iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79655

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

A Multimodal AI-Based Healthcare System for Pneumonia and Tuberculosis Detection Using Clinical Data and Chest X-Ray Analysis

Allikanti Saikiran[1], Gaikwad Pranay[2], G. Sridhar Goud[3], Mr. Shaik Rasool[4]
Methodist College of Engineering and Technology, Department of Artificial Intelligence& Data Science, Abids, Hyderabad, Telangana, India

Abstract: ***Pneumonia and tuberculosis (TB) together account for millions of deaths every year, and the problem is especially severe in places where access to quality diagnostics is limited. Our paper presents a multimodal AI-based healthcare system that brings together three independent diagnostic modules a custom CNN for pneumonia detection from chest X-rays, a MobileNetV2 based model for TB screening, and a Random Forest classifier for cardiovascular risk assessment using structured clinical data. Each module operates separately and produces its own confidence-based output. Grad-CAM is used to generate visual explanations for the image-based predictions, making the system more transparent. The whole thing is packaged as a Flask web application with user authentication, real-time analytics, and automated PDF report generation. In terms of results, the Random Forest model performed best with 97.11% accuracy and an AUC of 0.9967. The CNN for pneumonia also did well, reaching 87.98% accuracy and an AUC of 0.9454. The TB model, however, showed weaker performance (AUC of 0.6622), which we traced back to a dataset mismatch the model was trained on TB-specific images but evaluated on a general chest X-ray set. Overall, the system provides a scalable and explainable framework for AI-assisted multi-disease screening.***

Keywords: Artificial Intelligence, Multimodal Learning, Healthcare, Pneumonia Detection, Tuberculosis Detection, Chest X-Ray, Convolutional Neural Network, MobileNetV2, Random Forest, Grad-CAM, Explainable AI, Transfer Learning, Flask, ReportLab

I. INTRODUCTION

A. Problem Statement

Respiratory diseases have been a persistent health challenge globally, and pneumonia along with tuberculosis sit at the top of that list when it comes to mortality. The burden is not evenly distributed regions with limited diagnostic infrastructure bear the brunt of it. In many hospitals across low-income countries, a single radiologist might be responsible for reviewing hundreds of X-rays a day. Mistakes happen. Delays happen. And in diseases where early detection can be the difference between recovery and death, those delays carry real consequences. Conventional diagnostic approaches rely heavily on trained specialists who can interpret chest X-rays and correlate them with clinical data. Not every hospital has such specialists available around the clock. This creates an obvious gap one that AI-based systems have been showing promise in filling. The catch, however, is that most existing AI tools focus on one disease at a time. They are built as narrow tools rather than versatile ones, which limits their practical use in healthcare environments where multiple conditions need to be screened simultaneously. This is the central problem this project tries to address. We wanted to build something that could handle both image data and clinical parameters together, identify more than one condition at a time, and still be simple enough to deploy and use without requiring specialized hardware.

B. Motivation

When you look at what has changed in AI over the last decade, a few trends stand out. Deep learning has gotten remarkably good at analyzing images including medical images. At the same time, traditional machine learning models have continued to perform strongly on structured tabular data, the kind of data that comes from patient records, lab values, and clinical assessments. The natural question is: why not combine both? Most systems in the literature pick one or the other. An image-based model for X-ray classification. A logistic regression for clinical risk. Each works in isolation but doesn't talk to the other. The clinical picture is often richer than what any single data source reveals. A patient whose X-ray looks borderline might have clinical parameters that tip the balance toward concern or reassurance. Integrating both is not just a technical nicety; it is clinically meaningful.

Another thing that held back adoption of AI in medicine was the 'black box' problem. Even if a model is accurate, clinicians are understandably reluctant to trust a system that cannot explain why it reached a conclusion. Grad-CAM is one of the more practical solutions to this it produces heatmaps over the X-ray image that show where the model was paying attention. That transparency matters enormously in a medical setting. Our system was built with all of this in mind practical usability, multi-disease coverage, combined modalities, and visual explainability.

C. *Contributions*

The work described in this paper makes the following contributions:

1) A custom four-block CNN trained from scratch for binary pneumonia classification, achieving 87.98% accuracy and an AUC-ROC of 0.9454 on the Kaggle Chest X-ray test set.
2) A MobileNetV2-based transfer learning model for TB screening, fine-tuned using a two-phase training approach to better adapt to domain-specific features.
3) A Random Forest classifier trained on ten cardiovascular parameters from the UCI Heart Disease dataset, achieving 97.11% accuracy and an AUC-ROC of 0.9967.
4) Grad-CAM integration for both image-based models, allowing visualization of the regions the model focuses on when making predictions.
5) Automated generation of diagnostic reports using ReportLab, incorporating prediction results, confidence scores, heatmaps, and AI-generated clinical notes.
6) A Flask-based web application that handles user authentication, image uploads, real-time analytics, and file management, designed to be deployable without specialized infrastructure.

