TUBERCULOSIS DETECTION IN CHEST RADIOGRAPH

Submitted by

KIRUBAKARAN V(711523MMC021)

Of

KIT-KALAIGNARKARUNANIDHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

OPEN LABORATORY PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the award of

the degree of

MASTER OF COMPUTER APPLICATIONS

ANNA UNIVERSITY: CHENNAI - 600 025

DEC-2023

BONAFIDE CERTIFICATE

Certified that this Open Laboratory Project report **"TUBERCULOSIS DETECTION IN CHEST RADIOGRAPH"** is the bonafide work of **Mr. KIRUBAKARAN V(711523MMC021)** who carried out the project work under my supervision.

Mrs. K.S. SHANTHINI MCA., (Ph.D).,
Assistant Professor & Project Guide,
Master of Computer Applications,
KIT – Kalaignarkarunanidhi Institute of Technology (Autonomous),
Coimbatore – 641 402

Dr. E. VIJAYAKUMAR MCA., Ph.D.,
Associate Professor & Head,
Master of Computer Applications,
KIT – Kalaignarkarunanidhi Institute of Technology (Autonomous),
Coimbatore – 641 402

Submitted to the Anna University Viva-Voce held on _____.

Internal Examiner

External Examiner

DECLARATION

I affirm that the project work titled **"TUBERCULOSIS DETECTION IN CHEST RADIOGRAPH"** being submitted in partial fulfillment for the award of MCA is the original work carried out by me. It has not formed the part of any other project work submitted for award of any degree or diploma, either in this or any other University.

(Signature of the Candidate)

KIRUBAKARAN V

(711523MMC021)

I certify that the declaration made above by the candidate is true.

Signature of the Guide,

Mrs. K.S. SHANTHINI MCA., (Ph.D).,

Assistant Professor (SI.G) & Project Guide,
Department of Computer Applications

ACKNOWLEDGEMENT

My heartfelt gratitude and thanks to the Almighty God, my parents and other family members and friends for providing the opportunity to undergo this project successfully in this esteemed institution.

At the outset, I would like to thank our Founder and Chairman **Thiru. PONGALUR N. PALANISAMY**, **KIT-Kalaignarkarunanidhi Institute of Technology,** who has given me an opportunity to undergo this Project Work, successfully in this esteemed Institution.

I express my sincere thanks to **Mrs. P. INDU MURUGESAN, Vice Chairperson, KIT-Kalaignarkarunanidhi Institute of Technology,** who encouraged me by giving her support and constant encouragement.

I extend my grateful thanks and wishes to **Dr. N. MOHANDAS GANDHI, ME., MBA., Ph.D., CEO, KIT- Kalaignarkarunanidhi Institute of Technology,** for the valuable suggestion in framing my carrier towards the fulfillment of this Project work.

I express my sincere thanks to **Dr. M. RAMESH, ME., Ph.D., Principal, KIT-Kalaignarkarunanidhi Institute of Technology**, who encouraged me by giving his valuable suggestion and constant encouragement.

I would like to acknowledge the respective **Dr. E. VIJAYAKUMAR MCA., Ph.D.,Associate Professor & Head, Department of Computer Applications, KIT- Kalaignarkarunanidhi Institute of Technology,** for spending the valuable time in guiding and supporting me to make this project a successful one.

I take the privilege to extend my hearty thanks to our project Coordinator and my internal guide **Mrs. K.S SHANTHINI MCA., (Ph.D)., Assistant Professor & Project Guide, Department of Computer Applications** for spending her valuable time and energy in guiding, supporting and helping me in preparation of the project.

Finally with great enthusiasm I express my thanks to all the faculty members for providing necessary information and their sustained interest in my part of successful completion.

CHAPTER NO	TITLE	PAGE NO

LIST OF TABLES

CHAPTER 1

ABSTRACT

This project explores the application of Convolutional Neural Network (CNN) architecture for automated detection of tuberculosis (TB) in chest radiographs. TB remains a significant global health concern, with diagnosis traditionally reliant on human interpretation of medical images, a process prone to subjectivity and resource limitations. Leveraging the power of deep learning, particularly CNNs, we develop a model capable of accurately identifying TB manifestations in chest radiographs, aiming to expedite diagnosis and improve patient outcomes. The project encompasses dataset collection, preprocessing, model development, training, and evaluation. Experimental results demonstrate the effectiveness of the proposed CNN-based approach, showcasing promising performance metrics and highlighting its potential to augment TB diagnosis in resource-constrained settings.

