ÇANKAYAUNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Project Report

CENG407

Innovative System Design and Development I

MULTI-LABEL CHEST X-RAY DISEASE CLASSIFICATION AND EXPLAINABILITY WITH DEEP LEARNING

YAREN AKDOĞAN 202111046

SEZİN KÖROĞLU 202211046

İBRAHİM TAŞPINAR 202111072

CEVDET ALP ÖZGÜN 202211406

İPEK BAŞ 202211013

SALİH BARKIN AKKAYA 202211004

Abstract

In today's medical field, chest X-ray images have an indispensable role in the diagnosis of thoracic diseases such as pneumonia, cardiomegaly and atelectasis. Quick and accurate diagnosis is crucial but the growing number of patients and the limited availability of radiologists can delay the results of X-rays. Deep learning techniques such as convolutional neural networks have shown great potential in the automation of disease detection thanks to their ability to detect multiple diseases and them allowing for models with accuracies comparable to human professionals. However, high accuracy is not the only relevant factor for real world clinical adoption. Doctors need to understand why a model makes a certain decision in order to trust and effectively use it. This is where explainable artificial intelligence methods such as Grad-CAM play an important role by visually highlighting the parts of an image that are relevant to the model's prediction through heatmaps. Through this review, we aim to understand the challenges we may face in development and methods we can employ in overcoming them. We also attempt to understand methods used previously in studies to improve the interpretability and accuracy of models for clinical us

Table of Contents

1. Introduction

Chest X-ray (CXR) imaging serves as a fundamental diagnostic tool in modern medicine for identifying various thoracic conditions such as pneumonia, cardiomegaly, and
atelectasis. However, the increasing volume of medical images and the limited availability of expert radiologists often lead to delays in diagnosis. To address this challenge, deep learning techniques, particularly Convolutional Neural Networks (CNNs), have been widely adopted for automating multi-label disease detection.

While high accuracy is essential, clinical integration requires that these models are also interpretable and trustworthy for healthcare professionals. This project proposes an end-to-end system that utilizes a DenseNet-121 backbone integrated with Convolutional Block Attention Modules (CBAM) to improve feature extraction from complex radiographic patterns. To ensure transparency, Grad-CAM (Gradient-weighted Class Activation Mapping) is employed to provide visual explanations of the model's decisions through heatmaps. The final system is delivered via a Flutter-based mobile application supported by a robust MongoDB-backed RESTful API, providing a clinically relevant decision-support tool.

2. Literature Review

2.1. Introduction

Chest X-ray images play an important role in diagnosing thoracic conditions, including pneumonia, cardiomegaly, and atelectasis [1,4]. The increasing volume of medical images, along with limited expert availability, has motivated the development of automated diagnosis systems based on deep learning [2]. Multi-label classification is particularly challenging in this context, as a single chest X-ray image may indicate multiple illnesses. Convolutional neural networks (CNNs) have become the dominant approach for image based medical diagnosis [2]. Transfer learning from large scale datasets such as ImageNet has further improved performance gains by providing pre-trained feature extractors. Alongside high classification accuracy, explainability has become one of the most important requirements for adoption in hospitals. Explainable AI (XAI) methods, such as Gradient-weighted Class Activation Mapping (Grad- CAM), allow doctors to view which parts of an image influence a model's prediction [4]. This literature review focuses on deep learning models for multi-label chest X-ray classification, dataset characteristics, and explainability methods, in an attempt to understand what is required to improve on existing ideas and possible difficulties we may encounter during development.

2.2. Deep Learning and Convolutional Neural Networks

2.2.1. Fundamentals of Convolutional Neural Networks

Convolutional Neural Networks are a type of deep learning model particularly suited for image classification due to their ability to extract features via convolutional layers. Convolutional layers apply filters to the input image to learn patterns such as edges, textures, or shapes. As data goes through layers, the model learns how to recognize objects such as organs or lesions in medical images. Pooling layers are used to reduce the size of feature maps to decrease the computational cost and to allow the model to only focus on the most significant information. Activation functions are used to add non-linearity to models to allow models to learn more complex patterns. By applying these layers several times, models learn to differentiate and classify images effectively.

2.2.2. Transfer Learning in CNN Based Medical Imaging

Due to the limited availability of medical image datasets, transfer learning is widely adopted in an effort to make the most of the lacking data. CNNs that have been pre-trained on large datasets to detect general objects are fine-tuned on chest X-ray images [2,7]. This method allows models to benefit from the detection capabilities of the pre-trained models, while still being customizable enough for optimization for medical images. This approach also decreases the required training time for models as less work needs to be done during feature extraction compared to a standard CNN model. Although there are many different pre-trained model alternatives with different capabilities, ResNet, AlexNet and DenseNet appear to be the most popular [1,2,4,5,6,7].

2.2.3. CNN Architectures Used in Chest X-Ray Analysis

2.2.3.1. AlexNet

AlexNet has been applied in chest X-ray classification for pneumonia and other thoracic diseases. For example, [7] used AlexNet for pneumonia detection, reporting around 81% accuracy. In another study, [5] combined AlexNet features with image processing methods to classify multiple thoracic conditions.

2.2.3.2. ResNet & DenseNet

More modern CNNs, such as ResNet and DenseNet, are widely used due to better feature extraction and deeper architectures. Comparative studies on tuberculosis and COVID-19 detection show that DenseNet121 and ResNet variants achieve high AUC scores and outperform AlexNet in multi-label chest X-ray classification [1,2,3,4,6].

2.2.3.3. Transformer (SwinCheX)

Although CNNs dominate, transformer-based architectures have begun to emerge. SwinCheX [6] applied a Swin Transformer backbone to the ChestX-ray14 dataset and achieved an average AUC of 0.810, demonstrating that transformer architectures can also perform competitively in medical imaging tasks.

2.3. Datasets and Multi-Label Classification

2.3.1. NIH ChestX-ray14

The NIH ChestX-ray14 dataset contains over 100,000 frontal chest X-ray images labelled for 14 different thoracic diseases, and it has become one of the most commonly used datasets in this research field [1]. Because manually labelling medical images is slow and requires expert radiologists, the dataset uses NLP methods to automatically extract labels from the original radiology reports. This approach makes it possible to build a very large dataset, but it also means that some labels may be inaccurate or wrong due to the uncertainty tied to automated text processing [1]. Even with this drawback, ChestX-ray14 is the most popular choice for studies thanks to its size and easy accessibility for researchers [1].

2.3.2. CheXpert

CheXpert is a large chest X-ray dataset with over 224,000 images from 65,000 patients [4]. Although the higher image quantity is a positive, the main feature that differentiates this dataset from the NIH ChestX-ray14 dataset is that CheXpert includes uncertainty labels, which capture cases where the radiology reports are ambiguous or unclear. Properly handling these uncertain labels is important for training models that achieve high accuracy.

Researchers often use strategies such as U-Ones, U-Zeros, or U-Ignore, depending on the
specific disease being predicted, to make the most of the dataset while managing uncertainty [4]. The reason this dataset has seen less use is most likely due to it being newer compared to the NIH ChestX-ray14 dataset and its harder access requirements. This dataset has started overtaking the NIH ChestX-ray14 dataset in popularity in recent years due to it providing better results for most studies.

2.3.1. Multi-Label Classification Challenges

Unlike single label classification, multi-label chest X-rays require predicting multiple diseases present at once [1,4]. In practice, most studies use binary cross-entropy based losses such as BCEWithLogitsLoss to allow the model to treat each disease as its own prediction. However, some diseases appear far less often than others, which makes training uneven. To deal with this, previous studies have often given more weight to rare classes or have adjusted the loss based on how many samples each class has [1,4]. Others simply increased the number of images for less common diseases through oversampling or augmentation [1,4]. These strategies help the model reach a higher final accuracy.

2.4. Explainable AI (XAI) Approaches

2.4.1. Importance of Explainability

In clinical applications, model interpretability is critical. Doctors need to understand why a model made a particular decision to trust its output [2,3,4]. XAI methods provide this capability by highlighting regions influencing model predictions.

2.4.2. Grad-CAM

Grad-CAM is an explainability method that identifies which regions of an image contribute most to a model’s decision by calculating the gradient of a target class score with respect to the CNN. This produces a heatmap that highlights the most relevant areas for the prediction. Because it is simple to implement and can be added to existing CNN models without retraining, Grad-CAM has become one of the most commonly used tools for interpreting deep learning based medical imaging systems.

However, Grad-CAM comes with several limitations. Its visualizations provide only coarse localization and are not completely pixel accurate, which can be problematic for detecting small or subtle abnormalities [4]. Although for most diseases this is a minor issue, some diseases with subtler or smaller indications suffer due to this. Additionally, Grad-CAM shows what influenced the model rather than why the model was influenced, meaning the highlighted regions do not always represent the correct disease [4]. This can create uncertainty when doctors rely on the visualization to validate the model's decisions.

2.5. Findings from Related Works

2.5.1. DenseNet121, CheXNet, and Other Chest X-Ray Models

DenseNet121, particularly with the CheXNet dataset, continues to be widely adopted as a strong baseline architecture for multi-label chest X-ray classification [2,3,4,6]. CheXNet's densely connected layer architecture allows features from earlier layers to be reused in later layers preventing vanishing gradients and other similar problems which are common in deep learning.

CheXNet [2] demonstrated performance exceeding average radiologists in detecting pneumonia, achieving an AUROC of 0.76 to 0.84 across different test subsets. Subsequent studies using the NIH ChestX-ray14 dataset applied this approach to 14 thoracic diseases, reporting average AUROC values ranging from
0.80 to 0.87 depending on preprocessing and training methods [3,4]. In an attempt to further validate the findings of these studies, we have also trained a reimplementation of the original CheXNet model in Python 3. Our findings aligned almost exactly with the original study with an AUROC score of around 0.76 despite hardware limitations. These results indicate DenseNet121 as a good fit to be a backbone for chest X-ray predictions.

We have also attempted to train the transformer model SwinCheX used in an effort to compare transformer and CNN architecture results. The original SwinCheX study achieved a high average AUROC score of 0.81 [6]. Our retrained model experienced difficulties due to lacking hardware and as such did not utilize the optimal training parameters specified by the study. Due to this issue our model only

reached an average AUC score of 0.54. From these training results we have deduced that the transformer architecture is not viable for lower end hardware.

2.5.2. Uncertainty and Class Imbalance

The currently widely available chest X-ray dataset all present a few common issues such as class imbalance due to rare or hard to find diseases and uncertain labels due to the utilization of NLPs in the automated labelling of the data. Many classes, such as pneumothorax or hernia, occur rarely compared to more common conditions like cardiomegaly or effusion. In the NIH ChestX-ray14 dataset this issue is quite noticeable as some findings only have around 300 labels whereas the more common ones can have up to 20000 instances [1].

In addition, automated label extraction from radiology reports introduces ambiguity and potential errors. For instance, ChestX-Ray8[1] found that weakly supervised labelling could mislabel up to 10% to 15% of cases due to uncertainty in the original text reports [1]. To address these issues, studies have employed methods such as weighted cross-entropy loss, focal loss, or class-balanced loss to give rarer conditions greater influence during training. U-Ignore and U-Ones strategies for handling uncertain labels in CheXpert have been proposed, demonstrating that appropriate treatment of uncertainty can improve AUROC by 2 to 3 percent on some diseases [4]. These findings indicate the importance of proper preprocessing and appropriate loss design to achieve consistent and accurate performance across all classes.

2.5.3. Explainability and Grad-CAM

While model accuracy is essential, clinical adoption requires interpretable predictions. Grad-CAM is frequently used to visualize which areas of an image influence a model’s output [4]. CheXpert applied Grad-CAM to highlight lesions indicating pneumonia, showing that heatmaps generally overlapped with regions on the

X-ray identified by radiologists [4].