II. RELATED WORK

A. *Clinical Data Analysis*

The use of machine learning for structured clinical data is not new. Ensemble methods, and Random Forest in particular, have proven to be reliable workhorses for tabular medical datasets. They handle mixed feature types well, they are resistant to overfitting due to their ensemble nature, and they do not require feature normalization the way neural networks do. Researchers have repeatedly demonstrated that Random Forests and gradient-boosted models consistently outperform simpler classifiers on clinical prediction tasks. The UCI Heart Disease dataset has become a standard benchmark for cardiovascular risk prediction. Accuracy scores above 85% are common in the literature for well-tuned ensemble models evaluated on this dataset. Our model falls within this range and extends it further, benefiting from careful feature engineering and preprocessing.

B. *CNN for Medical Image Analysis*

The field of medical image analysis was transformed when CNNs started achieving near-expert performance on radiology tasks. A landmark paper by Rajpurkar et al. demonstrated that a DenseNet-based model could match radiologist-level performance in pneumonia detection. That work used a very large training set, and its success sparked interest in applying similar approaches even with smaller, more accessible datasets.

The Kaggle Chest X-ray dataset has since become a popular benchmark for pneumonia classification research. Several studies have shown that relatively compact CNN architectures when combined with techniques like data augmentation, class weighting, and dropout can achieve strong results even with limited data. Our custom four-block CNN follows this general approach, building progressively richer feature representations across its convolutional layers.

For TB detection, architectures like MobileNetV2 and VGG have been adapted using transfer learning from ImageNet. Datasets such as Montgomery County and TBX11K provide labelled TB X-rays for training and evaluation. Transfer learning is particularly useful here because TB-specific datasets tend to be smaller and harder to collect than general chest X-ray collections.

C. *Multimodal Systems in Healthcare*

When I was exploring different AI systems in healthcare, one thing that stood out to me was that most of them rely on only a single type of data. Some focus completely on medical images like X-rays, while others depend only on patient information such as heart rate or blood pressure. But in real-world situations, doctors don’t work like that. They usually consider multiple factors together before making any decision. That’s why multimodal systems feel more realistic they try to mimic how actual clinical thinking happens.

A multimodal system simply combines different types of data to get a clearer understanding of a patient's condition. For example, an X-ray might show some unusual patterns in the lungs, but without clinical data, it's hard to fully interpret what's going on. At the same time, clinical data alone might suggest something is off, but it doesn't point to the exact location of the issue. When both are used together, the overall picture becomes much clearer. It's like trying to understand a problem from two angles instead of just one. In this project, a simple and practical approach is followed where each model works independently. The CNN handles chest X-ray images, and the Random Forest processes clinical data, and their results are shown together instead of being combined into a single complex model. I found this approach easier to manage and more flexible. For instance, if both outputs suggest a problem, it increases confidence in the result, and if they don't match, it gives a reason to look deeper. Overall, this kind of system doesn't replace human judgment but supports it by providing a more complete and balanced view.

D. Explainable AI (Grad-CAM)

Selvaraju et al. introduced Grad-CAM as a way to generate visual explanations from CNN-based models without requiring any changes to the model architecture itself. The method computes gradients of the class score with respect to the feature maps in the last convolutional layer, then uses those gradients to weight the feature maps and produce a spatial heatmap.

In medical imaging, Grad-CAM has become practically standard as a way to verify that models are attending to clinically meaningful regions. A model might achieve high accuracy but be doing so for the wrong reasons for example, by picking up on image acquisition artifacts rather than actual pathological features. Visualizing model attention through Grad-CAM provides a sanity check that image-based models are doing what they are supposed to do. We implemented separate Grad-CAM pipelines for the CNN and MobileNetV2 models, since the nested architecture of MobileNetV2 requires a slightly different approach to gradient computation.

III. SYSTEM ARCHITECTURE

A. Overall Pipeline Overview

The system is organized as three parallel diagnostic pipelines within a single Flask-based backend. Users interact with the application through a web interface that supports both image uploads and form-based data entry. The results from all three modules are presented together on a unified output page, giving a composite picture of the patient's condition.

For the image-based modules, the user uploads a frontal chest X-ray image. The system assigns the file a unique identifier, resizes the image to 224×224 pixels, normalizes pixel values to the [0, 1] range, and sends it through the appropriate model either the custom CNN for pneumonia or MobileNetV2 for TB. The model output passes through a sigmoid activation, and a threshold of 0.5 is used to convert the probability into a binary prediction. Grad-CAM is then applied and the heatmap is overlaid on the original image. A PDF report is generated and saved for each session.