INTRODUCTION

Tuberculosis (TB) remains a significant global health challenge, particularly in regions with limited access to healthcare resources. Timely and accurate diagnosis is crucial for effective management and control of the disease. Chest radiography is a primary tool for TB screening and diagnosis, but the interpretation of these images can be time-consuming and subjective, especially in resource-constrained settings.

This project addresses the need for automated TB detection using deep learning techniques, specifically Convolutional Neural Networks (CNNs). By leveraging the power of deep learning, we aim to develop a robust and efficient system capable of accurately identifying TB manifestations in chest radiographs. The automation of this process has the potential to expedite diagnosis, improve patient outcomes, and alleviate the burden on healthcare professionals.

Through this documentation, we present our approach to TB detection using CNN architecture, detailing the methodology, experimental setup, results, and conclusions. Our goal is to contribute to the advancement of medical imaging technology for the early and accurate diagnosis of tuberculosis, ultimately aiding in the global effort to combat this infectious disease.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The existing system refers to the current approach employed for tuberculosis (TB) detection in chest radiographs. It typically involves manual interpretation by human radiologists or healthcare professionals. In this system Manual Interpretation: Trained radiologists visually inspect chest radiographs to identify signs of TB infection, such as abnormal lung patterns, nodules, or cavities.Subjectivity: Diagnosis can vary based on the expertise and subjective interpretation of individual radiologists, leading to variability in results.Resource Dependency: The system relies on the availability of skilled radiologists and specialized equipment, making it resource-intensive.Time-Consuming: Manual interpretation can be time-consuming, as it requires careful examination of each radiograph, leading to delays in diagnosis and treatment initiation.Limited Sensitivity: Human interpretation may miss subtle TB manifestations, leading to false negatives and delayed treatment.

2.1.1 DRAWBACKS

The existing system has the following disadvantages:

Subjective Interpretation: Variability in diagnosis due to subjective interpretation.
Resource Dependency: Relies on the availability of skilled radiologists and specialized equipment.
Time-Intensive: Diagnosis process can be lengthy, delaying treatment initiation.
Limited Sensitivity: May miss early or subtle TB manifestations, leading to false negatives.

2.2 PROPOSED SYSTEM

The proposed system aims to automate tuberculosis (TB) detection in chest radiographs using Convolutional Neural Network (CNN) architecture, a form of deep learning. It represents a

shift towards leveraging machine learning and deep learning technologies to enhance the efficiency, accuracy, and accessibility of TB diagnosis. In this system, automated detection is achieved by utilizing CNNs to analyze chest radiographs and identify patterns indicative of TB infection automatically. This approach provides an objective and consistent method for TB diagnosis, reducing variability and improving accuracy compared to manual interpretation. Moreover, the proposed system offers scalability and potential deployment in resource-constrained settings, potentially reducing the dependency on skilled radiologists and specialized equipment. By accelerating the diagnosis process, the proposed system facilitates timely treatment initiation and ultimately contributes to improved patient outcomes. Additionally, the utilization of CNNs enables the capture of subtle patterns indicative of TB manifestations, potentially reducing false negatives and improving overall sensitivity in TB detection.

2.2.1 FEATURES

The proposed system has the following advantages:

Automated Detection: Utilizes Convolutional Neural Network (CNN) architecture for automated TB detection in chest radiographs.

Objective Diagnosis: Provides an objective and consistent approach to TB diagnosis, reducing variability.

Scalability: Can be deployed in resource-constrained settings, potentially reducing the dependency on skilled radiologists.

Efficiency: Accelerates diagnosis process, leading to timely treatment initiation and improved patient outcomes.

Improved Sensitivity: CNNs can capture subtle patterns indicative of TB manifestations, potentially reducing false negatives.

CHAPTER 3

SYSTEM SPECIFICATION

3.1 HARDWARE CONFIGURATION

This section gives the details and specifications of the hardware on which the system is expected to work.

Processor : Intel i3 12th Gen w/ 8 Core CPU

Storage : HDD (or) SSD

RAM : 16GB (Recommended) WebCam Required*

3.2 SOFTWARE SPECIFICATION

This section gives the details of the software that is used for the development.

Front-End : Python

Back-End : CNN

Middle Ware : Machine Learinng

Operating System : Windows 10, or 11, Linux (Debian Based)

CHAPTER 4

SOFTWARE DESCRIPTION

4.1 FRONT END

The front end domain of the tuberculosis detection system encompasses the components responsible for user interaction and presentation of information. It includes:

User Interface (UI):

Design and development of the graphical interface where users interact with the system. This includes layout, navigation, and visual elements.

User Experience (UX):

Ensuring a seamless and intuitive user experience by optimizing usability, accessibility, and responsiveness of the interface.