However, Grad-CAM provides only rough localization, and pixel perfect interpretability has not yet been achieved in this area while using such tools. Later studies experimented with hybrid approaches where they combined Grad-CAM with attention mechanisms or integrated gradients to improve resolution [4,6]. These enhancements are particularly useful in finding subtle diseases, such as early stage edema or nodules, which are harder to find due to their small size, where precise localization can inform the clinical decisions of doctors.

2.5.4. Limitations of Current Models and Multimodal Models

Most existing models rely completely on frontal X-ray images, ignoring lateral views or complementary clinical data such as patient history, lab results, or prior imaging. Limiting the input to a single type of data decreases model performance. For example, several studies reported that including other patient data such as age, sex, and prior diagnoses improved average AUROC scores by 1 to 2 percent for certain rare diseases [4,6]. Multimodal models appear to be the next major step in a path toward models that are both more accurate and better aligned with the interests of doctors.

2.5.5. Summary

Overall, the literature shows DenseNet-121 based architectures as the most commonly used method for multi-label chest X-ray classification [2,3,4,6] and shows explainability methods like Grad-CAM as a critical requirement for trust from doctors [4]. Major challenges remain, including handling uncertain or wrong labels, class imbalance and integrating additional relevant patient information for multimodal predictions. Addressing these issues is important for the development of systems that are not only accurate but also interpretable, trustworthy and clinically relevant.

2.6. Conclusion

Deep learning has allowed multi-label chest X-ray classification to advance to a state where it’s viable in the real world. CNNs such as DenseNet-121 achieve high accuracy on the NIH ChestX-ray14 and the CheXpert datasets. Explainable AI techniques such as Grad-CAM have allowed doctors to visually understand model predictions, improving the interpretability of the models. Difficulties remain in handling rare or subtle diseases, uncertain or wrong labels and incorporating new data sources that may be relevant to patients. Future research can focus on higher resolution XAI, and optimizing architectures to improve clinical utility. This literature review demonstrates that while there has been significant development in the area, more attention towards explainability and dataset quality is required for real world adoption of this new technology as a form of trusted medical diagnosis.

3. DATASET DESCRIPTION & PREPROCESSING

3.1. Dataset Description

3.1.1. NIH ChestX-ray14

The NIH ChestX-ray14 dataset contains 112,120 frontal chest X-ray images labelled for 14 different thoracic diseases, and it has become one of the most commonly used datasets in this research field. The dataset is comprised of posterior- anterior (images taken with the patient facing detector) and anterior–posterior (images taken with the patient facing machine) X-ray images along with various details on each patient. All images in the dataset have a standard format of grayscale 1024x1024 images [8].

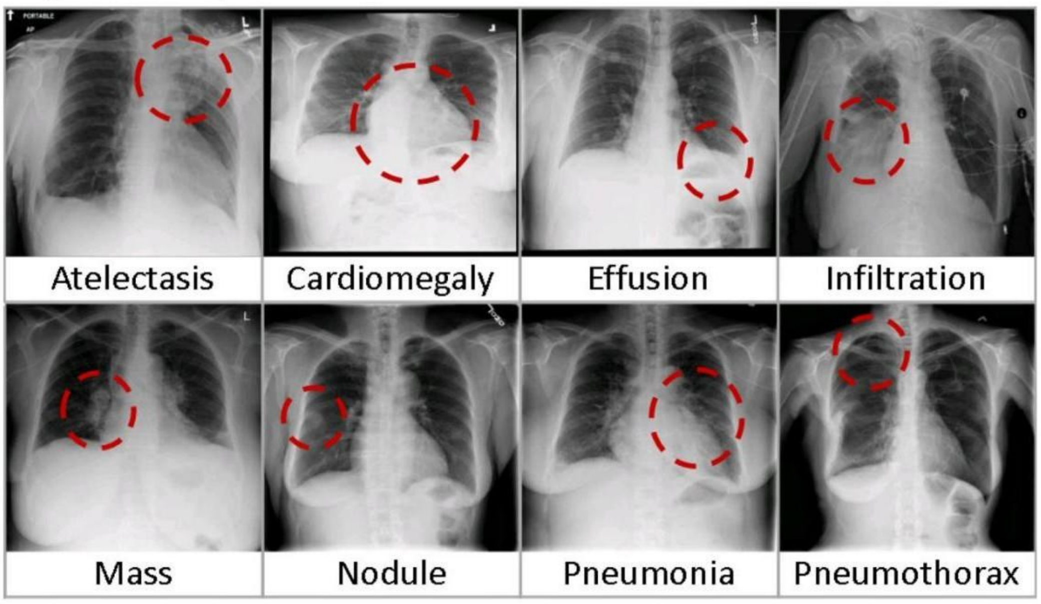


Figure 1. Eight visual examples of common thorax diseases

3.1.2. Label Information

The dataset has a multi-label structure for 14 thoracic diseases including Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema, Emphysema, Fibrosis, Effusion, Pneumonia, Pleural Thickening, Cardiomegaly, Nodule, Mass and Hernia. The labelling for the images has been done with the use of NLP techniques on real radiologist reports. Due to the uncertain nature of NLPs, the dataset contains some uncertain labels. However, the overall label accuracy of the dataset is >90%. The dataset also contains an issue due to the class imbalance of images. Although some classes have over 20,000 labels associated with them, there are also those with only around 2000 labels. To get around this issue we are planning to utilize weighted learning in our model [8].

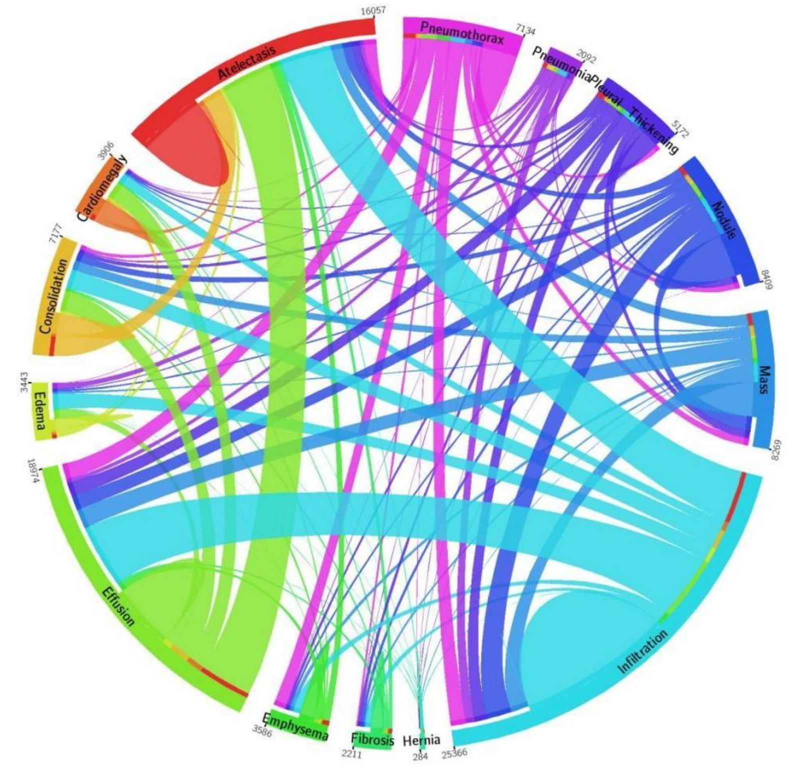


Figure 2. Chord diagram displaying the distributions of 14 disease categories with co-occurrence statistics

3.2. Frontal View–Based Filtering

First, we limit the dataset to chest X-ray images in posterior–anterior (PA) view. We read the original metadata file and select only the rows where the "View Position" column is "PA". Next, we scan the image directory tree and copy these PA images into a new folder called FinalDataset_PA. After this step, we save a filtered metadata file (metadata_pa_only.csv) that includes only the copied images. We also save a simple CSV file (image_names_pa_only.csv) that contains only the PA image names.

Then, we rearrange the original train and test splits for this PA-only subset. We read the PA metadata and create a set from the "Image Index" column. Using this set, we filter the original train_val_list.txt and test_list.txt files. For both of the file, we keep only the lines whose image name is in the PA set and write these lines into train_list_pa.txt and test_list_pa.txt. As a result of this way, every image in the train and test lists is a PA image that exists in the FinalDataset_PA directory.

3.3. Image Processing Scenarios

3.3.1. Scenario 1:

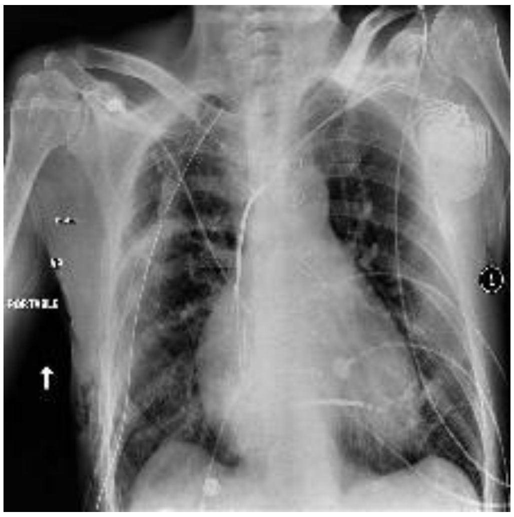


Figure 3. Processed version of the Chest X-Ray

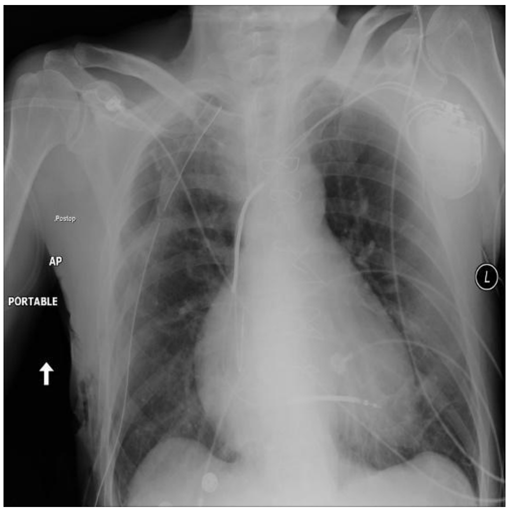


Figure 4. Unprocessed version of the Chest X-Ray

We plan to apply an intensity enhancement pipeline to chest X-ray images in the dataset. First, we will normalize each image to the range of [0,1]. Then, we will keep that range as a single grayscale channel. Later, we will apply gamma correction ($\gamma = 1.2$), CLAHE with a low clip limit, and an unsharp mask for sharpening [10][12]. All of these steps are planned to improve contrast and make the images more understandable for our system.

For each image, we plan to load it as a grayscale image with PIL and convert 16-bit images to 8- bit if necessary. We will then apply the enhancement pipeline and resize the result to 224×224 pixels using Albumentations (this will be needed for our deep learning model). After that, we will create a 3-channel RGB image by repeating the same grayscale channel three times, scale the values to [0, 255], convert to 8-bit, and save the result. This will give us images in a standard 224×224 RGB format as desired.