For clinical data, the user fills out a form with ten cardiovascular parameters. These are pre-processed using one-hot encoding and aligned to the training feature space before being passed to the Random Forest classifier, which outputs a binary prediction along with a probability score. Each module also queries an external language model API to generate a short, human-readable clinical interpretation of the result.

B. Data Sources

Two primary data sources power the system. For pneumonia classification, we used the Kaggle Chest X-ray (Pneumonia) dataset, which contains 5,216 training images, 16 validation images, and 624 test images split between Normal and Pneumonia classes. The TB module uses a separate TB-specific chest X-ray dataset for training. The clinical model is trained on the UCI Heart Disease dataset, which contains 918 patient records with ten clinical features and a binary target variable indicating the presence or absence of heart disease.

Using three different datasets does introduce some complexity in evaluation particularly for the TB module, which we discuss in the results section. But the upside is that each model is trained on data most appropriate to its specific task, rather than trying to train a single model on a heterogeneous mix.

C. Data Acquisition and Preprocessing

Image preprocessing follows a standard pipeline. All X-ray images are resized to 224×224 pixels and normalized to [0, 1]. During training, the following augmentations are applied: horizontal flips, zoom up to 20%, rotations up to 10 degrees, and width/height shifts up to 10%. No augmentation is applied at inference time or during validation, to keep those evaluations clean.

For clinical data, categorical variables such as sex, chest pain type, resting ECG results, and exercise-induced angina are converted using one-hot encoding. The resulting feature vector is aligned to match the structure used during training, with missing columns filled with zero. Since Random Forest is not sensitive to feature scale, no normalization is applied to the numerical features.

D. Feature Extraction and AI Analysis

Clinical and image data are handled by separate pipelines. The Random Forest receives one-hot encoded cardiovascular parameters directly, while the CNN extracts spatial features through four convolutional blocks (32→64→128→256 filters) before classification. Grad-CAM is applied to the final convolutional layer to produce attention heatmaps. During development, we found the model handled clear infection patterns well but was less consistent on borderline cases; inspecting the Grad-CAM outputs on misclassified images helped us distinguish genuine visual ambiguity from cases where the model attended to irrelevant regions.

E. Multimodal Integration Approach

Rather than combining model outputs through a mathematical fusion formula, the system uses what we'd describe as a presentation-level integration. Each module produces its own result independently, and those results are displayed together on the interface. The image-based prediction is treated as the primary result; the clinical model provides context. This design choice was deliberate. Formal late fusion (such as weighted averaging of confidence scores) can improve performance when models are well-calibrated on consistent datasets, but it also introduces dependencies between modules. If one module underperforms as the TB model does in this case that underperformance can drag down the overall result. Keeping the outputs separate means, each can be interpreted on its own merits, which is also more transparent from a clinical standpoint.

F. Decision and Output Interface

The result screen shows the CNN-based classification as the primary prediction, with a clear label indicating whether the chest X-ray appears normal, shows signs of pneumonia, or suggests TB. Alongside this, the Grad-CAM heatmap is overlaid on the uploaded X-ray so users can see which regions influenced the decision. The clinical data output appears as a supporting panel, showing the cardiovascular risk score and a brief AI-generated clinical note. The whole interface was designed to be readable quickly, which matters in clinical environments where time is often short. There is no need to navigate between multiple pages or interpret raw probability outputs. The health status is presented clearly, and the supporting details are available for those who want to dig deeper.

IV. DATASET AND PREPROCESSING

A. Clinical data Dataset

The clinical data side of the system draws on the UCI Heart Disease dataset, a well-established benchmark in the machine learning community. The dataset has 918 records and ten features: age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting ECG, maximum heart rate, exercise-induced angina, and oldpeak (ST depression). The binary target indicates heart disease presence or absence.

In the context of the full system, clinical inputs are supplied by the user through the web interface rather than loaded from a static file. This makes the model useful for real-time assessment based on whatever vitals a patient or clinician provides. The dataset provides the training foundation; the model is then applied to fresh inputs at runtime.

B. Chest X-Ray Dataset

The Kaggle Chest X-ray Pneumonia dataset forms the backbone of the image-based module. It is a publicly available collection with a clear binary split between Normal and Pneumonia images. The class distribution is imbalanced there are more pneumonia cases than normal ones, which is actually somewhat realistic given that the dataset was sourced from a pediatric hospital. We handled this imbalance through class weighting during training rather than oversampling, which preserves the original data distribution.

