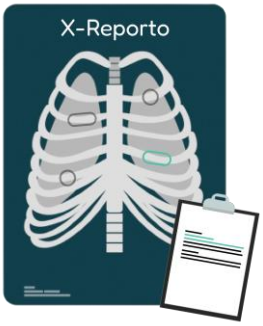
Cairo University
Faculty of Engineering
Department of Computer Engineering

X-Reporto



A Graduation Project Report Submitted

to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

Presented by

Ahmed Hosny Abdelrazik

Supervised by

Dr. Yahia Zakaria

July 2024

Abstract

This project addresses the increasing burden on radiologists due to the rising prevalence of chest diseases, which results in long queues of X-ray reports needing diagnosis. The objective of the project is to develop a semi-automated reporting system that supports radiologists by generating preliminary reports for each anatomical region and identifying diseases in those regions. Our approach involves enhancing chest X-ray images to correct defects caused by the X-ray devices, ensuring that critical diagnostic information is preserved and easily identifiable. The tool generates comprehensive, template-based reports tailored to the specific diseases identified, thereby streamlining the reporting process and reducing the time required for diagnosis.

The primary outputs of the project include the enhanced X-ray images and the detailed, disease-specific reports generated by the tool. Development and testing of the tool were conducted to ensure accuracy and reliability. Testing results demonstrate significant improvements in image clarity and diagnostic accuracy, as well as a reduction in the time required for report generation.

This project successfully developed and implemented the tool. The outcomes of this project not only alleviate the workload of radiologists but also contribute to better patient care by enabling faster and more precise medical interventions, ultimately increasing patient survival rates.

This project is sponsored by Voyance Health [1], a software development company in the medical field, which has provided invaluable support by offering server resources for training models on our large dataset. They also contributed the basic idea of the project and have been closely following our progress, offering guidance and feedback throughout the development process.

[1] https://voyance.health/

Table of Contents

List of Figures

List of Tables

List of Abbreviation

AI	Artificial Intelligence
API	Application Programming Interface
CAM	Class Activation Mapping
CNN	Convolutional Neural Networks
CXR	Chest X-Ray
DL	Deep Learning
EDA	Exploratory Data Analysis
ER	Entity-Relationship
FC	Fully Connected Layer
GAN	Generative Adversarial Networks
GAP	Global Average Pooling
GMP	Global Max Pooling
GPU	Graphical Processing Unit
GPT	Generative Pre-trained Transformer
HOG	Histogram of Oriented Gradients
NAN	Not a Number
NN	Neural Networks
PA	Posteroanterior
REST	Representational State Transfer
RCNN	Region-based Convolutional Neural Network
ROI	Region Of Interest
RPN	Region Proposal Network
STD	Standard Deviation
VRAM	Video Random Access Memory
VGG	Visual Geometry Group
VinDr-CXR	Vingroup Data Chest X-Ray

List of Symbols

σ (Noise standard deviation)
α (Learning rate)
β (Momentum term)
λ (Regularization parameter)
B (Buffer size, Batch size)
D (Discriminator)
G (Generator)
fop (Operating frequency)
L (Loss function)
N (Number of layers, Number of samples)
P (Probability)
R (Residual connection)
T (Threshold)
W (Weights)

Contacts

Name	Email	Phone Number
Ahmed Hosny	ahmed.alghany01@eng-st.cu.edu.eg	+2 01060668268

Supervisor

Name	Email	Number
Dr. Yahia Zakria	Yahiazakaria13@gmail.com	+2 0111 729 7303

Chapter 1: Introduction

X-Reporto is a specialized tool designed to alleviate the challenges faced by radiologists in managing an increasing number of patients and optimizing the medical reporting process for chest x-ray images. It features advanced capabilities such as enhancing chest x-ray images affected by device defects, ensuring clearer and more accurate diagnostic images. Additionally, X-Reporto automates the generation of detailed reports based on chest x-ray findings, utilizing templates that streamline the reporting workflow. By analyzing the images, X-Reporto provides insights into potential diseases present in the patient, pinpointing their locations within the chest. This functionality not only saves valuable time for radiologists but also enhances diagnostic accuracy, ultimately improving patient care and treatment outcomes in medical settings.

1.1. Motivation and Justification

The rising prevalence of chest diseases significantly increases the burden on radiologists, who are often confronted with extensive queues of X-ray reports awaiting diagnosis. Efficiently managing and accurately reporting these images is paramount to maintaining the quality of patient care. However, the challenge is compounded by the presence of noise and artifacts in X-ray images, frequently caused by defects in the X-ray devices themselves. These imperfections can obscure critical details, making it more difficult for radiologists to identify and diagnose conditions accurately. Consequently, there is a pressing need for advanced tools that can enhance the clarity of these images and streamline the reporting process. Such tools would not only alleviate the workload on radiologists but also improve diagnostic accuracy, ensuring that patients receive timely and appropriate treatment. This dual focus on efficiency and accuracy underscores the importance of innovative solutions in the field of radiology, ultimately justifying the development and integration of specialized technologies like X-Reporto.

1.2. Project Objectives and Problem Definition

The primary objective of this project is to implement a semi-automated reporting system that substantially supports radiologists in generating preliminary reports for each anatomical region and identifying diseases in those regions. This system aims to enhance chest X-ray images by reducing noise and artifacts caused by defects in X-ray devices, ensuring that critical diagnostic details are preserved and easily identifiable. By improving the quality and efficiency of X-ray image reporting, this project seeks to save both time and money, expediting the overall medical process. This acceleration is crucial as it enables quicker diagnosis and treatment, ultimately

increasing patient survival rates. The semi-automated system not only alleviates the workload of radiologists but also ensures a higher degree of accuracy in diagnosing chest diseases, thus improving patient outcomes and optimizing healthcare resources.

1.3. Project Outcomes

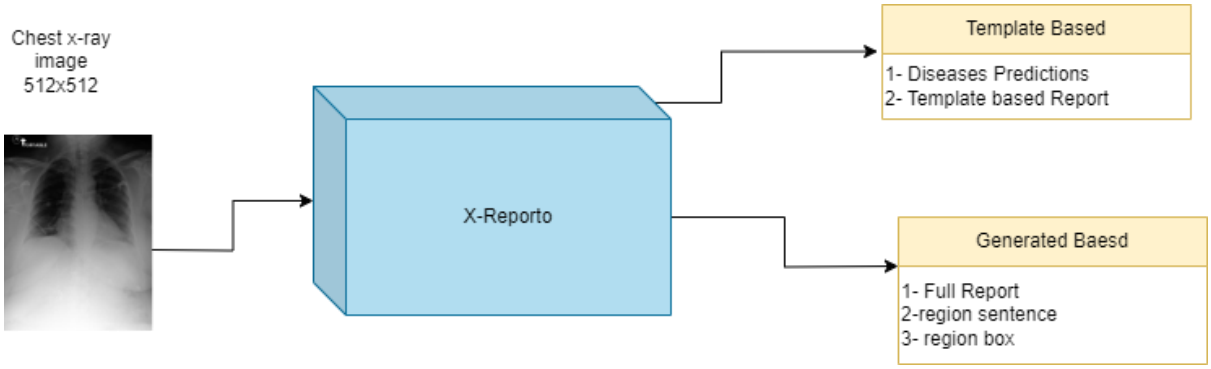


Figure1 X-Reporto Outcomes

The outcomes of our project are centered around the development of a tool that significantly improves the efficiency and accuracy of chest X-ray diagnostics. This tool generates comprehensive reports on chest X-ray images, detailing findings for each anatomical region and identifying specific diseases. It also enhances the quality of X-ray images by correcting defects caused by the X-ray device, ensuring that critical diagnostic information is clear and accurate. Additionally, the tool provides radiologists with template-based reports that are tailored to the specific diseases identified, streamlining the reporting process and reducing the time required for diagnosis. These improvements not only support radiologists in managing their workload more effectively but also contribute to better patient care by enabling faster and more precise medical interventions.

1.4. Document Organization

This report is organized into the following chapters:

1. **Introduction:**
   This chapter summarizes the report, giving a brief summary of the problem addressed by our study, project objectives, and problem definition.

2. **Market Visibility Study:**
   In this Chapter we present an overview on the available tools related to the x-ray reporting and we state the pros and cons of each tool

3. **Literature Survey:**
   In this chapter we explain all the technical required knowledge for the understanding of our project

4. **Dataset Literature Survey:**
   In this chapter show our data survey about the available chest dataset we searched.

5. **System Design and Architecture:**
   This chapter describes the design and architecture of X-Reporto, including the approach taken to develop the tool, its key features, and the technologies used in its development.

6. **System Testing and Verification:**
   This chapter presents the results of the testing and verification process for X-Reporto, including its performance, accuracy, and efficiency in producing medical reports and diagnosis.

7. **Conclusions and Future Work:**
   This chapter outlines X-Reporto's significant results and concludes the importance of our work in enhancing medical reporting. It also identifies expected future work and development areas.

8. **References:**
   This chapter lists all the references cited in the report.

Chapter 2: Literature Survey

2.1. CNN:

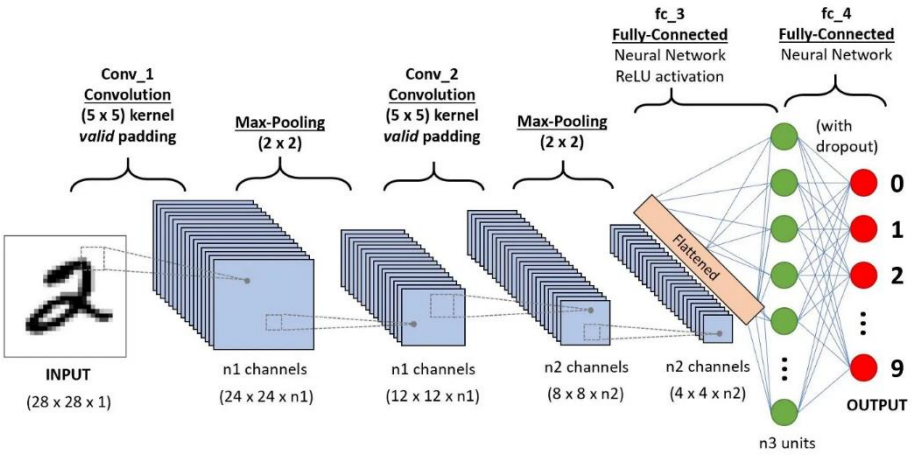


Figure 2 Conv Neural Networks

O'shea et all [1] Proposed the CNN network which is a network architecture for deep learning which learns directly from data. CNNs are particularly useful for finding patterns in images to recognize objects. They can also be quite effective for classifying non-image data such as audio, time series, and signal data.
Layers used to build CNN: Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs.

They have three main types of layers, which are:

i. Convolutional Layer
ii. Pooling Layer
iii. Fully Connected (FC) Layer

Convolutional Layer: This layer is the first layer that is used to extract the various features from the input images. In this layer, we use a filter or Kernel method to extract features from the input image.

$$\frac{W - F + 2P}{S} + 1$$

Pooling Layer: The primary aim of this layer is to decrease the size of the convolved feature map to reduce computational costs. This is performed by decreasing the connections between layers and independently operating on each feature map. Depending upon the method used, there are several types of Pooling operations. We have Max pooling and average pooling.

Fully connected layer: The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of CNN Architecture.

Dropout: Another typical characteristic of CNNs is a Dropout layer. The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others**.**

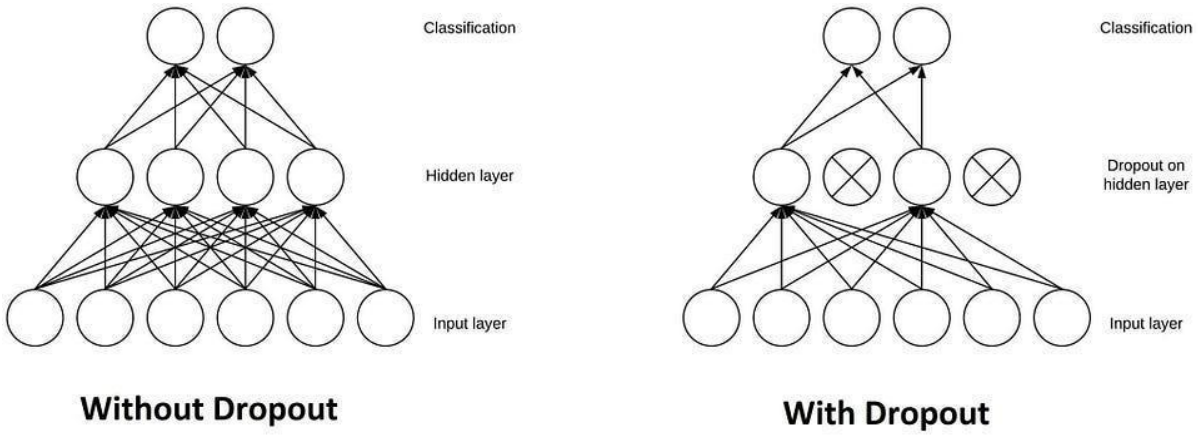


Figure 3 Dropout Technique in NN

Activation Function: It decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction. There are several commonly used activation functions such as the Relu, SoftMax, tanh, and the Sigmoid functions. Each of these functions has a specific usage.

Sigmoid: For a binary classification in the CNN model

Tanh: The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values, in this case, is from -1 to1.

SoftMax: It is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a

probability distribution over predicted output class. Elu- the main advantage of using the Relu function over other activation functions is that it does not activate all the neurons at the same time.

2.2. VGG-19:

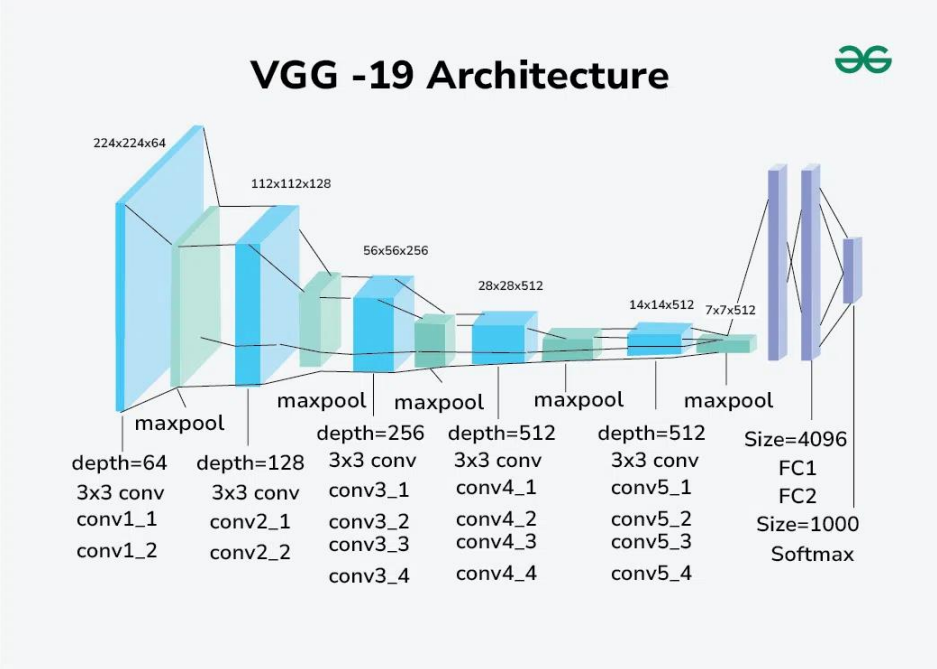


Figure 4 VGG Network Architecture

Simonyan, Karen et al [2] Proposed the basic idea of the VGG16 model, with the exception that it supports 19 layers. The numbers “16” and “19” refer to the model’s weight layers (convolutional layers). In comparison to VGG16, VGG19 contains three extra convolutional layers. In the final section of this essay, we’ll go into greater detail on the features of the VGG16 and VGG19 networks.
Very tiny convolutional filters are used in the construction of the VGG network. Thirteen convolutional layers and three fully connected layers make up the VGG-16.

2.3. ResNet50

ResNet50 (Residual Network with 50 layers) as shown in figure 3.7.1 is a deep convolutional neural network (CNN) architecture that is part of the ResNet family. ResNet architectures are known for their innovative use of residual connections, which help mitigate the problem of vanishing gradients in very deep networks.At the reset of this section we will explain the building block of this network.

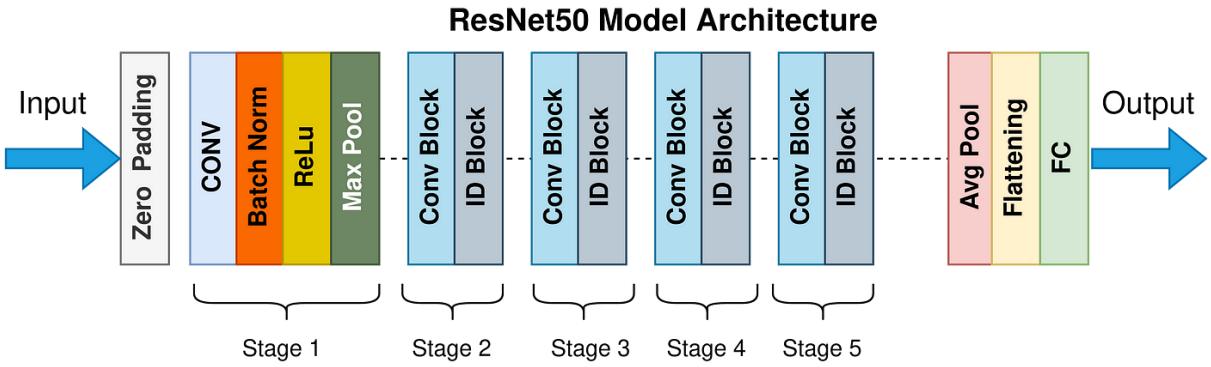


Figure 5 ResNet50 Architecture

2.3.1. Residual Blocks

Network Depth: ResNet50 consists of 50 convolutional layers, making it a deep network. The layers include convolutional layers, batch normalization layers, ReLU activation functions, and pooling layers, organized into a series of residual blocks.The Network has several types of layers listed below

1. **50 Layers:** ResNet50 consists of 50 convolutional layers, making it a deep network. The layers include convolutional layers, batch normalization layers, ReLU activation functions, and pooling layers, organized into a series of residual blocks.
2. **Conv1:** The first layer is a convolutional layer with 64 filters, a kernel size of 7x7, a stride of 2, followed by batch normalization and a ReLU activation. This is followed by a max pooling layer.
3. **Conv2_x to Conv5_x:** The main body of ResNet50 consists of four stages of convolutional layers, each with multiple residual blocks. The number of filters increases as the network goes deeper:
    - Conv2_x: 3 residual blocks with 64 filters.
    - Conv3_x: 4 residual blocks with 128 filters.
    - Conv4_x: 6 residual blocks with 256 filters.
    - Conv5_x: 3 residual blocks with 512 filters.
4. **Fully Connected Layer:** The final part of the network is a fully connected layer with a softmax activation function for classification.

2.4. Faster RCNN

Faster R-CNN [1] (Faster Region-based Convolutional Neural Network) is an advanced object detection framework designed to efficiently and accurately detect objects within

an image. It builds upon the success of its predecessors, R-CNN and Fast R-CNN, by introducing several innovations that significantly improve both speed and performance.The Faster RCNN consists mainly of 4 blocks explained below in detail.

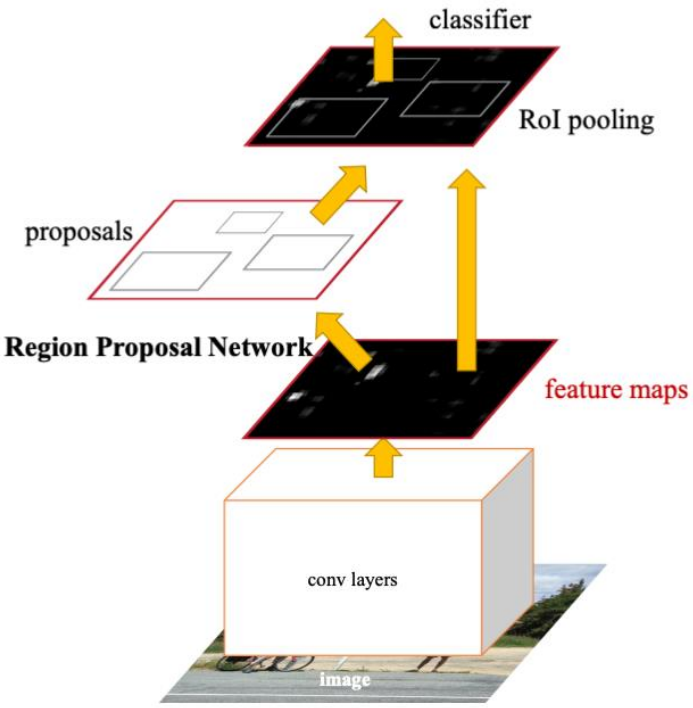


Figure 6 Faster RCNN Architecture

:

2.4.1. CNN (Backbone)

A CNN (e.g., ResNet, VGG) is used to extract feature maps from the input image. This CNN backbone processes the image through multiple convolutional and pooling layers to generate rich, hierarchical feature representations.

2.4.2. RPN

Region Proposals Network: it is the enhancement in RCNN that proposes candidate regions (regions of interest or RoIs) in the feature map that are likely to contain objects. It slides a small network over the feature map output by the CNN backbone.

Anchor Boxes: The RPN generates multiple region proposals using a set of predefined anchor boxes of different sizes and aspect ratios.
Bounding Box Regression and Classification: For each anchor box, the RPN predicts an objectness score (i.e., whether it contains an object) and adjusts the coordinates to refine the region proposals.

2.4.3. ROI

Fixed size feature maps : the RoI pooling layer takes the proposed regions from the RPN and extracts fixed-size feature maps for each proposal, regardless of their size and shape, by using a pooling operation.

2.4.4. Classification and Bounding Box Regression

Final prediction: the extracted features for each RoI are fed into fully connected layers to perform classification (assigning an object class) and bounding box regression (refining the bounding box coordinates)

Chapter 3: Datasets Literature Survey

In this chapter, we provide an overview of the available datasets in the field of X-ray. The chapter is organized into three main sections. In Section 1, we conduct a survey of the available chest X-ray datasets, offering brief descriptions of their usage, metadata, and availability. Section 2 focuses on the primary dataset chosen for our project, detailing the process of obtaining access and explaining its significance. Finally, in Section 3, we describe our data preprocessing steps, including data cleaning and preparation, to facilitate effective learning from the dataset.

3.1. Survey of Available Chest X-ray Datasets

In this section we provide an overview of the available chest x-ray datasets

3.1.1. Chest Diseases [2]

ChestX-Det is a chest X-Ray dataset with instance-level annotations (boxes and masks). It is a subset of the public dataset NIH ChestX-ray14. It contains ~3500 images of 13 common disease categories labeled by three board-certified radiologists.The problem we faced with this data set is that its size is small.

[2] https://paperswithcode.com/dataset/chestx-det

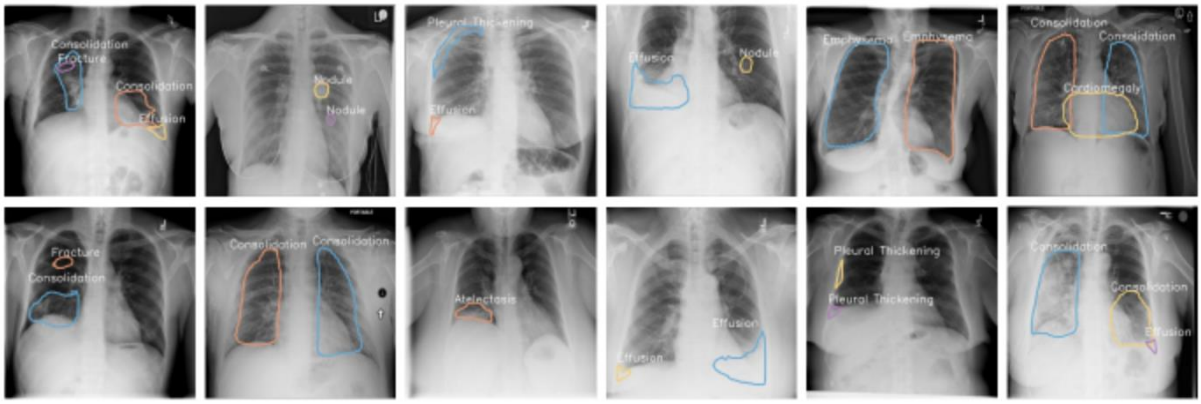


Figure 7 Chest Diseases Dataset examples

3.1.2. PADCHEST [3]

Large-scale, high-resolution labeled data set of chest radiographs for automated medical image exploration along with their associated reports. This dataset includes more than 160,000 images of 67,000 patients that were interpreted and reported by radiologists at Hospital San Juan (Spain) from 2009 to 2017, covering six different position views and additional information on image acquisition and demographics. of the patient.The problem we faced with this dataset is its size is about 1TB so we couldn't download it.

[3] https://bimcv.cipf.es/bimcv-projects/padchest/

Figure 8 PADCHEST Dataset examples

3.1.3. VinDr-CXR [4]

Large dataset of chest x-ray (CXR) images with high-quality labels for the research community, Built from more than 100,000 raw images in DICOM format that were retrospectively collected from the Hospital 108 and the Hanoi Medical University Hospital, two of the largest hospitals in Vietnam. The published dataset consists of 18,000 postero-anterior (PA) view CXR scans that come with both the localization of critical findings and the classification of common thoracic diseases. These images were annotated by a group of 17 radiologists with at least 8 years of experience for the presence of 22 critical findings (local labels) and 6 diagnoses (global labels); each finding is localized with a bounding box. The local and global labels correspond to the “Findings” and “Impressions” sections, respectively, of a standard radiology report. The problem we faced with this dataset is that it is a DICOM format so the size is huge and saved on cloud and we don’t need the DICOM files

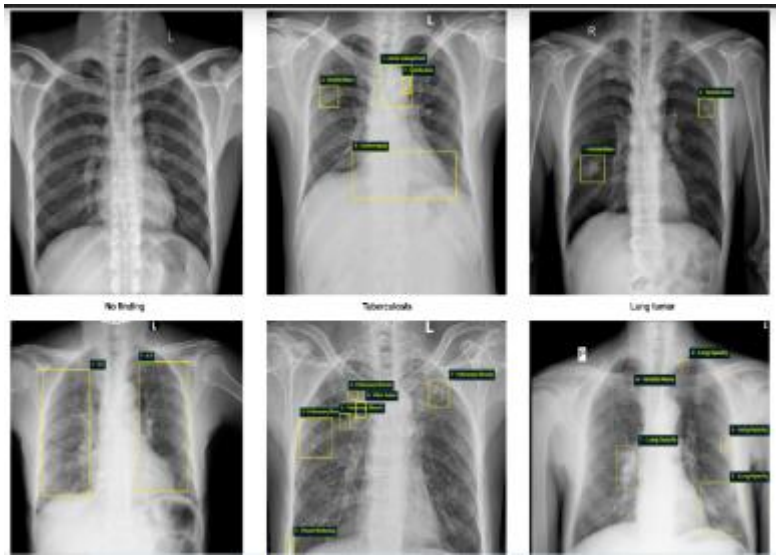


Figure 9 VinDr-CXR Dataset examples

3.1.4. MIMIC-CXR [5]

A set of 10 folders (p10 - p19), each with ~6,500 sub-folders. Sub-folders are named according to the patient identifier, and contain free-text reports and DICOM files for all studies for that patient

[4] https://physionet.org/content/vindr-cxr/1.0.0/
[5] - https://physionet.org/content/mimic-cxr/2.0.0/

cxr-record-list.csv.gz - a compressed file providing the link between an image, its corresponding study identifier, and its corresponding patient identifier
cxr-study-list.csv.gz - a compressed file providing a link between anonymous study and patient identifiers
mimic-cxr-reports.tar.gz - for convenience, all free-text reports have been compressed in a single archive file
The problem we faced in such dataset is that the data size is around 6 TB and cant download it as it contain DICOM files we didn't need DICOM we just need images as JPG format and the report

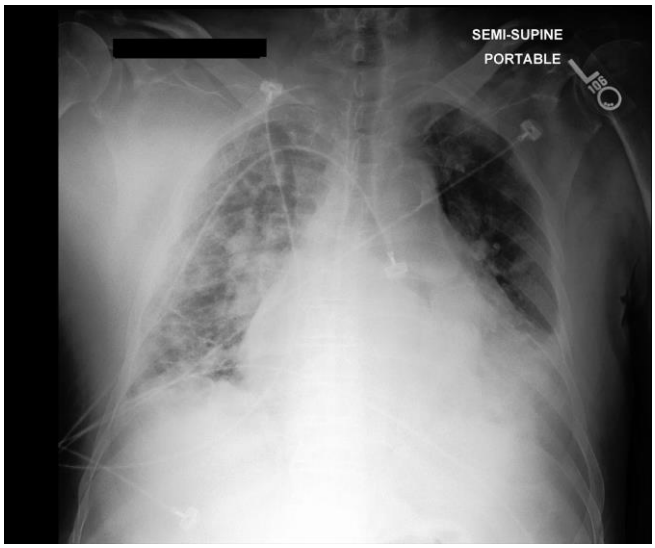


Figure 10 MIMIC-CXR Dataset examples

3.2. Selection and Access of Primary Dataset [6]

We chose the MIMIC-CXR-JPG dataset, which is a version of the MIMIC-CXR dataset containing PNG images instead of DICOM format images.

3.2.1 Reasons for Selection

Image Format: We selected this dataset because it provides chest images in PNG format, which is easier to work with for many machine learning applications compared to DICOM.
Available Reports: We found corresponding radiology reports available in the mimic-cxr-2.0.0-chexpert.csv file, making it easier to link images with textual data for training and evaluation purposes.
Accessibility: We obtained access to the dataset and used it for training our models.

[6] https://physionet.org/content/mimic-cxr-jpg/2.1.0/

3.2.2 Meta Description of the the dataset

Data Size: The dataset is enormous, approximately 500GB, containing 377,110 JPG format images and structured labels derived from the 227,827 free-text radiology reports associated with these images.

Subset Download: We downloaded a subset of the dataset and uploaded it to Google Drive for easier access and handling.
Labeling: The data is labeled using CheXpert, an open-source rule-based tool, ensuring consistency and reliability in the labeling process.
Diseases: The reports indicate the presence of the following diseases: Atelectasis, Cardiomegaly, Consolidation, Edema, Enlarged Cardiomediastinum, Fracture, Lung Lesion, Lung Opacity, Pleural Effusion, Pneumonia, Pneumothorax, Pleural Other, Support Devices, and No Finding.

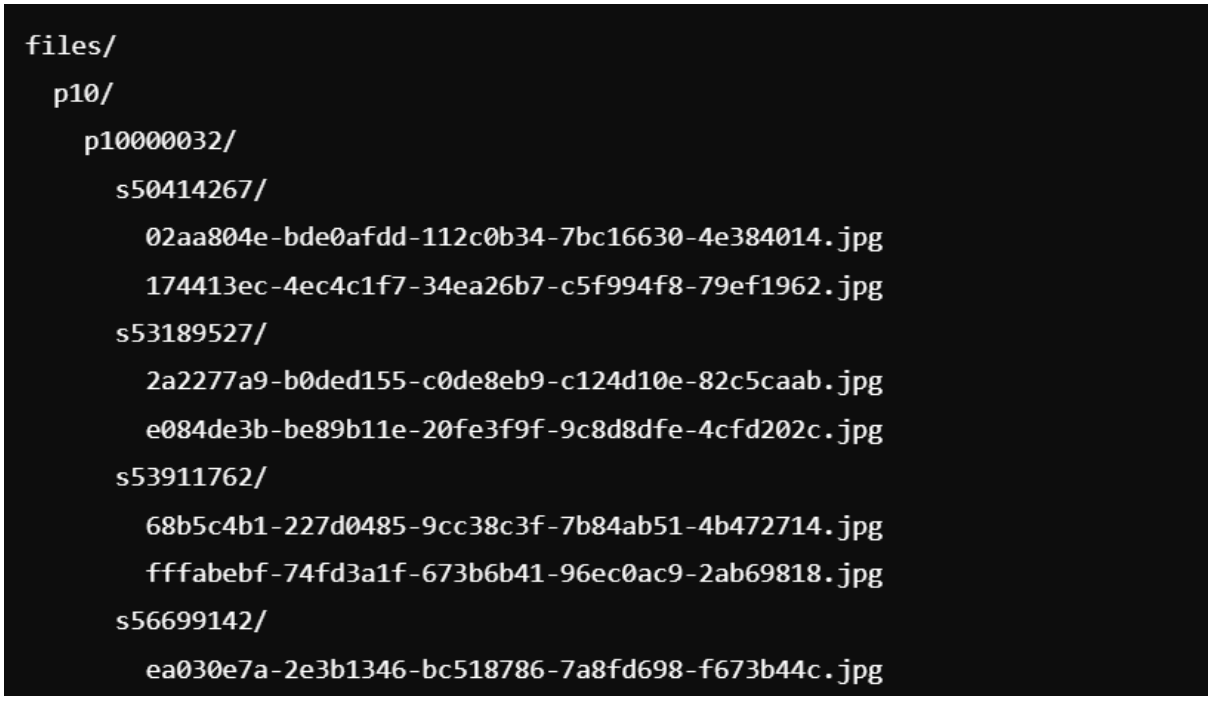


Figure 11 MIMIC-CXR Folder Structure

Chapter 4: System Design and Architecture

In this chapter, we provide a comprehensive overview of the design and architecture of the project, detailing the steps taken to develop a robust system for generating medical reports from X-ray images. The project is structured around a series of interconnected modules, each contributing to the overall functionality. By employing a systematic approach, we designed the system to ensure efficient processing, cleaning image, detecting anatomical regions, accurate outputs of medical reports, and efficient diagnosis . This chapter will discuss the methodologies employed, from the initial design concepts to the implementation of each module, allowing readers to understand and replicate our work effectively.

4.1. Overview and Assumptions

In designing the system, we aimed to create an integrated workflow that facilitates the transformation of raw medical images into detailed reports. The architecture is built on five primary modules: image denoising, object detection, selection binary classifiers, language model generation for medical reporting, and a heatmap module for disease classification. Each module operates cohesively, with defined inputs and outputs, ensuring smooth data flow throughout the system. Key assumptions include the availability of high-quality medical images, the capability of the object detector to accurately identify anatomical features, and the reliability of the language model in generating coherent text. These assumptions underpin our design choices and guide the development process.

4.2. System Architecture

The architecture of the system is structured around a modular design that enhances scalability and maintainability. The overall system is represented as a block diagram that shows interactions between the different modules and their respective functions.

As we have 6 separated modules, each module has specific functionality and expects certain modules output and generates correct output for next modules. Follow of our modules is that

1. We first should remove any noise in the x-ray chest.
2. We detect anatomical regions in chest and generate visual features for them
3. We Select some regions that have findings in it.
4. From these visual features we generate corresponding findings in it including abnormality and diagnosis.
5. Finally, we generate predictions for possible diseases and its location in x-ray.

4.2.1. Block Diagram

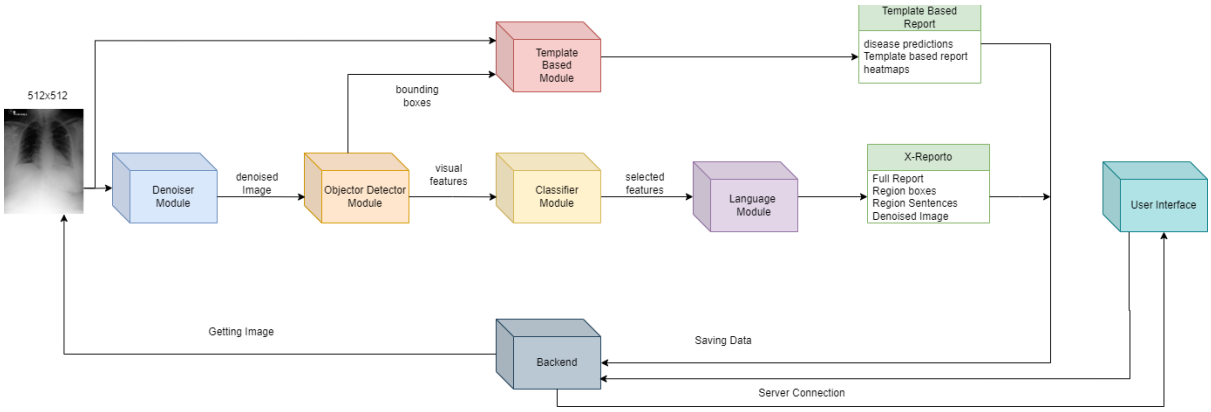


Figure 12 X-Reporto System Block Diagram

4.3. Denoiser Machine Learning Approach

4.3.1. Functional Description

X-ray image medical information must be preserved for getting a report on it correctly.X-ray devices may have defects which affect medical information. Denoiser takes an image to enhance it and remove noise without effect on medical information of the image.

4.3.2. Modular Decomposition

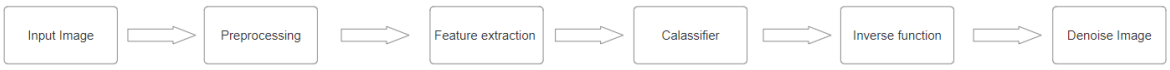


Figure 13 Denoiser Classical Approach Block Diagram

4.3.2.1. Input Image

We have many noises type:

- Block pixel
- Convolution
- Keep patch
- Extract patch
- Pad rotate
- Random noise
- Line strip
- Salt and pepper
- no noise

We will mainly concentrate on (black pixel, convolution, random noise, no noise) as they have high priority to occur from the X-ray device

4.3.2.2. Feature Extraction

We used several methods to extract features from noised image

- **All image (flatten the image as feature):** it works but not the best accuracy
- **HOG:** it is the best to extract noise features and we try different parameters
- **Fourier transform:** it is the best to extract salt and pepper noise
- **Mixed feature:** add HOG and fourier transform features together

4.3.2.3. Classifier

We used different classifier to detect the noise type in the image then send it to the correct inverse function to denoise it

- **SVM:** kernel is (rb), C is (1) and gamma is (0.1)
- **Random forest (the best):** n_istemators is (100)
- **Adaboost Classifier:** we use random forest to train it and the parameters are, class weight is (balanced) as each image has all noises types so each noise type has equal images in train data , number of estimators is (11), max features is (sqrt), min samples leaf is (2) and min sample split (2)

4.3.2.4. Inverse Function

After detecting the noise type we try to apply the inverse function of generating the noise so for each type we make inverse for it

- **Block pixel** ⇒ we apply median blur to remove the black pixels from the image
- **Keep patch** ⇒ we apply interpolation
- **Extract patch** ⇒ we apply interpolation
- **Pad Rotate** ⇒ sadly it can't solve as we need to know the pad rotation angle to reverse the operation but we don't know it but lucky the pad rotation angle in range [-2,2] so the doctor can diagnose the image correctly and the object detector handel this part
- **Line strip** ⇒ we apply interpolation
- **Salt and pepper** ⇒ we apply median blur
- **Random noise** ⇒ we apply Non-Local Means Denoising
- **Convolution** ⇒ try to detect best gaussian kernel then apply convolution finally we apply Non-Local Means Denoising

4.3.4. Methodologies

4.3.4.1. Data Preprocessing

The input image is X-ray image grayscale in rectangular shape, We performed the same types of processing for both training and testing Modes explained below

- Resize the image to in shape (512,512) but resize it on the longest dimension as the input image is rectangular.
- Add padding to the image as the object detector expected

Add all noises for each image e.g (we have 150 image and 3 types of noises after this step we will have 450 image each image with all types independent)

4.4. Object Detection Deep Learning Approach

4.4.1. Functional Description

Object detector to detect 29 regions in the chest and extract the features from them to send to the Binary classifiers .The 29 regions are (right lung, right upper lung zone, right mid lung zone, right lower lung zone, right hilar structures, right apical zone, right costophrenic angle, right hemidiaphragm, left lung, left upper lung zone, left mid lung zone, left lower lung zone, left hilar structures, left apical zone, left costophrenic angle, left hemidiaphragm, trachea, spine, right clavicle, left clavicle, aortic arch, mediastinum, upper mediastinum, svc, cardiac silhouette, cavoatrial junction, right atrium, carina, abdomen.It will always return 29 region and their features and classifiers will filter them .The object detectors return 30 region 29 + background which is 0 and we not interested in this

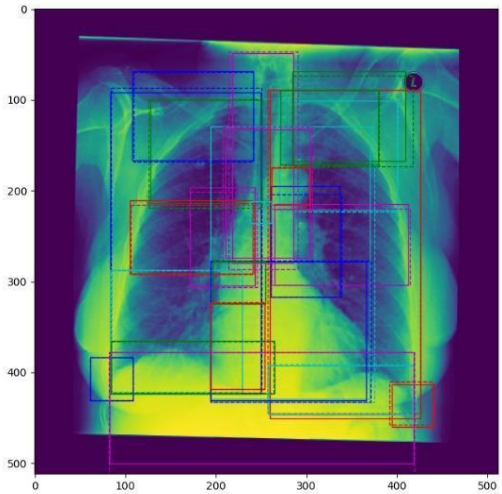


Figure 14 Example for the target 29 regions to detect

4.4.2. Modular Decomposition

The following sections are organized as follows: Input Image, Backbone, RPN, ROI, and Training.The input image is X-ray image grayscale in rectangular shape.

4.4.2.1. Backbone

We used resnet50 as it is the feature extractor and all the pipeline depends on this part

4.4.2.2. Anchor Generator

We customize the anchor generatore to generate anchor boxes in the size as we need for chest and customize its hyperparameter to sizes=((20, 40, 60, 80, 100, 120, 140, 160, 180, 300),) and the aspect_ratios=((0.2, 0.25, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.3, 1.5, 2.1, 2.6, 3.0, 5.0, 8.0),)

4.4.2.3. RPN

We customize the RPN to return the losses in case of evaluation and give targets as input

4.4.2.4. ROI

We customize the ROI to return the best bounding box for each region (the normal FasterRCNN can return many bounding boxes for the same region or didn't return any bounding box for one region which isn't suitable for our criteria we need to guarantee that 29 bounding box return each time and the best one for each region then the binary classifier will remove the unwanted or wrong boxes)
We add 2 layers on AvgPool2d do we average over the whole feature maps and the second layer for reduce the dimension from 2048 to 1024

4.4.3. Design Constraints

We need to return feature of each bounding box to send it to the classifier and GPT-2 so if we use any other Object detector we can't get the feature from it even there are efficient object detection and very good performance in time like YOLO but we can't get features from it
We need to get very high accuracy from this module as all other modules depends on this module if it detect wrong region or bad features the rest of the pipeline will be useless (garbage in garbage out) so we recommend use deep learning approach here
We need to return feature of each bounding box to send it to the classifier and GPT-2 so if we use any other Object detector we can't get the feature from it even there are

efficient object detection and very good performance in time like YOLO but we can't get features from it
We need to get very high accuracy from this module as all other modules depends on this module if it detects wrong region or bad features the rest of the pipeline will be useless (garbage in garbage out) so we recommend use deep learning approach here

4.4.4. Methodologies

4.4.4.1. Data Preprocessing

The input image is X-ray image grayscale in rectangular shape, We performed the same types of processing for both training and testing Modes explained below

- Resize the image in the longest dimension to (512,512)
- Add padding in the shortest dimension with zeros
- Convert image to tensor
- Normalize the image with mean=0.474 and standard deviation=0.301

As we add augmentation for the data to increase the diversity of the data seen by the module. We perform Random Rotate the image with random value between [-2,2] to train the object detector to handle this error in image and detect the 29 regions

4.4.4.1. Training

First we train the Backbone and the RPN then we freeze the Backbone and RPN and train the ROI finally we train the Backbone, RPN and ROI together
we customize the forward pass to return the feature and losses in the evaluation mode if we send the target
we use Adam as optimizer and the learning rate is 0.001
we use lr scheduler StepLR
we make collate_fn customize and send it to the DataLoader to return target and images as expected for training
we apply cumulative gradient as we make many batches then backpropagate this method help us to use batch size 64 which our PCs can't run by default
Ww used tensorboard to see the resulting image from the training and when to stop training

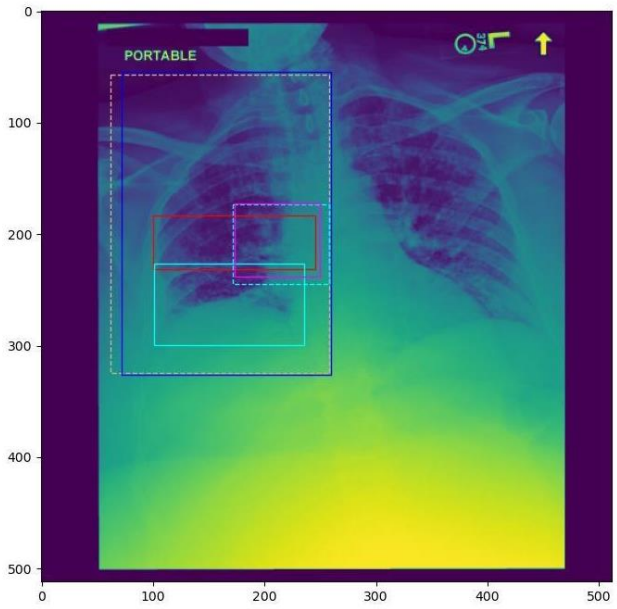


Figure 15 Example for the target 29 regions to detect

Chapter 5: System Testing and Verification

In this chapter, we outline the comprehensive approach taken to ensure that the outcomes of the project are realized accurately and reliably. Testing and verification are critical components of the development process, and we implemented a structured methodology to validate each module and the overall system. By utilizing a combination of unit testing and integrated system testing, we aimed to identify and rectify potential issues at every stage of development. This thorough testing framework not only guarantees the functionality of the system but also enhances confidence in its deployment in real-world applications.

5.1. Testing Setup

The testing setup for the project involves a dedicated environment that mirrors the production configuration to ensure reliable results. The system is deployed on a local server equipped with the necessary hardware and software requirements, including high-performance GPUs for model inference and a robust database management system for data storage. The testing environment utilizes Python and relevant libraries such as TensorFlow and PyTorch for model implementation, along with a REST API framework to facilitate interaction between modules. This controlled setup allows for consistent testing conditions, enabling accurate performance evaluation across different scenarios.

5.2. Testing Plan and Strategy

The testing plan comprises multiple phases, starting with unit testing of individual ai modules followed by integration of a full pipeline of ai modules and finally integrated system testing to assess overall functionality. Each module is evaluated against predefined metrics to ensure it meets the specified performance criteria.

5.2.1. Module Testing

5.2.1.2 Denoiser Machine Learning Approach Testing

We use normal accuracy to calculate the result for each classifier so the result depends on the selected feature extraction model then we use SSIM and PSNR to calculate the accuracy for the denoise imageNotice that this error is accumulating

Features	SVM	Random forest	Adaboost
All image	Accuracy : 0.33 SSIM: 0.97 PSNR: 36.36	Accuracy : 0.44 SSIM: 0.83 PSNR: 30.89	Accuracy : 0.44 SSIM: 0.89 PSNR: 33.41
HOG (pixel per cell 16)	Accuracy : 0.33 SSIM: 0.97 PSNR: 36.41	Accuracy : 1.0 SSIM: 0.93 PSNR: 34.47	Accuracy : 1.0 SSIM: 0.937 PSNR: 34.51
HOG (pixel per cell 8)	Accuracy : 0.33 SSIM: 0.70 PSNR: 26.02	Accuracy : 958 SSIM: 0.93 PSNR: 34.58	Accuracy : 92 SSIM: 0.93 PSNR: 34.45
Fourier transform	Accuracy : 97 SSIM: 0.97 PSNR: 36.38	Accuracy : 1.0 SSIM: 0.93 PSNR: 34.49	Accuracy : 0.98 SSIM: 0.93 PSNR: 34.54
mix feature	Accuracy : 0.33 SSIM: 0.97 PSNR: 36.34	Accuracy : 1.0 SSIM: 0.93 PSNR: 34.48	Accuracy : 0.33 SSIM: 0.93 PSNR: 34.5

Table 1 ML Denoiser Results

Results

Figure 16 Convolution noise (Denoiser ML)

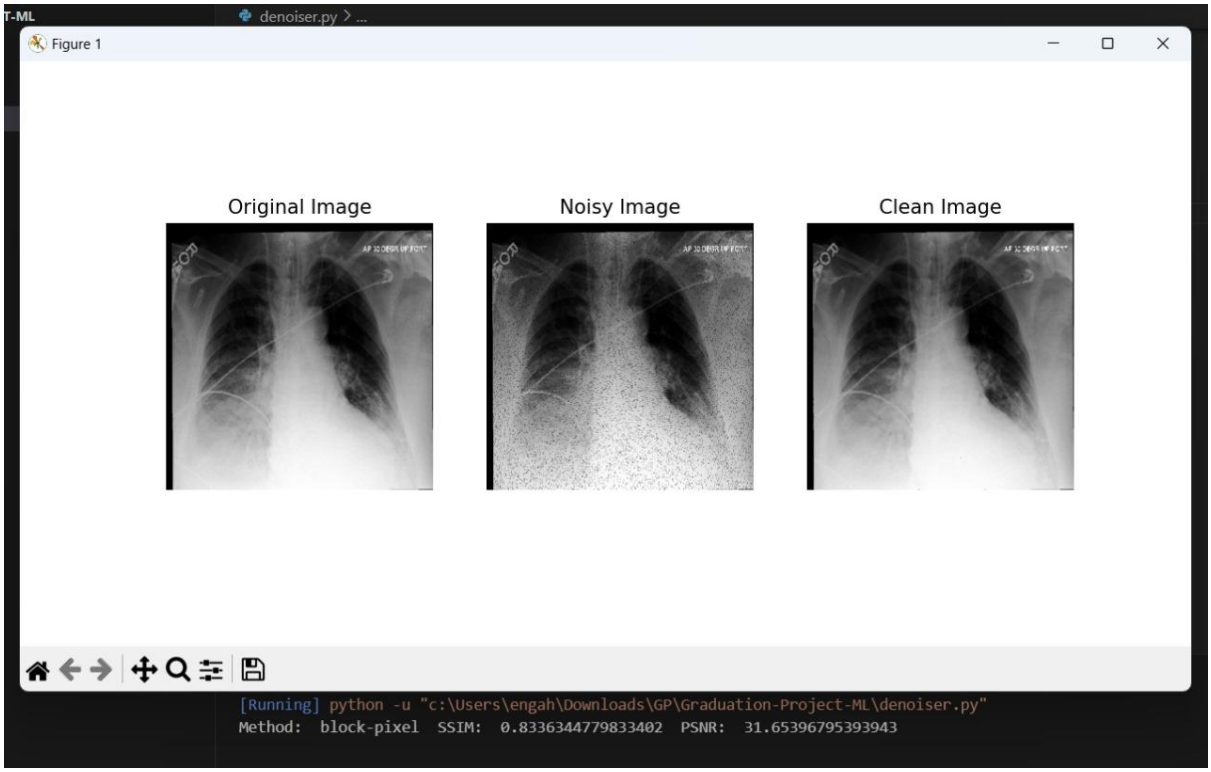


Figure 17 Block pixel (Denoiser ML)

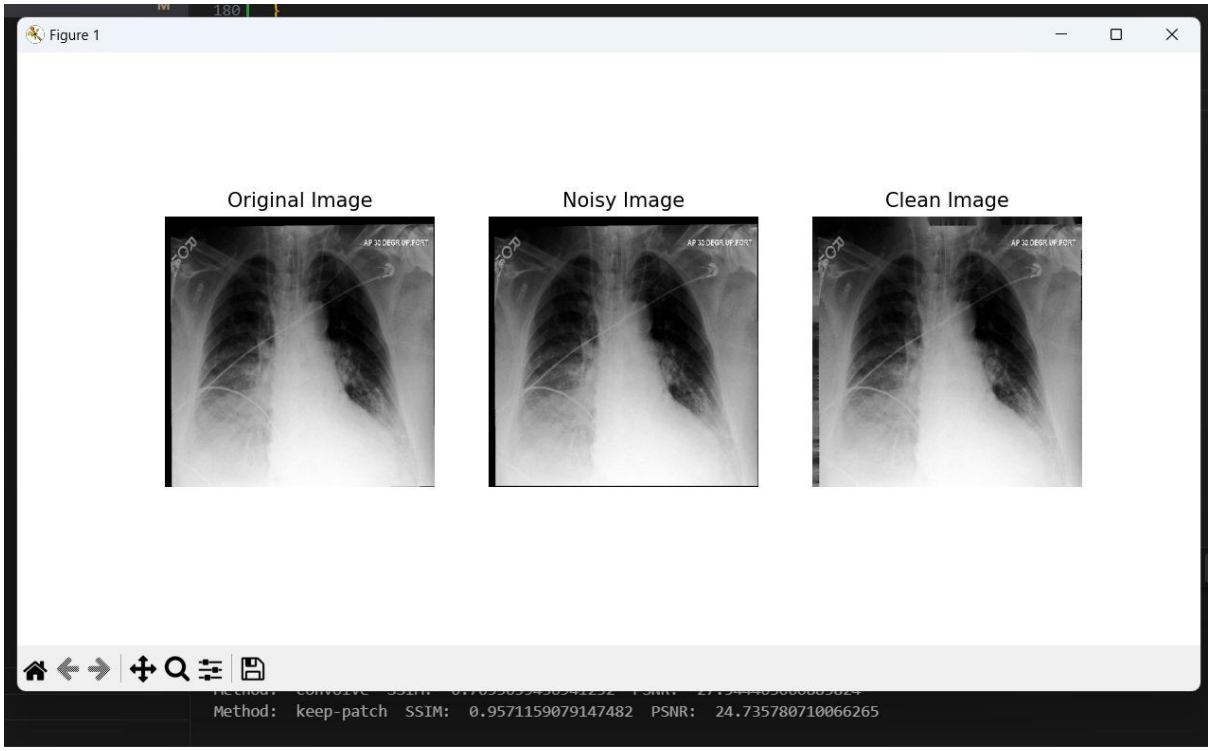


Figure 18 Keep batch (Denoiser ML)

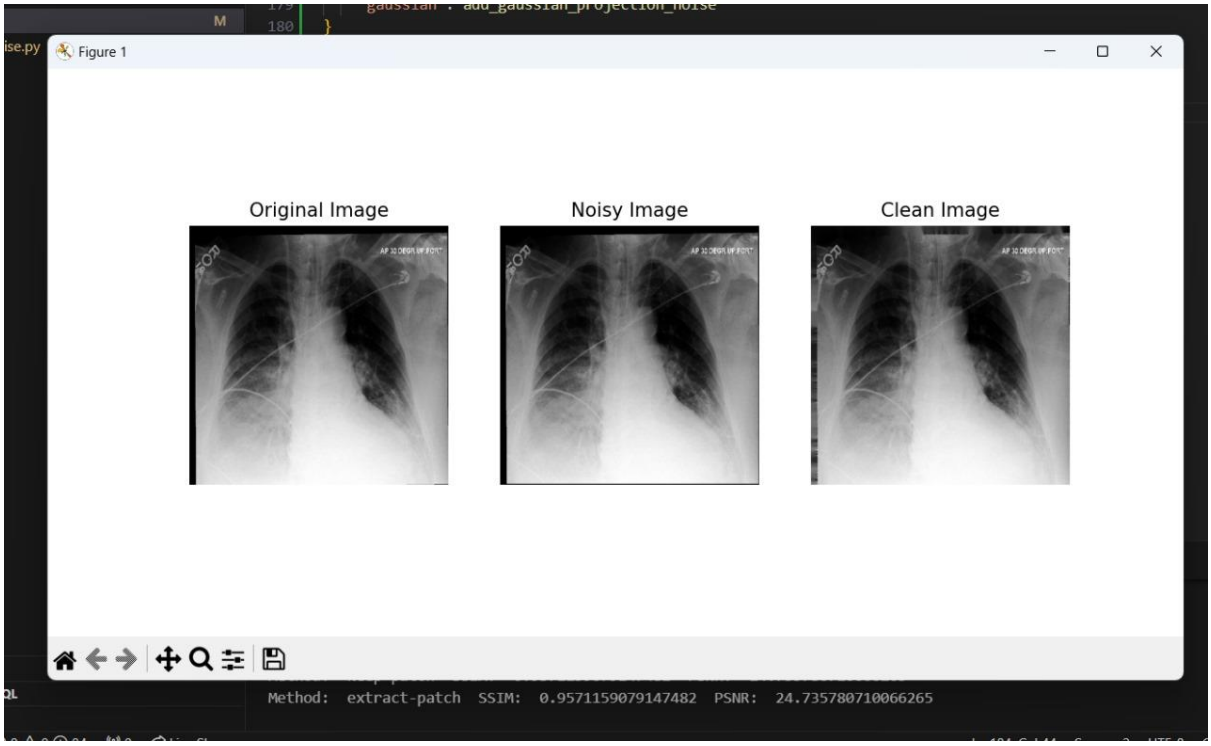


Figure 19 Extract patch (Denoiser ML)

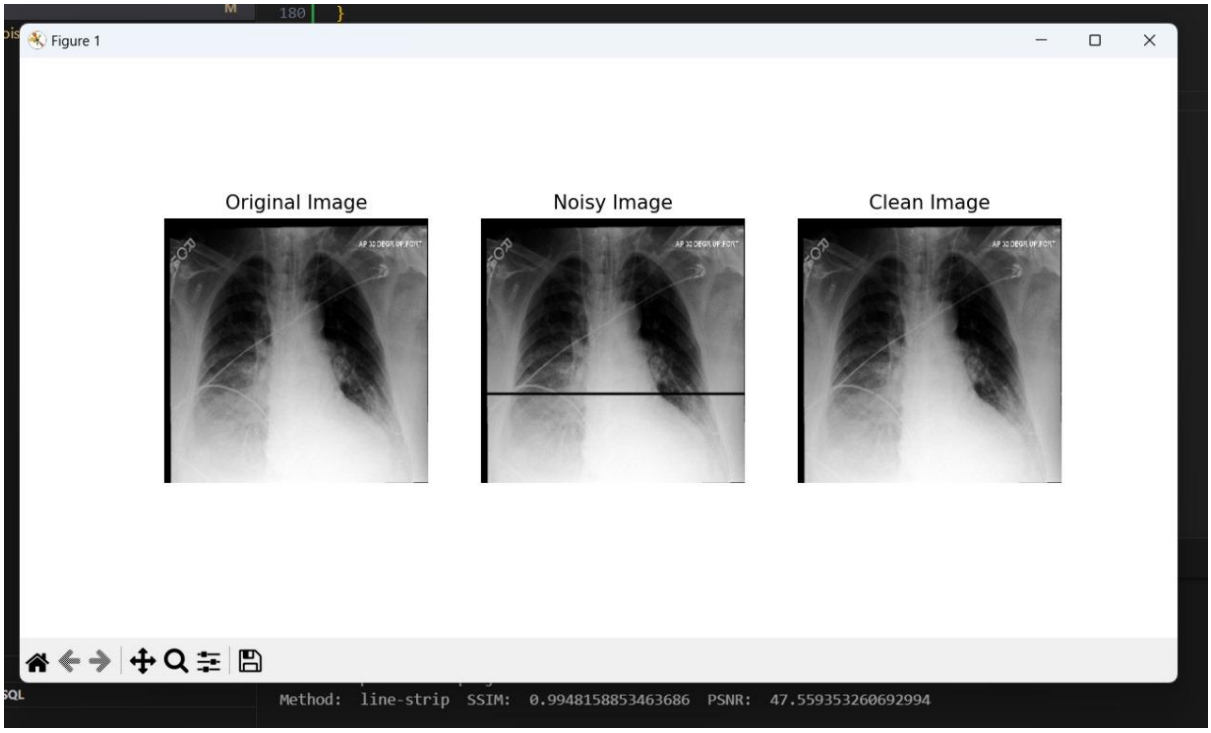


Figure 20 Line strip (Denoiser ML)

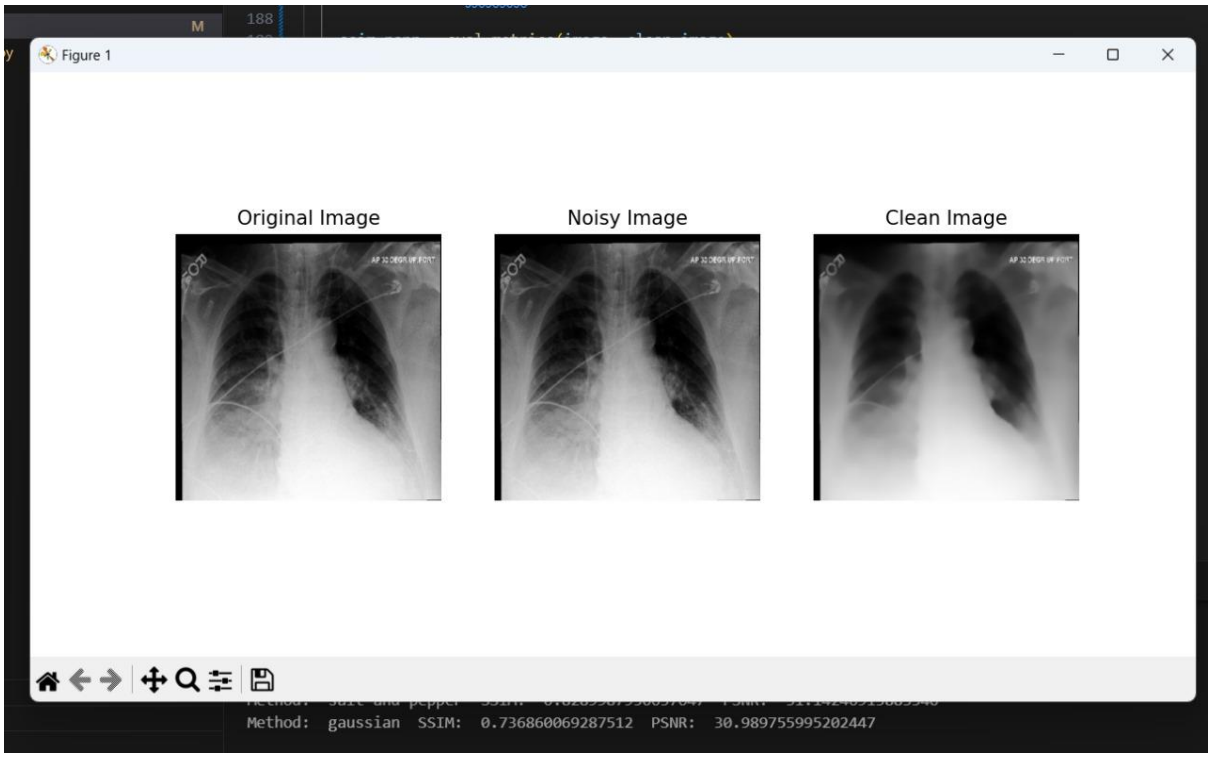


Figure 21 Random noise (Denoiser ML)

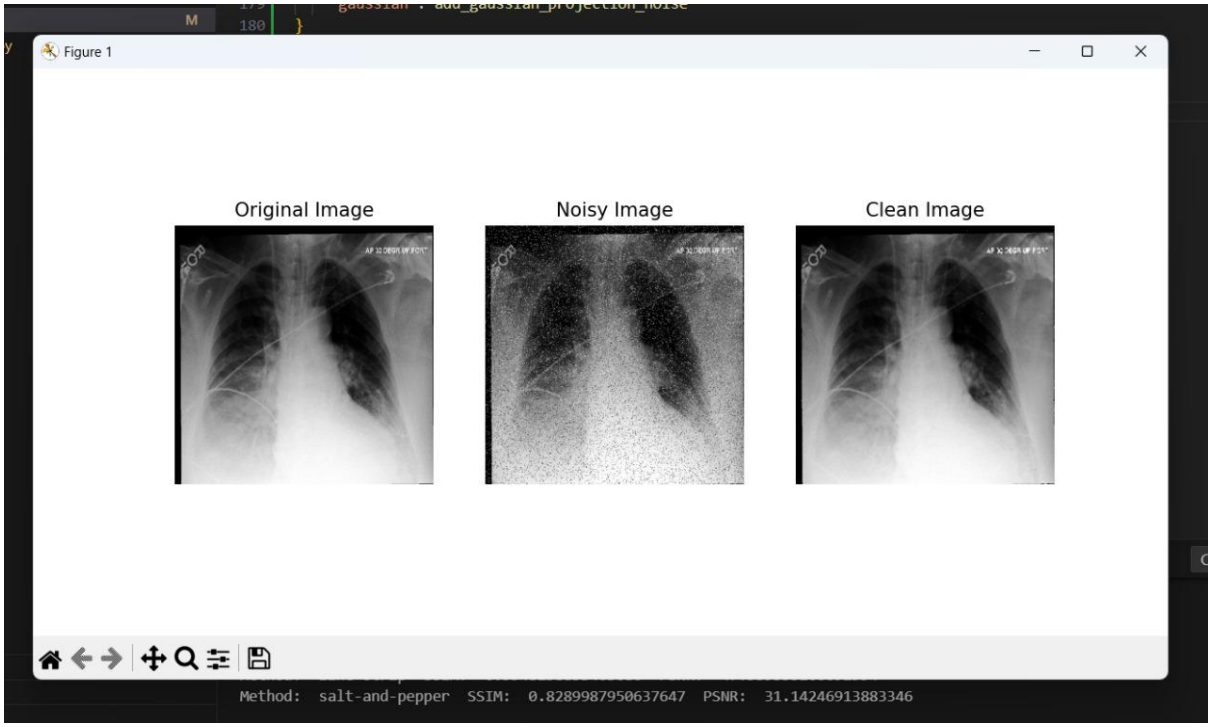


Figure 22 Salt and pepper (Denoiser ML)

5.2.1.3 Object Detector Deep Learning Approach Testing

- We use F1 Score as accuracy metric
- Alos IOU as accuracy metric

Figure 23 False Positive True Positive and False Negative Object Detector DL Results

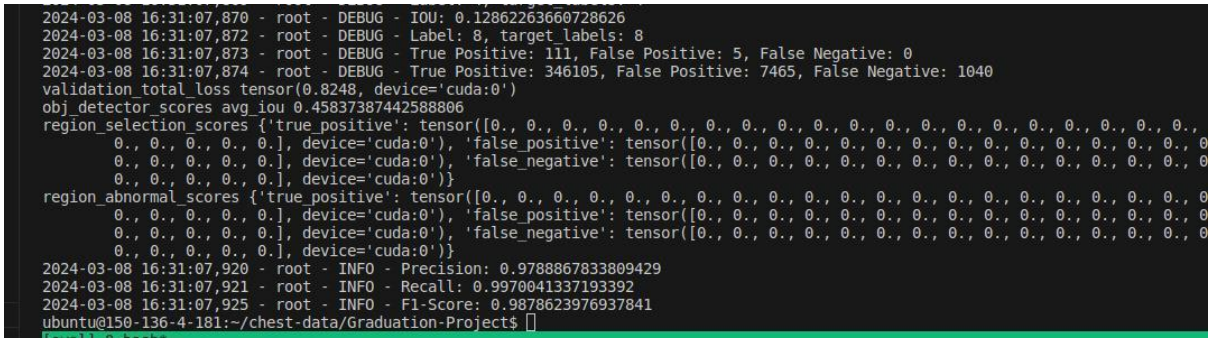


Figure 24 Precision, Recall and F1-Score

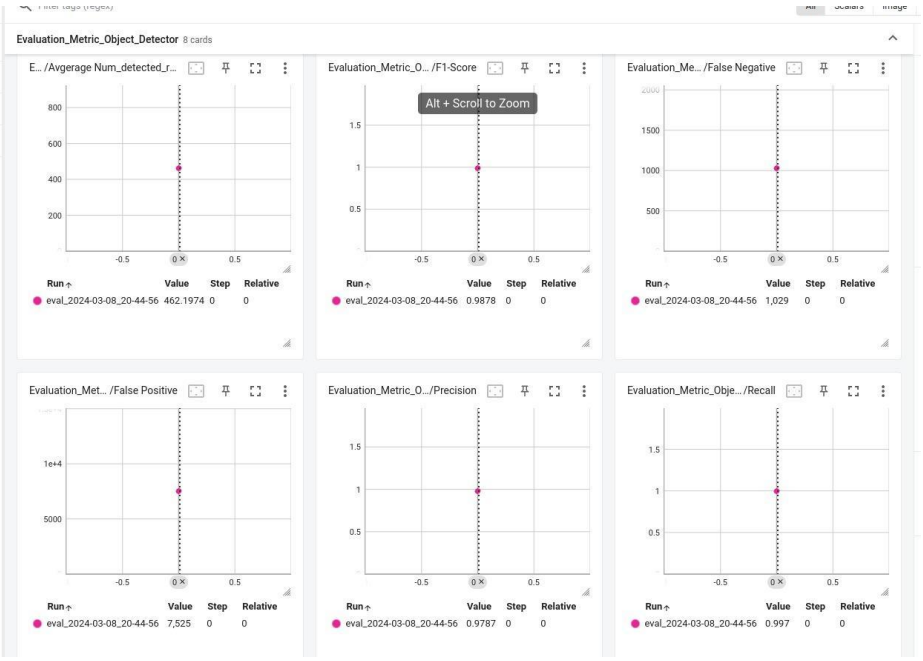


Figure 25 f1 score from tensorboard (object detector DL)

Figure 26 IOU result from tensorboard (object detector DL)

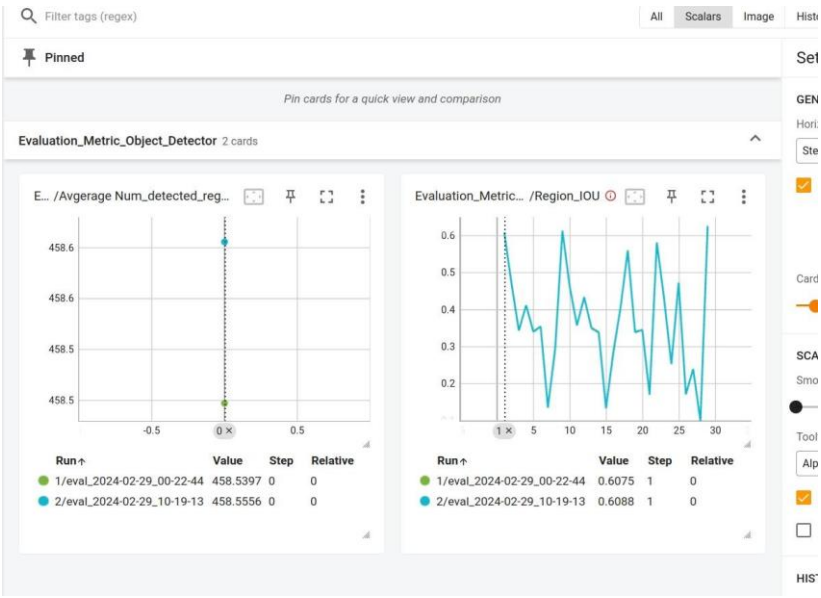


Figure 27 IOU result from tensorboard (object detector DL)

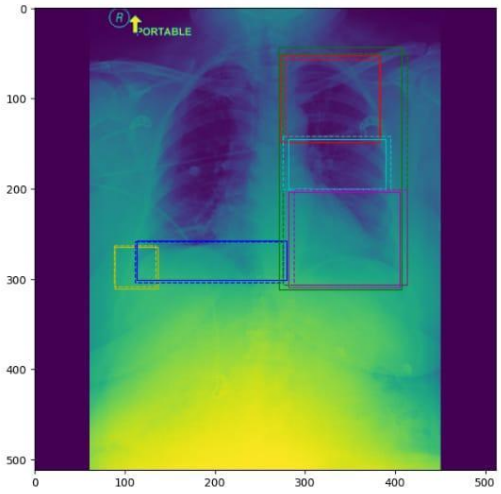


Figure 28 Resulting image from evaluation (object detector DL)

Chapter 6: Conclusions and Future Work

6.1. Faced Challenges

6.1.1 Denoiser Machine Learning

Unfortunately, there are some types of noise can't remove using image processing or classical approach like pad rotate but likely we don't care about it but there is mandatory type which is random noise and gaussian convolution which have no inverse so it can't remove this noise we handle this problem by Non-Local Means Denoising, but it makes the image so blurry which remove medical information
Even if the classifier got high accuracy and the SSIM and PSNR are so high, but it is fake results as the medical information after displaying the denoised image have removed specially in convolution and random noise case
If the classifier classifies the noise type wrong, it will make the noising function wrong so it will generate more noise
if there are more than one type of noise the model can't solve it as denoising function for one type may generate noise with the other type of medical information

6.1.2 Object detection Deep Learning

There are small regions and large regions so it's harder to detect both of them
the training process of the object detector is so hard and to it's hard to detect which part need to detect RPN or ROI or both of them
the network size is huge due to use resnet50 as we need very strong feature extractor but we need large batch size
how to optimize the object detector to use large batch size on local PC

6.2. Gained Experience

6.2.1 Denoiser Machine Learning

- How to deal with noisy image
- Image processing approach to remove noise from image
- Use classifier to detect noises type in the image
- Possible noise type generated in images

6.2.2 Object Detector Deep Learning

- what is ROI and RPN and difference between them
- what is the difference between RCNN, Fast RCNN and Faster RCNN
- how to use Faster RCNN and its implementation
- when to use accumulate gradient

7.3. Conclusions

The project aimed to address the complex and challenging task of automated medical report generation by leveraging state-of-the-art machine learning models and robust backend infrastructure. The project was driven by the need to improve the efficiency, accuracy, and consistency of radiology report generation, which is a critical component in medical diagnostics and patient care.

The system design incorporated several advanced AI models, including image denoising, object detection, selection binary classifiers, and a language model for report generation. Each of these models played a crucial role in processing medical images, extracting relevant features, and generating coherent and detailed medical reports. The integration of a heatmap module and a disease classification component further enhanced the system's ability to focus on abnormal regions and provide precise diagnostic information.

The backend architecture was built using FastAPI, providing a high-performance and scalable framework for handling API requests and facilitating communication between the frontend and AI models. PostgreSQL was chosen as the database management system to efficiently store and manage patient data, study results, and report templates. The use of Docker for containerization ensured consistent deployment and easy maintenance of the application across various environments.

Throughout the development process, rigorous testing and verification were conducted to ensure the system's reliability and accuracy. Unit testing and integrated system testing were employed to validate the functionality of individual modules and the overall system, respectively. Metrics such as accuracy, precision, recall, and F1-score were used to evaluate model performance, while continuous integration practices ensured the robustness of the deployed system.

In summary, this project successfully demonstrated the feasibility and effectiveness of using advanced AI techniques for automated medical report generation. By combining sophisticated models with a well-architected backend and database infrastructure, the system was able to generate detailed and accurate radiology reports. This work not only enhances the efficiency of medical diagnostics but also provides a foundation for future advancements in the field of automated medical reporting.

6.4. Future Work

6.4.1 Denoiser Machine Learning

- Use Multilabel classifier instead of normal classifier to detect if image have more than one type

- solve convolution and random noise in better way
- make loop to make the classifier detect all noises then apply denoising function then apply classifier again and loop in this way until reach maximum number of looping or no finding noise in the image
- try to make function to remove more than one type of noise

6.4.2 Object Detector Deep Learning

References

[1] Girshick R. Fast r-cnn. InProceedings of the IEEE international conference on computer vision 2015 (pp. 1440-1448).

Appendix A: Development Platforms and Tools

A.1. Software Tools

A.1.1. PyTorch

PyTorch is an open-source machine learning library widely used for deep learning applications. It provides a flexible and dynamic computational graph, which is essential for developing complex models efficiently. In this project, PyTorch was utilized for training AI models, including the image denoiser, object detector, and language model. Its extensive support for tensor operations and rich ecosystem of libraries enabled rapid prototyping and experimentation with various model architectures.

A.1.2. Tmux

Tmux is a terminal multiplexer that allows users to manage multiple terminal sessions from a single window. This tool was particularly useful during the training and evaluation phases of the project, enabling seamless monitoring of long-running processes on the development server. By using Tmux, we could run multiple scripts simultaneously and maintain persistent sessions, ensuring efficient resource management and workflow organization.

A.1.3. Figma

Figma is a cloud-based design tool used for UI design and UX studies. In this project, Figma facilitated the creation of user interfaces and visual mockups, allowing for collaborative design efforts among team members. Its real-time collaboration features enabled feedback and iteration on design elements, ensuring that the final interface met user needs and project objectives.

A.1.4. React (TypeScript)

React, combined with TypeScript, is a powerful front-end framework used for building user interfaces. In this project, React provided a robust structure for developing dynamic and interactive web applications. The use of TypeScript added type safety and enhanced code maintainability, facilitating collaboration and reducing the likelihood of runtime errors in the application.

A.1.5. Ant Design (Antd)

Ant Design is a comprehensive design system and UI framework that provides a set of high-quality React components. Antd was used in the project to streamline the development of the user interface, offering pre-built components that adhere to modern design standards. This allowed for rapid development while ensuring a consistent and visually appealing user experience across the application.

Appendix B: Use Cases

B.1. Custom report

Doctor can make his own boxes on the region , insert the region name and the diseases in this region

B.2. AI report

Doctor can ask X-Reporto (our AI model) to diagnose the X-ray and return the ai generated report with the boxes and sentence on each region

B.3. Disease probability & Heatmap

Doctor can ask X-Reporto (our AI model) to generate heatmap to know each disease probability and where each disease can be found in this X-Ray

B.4. Template report

Doctor can ask X-Reporto (our AI model) to generate template report just contain the disease in the X-ray and the probability for each disease in report

Appendix C: User Guide

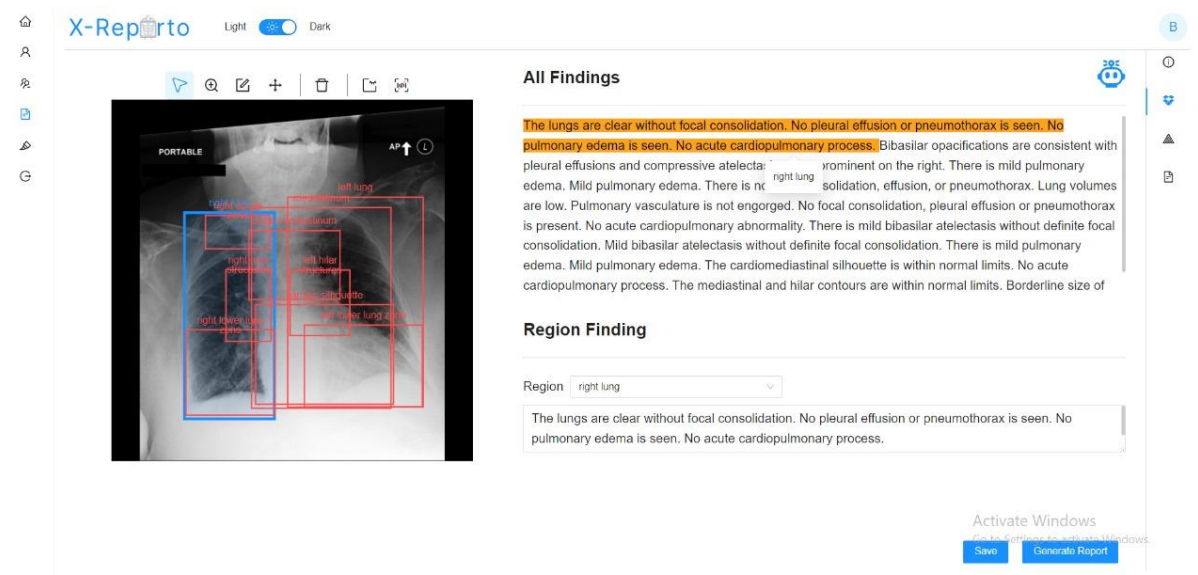


Figure 29 generate AI boxes in abnormal region

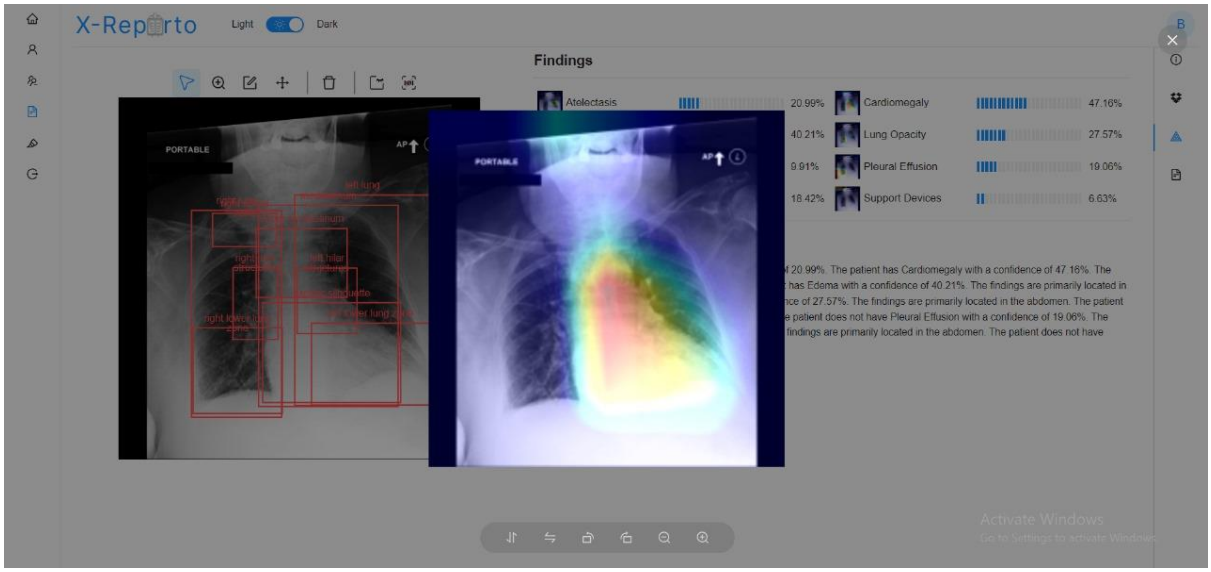


Figure 30 generate heatmap for each disease

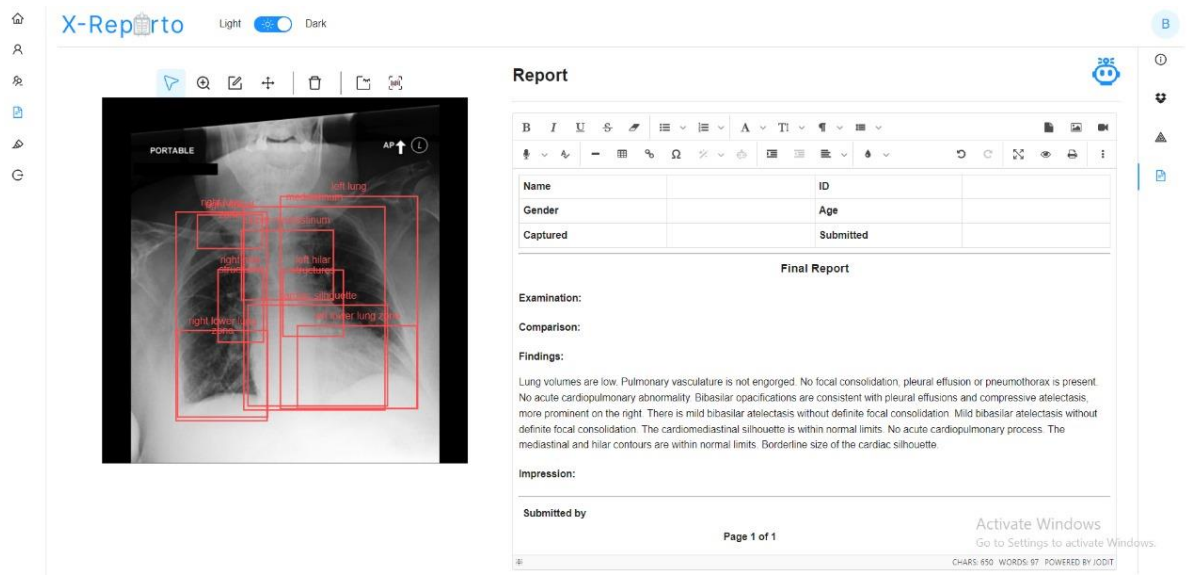


Figure 31 generate AI report and doctor can edit it

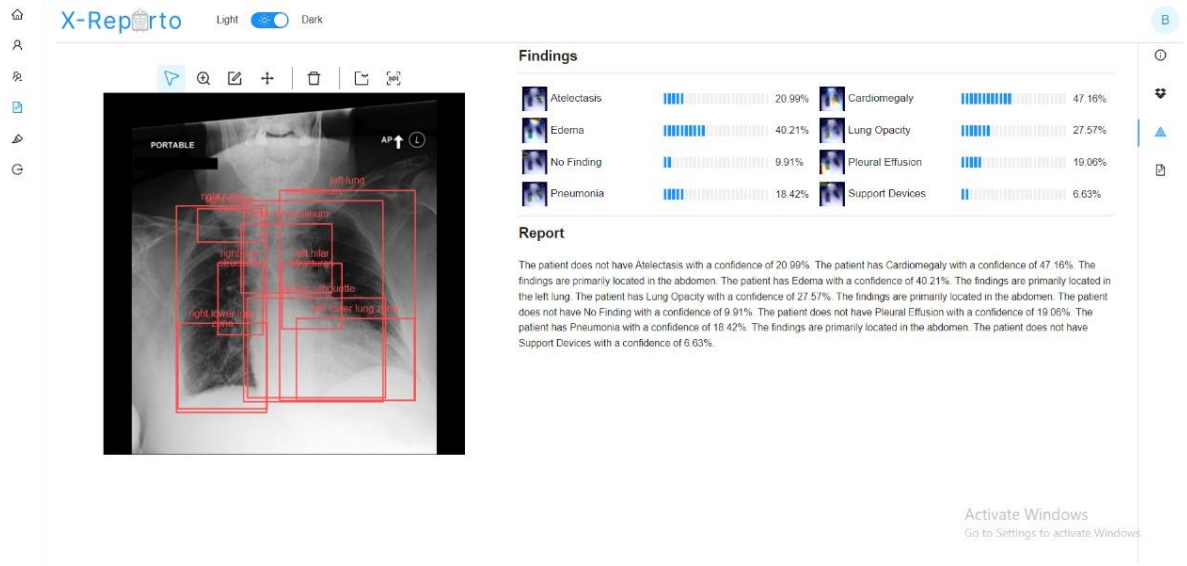


Figure 32 generate each disease probability and template report

Appendix E: Feasibility Study

- The project will be available in U.S hospitals
- Doctors will verify their diagnoses use this but in the future after doctors trust the AI solution the project will be trusted without review its result
- then the project will spread all over the world as it need no cost and get better result
- fresh graduate doctors can use as it will be available tool for them to use as they have less experience