3.3.2. Scenario 2:

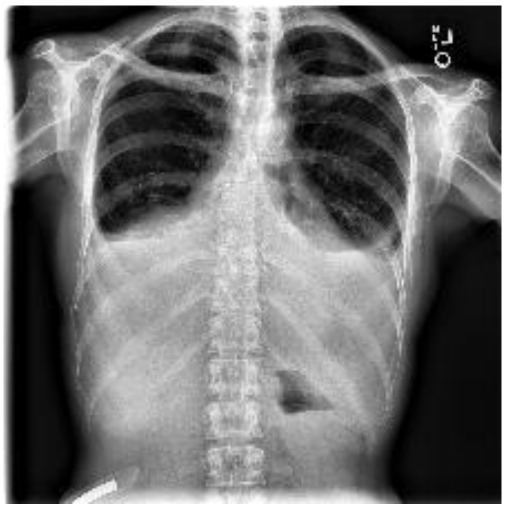
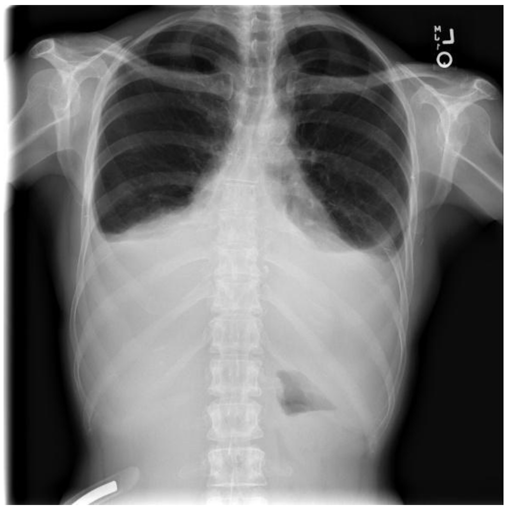



Figure 5. Processed version of the Chest X-Ray *Figure 6. Unprocessed version of the Chest X-Ray*

We plan to use a second preprocessing pipeline based on OpenCV and NumPy. First of all, we will take each input image and convert them to 8-bit grayscale in the range of [0, 255]. Then, we will make sure the image has a single channel and apply a median filter [11]. This method will reduce noise in images while keeping their edges. After that, we will use CLAHE to increase local contrast and then apply a simple sharpening kernel to highlight important structures [10]. Finally, we will normalize the processed image back to the range of [0, 1] in float32 format.

For each image, we plan to load it as a grayscale image with PIL and run this pipeline. We will resize the processed output to 224×224 pixels with Albumentations. After resizing, we will create a 3-channel RGB image by repeating the grayscale channel three times. Finally, we will scale the values to [0, 255], convert them to 8-bit, and save the result. This will give us images in a standard 224×224 RGB format that our deep learning model can use.

3.3.3. Scenario 3:

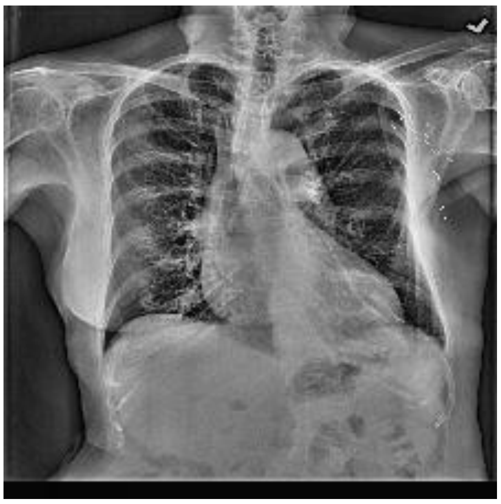


Figure 7. Processed version of the Chest X-Ray

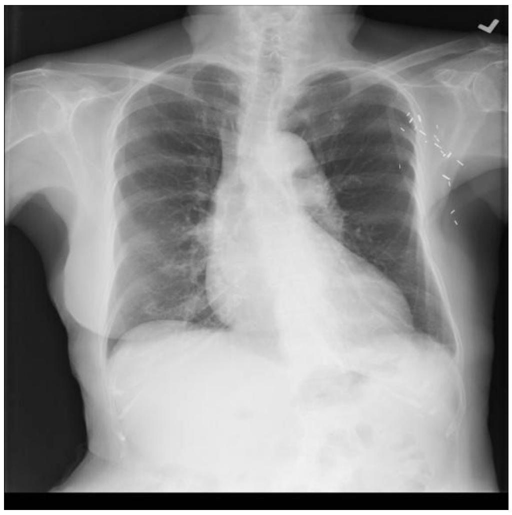


Figure 8. Unprocessed version of the Chest X-Ray

In this scenario, we plan to use a simple bone suppression step for chest X-ray images. First of all, we will take each input image and convert them to 8-bit grayscale in the range of [0, 255]. Then we will make sure the image has a single channel and apply a Gaussian blur to obtain a “bone-like” structure. Next, we will subtract a weighted part of this blurred image(original) to reduce the level of brightness of bones. After that, we will apply CLAHE to increase local contrast [3]. Finally, we will normalize the result to the [0, 1] range as a float32 image. The goal is to suppress bones and make soft tissue regions (lungs), more visible.

For each file, we will load the image in grayscale mode using PIL and convert it to a NumPy array. We will then apply the bone suppression processing and prepare the output for resizing by adding a channel dimension. The processed image will be resized to 224×224 pixels with Albumentations. After resizing, we will create a 3-channel RGB image by repeating the same grayscale channel three times. Last, we will scale the pixel values to [0, 255], convert them to 8- bit, and save the result to disk. This will give us bone-suppressed images in a standard 224×224 RGB format that our deep learning model can use.

3.3.4. Scenario 4:

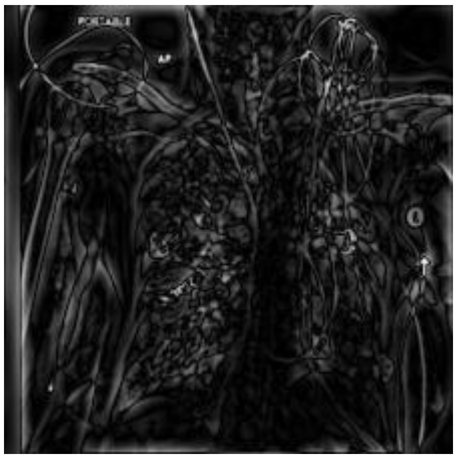


Figure 9. Processed version of the Chest X-Ray

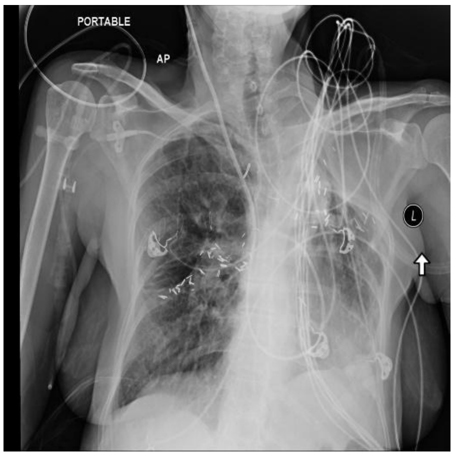


Figure 10. Unprocessed version of the Chest X-Ray

In this method, we plan to combine CLAHE and a Butterworth band-pass filter for preprocessing. First of all, convert each image to 8-bit grayscale and apply CLAHE to improve local contrast [10]. Then we will move to the frequency domain with a 2D FFT and apply a Butterworth band-pass filter to emphasize structural details while reducing very low and very high frequencies. This idea follows previous work on frequency-domain enhancement of chest X-rays [2].

For each image, we will load it as a grayscale image with PIL, apply the CLAHE + Butterworth pipeline, and then resize the result to 224×224 pixels using Albumentations. After resizing, we will create a 3-channel RGB image by repeating the grayscale channel three times, scale the values to [0, 255], convert to 8-bit, and save the result. This will give us a 224×224 RGB format that our deep learning model can use.

4. Methodology

4.1. Dataset & Preprocessing Methodology

4.1.1. NIH ChestX-ray14

Selection and the analysis of the dataset for this project, we selected the NIH ChestX-ray14 dataset, which is a standard for thoracic disease classification. We chose this dataset because it provides a sufficient amount of collection (112,120 chest X-ray images) which are labeled for 14 different conditions. During our analysis, we observed that the dataset has a multi- label structure. That means a patient could have multiple diseases at the same time.

We also identified a significant class imbalance issue. While common diseases like Infiltration had over 20,000 examples. Rare diseases like Hernia had very few samples. To prevent the model from ignoring these rare classes, we decided to create a weighted learning strategy for our training process. [8]

4.1.2. Dataset Filtering (PA view only)

Before starting the training process, we decided to filter the dataset based on the position of X-Ray views. The original dataset contains both Anterior-Posterior (AP) and Posterior- Anterior (PA) views. We chose to limit our dataset to PA views only, as they usually offer better standardization for deep learning models. We implemented a script that scanned the metadata and selected only the rows where the position of data was labeled as "PA". Based on this filtered list, we recreated the file directory and updated the training/testing split to ensure that no AP images were included in our final dataset.

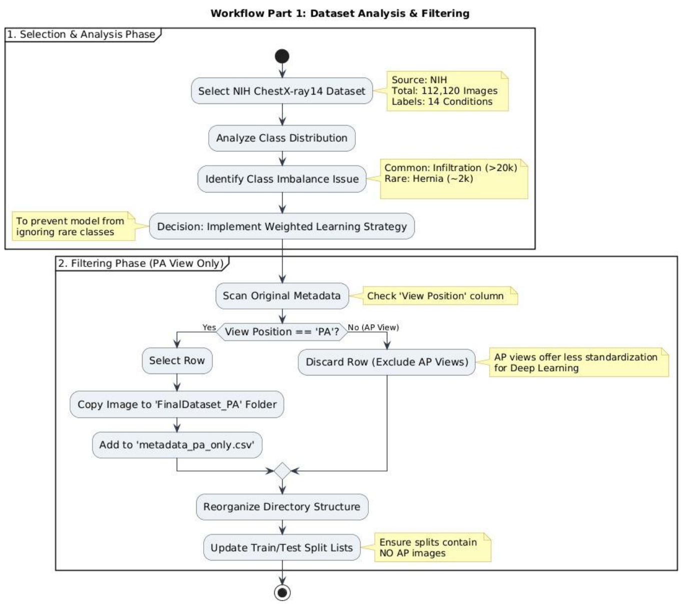

Workflow Part 1: Dataset Analysis & Filtering
1. Selection & Analysis Phase
Select NIH ChestX-ray14 Dataset
Source: NIH
Total: 112,120 Images
Labels: 14 Conditions
Analyze Class Distribution
Identify Class Imbalance Issue
Common: Infiltration (>20k)
Rare: Hernia (~2k)
To prevent model from ignoring rare classes
Decision: Implement Weighted Learning Strategy
2. Filtering Phase (PA View Only)
Scan Original Metadata
Check 'View Position' column
Yes
View Position == 'PA'?
No (AP View)
Select Row
Discard Row (Exclude AP Views)
AP views offer less standardization for Deep Learning
Copy Image to 'FinalDataset_PA' Folder
Add to 'metadata_pa_only.csv'
Reorganize Directory Structure
Update Train/Test Split Lists
Ensure splits contain NO AP images

Figure 11. Dataset Analysis and Filtering Workflow Diagram

4.1.3. Preprocessing Pipelines

Since unprocessed X-ray images generally contain noise or varying lighting conditions, we designed four different preprocessing scenarios to improve the model's performance.

- Intensity Enhancement: In the first scenario, we focused on improving contrast. We applied Gamma Correction and Contrast Limited Adaptive Histogram Equalization (CLAHE) to make the details of lungs more visible. We also added an unsharp mask to make clear the edges [10],[15].
- Noise Reduction: In the second scenario, we decided to reduce noise. We used a median filter to make the image smoother while preserving edges. Also, by CLAHE and a sharpening kernel, we highlighted the desired structures [10],[11].
- Bone Suppression: We experimented with a bone suppression technique to prevent the ribs from hiding lung abnormalities. We achieved this by subtracting a blurred version of the image from the original. That caused reducing the brightness of the bone structures and made the soft tissue clearer [10].
- Frequency Domain Filtering: Finally, we tested a method combining CLAHE with a Butterworth band-pass filter. This allowed us to emphasize specific structural details in the frequency domain while removing high-frequency noise [9],[10].