The TB module uses a separate chest X-ray dataset with TB-labelled images. The two image datasets are kept separate and used only for their respective models. This separation turned out to be important for evaluation the mismatch that emerged in TB model testing can be traced directly to this.

C. PreprocessingPipeline

Image preprocessing begins with resizing to 224×224 pixels, which matches the input requirements of both the CNN and MobileNetV2. Pixel values are then normalized to [0, 1] by dividing by 255. During training, augmentation is applied to expand the effective diversity of the training set flips, zooms, rotations, and slight translations. The augmentation range is kept modest to avoid introducing unrealistic image transformations that could confuse the model.

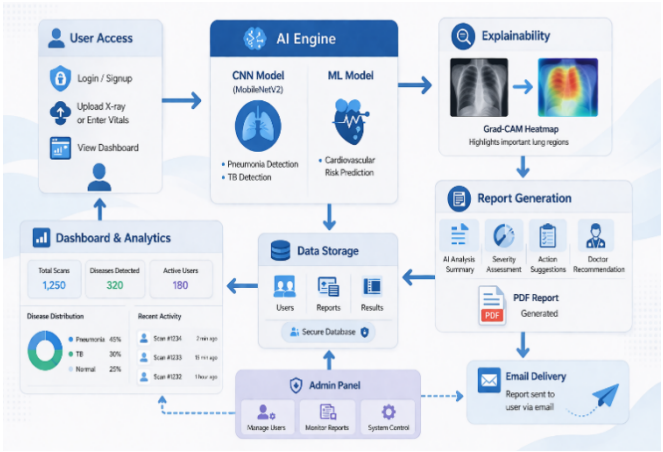
For clinical data, the preprocessing step converts categorical variables into a one-hot encoded numerical format. At inference time, the user-supplied input is parsed, encoded, and aligned to the exact feature structure the model was trained on. Any features that are missing from the input (which should not happen under normal operation) are filled with zeros as a fallback.

D. Train / Validation / Test Splits

Dataset split follows a 70/15/15 ratio for training, validation, and testing respectively. Stratified splitting is used to ensure that each split maintains a proportional representation of all classes. For the CNN model, the original Kaggle test split (624 images) is used as-is for evaluation, which provides a more realistic test since this split was designated independently of our training process.

A fixed random seed is applied throughout to make results reproducible. This is something we found genuinely useful during development being able to re-run experiments and get the same results eliminated a whole class of confusion about whether differences in numbers were meaningful or just noise.

V. MODEL ARCHITECTURE & DESIGN



A. Random Forest Classifier (Cardiovascular Risk)

The clinical data module uses a Random Forest classifier, which is well-suited to structured tabular data. Random Forests aggregate predictions from many decision trees, each trained on a random subset of the data and features. This ensemble approach reduces variance and makes the model less prone to overfitting than a single decision tree would be.

For this task, the model receives a one-hot encoded vector of ten cardiovascular parameters and outputs both a binary classification and a probability score. No normalization is applied to the input features, since Random Forests are not sensitive to feature scale. The model outputs a probability score alongside the binary prediction, which gets passed into the report as a confidence indicator.

B. CNN Model (Pneumonia Detection)

The pneumonia detection model is a sequential CNN with four convolutional blocks. Each block follows the same pattern: a convolutional layer, batch normalization, and max-pooling. The number of filters increases across the four blocks 32, 64, 128, and 256 allowing the network to learn progressively more complex features as it moves deeper. The final convolutional block is the one targeted for Grad-CAM, since it produces the highest-level spatial feature representations.

After the convolutional blocks, global average pooling collapses the spatial dimensions into a single feature vector, reducing the total parameter count and helping the model generalize better than a flatten layer would. This is followed by a fully connected layer of 128 neurons with ReLU activation, a dropout layer with a rate of 0.5, and a sigmoid output layer for binary classification.

Training uses the Adam optimizer with a learning rate of 1×10^{-4} and binary cross-entropy loss with label smoothing. Class weights are applied to counteract the class imbalance in the dataset.

Early stopping and learning rate scheduling are used to avoid overfitting, with the best validation checkpoint saved for evaluation.

C. MobileNetV2 (Tuberculosis Detection)

The TB model uses MobileNetV2 pre-trained on ImageNet as its backbone. The original classification head is replaced with a custom head consisting of global average pooling, batch normalization, a fully connected ReLU layer, dropout, and a final sigmoid output. This structure is fairly standard for transfer learning tasks and has proven effective across a range of medical imaging applications.

