
            Clock.unschedule(self.percent_counter)
            #ProgressBarScreen.reload_screen(self)



ScreenManager = ScreenManager()
ScreenManager.add_widget(HomeScreen(name='Home'))
ScreenManager.add_widget(UploadScreen(name='Upload'))
ScreenManager.add_widget(TBDetectedScreen(name='Detected'))
ScreenManager.add_widget(TBNotDetectedScreen(name='Unverified'))
ScreenManager.add_widget(ProgressBarScreen(name='Progress'))
ScreenManager.add_widget(MultipleImagesScreen(name='Multiple'))



class TBAPP(MDApp):

    def build(self):
        self.title = "Tuberculosis Detector"
        self.icon = "x-ray-with-neon-colors.jpg"
        self.theme_cls.theme_style_switch_animation = True
        self.theme_cls.theme_style_switch_animation_duration = 0.8
        self.theme_cls.theme_style = "Light"
        self.theme_cls.primary_palette = "Darkgray"
        file = Builder.load_file("TBDetection.kv")

        return file

    def switch_mode(self):
        self.theme_cls.primary_palette = (
            "Darkgray" if self.theme_cls.primary_palette == "Cyan" else "Cyan"
        )
        self.theme_cls.theme_style = (
            "Dark" if self.theme_cls.theme_style == "Light" else "Light"
        )

    def show_alert_dialog(self, *args):
        MDDialog(
            # ----------------------------Icon-----------------------------
            MDDialogIcon(
                icon="information",
            ),
            # -----------------------Headline text-------------------------
            MDDialogHeadlineText(
                text="20/7cs/00231",
            ),
            # -----------------------Supporting text-----------------------
            MDDialogSupportingText(
                text="Application developed by Adebisi Victor Omoikhefe "
                     "in partial fulfillment of the undergraduate program\n"
                     "B.sc computer science.\n\n"
                    "Supervisor: Dr. RM Isiaka.",
            ),

            # -----------------------------------------------------------
        ).open()

    def show_alert_dialog2(self, *args):
        MDDialog(
            # ---------------------------Icon----------------------------
            MDDialogIcon(
                icon="alert",
            ),
            # ----------------------Headline text------------------------
            MDDialogHeadlineText(
                text="critical",
            ),
            # ----------------------Supporting text----------------------
            MDDialogSupportingText(
                text="Please cross check results from this application\n "
                     "As this is an artificial intelligence model with 98% accuracy,\n"
                     "some of it's results might be false.\n\n"
                "",
            ),

            # -------------------------------------------------------------
          ).open()


    if __name__ == '__main__':
       TBAPP().run()
```