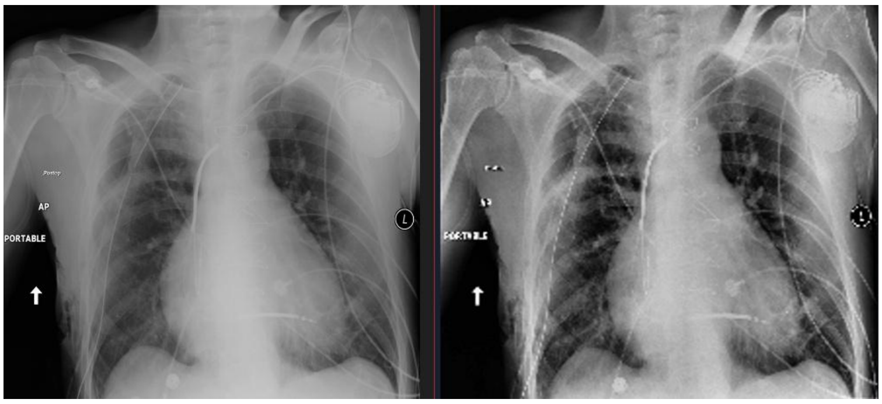


Figure 12. Intensity Enhancement

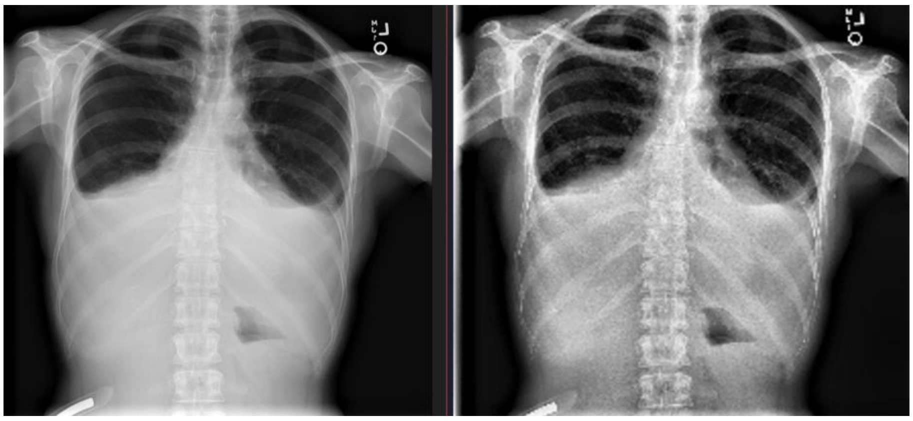


Figure 13. Noise Reduction

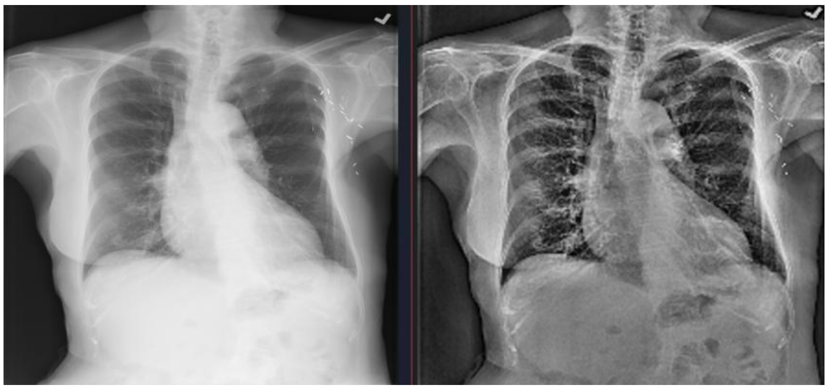


Figure 14. Bone Suppression

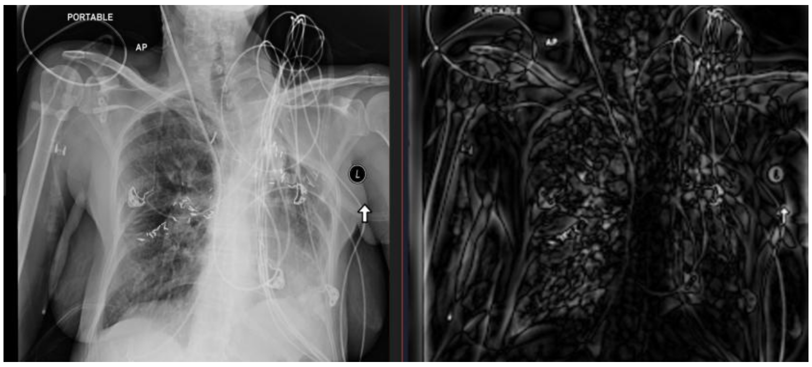
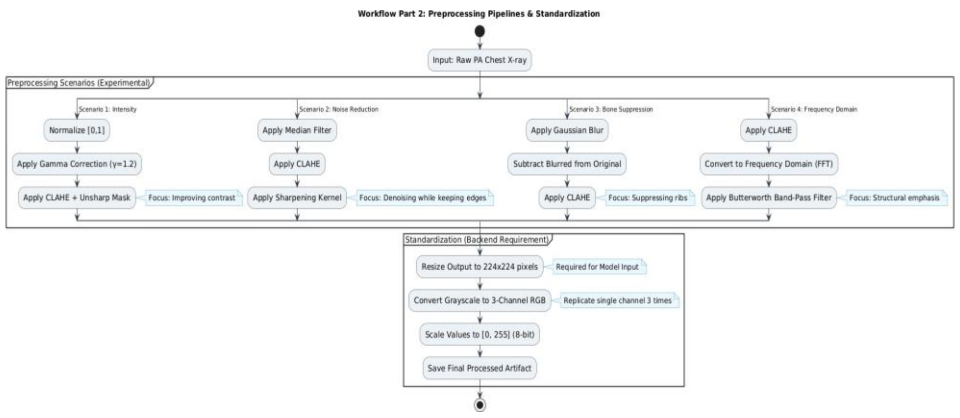


Figure 15. Frequency Domain Filtering

4.1.4. Standardization for the Backend

Independent of the chosen preprocessing strategy, we decided to put our processed dataset in a standard format. This method ensured the compatibility with our backend system. For every image in our dataset, we resized the resolution to 224x224 pixels. Even though the X-rays images in our dataset are originally in grayscale format we converted them into a 3- channel RGB format by repeating the grayscale channel three times. This decision was made to give us the opportunity to use pre-trained deep learning architectures (DenseNet, ResNet, etc.) that expect 3-channel input by default.



Workflow Part 2: Preprocessing Pipelines & Standardization
Input: Raw PA Chest X-ray
Preprocessing Scenarios (Experimental)
Scenario 1: Intensity
Normalize [0,1]
Apply Gamma Correction (γ=1.2)
Apply CLAHE + Unsharp Mask
Focus: Improving contrast
Scenario 2: Noise Reduction
Apply Median Filter
Apply CLAHE
Apply Sharpening Kernel
Focus: Denoising while keeping edges
Scenario 3: Bone Suppression
Apply Gaussian Blur
Subtract Blurred from Original
Apply CLAHE
Focus: Suppressing ribs
Scenario 4: Frequency Domain
Apply CLAHE
Convert to Frequency Domain (FFT)
Apply Butterworth Band-Pass Filter
Focus: Structural emphasis
Standardization (Backend Requirement)
Resize Output to 224x224 pixels
Required for Model Input
Convert Grayscale to 3-Channel RGB
Replicate single channel 3 times
Scale Values to [0, 255] (8-bit)
Save Final Processed Artifact

Figure 16. Preprocessing Pipelines and Workflow Diagram

4.2. Model Architecture

4.2.1. DenseNet-121 Backbone

We have chosen DenseNet as the backbone of our model due to its proven success in medical image classification tasks. The dense connectivity technique used in DenseNet allows the outputs of previous layers to be utilized in all future layers within a dense block in the model, which mitigates the possible vanishing gradient issues we may encounter [6]. The pretrained ImageNet weights of DenseNet also allow for faster convergence and stable training. The segmented structure of DenseNet allows for the insertion of different modules between its dense blocks, which is perfect for our project. The dense blocks are made up of bottleneck layers with dense connectivity and make up a large part of the model. Bottleneck layers start with batch normalization and ReLU followed by a 1x1 convolutional layer, after which another batch normalization and ReLU, and end with one final 3x3 convolutional layer [15].

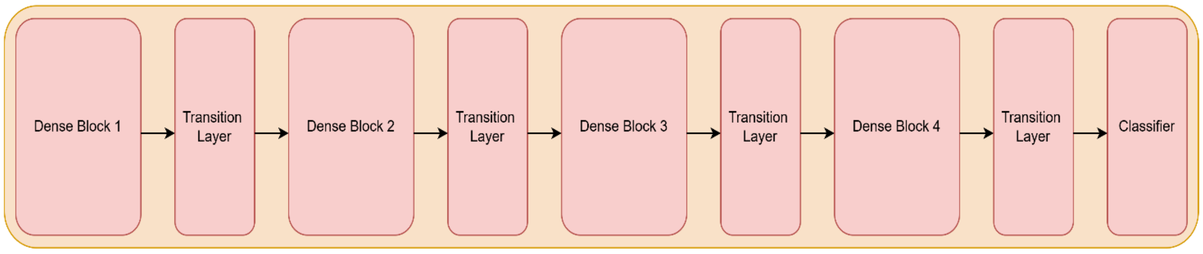


Figure 17. High Level Architecture of DenseNet-121

4.2.2. Convolutional Block Attention Module (CBAM)

CBAM is an attention module comprised of two smaller components called channel attention module and spatial attention module respectively [16]. Neither component affects the input or output sizes of the layers before and after them thus allowing for easy insertion into our model.

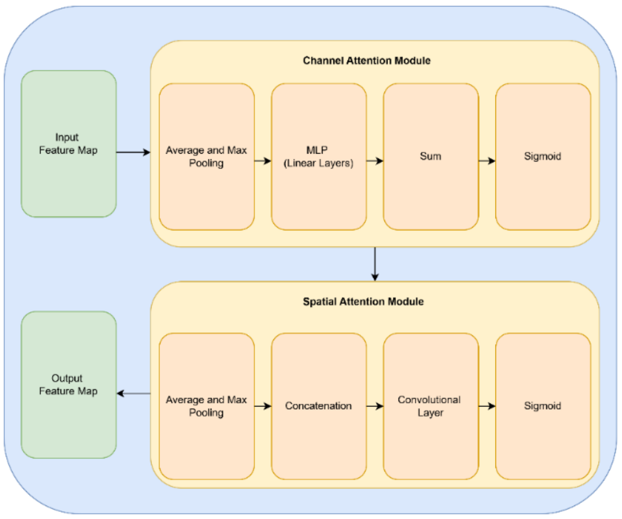


Figure 18. High Level Architecture of CBAM

4.2.2.1. Channel Attention Module

The channel attention module decides what features are important for a given task. The module applies average and max pooling to model the current features and passes them to an MLP. The sigmoid of the sum of the resulting features is multiplied with the current model features to update the feature map of the model [16].

4.2.2.2. Spatial Attention Module

The spatial attention module focuses on where important regions are on the feature maps. Similarly to the channel attention module, it applies average and max pooling across channels to generate maps that highlight where important features are. These feature maps are concatenated to make a feature map with two channels which is passed through a convolutional layer without changing its size while reducing the channel count back down to one. The sigmoid of the resulting map is multiplied with the current feature map to highlight important spatial regions [17].

4.2.2.3. Overall Proposed Architecture

Our model adds a CBAM module after each dense block without changing the feature map size to avoid compatibility issues with subsequent dense blocks. By adding this module after each dense block, we aim to create a model that is more capable of detecting the small differences present in X-ray images without causing an excessive processing load on devices. Due to the precise nature of diagnosing thoracic diseases from X-ray images, we aim to optimize our model's confidence and interpretability. We are planning on using F1 and F1 macro as our metrics during development to evaluate the performance of our model.

