Lung Disease Classification from Chest X-Ray Images using Deep Ensemble CNNs and a Python Streamlit Interface

Author: Eshan Puri (8025340013), M.E. – Artificial Intelligence

----

1. Introduction

Medical imaging is a cornerstone of modern medicine, with Chest X-rays (CXRs) serving as a critical and widely available modality for diagnosing respiratory diseases. The recent global health landscape has significantly increased the workload on radiologists, creating a pressing need for tools that can enhance diagnostic efficiency and consistency. Automated analysis powered by deep learning presents a powerful solution, offering the potential to assist clinicians by providing a fast, reliable, and quantitative "second opinion" that can aid in triage and diagnosis.

This project details the development of an end-to-end Python system designed for the multi-class classification of lung diseases from CXR images. The system is engineered to distinguish between five distinct conditions: **Bacterial Pneumonia**, **Corona Virus Disease**, **Normal**, **Tuberculosis**, and **Viral Pneumonia**. This is achieved through a robust technical stack built on a PyTorch backend, leveraging three proven Convolutional Neural Network (CNN) architectures—DenseNet121, EfficientNet-B0, and ResNet50—and culminating in a high-performance soft-voting ensemble model. To ensure practical usability, the system is deployed through a user-facing web interface built with Streamlit.

This project is not merely a modeling exercise but a comprehensive demonstration of Python's unique capacity to unify research, engineering, and user-facing deployment into a single, cohesive system. It showcases an integrated application of Python's scientific libraries (NumPy, Pandas, Matplotlib), advanced deep learning frameworks (PyTorch, timm), and modern web deployment tools (Streamlit). This highlights the full project lifecycle, from data preparation and modular code organization to interactive user interface deployment, proving Python's power in bridging the gap between complex backend models and intuitive frontend user experiences. This report will now explore the foundational concepts and related work that informed the project's design.

2. Background and Related Work

A thorough review of existing literature and foundational principles is essential to situate this project within the broader technical context. This section justifies the selection of specific methodologies—namely CNNs, transfer learning, and interactive AI systems—by discussing their evolution and established effectiveness in the field of medical imaging analysis.

2.1. Automated Analysis of Chest X-rays

The field of computer-aided diagnosis (CAD) for CXRs has evolved significantly over the past decades. Early systems relied on handcrafted features, such as texture descriptors and edge maps, which were fed into classical machine learning classifiers. While pioneering, these approaches were often brittle and required extensive, domain-specific feature engineering. The advent of deep CNNs, propelled by their success on large-scale datasets like ImageNet, marked a paradigm shift. CNNs learn discriminative features directly and hierarchically from image data, largely automating the feature extraction process and leading to substantial performance gains. A common and highly effective strategy in this domain is **transfer learning**, where models pre-trained on ImageNet are fine-tuned on medical imaging datasets, leveraging the general visual features learned from natural images.

2.2. Ensemble Learning for Medical Imaging

Ensemble methods are a powerful technique for improving the performance and reliability of machine learning models. The primary benefit of an ensemble is its ability to exploit model diversity to reduce predictive variance and enhance generalization. By combining the outputs of multiple, distinct models, an ensemble can mitigate the weaknesses of any single architecture and produce more stable and accurate predictions. This project trains three complementary architectures but constructs a **Top-2 soft-voting ensemble** for the final classifier. In this approach, the final prediction is determined by averaging the class probabilities produced by the two best-performing models (`DenseNet121` and `EfficientNet-B0`), creating a final classifier that is more robust than any individual model.

2.3. The Role of Interactive Interfaces in AI

The practical impact of an AI model is ultimately determined by its accessibility and usability for end-users. A highly accurate model is of little value if it cannot be easily integrated into a real-world workflow. Modern Python web frameworks like Streamlit have democratized the deployment process, enabling data scientists and AI engineers to convert complex models into interactive and user-friendly interfaces with minimal web development overhead. This project's Streamlit application is designed with this principle in mind, aiming to enhance user trust and model interpretability through features like clear confidence scores, per-class probability visualizations, and automatically generated, structured reports. These features transform the underlying model from a black box into a tangible and interpretable diagnostic aid. The following sections will now define the specific problem this project seeks to address.