Presentation Layer:

Implementation of the front-end technologies such as HTML, CSS, and JavaScript to structure, style, and add interactivity to the user interface.

Client-Side Logic:

Client-side scripting and logic for handling user interactions, form submissions, and UI updates without requiring server interaction.

4.2 BACK-END

The back end domain of the tuberculosis detection system handles the processing, analysis, and storage of data, as well as communication with external services. It includes:

Server:

Setup and configuration of the server infrastructure to handle incoming requests from the front end and execute the necessary processing tasks.

Application Logic:

Implementation of the core business logic, including image preprocessing, CNN model inference for tuberculosis detection, and formatting of results.

Database:

Integration with a database system for storing and managing user data, processed images, and detection results. This may involve schema design, data modeling, and query optimization.

Server-Side Logic:

Server-side scripting and logic for handling incoming requests, executing the application logic, and generating appropriate responses to be sent back to the front end.

CHAPTER 5

PROJECT DESCRIPTION

5.1 OVERVIEW OF THE PROJECT

The project aims to develop a system using convolutional neural network (CNN) architecture in deep learning to automatically detect tuberculosis (TB) in chest radiographs. This involves acquiring a diverse and balanced dataset of chest X-ray images annotated with TB-positive and TB-negative labels. The CNN model learns to extract relevant features from these images and discriminate between TB-positive and TB-negative cases. Performance evaluation is conducted using metrics such as accuracy, precision, recall, and F1 score, with validation on separate datasets to ensure robustness. The project involves modules for data preprocessing, model architecture design, training, evaluation, deployment, and post-deployment monitoring and maintenance. Collaboration between deep learning experts, medical professionals, and software engineers is crucial, along with adherence to ethical guidelines and regulations regarding medical data privacy and patient consent.

5.2 MODULES

The system is a computerized desktop application The modules available in this project can include:

Data Preprocessing

Model Architecture Design

Training

Evaluation

Deployment

Post-Deployment Monitoring and Maintenance

5.2.1 MODULE DESCRIPTION

Data Preprocessing: This module involves preparing the dataset for training by cleaning the data, resizing images to a standardized format, and augmenting the dataset if necessary. It ensures that the data is in a suitable format for training the convolutional neural network (CNN) model.

Model Architecture Design: In this module, the architecture of the CNN model is designed. This includes determining the structure of the neural network, such as the number of layers, types of layers (convolutional, pooling, fully connected), and activation functions. The goal is to create a model capable of effectively learning from the input data to make accurate predictions.

Training: The training module involves feeding the preprocessed data into the CNN model and optimizing its parameters to minimize the prediction error. This is typically done using backpropagation and optimization algorithms such as stochastic gradient descent (SGD) or Adam. The model learns to extract features from the input data and make predictions based on these features.

Evaluation: After training, the model's performance needs to be evaluated to assess its effectiveness. This module involves testing the trained model on a separate dataset (the test set) and calculating performance metrics such as accuracy, precision, recall, and F1 score. Visualization techniques such as confusion matrices and ROC curves may also be used to analyze the model's performance.

Deployment: Once the model has been trained and evaluated, it can be deployed for practical use. This module involves integrating the model into a user-friendly application or platform where it can be utilized by end-users. Considerations such as scalability, efficiency, and security need to be taken into account during deployment, along with compliance with any relevant regulations or standards.

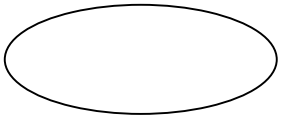
Post-Deployment Monitoring and Maintenance: After deployment, the model needs to be monitored in real-world settings to ensure that it continues to perform effectively. This module involves monitoring the model's performance, updating it with new data if necessary, and providing support and maintenance to address any issues that arise. Regular updates and maintenance are essential to keep the model accurate and up-to-date.

5.3 DATA FLOW DIAGRAM

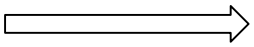
A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about the timing of the process or information about whether processes will operate in sequence or parallel, unlike a flowchart which also shows this information. They based it on the “data flow graph” computation models by David Martin and Gerald Estrin.