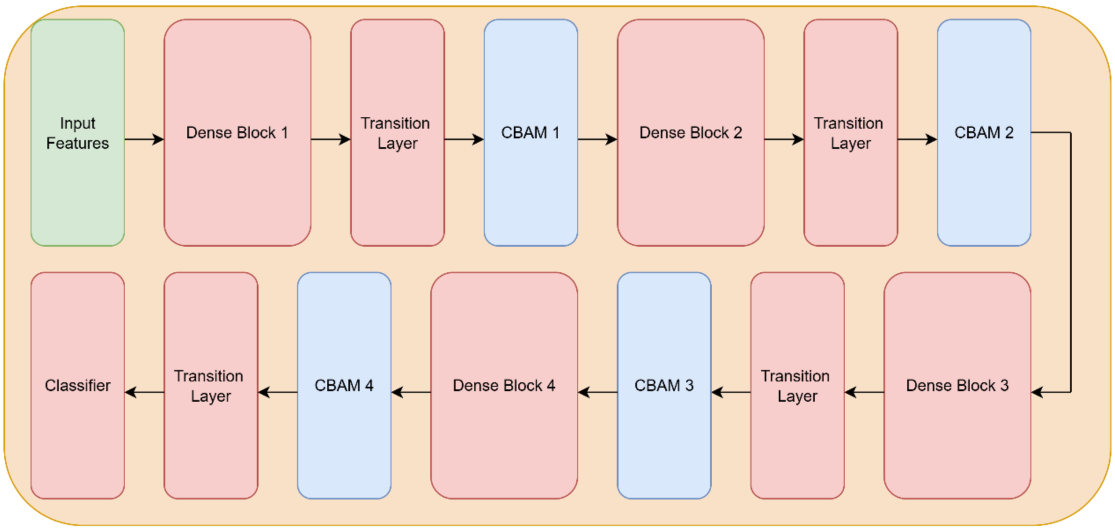


Figure 19. High Level Architecture of Our Model

4.2.3. Model Explainability Using Grad-CAM

Grad-CAM will be integrated into the model outputs to improve model interpretability. Grad-CAM will be used to visualize the areas that are relevant to the predictions our model makes by generating individual heatmaps over the original X-ray image for each possible disease class [4]. Through this technique, we aim to increase the trust of users by giving them the ability to see why the model came to a decision.

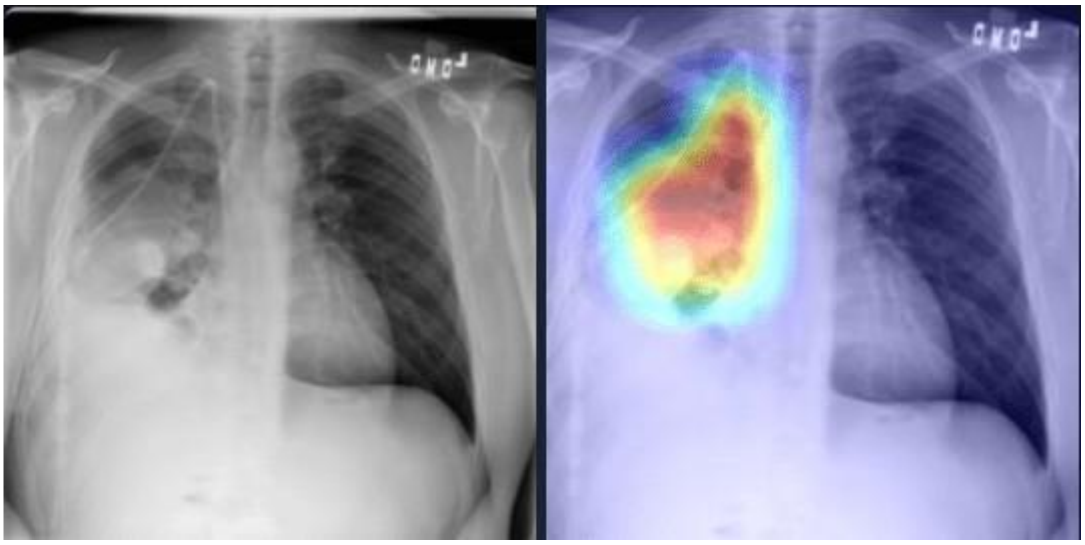


Figure 20. Original X-ray Image and The Grad-CAM Heatmap of The Same Image

4.3. Backend & Database Methodology with MongoDB

4.3.1. Backend Role in the System

In this project, the backend will be designed to manage the machine learning pipeline in a controlled manner and to provide a clean API for the system. After reviewing similar medical imaging projects, the system architecture will be designed to separate the model code from the application logic. This design will prevent the frontend from directly interacting with complex training scripts or local file structures. Instead, the backend will handle the entire workflow, including request handling, data preparation, model inference, and the delivery of consistent JSON responses.

This approach supports a workflow that includes steps such as preprocessing and explainability and it ensures that every output remains traceable. Managing these steps manually in medical imaging can lead to mistakes or inconsistent results [1], [4]. By positioning the backend as a central coordinator, each request follows the same execution logic, which increases overall reliability.

4.3.2. API Design Decisions

The backend will be implemented as a RESTful API service. During planning, the API will focus on the endpoints required by the user—such as uploading an image or retrieving a prediction—rather than exposing internal training details. This design will keep the communication between the frontend and backend simple and clear.

We also plan to standardize model outputs before sending them to the UI. Since the task is multi-label, the model will produce a probability value for each label [1], [4]. These raw probability scores will remain the primary stored result. A thresholding strategy will be used to convert probabilities into final predicted labels when required, and the selected threshold strategy will be stored so there is no confusion during later review.

4.3.3. Why MongoDB?

The database layer will use MongoDB. This choice aligns with the structure of the generated outputs: for each inference run, the system produces a "package" that includes metadata, predictions, and links to images. This naturally fits MongoDB's document- based storage model.

Medical image ML pipelines change frequently during development—preprocessing steps may change, and model versions may be updated [4]. MongoDB's flexible schema supports adding new fields or adjusting the stored structure without repeatedly redesigning strict relational tables, which supports faster iteration.

4.3.4. Data Stored in MongoDB

The system does not lock into an overly rigid schema early, but it defines the core record categories required for organization and traceability:

- Image Records: Unique IDs, basic metadata, and the path to where the file is stored.
- Preprocessing Records: Details on which filters or methods are used for repeatability [20].
- Model Records: Version information and checkpoint locations to identify which model produces each prediction.

- Prediction Records: Multi-label probability scores and the final labels based on the threshold strategy [1], [4].
- Explanation Records: Grad-CAM settings (e.g., target layer) and links to generated heatmap files [18].

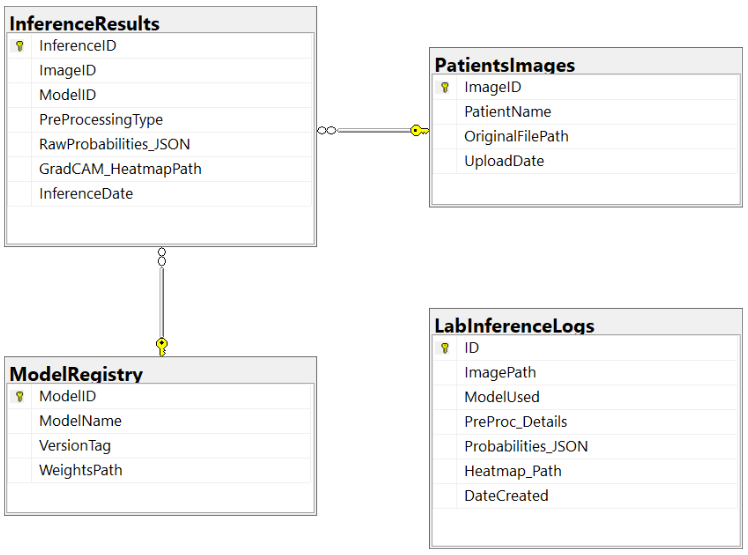


Figure 21. This table shows the logical structure of our database and how we link images to their specific model predictions and preprocessing steps.

4.3.5. File Storage Strategy

We plan to keep large files such as X-ray images and Grad-CAM overlays as regular files on the storage layer rather than storing them inside the database. In MongoDB, we will store only the file path (URI). This strategy will keep the database responsive and will allow the frontend to fetch images directly using the provided links.

4.3.6. Inference Workflow

When the backend receives an inference request, it will follow a consistent pipeline:

1. Input preparation: The backend will load the selected image artifact (original or preprocessed), apply the required resizing and channel formatting expected by the model, and normalize the input according to the model configuration.

2. Model selection: Inference will run using a specific model version (checkpoint). The backend will load the weights and maintain label ordering consistent with training [1], [4].

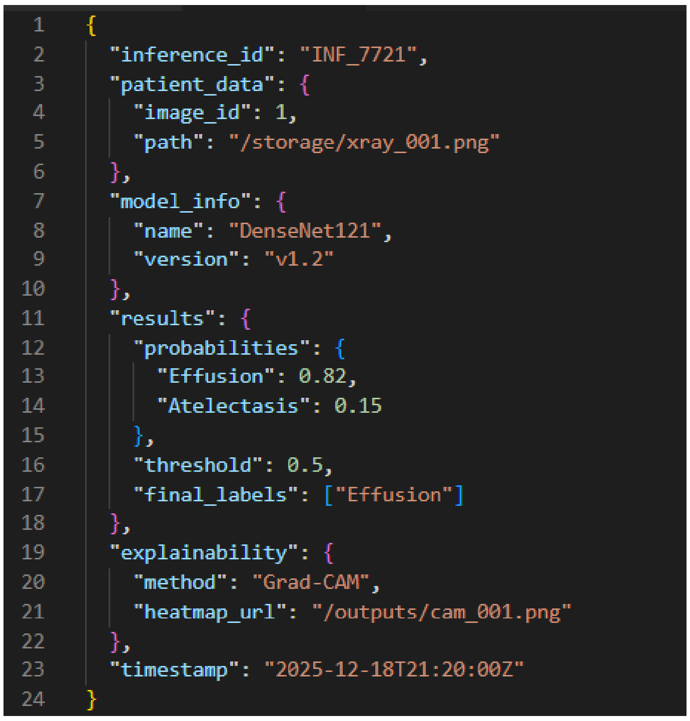
3. Forward pass and outputs: The model will produce a probability value per label (multi-label setup). These probabilities will be stored as the main prediction output.
4. Decision logic (thresholding): When needed, a threshold strategy will convert probabilities into a predicted label set, and the applied threshold strategy will be recorded to avoid later ambiguity.

A key decision is that prediction results will not only be displayed once; they will be stored as a structured record linked to the image reference and the model version that produces them. This will enable later retrieval (e.g., showing the latest prediction for a model) without rerunning inference.

4.3.7. Explainability Workflow (Grad-CAM)

We plan to integrate Grad-CAM directly into the backend so that explainability functions as a core capability rather than an external script [4]. This will help users understand which parts of the X-ray contribute to the model's decision.

We plan to treat Grad-CAM as a separate action because it typically requires additional computation compared to a standard prediction. The backend will generate the heatmap, save it as an image, and store the corresponding settings in the database. This will allow later inspection of how a specific model version produces a specific explanation for a given case.



```json
{
  "inference_id": "INF_7721",
  "patient_data": {
    "image_id": 1,
    "path": "/storage/xray_001.png"
  },
  "model_info": {
    "name": "DenseNet121",
    "version": "v1.2"
  },
  "results": {
    "probabilities": {
      "Effusion": 0.82,
      "Atelectasis": 0.15
    },
    "threshold": 0.5,
    "final_labels": ["Effusion"]
  },
  "explainability": {
    "method": "Grad-CAM",
    "heatmap_url": "/outputs/cam_001.png"
  },
  "timestamp": "2025-12-18T21:20:00Z"
}
```

Figure 22. An example of a JSON document from our database. It shows how we store the model's probability scores and the file paths for the Grad-CAM outputs in one place.

