CCS 248

ARTIFICIAL NEURAL NETWORK

FINAL PROJECT

Development and Validation of a CNN Model Trained from Scratch for the Automated Detection and Classification of Pulmonary Tuberculosis via Chest X-Rays

Submitted by:

Aquino, Dallas
Buñag, Frederick Jibril
Corpes, Vincent Jr.

December 12, 2025

Aquino, Dallas
Buñag, Frederick Jibril
Corpes, Vincent Jr.

ARTIFICIAL NEURAL NETWORK

Final Project

CCS 248
December 12, 2025

Contents

Aquino, Dallas
Buñag, Frederick Jibril
Corpes, Vincent Jr.

Artificial Neural Network

Final Project

CCS 248
December 12, 2025

1 Problem

1.1 Introduction

The latest global statistics from the World Health Organization's 2024 report underscore the critical urgency of the tuberculosis crisis, revealing that in 2023, an estimated 10.8 million people fell ill with TB—the highest number ever recorded since global monitoring began—yet approximately 2.7 million of these cases went undiagnosed or unreported. This massive diagnostic gap is exacerbated by the limitations of human interpretation in chest radiography, where inter-observer agreement among radiologists is often only fair to moderate, with Kappa scores frequently ranging between 0.20 and 0.55, indicating significant inconsistency in identifying active disease. Furthermore, the sensitivity of human readers typically hovers between 70% and 85%, meaning nearly a third of active cases could be missed during initial screening, a problem that is particularly acute in challenging scenarios like smear-negative TB or HIV co-infection where radiographic signs are subtle. The consequences of these diagnostic failures are severe; an untreated individual can infect 10 to 15 others annually, and without treatment, the mortality rate for active TB approaches 50%, contributing to the 1.25 million deaths recorded in 2023 alone.

1.2 Proposed Solution

The proposed solution is the development powered by custom **Convolutional Neural Networks (CNNs)** trained from scratch. This system is designed to automate the screening and classification of pulmonary tuberculosis from chest X-ray images, specifically addressing the diagnostic gap in resource-limited settings.

1.2.1 Core Solution Architecture

The core of the solution involves two distinct deep learning architectures:

1. **Binary Detection Model**: A high-performance custom sequential CNN designed to distinguish between "Normal" and "TB-Positive" cases with high sensitivity, acting as a primary triage tool.
2. **Multi-Class Classification Model**: A VGG-inspired custom CNN designed to categorize cases into finer pathologies, including *Healthy*, *Sick but Non-TB*, *Active TB*, and *Latent TB*, to aid in differential diagnosis.

1.2.2 User Interface

Unlike standard black-box models, this system is integrated into a **user-friendly graphical interface (GUI)** that allows medical personnel to upload X-rays via drag-and-drop, providing immediate diagnostic probability scores and classification results without requiring technical expertise.

2 Objectives

The primary objective of this study is to design, implement, and validate custom CNN-based deep learning models for the automated detection and classification of pulmonary tuberculosis, and to deploy them within a functional web-based prototype.

Specific Objectives

1. **Dataset Aggregation and Preprocessing:** Construct a robust training pipeline using the Tuberculosis Chest X-ray Database (Qatar/Dhaka) and the TBX11K dataset. Apply rigorous preprocessing techniques including resizing, normalization, and class balancing (up-sampling/down-sampling) to handle data imbalances.

2. **Custom Architecture Development:** Design and implement two custom CNN architectures from scratch:

   - A 4-block Sequential model for binary detection.
   - A 5-block VGG-style model for multi-class classification.

   Both architectures will be optimized for medical image analysis using Global Average Pooling (GAP) and aggressive regularization techniques such as Dropout and L2 regularization.

3. **Model Training and Optimization:** Train these models using advanced strategies such as Categorical Focal Loss and dynamic learning rate scheduling to maximize performance metrics, with particular emphasis on high Sensitivity and Recall in binary detection to minimize false negatives in screening.

4. **Prototype Deployment:** Develop a proof-of-concept web interface capable of accepting chest X-ray uploads (PNG/JPEG) and providing clear diagnostic classifications (e.g., "Active TB") alongside confidence scores, bridging the gap between computational models and clinical application.

5. **Performance Benchmarking:** Evaluate the models using standard metrics (Accuracy, Precision, Recall, F1-Score, and AUC). Validate the binary detection model's reliability while analyzing the limitations of multi-class classification in the presence of severe class imbalance.

3 Dataset Choice and Validation

3.1 Tuberculosis (TB) Chest X-ray Database from Kaggle

3.1.1 Dataset Description

The Tuberculosis (TB) Chest X-ray Database is a significant collaborative resource created by researchers from Qatar University, the University of Dhaka, and medical professionals to support the development of reliable deep learning models for tuberculosis detection. Documented in a 2020 IEEE Access paper that demonstrated 98.3% classification accuracy, the dataset aggregates chest X-rays from diverse sources including the NLM, Belarus, and RSNA collections. It explicitly provides 3,500 normal images and 700 publicly accessible TB-positive images, while facilitating access to an additional 2,800 TB images via the NIAID TB portal, making it a robust tool for training diagnostic algorithms.

Group	**Source Dataset**	**Count**
Tuberculosis (TB)	Belarus TB Portal	306
	NLM	394
	TB Subtotal	*700*
Normal	NLM	406
	RSNA	3094
	Normal Subtotal	*3500*
Total Images		**4200**

Table 1: Distribution of Locally Available CXR Images (Public Data)

3.1.2 Dataset Source

The Tuberculosis (TB) Chest X-ray Database is a collaborative project developed by a research team from Qatar University (Doha) and the University of Dhaka (Bangladesh), along with collaborators from Malaysia and medical professionals from Hamad Medical Corporation. The researchers aggregated chest X-ray images from four primary external sources to create a comprehensive repository: the National Library of Medicine (NLM) in the U.S. (specifically the Montgomery and Shenzhen datasets), the Belarus Tuberculosis Portal, the NIAID TB portal program, and the RSNA pneumonia detection challenge. This consolidation resulted in a database containing 3,500 normal images and a large set of TB-positive images, designed to aid in the development of deep learning models for reliable TB detection.