3. Problem Statement

This section precisely defines the project's core challenge, its primary objectives, and the specific boundaries of the work undertaken. The goal is to articulate a clear and measurable plan for creating a robust and usable diagnostic aid for lung disease classification.

The primary objective of this project is to design and implement a Python-based, end-to-end system for the multi-class classification of lung diseases from Chest X-ray images, complete with an interactive and user-friendly interface.

This high-level goal is broken down into the following specific objectives:

1. **Develop a Robust Multi-Class Classifier.**
   - The system must accurately classify CXR images into one of five target lung conditions: `Bacterial Pneumonia`, `Corona Virus Disease (COVID-19)`, `Normal`, `Tuberculosis`, and `Viral Pneumonia`.
2. **Provide an Interactive, User-Friendly Interface.**
   - The interface must provide functionalities for users to upload single or multiple X-ray images, select from different classification models, view image quality diagnostics and predicted class probabilities, visualize model evaluation plots like confusion matrices, and download structured PDF reports summarizing the analysis.
3. **Demonstrate Key Python Engineering Concepts.**
   - The project must serve as a practical demonstration of core Python concepts relevant to AI engineering, including: modular code organization and configuration handling, data processing with NumPy and Pandas, visualization with Matplotlib and Seaborn, model training and inference with PyTorch and the `timm` library, and web deployment via Streamlit and GitHub.

The final system must be reproducible and efficient. The deployed application is designed to load pre-trained model weights (`.pth` files) to perform fast inference, separating the time-intensive training phase from the real-time prediction task.

4. Methodology

This section provides a detailed, end-to-end overview of the technical pipeline implemented in this project. It covers the entire workflow, from data acquisition and preprocessing to the specific strategies used for model training, evaluation, and ensembling.

4.1. Dataset

The project utilizes a curated dataset of chest X-ray images for lung disease classification. The dataset is organized into five balanced classes and was pre-split into training, validation, and test sets.

Dataset Split	Number of Images
Training	6,054

Validation	2,016
Test	2,027

The classes within the dataset are well-balanced, with each containing approximately 1,200 samples. This balance is crucial for preventing model bias towards more prevalent classes during training. The data is organized in a folder hierarchy compatible with PyTorch's `torchvision.datasets.ImageFolder`, simplifying the data loading process.

4.2. Pre-processing and Augmentation

A standardized pre-processing pipeline was applied to all images to ensure consistency and compatibility with the pre-trained models.

- **Resizing:** All images were resized to a uniform dimension of 224×224 pixels.
- **Normalization:** Images were normalized using the standard mean (`[0.485, 0.456, 0.406]`) and standard deviation (`[0.229, 0.224, 0.225]`) from the ImageNet dataset.

To improve model generalization, standard data augmentations like random horizontal flips were applied to the training set via `torchvision.transforms`. While more advanced techniques such as MixUp and CutMix were configured within the project's framework, they were ultimately disabled for the final training runs to ensure stability. For evaluation, **Test-Time Augmentation (TTA)** was employed to enhance robustness. Under TTA, predictions for each test image are generated twice—once for the original image and once for its horizontally flipped version—and the final probabilities are averaged.

4.3. Model Architectures and Hyperparameters

Three distinct and powerful CNN architectures were selected from the `timm` library, all of which were pre-trained on ImageNet to leverage transfer learning:

- `DenseNet121`: Features a dense connectivity pattern that promotes feature reuse and efficient gradient flow.
- `EfficientNet-B0`: A highly parameter-efficient model known for its excellent balance of accuracy and computational cost.
- `ResNet50`: A classic deep residual network that uses skip connections to enable the training of very deep models.

For each model, the final fully connected classification head was replaced with a new layer tailored for the five-class lung disease problem. The parameter counts for the adapted models are as follows:

Model	Total Parameters (M)
`densenet121`	6.96
`efficientnet_b0`	4.01
`resnet50`	23.52

4.4. Training Strategy

A two-phase training regime was implemented to effectively fine-tune the pre-trained models:

1. **Frozen Phase:** For the first 10 epochs, the pre-trained backbone layers were frozen, and only the newly added classifier head was trained. This was done with a learning rate of `1e-4` to adapt the model to the new task without disrupting the learned features.
2. **Fine-tuning Phase:** For the subsequent 10 epochs, the last two blocks of the backbone (`gradual_unfreeze_k = 2`) were unfrozen and trained alongside the classifier head with a lower learning rate of `5e-5`. This allows the pre-trained features to be subtly adjusted to the specifics of the CXR dataset.

The training process was governed by the following key components and hyperparameters:

- **Loss Function:** Cross-Entropy Loss, standard for multi-class classification.
- **Optimizer:** AdamW, selected via a flexible helper function (`build_optimizer`) that also supported Adam, SGD, and Ranger.
- **Scheduler:** A Cosine Annealing schedule with a linear warm-up phase to manage the learning rate, with `ReduceLROnPlateau` available as an alternative.
- **Regularization:** Early stopping with a patience of 5 epochs was used to prevent overfitting by halting training if validation performance did not improve.
- **Efficiency:** Automatic Mixed Precision (AMP) was enabled to accelerate training on compatible GPUs.
- **Reproducibility:** Random seeds for NumPy and PyTorch were fixed to ensure consistent results.

4.5. Ensemble Strategy

To further boost performance, a **Top-2 soft-average ensemble** was constructed. Based on their superior performance on the test set, the two best individual models—`EfficientNet-B0` and `DenseNet121`—were selected. The ensembling mechanism works by first obtaining the class probabilities from each model via a softmax function. These probability vectors are then averaged to produce a final, combined probability distribution, from which the final prediction is derived. This approach leverages the complementary strengths of the two architectures to create a more accurate and reliable classifier. The following section will detail the practical implementation of this methodology.

5. Implementation Details and Snapshots

This section focuses on the practical software engineering aspects of the project, detailing the technology stack, project structure, and the functionality of the user-facing Streamlit application. It provides a walkthrough of how the trained models are integrated into a cohesive and interactive tool.

5.1. Technology Stack and Project Structure

The project was built entirely within the Python ecosystem, leveraging a suite of powerful open-source libraries.

- **Programming Language:** Python 3.x
- **Core Libraries:** NumPy, Pandas, Matplotlib, Seaborn, scikit-learn
- **Deep Learning:** PyTorch, Torchvision, timm
- **Web UI:** Streamlit
- **PDF Generation:** fpdf
- **Deployment & Version Control:** GitHub, Streamlit Cloud

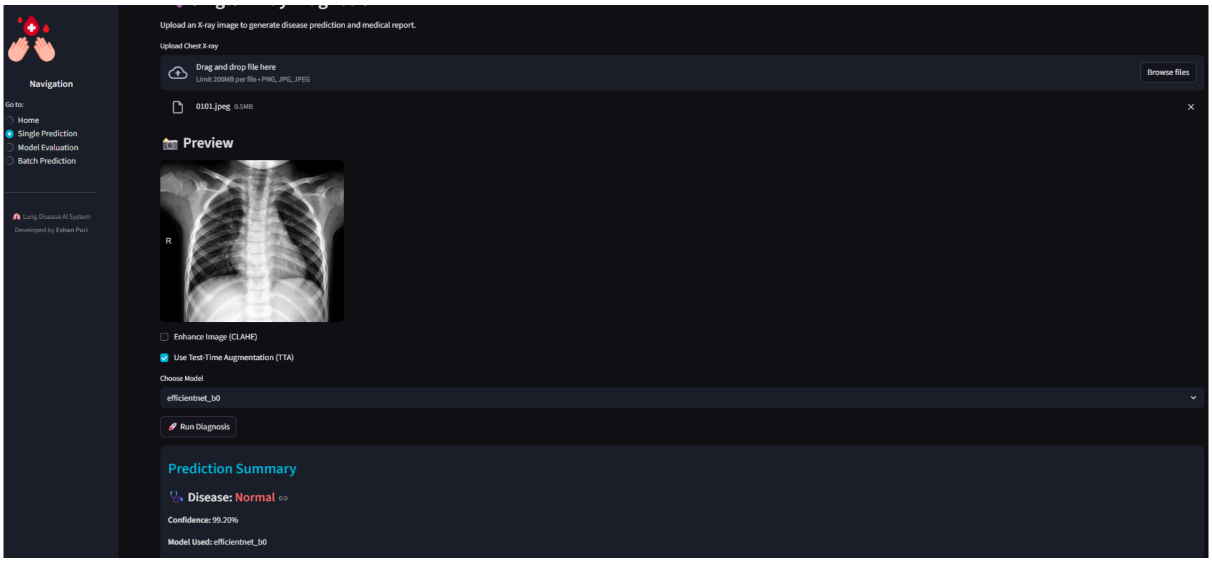
The project is organized into a clean and modular folder structure to separate concerns:

- `main.py`: The main script and entry point for the Streamlit application.
- `models/`: Contains the saved PyTorch model weights (`.pth` files) for fast inference.
- `results/`: Stores pre-generated evaluation plots, such as confusion matrices and ROC curves.
- `requirements.txt`: A file listing all Python dependencies required to run the project.

5.2. User Interface (UI) Walkthrough

The UI, built with Streamlit, is logically organized into three distinct pages, each serving a specific user workflow.

Single Prediction Page



Upload an X-ray image to generate disease prediction and medical report.

Upload Chest X-ray

0101.jpeg 0.5MB

Preview

☐ Enhance Image (CLAHE)

☑ Use Test-Time Augmentation (TTA)

Choose Model: efficientnet_b0

Prediction Summary

Disease: Normal

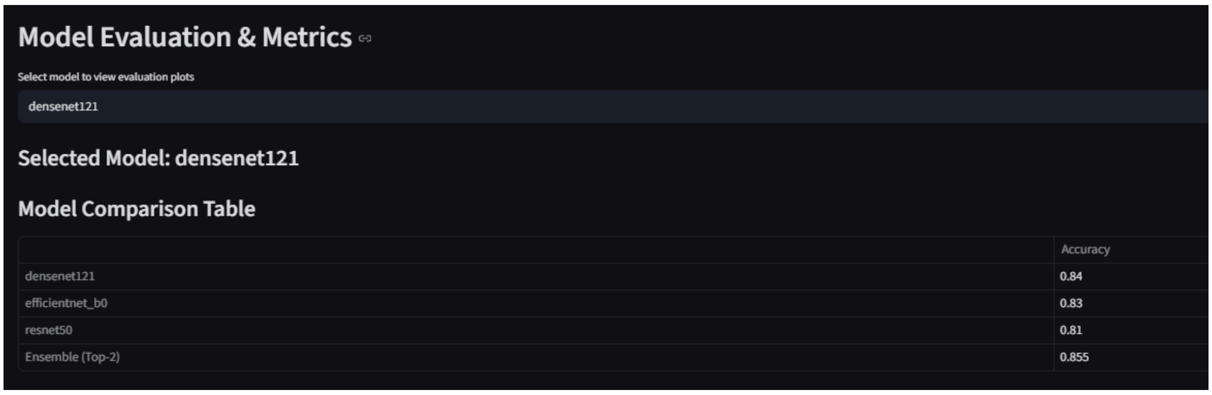
Confidence: 99.20%

Model Used: efficientnet_b0

This is the primary interface for analyzing an individual CXR image.

- **Workflow:** The user uploads a single image, selects a classification model from a dropdown menu (including the ensemble), and clicks the "Run Prediction" button. Optional image enhancement via CLAHE is available.
- **Outputs:** The application displays a comprehensive analysis, including:
  - Basic image quality metrics (blur, brightness).
  - The pre-processed image as seen by the model.
  - The final predicted disease with a corresponding confidence score, color-coded to indicate high or low confidence.
  - A horizontal bar chart visualizing the probability for each of the five classes.
  - A brief, pre-defined text description of the predicted disease (from `DISEASE_INFO`).
- **Reporting:** A key feature is the automated generation of a medical-style PDF report using `fpdf`. This report includes a patient ID, the uploaded image, the prediction details, probability scores, quality assessment, and a disclaimer. The user can download this report with a single click.

Model Evaluation Page



Model Evaluation & Metrics

Select model to view evaluation plots

densenet121

Selected Model: densenet121

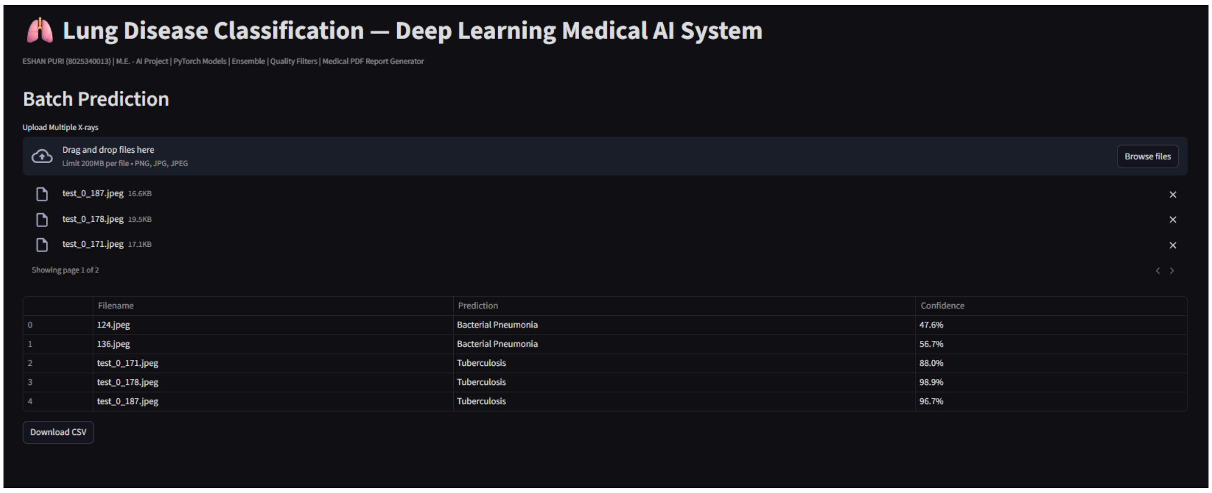
Model Comparison Table

	Accuracy
densenet121	0.84
efficientnet_b0	0.83
resnet50	0.81
Ensemble (Top-2)	0.855

This page provides transparency into the performance of the underlying models, connecting the offline evaluation with the deployed application.

- **Workflow:** The user selects a model from a dropdown to view its pre-calculated performance metrics.
- **Displayed Information:** The page presents:
    - A summary table comparing the test accuracies of all individual models and the ensemble.
    - The corresponding confusion matrix and ROC curve for the selected model. These are pre-generated images loaded directly from the `results/` folder, ensuring a seamless connection between offline evaluation and the deployed application.

Batch Prediction Page



Lung Disease Classification — Deep Learning Medical AI System

ESHAN PURI (8025340013) | M.E. - AI Project | PyTorch Models | Ensemble | Quality Filters | Medical PDF Report Generator

Batch Prediction

Upload Multiple X-rays

Drag and drop files here
Limit 200MB per file • PNG, JPG, JPEG

test_0_187.jpeg 16.6KB
test_0_178.jpeg 19.5KB
test_0_171.jpeg 17.1KB

Showing page 1 of 2

	Filename	Prediction	Confidence
0	124.jpeg	Bacterial Pneumonia	47.6%
1	136.jpeg	Bacterial Pneumonia	56.7%
2	test_0_171.jpeg	Tuberculosis	88.0%
3	test_0_178.jpeg	Tuberculosis	98.9%
4	test_0_187.jpeg	Tuberculosis	96.7%

This page is designed for processing multiple images simultaneously, simulating a more realistic clinical workflow.

- **Workflow:** The user uploads a batch of CXR images at once.
- **Output:** The application processes each image using the ensemble model and displays the results in a summary table with columns for `Filename`, `Prediction`, and `Confidence`. This table can be conveniently downloaded as a `.csv` file for offline analysis.

This implementation effectively translates the trained deep learning models into a practical and interactive tool, which is now ready for a detailed performance evaluation.

6. Results and Discussion

This section presents the empirical outcomes of the project, detailing the quantitative performance of each individual model and the final ensemble. This is followed by a qualitative discussion of the key findings, an analysis of common error patterns, and an acknowledgment of the project's limitations.

6.1. Quantitative Results

All models were evaluated on the held-out test set of 2,027 images. The final test accuracies and parameter counts are summarized below.

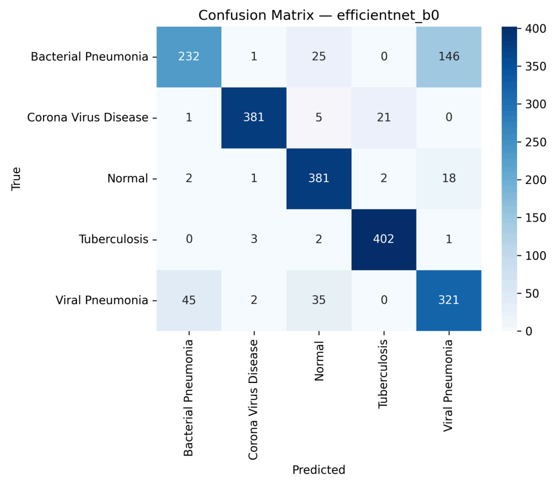
Model	Test Accuracy	Parameters (M)
Ensemble (Top-2)	85.40%	N/A
`efficientnet_b0`	84.71%	4.01
`densenet121`	83.03%	6.96
`resnet50`	76.81%	23.52

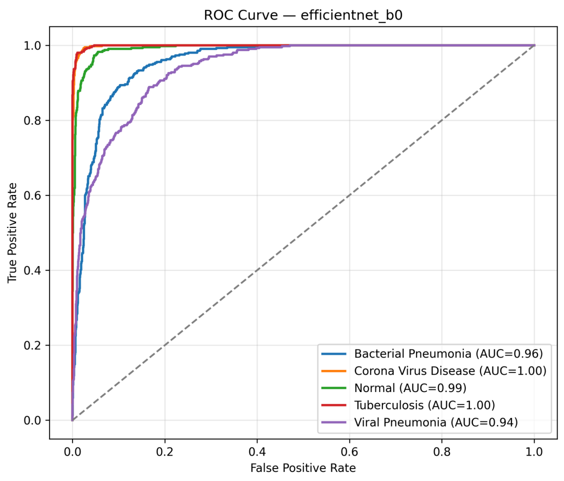
The **Top-2 ensemble** achieved the highest accuracy, demonstrating the effectiveness of combining diverse models. The detailed classification report for this best-performing ensemble model reveals strong performance across most classes.

Class	Precision	Recall	F1-Score
Bacterial Pneumonia	0.81	0.62	0.70

Corona Virus Disease	0.98	0.96	0.97
Normal	0.85	0.95	0.90
Tuberculosis	0.97	0.98	0.97
Viral Pneumonia	0.68	0.76	0.72

Furthermore, the Receiver Operating Characteristic (ROC) curves for the top-performing models consistently showed Area Under the Curve (AUC) values well above 0.9, indicating strong class separability and robust discriminative power.





6.2. Discussion

The quantitative results lead to several key insights:

- **Ensemble Effectiveness:** The Top-2 soft-voting ensemble outperformed the best individual model (`EfficientNet-B0`), confirming that averaging the probabilistic outputs of diverse, high-performing models leads to a more stable and accurate final classifier, effectively smoothing out individual model biases.
- **Performance vs. Efficiency Trade-off:** The results highlight a crucial principle in modern deep learning: more parameters do not guarantee better performance. The compact `EfficientNet-B0` (4.01M parameters) significantly outperformed the much larger `ResNet50` (23.52M parameters), demonstrating the superior architectural design and efficiency of the former, which is rooted in its novel compound scaling methodology.

- **Qualitative Observations:** Analysis of the confusion matrices revealed a clear pattern of misclassification. The most frequent errors occurred between **Bacterial Pneumonia** and **Viral Pneumonia**. This is an expected challenge, as these two conditions often present with similar radiographic opacities, making them difficult to distinguish even for human experts.
- **The Importance of the Python Ecosystem:** The project's success is defined not just by its final accuracy score but by its transformation into a usable tool. The seamless integration of a high-performance PyTorch backend with a user-friendly Streamlit frontend, complete with automated PDF reporting, showcases the unique power of the Python ecosystem for building end-to-end AI solutions that connect backend AI with frontend UX.

6.3. Limitations

A critical evaluation of the project reveals several limitations that provide avenues for future improvement:

- **Dataset Scope:** The model was trained on a specific, curated dataset. This dataset may not fully represent the vast real-world variability encountered in clinical practice, such as images from different scanner types, patient demographics, or the presence of co-morbidities.
- **Interpretability:** While the application provides confidence scores and probability distributions, it lacks advanced interpretability tools like Gradient-weighted Class Activation Mapping (Grad-CAM). Such tools could provide visual heatmaps to explain *why* a model made a particular prediction, enhancing clinical trust.
- **Image Quality Checks:** The implemented image quality assessments (blur, brightness) are basic. The system could be improved by incorporating more sophisticated checks to detect complex artifacts, cropping issues, or incorrect patient positioning.

These findings and limitations set a clear stage for the project's conclusion and future directions.

7. Conclusion and Future Work

This final section synthesizes the project's key achievements, summarizes the main findings, and proposes several concrete directions for future research and development that could build upon the current system.

7.1. Conclusion

This project successfully demonstrated the development of a complete, Python-based system for multi-class lung disease classification from chest X-ray images. The system spans the entire pipeline, from offline model training and evaluation to deployment as a fully interactive web application. The core technical achievement was the implementation of a **Top-2 soft-voting ensemble** combining `EfficientNet-B0` and `DenseNet121`, which achieved a final test accuracy of **85.4%**. This result validated the chosen methodology of leveraging transfer learning, diverse architectures, and ensemble techniques for a complex medical imaging task.

Ultimately, the system serves as an effective M.Tech-level demonstration of how to integrate cutting-edge AI (PyTorch), robust software engineering best practices, and modern UI/UX design (Streamlit) entirely within the Python ecosystem. It successfully transforms a set of trained models into a tangible and usable tool for academic and demonstration purposes.

7.2. Future Work

Several promising avenues exist for extending and improving this work:

- **Advanced Interpretability:** Re-integrate Grad-CAM to generate heatmaps showing which lung regions influenced a prediction and include them in the PDF report.
- **Advanced Augmentation and Training:** Experiment with more sophisticated techniques like MixUp, CutMix, and Stochastic Weight Averaging (SWA) to improve model robustness.
- **Expanded Model Architectures:** Evaluate newer architectures such as Vision Transformers (ViT) or ConvNeXt available in the `timm` library.
- **Domain Generalization:** Test and adapt the models using multi-center datasets to improve robustness to different imaging protocols.
- **Active Learning Loop:** Implement a feedback mechanism in the UI for expert corrections, which could be used to periodically fine-tune the models.

8. Github Link - ESPK-Eshan/lung_app: Lung Disease Classification – AI X-ray Analysis (Streamlit App)