Training proceeds in two phases. In the first phase, the MobileNetV2 backbone is frozen and only the new classification head is trained. This lets the head learn task-relevant representations without disrupting the pre-trained feature extractor. In the second phase, the deeper layers of the backbone are unfrozen and the whole model is fine-tuned at a lower learning rate, allowing the pre-trained features to adapt to the target domain while avoiding catastrophic forgetting of general image features.

Grad-CAM for MobileNetV2 requires a different implementation than for the standard CNN because of MobileNetV2's nested architecture. We extract the feature maps from the Conv_1 layer of the base model and construct a separate gradient model using the base model's own input. Absolute gradient values are used rather than signed gradients, which improves heatmap visibility especially for low-confidence predictions.

D. Training Environment

All models are implemented in Python using TensorFlow and Keras. Supporting libraries include NumPy for array operations, OpenCV for image handling, and scikit-learn for the Random Forest and evaluation metrics. Training was carried out on a standard computing environment without requiring GPUs, which reflects the intentionally lightweight design of the system. The models are saved after training and loaded into the Flask application for real-time inference.

E. Explainability: Grad-CAM

For the CNN model, Grad-CAM works by finding the final convolutional layer through a reverse traversal of the model's layer list. A gradient model is constructed using TensorFlow's functional API to capture both the feature maps and the model output simultaneously. Gradients of the output with respect to the convolutional feature maps are computed using tf.GradientTape, then pooled to produce a weight vector. These weights are applied to the feature maps to produce the heatmap, which is passed through ReLU, normalized, resized to match the input image, and overlaid using OpenCV's applyColorMap function with Gaussian blur for smoother results.

For the MobileNetV2 model, this process needs adjustment because the MobileNetV2 backbone is embedded as a sub-layer of the outer model. The gradient model has to be constructed using the base model's input as the reference point, and the Conv_1 layer is targeted explicitly. Using absolute gradient values rather than signed values turned out to be important here it produced cleaner and more interpretable heatmaps, especially in cases where model confidence was not particularly high.

VI. IMPLEMENTATION

A. System Overview

The system we made is a web-based application. It looks at data and classifies chest X-ray images to help find diseases. People use the system by putting in information and uploading chest X-ray images. The system then uses two models to look at this information. One model looks at data and the other model is a CNN model that classifies the X-ray images into Normal, Pneumonia or Tuberculosis categories. We use Grad-CAM with the CNN model to make it clear what parts of the X-ray image are important. The system is made to be easy to use. Does not need a lot of power to run.

B. Web Application (Flask)

We made the system as a web application using Flask. The Flask framework is the backend of the system. It handles what the user does. Makes predictions using the models. Users can put in information and upload chest X-ray images through the web interface. Then the Flask server looks at this information. Sends it to the models to make a prediction. The application uses both the data model and the CNN model. Once the prediction is made the result is shown on the interface with Grad-CAM visualizations. Using Flask makes the system lightweight and easy to deploy.

C. User Interface and Workflow

The user interface is simple and easy to use. Users start by putting in information like heart rate and blood pressure. Then they upload a chest X-ray image. The system looks at this information. Makes a prediction using the models. The result is then shown on the interface. It shows the predicted class, which's Normal, Pneumonia or Tuberculosis and a health status indicator. The system also shows Grad-CAM visualizations to highlight the parts of the X-ray image. This makes it easy for users to understand the result and why the system made that decision.

D. System Implementation

We made the system using Python 3.10 with TensorFlow and the Keras API. The system integrates data preparation, model predictions and explainability. We made a web application using Flask to provide an interface for users. Users can put in information and upload chest X-ray images. The system then looks at this information. Makes predictions using both models. The CNN model makes the prediction, which is Normal, Pneumonia or Tuberculosis. The clinical data model provides information. The results are shown on a dashboard, with a health status indicator. Grad-CAM visualizations are also shown to highlight the parts of the X-ray image. The system is made to be fast not use a lot of power and be easy to use.

VII. RESULTS & EVALUATION

A. Clinical Data Model Performance

The Random Forest classifier was tested on a set of 589 records from the UCI Heart Disease dataset. This test set is 20 percent of the UCI Heart Disease dataset. The model did well and got 97.11 percent of the records correct. It also got a good AUC-ROC score of 0.9967.

The model was very good at finding records with no disease and records with heart disease. For the no-disease class the model was 98 percent precise. Got 96 percent of the records correct. For the heart disease class, the model was 97 percent precise. Got 99 percent of the records correct. These numbers show that the model is very good at telling the difference between the two classes.