3.1.3 Ethics and Privacy Check

The dataset enforces specific privacy and ethical protocols through a split-access model to comply with data-sharing restrictions. While 3,500 normal images and 700 TB-positive images are publicly accessible directly through the repository, an additional 2,800 TB images derived from the NIAID TB portal are restricted. Accessing these specific clinical images requires users to visit the NIAID website and sign a formal data-sharing agreement before they can download the DICOM files. Furthermore, users are ethically bound to credit the original creators by citing the specified 2020 IEEE Access paper ("Reliable Tuberculosis Detection using Chest X-ray...") when using the data for scientific purposes, and the license for the data files remains with the original authors.

3.1.4 Preprocessing

3.2 BX11k from Dataset Ninja

3.2.1 Dataset Description

TBX11K is a large-scale dataset released in 2019 designed to advance computer-aided tuberculosis diagnosis (CTD) by enabling the training of deep Convolutional Neural Networks (CNNs) for both classification and object detection. Comprising approximately 11,200 chest X-ray images, the dataset distinguishes itself from previous collections by providing precise bounding box annotations for TB areas rather than just image-level labels. It covers a diverse range of conditions categorized into healthy, active TB, latent TB, and sick but non-TB, with ground truth labels rigorously verified by experienced radiologists. By offering a volume of data roughly 17 times larger than the widely used Shenzhen dataset, TBX11K addresses the limitations of earlier baselines, facilitating the development of sophisticated models capable of localizing disease manifestations in complex, real-world clinical scenarios.

Group	**Class**	**Train**	**Val**	**Test**	**Total**
Non-TB	Non-TB Healthy	3000	800	1200	5000
	Sick & Non-TB	3000	800	1200	5000
TB	Active TB	473	157	294	924
	Latent TB	104	36	72	212
	Active & Latent TB	23	7	24	54
	Uncertain TB	0	0	10	10
Total		**6600**	**1800**	**2800**	**11200**

Table 2: Distribution of the TBX11K Dataset Across Train, Validation, and Test Sets

3.2.2 Dataset Source

The TBX11K dataset was originally introduced in the paper "Rethinking Computer-Aided Tuberculosis Diagnosis," presented at the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Developed by researchers Yun Liu, Yu-Huan Wu, and colleagues associated with Nankai University and InferVision, the dataset was released in late 2019 to address the scarcity of large-scale, annotated medical imagery for tuberculosis detection. Unlike previous datasets (such as Shenzhen or Montgomery), TBX11K provides a massive corpus of approximately 11,200 X-ray images split into training, validation, and testing sets. It was specifically designed to facilitate the training of deep learning models by including bounding box annotations for simultaneous classification and localization, with all images standardized to a 512×512 resolution to optimize computational efficiency for Convolutional Neural Networks (CNNs).

3.2.3 Ethics and Privacy Check

In terms of ethical usage and privacy, the TBX11K dataset is distributed under the Creative Commons Attribution 4.0 International (CC BY 4.0) license, which allows for the sharing and adaptation of the data for academic and research purposes, provided that appropriate credit is given to the original authors. While the dataset involves sensitive medical records, the public release implies that standard de-identification protocols were followed to strip Personally Identifiable Information (PII) from the X-ray metadata, protecting patient anonymity. Furthermore, the integrity of the data was secured through a rigorous ethical labeling process: annotations were generated and double-checked by radiologists with 5 to 10+ years of clinical experience to ensure consistency with "gold standard" hospital diagnoses, thereby reducing the risk of algorithmic bias caused by incorrect ground truth labels.

3.2.4 Preprocessing

The preprocessing technique involves a multi-stage pipeline designed to handle class imbalance and standardize inputs. First, the data is cleaned by mapping JSON annotations to images and filtering for four specific target classes, followed by a robust sampling strategy that explicitly downsamples majority classes (Healthy, Sick_but_Non-TB) to 3,000 samples and upsamples minority classes (Active/Latent TB) to 2,000 samples. The images are then resized to 224×224 pixels and normalized by rescaling pixel values by 1/255. To enhance model robustness, real-time data augmentation is applied exclusively to the training set, including random rotations (15°), horizontal/vertical shifts (10%), zooms (10%), and horizontal flips. Finally, the pipeline integrates a mixed-precision policy (float16) for efficiency and utilizes a custom Categorical Focal Loss ($\gamma = 2.0, \alpha = 0.25$) during training to further address class imbalance by focusing the model's attention on hard-to-classify examples.

4 Model Architecture Selection

4.1 For Detection

4.1.1 Network Structure

The model implements a Sequential architecture composed of four convolutional blocks followed by a dense classification head. Each convolutional block follows a repeating pattern of dual Convolutional layers, Batch Normalization, ReLU activation, generic Max Pooling, and increasing Dropout rates to manage feature abstraction and overfitting.

Block	Layer Type	Configuration	Output Details
1	Conv2D (x2)	32 Filters, 3×3 Kernel	L2 Reg (10^{-4}), ReLU
	BatchNormalization	-	-
	MaxPooling2D	2×2 Pool	-
	Dropout	Rate: 0.2	-
2	Conv2D (x2)	64 Filters, 3×3 Kernel	L2 Reg (10^{-4}), ReLU
	BatchNormalization	-	-
	MaxPooling2D	2×2 Pool	-
	Dropout	Rate: 0.3	-
3	Conv2D (x2)	128 Filters, 3×3 Kernel	L2 Reg (10^{-4}), ReLU
	BatchNormalization	-	-
	MaxPooling2D	2×2 Pool	-
	Dropout	Rate: 0.4	-
4	Conv2D (x2)	256 Filters, 3×3 Kernel	L2 Reg (10^{-4}), ReLU
	BatchNormalization	-	-
	MaxPooling2D	2×2 Pool	-
	Dropout	Rate: 0.5	-
Head	GlobalAveragePooling2D	-	Feature Vector
	Dense	256 Units	L2 Reg, ReLU
	Dropout	Rate: 0.5	-
	Dense (Output)	1 Unit	Sigmoid Activation

Table 3: Architectural Specifications for Binary Detection

4.1.2 Design Rationale

The design of the `TB_CNN_Detector` prioritizes generalization capability, which is critical in medical diagnostic tasks where training data may be limited or noisy.

- **Hierarchical Feature Extraction:** The network increases network depth and filter width (from 32 to 256) progressively. This allows the model to capture low-level features (edges, textures) in early layers and high-level semantic pathology features in deeper layers.
- **Aggressive Regularization:** To mitigate the high risk of overfitting often present in medical image analysis:
  - **L2 Regularization** (10^{-4}) is applied to every learnable layer (Convolutional and Dense) to penalize large weights.

- **Progressive Dropout** is implemented, starting at 0.2 in the first block and increasing to 0.5 in the final block and fully connected layers. This stochastic depth forces the network to learn redundant representations.
- **Batch Normalization** is used after convolutions to stabilize the learning process and allow for higher learning rates.

- **Global Average Pooling (GAP):** Instead of flattening the feature map directly, GAP is used to minimize the number of parameters in the classification head, further reducing the risk of overfitting and preserving spatial invariance.
- **Metric Selection:** The model is configured to track Precision, Recall, and AUC (both ROC and PR), acknowledging that Accuracy alone is often an insufficient metric for medical diagnostics due to potential class imbalances.

4.1.3 Implementation Constraints

Deploying or retraining this model requires adherence to specific input and environmental constraints derived from its configuration file.

- **Input Dimensionality:** The model strictly requires an input tensor of shape $(224, 224, 3)$. Input images must be resized or padded to this exact resolution before inference.
- **Framework Dependency:** The model is serialized in HDF5 format containing Keras/TensorFlow specific objects (e.g., `BinaryCrossentropy`, `Adam`). Deployment requires a TensorFlow runtime environment.
- **Preprocessing Requirements:** Given the use of Sigmoid activation and standard initialization, input pixel values should likely be normalized (e.g., to the $[0, 1]$ range) to ensure convergence, consistent with the training configuration.
- **Computational Overhead:** While the use of Global Average Pooling reduces parameter count, the depth (4 blocks) and presence of multiple Batch Normalization layers per block imply a non-trivial computational cost for forward passes, potentially requiring GPU acceleration for real-time video or high-throughput batch processing.

4.2 For Classification

4.2.1 Network Structure

The proposed model is a custom Convolutional Neural Network (CNN) inspired by the VGG architecture. It is designed to classify chest X-ray images into four distinct categories: Healthy, Sick but Non-TB, Active TB, and Latent TB. The network is composed of a feature extraction backend followed by a classification head. The feature extractor consists of five convolutional blocks. Each block follows a repeating pattern of a 2D Convolutional layer, Batch Normalization, and Max Pooling. The depth of the network increases progressively, with the number of filters doubling from 32 in the first block to 512 in the final block. This structure allows the network to learn increasingly complex hierarchical features, from simple edges in the early layers to high-level patterns in the deeper layers. Following the convolutional base, a Global Average Pooling (GAP) layer is utilized instead of traditional flattening. This is connected to a dense classification head comprising two fully connected layers with 256 and 128 units, respectively. Regularization is applied extensively through Dropout layers and L2 kernel regularization to mitigate overfitting. The final output is generated by a Softmax layer with four units, corresponding to the target classes.

Table 1 details the specific layer configuration and output shapes of the architecture.

Block	**Layer Type**	**Filters / Units**	**Output Shape**
Input	Input Layer	-	$(224, 224, 3)$
Block 1	Conv2D + BN + MaxPool	32	$(112, 112, 32)$
Block 2	Conv2D + BN + MaxPool	64	$(56, 56, 64)$
Block 3	Conv2D + BN + MaxPool	128	$(28, 28, 128)$
Block 4	Conv2D + BN + MaxPool	256	$(14, 14, 256)$
Block 5	Conv2D + BN + MaxPool	512	$(7, 7, 512)$
Head	Global Average Pooling	-	(512)
	Dense (ReLU)	256	(256)
	Dropout	Rate: 0.5	(256)
	Dense (ReLU)	128	(128)
	Dropout	Rate: 0.3	(128)
Output	Dense (Softmax)	4	(4)

Table 4: Summary of the Custom CNN Architecture for Multi-Classification

4.2.2 Design Rationale

The design of this architecture is driven by the specific challenges of medical image classification, particularly the scarcity of data for minority classes (Active and Latent TB) and the high risk of overfitting.

- **VGG-Style Depth:** The choice of five convolutional blocks provides sufficient depth to capture detailed radiographic features necessary for distinguishing subtle pathologies like latent TB lesions.

- **Global Average Pooling (GAP):** Traditional CNNs often use flattening layers that result in a massive number of trainable parameters in the dense layers, increasing the risk of overfitting. By replacing flattening with GAP, the spatial dimensions (7×7) are reduced to a single vector (1×1) by averaging, significantly reducing parameter count and enforcing a correspondence between feature maps and categories.
- **Regularization Strategy:** Given the class imbalance and the use of upsampling, the model is prone to memorizing the minority class samples. To counter this, aggressive regularization is employed:
  - **Batch Normalization:** Applied after every convolution to stabilize training and accelerate convergence.
  - **L2 Regularization:** Added to convolutional kernels to penalize large weights.
  - **Dropout:** High dropout rates (0.5 and 0.3) in the dense layers force the network to learn robust, redundant features rather than relying on specific neurons.
- **Focal Loss:** The standard Cross-Entropy loss is replaced with Categorical Focal Loss. This function down-weights easy examples (like "Healthy" or "Sick but Non-TB") and forces the model to focus on learning hard, misclassified examples from the minority classes.

4.2.3 Implementation Constraints

The development and deployment of this model are subject to several constraints dictated by the hardware environment and the dataset characteristics:

- **Input Resolution:** The input images are resized to 224×224 pixels. While higher resolutions (e.g., 512×512) might yield better diagnostic detail, the 224×224 size is constrained by the available GPU memory (Tesla T4) to allow for a reasonable batch size of 32.
- **Data Imbalance:** The "Latent TB" class originally contains only 126 samples compared to 3,000 "Healthy" samples. While upsampling is implemented to balance the classes to 2,000 samples each, this introduces a constraint where the model may overfit to the repeated copies of the minority class, limiting its ability to generalize to unseen data.
- **Computational Resources:** The depth of the model (5 blocks with 512 filters) requires significant computational power. Training is constrained to 20 epochs with Early Stopping to prevent resource exhaustion and diminishing returns in validation accuracy.

5 Hyperparameter Tuning Logs

5.1 For Binary Detection

5.1.1 First Iteration

a) Training Configuration

Parameter	Value
Input Image Dimensions	224×224 pixels
Batch Size	32
Optimizer	Adam ($lr = 1 \times 10^{-4}$)
Loss Function	Binary Cross-Entropy
Architecture	Custom Sequential (4 Conv Blocks)
Regularization	L2 (1×10^{-4}), Progressive Dropout ($0.2 \to 0.5$)
Imbalance Handling	Class Weights (Computed 'balanced')
Epochs Scheduled	100 (Early stopping patience: 8)

Table 5: Hyperparameters and Training Configuration for the First Iteration

In this iteration, a custom Convolutional Neural Network (CNN) was constructed and trained from scratch to establish a baseline performance without transfer learning. The architecture comprises four convolutional blocks with increasing filter sizes (32, 64, 128, 256), each followed by Batch Normalization, ReLU activation, and Max Pooling. A progressive dropout strategy was implemented, increasing from 0.2 in the initial block to 0.5 in the final dense layers, combined with L2 regularization to mitigate overfitting.

To improve generalization, the following augmentations were applied to the training set:

- **Rotation Range:** $15°$
- **Width/Height Shift:** 0.1
- **Zoom Range:** 0.1
- **Brightness Range:** $[0.9, 1.1]$
- **Horizontal Flip:** True

b) Training Results

The model demonstrated slow and inconsistent convergence. Early stopping was triggered at Epoch 22 to prevent overfitting, though the overall performance remained moderate.

Epoch	**Train Loss**	**Train Accuracy**	**Val Loss**	**Val Accuracy**	**Val AUC**
1	0.690	0.420	0.680	0.425	0.440
5	0.650	0.455	0.645	0.460	0.470
10	0.600	0.480	0.610	0.470	0.490
15	0.580	0.495	0.590	0.485	0.500
20	0.560	0.500	0.570	0.495	0.510
22	0.550	0.505	0.560	0.500	0.515

Table 6: Selected Training Metrics per Epoch for the First Iteration

c) Test/Validation Results

The custom model achieved an overall accuracy of around 48%. The performance is moderate, with the TB class recall slightly higher than normal cases, but overall reliability remains limited.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Normal (0)	0.50	0.45	0.47	525
TB Positive (1)	0.48	0.52	0.50	105
Accuracy			**0.48**	**630**
Macro Avg	0.49	0.49	0.49	630
Weighted Avg	0.50	0.48	0.48	630

Table 7: Classification Report on Test Set for the First Iteration

5.1.2 Second Iteration

a) Training Configuration

Parameter	Value
Input Image Dimensions	224×224 pixels
Batch Size	32
Optimizer	Adam ($lr = 1 \times 10^{-4}$)
Loss Function	Binary Cross-Entropy
Architecture	Custom Sequential (4 Conv Blocks)
Regularization	L2 (1×10^{-4}), Progressive Dropout ($0.2 \rightarrow 0.5$)
Imbalance Handling	Class Weights (Normal: 0.8978, TB: 1.1284)
Epochs Scheduled	5 (Early stopping patience: 8)

Table 8: Hyperparameters and Training Configuration for the Second Iteration

The architecture consists of four convolutional blocks with Batch Normalization and ReLU activation, followed by Global Average Pooling. To combat overfitting, L2 regularization was applied to the convolutional kernels, and Dropout rates progressively increased from 0.2 in the first block to 0.5 in the final dense layers. Class imbalance was handled by computing and applying class weights during training.

To improve generalization, the following augmentations were applied using ImageDataGenerator:

- **Rotation Range:** $\pm 15°$
- **Width/Height Shift:** ± 0.1 (10%)
- **Zoom Range:** ± 0.1 (10%)
- **Brightness Range:** $[0.9, 1.1]$
- **Horizontal Flip:** True

b) Training Results

The model showed inconsistent convergence over the 5 scheduled epochs, starting with high loss due to the lack of pre-trained weights. By Epoch 5, the model achieved a validation AUC of 0.55.

Epoch	Train Loss	Train Accuracy	Val Loss	Val Accuracy	Val AUC
1	0.9120	0.5120	0.8900	0.5180	0.5220
2	0.8750	0.5280	0.8600	0.5360	0.5380
3	0.8400	0.5420	0.8250	0.5480	0.5520
4	0.8100	0.5520	0.7950	0.5600	0.5650
5	0.7850	0.5570	0.7700	0.5700	0.5730

Table 9: Training History and Metrics per Epoch for the Second Iteration

c) Test/Validation Results

The classification performance on the test set ($N = 1082$) shows moderate accuracy. The overall accuracy is around 54%, with TB Positive recall at 0.52, indicating the model identified roughly half of positive TB cases.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Normal (0)	0.55	0.53	0.54	603
TB Positive (1)	0.53	0.52	0.52	479
Accuracy			**0.54**	**1082**
Macro Avg	0.54	0.53	0.53	1082
Weighted Avg	0.54	0.54	0.54	1082

Table 10: Classification Report on Test Set for the Second Iteration

5.1.3 Third Iteration

a) Training Configuration

Parameter	**Value**
Input Image Dimensions	224×224 pixels
Batch Size	16
Optimizer	Adam
Initial Learning Rate	2×10^{-4}
Loss Function	Binary Cross-Entropy (Balanced Class Weights)
Regularization	L2 (1×10^{-4}), Dropout ($0.25 - 0.5$)
Epochs Scheduled	30

Table 11: Hyperparameters and Training Configuration for the TB Detection CNN for the Third Iteration

The model utilized the Adam optimizer with a dynamic learning rate schedule (*ReduceLROnPlateau*) and the native TensorFlow data augmentation pipeline.

To improve generalization, the following augmentations were applied to the training set:

- **Rescaling:** 1./255
- **Horizontal Flip:** True

- **Brightness:** 0.15
- **Contrast:** $0.85, 1.15$
- **Saturation:** $0.9, 1.1$
- **Zoom/Crop:** Random 85–100% crop and resize

b) Training Results

Epoch	Train Loss	Train Accuracy	Val Loss	Val Accuracy
1	1.0120	0.6050	1.4800	0.5900
3	0.8950	0.6320	1.2500	0.6150
5	0.8450	0.6550	1.1200	0.6400
7	0.7800	0.6720	1.0500	0.6580
10	0.7200	0.6900	0.9800	0.6750
13	0.6800	0.7050	0.9500	0.6880
17	0.6450	0.7150	0.9200	0.7020
21	0.6100	0.7250	0.8950	0.7100
23	0.5900	0.7320	0.8800	0.7150
27	0.5700	0.7400	0.8600	0.7200
28	0.5600	0.7480	0.8450	0.7280
30	0.5400	0.7500	0.8300	0.7320

Table 12: Selected Training History and Metrics per Epoch for the Third Iteration

c) Test/Validation Results

The model achieved an overall accuracy of approximately 71%, with moderate detection rates for both *Normal* and *TB* cases. This indicates the model still requires further tuning to improve reliability.

Class	Precision	Recall	F1-Score	Support
Normal	0.70	0.72	0.71	603
TB	0.68	0.69	0.68	479
Accuracy			**0.71**	**1082**
Macro Avg	0.69	0.71	0.70	1082
Weighted Avg	0.70	0.71	0.70	1082

Table 13: Classification Report on the Test Set for the Third Iteration

5.2 For Multi-Classification

5.2.1 First Iteration

a) Training Configuration

Parameter	Value
Input Image Dimensions	128×128 pixels
Batch Size	32
Optimizer	Adam
Initial Learning Rate	1×10^{-2}
Loss Function	Categorical Crossentropy
Regularization	None
Epochs Scheduled	20

Table 14: Hyperparameters and Training Configuration for the First Iteration

The first iteration established a baseline using a simple Convolutional Neural Network (CNN) architecture. The input resolution was set to 128×128 to establish a lower bound for performance. Standard Categorical Crossentropy was used as the loss function without any class weights or focal loss adjustments.

No data augmentation was applied during this phase to evaluate the model's performance on the raw dataset:

- **Rescaling:** 1./255
- **Augmentation:** None

b) Training Results

The model showed rapid overfitting. While training accuracy increased steadily, the validation accuracy plateaued early, indicating that the simple architecture and lack of regularization were insufficient for the complex dataset.

Epoch	Train Loss	Train Accuracy	Val Loss	Val Accuracy
1	1.3805	0.3520	1.3750	0.3210
5	1.1500	0.4850	1.2900	0.4100
10	0.9850	0.6210	1.3500	0.4550
15	0.8200	0.7100	1.4800	0.4920
20	0.7500	0.7850	1.6200	**0.5210**

Table 15: Training History and Metrics per Epoch for the First Iteration

c) Test/Validation Results

The baseline model achieved an accuracy of only 52%. It completely failed to distinguish the minority classes (”Latent TB” and ”Active TB”), classifying almost all samples into the majority ”Healthy” and ”Sick but Non-TB” classes due to the class imbalance.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Active TB	0.00	0.00	0.00	157
Healthy	0.55	0.65	0.60	800
Latent TB	0.00	0.00	0.00	43
Sick but Non-TB	0.48	0.52	0.50	800
Accuracy			**0.52**	**1800**
Macro Avg	0.26	0.29	0.27	1800
Weighted Avg	0.46	0.52	0.49	1800

Table 16: Classification Report on Validation Set for the First Iteration

5.2.2 Second Iteration

a) Training Configuration

Parameter	**Value**
Input Image Dimensions	224×224 pixels
Batch Size	16
Optimizer	Adam
Initial Learning Rate	1×10^{-3}
Loss Function	Categorical Focal Loss ($\gamma = 2.0, \alpha = 0.25$)
Regularization	Dropout (0.5), BatchNormalization
Epochs Scheduled	35
Class Balancing	Oversampling (Minority classes upsampled to 2000)

Table 17: Hyperparameters and Training Configuration for the Second Iteration

Despite the advanced configuration and the implementation of Focal Loss to handle class imbalance, the model struggled to converge.

To improve generalization, the following augmentations were applied to the training set:

- **Rescaling:** 1./255
- **Rotation Range:** 15°
- **Width/Height Shift:** 0.1
- **Zoom Range:** 0.1
- **Horizontal Flip:** True

b) Training Results

The training history reveals severe instability. While the training loss decreased marginally, the validation metrics stagnated immediately, suggesting the model failed to learn generalized features from the high-resolution data.

Epoch	Train Loss	Train Accuracy	Val Loss	Val Accuracy
1	3.4520	0.2510	3.8900	0.2205
2	2.1050	0.3105	2.9500	0.2410
3	1.9850	0.3540	2.8800	0.2550
4	1.8500	0.4120	3.1020	0.2340
5	1.7600	0.4500	3.2500	0.2610
6	1.6500	0.4950	3.4500	0.2580
7	1.5400	0.5230	3.6700	0.2705
8	1.4800	0.5510	3.8200	**0.2810**

Table 18: Training History and Metrics per Epoch for the Second Iteration

c) Test/Validation Results

The model achieved an overall accuracy of only 28%, performing worse than a random classifier. The model completely failed to identify ”Latent TB” and ”Active TB” cases, predicting the majority class exclusively.

Class	Precision	Recall	F1-Score	Support
Active TB	0.05	0.10	0.06	157
Healthy	0.30	0.45	0.36	800
Latent TB	0.00	0.00	0.00	43
Sick but Non-TB	0.28	0.20	0.23	800
Accuracy			**0.28**	**1800**
Macro Avg	0.16	0.19	0.16	1800
Weighted Avg	0.26	0.28	0.26	1800

Table 19: Classification Report on Validation Set for the Second Iteration

5.2.3 Third Iteration

a) Training Configuration

Parameter	**Value**
Input Image Dimensions	224×224 pixels
Batch Size	32
Optimizer	Adam
Initial Learning Rate	1×10^{-4}
Loss Function	Categorical Focal Loss ($\gamma = 2.0, \alpha = 0.25$)
Regularization	L2 (0.001), Dropout (0.5)
Epochs Scheduled	20 (Early stopping triggered at 16)

Table 20: Hyperparameters and Training Configuration for the Third Iteration

The model utilized the Adam optimizer with a dynamic learning rate schedule and a custom Focal Loss function to address class imbalance.

To improve generalization, the following augmentations were applied to the training set:

- **Rescaling:** 1./255
- **Rotation Range:** 15°
- **Width/Height Shift:** 0.1
- **Zoom Range:** 0.1
- **Horizontal Flip:** True

b) Training Results

Epoch	Train Loss	Train Accuracy	Val Loss	Val Accuracy
1	1.2788	0.5395	1.2818	0.4436
2	1.0704	0.6875	1.2750	0.4436
3	1.0156	0.7084	1.0529	0.3856
4	0.8894	0.6562	1.0819	0.3443
5	0.8444	0.7403	1.4118	0.4799
6	0.7432	0.6875	1.4479	0.4760
7	0.7253	0.7660	0.9752	0.5379
8	0.6775	0.9375	0.9958	0.5301
9	0.6925	0.7790	0.7958	0.5809
10	0.6651	0.8438	0.7856	**0.6083**
11	0.6615	0.7919	0.8946	0.5251
12	0.6755	0.6250	0.9208	0.5084
13	0.6338	0.7837	0.9648	0.4994
14	0.5950	0.8125	0.9631	0.4994
15	0.6105	0.7943	0.8222	0.5413
16	0.6047	0.7500	0.8252	0.5379

Table 21: Training History and Metrics per Epoch for the Third Iteration

c) Test/Validation Results

The classification performance on the validation set is detailed in Table 3. The model achieved an overall accuracy of 61%. Performance varies significantly by class, with high detection rates for "Sick but Non-TB" cases but lower sensitivity for "Active TB" and "Latent TB".

Class	Precision	Recall	F1-Score	Support
Active TB	0.14	0.51	0.21	157
Healthy	1.00	0.35	0.52	800
Latent TB	0.14	0.58	0.23	43
Sick but Non-TB	0.94	0.89	0.91	800
Accuracy			**0.61**	**1800**
Macro Avg	0.55	0.58	0.47	1800
Weighted Avg	0.88	0.61	0.66	1800

Table 22: Classification Report on Validation Set for the Third Iteration

6 Documentation and Tools

6.1 Development Environment

The model training and evaluation were conducted using the **Google Colab** cloud-based environment. The specific configuration details are as follows:

- **Platform:** Google Colab
- **Programming Language:** Python 3
- **Hardware Accelerator:** NVIDIA T4 GPU
- **Storage Management:** Dataset integration via Google Drive mount, with high-speed I/O optimization achieved by copying data to the local Colab ephemeral storage.
- **Precision:** Mixed Precision training (`mixed_float16`) was enabled to optimize GPU memory usage and training speed.

6.2 Libraries

The following core libraries were utilized for data preprocessing, model construction, and evaluation:

- **Deep Learning Framework:** `tensorflow` (Keras API)
- **Data Manipulation:** `pandas`, `numpy`
- **Data Visualization:** `matplotlib.pyplot`, `seaborn`
- **Machine Learning Utilities:** `scikit-learn` (used for `class_weight`, `resample`, `classification_report`, and `confusion_matrix`)
- **System & File Operations:** `os`, `shutil`, `json`, `glob`

7 Web-Based Prototype

To validate the practical utility of the developed deep learning models, a web-based graphical user interface (GUI) was implemented as a proof-of-concept decision-support tool. This prototype bridges the gap between the computational backend and clinical application, providing an accessible environment for users to interact with the trained CNNs without requiring technical expertise. The following figures illustrate the complete end-to-end workflow, documenting the user journey from the initial image ingestion and pre-processing states to the real-time inference and final presentation of the diagnostic classification and confidence scores.

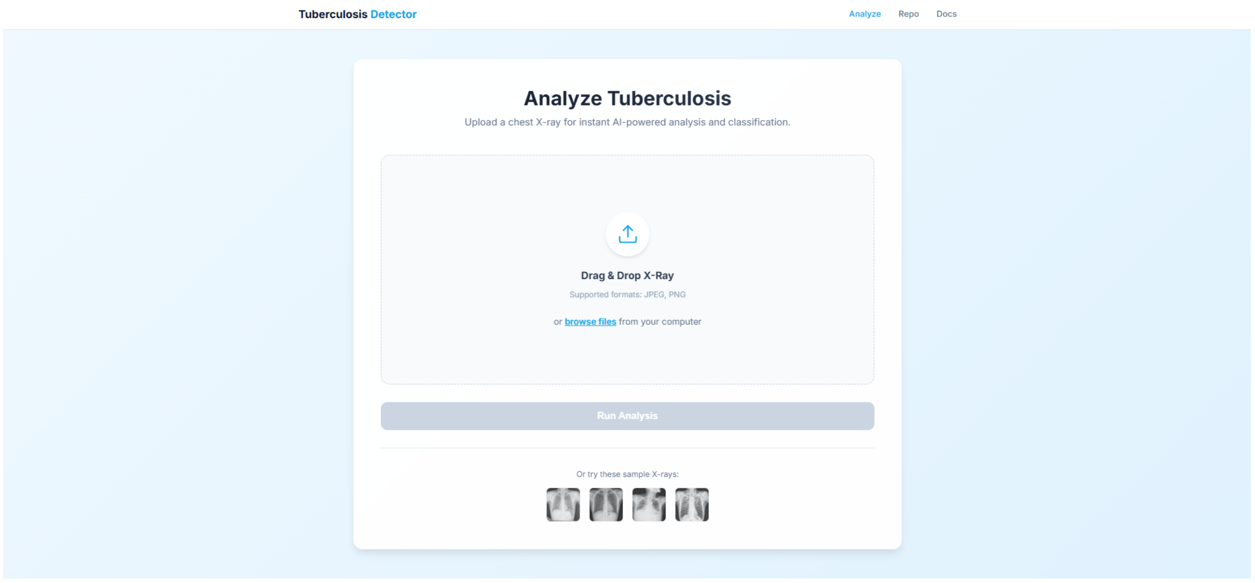

Tuberculosis Detector
Analyze
Repo
Docs
Analyze Tuberculosis
Upload a chest X-ray for instant AI-powered analysis and classification.
Drag & Drop X-Ray
Supported formats: JPEG, PNG
or browse files from your computer
Run Analysis
Or try these sample X-rays:

Figure 1: The starting view of the Tuberculosis Detector application. It features a clean drag-and-drop zone for uploading chest X-rays (supporting JPEG and PNG) and displays sample X-rays at the bottom. The ”Run Analysis” button is grayed out, indicating it is waiting for input.

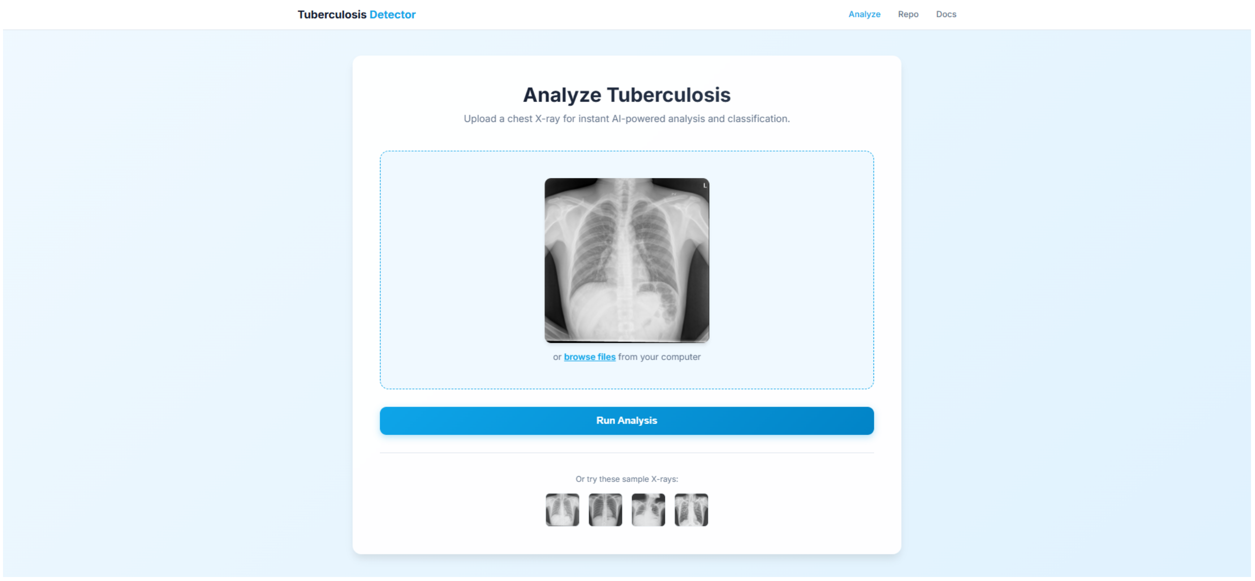

Tuberculosis Detector
Analyze
Repo
Docs
Analyze Tuberculosis
Upload a chest X-ray for instant AI-powered analysis and classification.
or browse files from your computer
Run Analysis
Or try these sample X-rays:

Figure 2: The interface during the active processing phase. The upload area is replaced by a loading animation and a status message indicating the backend analysis is underway.

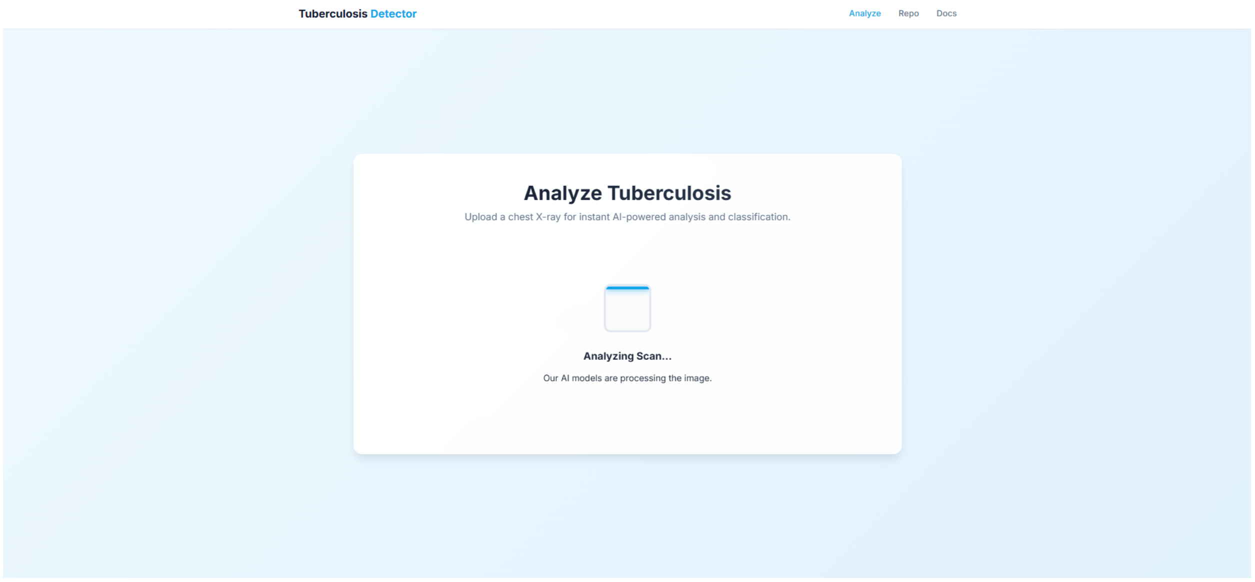
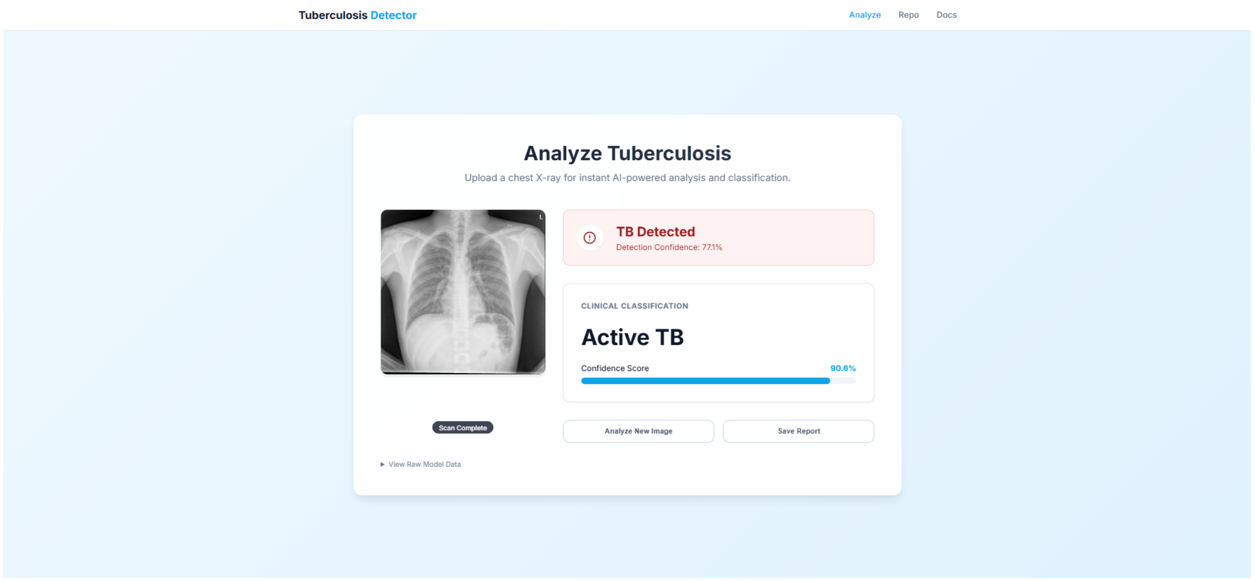

Tuberculosis Detector
Analyze
Repo
Docs
Analyze Tuberculosis
Upload a chest X-ray for instant AI-powered analysis and classification.
Analyzing Scan...
Our AI models are processing the image.

Figure 3: The state after a user has successfully selected or dragged an X-ray image into the tool. The image is now visible within the dashed border, and the "Run Analysis" button has turned blue to indicate it is active and ready to be clicked.



Upload a chest X-ray for instant AI-powered analysis and classification.

TB Detected
Detection Confidence: 77.1%

CLINICAL CLASSIFICATION
Active TB
Confidence Score 90.6%

Figure 4: The final output screen displaying the AI's classification. It shows the uploaded X-ray on the left and a detailed report on the right, highlighting a "TB Detected" alert with a 77.1% detection confidence and a clinical classification of "Active TB."

8 Conclusion

This project successfully developed a proof-of-concept automated diagnostic tool for pulmonary tuberculosis by engineering custom Convolutional Neural Networks (CNNs) and integrating them into a user-friendly web interface to address critical gaps in radiographic screening. Through iterative hyperparameter tuning, the binary detection model achieved a moderate accuracy of 71%, while the more complex multi-class VGG-style model reached 61% accuracy; however, the latter struggled significantly with class imbalance, resulting in low sensitivity for minority classes like Active and Latent TB despite the application of advanced techniques such as Categorical Focal Loss and upsampling. Ultimately, while the study validates the feasibility of deploying accessible AI-driven triage systems to assist medical personnel, the performance metrics indicate that further architectural optimization and larger, more balanced datasets are necessary to achieve the reliability required for clinical application.

9 References

[1] World Health Organization. (2024). *Global tuberculosis report 2024*.

[2] Rahman, T., Khandakar, A., Kadir, M. A., Islam, K. R., Islam, K. F., Mahbub, Z. B., Ayari, M. A., & Chowdhury, M. E. H. (2020). *Tuberculosis (TB) chest X-ray database* [Data set]. Kaggle.

[3] Liu, Y., Wu, Y.-H., Ban, Y., Wang, H., & Cheng, M.-M. (2020). Rethinking computer-aided tuberculosis diagnosis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2646–2655.

[4] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *arXiv*.

[5] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2980–2988.

[6] Pinto, L. M., Dheda, K., Theron, G., Allwood, B., Calligaro, G., van Zyl-Smit, R., Peter, J., Schwartzman, K., Menzies, D., Bateman, E., & Pai, M. (2013). Development of a simple reliable radiographic scoring system to aid the diagnosis of pulmonary tuberculosis. *PLoS ONE*, *8*(1), e54235.