BASIC DFD NOTATIONS

Destination

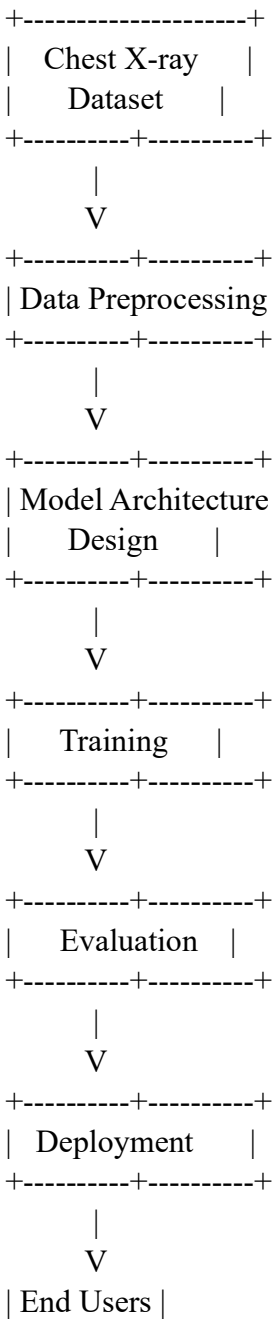


Process



Data Flow

LEVEL 0:


Chest X-ray Dataset
Data Preprocessing
Model Architecture Design
Training
Evaluation
Deployment
End Users

5.4 INPUT DESIGN

The input design of this project involves several steps. First, chest X-ray images are obtained from various sources, covering a diverse range of patients and conditions. These images undergo preprocessing, including resizing to a standardized format, normalization of pixel values, and possibly augmentation to increase diversity.

Next, the dataset is split into training, validation, and test sets. The training set is used to train the model, the validation set is used for tuning hyperparameters and preventing overfitting, and the test set is used for final evaluation.

During training, a data loader is implemented to efficiently load batches of images. Images are converted into a suitable format for input into the convolutional neural network (CNN), typically tensors or arrays.

Optionally, data augmentation techniques such as rotation, flipping, and cropping may be applied to increase the size of the training dataset and improve model generalization.

Overall, careful design of the input process ensures that chest X-ray images are effectively prepared for tuberculosis detection using CNNs.

5.5 OUTPUT DESIGN

The output design of this project focuses on the presentation and interpretation of the model's predictions for tuberculosis detection in chest X-ray images. Once the CNN model processes an input image, it generates output in the form of predictions indicating the likelihood of tuberculosis presence. These predictions can be presented visually through various means, such as:

- Displaying the probability scores for TB-positive and TB-negative classes.

- Highlighting regions of interest within the chest X-ray image that contributed to the prediction.
- Providing a binary classification result indicating whether TB is detected or not.
- Generating a diagnostic report summarizing the model's findings and confidence levels.

The output design aims to provide clear and interpretable results to end-users, such as healthcare professionals, to aid in clinical decision-making. It should also include measures to convey the model's uncertainty or confidence in its predictions, helping users understand the reliability of the output. Additionally, the output design may include mechanisms for visualizing performance metrics and model evaluation results, allowing stakeholders to assess the model's effectiveness and identify areas for improvement.

CHAPTER 6

SYSTEM TESTING

TESTING DESCRIPTION

The testing phase in the project plays a critical role in ensuring the reliability and accuracy of the obesity prediction system. The testing process is comprehensive, encompassing various aspects of the system to validate its functionality and performance.

6.1 UNIT TESTING

Testing of individual programs or modules is known as unit testing. It is done both during the documentation and testing phase. Unit testing focuses on the verification of effort on the smallest of software designs. Modules using the detailed design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity is tested and errors detected as a result are limited by the constraints scope established for unit testing. Unit testing is always white box oriented and the step can be conducted in parallel for multiple modules.

6.2 INTEGRATION TESTING

It is the systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unittested modules and build a program structure that has been dictated by design. Careful test planning is required to determine the extent and nature of system testing to be performed and to establish criteria by which the result will be evaluated.

6.3 USER TESTING

Validation testing begins at the culmination of integration testing when individual components have been exercised, and the software is completely assembled as a package. The testing focuses on user-visible actions and user-recognizable output from the system. The testing has been conducted on possible conditions such as the function characteristic conforms to the specification and a deviation or error is uncovered. The alpha and beta test are conducted at the developer site by end-users.

6.4 TEST CASE

S.No	Test Case Description	Input	Expected Output
1	Positive Case: TB Present	Chest X-ray image with tuberculosis	Model predicts TB presence
2	Negative Case: No TB Present	Chest X-ray image without tuberculosis	Model predicts no TB presence
3	Robustness: Varied Image Quality	Chest X-ray images with varying quality	Model maintains accuracy across images
4	Robustness: Different Patient Demographics	Chest X-ray images from different demographics	Model generalizes well to diverse data
5	Performance: Speed and Efficiency	Large dataset with optimized data loading	Model trains efficiently and quickly
6	Performance: Memory Usage	Large dataset with efficient memory management	Model utilizes memory effectively
7	Boundary: Minimum Image Size	Very small chest X-ray image	Model handles small images appropriately