4.4. Backend

4.4.1. Frontend Interaction and API Integration

This section of our report explains the technical details of how our deep learning model will reach the end user and how data will be transferred.

4.4.1.1. API Design and Endpoints

We plan to use a modular RESTful API architecture on the server side to handle Flutter/Web (client) requests. Of course, endpoints such as POST/analyze, GET/results/{id}, and GET/history will be defined for basic functions.

The POST/analyze endpoint will be used as the gateway to initiate the analysis process when the user uploads the x-ray image. The GET/results/{id} endpoint will be where the results of the analysis and the outputs obtained through Grad-CAM are queried.
GET/history is planned to be the structure that lists the user's past queries[19].

4.4.1.2. Data Format and Communication Protocol

Due to platform independence and lightness, it has been decided that the format for all data exchanges on both the client and server sides will be JSON (JavaScript Object Notation). While this methodology and visual data are sent in the body of HTTP requests, the model's results, disease probabilities, and metadata will be communicated to the interface as JSON objects.

4.4.1.3. Integration Flow and Grad-CAM Visualization

The file referencing method will be the most useful bridge for integrating the Grad-CAM heat maps generated by our model into the interface. This method will work as follows: the heatmap to be generated will be saved to a directory on the server side, and the URL link for this image will be shared in the JSON response returned to the interface. As a result, instead of large image data being pulled directly into the database, it will be loaded as a media object via the server.

Analysis results and Grad-CAM outputs transmitted via the API are permanently stored on the backend for later access and traceability. This enables the system to support both real-time analysis and the management of past results.

4.4.1.4. Error Handling and Status Codes

System resilience is important in this project, as it is in every project. For this reason, we are considering developing an integrated structure with HTTP Status Codes for potential risks.

If an invalid file is uploaded (anything other than an x-ray), the user will be notified with a 400 Bad Request error and an explanatory text. Any issues that may occur on the server or model side will be prevented with a 500 Internal Server Error. We will enhance the resilience of our system with numerous alerts and informative notifications for other potential issues of this nature.

4.5. Frontend

4.5.1. Frontend Interaction and User Interface Design

This section of our report explains how the user will interact with our system through the Flutter-based frontend and how the results produced by the deep learning model will be presented to the user in a clear and understandable way.

The Flutter application will act as the client side of the system and will be responsible only for user interaction and visualization. All heavy operations such as image preprocessing, model inference, and Grad-CAM generation will be handled on the backend side [20],[21].

4.5.2. User Flow and Application Structure

We plan to design the Flutter application with a simple and intuitive user flow. First, the user will be able to select a chest X-ray image from their device. After the image is selected, it will be displayed on the screen so that the user can confirm the correct image before analysis.

Once the user starts the analysis, the selected image will be sent to the backend through the defined API endpoint. During this process, a loading indicator will be shown to inform the user that the analysis is in progress. After the backend completes the inference, the results will be sent back to the Flutter application and displayed to the user [23].

4.5.3. Result Visualization and Grad-CAM Integration

The results page of the application will display the predicted diseases together with their probability values. These values will help the user understand how confident the model is about each prediction.

For model explainability, Grad-CAM heatmaps generated on the backend will be visualized in the Flutter interface.

4.5.4. API Communication and Data Handling

Image uploads will be handled using multipart requests, while prediction results and metadata will be transferred in JSON format. [21],[22].

On the client side, the received JSON responses will be parsed and mapped to internal data structures. This structure will allow the application to safely display disease probabilities, Grad-CAM image references, and possible error messages returned by the backend. [22].

4.5.5. Error Handling and User Feedback

User experience and system robustness are important aspects of the frontend design. For this reason, the Flutter application will handle different error scenarios and provide informative feedback to the user.

If no image is selected or an unsupported file is uploaded, the user will be warned before the request is sent. In case of network problems or server-side errors, meaningful messages will be displayed and the user will be allowed to retry the operation [14].

5. Test Phase

5.1. Experimental Setup and Configuration:

For the system validation, we conducted the training using the NIH ChestX-ray14 dataset. As detailed in our methodology, we filtered the dataset to include only Posterior-Anterior (PA) views to ensure the standardization of the dataset. The dataset was split into three subsets: 80% Training, 10% Validation, and 10% Testing. Different from the standard 224x224 input size mentioned in the initial design, we increased the input resolution to 512x512 pixels during this training phase to capture finer details in the X-ray images.

We utilized the DenseNet-121 backbone integrated with Convolutional Block Attention Modules (CBAM). The training hyperparameters were configured as follows:

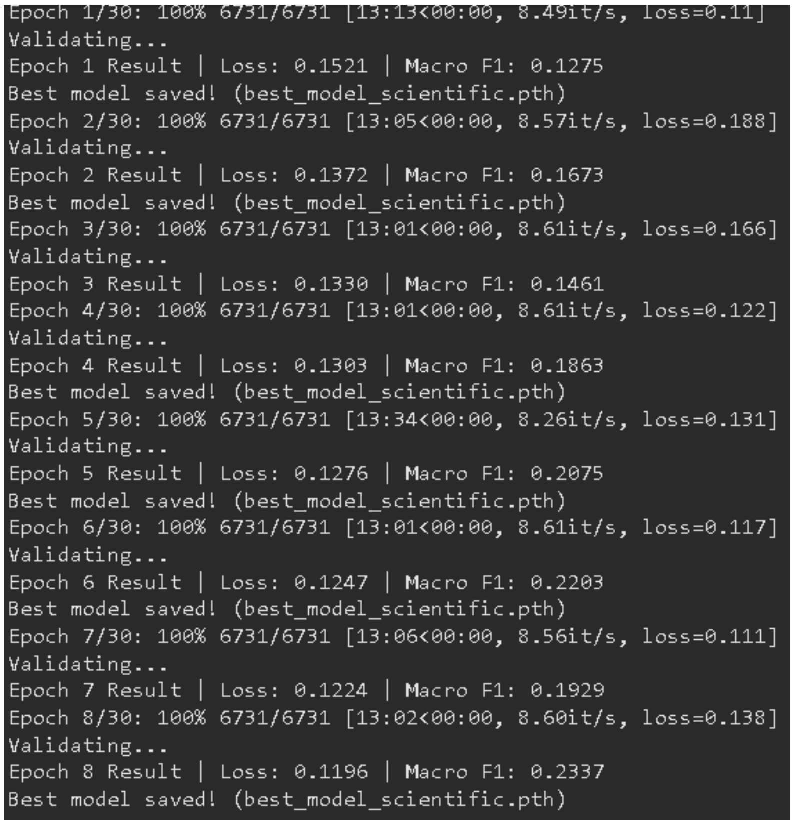
- Epochs: 30
- Batch Size: 8
- Gradient Accumulation Steps: 4
- Learning Rate: 1e-4
- Decision Threshold: 0.5

Also, to improve the model's generalization capabilities, we applied data augmentation techniques, including random mirror (horizontal flip) and random rotation within a range of -7 to +7 degrees.

5.2. Training Analysis and Resource Duration:

The model achieved a stable training process throughout the 30 epochs. This took approximately 13 minutes for one epoch in total for the training process, as well as 1.5 minutes for every validation step, making a total of 7 hours and 15 minutes for the entire training process.

According to the logs, the model showed consistent convergence as the loss value decreased from 0.152 in the first epoch to 0.046 in the final epoch. Additionally, the Macro F1 score increased in the initial stages, peaking at 0.2868 during Epoch 16. As a result, the model weights from this specific epoch were saved as the "Best Model" to guarantee that the most optimal version was used for the subsequent testing phase.

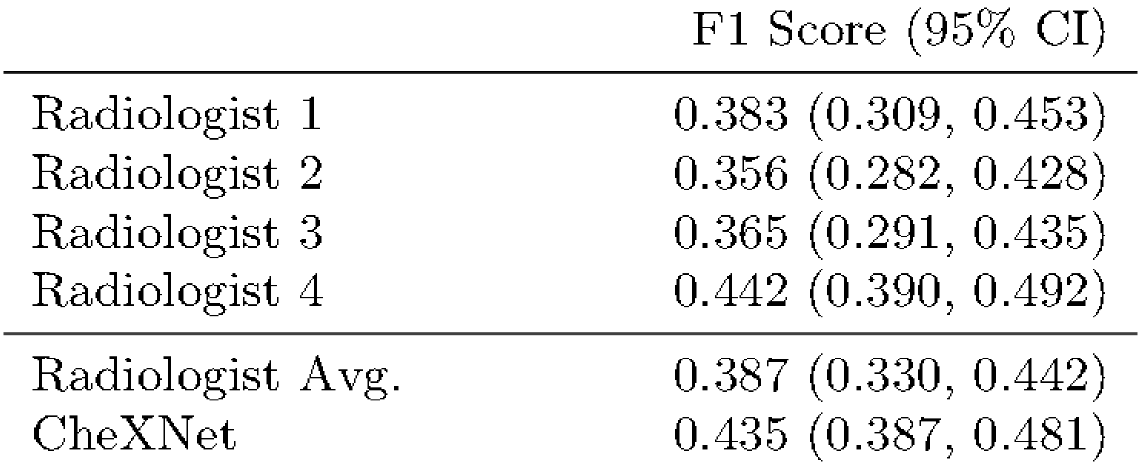


```
Epoch 1/30: 100% 6731/6731 [13:13<00:00, 8.49it/s, loss=0.11]
Validating...
Epoch 1 Result | Loss: 0.1521 | Macro F1: 0.1275
Best model saved! (best_model_scientific.pth)
Epoch 2/30: 100% 6731/6731 [13:05<00:00, 8.57it/s, loss=0.188]
Validating...
Epoch 2 Result | Loss: 0.1372 | Macro F1: 0.1673
Best model saved! (best_model_scientific.pth)
Epoch 3/30: 100% 6731/6731 [13:01<00:00, 8.61it/s, loss=0.166]
Validating...
Epoch 3 Result | Loss: 0.1330 | Macro F1: 0.1461
Epoch 4/30: 100% 6731/6731 [13:01<00:00, 8.61it/s, loss=0.122]
Validating...
Epoch 4 Result | Loss: 0.1303 | Macro F1: 0.1863
Best model saved! (best_model_scientific.pth)
Epoch 5/30: 100% 6731/6731 [13:34<00:00, 8.26it/s, loss=0.131]
Validating...
Epoch 5 Result | Loss: 0.1276 | Macro F1: 0.2075
Best model saved! (best_model_scientific.pth)
Epoch 6/30: 100% 6731/6731 [13:01<00:00, 8.61it/s, loss=0.117]
Validating...
Epoch 6 Result | Loss: 0.1247 | Macro F1: 0.2203
Best model saved! (best_model_scientific.pth)
Epoch 7/30: 100% 6731/6731 [13:06<00:00, 8.56it/s, loss=0.111]
Validating...
Epoch 7 Result | Loss: 0.1224 | Macro F1: 0.1929
Epoch 8/30: 100% 6731/6731 [13:02<00:00, 8.60it/s, loss=0.138]
Validating...
Epoch 8 Result | Loss: 0.1196 | Macro F1: 0.2337
Best model saved! (best_model_scientific.pth)
```

Figure 23. Results of the First 8 Epoch

5.3. Literature Benchmark and Comparison:

We used the results from CheXNet as a benchmark for the success of our model. Our model achieved an F1 score of 0.28 in multi-label classification. Although this result cannot be directly compared to the results of CheXNet due to binary pneumonia classification being the target of the F1 scores provided, CheXNet can still be considered a relevant comparison. The CheXNet paper has reached an F1 score of 0.44, which indicates that their model performance was better overall compared to our model. The lower F1 score our model yields is likely due to the multi-label nature of our model along with class imbalance issues, therefore this comparison needs to be considered with caution. The radiologist results provided alongside the CheXNet results are also high compared to our model. The radiologists yielded an average F1 score of 0.39. This result is also high compared to our model but needs to be considered with the same caveats as the CheXNet model. We believe a large part of our model's low performance to be due to a lack of training optimization. Due to insufficient hardware and time constraint issues, we trained our model within our means. With better optimized training parameters and hardware, we expect our model to achieve better performance.