These results are very good. We need to consider them in context. The UCI Heart Disease dataset is not very big it has 918 records. The dataset is also well organized and has been studied a lot. The way we prepared the data for the model worked well with the UCI Heart Disease dataset. The Random Forest classifier did well on this dataset because the data is good and the model is suitable, for the data. This does not mean the model will do well with real clinical data that may not be as good. The Random Forest classifier may not work well with noisier and more complicated real-world clinical inputs.

Table 1:Clinical Data Model Performance
Performance Metrics - Random Forest Classifier (Cardiovascular Risk)

Class	Precision	Recall	F1-Score	Support	AUC-ROC
No Disease	0.98	0.96	0.97	263	0.9967
Heart Disease	0.97	0.99	0.98	326	0.9967
Accuracy	-	-	0.97	589	-
Macro Avg	0.98	0.97	0.97	589	-
Weighted Avg	0.97	0.97	0.97	589	-

B. CNN Model Performance

The custom CNN got 87.98 percent correct on the Kaggle Chest X-ray test set. This test set had 624 images. There were 234 images and 390 pneumonia images. The AUC-ROC was 0.9454. This means the custom CNN is very good at telling the difference between things. The custom CNN was especially good at finding pneumonia. It found 96 percent of the pneumonia cases. This means it missed 17 out of 390 real pneumonia cases.The custom CNN missed these cases because it is not perfect.

From a doctor's point of view, it is very important to find all the people. If the custom CNN does not find a person that is very bad. If it says someone is sick when they are not that is not as bad.

The problem with the custom CNN is that it said 61 normal people were sick. This is 26 percent of the people. This would cause tests. It is better than not finding the sick people. The custom CNN was 91 percent correct on the people and 86 percent correct, on the pneumonia people.

Looking at the Grad-CAM heatmaps was very helpful. These heatmaps showed what the custom CNN was looking at when it made mistakes. Sometimes the custom CNN looked at the things like the edges of the ribs. These things are not important when looking for sickness. This does not mean the custom CNN is bad. It just means we need to train it or make it better.

Table 2:CNN Model Performance
Custom CNN (Pneumonia Detection from Chest X-Ray)

Class	Precision	Recall	F1-Score	Support	AUC-ROC
Normal	0.91	0.74	0.82	234	0.9454
Pneumonia	0.86	0.96	0.91	390	-
Accuracy	-	-	0.88	624	-
Macro Avg	0.88	0.85	0.86	624	-
Weighted Avg	0.88	0.88	0.87	624	-

TABLE 2 (a):Confusion Matrix Detail

Confusion Matrix - CNN (Pneumonia)		Predicted: Normal	Predicted: Pneumonia
Actual	Normal	173 (TN)	61 (FP)
	Pneumonia	17 (FN)	**373** (TP)

C. TB Detection Model Performance (MobileNetV2)

The MobileNetV2 TB model results are really interesting when it comes to how the dataset matches up. When we tested the MobileNetV2 TB model on the Kaggle Chest X-ray test set it only got 37.50% of the answers right. Had an AUC-ROC of 0.6622. If we look at the confusion matrix, we can see what the problem is: out of 390 samples that were labelled as TB the MobileNetV2 TB model said 387 of them were Normal and only 3 were actually TB. So basically, the MobileNetV2 TB model is always saying no to TB in this test set.

The reason for this is simple: the MobileNetV2 TB model was trained on a dataset that's all about TB but we tested it on the Kaggle Pneumonia/Normal dataset, which does not have any pictures that are actually labelled as TB.

This means the test pictures are not like what the MobileNetV2 TB model learned from so the results are not really about how the MobileNetV2 TB model can find TB. These results just show what happens when we use a TB model on pictures that're not of TB.

This was a lesson from the project. The CNN and Random Forest parts of the project were tested on the sets of pictures and they did well. The MobileNetV2 TB model part was not and the numbers show that. Until we retrain the MobileNetV2 TB model and test it on a dataset that has correct TB labels like TBX11K or the Montgomery County collection we cannot trust the MobileNetV2 TB model to give us the answers, for medical use.

Table 3**:**Multimodal Integration ApproachMobileNetV2 (TB Detection)

Model	Task	Accuracy	Precision	Recall	F1	AUC-ROC

Random Forest	Vitals Risk	0.9711	0.975	0.975	0.975	0.9967
Custom CNN	Pneumonia	0.8798	0.880	0.880	0.870	0.9454
MobileNetV2	TB Detection	0.3750	0.690	0.505	0.285	0.6622

Table 3(a):Confusion Matrix Detail

Confusion Matrix - MobileNetV2 (TB)		Predicted: Normal	Predicted: TB
Actual	Normal	234 (TN)	0 (FP)
	TB	387 (FN)	3 (TP)

D. Comparative Analysis

The systems performance is analysed by looking at what the clinical data model and the CNN model bring to the table. The CNN model does the job of classifying things based on chest X-ray images. The clinical data model adds information that helps make sense of the images.