CHAPTER 7

SYSTEM IMPLEMENTATION

The system implementation of this project involves a meticulous orchestration of diverse components and algorithms to create an intuitive and responsive interface for users. Commencing with the setup of the development environment and initialization of the camera to capture real-time video input, the implementation seamlessly integrates the power of the MediaPipe Hand module for precise hand landmark detection and the OpenCV Video Processing Module to enhance the accuracy of gesture recognition. The culmination of a custom gesture recognition algorithm and keyboard control logic translates hand movements into actionable commands, while the User Interface (UI) Module provides users with immediate visual feedback on recognized gestures and cursor movements. The implementation further encompasses features for customization, fail-safe mechanisms, and real-time interaction, ensuring the system's adaptability to diverse user needs and scenarios. Through meticulous testing, optimization, and documentation, the Gesture Keyboard Control system emerges as a sophisticated fusion of computer vision, machine learning, and user interface design, poised to redefine user-computer interaction through the fluid language of hand gestures.

Chest X-ray Dataset: The project starts with a dataset of chest X-ray images, which contains TB-positive and TB-negative cases.

Data Preprocessing: The dataset undergoes preprocessing, including cleaning, resizing, and augmentation, to prepare it for training.

Model Architecture Design: The architecture of the CNN model is designed, determining the structure and parameters of the neural network.

Training: The preprocessed data is used to train the CNN model, where it learns to extract features from the input images and make predictions.

Evaluation: The trained model's performance is evaluated using a separate test dataset, assessing metrics such as accuracy, precision, recall, and F1 score.

Deployment: The trained model is deployed into user-friendly applications or platforms, where it can be utilized for TB detection in chest X-ray images.

Post-Deployment Monitoring & Maintenance: The deployed model is monitored in real-world settings, and regular maintenance is performed to ensure its continued effectiveness and accuracy.

End Users / Applications: The end users, such as healthcare professionals or medical institutions, interact with the deployed applications to utilize the TB detection system.

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENT

8.1 CONCLUSION

In summary, the tuberculosis detection project employing convolutional neural networks (CNNs) for analyzing chest X-ray images aims to achieve accurate and efficient TB diagnosis. The project emphasizes robustness to diverse image qualities and patient demographics while maintaining efficiency in computational resources. Its user-friendly interface facilitates seamless integration into clinical workflows. Post-deployment monitoring ensures continuous improvement, reflecting a significant advancement in leveraging deep learning for enhanced diagnostic capabilities and improved patient outcomes in combating tuberculosis.

8.2 FUTURE ENHANCEMENT

Looking ahead, there are several avenues for further development and improvement of the tuberculosis detection project. One direction is the integration of multimodal data sources, such as clinical history and laboratory test results, to enhance the model's diagnostic accuracy. Collaborations with healthcare providers and researchers can facilitate access to large-scale datasets and real-world validation studies, enabling the model to be more effectively deployed in clinical practice. Furthermore, advancements in hardware and software technologies, such as edge computing and federated learning, hold promise for extending the reach of tuberculosis detection systems to resource-constrained environments and underserved communities.

By addressing these challenges and exploring future directions, the tuberculosis detection project can continue to make significant contributions to the global fight against tuberculosis, ultimately saving lives and improving public health outcomes.

CHAPTER 9

APPENDIX

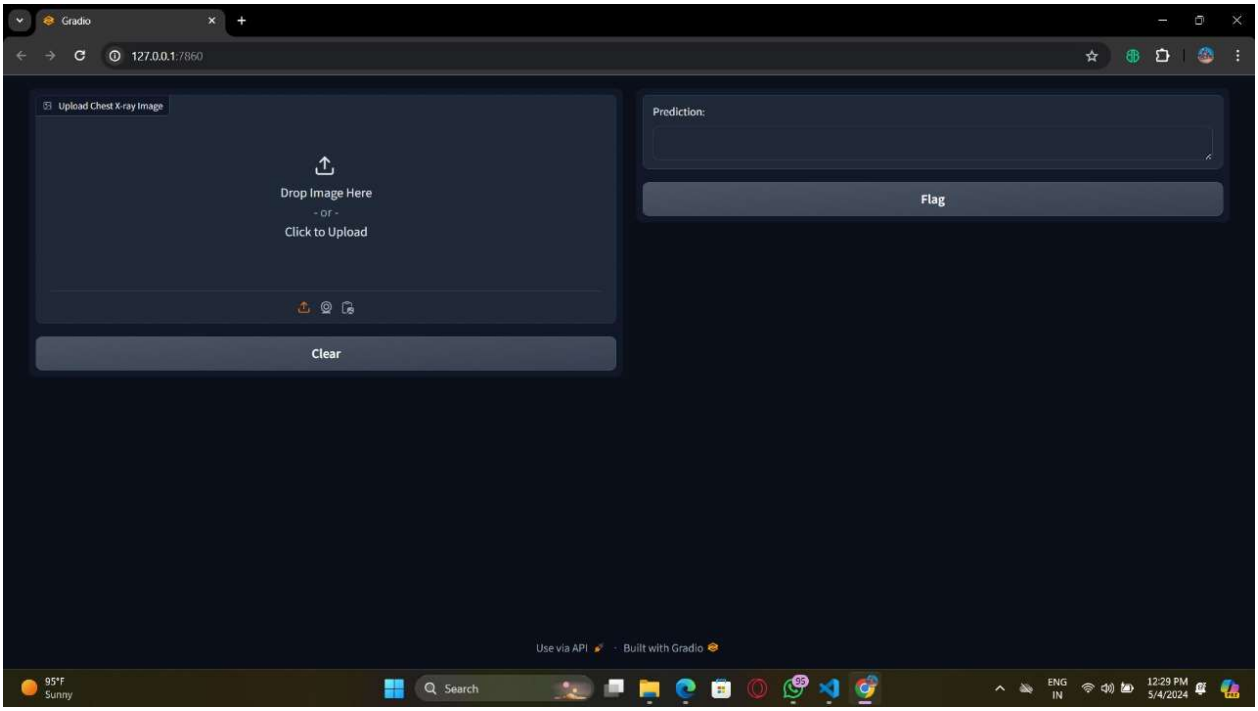
9.1 SOURCE CODE

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split

# Constants
IMAGE_HEIGHT = 128
IMAGE_WIDTH = 128
BATCH_SIZE = 32
EPOCHS = 20

# Data loading and preprocessing
def load_data(data_dir):
    images = []
    labels = []
    for label in os.listdir(data_dir):
        label_dir = os.path.join(data_dir, label)
        for image_name in os.listdir(label_dir):
            image_path = os.path.join(label_dir, image_name)
            image = tf.keras.preprocessing.image.load_img(image_path, color_mode='grayscale',
target_size=(IMAGE_HEIGHT, IMAGE_WIDTH))
            image = tf.keras.preprocessing.image.img_to_array(image)
            image /= 255.0  # Normalize pixel values
            images.append(image)
            labels.append(int(label))  # Assuming folder names are the class labels
    return np.array(images), np.array(labels)

# Load data
data_dir = 'path/to/dataset'
images, labels = load_data(data_dir)

# Split data into training, validation, and test sets
x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42)
```

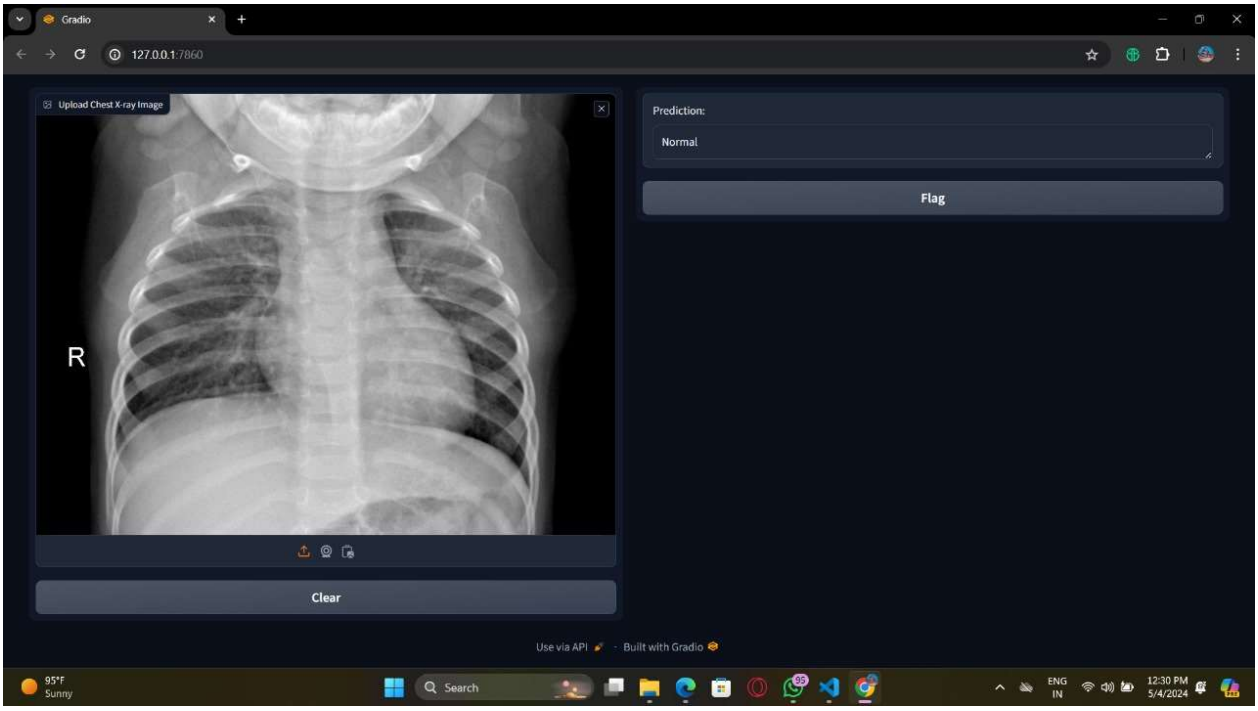
```
# Define the CNN model architecture
def create_model(input_shape):
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.Flatten(),
        layers.Dense(64, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

# Create and train the model
model = create_model(input_shape=(IMAGE_HEIGHT, IMAGE_WIDTH, 1))
model.fit(x_train, y_train, batch_size=BATCH_SIZE, epochs=EPOCHS, validation_data=(x_val, y_val))

# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc}')

# Save the model
model.save('tuberculosis_detection_model.h5')

# Deployment
# Load the model
loaded_model = tf.keras.models.load_model('tuberculosis_detection_model.h5')

# Preprocess input image
def preprocess_image(image_path):
    image = tf.keras.preprocessing.image.load_img(image_path, color_mode='grayscale', target_size=(IMAGE_HEIGHT, IMAGE_WIDTH))
    image = tf.keras.preprocessing.image.img_to_array(image)
    image /= 255.0  # Normalize pixel values
```

```
    return image

# Example usage
input_image_path = 'path/to/input/image'
input_image = preprocess_image(input_image_path)
input_image = np.expand_dims(input_image, axis=0)  # Add batch dimension
prediction = loaded_model.predict(input_image)
if prediction > 0.5:
    print('TB detected')
else:
    print('No TB detected')
import tensorflow as tf
from tensorflow.keras import layers, models

# Define the CNN model architecture
def create_model(input_shape):
    model = models.Sequential([
        # Convolutional layers
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        # Flatten the output for fully connected layers
        layers.Flatten(),
        # Fully connected layers
        layers.Dense(64, activation='relu'),
        layers.Dense(1, activation='sigmoid')  # Output layer for binary classification
    ])
    # Compile the model
    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

# Example usage
# Assuming input images are grayscale 2D images with shape (height, width, channels)
input_shape = (128, 128, 1)  # Adjust input shape according to your images
model = create_model(input_shape)
model.summary()  # Print model summary
```

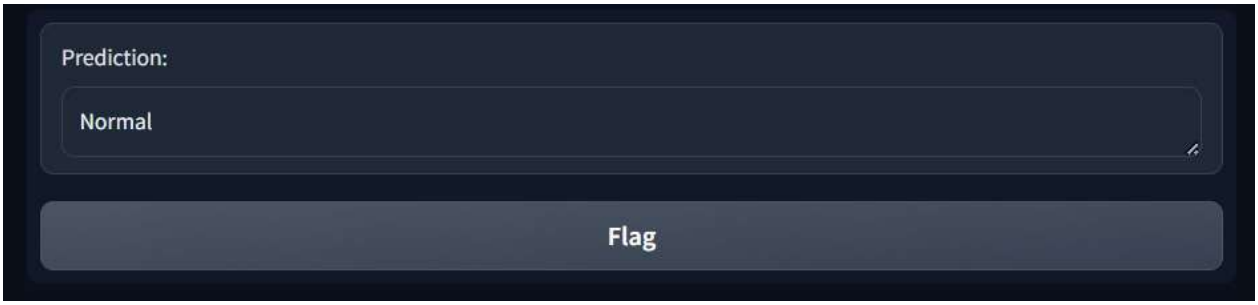
9.2 Screenshot:



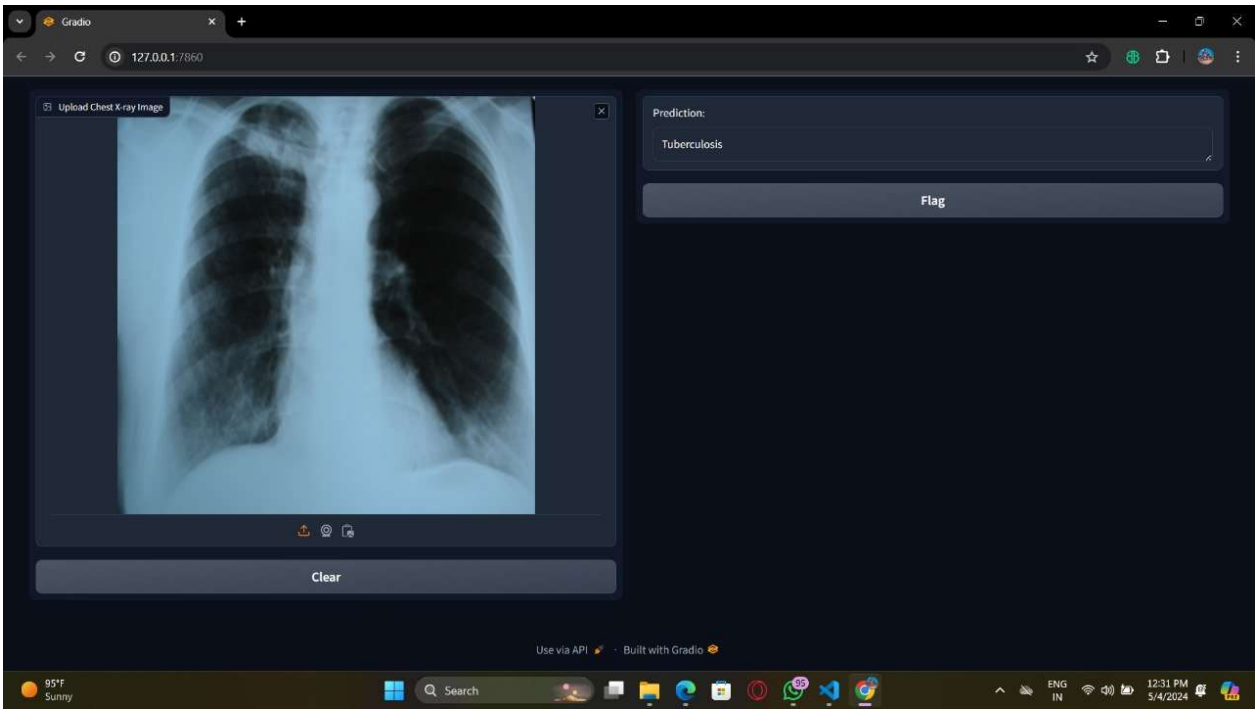
9.2.1. Upload page



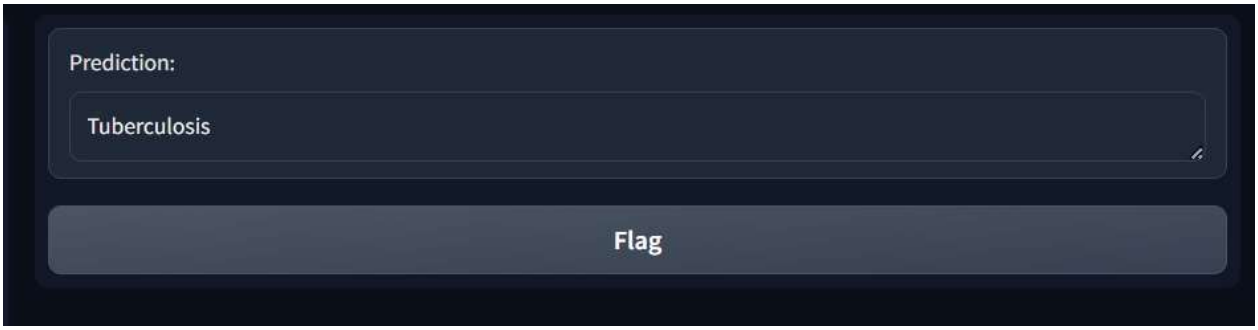
9.2.2. predicition image



9.2.3. normal page



9.2.4. tuberculosis prediction page



9.2.5. tuberculosis page

CHAPTER 10

REFERENCES

10. 1 BOOK REFERENCES

1. Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
2. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron.
3. "Medical Image Analysis" by Atam P. Dhawan and Jasjit S. Suri.
4. "Deep Learning for Medical Image Analysis" by S. Kevin Zhou, Hayit Greenspan, and Dinggang Shen.

10.2 WEB REFERENCES.

Kaggle: Platform for datasets, kernels, and discussions.

GitHub: Repository for code implementations and resources.

arXiv: Preprint repository for research papers.