	F1 Score (95% CI)
Radiologist 1	0.383 (0.309, 0.453)
Radiologist 2	0.356 (0.282, 0.428)
Radiologist 3	0.365 (0.291, 0.435)
Radiologist 4	0.442 (0.390, 0.492)
Radiologist Avg.	0.387 (0.330, 0.442)
CheXNet	0.435 (0.387, 0.481)

Figure 24. F1 Scores in the CheXNet Article

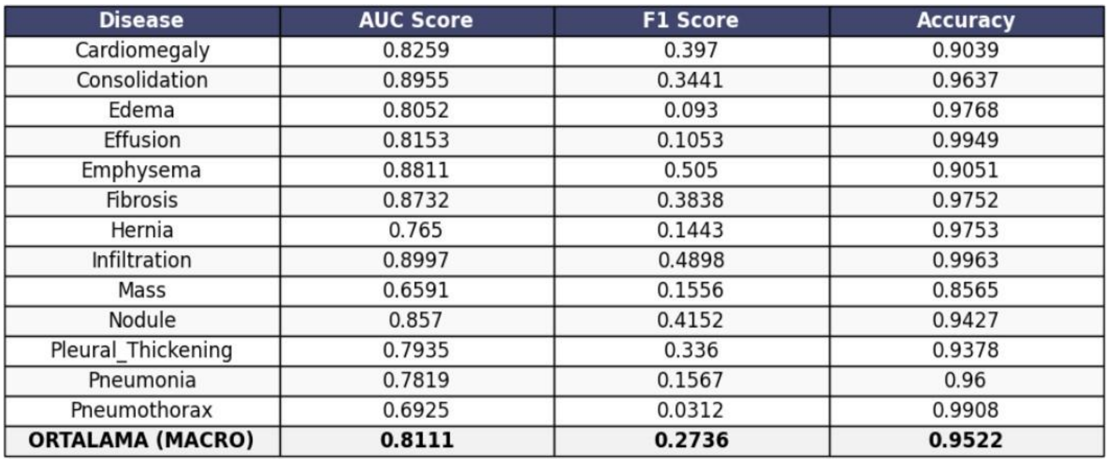
5.4. Quantitative Results and Performance:

Evaluating the model on the unseen test data was done using five different metrics: AUC, F1, Accuracy, Precision, and Recall. The following are the results:

Overall Performance: The model was able to record a macro average AUC Score of 0.811, which shows that it is capable of distinguishing between classes. The average Accuracy was 95.2%, although this number is influenced by the high number of negative samples.

Class-Specific Analysis: The highest performance was seen in Consolidation, Infiltration, and Emphysema. AUC scores of these diseases were between 0.88-0.90 and they were signifying strong feature extraction for these distinct pathologies. The classes that had low performance were Mass and Pneumothorax with an AUC of 0.65 and 0.69. This shows that while the model is effective for most classes, it faces challenges with less defined visual patterns.

The Imbalance Problem: Despite applying data augmentation techniques like random rotation and mirroring, class imbalance remains as a challenge. Rare classes such as Hernia show high accuracy (97.5%) but extremely low Recall (0.09). This shows that the model is effective at identifying negative cases but struggles to detect positive instances. This proves that augmentation alone was insufficient to fully resolve the data scarcity for rare pathologies.



Disease	AUC Score	F1 Score	Accuracy
Cardiomegaly	0.8259	0.397	0.9039
Consolidation	0.8955	0.3441	0.9637
Edema	0.8052	0.093	0.9768
Effusion	0.8153	0.1053	0.9949
Emphysema	0.8811	0.505	0.9051
Fibrosis	0.8732	0.3838	0.9752
Hernia	0.765	0.1443	0.9753
Infiltration	0.8997	0.4898	0.9963
Mass	0.6591	0.1556	0.8565
Nodule	0.857	0.4152	0.9427
Pleural_Thickening	0.7935	0.336	0.9378
Pneumonia	0.7819	0.1567	0.96
Pneumothorax	0.6925	0.0312	0.9908
ORTALAMA (MACRO)	**0.8111**	**0.2736**	**0.9522**

Figure 25. Results of Our Model

5.5. Conclusion:

The test results confirm that our system, powered by DenseNet-121 and CBAM, is capable of multi-label classification with a satisfactory AUC of 0.81. While the model is accurate in identifying negative cases, future work should focus on improving Recall for rare classes to make the system more sensitive in a clinical setting.

6. Flutter

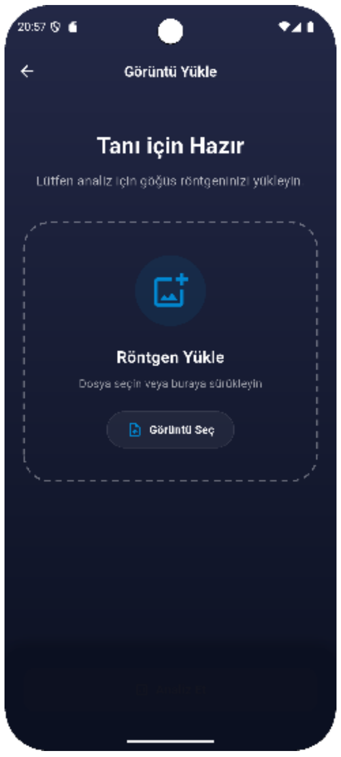
6.1. Interface Design Architecture and User Flow

The Flutter-based mobile application was designed as a lightweight, client-side interface whose primary purpose is to facilitate user interaction and visualization. All computationally intensive operations, including image preprocessing, model inference, and Grad-CAM generation, are handled entirely on the backend server. The mobile application is responsible solely for image acquisition, user feedback, and result presentation.

The user interface follows a minimalist and functional design approach to ensure that it does not interrupt the workflow of healthcare personnel. The system interaction is structured into three fundamental phases.

6.1.1. Image Acquisition and Pre-processing

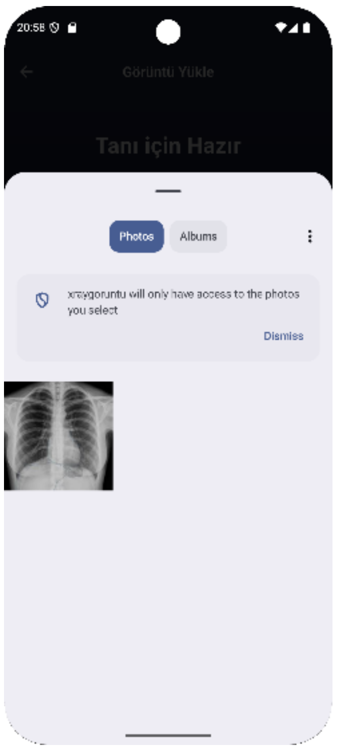
Users can upload chest X-ray images to the system by selecting an image from the device gallery via the Image Upload module.The interface incorporates visual affordances to guide correct user interaction and prevent erroneous data submission. To clearly indicate the active input area, the upload zone is visually highlighted, ensuring that users understand where and how to submit the image. Once an image is selected, it is displayed on the screen for confirmation, allowing the user to verify the correctness of the input before initiating the analysis process.



Lütfen analiz için göğüs röntgeninizi yükleyin

Dosya seçin veya buraya sürükleyin

Figure 26. Initial Upload Screen with Clean UI



xraygoruntu will only have access to the photos you select

Figure 27. Image Selected State

6.1.2. Process Feedback

During the inference phase, the analysis is performed on the backend server. To prevent the user from perceiving the system as unresponsive during this process, real-time visual feedback mechanisms are employed within the Flutter interface.

Circular progress indicators are displayed while the backend processes the submitted image. This visual waiting screen communicates that the analysis is ongoing and effectively minimizes the negative impact of computational latency on the user experience.

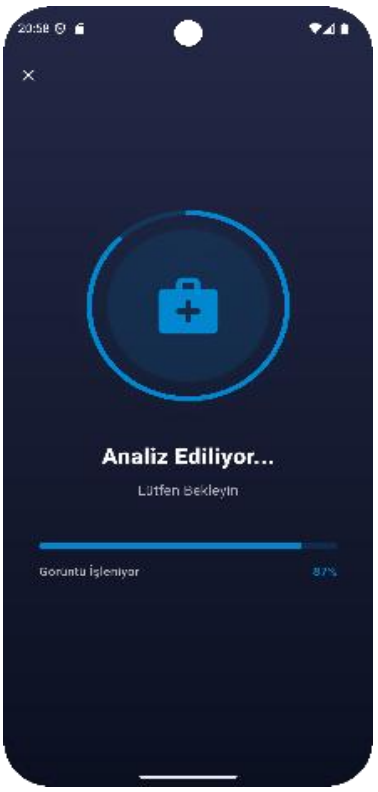


Figure 28. Process Feedback and Analysis Screen

6.1.3. Result Visualization and Decision Support

TheThe multi-label classification results returned by the backend are presented to the user through a structured, two-layer visualization approach, as illustrated in Figure 4.

Grad-CAM Integration:

Grad-CAM activation maps generated on the backend are overlaid on the original X-ray image and displayed in the application interface. This visualization highlights the regions that contributed most significantly to the model's predictions, thereby enhancing model interpretability and supporting explainable artificial intelligence (XAI) principles. The heatmap visualization is enabled by default to ensure immediate interpretability.

Probability Distribution Panel:

In addition to visual explanations, a probability distribution panel is used to present the predicted likelihoods for each pathology. A dynamic color-coding strategy is applied to reduce the risk of overlooking critical findings. The class with the highest predicted probability (dominant class) is emphasized using red-colored indicators, while secondary predictions are displayed using neutral tones, providing a clear hierarchical risk representation.

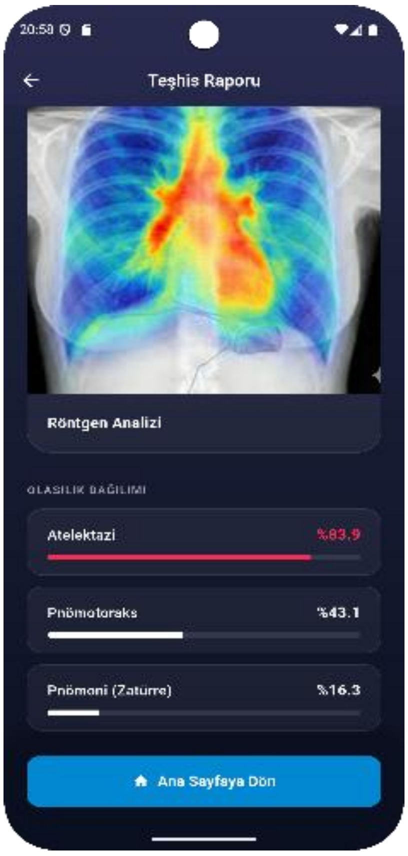


Figure 29. Result Screen with Heatmap & Probability Distribution

6.2. Visual Language and Ergonomics

The visual design of the application was guided by usability and ergonomic considerations. Medical Blue (#0288D1), a color commonly associated with trust and professionalism in healthcare-related literature, was selected as the primary accent color. Dark navy tones (#0B1021) were used for background and card components to reduce eye strain during extended usage and to ensure high contrast for radiographic imagery.

Typography choices prioritized readability and clarity, establishing a clear visual hierarchy between numerical data, such as probability values, and textual information, including disease labels. This approach improves information scanning and reduces cognitive load for the user.

6.3. Conclusion

In conclusion, the developed Flutter interface effectively transforms the complex deep learning pipeline operating on the backend into an accessible and interpretable user experience. By focusing exclusively on interaction design, visualization, and explainability, the mobile application serves as a prototype clinical decision-support interface suitable for research and demonstration purposes.

This design ensures that advanced AI outputs can be presented in a clear and user-friendly manner without embedding computational logic within the client-side application.

7. TİGA & Survey

7.1. Interaction and Mentorship Engagement with TİGA

As part of the project development process, our team actively requested feedback and mentorship from industry professionals to evaluate the real-world use of our system. In this context, we conducted an on-site meeting with TİGA[1], a company operating in the field of defense and health-related technologies. During this visit, we presented our project through a structured technical presentation, outlining the problem definition, dataset characteristics, model architecture, explainability approach, and the planned end-to-end system design.

The primary objective of this meeting was to introduce our proposed solution, request technical mentorship, and assess whether our design choices aligned with real-world expectations in medical AI applications. The presentation emphasized our focus on multi-label chest X-ray disease classification, the use of Grad-CAM for explainability, and the integration of the model into a user-facing Flutter-based interface.

Following the presentation, we received expert feedback regarding model selection, explainability requirements, and system usability. In particular, the importance of transparent model outputs, clear visualization of attention regions, and non-intrusive clinical usage was highlighted. These discussions supported our decision to treat explainability as a core system component rather than an optional feature. Additionally, the need for a modular backend structure and a simple, platform-independent frontend was emphasized, supporting our existing design choices.

Although the mentorship process is ongoing, this interaction played a guiding role in validating our project direction. It helped us frame the system not only as an academic deep learning model, but as a prototype that considers clinical workflow constraints, interpretability expectations, and user experience.

7.2. User Survey Study and Design-Driven System Adaptation

To better understand potential end users' expectations and perceptions of AI-supported medical imaging systems, we conducted a pilot survey study targeting healthcare professionals. The survey collected responses from participants with diverse roles, including radiologists, medical doctors from different specialties, nurses, interns, and healthcare staff.

Survey Structure

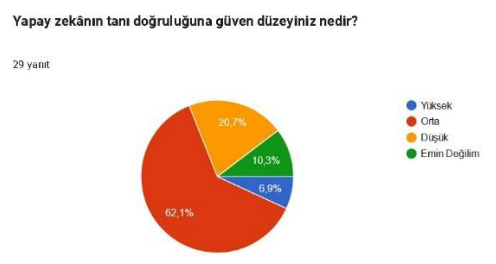
The survey included questions covering:

- Trust level in AI-based diagnostic accuracy
- Perceived usefulness of explainability visuals (e.g., Grad-CAM heatmaps)
- Preferred usage platform (mobile vs. desktop/web)
- Preferred result representation (text-based, visual, or combined)
- Intended usage scenario in clinical or educational workflows

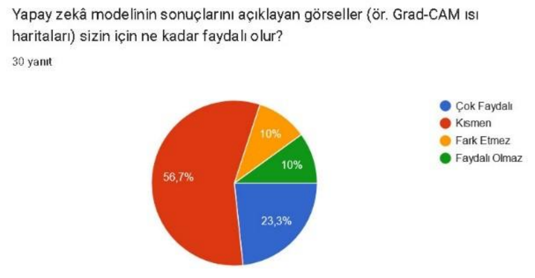
Key Findings

Analysis of the responses revealed several notable trends:

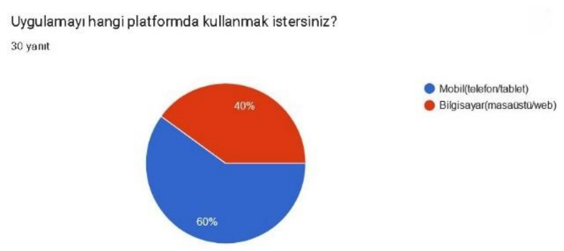
- **Trust in AI systems** was generally reported as low to moderate. Only a small portion of participants indicated high trust, suggesting that AI tools are currently viewed more as supportive systems rather than primary decision-makers.



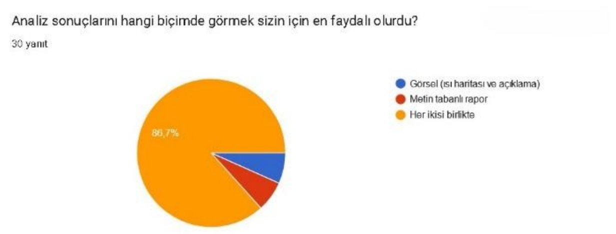
- **Explainability visuals** such as Grad-CAM heatmaps were perceived as partially or highly useful by most respondents, particularly among radiologists, interns, and emergency medicine professionals. Very few participants stated that such visuals would not be useful.



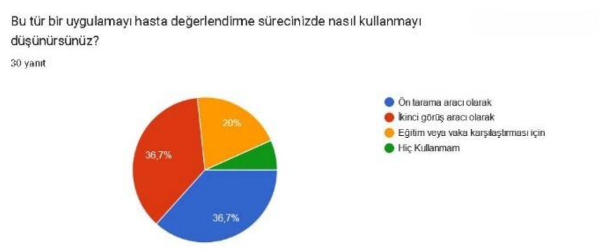
- **Platform preference** showed a strong inclination toward mobile devices (phones/tablets), especially among interns, nurses, and general practitioners, while radiology specialists more frequently preferred desktop or web-based platforms.



- **Result format preference** was dominated by a combined presentation, where numerical probabilities are supported by visual explanations. This indicates that neither text nor visuals alone are sufficient for most users.



- Regarding **intended usage**, the most common responses were pre-screening tool and second opinion support, followed by educational and case comparison purposes. Only a small subset stated they would not use such a system.



7.3. Impact on System Design

The survey results directly influenced several design decisions in our project. Based on the preference for combined outputs, we ensured that the frontend presents both probability scores and Grad-CAM visualizations together. The strong mobile usage tendency motivated us to prioritize a Flutter-based mobile-first interface, while still maintaining compatibility with web platforms.

Furthermore, the moderate trust levels reported by participants reinforced our emphasis on explainability and transparency. Instead of positioning the system as a diagnostic authority, we explicitly designed it as a decision-support and educational tool, aligning with how healthcare professionals indicated they would realistically use such an application.

Overall, the survey study allowed us to align technical development with user expectations, ensuring that our system design decisions were informed not only by literature and performance metrics, but also by real user perspectives.

8. Conclusion

The first phase of this project successfully established a functional end-to-end pipeline for multi-label chest X-ray classification. Our experimental results, using the DenseNet-121 and CBAM architecture, yielded a macro average AUC score of 0.811 and an overall accuracy of 95.2% on the NIH ChestX-ray14 dataset. The integration of Grad-CAM has proven effective in highlighting relevant pathological regions, addressing the critical requirement for explainability in medical AI.

However, the results also identified key challenges that will be the focus of the upcoming semester. While the model excels at identifying negative cases, the recall for rare diseases (such as Hernia) remains low due to significant class imbalance. Additionally, mentorship feedback from TİGA and user surveys confirmed that while the Flutter-based mobile interface is highly desired, increasing user trust through further model optimization is essential. In the next stage of development (CENG 408), we aim to refine our training strategies to handle data scarcity better, explore multimodal data integration, and conduct more rigorous testing to enhance the system's clinical utility as a second-opinion tool.

9. References

[1] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., & Summers, R. M. (2017). ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* https://openaccess.thecvf.com/content_cvpr_2017/papers/Wang_ChestX-ray8_Hospital-Scale_Chest_CVPR_2017_paper.pdf

[2] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., Lungren, M. P., & Ng, A. Y. (2017). CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning. *arXiv preprint arXiv:1711.05225.* https://arxiv.org/abs/1711.05225

[3] Rajpurkar, P., Irvin, J., Bagul, A., Ko, M., Taylor, R., Duan, T., Poplin, R., Chen, P., Jones, E., Seal, P., & Ng, A. Y. (2018). Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. *PLoS Medicine, 15*(11), e1002686. https://journals.plos.org/plosmedicine/article/file?id=10.1371/journal.pmed.1002686&type=printable

[4] Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., Seekins, J., Mong, D. A., Halabi, S. S., Wilson, M., Mooney, C., Anand, V., Pursnani, D., Goyal, P., Naidoo, K., ... Ng, A. Y. (2019). CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. *AAAI Conference on Artificial Intelligence.* https://arxiv.org/abs/1901.07031

[5] Akmal, M. R., Bintoro, K. B., & Purboyo, W. T. (2019). Chest X-Ray Image Classification on Common Thorax Diseases using GLCM and AlexNet Deep Features. *International Journal of Integrated Engineering, 11*(4), 183–193.

[6] Taslimi, S., Taslimi, M., & Aghabozorgi, S. (2022). SwinCheX: Multi-label classification on chest X-ray images with transformers. *arXiv preprint arXiv:2206.04246.* https://arxiv.org/abs/2206.04246

[7] Gao, Y., Xie, H., Chen, R., Huang, Y., Guo, Y., Zhang, Y., & Dong, F. (2025). Pneumonia Detection and Analysis Using AlexNet. *Applied and Computational Engineering, 190*(1), 1– 7. https://www.researchgate.net/publication/396128636_Pneumonia_Detection_and_Analysis_Using_AlexNet

[8] National Institutes of Health, “NIH Chest X-rays Dataset,” Kaggle, [Online].

Available: https://www.kaggle.com/datasets/nih-chest-xrays/data.

Accessed: 9 Dec. 2025.

[9] A. Giełczyk, A. Marciniak, M. Tarczewska & Z. Lutowski, “Pre-processing methods in chest X-ray image classification,” *PLOS ONE*, vol. 17, no. 4, e0265949, Apr. 2022.

Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0265949

Accessed: 9 Dec. 2025.

[10] IEEE Xplore, “Article (document no. 10955260),” [Online].

Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10955260.

Accessed: 9 Dec. 2025.

[11] “Median Filter – an overview,” *ScienceDirect Topics*, Elsevier, [Online].

Available: https://www.sciencedirect.com/topics/engineering/median-filter.

Accessed: 9 Dec. 2025.

[12] N. F. Sahib and Z. A. Hashim, “Contrast Image Enhancement by Gamma Correction,” *Computer Engineering and Intelligent Systems*, vol. 9, no. 7, 2018.

Available:https://scispace.com/pdf/contrast-image-enhancement-by-gamma-correction-3upr34p2k1.pdf.

Accessed: 9 Dec. 2025.

[13] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, [Online].
Available: https://arxiv.org/abs/1608.06993
Accessed: 20 Dec. 2025.

[14] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, “CBAM: Convolutional Block Attention Module,” *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, [Online].
Available: https://arxiv.org/abs/1807.06521
Accessed: 20 Dec. 2025.

[15] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” *IEEE International Conference on Computer Vision (ICCV)*, 2017, [Online].
Available: https://arxiv.org/abs/1610.02391
Accessed: 19 Dec. 2025.

[16] I. E. Ihongbe et al., “Evaluating Explainable Artificial Intelligence (XAI) techniques in chest radiology imaging through a human-centered lens,” *PLOS ONE*, 2024.

[17] J. Han et al., “BO-CLAHE enhancing neonatal chest X-ray image quality for improved lesion classification,” *Scientific Reports*, 2025.

[18] Flutter Team. Flutter Documentation.
https://docs.flutter.de

[19] Google Developers. Flutter Architectural Overview.
https://docs.flutter.dev/resources/architectural-overview

[20] ISO/IEC 25010:2011. Systems and software Quality Requirements and Evaluation (SQuaRE).

[21] ECMA International. (2017). The JSON Data Interchange Syntax (ECMA-404).

[22] Mozilla Developer Network (MDN). HTTP Overview and Status Code.