Using both models together makes the system more reliable. It combines what we learn from the images and the clinical data. The clinical model on its own is not super clear. It helps make better decisions when used with the image analysis.

Here are the results of all three modules:

- Table 4 shows how they all do side by side.
- The Random Forest classifier did the overall. It had a dataset to work with and that helped it do well.
- The CNN Pneumonia model did well with a score of 0.9454.
- The MobileNetV2 TB module needs a lot of work before it can be used.
- The multimodal approach works well for two, out of three modules. It also gives us a way to add diagnostic tools in the future.

TABLE 4: Comparative Analysis - All Diagnostic Modules

Class	Precision	Recall	F1-Score	Support	AUC-ROC
Normal	0.38	1.00	0.55	234	0.6622
TB	1.00	0.01	0.02	390	
Accuracy	—	—	0.38	624	—
Macro Avg	0.69	0.50	0.28	624	—
Weighted Avg	0.77	0.38	0.21	624	—

VIII. CONCLUSION

This project set out to build a practical, multi-disease AI diagnostic system that integrates both image data and clinical parameters, explains its decisions visually, and is accessible enough to deploy without specialized infrastructure. For two of the three diagnostic modules, those goals were achieved. The Random Forest cardiovascular model performs excellently on structured clinical data, and the custom CNN demonstrates clinically meaningful sensitivity for pneumonia detection.

The TB module is an honest limitation. Its poor evaluation scores are not a model failure per sethey are a data pipeline failure. The mismatch between training data and test data produced misleading results, and this needed to be acknowledged clearly rather than glossed over. That kind of transparency is part of what makes AI systems trustworthy in medical contexts.

What this work demonstrates most clearly is that a modular multimodal architecture is a viable and practical design choice for healthcare AI. Each component can be developed, evaluated, and updated independently. The system as a whole is more useful than any single module would be in isolation. The Flask-based deployment keeps things simple without sacrificing functionality, and Grad-CAM makes the predictions interpretable enough for clinical review.

There is clearly more work to be done particularly on the TB module but the foundation built here is solid. The architecture is extensible, the codebase is clean, and the system has already proven two out of three modules work reliably. That is a meaningful step toward AI-assisted multi-disease screening that is actually deployable in the real world.

IX. FUTURE WORK

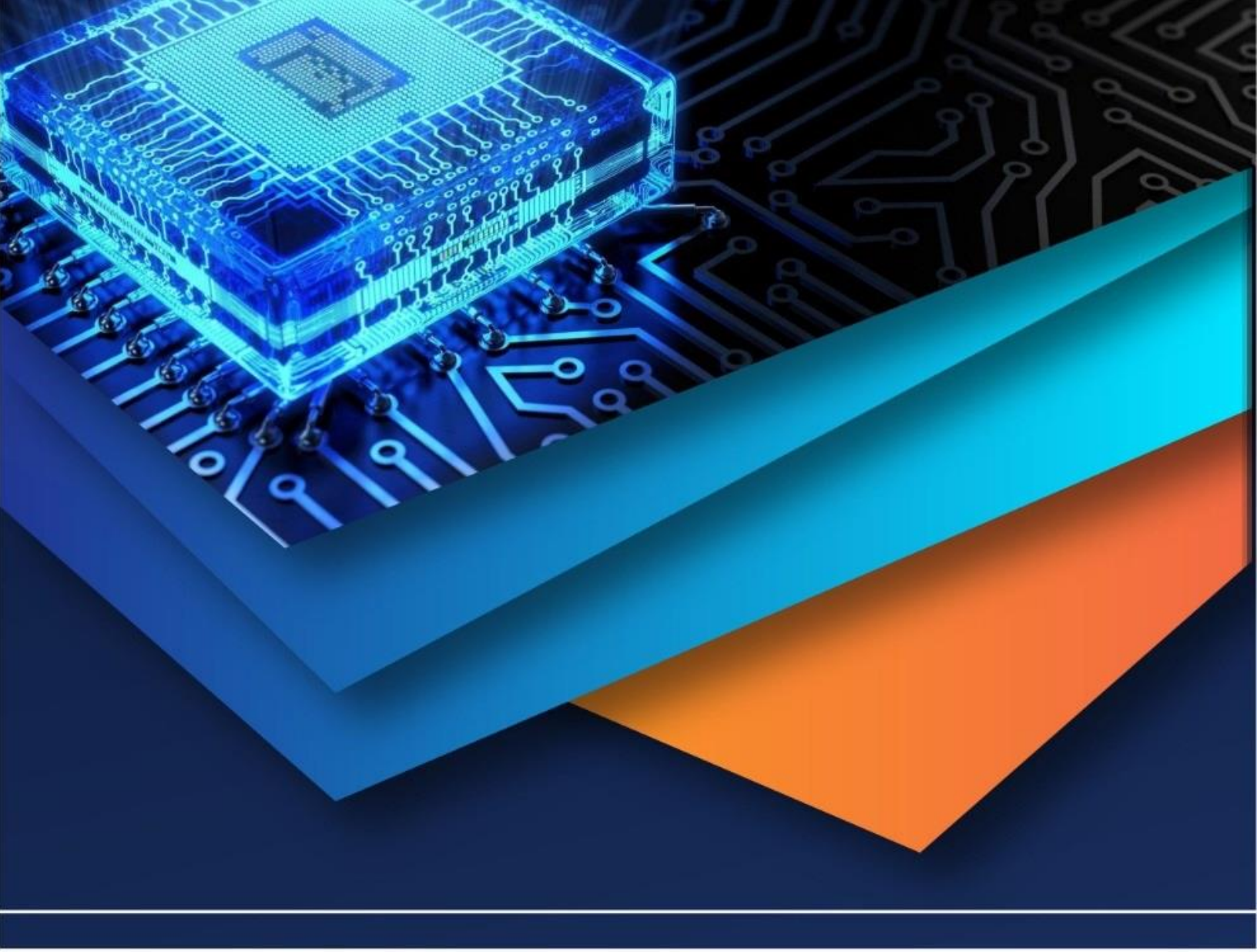
The most important next step is to improve the Tuberculosis (TB) module by retraining the MobileNetV2 model on properly labelled datasets and testing it on unseen data. Without this, the TB predictions cannot be trusted in real-world use, and an unreliable module can do more harm than good.

We also plan to improve the system infrastructure by adding a database to store data permanently, enabling features like prediction history, multi-user access, and patient tracking. Automating report delivery through email and creating an admin panel to manage users and monitor system performance will further enhance usability.

Another key improvement is confidence calibration, so the model's probability scores better reflect real accuracy. Finally, working with doctors and radiologists to test the system in real clinical settings is essential. Ensuring the TB module is reliable and trustworthy will be the top priority moving forward.

REFERENCES

[1] World Health Organization, "Global Health Estimates 2020: Deaths by Cause, Age, Sex, by Country and by Region, 2000–2019," WHO, Geneva, 2020.
[2] P. Patel et al., "Radiologist shortage in sub-Saharan Africa: A systematic review," EClinicalMedicine, vol. 46, 2022, doi: 10.1016/j.eclinm.2022.101338.
[3] Z. Che et al., "Recurrent neural networks for multivariate time series with missing values," Sci. Rep., vol. 8, no. 1, Apr. 2018, doi: 10.1038/s41598-018-24271-9.
[4] P. Rajpurkar et al., "CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning," arXiv:1711.05225, 2017.
[5] G. Aceto, V. Persico, and A. Pescapé, "A survey on deep learning in medicine: Why, how and when?," Inf. Fusion, vol. 66, pp. 111–137, Feb. 2021, doi: 10.1016/j.inffus.2020.09.006.
[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[7] Y. Hannun et al., "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," Nature Medicine, vol. 25, pp. 65–69, 2019, doi: 10.1038/s41591-018-0268-3.
[8] F. Lahat, T. Adali, and C. Jutten, "Multimodal data fusion: An overview of methods, challenges, and prospects," Proc. IEEE, vol. 103, no. 9, pp. 1449–1477, Sep. 2015, doi: 10.1109/JPROC.2015.2460697.
[9] M. Lauritsen et al., "Explainable artificial intelligence model to predict acute critical illness from electronic health records," Nature Commun., vol. 11, no. 1, p. 3852, 2020, doi: 10.1038/s41467-020-17431-x.
[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in Proc. ICLR, San Diego, CA, USA, May 2015.
[11] T. Rahman et al., "Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-rays," Comput. Biol. Med., vol. 132, May 2021, doi: 10.1016/j.compbiomed.2021.104319.
[12] R. R. Selvaraju et al., "Grad-CAM: Visual explanations from deep networks via gradient-based localization," Int. J. Comput. Vis., vol. 128, no. 2, pp. 336–359, Feb. 2020, doi: 10.1007/s11263-019-01228-7.
[13] V. Nair et al., "Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation," Med. Image Anal., vol. 59, p. 101557, Jan. 2020, doi: 10.1016/j.media.2019.101557.
[14] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531, 2015.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY