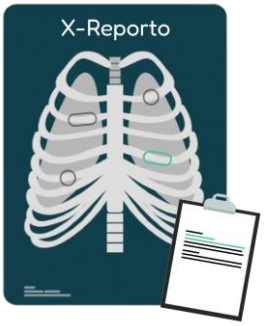
Cairo University
Faculty of Engineering
Department of Computer Engineering

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
Ahmed Sabry
Basma Elhoseny
Zeinab Moawad

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/

الملخص

هذا المشروع يعالج العبء المتزايد على أطباء الأشعة بسبب الانتشار المتزايد لأمراض الصدر، مما يؤدي إلى طوابير طويلة من تقارير الأشعة السينية التي تحتاج إلى تشخيص. هدف المشروع هو تطوير نظام تقارير شبه آلي يدعم أطباء الأشعة من خلال إنشاء تقارير أولية لكل منطقة تشريحية وتحديد الأمراض في تلك المناطق. نهجنا يتضمن تحسين صور الأشعة السينية لتصحيح العيوب التي تسببها أجهزة الأشعة السينية، وضمان الحفاظ على المعلومات التشخيصية الحيوية وجعلها سهلة التعرف. يقوم الأداة بإنشاء تقارير شاملة تعتمد على قوالب مخصصة للأمراض المحددة، مما يسهل عملية إعداد التقارير ويقلل من الوقت المطلوب للتشخيص.

تشمل النتائج الرئيسية للمشروع الصور المحسنة للأشعة السينية والتقارير التفصيلية الخاصة بالأمراض التي تنتجها الأداة. تم إجراء تطوير واختبار الأداة لضمان الدقة والموثوقية. تُظهر نتائج الاختبار تحسينات كبيرة في وضوح الصور ودقة التشخيص، وكذلك تقليل الوقت المطلوب لإعداد التقارير.

نجح هذا المشروع في تطوير وتنفيذ الأداة. نتائج هذا المشروع لا تخفف فقط من عبء العمل على أطباء الأشعة، بل تساهم أيضًا في تحسين رعاية المرضى من خلال تمكين التدخلات الطبية السريعة والدقيقة، مما يزيد في نهاية المطاف من معدلات بقاء المرضى على قيد الحياة.

هذا المشروع مدعوم من شركة Voyance Health، والتي قدمت دعمًا لا يقدر بثمن من خلال توفير موارد الخادم لتدريب النماذج على مجموعة بياناتنا الكبيرة. كما ساهموا بالفكرة الأساسية للمشروع وتابعوا تقدمنا عن كثب، مقدمين التوجيه والملاحظات طوال عملية التطوير

ACKNOWLEDGMENT

We extend our sincerest gratitude and appreciation to our esteemed supervisor, Dr. Yehia Zakaria, for his dedicated supervision, invaluable guidance, and unwavering support throughout our project. His mentorship has been instrumental in our journey.

We also wish to thank Voyance Health for their sponsorship and provision of servers, which were essential for training our models.

Special thanks go to Eng. Mohamed Shawky and Eng. Omar Samir, technical assistants at the Faculty of Engineering, Cairo University, for their continuous support and assistance.

Additionally, we express our heartfelt appreciation to our department for providing us with the necessary courses and resources to undertake this project. Their support and encouragement have been pivotal.

Thank you all for making this journey possible and for your unwavering belief in our capabilities.

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

Team Members

Name	Email	Phone Number
Ahmed Hosny	ahmed.alghany01@eng-st.cu.edu.eg	+2 01060668268
Ahmed Sabry	ahmed.ahmed017@eng-st.cu.edu.eg	+2 01067371115
Basma Elhoseny	basma.elhoseny01@eng-st.cu.edu.eg	+2 01152668680
Zeinab Moawad	zeinab.hassan00@eng-st.cu.edu.eg	+2 01111125861

Supervisor

Name	Email	Number
Dr. Yahia Zakria	Yahiazakaria13@gmail.com	+2 0١11 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

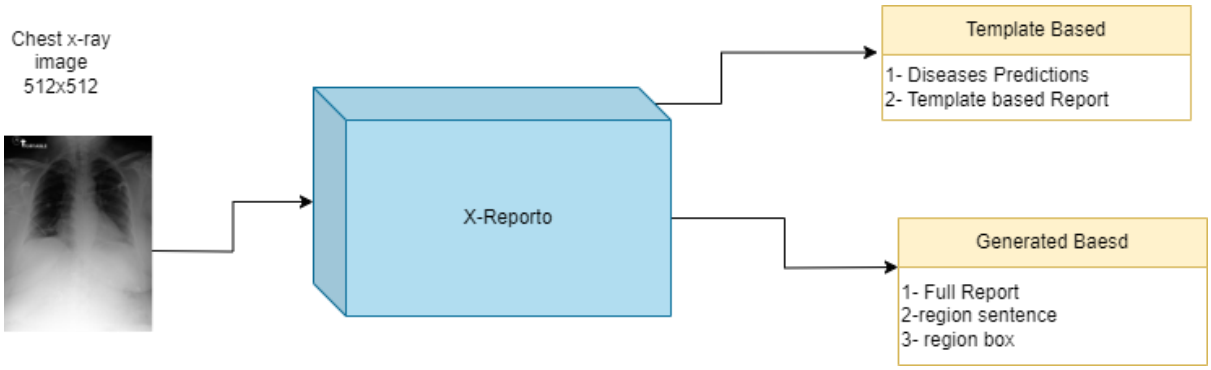


Figure١ X-Reporto Outcomes

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 .

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: Market Visibility Study

The market for radiological reporting tools and platforms is rapidly evolving, driven by the increasing demand for efficient, accurate, and standardized medical imaging interpretations. As radiologists face mounting pressures from high case volumes and the need for timely reports, innovative solutions are emerging to streamline workflows and enhance diagnostic accuracy. However, many existing tools have notable limitations, including insufficient automation, dependency on manual input, and lack of advanced AI integration. Our project aims to address these shortcomings by offering a comprehensive, AI-powered radiological reporting solution that improves efficiency, reduces errors, and provides a user-friendly interface, thereby meeting the critical needs of modern medical institutions.

2.1. Targeted Customers

Our product primarily targets radiologists and employees in medical institutions. Radiologists often face a high volume of cases that need thorough investigation, leading to significant delays due to the overwhelming number of X-rays requiring reports. Our tool is designed to assist radiologists by providing initial reports, thus streamlining their workflow and reducing their workload.

Additionally, our product benefits employees in medical institutions by helping manage reports, track doctors' accounts, and oversee unassigned and pending cases. The system grants radiologists full control over the final reports while offering an intuitive and user-friendly interface for seamless interaction.

Furthermore, our system is designed to handle errors caused during the acquisition of X-rays, ensuring more accurate and reliable reports. The easy-to-use UI enhances user experience, making the process of reviewing and managing cases more efficient and effective.

By implementing our solution, we aim to enhance efficiency and accuracy in the radiology department, ultimately improving patient care and institutional productivity.

2.2. Market Survey

2.2.1. Radio Report [2]

RadioReport is a software tool designed to streamline the radiological reporting process. It guides users through the reporting process like a virtual interview, generating complete and standardized reports efficiently.

Pros:

- **Efficiency:** RadioReport® significantly reduces reporting time by enabling users to generate complex sentences with just a few mouse clicks.
- **Standardization:** Ensures that reports are complete and adhere to standardized formats, improving the quality and consistency of reports.

Cons:

- **Dependency on Radiologists:** Despite its efficiency, radiologists must still be involved in the report generation process. The software functions more like a guided form-filling tool rather than an autonomous reporting system.
- **Low level of AID:** RadioReport is just reformulating the problem of report writing as paragraphs to be like surveys being filled by the radiologists instead of a more advanced level aid.
- **Lack of AI Integration:** RadioReport does not utilize artificial intelligence to enhance its reporting capabilities. This limits its potential to automate and improve reporting accuracy and efficiency further.
- **Limited Innovation:** While it improves reporting speed and standardization, it does not introduce groundbreaking advancements in radiological reporting workflows or image analysis.
- **Cost and Accessibility:** The software may be expensive for some institutions and might require significant investment in training and integration.

[2] https://radioreport.com/

2.2.2. ViTAL[3]

Developed by researchers at Berkeley University in the summer of 2023, ViTAL is an annotation tool that provides sentence completion suggestions for radiologists analyzing chest X-rays. Similar to Gmail's autocomplete feature, ViTAL aids in generating report text based on the specific X-ray being analyzed. The tool also includes a dynamic saliency map that highlights the parts of the X-ray the model focuses on when providing autocomplete suggestions.

Pros:

- **Increased Efficiency:** ViTAL speeds up the reporting process by providing sentence completion suggestions, reducing the time radiologists spend on report writing.
- **Dynamic Saliency Map:** The saliency map feature helps radiologists understand which parts of the X-ray the model considers most relevant, potentially improving report accuracy.
- **User-Friendly:** The tool's autocomplete functionality is intuitive and easy to use, similar to familiar systems like Gmail.

Cons:

- **Medium Level of Assistance:** While ViTAL provides sentence completion suggestions, it still requires radiologists to initiate and complete the report writing. This level of assistance is beneficial but not as transformative as full AI-driven report generation.
- **No Denoising Functionality:** ViTAL does not offer image denoising or enhancement features, which could further improve the quality of X-ray analysis and reporting.
- **Dependency on Radiologists:** Like other tools, ViTAL still depends on radiologists to review and finalize reports, limiting the extent of automation and efficiency gains.
- **Error Rates:** Although ViTAL aims to speed up the reporting process, the reliance on radiologists to initiate the writing may still lead to increased error rates if radiologists work at a faster pace without the tool addressing potential mistakes.

[3] https://www.berkeley.edu/

2.3. Business Case and Financial Analysis

We are already sponsored a Voyance health company which is interested in the project to use it in U.S hospitals

- **Business Case:** we introduce tool for doctors and will help them in the next 5 years it will used in all US hospitals and in the next 10 years it will spread all over the world special in the country which have leakage of doctors in Africa, we will make the price for our tool is less than X-ray doctor as it serve people and care about their life not luxury for people

Month	1st month	2nd month	3rd month	4th month	5th month	6th month	7th month	8th month
Total Paid doctors	5.00	20.00	30.00	75.00	150.00	240.00	300.00	296.00
Free Trial doctors	50.00	100.00	150.00	250.00	500.00	800.00	750.00	760.00

- **Financial Analysis:**
  - The Capex (Capital Expenditure):

Resource	Price
Marketing and Promotion	$2,000
Initial development of the web interface	$1,000
Assets for the office (chairs, tables, etc.)	$10,000
Total	$13,000

  - The Opex (Operational Expenses):

Resource	Price
Free trial per doctor	$0.16
Paid members hardware per doctor per month	$65.70
Hosting servers per month	$62.40
Office rent per month	$2,500

- **Cost flow:**

Month	1st month	2nd month	3rd month	4th month	5th month	6th month	7th month	8th month
OpEx								
Free trial costs	$8	$16	$24	$40	$80	$128	$120	$122
Paid doctor support cost	$329	$1,314	$1,971	$4,928	$9,855	$15,768	$19,710	$19,447
Constant Monthly costs	$17,562	$17,562	$17,562	$17,562	$17,562	$17,562	$17,562	$17,562
Service Details								
Total Paid doctor	5.00	20.00	30.00	75.00	150.00	240.00	300.00	296.00
Free Trial doctor	50.00	100.00	150.00	250.00	500.00	800.00	750.00	760.00
Service Revenue								
Revenue (paid user)	$900	$3,600	$5,400	$13,500	$27,000	$43,200	$60,000	$59,200
Final details								
Profit	-$16,999	-$15,292	-$14,157	-$9,030	-$497	$9,742	$22,608	$22,069
Comments	Initial release	Increase in popularity	Increase in popularity	Increase in popularity	Increase in popularity	Hype and first positive profit	Increase the price here	Settling of the number of users

Chapter 3: Literature Survey

3.1. Double Convolution

Convolutional Operation with Filters: The convolution operation involves sliding the filter over the input data and computing the element-wise product between the filter and the overlapping region of the input. The sum of these products forms a single element in the output feature map. The filter is then moved to the next position, and the process is repeated until the entire input is traversed.

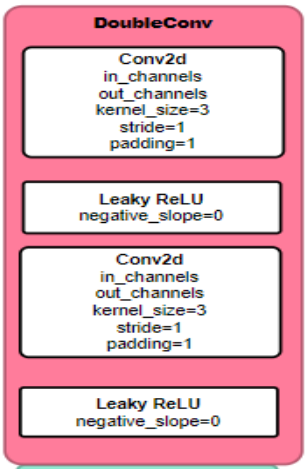


Figure 2 2D- Conv

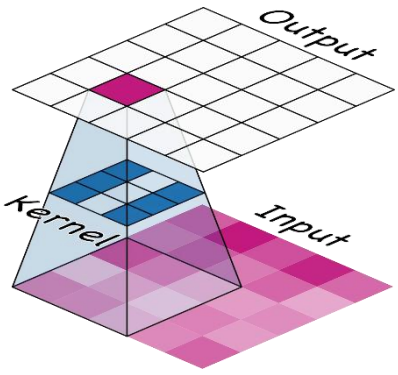


Figure 3 Convolution

After filtering, the feature maps pass through the activation function. The function introduces non-linearity to the network, allowing it to learn complex patterns and make better predictions. The rectifier function has a graph like this:

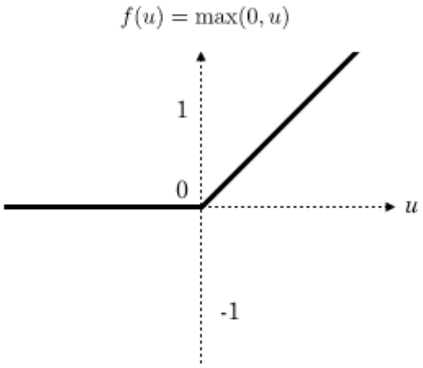


Figure 4 Relu Function

3.2. Up Sampling Bilinear:

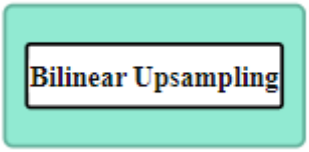


Figure 5 Up Sampling Bilinear Conv Arch

A more sophisticated approach that uses linear interpolation to compute the values of the new pixels, resulting in smoother transitions than nearest neighbor up sampling.

3.3. MAX Pooling:

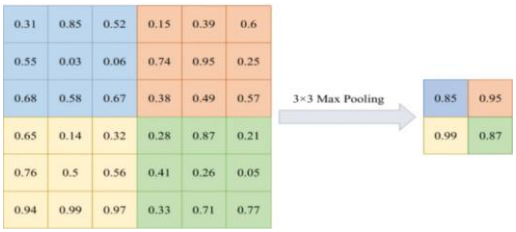


Figure 6 Max Pooling

Max pooling is a commonly used feature extraction operation, typically applied in convolutional neural networks. It reduces the spatial dimensions of features by selecting the maximum value within each small window or region.

3.4. CNN:

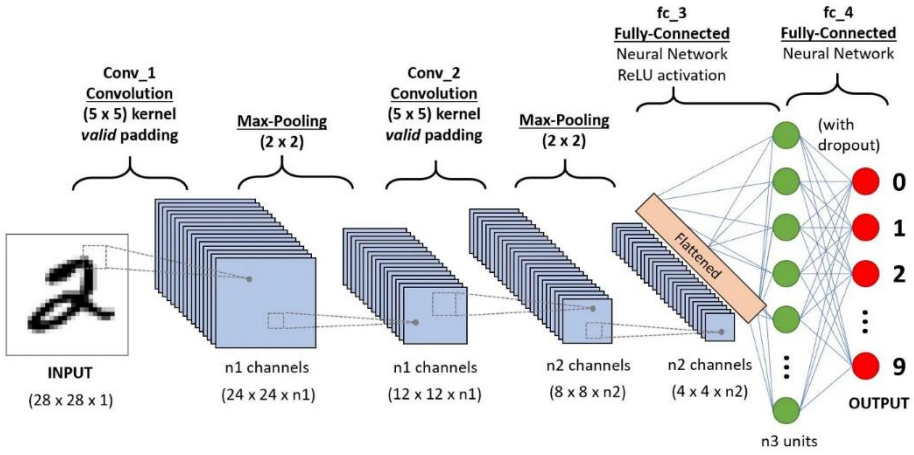


Figure 7 Conv Neural Networks

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**.**

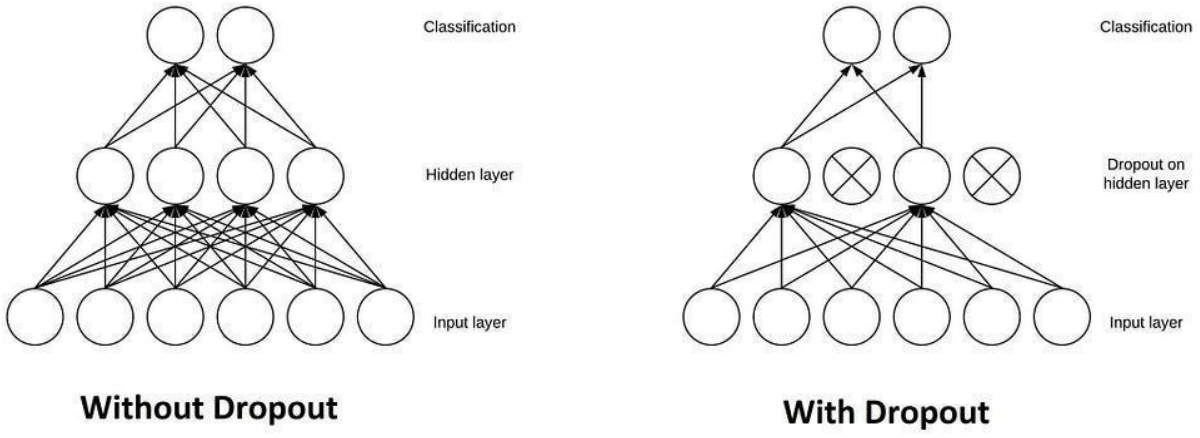


Figure 8 Dropout Technique in NN

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.

3.5. VGG-19:

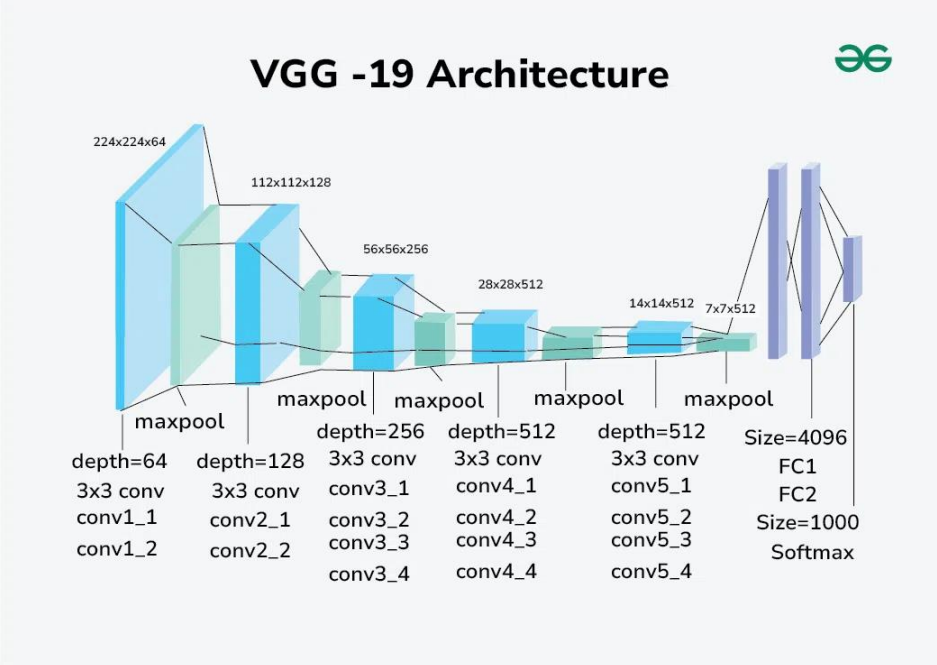


Figure 9 VGG Network Architecture

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.

3.6. U-Net:

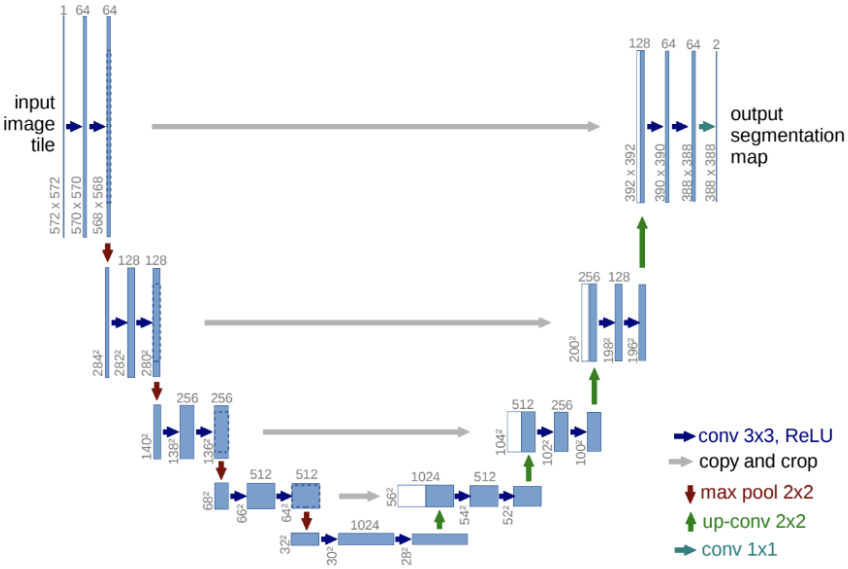


Figure 10 U-Net Architecture

The U-Net proposed by [3] Ronneberger O et al is an architecture for semantic segmentation. It consists of a contracting path and an expansive path. The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (Relu) and a 2x2 max pooling operation with stride 2 for down sampling. At each down sampling step, we double the number of feature channels. Every step in the expansive path consists of an up sampling of the feature map followed by a 2x2 convolution (“up-convolution”) that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a Relu. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

3.7. GANs:

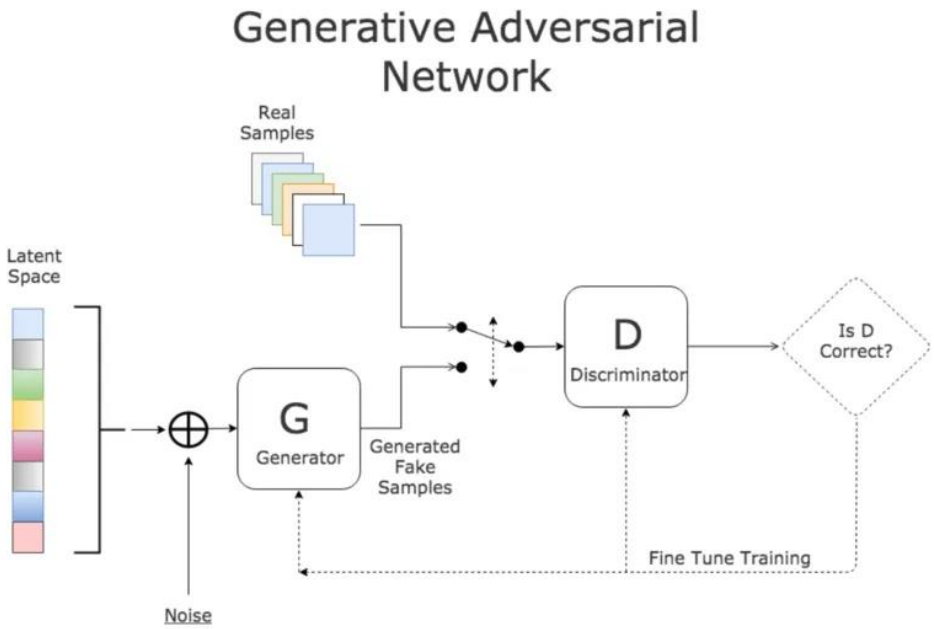


Figure 11 Generative Adversarial Network

The GANs proposed by [3] Ahmadi et al, represent a cutting-edge approach to generative modeling within deep learning, often leveraging architectures like convolutional neural networks. The goal of generative modeling is to autonomously identify patterns in input data, enabling the model to produce new examples that feasibly resemble the original dataset. Generative Adversarial Network (GAN) consists of two neural networks, namely the Generator and the Discriminator, which are trained simultaneously through adversarial training.

1. **Generator:** This network takes random noise as input and produces data (like images). Its goal is to generate data that's as close as possible to real data.
2. **Discriminator:** This network takes real data and the data generated by the Generator as input and attempts to distinguish between the two. It outputs the probability that the given data is real.

3.8. 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.

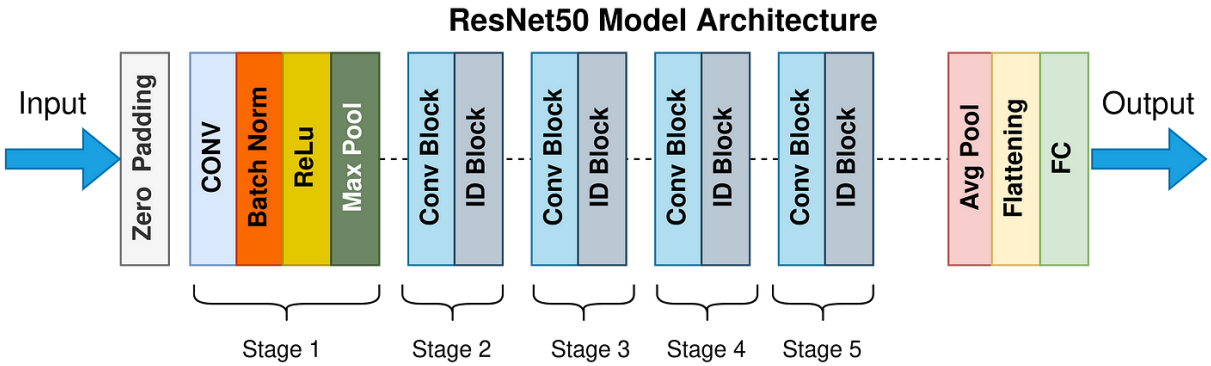


Figure 12 ResNet50 Architecture

3.8.1. 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.

3.9. Faster RCNN

Faster R-CNN [12] (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.

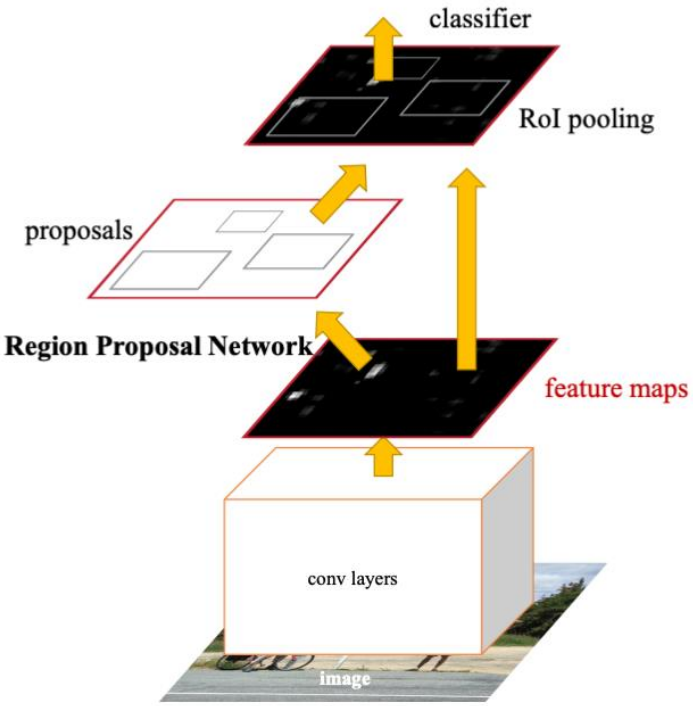


Figure 13 Faster RCNN Architecture

:

3.9.1. 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.

3.9.2. 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.

3.9.3. 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.

3.9.4. 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)

3.10. Felzenszwalb Segmentation Algorithm

1. Graph Representation: The algorithm constructs a graph where pixels are represented as nodes. Edges are defined between neighboring pixels based on a measure of similarity, typically based on color and intensity differences.
2. Segmentation Process: Starting with each pixel as its own segment, the algorithm iteratively merges segments based on a criterion involving a threshold parameter and the difference in intensity between segments.
3. Merge Criterion: Segments are merged if the difference in intensity between them is less than a specified threshold. This threshold is adaptive and depends on the size of the segments being considered for merging.
4. Resulting Segments: The output of the algorithm is a set of segmented regions where pixels within each region are similar in color and intensity. This segmentation is hierarchical and can produce a range of segmentations depending on the chosen threshold parameter.
5. Applications: Felzenszwalb’ s algorithm is commonly used in computer vision tasks such as object detection, image segmentation, and image analysis where identifying coherent regions based on pixel similarity is necessary.

The algorithm is efficient and provides good results for various types of images, making it a popular choice in the field of image processing and computer vision.

3.11. HOG (Histogram of Oriented Gradients)

HOG is a feature descriptor that counts occurrences of gradient orientation in localized portions of an image. It divides the image into small regions called cells, computes a histogram of gradient directions within each cell, and then normalizes local contrast in overlapping blocks.
The main steps for computing HOG features from an image are:

1. **Compute the gradient magnitude and angle:** We calculate the gradient of the pixel intensities in each direction (horizontal and vertical). It is like doing convolution with a kernel of [-1,0, +1] in X and Y direction. This will eventually give Gx and Gy value for every pixel. From Gx and Gy, Gradient magnitude and angle is calculated as follows. The magnitude of this gradient tells us how strong the edge is, while the angle tells us its direction.

$$g = \sqrt{g_x^2 + g_y^2}$$
$$\theta = \arctan \frac{g_y}{g_x}$$

Figure 14 Magnitude and Angle of gradients

2. **Calculate Histogram of Gradient for cell:** Divide the image into small, connected regions called cells Compute a histogram of gradient over each cell; The histogram bins represent the discretization of orientations into 9 bins each of 20 degrees. Compute the sum of all the gradient magnitudes of all the pixels with gradient angle in the respective bin. Each cell's histogram will have 9 values of gradient magnitude.

3. **Normalize the histogram for Blocks:** To counteract inconsistencies caused by varying lighting and noise, the histograms are normalized over larger, overlapping areas called blocks. Each Block can have multiple cells. (Ex: 2X2 Cells i.e. 16X16 Pixels). Concatenate all cell's HOGs in single array and then normalize it with L2 Norm. This calculation is done with overlapping block with

stride of 1 cell. Collect all HOG Features: Collect HOG features from all blocks in the image and concatenate them into a combined feature vector representing the HOG feature for the entire image. As shown in the following equation

$$(\frac{M}{8} - 1)^2 \times 36.$$

3.12. Haar

Haar features are extracted from rectangular areas in an image. The feature's value is based on the pixel intensities. Usually, it is calculated using a sliding window, and the area within the window is partitioned into two or more rectangular areas.
the key strength of Haar features lies in their ability to represent three patterns:

1. Edges: Either vertical or horizontal due to how we oriented the rectangular area. They are useful for identifying boundaries between different image regions.
2. Lines: The diagonal edges in an image. They are useful for identifying lines and contours in objects.

Center-surrounded features: This detects the changes in intensity between the center of a rectangular region and its surrounding area. This is useful to identify objects with a distinct shape or pattern. They are similar to convolution kernels taught in the Convolution Neural Networks course. We will apply these haar.

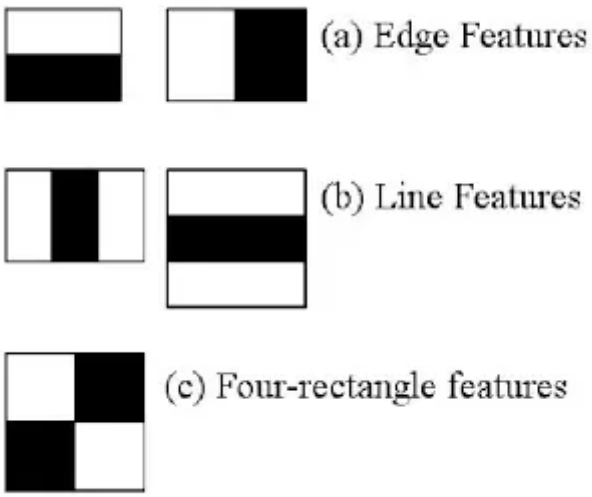


Figure 15 Feature Extraction Filters

3.13 Transformer

3.13.1. Overview

The Transformer architecture revolutionized the field of natural language processing by utilizing a mechanism known as self-attention, which allows the model to weigh the importance of different words in a sentence when making predictions. This architecture facilitates parallel processing of data, leading to significantly faster training times compared to traditional recurrent neural networks. Building on the Transformer framework, GPT-2, a large-scale, unsupervised language model pre-trained on diverse internet text. GPT-2 excels at generating coherent and contextually relevant text by predicting the next word in a sequence, given all the previous words within the context. This capability makes GPT-2 highly effective for a wide range of natural language understanding and generation tasks. My work leverages GPT-2's advanced language generation capabilities to produce detailed and accurate medical reports from visual features extracted from X-rays, ensuring that the generated text is both medically accurate and contextually appropriate.

3.13.2. Architecture

GPT-2, or Generative Pre-trained Transformer 2, is built upon the Transformer architecture illustrated in figure 3.12.2 , which relies on a self-attention mechanism to model long-range dependencies in text. The core of GPT-2's architecture consists of multiple layers of transformer decoders, each comprising multi-head self-attention and feed-forward neural networks. The model employs layer normalization and residual connections to improve training stability and performance. Each self-attention layer allows GPT-2 to consider the context from all positions in the input sequence simultaneously, enabling it to generate coherent and contextually relevant text. With 1.5 billion parameters for large or 170 million parameters for small, GPT-2 is capable of understanding and generating a wide range of human-like text, thanks to its extensive pre-training on diverse internet text data. This vast pre-training allows GPT-2 to capture a broad spectrum of linguistic patterns and nuances, making it highly effective for tasks such as text generation, summarization, translation, report generation and more. By fine-tuning GPT-2 on specific domains, such as medical literature, its performance can be further optimized for generating specialized content, like detailed medical reports from visual data.We will discuss the building blocks of the network in the following subsections.

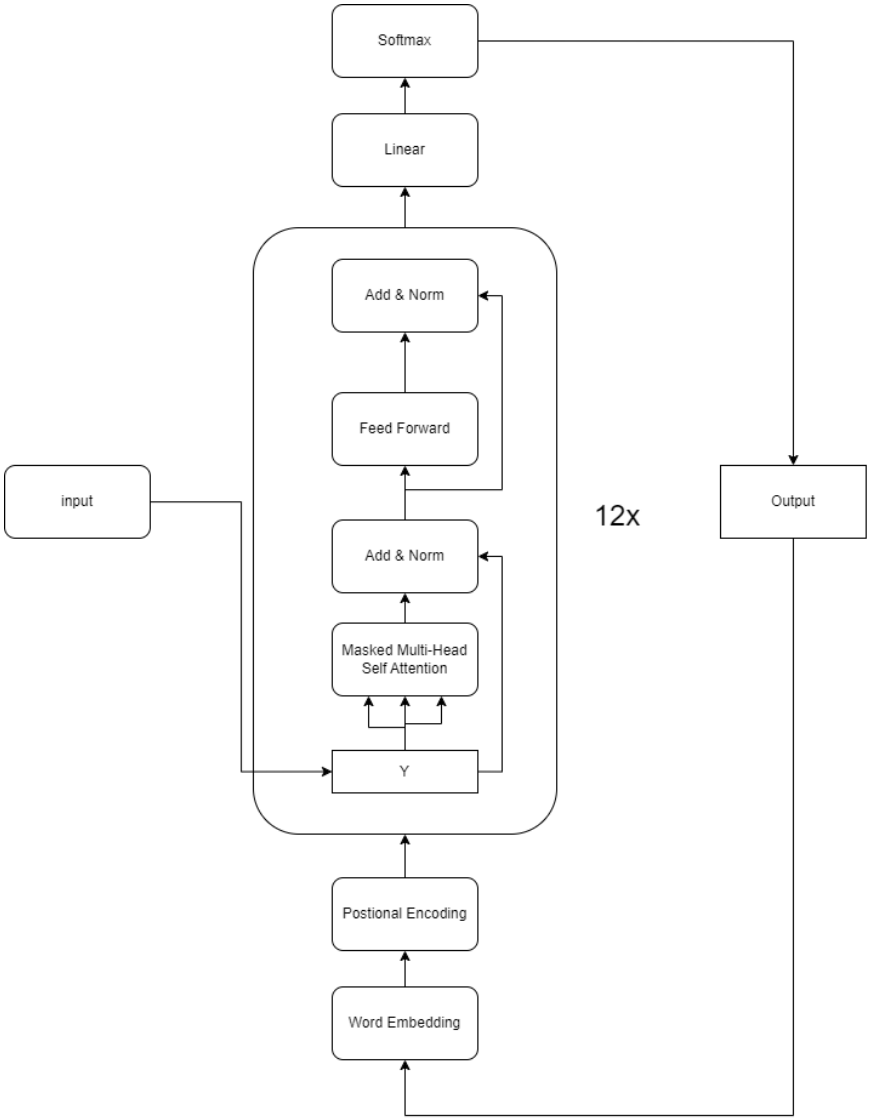


Figure 16 Transformer Decoder Architecture

3.13.2.1 Word Embedding

We used a word embedding of a pre-trained GPT2 model that trained self-supervised on medical articles so that we could understand medical words and generate good word representations of them.

3.13.2.2 Positional Encoding Embedding

We used the position information of the current token to help the model predict the next word correctly and calculate correct attention scores for other words in a sentence.

3.13.2.3 Masked Multi-Head Self-Attention

In this step, the model uses masked multi-head self-attention mechanisms to focus on different parts of the input sequence simultaneously. The masking ensures that the prediction for a given position only depends on the known outputs at prior positions, maintaining the autoregressive property. The equation of the normal self-attention equation is shown below

$$SA(Y) = softmax\left((YW_q)(YW_k)^T\right)(YW_v)$$

where Y represents the token embeddings and Wq, Wk, Wv are the query, key, and value projection parameters

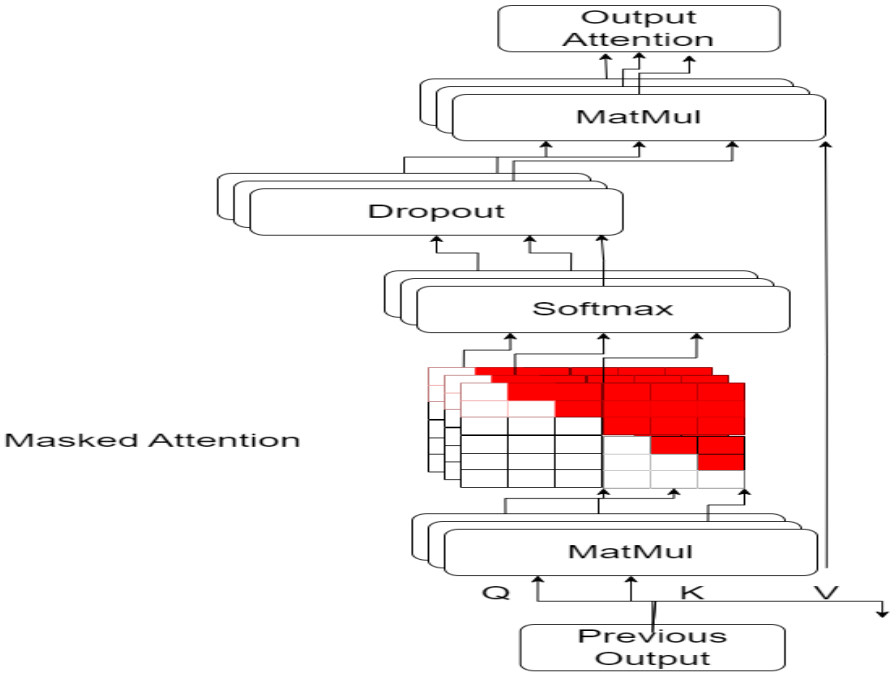


Figure 17 Self Attention Architecture

3.13.2.4 Residual Connections and Normalization

After the self-attention mechanism, the model adds a normalization layer to the input and then applies the self-attention mechanism then the dropout layer, and finally adds the input and output of the dropout layer, These steps stabilize and speed up the training process by maintaining consistent input distributions across layers and prevent overfitting of model and reduce dependency between blocks.

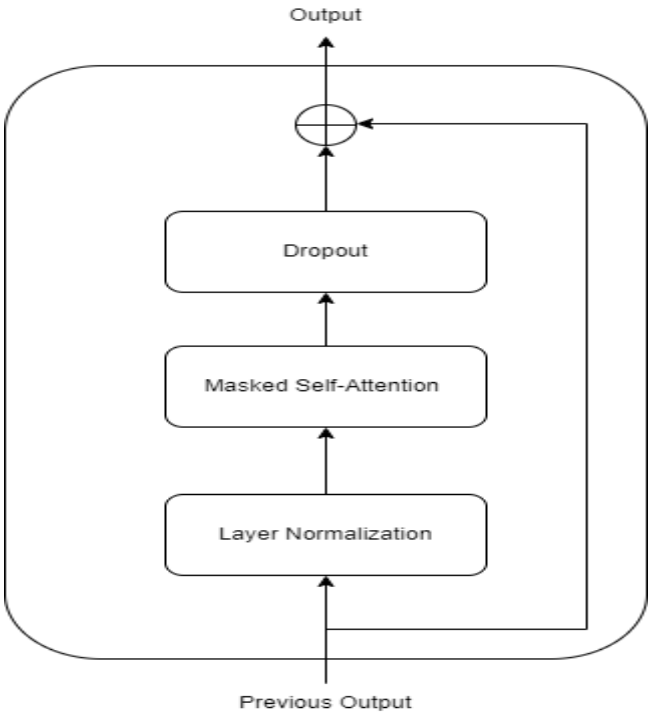


Figure 18 Residual Block

3.13.2.5 Feed Forward

The output from the addition and normalization step is then passed through a feed-forward neural network. This network typically consists of two linear layers with a ReLU activation function in between, allowing the model to learn complex transformations.

3.13.2.6 Softmax

Finally, the model applies a softmax function to the output logits, converting them into probabilities. This step is crucial for generating the most likely next word in the sequence, completing the process of text generation based on the input visual features.

3.14 Class Activation Mapping (CAM)

Zhou B et al [3] proposed a method to explain decisions made by neural networks (NN) for classification tasks. CNNs have been shown to function as effective object detectors, allowing for region localization without requiring prior network training on localization tasks. However, this capability diminishes upon adding fully connected (FC) layers. To address this, Zhou B et al [3] introduced Global Average Pooling (GAP), which performs average pooling across the output of the backbone layers. By using weights in the FC layer, we can then pinpoint the regions of activation that underpin the classification decisions made by the network.

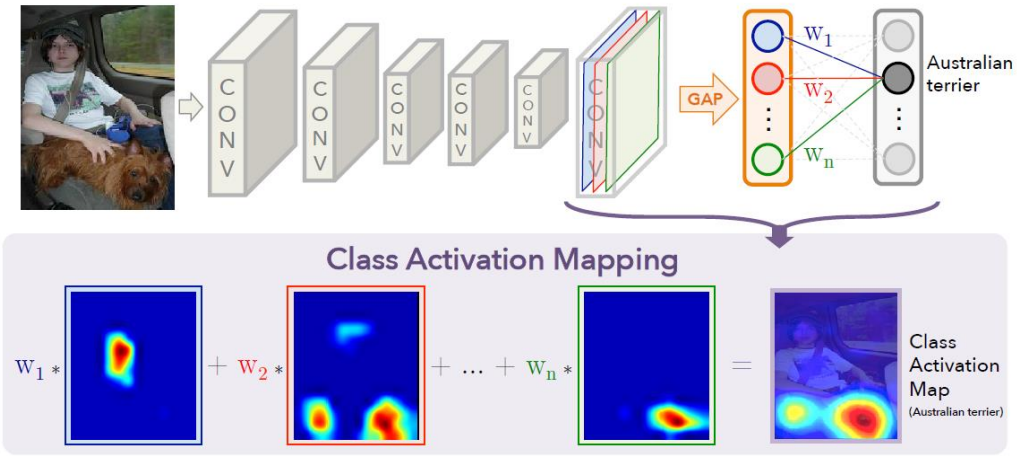


Figure 19 Class Activation Mapping describing activation regions for label prediction

The activation Maps are computed using the weights of the FC layer and the Final Transition Layer in the CNN.

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y)$$

$$= \sum_{x,y} \sum_k w_k^c f_k(x,y).$$

3.15. Comparative Study of Previous Work

Recent advancements in automatic radiology report generation have increasingly focused on Transformer-based architectures. Early approaches primarily utilized CNN-RNN frameworks, but the shift towards Transformers has highlighted their effectiveness in integrating visual and textual features. Studies have adapted the standard Transformer by incorporating relation memory units or memory matrices to enhance cross-modal interactions. Various methods have also been proposed to improve attention to abnormal regions, such as aligning visual features with disease tags and leveraging medical knowledge graphs. In contrast, our approach emphasizes the direct extraction of visual features from abnormal regions through object detection, supplemented by an abnormality classification module to encode relevant information.

Furthermore, the literature has explored hierarchical strategies for generating coherent reports by breaking down the task into manageable steps. Similar to prior work, we also decompose report generation into multiple stages, with a particular focus on region-level feature extraction. Unlike other methods that employ multi-head transformers for specific anatomical regions, our approach utilizes object detection to generate region-level sentences through a shared transformer decoder. While requiring more supervision, this method is less technically complex, enhancing interactivity and transparency in report generation.

3.15.1 Comparison With Previous Work

Aspect	Previous Work	Our Approach
Architecture	CNN-RNN, adapted Transformers	Transformer with object detection
Feature Extraction	General image-level features	Specific region-level features
Attention Mechanism	Enhanced with memory units or graphs	Direct extraction from abnormal regions
Report Generation Method	Hierarchical or multi-head transformers	Shared transformer decoder
Complexity	Often more complex due to multiple heads	Less technically complex, more interactive
Supervision Requirement	Varies	Requires more supervision
Transparency	Limited transparency	Higher degree of transparency and explainability

Table 1 Comparison to X-Reporto Competitors

3.16. Implemented Approach

3.16.1. Denoiser Approach

GANs have been applied successfully in medical imaging [5] but have not previously been used with high-resolution imaging techniques. The challenge is that the high-resolution images produced include finely detailed features with high-frequency content.

Our GAN-based method adapts the U-Net network architecture to meet the specialized requirements of improving the quality of images generated.

We demonstrate that they can be trained with limited data, perform well with high-resolution datasets, and generate greatly improved images.

3.16.2. Object detector

We use Faster RCNN as we can get the features from it if we use YOLO we can't get the feature

Features represent the main part from our code as the classifiers and GPT-2 depends on it no other module in the pipeline use the image all use the features only so so need to get them from it using high performance model like resnet50 thats why we didn't replace it with Vgg19

3.16.3. Language Model

We used GPT2 architecture as basics of our language model as it's decoder based only which suits our problem of image captioning where encoder is object detector, and transformer based architecture as language model is best language models can be used as it's fast for training and best performance it can achieve with.

Chapter 4: 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.

4.1. Survey of Available Chest X-ray Datasets

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

4.1.1. Chest Diseases [4]

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.

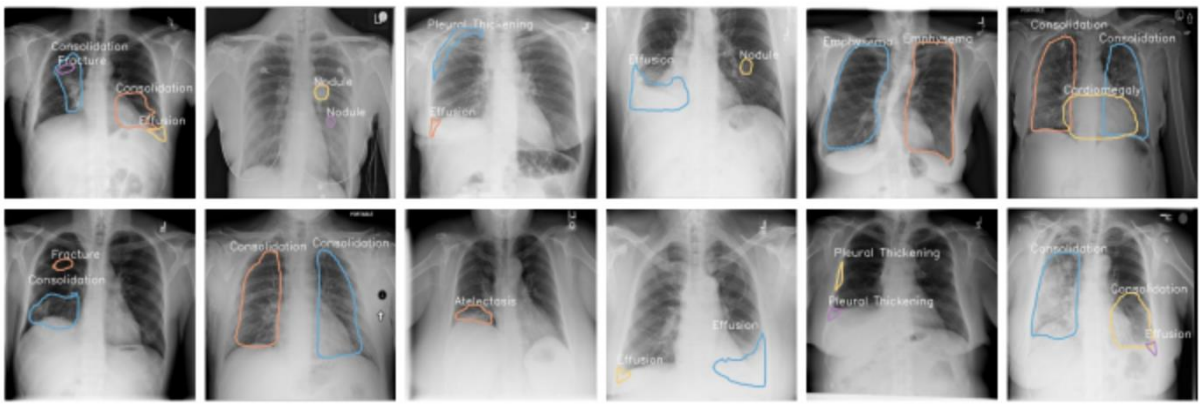


Figure 20 Chest Diseases Dataset examples

4.1.2. PADCHEST [5]

Large-scale, high-resolution labeled data set of chest radiographs for automated medical image exploration along with their associated reports. This dataset includes

[4] https://paperswithcode.com/dataset/chestx-det
[5] https://bimcv.cipf.es/bimcv-projects/padchest/

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.

Figure 21 PADCHEST Dataset examples

4.1.3. VinDr-CXR [6]

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

[6] https://physionet.org/content/vindr-cxr/1.0.0/

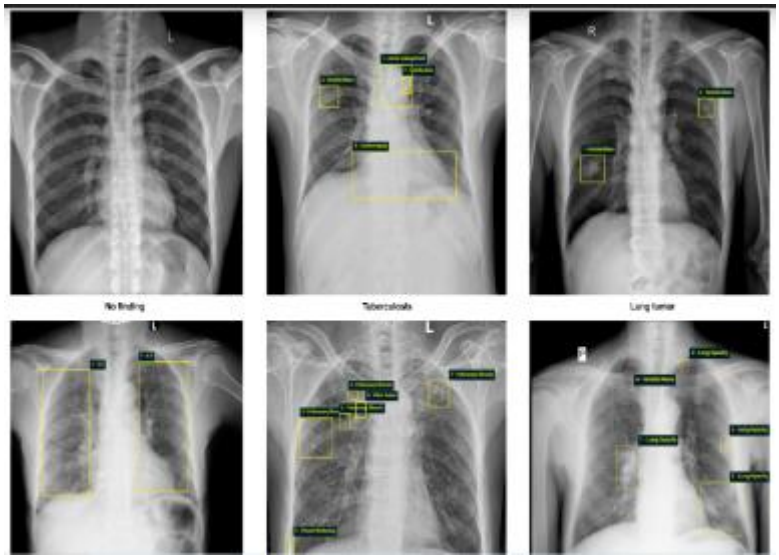


Figure 22 VinDr-CXR Dataset examples

4.1.4. MIMIC-CXR [7]

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

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

[7] - https://physionet.org/content/mimic-cxr/2.0.0/

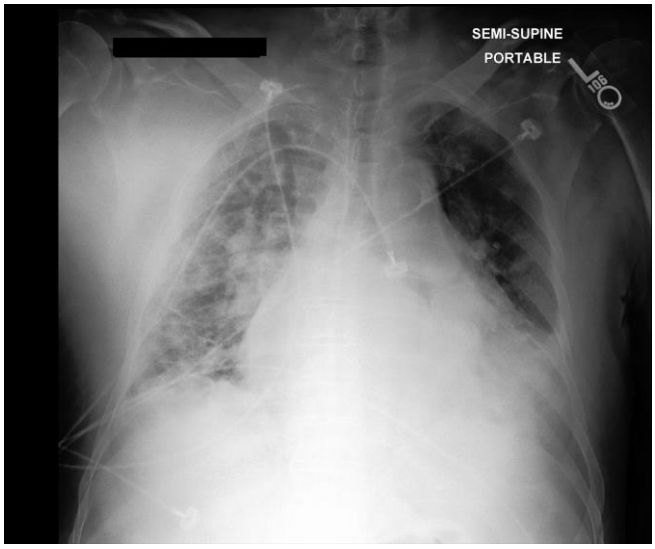


Figure 23 MIMIC-CXR Dataset examples

4.2. Selection and Access of Primary Dataset [8]

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

4.2.1 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.

4.2.2 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.

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

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.

```
files/
  p10/
    p10000032/
      s50414267/
        02aa804e-bde0afdd-112c0b34-7bc16630-4e384014.jpg
        174413ec-4ec4c1f7-34ea26b7-c5f994f8-79ef1962.jpg
      s53189527/
        2a2277a9-b0ded155-c0de8eb9-c124d10e-82c5caab.jpg
        e084de3b-be89b11e-20fe3f9f-9c8d8dfe-4cfd202c.jpg
      s53911762/
        68b5c4b1-227d0485-9cc38c3f-7b84ab51-4b472714.jpg
        fffabebf-74fd3a1f-673b6b41-96ec0ac9-2ab69818.jpg
      s56699142/
        ea030e7a-2e3b1346-bc518786-7a8fd698-f673b44c.jpg
```

Figure 24 MIMIC-CXR Folder Structure

4.3. Data Preprocessing for Effective Learning

We followed the steps below for preparing our dataset for the training process.

1. We need to create a file for reading data for each module so we upload all the dataset in a subset in colab which we can create a file of data information on the server easily for training, evaluation and testing.

2. Apply statistics on data to set parameters in Modules and the result

Mean	0.474
STD	0.301
Range Bounding box numbers per image	24-29 Boxes
Region Classifiers positive weights	2.24
Abnormal Classifiers positive weights	6
Maximum Sentence Length	264
Average Number of sentences	16

Table 2 MIMC CXR Data Statistics

3. Apply Data preprocessing:
   1- preprocessing image: resize longest dimension to 512 and pad image , all operations consider bounding box transformation.
   2- preprocessing sentences:by tokens all box phrases, also padding attention mask,label ids, input ids to be maximum sequence length.

4.4. Labels for MIMIC-CXR

As mentioned in section 4.2, we gained access to the MIMIC-CXR dataset, which contains DICOM format images accompanied by free-text radiology reports. However, for our template-based report generation detailed in section x.x, we require labeled findings from these X-rays. These labels were sourced from external resources, specifically extracted using CheXpert, an open-source rule-based tool built on NegBio. The labels correspond to the following 14 most prevalent observations in chest diseases:

- No Finding
- Enlarged Cardiom.
- Cardiomegaly
- Lung Lesion
- Lung Opacity
- Edema
- Consolidation
- Pneumonia
- Atelectasis
- Pneumothorax
- Pleural Effusion
- Pleural Other
- Fracture
- Support Devices

Each label may have one of the following values indicating the presence and certainty of the finding in the report: 0 for negative presence, 1 for positive presence, -1 for uncertain polarity, and NaN for not mentioned or failed to parse.

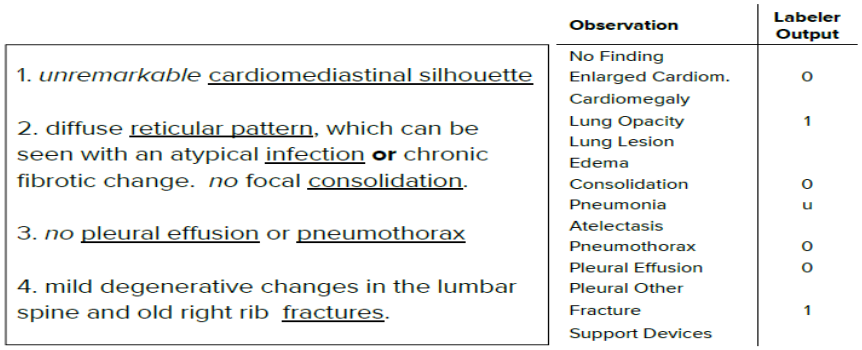


Figure 25 Output example for CheXpert labeler on a free-text Report

Chapter 5: 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.

5.1. 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.

5.2. 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.

5.2.1. Block Diagram

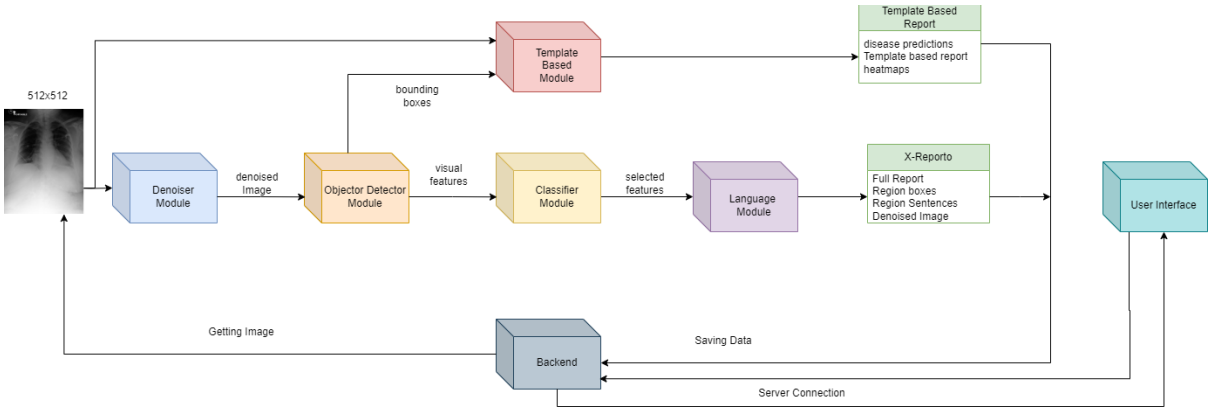


Figure 26 X-Reporto System Block Diagram

5.3. Denoiser Deep Learning Approach

5.3.1. Functional Description

X-ray images of 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.

5.3.2. Modular Decomposition

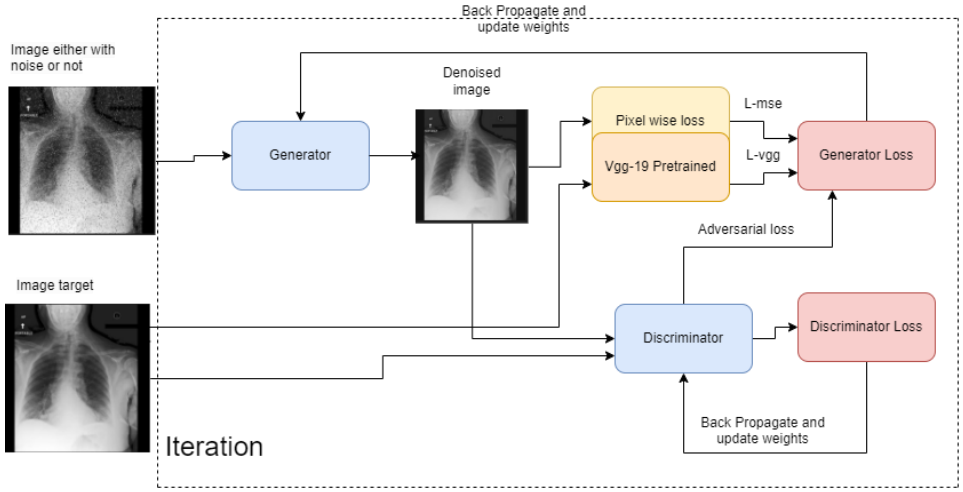


Figure 27 Denoiser Architecture

5.3.2.1 Generator Architecture

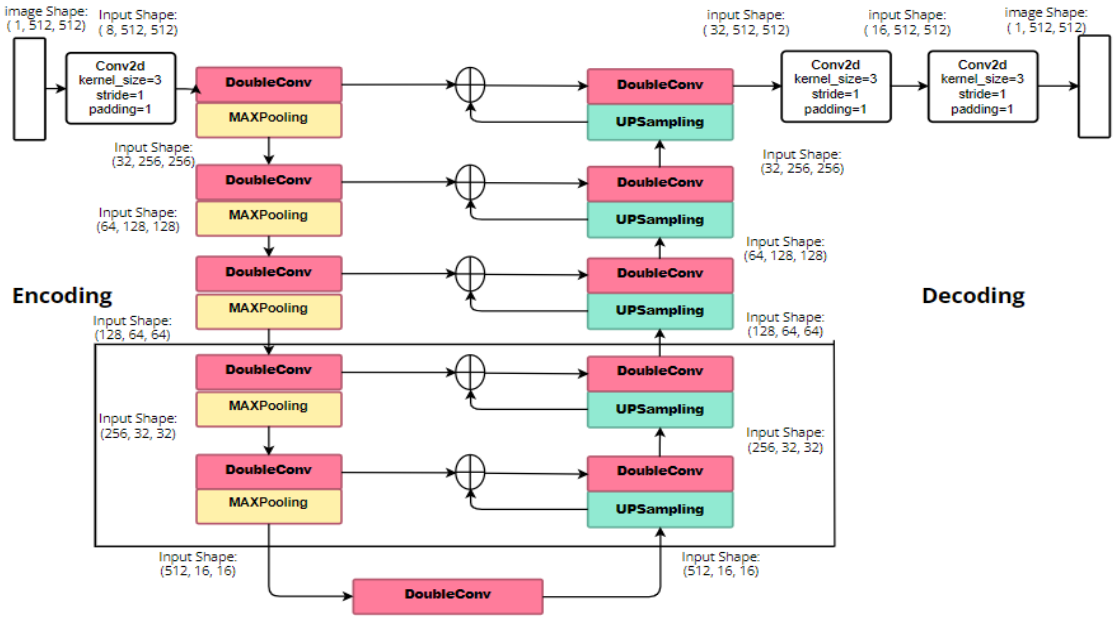


Figure 28 Generator Architecture

U-Net architecture proposed for biomedical image segmentation. It comprises a down-sampling network followed by an up-sampling network It adapts the U-Net architecture in three main ways:

- There are five (instead of four) down-sampling layers and up-sampling layers.
- All convolution layers keep the same image size.
- Eight 1 × 1 convolution kernels are applied to the input.

In the down-sampling process, three sets of two convolution kernels (the three boxes) extract feature maps. Then, followed by a pooling layer, the feature map projections are distilled to the most essential elements by using a signal maximizing process. Ultimately, the feature maps are 1/32 of the original size: 16x16.Successful training should result in 512 channels in this feature map, retaining important features.

In the up-sampling process, bi-linear interpolation is used to expand feature maps. At each layer, high-resolution features from the down-sampling path are concatenated to the up-sampled output from the layer below to form a large number of feature channels.

This structure allows the network to propagate context information to higher-resolution layers, so that the following convolution layer can learn to assemble a more precise output based on this information.

5.3.2.2 Discriminator Architecture

Discriminator has six 2D 3×3 CNN layers and two fully connected layers. Each CNN layer is followed by a leaky rectified linear unit as the activation function. Following the same logic as in G, all convolutional layers in D have the same small 3×3 kernel size. Let CkSs-n denote a convolution layer with a kernel size of k × k, a stride of s × s, n output channels, and leaky Relu activation function. The discriminator network consists of C3S1-64, C3S2-128, C3S1-128, C3S2-128, C3S1-256, C3S2-4, one hidden fully connected layer with 64 neurons and leaky Relu activation, and an output layer with one neuron and linear activation. There is no sigmoid cross-entropy layer at the end of the discriminator.
This combination of strides helps in gradually reducing the spatial dimensions while increasing the number of channels (feature depth), which is a common practice in convolutional neural networks for tasks like discrimination.

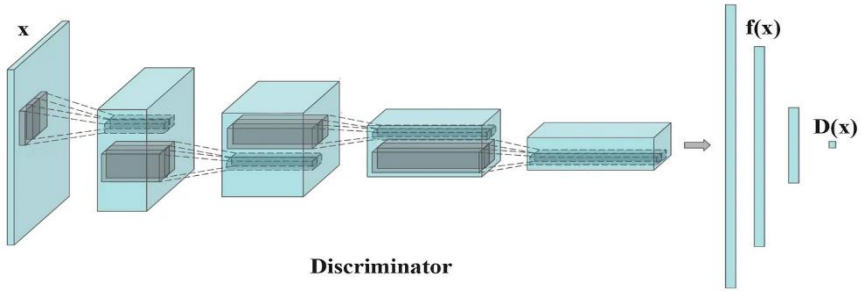


Figure 29 Discriminator Architecture

5.3.3. Design Constraints

- **Structure of data for model:** Changing data to make input image be fit in our project so making data preprocessing to be suitable in pipeline and generating multi types of noise on it is required.
- **Choosing the number of up sampling and down sampling layers is 5 in the generator:** change layers of U-Net to extract important features in image and recover noise.
- **All convolution layers keep the same image size:** in Convolution step images keep the same size does not change as want to filter noise To make down sampling and up sampling be able to extract important features in image and recover noise.

- **Choosing number of training generator and discriminator for each batch in epoch:** In GANs training needs to train generator and discriminator independently to be able to learn so after tuning the best result after train is to train generator 4 times and discriminator 3 times.

5.3.4. Methodologies

5.3.4.1. Noise Generation

The Types of noise handled:

- Block-Pixel noise: each pixel is set to zero with probability 0.05.
- Add Convolve noise: convolved with a Gaussian kernel k, and noise is added.
- Gaussian-Projection noise: add Gaussian noise.
- Salt and Pepper noise.

5.3.4.2. Data Preprocessing

- Resize the image by the longest dimension.
- Choose the type of noise (in the next point) to add noise by equal probability or choose to not add any noise by high probability.
- Pad image to be 512 x512 pixel
- Normalize to be in range 1-0

5.3.4.3. Experience

We tried to implement a denoiser model from Paper[6] but their approach did not work for our problem due to the section of data used and we want full image x-ray.
Try to change the data loader for the model to be suitable for our application and the results illustrated in figure 4.3.4.2.1. This is because model is not able to extract important features from the image so perceptual loss *lvgg* effect on trains to try to reduce it, so no learning and losses are still constant.

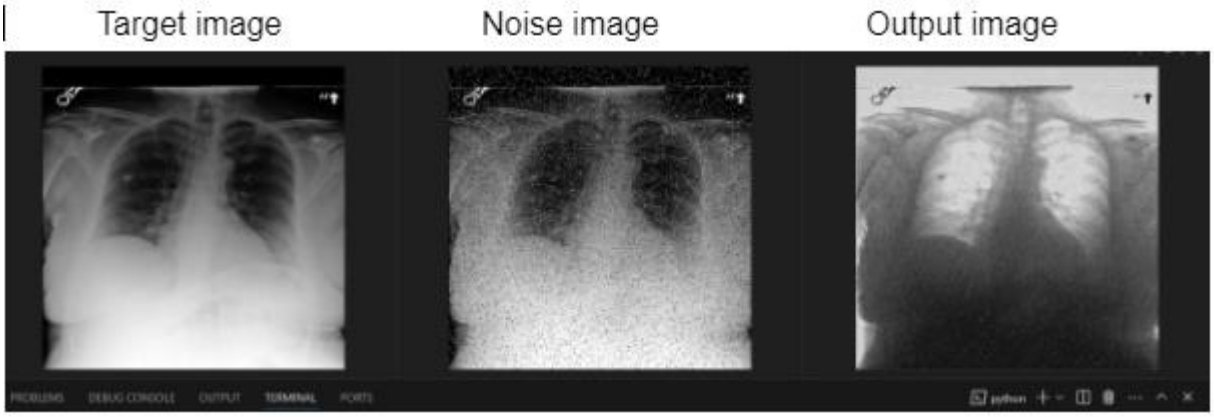


Figure 30 First Trial Denoiser Result

Increasing layer to five (instead of four) down-sampling layers and up-sampling layers and results improved as illustrated these results on 150 images of train and 50 images of test by ssim: 0.69 by 25 epoch.

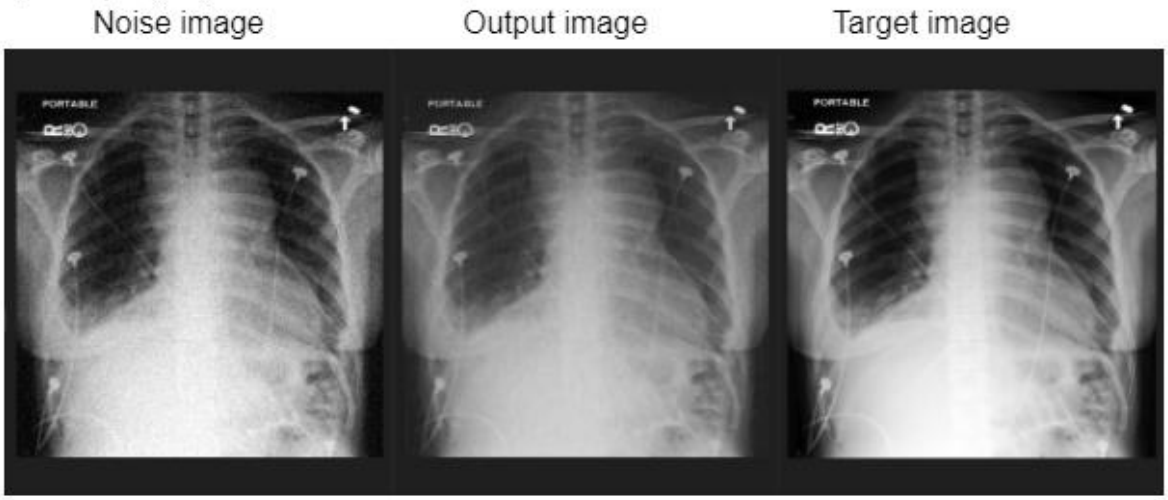


Figure 31 Second Successful Denoiser Result

5.3.4.4. Trainer:

In this section we show the loss functions used for the training procedure.

GAN loss: The discriminator's job remains unchanged which is *ladv*, but the generator is tasked with not only generating adversarial samples but also being near the ground truth output by pixel wise loss an *lmse*. Moreover, perceptual losses *lvgg* are also used to penalize any structure that differs between output and target. Thus, the generator loss is a weighted average of three losses *ltotal*.

- **Adversarial loss:** we compute the adversarial loss by using binary cross entropy loss.

Generator Loss:

- **Perceptual loss:** To allow the generator to retain a visually desirable feature representation, we also use the mean squared error (*MSE*) of features extracted by the pre-trained VGG network to represent a given image. We then define the perceptual loss as the Euclidean distance between the feature representations of a ground truth image and the corresponding denoised image.

$$lvgg = \sum_{i=1}^{W} \sum_{j=1}^{H} (vgg(I^{ND})_{i,j} - vgg(G(I^{LD}))_{i,j})^2$$

Pixel-wise MSE. The pixel-wise MSE loss is calculated as

$$lmse = \sum_{c=1}^{W} \sum_{r=1}^{H} (I_{c,r}^{ND} - G(I^{LD})_{c,r})^2$$

$$-l_{total} = \lambda_g ladv + \lambda_p lmse + \lambda_v lvgg$$

Where $\lambda_g = 20$, $\lambda_p = 1$, $\lambda_v = 100$ by tuning

5.4. Denoiser Machine Learning Approach

5.4.1. 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.

5.4.2. Modular Decomposition

Figure 32 Denoiser Classical Approach Block Diagram

5.4.2.1. 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

5.4.2.2. 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

5.4.2.3. 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)

5.4.2.4. 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

5.4.4. Methodologies

5.4.4.1. 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 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)

5.5. Object Detection Deep Learning Approach

5.5.1. 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

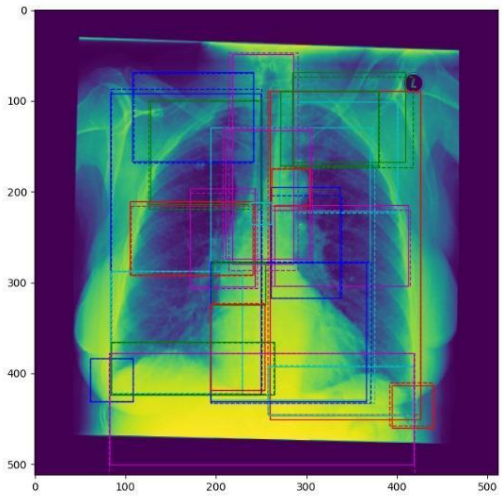


Figure 33 Example for the target 29 regions to detect

5.5.2. 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.

5.4.2.1. Backbone

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

5.4.2.2. 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),)

5.4.2.3. RPN

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

5.4.2.4. 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

5.4.3. 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

5.4.4. Methodologies

5.4.4.1. 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

5.4.4.1. 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

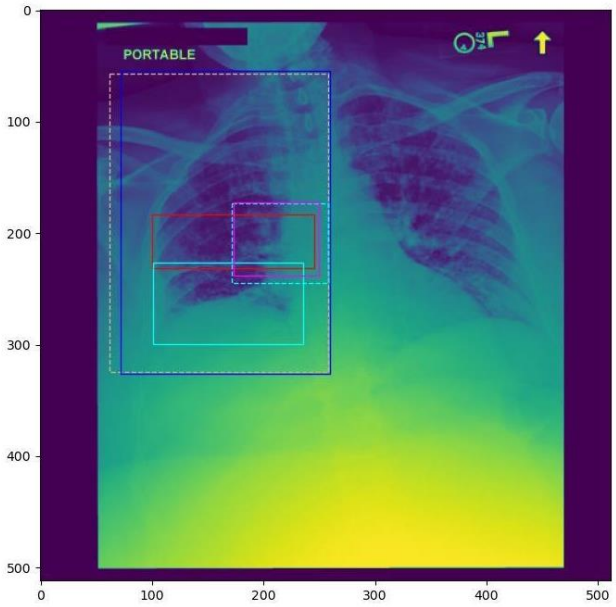


Figure 34 Example for the target 29 regions to detect

5.6. Object Detector Classical Learning Approach

5.6.1. Functional Description

Object detector to detect 29 regions in the chest and extract the features by leveraging a combination of selective search, feature extraction, and machine learning models.

5.6.2. Modular Decomposition

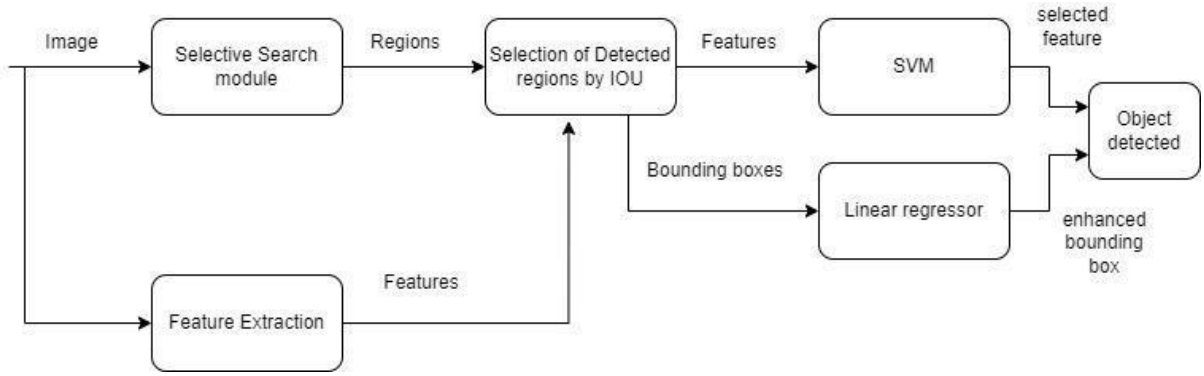


Figure 4.6.2 1 Object Detector ML Block Diagram

Region Proposals: Generate region proposals using selective search. Which implement from scratch.

Selective Search:

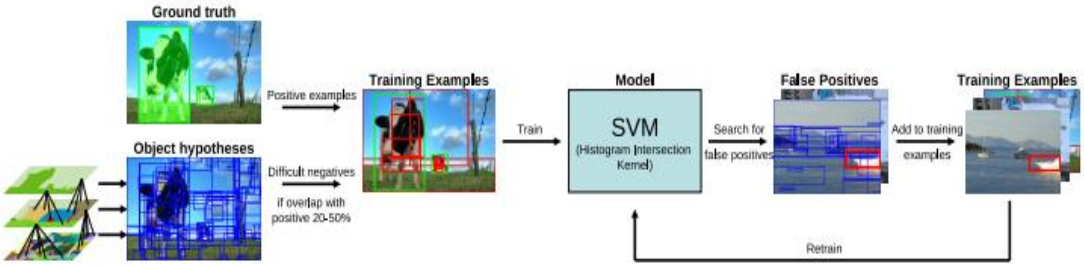


Figure 35 RCNN Selective Search

- **Initial Segmentation**: The algorithm starts by performing an initial segmentation of the image using a method like Felzenszwalb's segmentation algorithm. This divides the image into multiple small regions based on color, texture, and intensity information.
- **Feature Extraction**: For each region obtained from the initial segmentation, various features are extracted, including color histograms, texture descriptors (e.g., Local Binary Patterns), size, and shape features.
- **Region Similarity Calculation**: The algorithm computes the similarity between all pairs of regions.

- **Region Merging**: The most similar pairs of regions are iteratively merged. After each merger, the features of the new region are recalculated, and similarities with other regions are updated. This process continues until a stopping criterion is met, such as a fixed number of regions or a similarity threshold.
- **Bounding Box Generation**: The algorithm outputs the bounding boxes of the merged regions as candidate region proposals for object detection.

Feature Extraction: Extract features from each proposed region using Hog of Haar from scratch to be suitable for medical images.

Classification: Classify each region using a trained SVM.

Bounding Box Regression: Refine the bounding box coordinates using a regression layer.

5.6.3. Design Constraints

Choosing Parameter for Felzenszwalb's segmentation algorithm: scale, sigma, Minimum area size to not detect unwanted region not related to chest.
Choosing feature Extraction: as images are related to the medical field so hog is more appropriate as the ability of HOG to robustly capture edge, shape and texture information makes it applicable to many medical imaging tasks involving detection. Its invariance to geometric transformations and lighting variations helps in analyzing the wide variability encountered in real-world medical images.

5.6.4. Experience

Our trail for training on small data by using but result is not good enough, is illustrated in figure 4.6.4.1 where the green boxes detected and red boxes is targets

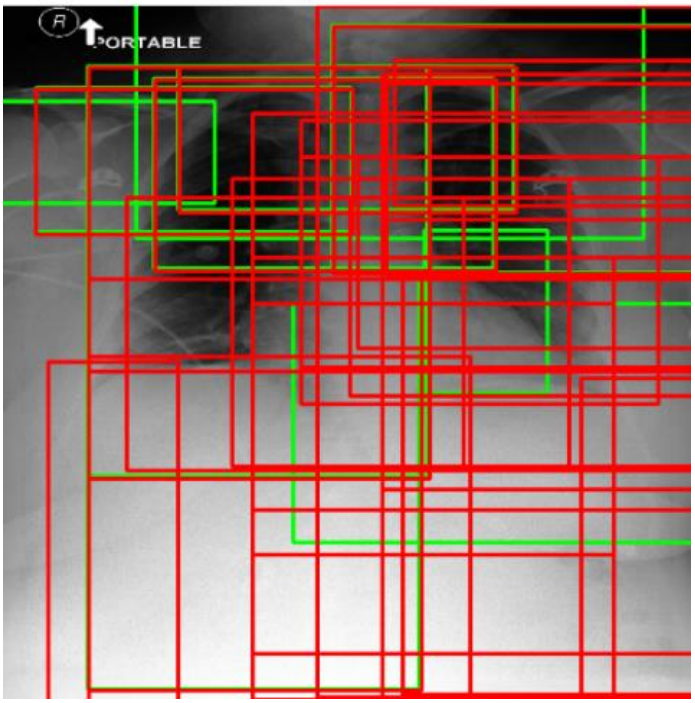


Figure 36 HOG Feature Extractor

Our use of Haar as feature extraction but not give us good results, is illustrated in figure 4.6.4.2 where the green boxes detected and red boxes is targets. After trials, using hog is good as the ability of HOG to robustly capture edge, shape and texture information makes it applicable to many medical imaging tasks involving detection. Its invariance to geometric transformations and lighting variations helps in analyzing the wide variability encountered in real-world medical images.

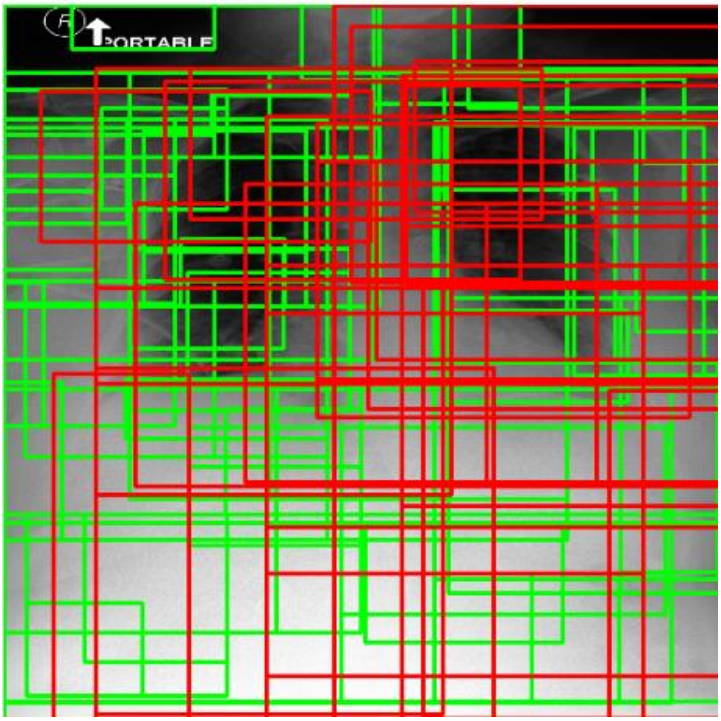


Figure 37 HAAR Feature Extraction

5.7. Classifiers

5.7.1. Functional Description

We have two types of classifiers, each serving a specific function:

- **Region Selection Classifier:** This classifier's primary role is to determine which of the 29 anatomical regions should be included in the report. It identifies and selects relevant regions based on the input data.
- **Abnormal Classifier:** Serving as an intermediate module, the Abnormal Classifier enhances visual features extracted by the object detector. Although lacking definitive ground truth for abnormalities, this model is instrumental in training and fine-tuning the object detector to enhance its ability to detect abnormal visual features.

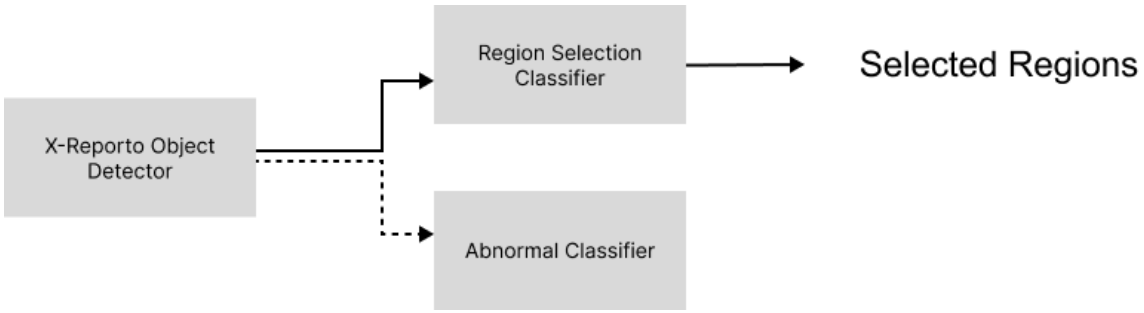


Figure 38 Classifier in X-Reporto System

5.7.2. Modular Decomposition

We used the same architecture for both classifiers because they literally have the same input a lot different to put so we kept fine tuning the lianera layer and nonlinear layers and used our final design for types. As illustrated figure

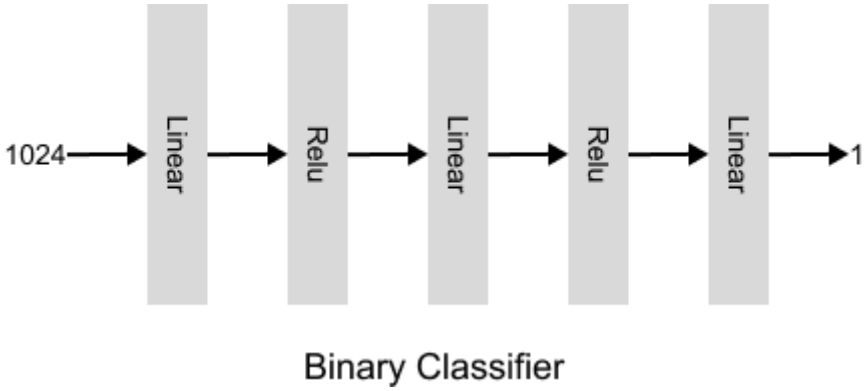


Figure 39 Classifier Architecture

The input of these classifiers is the visual feature for all the 29 anatomical region detect by the object detector the visual feature is of dimension 1024 we used 3 linear layers each one is followed by a Relu layer as an nonlinear activation function

5.7.3. Design Constraints

Since we have imbalance in the data set for both labels of region selection and the abnormal labels we decided to use a weighted loss function Binary cross entropy.
to obtain these weights we have run statistical analysis on the labels on the training dataset as referred to in section 4.3

5.7.4. Methodologies

5.7.4.1. Training

We trained both the Region Selection Classifier and the Abnormal Classifier simultaneously, updating separate sets of weights. Initially, we froze the object detector for several epochs, exclusively updating the weights of both classifiers. Subsequently, we unfroze the object detector, for more epochs with a small learning rate allowing the Abnormal Classifier to propagate its losses to update the object detector's weights, while the losses from the Region Selection Classifier solely update its own weights without backpropagating to the object detector

5.8. Language Model

5.8.1. Functional Description

To generate an accurate report for an X-ray that describes it correctly and includes all medical information and abnormalities present in the chest, we need a language

model capable of producing reports based on visual features obtained from object detection and classifiers. This model should generate human-like text that comprehensively describes all the findings in the X-ray.

We built our model upon gpt2 architecture, we started with gpt2 small basic structure and modified it according to our problem requirements and constraints. We modified the gpt2 attention mechanism and added a new attention mechanism that incorporates visual feature information along with the attention of generated tokens to generate output scores of attentions.

Then we added an encoder model to apply transformation from visual features generated by the object detector to text features that can be understood by language models and act as intermediate information carrying medical information and abnormality exists in each region in the x-ray chest.

Finally, we added gradient checkpointing for each block so that we could fix the memory requirement of the model while training so that it can train the model with higher batch sizes and fit our requirement so that we can train locally or on a server with high resources and gain the best performance of the model.

5.8.2. Modular Decomposition

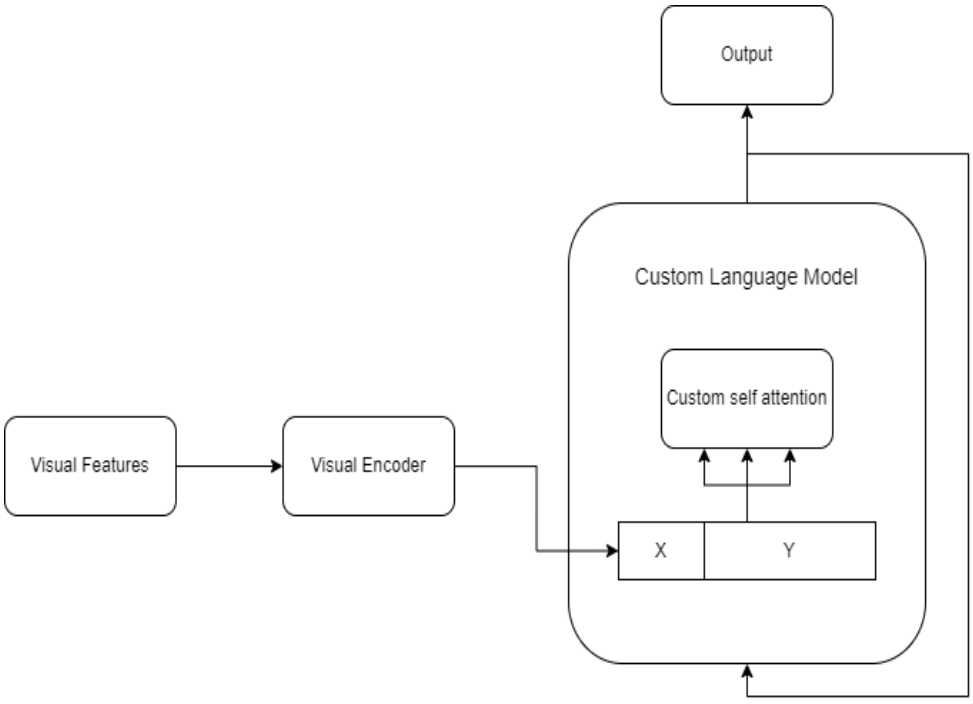


Figure 40 Language Model Block Diagram

In Our model we have added new blocks to fix our problems on how to make transformer decoder works on visual features

1. Visual Encoder model
2. Custom Self attention block

5.8.2.1. Visual Encoder

Image Transformation Encoder used for transforming visual features from image space to text space features can be understood by language model to generate text from.We used two linear layers where the input is a visual feature vector of length 1024 and the output is also a vector of the same length.

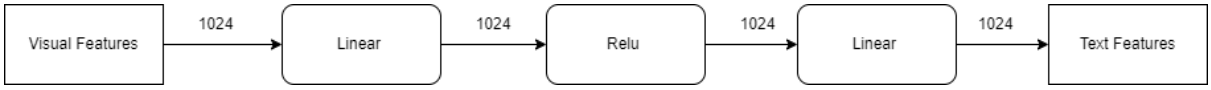


Figure 41 Visual Encoder Block Diagram

5.8.2.2. Custom Multi-Head Self-attention

In this step, the model uses masked multi-head self-attention mechanisms to focus on different parts of the input sequence simultaneously. The masking ensures that the prediction for a given position only depends on the known outputs at prior positions, maintaining the autoregressive property.

Normal self-attention equation is $SA(Y) = softmax\left((YW_q)(YW_k)^T\right)(YW_v)$

where Y represents the token embeddings and Wq, Wk,
Wv are the query, key, and value projection parameters.
To condition the language model on the region's visual features, we use pseudo-self-attention to directly inject the region's visual features into the self-attention of the model,

Pseudo self-attention equation is $PSA(Y) = softmax\left((YW_q)\begin{bmatrix} XU_k \\ YWk \end{bmatrix}^T\right)(YW_v)$

where X represents the region's visual features, and Uk and Uv are the corresponding (newly initialized) key and value projection parameters. This allows text generation conditioned on both previous tokens and region visual features.

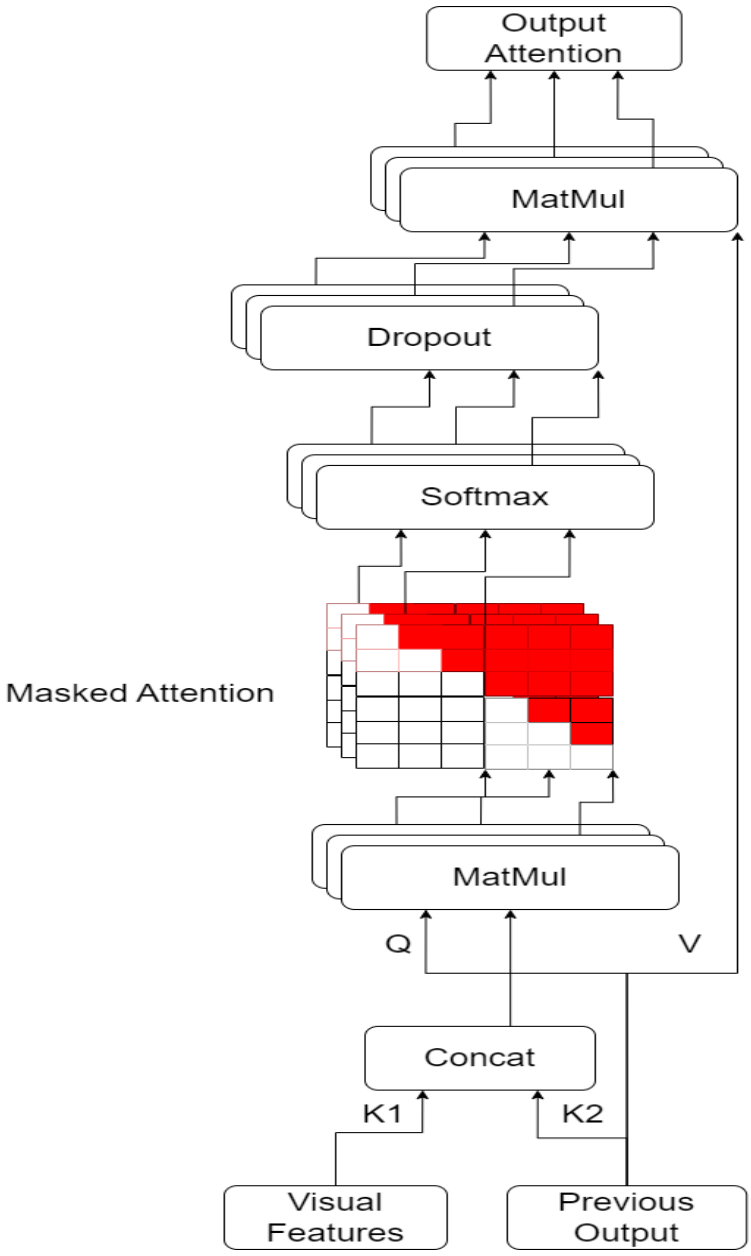


Figure 42 Custom Self Attention

5.8.3. Design Constraints

Visual Features Encoding: Our first problem was how to integrate visual features generated by the object detector and classifier into our language model gpt2 and maintain consistent features that both models can learn correctly so that object detection can correctly use these features in generating bounding boxes and at the same time incorporate medical information and abnormality in it, and language

model can understand these features and get benefits from it then generate auto regressively output sentence that truly describes regions.

Our solution with Image transformation uses two linear layers with relu between them, these layers should be responsible for transforming visual features and medical information into text representation as an encoding layer that language model understands and can use to generate sentences using a decoder block.

Visual Features Integration: The second constraint was how to integrate information of visual features in the attention mechanism so as to generate the next tokens from previous tokens and visual features and calculate the attention score.
Our solution was using a pseudo-self-attention mechanism that incorporates visual information as key input concatenated with the key of token information and then applies a normal self-attention mechanism.

Pseudo self-attention equation is $PSA(Y) = softmax\left((YW_q)\begin{bmatrix} XU_k \\ YWk \end{bmatrix}^T\right)(YW_v)$.

This reduces model computation instead of applying two attention blocks, one for previous tokens and one for visual features, and at the same time applies correct conditioning on these inputs.

Model Size: The third constraint was the model size, what number of layers to be used that can achieve the best performance, not overfitted, and can be trainable using not many resources.
We used 12 layers of attention block in the above diagrams. these helped us to train our full pipeline with minimal resources we have. The size of the model is around 170M parameters and 700 MB in memory.

5.8.4. Methodologies

5.8.4.1. Training

To train a full pipeline of object detectors, classifiers, and language model with at least one example “batch size = 1”, and one example contains 29 sentences which correspond to that input to language model is 29 examples which are very huge and can not fit in our memory.
We fixed this problem by applying gradient checkpoint which is a technique used in deep learning to reduce memory usage during training of neural networks, especially those with large memory requirements. Normally, deep neural networks store intermediate activations (tensors) during forward propagation to use during backpropagation for computing gradients. This storage can become quite memory-intensive, particularly in models with many layers or parameters which occurs in our case in gpt2.
Gradient checkpointing addresses this issue by selectively storing only a subset of intermediate activations, known as checkpoints, rather than all of them. During

backpropagation, the network can then recompute the omitted activations dynamically as needed, trading off computation for reduced memory consumption.
We applied Gradient checkpointing on each layer block of 12 layers of gpt2. so that storing only the inputs and outputs of each block. With this enhancement, we reduced the memory requirements of the gpt2 model by 1/12 factor of memory requirements of gpt2 without gradient checkpointing, but we doubled the training time of the model as we nearly applied forward pass twice.
We were able to apply batch size = 4 for the object detector and batch size = 128 for language model sentences without any accumulating gradients as in object detector. trained for **2 epochs** with a very small **learning rate** equals 10^{-5} We used cross-entropy loss as our loss function and Adam as our optimizer.
Not only was the language model trained, but the full pipeline was trained. We applied a weight mechanism for each model while training where the object detector weight equals 1, classifier weight equals 5, and language model equals 2. These parameters needed to focus more on the selection of regions and abnormality in features then focus on the language model and finally object detector where its weights were trained 3 times.

5.8.4.2. Post Processing

After Obtaining the report results we found redundant repeats sentences in the description for each region. After report generation, we have k selected regions have k sentences that describe the finding in each anatomical regions of chest that detected by object detector, since regions overlaps with each other, we found out that model outputs nearly same finding of overlapped regions therefore we cannot just concatenate sentences to form report but we should select unique sentences to form final report. We tried different solutions to fix this problem either software of ai solutions

Software Solution: We tried to filter duplicate sentences by using string matching to remove copies of these sentences. but it was not very good as sentences can differ in a small number of characters or words but have the same meaning or can be totally different words but have exactly the same meaning.

AI Solution: We tried to use word embeddings to filter sentences as and remove duplicates. as similar words in meaning will have high score “small distance between word embeddings vectors”. We used the Roberta Language model to decide if two sentences are equivalent or not by giving it two sentences and output -1 to 1 score for equivalence. we then check if score > 0.9 that means two sentences has same meaning then finally we remove shorter sentences to keep as much as possible information in findings in report.

5.8.4.2. Model Hallucination

Model hallucination happens when the language model generates output that is nonsensical, irrelevant, or factually incorrect. because of producing text that deviates from the input context, resulting in meaningless sentences or random sequences of characters. For instance, while generating a medical report, the model might output medically irrelevant information, random jargon, or strings of unrelated words and symbols. such sentences must be removed from the final report.

We fixed this problem by making Regex that searches for non-meaning words or patterns of repeated characters or longer words than longest English words. then we remove this sentence from the model.

ex: `r"(.)\1{2,}" # match words have repeated chars more than 3`

5.8.5. Report Generation

During the prediction phase, the language model generates the report by sequentially predicting the next word based on the visual features and previously generated words. The process begins with the encoded visual features, which provide the context for the report. The model utilizes the word embeddings and positional encodings to maintain an understanding of the medical terminology and the structure of the report. As each word is generated, it is fed back into the model to predict the next word, ensuring coherence and relevance throughout the report. The masked multi-head self-attention mechanism allows the model to focus on different parts of the input sequence, capturing intricate relationships between the visual features and the generated text. This iterative process continues until a complete, detailed, and accurate medical report is produced, describing the findings in the X-ray, including any abnormalities and relevant medical information. The use of softmax ensures that the most probable words are selected at each step, resulting in a human-like, comprehensive report.

To enhance the quality of the generated medical report, we used different decoding strategies, such as greedy search and beam search. Greedy search selects the most probable word at each step, aiming for immediate local optimization, which can lead to less coherent overall reports due to potentially missing better alternatives. Beam search, on the other hand, explores multiple possible word sequences simultaneously by keeping track of the top-k most likely sequences at each step. This method allows the model to consider a broader context and make more informed decisions, resulting in more coherent and accurate reports. Beam search is particularly beneficial for medical report generation as it helps in capturing complex relationships and dependencies within the visual features and the medical terminology, ensuring that the final report is both comprehensive and precise.so we

used beam search with k = 8 which was quite enough to generate complex sentence correctly and does not hurt much running time of prediction.

5.9. Template Based Report Generator

5.9.1. Functional Description

The template-based report generator Module main functionality is to take the chest x-ray as input and generate a detailed, template-based report that summarizes findings in the x-ray. Advanced features are incorporated into these reports to provide more insights to radiologists, making the reports not only comprehensive but also informative. Additionally, the module outputs heatmaps that assist in reasoning and explainable classification by localizing areas of interest for each finding, thereby enhancing the radiologist's ability to interpret the results accurately.

5.9.2. Modular Decomposition

The Template-based report generator is mainly divided into 3 sub-modules discussed in detail in the following subsections.

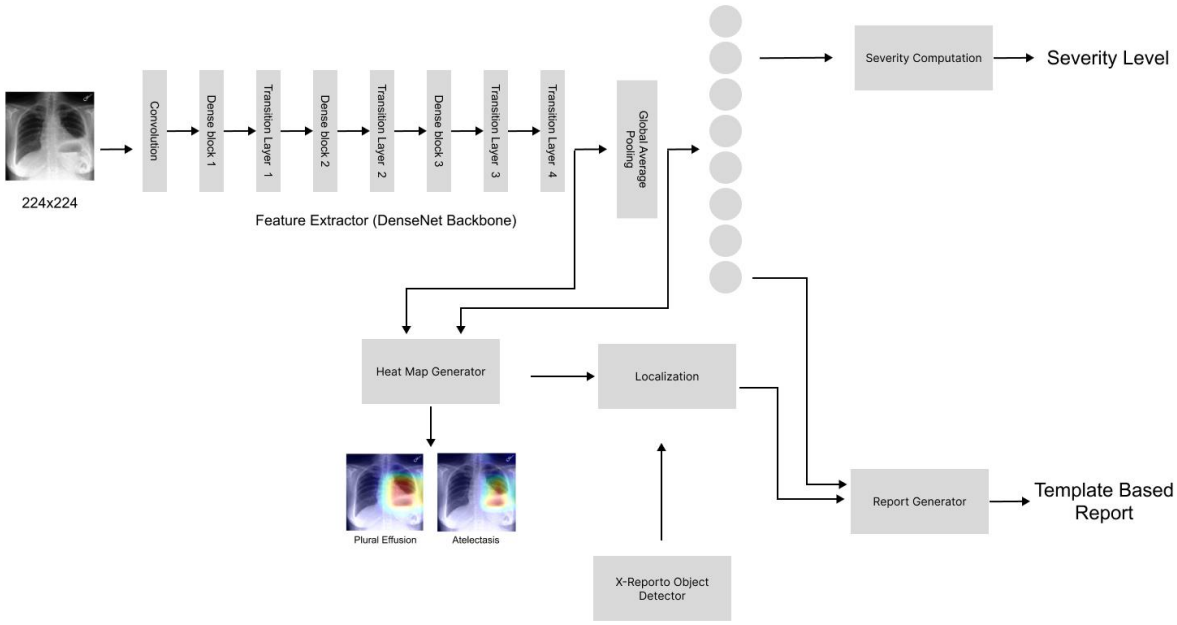


Figure 43 Template-based Report Generator Block Diagram

5.9.2.1. Findings Classifier

The main function of this submodule is to perform multilabel classification on input chest X-rays, focusing on eight specific findings identified from our dataset. The

problem formulation for this module is centered around the multilabel classification problem.
Given the state-of-the-art performance of CNNs in image classification tasks, we chose a CNN backbone for feature extraction. Initially considering popular architectures like AlexNet, GoogleNet, VGGNet, and ResNet as proposed by Wang X et al [8], we found DenseNet-121, proposed by Huang G [9], to be the most effective feature extractor. DenseNet-121 utilizes both forward and backward skip connections, addressing the vanishing gradient problem by facilitating direct gradient flow between all layers, thereby enhancing information propagation.
The output of DenseNet-121's backbone is a 7x7x1024 feature map. To accurately localize activations within the feature map, we adopted Global Average Pooling (GAP) based on the recommendation by Zhou B et al.[10] Unlike Global Max Pooling (GMP), which limits activation localization to specific points within the object boundaries, GAP provides a comprehensive understanding of the spatial extent of identified findings.
The output of the GAP layer is a flattened vector of size 1024, which serves as input to the final fully connected (FC) layer consisting of 8 neurons. Sigmoid activation is applied to the outputs of these neurons, leveraging the probability of occurrence of each finding in the provided chest X-ray.

5.9.2.2. Heatmap Generator

We generate feature maps for each class independently, following the methodology proposed by [10]. With 8 labels to classify, our goal is to highlight activations specific to each finding, guiding radiologists' attention to relevant regions. Similar to [10], we multiply the weights of the fully connected (FC) layer with the activations from the last dense block in our DenseNet-121 feature extractor, as our design omits the transition layer found in their module.To ensure only positive values contribute to the heatmap, we apply the ReLU function. We conclude with a final FC layer containing 8 neurons. Sigmoid activation is applied to these outputs. Subsequently, we normalize the feature map values to a range of 0 to 1, ensuring scale consistency and enhancing contrast for better insights given to the radiologist.

Following the procedure described above, we generate 7x7 feature maps for each label. During inference, these feature maps are resized to 224x224 pixels using the same resizing method applied to the original image, ensuring aspect ratio preservation by padding along the smaller dimension.
Next, we apply the applyColorMap function implemented in OpenCV, which maps higher values to warmer colors (such as red and yellow) and lower values to cooler colors (such as blue and green) in the heatmap.
Finally, we blend the heatmap with the original image using a weighted sum, expressed by the equation:

$$blended\_image = image\_resized \times 1 + heatmap\_resized \times 0.35 + 0$$

5.9.2.3. Report Generator

Our Main Focus was to not just classify the image according to the class we previously defined we need to produce a template-based report. Our first version of the report generator, as shown in figure 5.9.2.3.1, was just taking the output of the sigmoid functions. According to predefined thresholds obtained during model training we state whether the patient has this finding or not.But we need to give insight to the radiologist and he has the final word so it was essential to add the model confidence to the report so that we don't mislead the radiologist.This info was very essential because we already has imbalanced data so choosing the optimal threshold based on the validation data set will make the model unable to generalize.

```
The patient does not have Atelectasis with a confidence of 26.28%.
The patient does not have Cardiomegaly with a confidence of 5.78%.
The patient has Edema with a confidence of 36.51%.
The patient has Lung Opacity with a confidence of 27.50%.
The patient has No Finding with a confidence of 43.78%.
The patient does not have Pleural Effusion with a confidence of 7.62%.
The patient does not have Pneumonia with a confidence of 12.35%.
The patient does not have Support Devices with a confidence of 3.95%.
```

Figure 5.9.2.3 1 First version of the template-based report with findings confidence

Moreover, we need to make our report informative, we need not to just give the doctor probability of a finding, instead we need to help him in the location of the findings in case the heatmap isn't available. So we used the bounding boxes generated by the X-Report object-detector explained in section(5.5) our product is as follows we have only the resized 224x224 feature map without being blended which is in RGB so we compute the mean over the 3 channel with equal weights.
Then we take as input the Bounding boxes from the object detector but these measurements are relative to the 512x512 image outputted from the denoiser explained in section(5.1) so we need to rescale them to be relative to the 224x224 heat map we have. Using the coordinates of the fixed bounding boxes we get mean of activations for each regions and choose the one with the max activation , but following this procedure we overcome the problem of overlapped regions in case we just check where pixels are inside region boundary
To enhance the informativeness of our reports, we go beyond providing the probability of findings to assist doctors in pinpointing their locations, especially when

heatmaps are unavailable. We achieve this using bounding boxes generated by the X-Report object detector (discussed in section 5.5) and we got our final template-based report as illustrated in figure 5.9.2.3 2. Our procedure is as follows:
We start with the resized 224x224 RGB feature map, without blending, and compute the mean activation over the three channels equally weighted. The bounding boxes provided by the object detector are relative to the 512x512 image output from the denoiser (explained in section 5.1). Thus, we rescale these coordinates to be relative to our 224x224 heatmap, expressed by the equation:

$$box_{relative\ to\ 224} = box_{relative\ to\ 512} * \frac{224}{512}$$

From the fixed bounding box coordinates, we compute the mean activations for each region and identify the region with the maximum activation. This method initially involved checking the most active pixel in the heatmap and determining its corresponding boundary. However, this approach proved inadequate due to potential overlaps between regions.

```
The patient does not have Atelectasis with a confidence of 26.28%.
The patient does not have Cardiomegaly with a confidence of 5.78%.
The patient has Edema with a confidence of 36.51%.
The findings are primarily located in the trachea.
The patient has Lung Opacity with a confidence of 27.50%.
The findings are primarily located in the right apical zone.
The patient has No Finding with a confidence of 43.78%.
The findings are primarily located in the right costophrenic angle.
The patient does not have Pleural Effusion with a confidence of 7.62%.
The patient does not have Pneumonia with a confidence of 12.35%.
The patient does not have Support Devices with a confidence of 3.95%.
```

Figure 5.9.2.3 2 Second version of the template-based report with findings confidence

5.9.3. Design Constraints

5.9.3.1. Network Design

We opted to use DenseNet-121 as our backbone feature extractor instead of ResNet-50, which we found less effective due to the vanishing gradient problem. With our training relying heavily on gradient descent optimization, the inability of gradients to propagate effectively was hindering our progress, resulting in stock losses. DenseNet-121's bidirectional skip connections were pivotal for addressing our needs, particularly in handling images with multiple findings, requiring decisions based on comprehensive feature complexities extracted through convolutional layers. This architecture proved effective during training, demonstrated by a gradual decrease in losses, indicating the network's improved learning capability compared to our unsuccessful trial with ResNet.
In contrast to the approach proposed by [10], which utilized a transition layer before Global Average Pooling (GAP), we determined that such a layer was unnecessary for our setup. This decision was influenced by the unique characteristics of DenseNet-121, which incorporates bidirectional transition layers designed to enhance the discriminative power of features before applying GAP. Thus, our choice

of DenseNet-121 as the backbone aligns with its intended design for feature extraction.
We used FC Layer with 8 outputs followed by a sigmoid since our problem is multi-label classification, so we need probability for each finding independently.

5.9.3.2. Accumulative Learning

Due to the limitation of the training resources, training this network with large batch sizes was a limitation for us, so we followed the technique of accumulating learning that increases the training time but reduces the resources usage making it double to train such a network on our limited resources. Following this technique, we were able to train this classifier in Colab [9] free trial plan with 15 GPU VRAM

5.9.4. Methodologies

5.9.4.1. Data Preprocessing

We used the same tarin-validation-test split used in X-Reporto trainer to prevent data leakage, especially when we decided to integrate the object detector with the report generator submodule. The first step involved running Exploratory Data Analysis (EDA) or on the data labels we obtained, as explained in the dataset section. During the analysis, we discovered the problem of null and -1 labels in the dataset. As shown below the histogram for the training subset in figure 5.9.4.1. 1. There are a lot of null values compared to the -1 or 0 labels.

We proposed several approaches with dealing with -1 labels:

1. **Dropping Examples with -1 Labels:** Initially we considered dropping examples with uncertain labels (-1). However, since this is a multi-label problem, an example might have a -1 for some labels while having 0 or 1 for others. Dropping these examples resulted in a significant loss of data, leaving us with too little to learn from.
2. **Converting -1 to 0:** Our next approach was to convert -1 labels to 0, marking them as not. However, this led to confusion, as it contradicted our understanding. For instance, if we had an example, we learned to be true, seeing a similar one marked as false could be misleading. This inconsistency affected the effectiveness of our model training.
3. **Converting -1 to 1:** Our final approach and the most logical one was to convert the -1 labels to 1 label marking them as they need attention from the model. This reframed our problem from deciding whether a label is present to determining whether the model needs to focus on that label or not. We got the histogram after aping the U-Ones-Approach as illustrated in figure 4.3.3.6.

[9] https://colab.research.google.com/

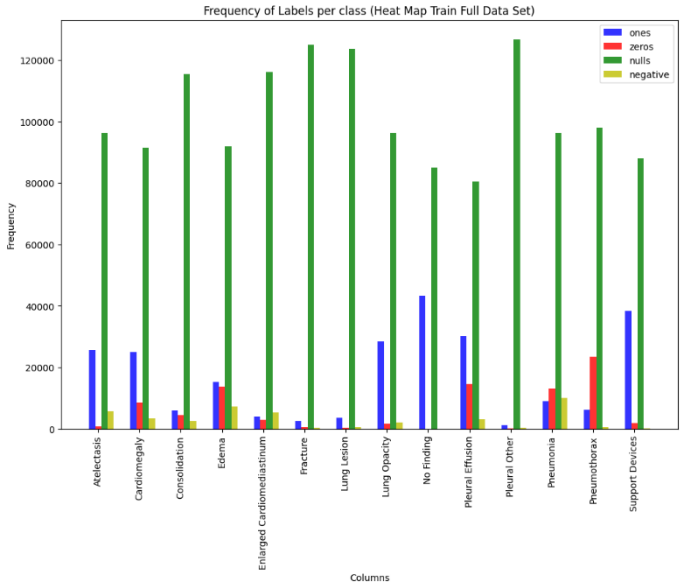


Figure 44 Histogram of the training split (4 Labels)

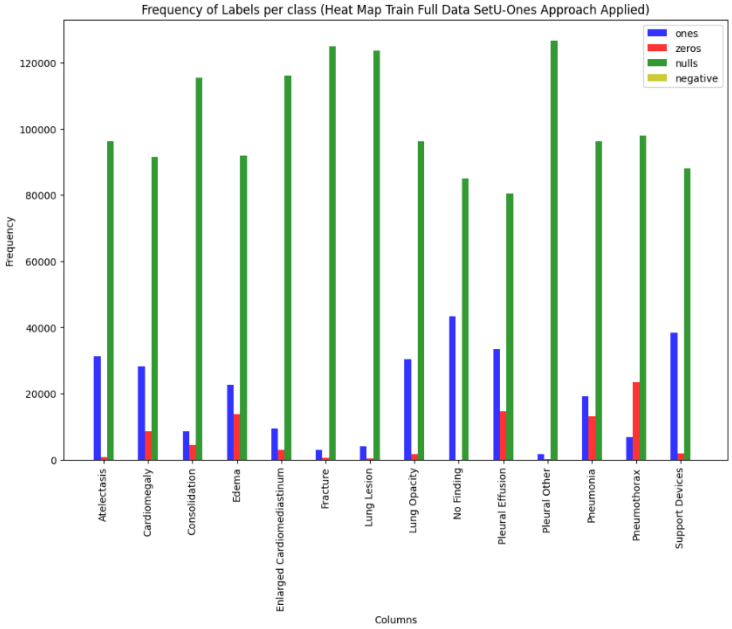


Figure 45 Histogram of the training split with -1 converted to 1

After fixing the problem of the -1 labels, it was necessary to think about how we will deal with the null labels. In addition to the problem of data imbalance, we have a huge number of nulls compared to the 1 and even the 0 labels. We need to balance our data,

We proposed several approaches with dealing with Nan labels:

1. **Nan Balancing Approach:** We tried to propose a new approach to deal with this imbalance, so we decided to convert some null labels to 0 till we reach an equal number of zeros and ones for each label. So, we got balanced 0,1 labels but still a lot of nulls as shown in figure. During the training we applied a mask on the loss computation for the unconverted null labels, demonstrating our ignorance to the decision taken by the model on such labels. Unfortunately, we found out that this approach causes oscillations, and they couldn't achieve model convergence
2. **Converting Nan to 0:** Following the same procedure we used in dealing with -1, we decided to convert the null labels to zero. This decision is taken based on the evidence that not mentioning means not occurring because since the radiologist hasn't mentioned that this finding is neither positive nor negative, it is sufficient to label it as negative. This is due to the nature of such a problem, while diagnosing the X-Ray the radiologist checks every finding not just focus on some findings and ignoring the presence of others it is his responsibility.

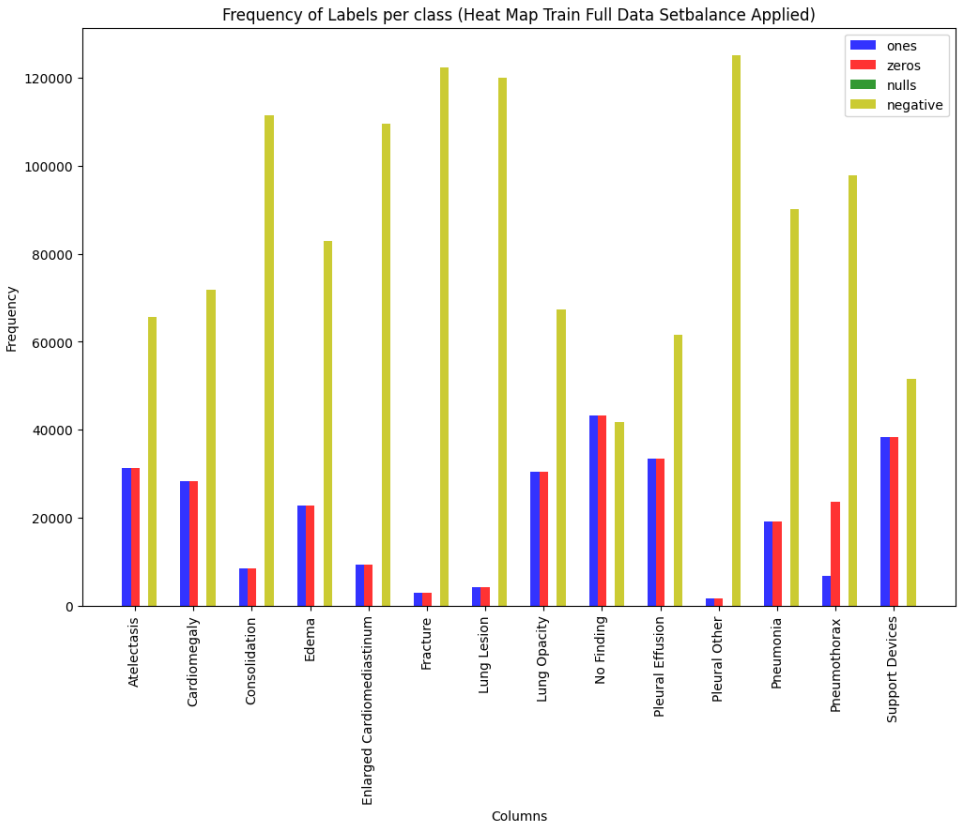


Figure 46 Histogram of the training subset with balanced 0 and 1 labels

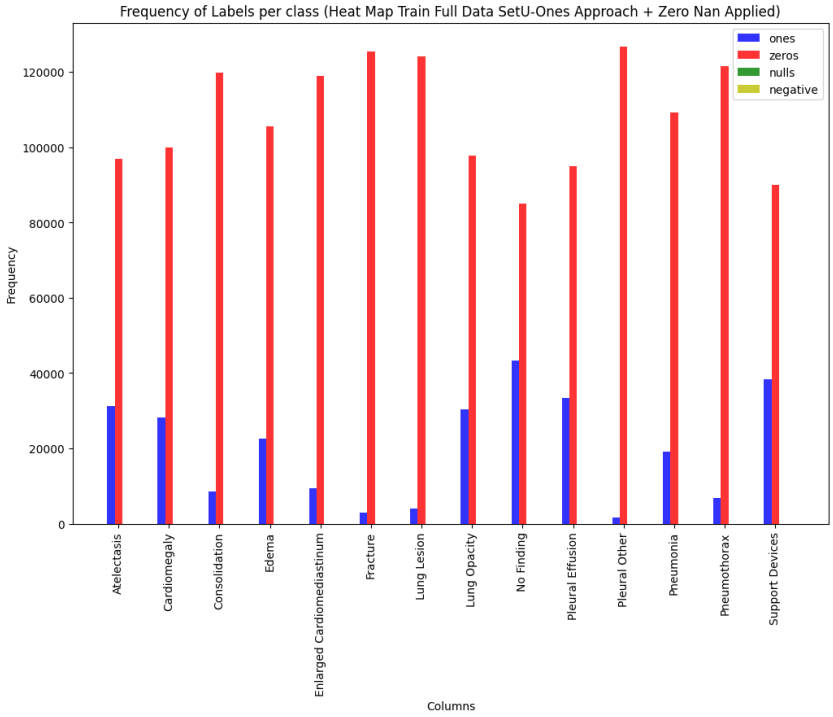


Figure 47 Histogram of the training subset with 0 and 1 labels

The final preprocessed version of the dataset is shown in figure 5.9.4.1 4. We converted the -1 to be 1 and Nan to be 0 resulting in just 0,1 labels. This data has a severe class imbalance. The dataset lacks positive labels using such data will result in a biased classifier.

We proposed several approaches with dealing with Nan labels:

1. **Dropping examples:** We tried to drop the training examples with the number of 1 label below a certain threshold, but this resulted in the issue when we tried to drop the examples with -1 labels. No examples left to learn from due to the multi-label nature of the problem.
2. **Data Augmentation:** We applied several Data Augmentation techniques such as applying random Gaussian noise on the training examples. In addition, we applied random rotation for the image from -2 degrees and 2 degrees. Since data augmentation is a commonly used technique to address the problem of data imbalance. Although the data approach of augmenting data to address the problem of class imbalance is a common technique for dealing with class imbalance since here, we have multi-labels augmenting increase examples of 1 in certain label results in increasing more examples in Nans in other labels since no of Nans are a lot no patient doesn't have nan in its gold labels.
3. **Drop Labels:** Some Labels occur rarely such as:
    i. Consolidation
    ii. Enlarged Cardiomediastinum

iii. Fracture
iv. Lung Lesion
v. Pleural Other
vi. Pneumothorax

So, we decided to drop out these examples since there is actually nothing to learn from and using them in the training will only increase the complexity of the problem making it harder for the model to learn.

4. **Weighted Loss:** We decided to use weight loss to solve this issue. We computed weight per each label from the training data set computed as shown in the following equation. This was the most effective solution that worked with our trails

$$weight_{c_i} = \frac{no\ of\ zeros\ in\ c_i}{no\ of\ ones\ in\ c_i}$$

5.9.4.2. Training

In our training setup we used Adam optimizer to update the model's parameters. The optimizer is initialized with a learning rate of 5.12e-05, and beta values of 0.9 and 0.999 for the first and second moment estimates, Additionally, we employ a learning rate scheduler to further refine the training process. Specifically, we use the StepLR scheduler, which reduces the learning rate by a factor of 0.9 every epoch.

Regarding the loss function for the optimizer, we used Binary cross entropy with mean as the reduction function. These losses are used to update every 2 successive batches since batch is 32 and the effective batch size is 64. Training using the whole dataset with batch size 32 we get the total no of batches for one training epoch 4011. After finishing a complete epoch, we run another epoch on the validation dataset to keep track of overfitting. We first trained the network for 5 epochs, but it didn't saturate the losses in a continuous decrease so we realized we needed to add more epochs, but we were afraid of the overfitting, so we kept track of the validation losses after each complete epoch.

So continued training for 5 more epochs but the best model was saved after the 8^{th} epoch. After that we saw an increase in the validation losses. We suggest that the model overfit after completing 8 epochs due to the usage of GAP which is robust to overfitting which was proposed by [11] a technique for regularizing training.

5.10. Backend

5.10.1. Functional Description

In this section we present the backend System OvervieThe backend system is designed to manage the complete workflow of medical image reporting, incorporating a REST API, an AI server, and a database. This integrated system ensures seamless communication between various components, facilitating efficient data processing, report generation, and data storage.

REST API:

- Endpoint Management: The REST API provides various endpoints for uploading medical images, retrieving generated reports, and managing user authentication.
- Request Handling: It handles incoming requests, validates the data, and directs them to the appropriate services, such as the AI server or the database.
- Response Generation: After processing, the REST API sends responses back to the client, including the status of the request and any requested data.

AI Server:

- Report Generation: The AI server is responsible for processing the uploaded medical images using object detection and classification models to extract visual features. These features are forwarded to the language model to generate the actual report using a beam search algorithm.
- Heatmap Generation: The AI server is responsible for processing the uploaded medical images using a heatmap classifier, generating predictions of possible diseases, and heatmap for each positive disease to determine the location of the disease.

DataBase:

- Data Storage: We built a database system for medical application that serves doctors , employees, studies, and results, where we store results of ai models and written reports , tracking activity of doctor and new studies added.
- Query Management: It supports complex queries to retrieve specific reports, study data, or ai results. This facilitates efficient data management and quick access to necessary information.

5.10.2. Modular Decomposition

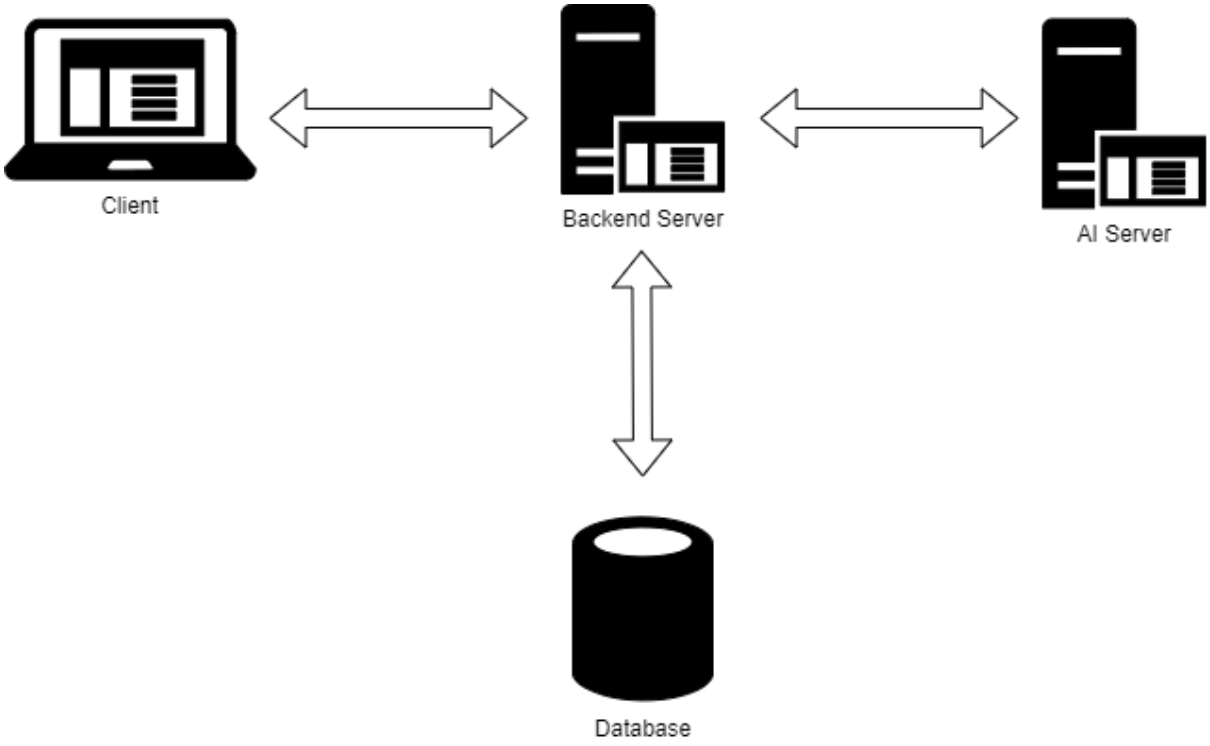


Figure 48 System Design

Our system is designed with three main components: the backend server, the AI server, and the database server. Here's an overview of how the system works:

1. Backend Server: The backend server acts as an intermediary, handling communication between the client and the other servers. It uses REST APIs to process client requests, perform CRUD operations, execute complex queries, and manage AI model execution.
2. Client Communication: Clients interact with the backend server through REST API requests. These requests can involve creating, reading, updating, or deleting data (CRUD operations), as well as executing more complex queries and invoking AI models.
3. Database Server: The backend server communicates with the database server to perform various operations such as inserting, deleting, updating, and retrieving data. The database server handles all relations in the database, ensuring data integrity and efficient query execution.
4. AI Server: Medical images are sent from the backend server to the AI server via REST APIs. The AI server processes these images to generate diagnostic

reports, labels, and corresponding heatmaps. The results from the AI server are then stored back in the database.

5. Response to Client: Finally, the backend server compiles the results generated by the AI server and the database queries, and sends a comprehensive response back to the client. This response includes AI-generated results such as diagnostic reports and heatmaps.

This system design ensures efficient and accurate processing of client requests, seamless interaction between the different servers, and the generation of valuable medical insights through AI.

5.10.3. Design Constraints

5.10.3.1. AI Server

We built two inference classes for our two full models x-reporto and heatmap, then we built our ai server with fastapi.

we mainly have 3 endpoints:

1. x-reporto/report/: returns bounding boxes - boxes sentences - full report
2. x-reporto/denoise/: returns denoised image of GANs model
3. heatmap/generate/: returns prediction of disease - heatmaps - template base report

5.10.3.2. Database Design

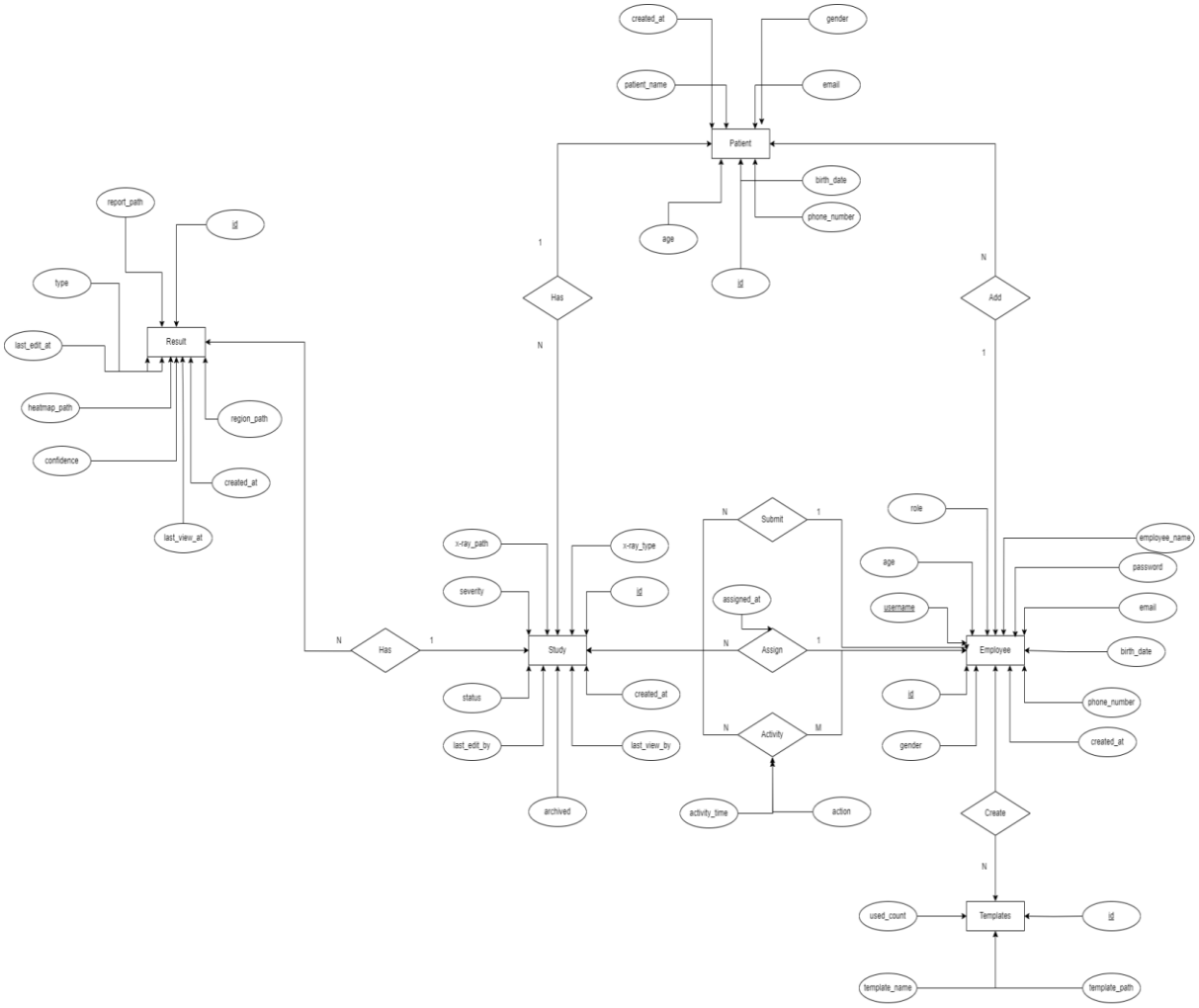


Figure 49 Database ER Diagram

- **Entities**

Patient Entity: The Patient entity represents individuals who undergo medical examinations and studies. This table includes attributes such as patient_id, name, birth_date, gender, contact_information, and other relevant demographic information. The patient_id serves as the primary key, ensuring each patient is uniquely identifiable. This entity is essential for maintaining patient records and linking them to their respective medical studies and results.

Employee Entity: The Employee entity represents the staff members within the medical facility, including doctors, nurses, and administrative personnel. Key attributes in this table include employee_id, name, role, department,

contact_information, and other professional details. The employee_id is the primary key, ensuring unique identification of each employee. This entity is crucial for tracking which employees create patient records, conduct studies, and manage report templates. Additionally, employees with the role of "doctor" are responsible for creating and managing templates, as well as being assigned to specific studies.

Study Entity: The Study entity represents the medical studies or examinations conducted on patients. Attributes include study_id, patient_id (foreign key referencing the Patient entity), employee_id (foreign key referencing the Employee entity), doctor_id (foreign key referencing the Employee entity), severity, and notes. The study_id is the primary key, uniquely identifying each study. This entity is vital for recording and managing the various medical examinations each patient undergoes, and it establishes a direct relationship between patients and the studies conducted on them. Additionally, an employee (doctor) is assigned to each study, indicating the responsible medical professional.

Result Entity: The Result entity contains the outcomes or findings from each medical study. Attributes in this table include result_id, study_id (foreign key referencing the Study entity), type, date, and paths (such as images or documents). The result_id is the primary key, ensuring each result is uniquely identifiable. This entity is essential for documenting the detailed findings of each study, allowing for thorough analysis and review by medical professionals. The type is identifying what is the result is generative ai, template based, or custom based by doctor. The direct linkage to the Study entity ensures that each result is associated with a specific study.

Template Entity: The Template entity is used for creating standardized report formats that doctors can use when generating study results. Attributes include template_id, employee_id (foreign key referencing the Employee entity), template_name, content, and date_created. The template_id serves as the primary key, ensuring unique identification of each template. This entity allows doctors to create and manage templates, facilitating consistency and efficiency in report generation. Each template is linked to the doctor who created it, promoting accountability and ease of access.

- ## Relationships

Employee Creates Patients: This relationship indicates that employees, such as administrative staff or medical personnel, are responsible for creating

and managing patient records. This ensures that patient data is accurately recorded and maintained.

Employee Creates Studies: This relationship highlights that employees initiate and document medical studies for patients. It underscores the role of employees in managing the entire process of patient examination.

Patient Has Studies: This relationship establishes that each patient can have multiple studies associated with them. It is fundamental for tracking the medical history and examinations of each patient.

Study Has Results: This relationship links each study to its corresponding results, ensuring that the findings from each examination are properly recorded and associated with the correct study.

Employee (Doctor) Creates and Has Templates: This relationship signifies that doctors create and manage report templates, which are used to standardize the reporting process for study results.

Employee (Doctor) Assigned to Study: This relationship indicates that each study is assigned to a specific doctor, who is responsible for conducting the study and generating the report. This ensures accountability and clarity in the assignment of medical examinations.

Chapter 6: 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.

6.1. 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.

6.2. 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.

6.2.1. Module Testing

6.2.1.1 Denoiser Deep Learning Approach Testing

Metrics used:

- The structural similarity index measure (SSIM) is used for measuring the similarity between two images. The Structural Similarity Index (SSIM) metric extracts 3 key features from an image: Luminance, Contrast, Structure.
- Peak signal-to-noise ratio is a widely used metric that measures the quality of a processed image by comparing it to the original, assuming both images share the same resolution. PSNR represents the ratio between the maximum possible power of a signal, which is the original image, and the power of the

noise, which is based on the discrepancy between the original and processed images.

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \qquad MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{i=0}^{n-1}[I(i,j) - K(i,j)]^2.$$

Results After 2 Epoches:

Size	10000	20000
Data Split	Validation	Testing
SSIM	71.5	78.48
PSNR	25.4	28.55

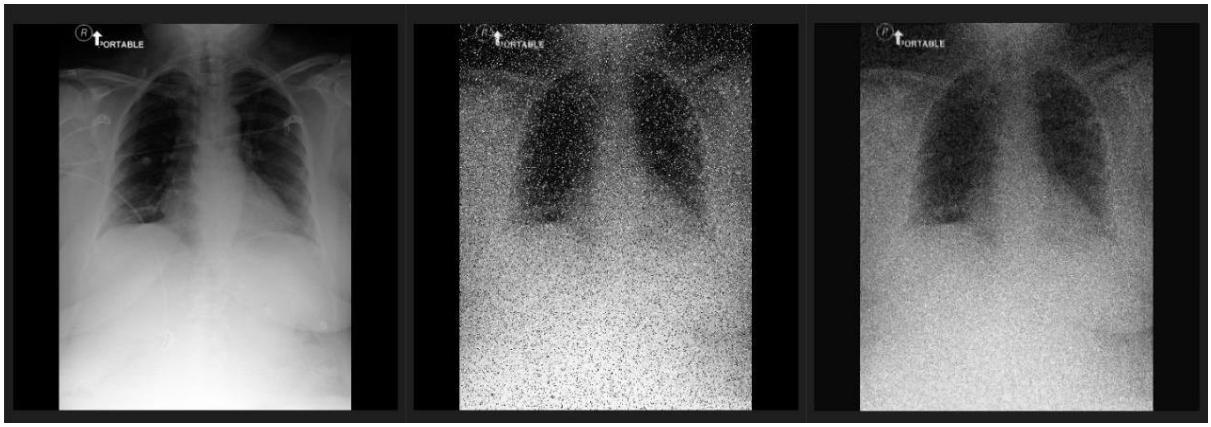
Table 3 Denoiser Metrics Results

Output Noised Target

Table 4 Denoiser Image Results

Then we tried our module on images with several image types in the input image and we obtained robust results as illustrated in Table 5

Target Noised Output



Noise type: convolve noise, salt and pepper noise, gaussian projection noise
PSNR: 21.79

Noise type:block pixel noise, salt and pepper noise, convolve noise
PSNR: 21.39

Table 5 Denoiser Results On Images with Multiple Noise Types

6.2.1.2 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

Table 6 ML Denoiser Results

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

Results

Figure 50 Convolution noise (Denoiser ML)

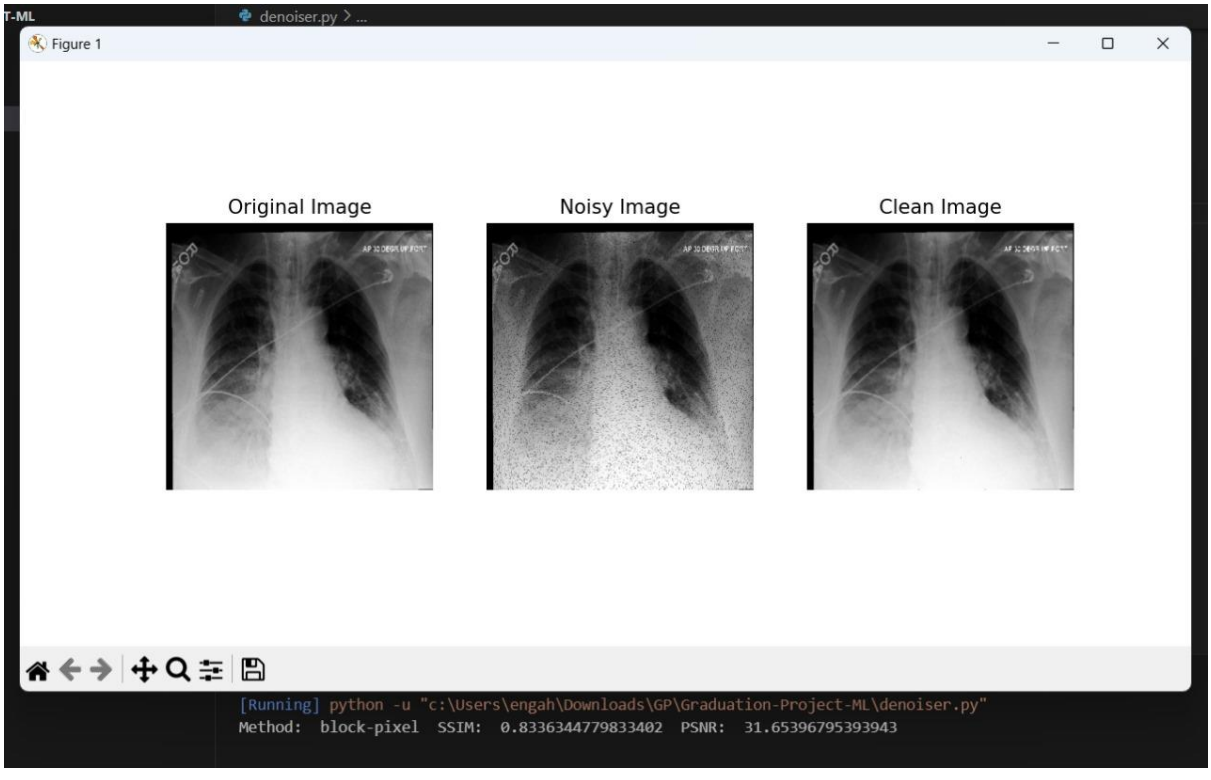


Figure 51 Block pixel (Denoiser ML)

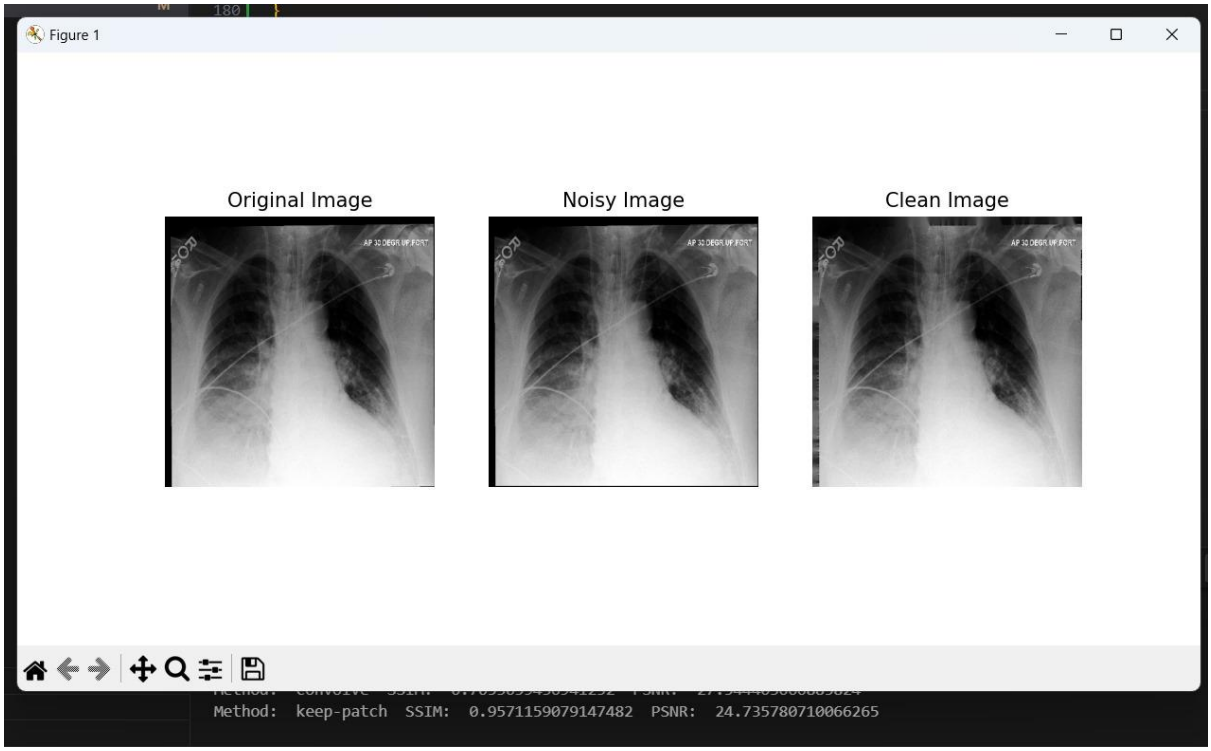


Figure 52 Keep batch (Denoiser ML)

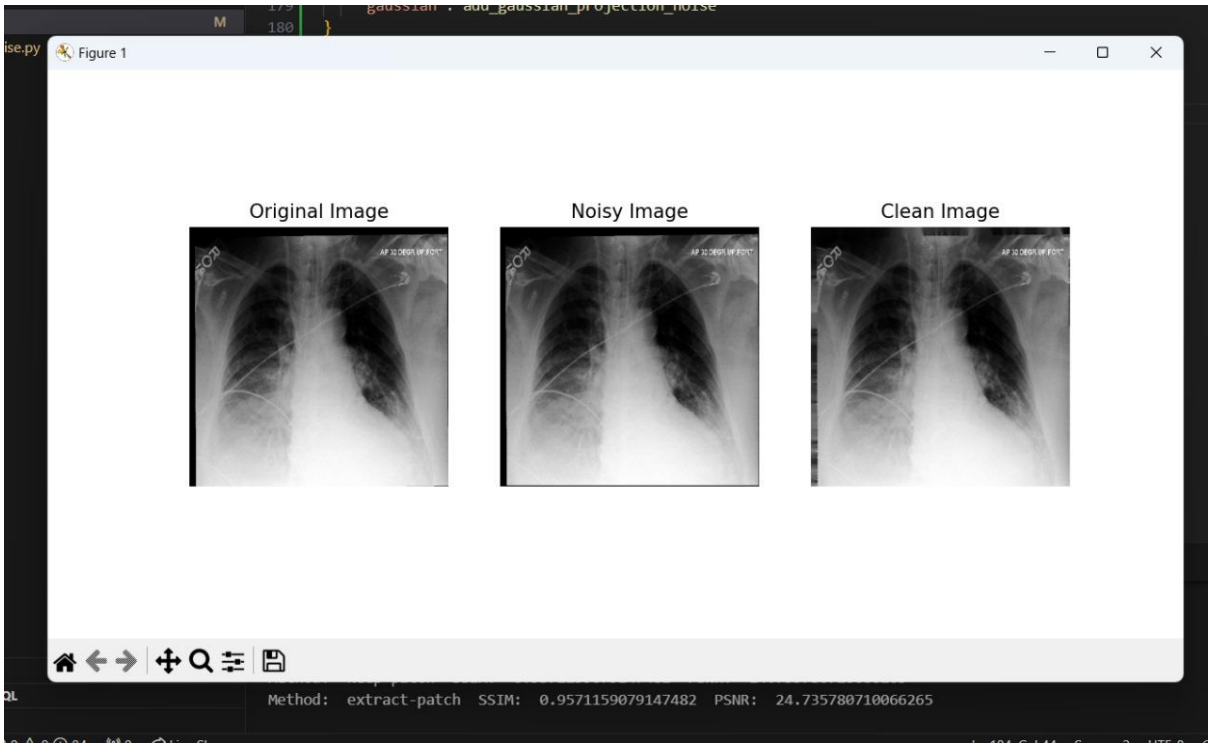


Figure 53 Extract patch (Denoiser ML)

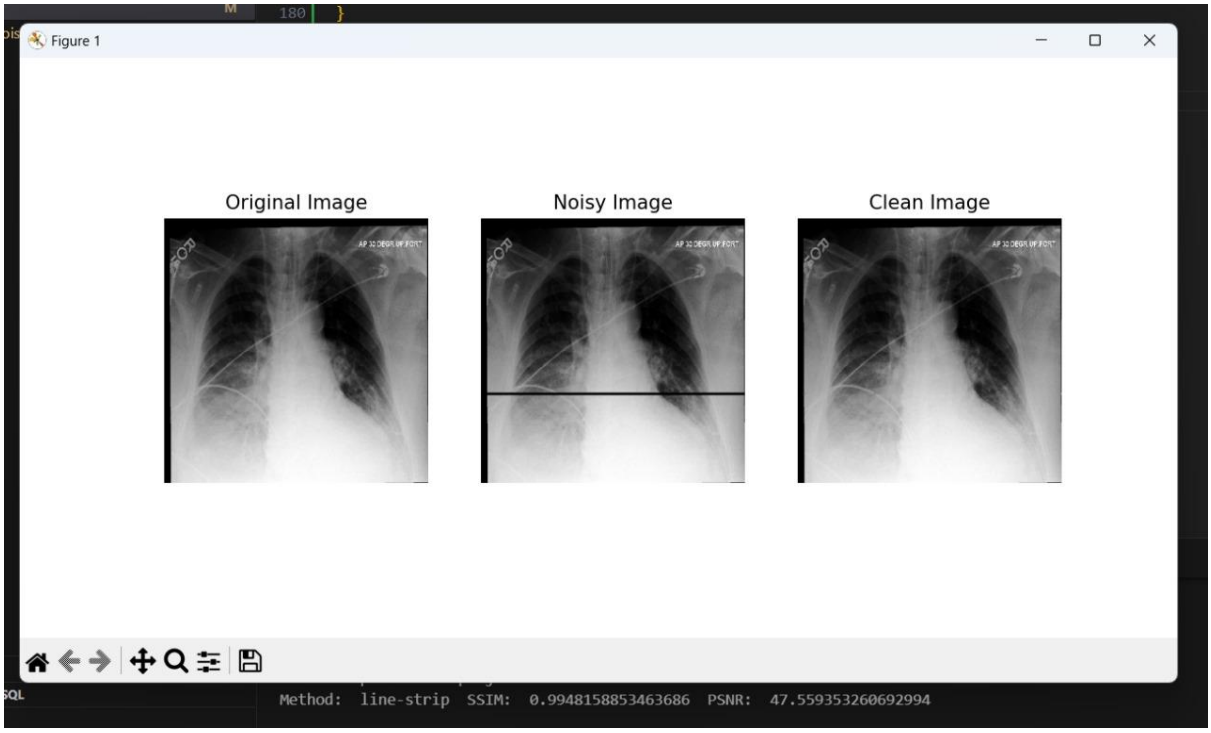


Figure 54 Line strip (Denoiser ML)

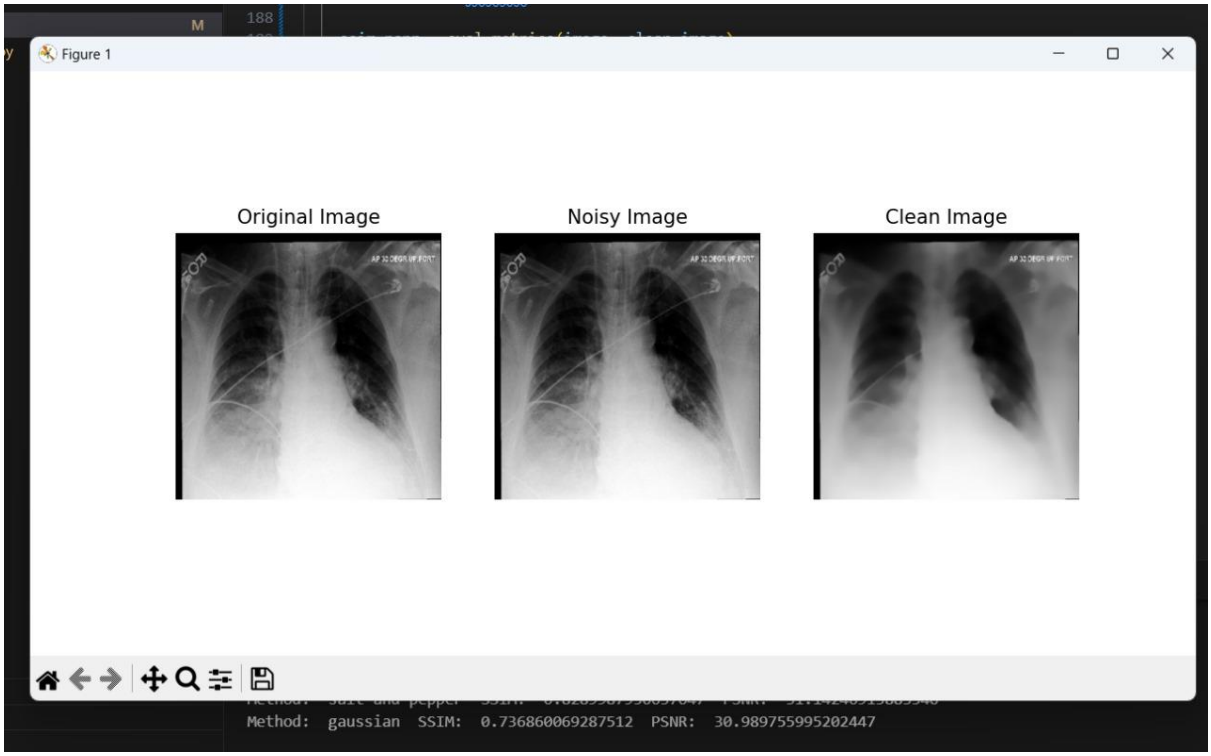


Figure 55 Random noise (Denoiser ML)

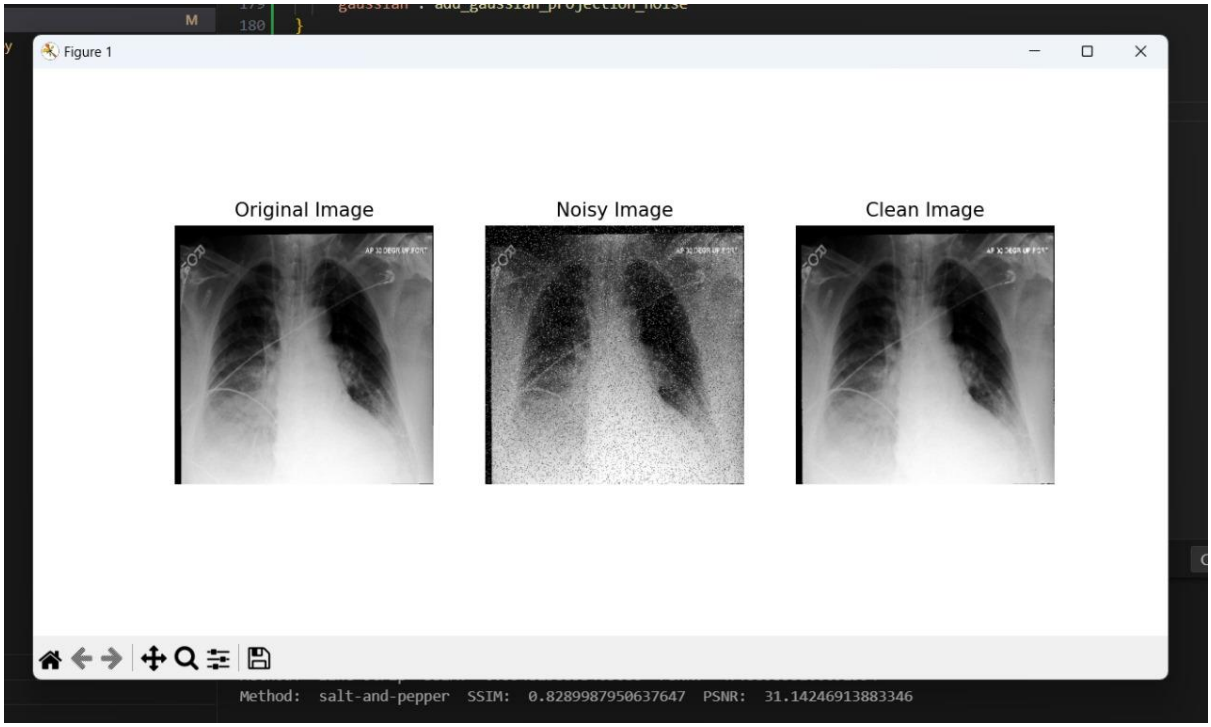


Figure 56 Salt and pepper (Denoiser ML)

6.2.1.3 Object Detector Deep Learning Approach Testing

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

Figure 57 False Positive True Positive and False Negative Object Detector DL Results

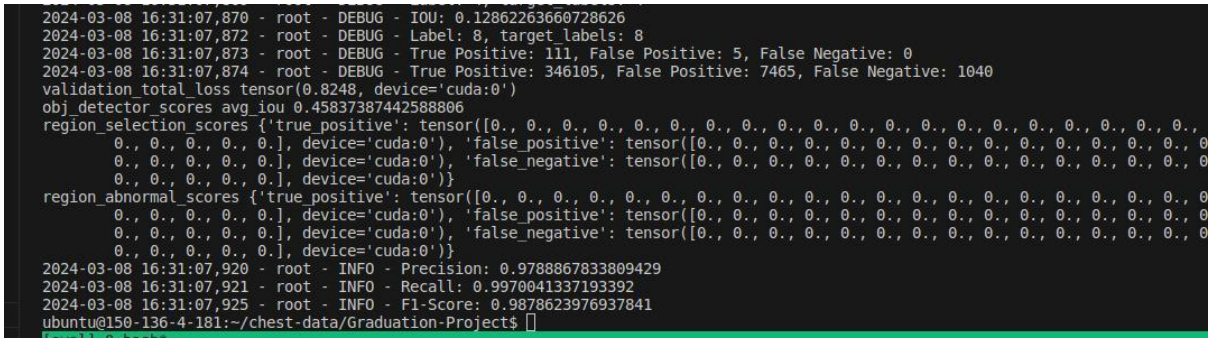


Figure 58 Precision, Recall and F1-Score

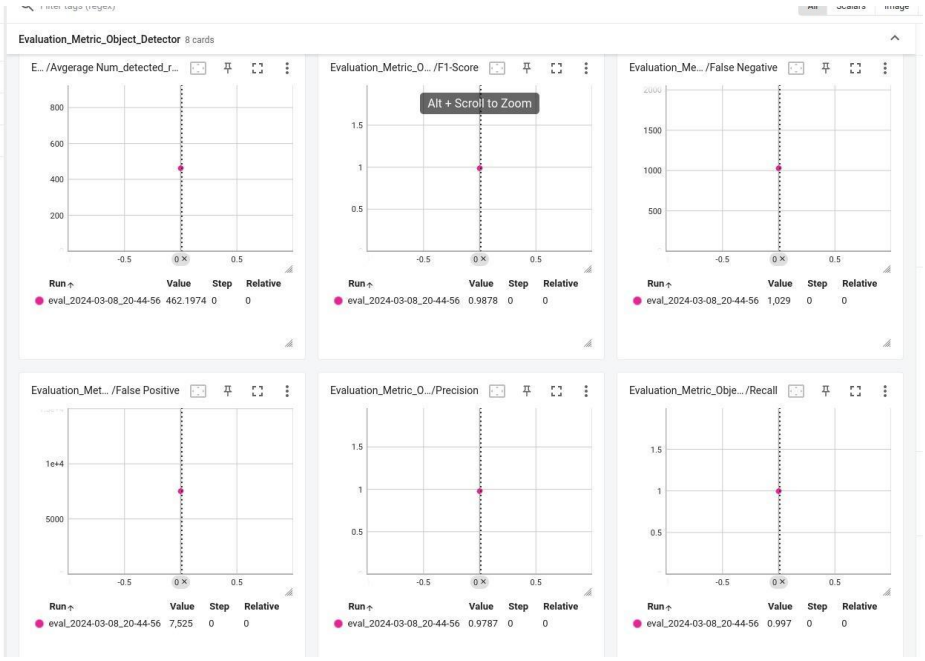


Figure 59 f1 score from tensorboard (object detector DL)

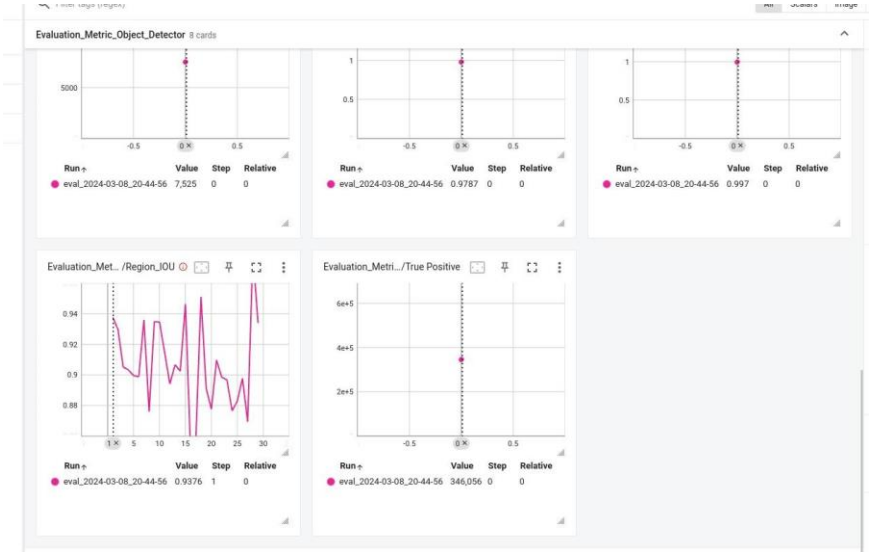


Figure 60 IOU result from tensorboard (object detector DL)

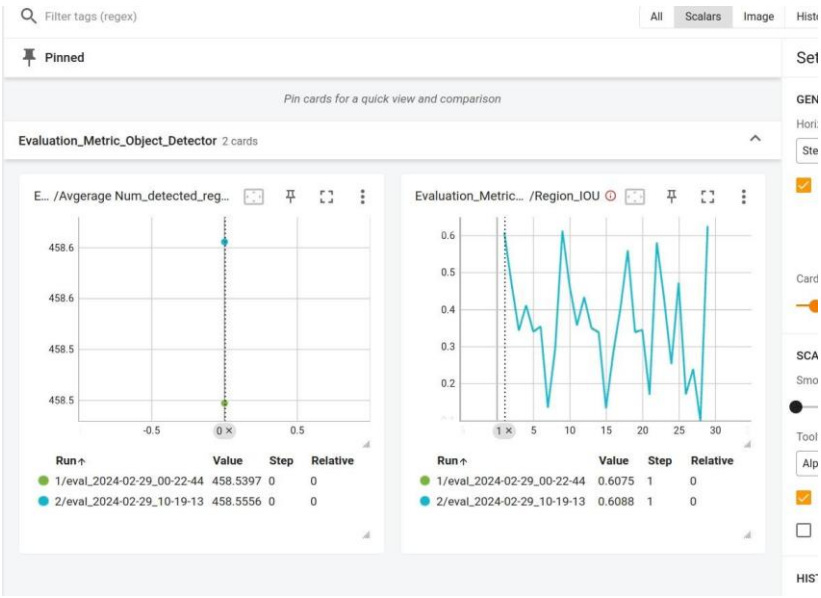


Figure 61 IOU result from tensorboard (object detector DL)

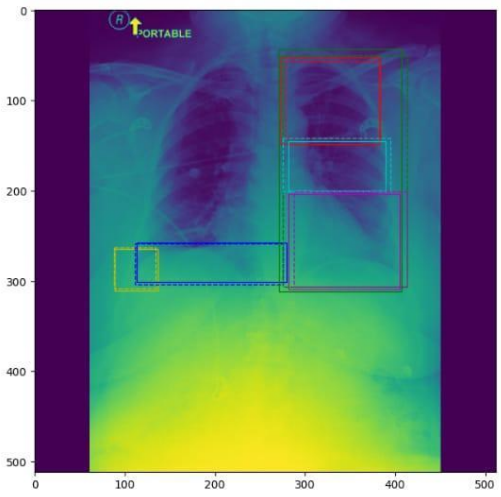


Figure 62 Resulting image from evaluation (object detector DL)

6.2.1.4 Object detector Machine Learning Approach Testing

After Training SVM and linear regression by using hog feature extraction we get results:

Hog Feature Extraction:

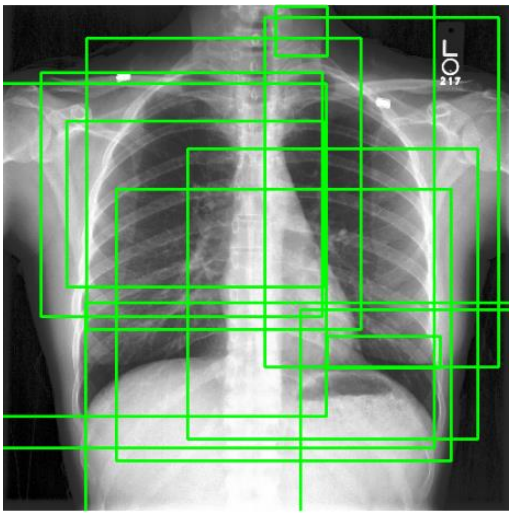


Figure 63 Bounding Boxes example (1) results from HOG

IOU Region(1)	0
IOU Region(2)	0.44526825922402635
IOU Region(3)	0.5779246233969871,
IOU Region(4)	0.524560651521646
IOU Region(5)	0.5720338983050848,
IOU Region(6)	0.6641872261495803
IOU Region(7)	0.5597651335104152
IOU Region(8)	0.5094604484864482
IOU Region(9)	0.6595209966706047
IOU Region(10)	0.6264288980338363,
IOU Region(11)	0
IOU Region(12)	0.4384176182707993

Table 7 IOU Results for example (1)

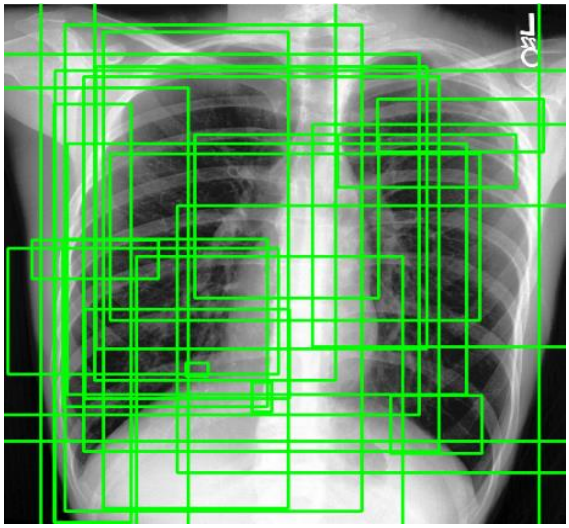


Figure 64 Bounding Boxes example (2) results from HOG

IOU Region(1)	0.5106138621442107
IOU Region(2)	0.44819189721272407
IOU Region(3)	0.5439073016042923,
IOU Region(4)	0
IOU Region(5)	0
IOU Region(6)	0
IOU Region(7)	0
IOU Region(8)	0
IOU Region(9)	0.
IOU Region(10)	0.44059054516700064
IOU Region(11)	0
IOU Region(12)	0.7573651698740793
IOU Region(13)	0.4669576059850374

IOU Region(14)	0.49874109010957834
IOU Region(15)	0.49874109010957834
IOU Region(16)	0.6179673321234119
IOU Region(17)	0.5057440425376228
IOU Region(18)	0
IOU Region(19)	0.6265867550161933
IOU Region(20)	0
IOU Region(21)	0.6019587957462837
IOU Region(22)	0
IOU Region(23)	0
IOU Region(24)	0.6531010138324509
IOU Region(25)	0
IOU Region(26)	0

Table 8 IOU Results for example (2)

6.2.1.5 Classifier Testing

To test our classifiers, we conducted validation testing on a test split of our dataset and computed Precision, Recall, and F1-score for each classifier. The results for the Abnormal Classifier are illustrated in table

Regions	Precision	Recall	F1 Score
All	0.5915	0.8991	0.7133
Normal	0.607	0.4594	0.8973
Abnormal	1	0.8991	0.9469

Table 9 Abnormal Classifier Evaluation Metric Results

- **All**: Classification of regions as either normal or abnormal.

- **Normal**: Metrics computed only on regions classified as normal to assess detection accuracy.
- **Abnormal**: Metrics computed only on regions classified as abnormal to assess detection accuracy."

This structure provides a clear overview of how the results are organized in the table, distinguishing between overall classification and specific performance metrics for normal and abnormal classifications. Adjust the table headers and content based on your actual results for precision, recall, and F1-score.

The testing result for the region selection classifier is shown below in table 6.2.1.5 2

Regions	Precision	Recall	F1 Score
All	0.372	0.9076	0.47

Table 10 Region Selection Classifier Evaluation Metric Results

6.2.1.6 Custom Language Model Testing

To test Language model prediction, we have several metrics that help identify performance of our language model to capture medical information and report i.
we used 5 metrics

- BLEU-1
- BLEU-2
- BLEU-3
- BLEU-4
- ROUGE-L

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L
Language model	0.213	0.133	0.10	0.08	0.33

Table 11 Language Model Validation results

6.2.1.7 Report Generator Classifier Submodule Testing

Since the data is highly imbalanced then using f1-score as an evaluation metric for such a problem won't be effective so we used the AUC as an evaluation metric We compute the optimal thresholds from the true positive curve but since computing the ROC takes a lot of time so we decided to finish training and just validate each epoch based on the loss to save the best models and later on we loaded the best trained model saved based on the average loss on the validation dataset and get the optimal thresholds per each class as shown in table 6.2.1.6.2 .It is clear that our thresholds are very small, near 0.2. This is due to the class imbalance problem we discussed before. This forces us to display the confidence level in correlation with the report, not just the polarity of the findings. The high sensitivity to the chosen threshold results from the significant data imbalance.

Investigating results in Table 6.2.1.6 1 computed the ROC Curve of our classification network on the validation subset we found that the results are perfect. We are having AUC for most of the labels above 0.8.

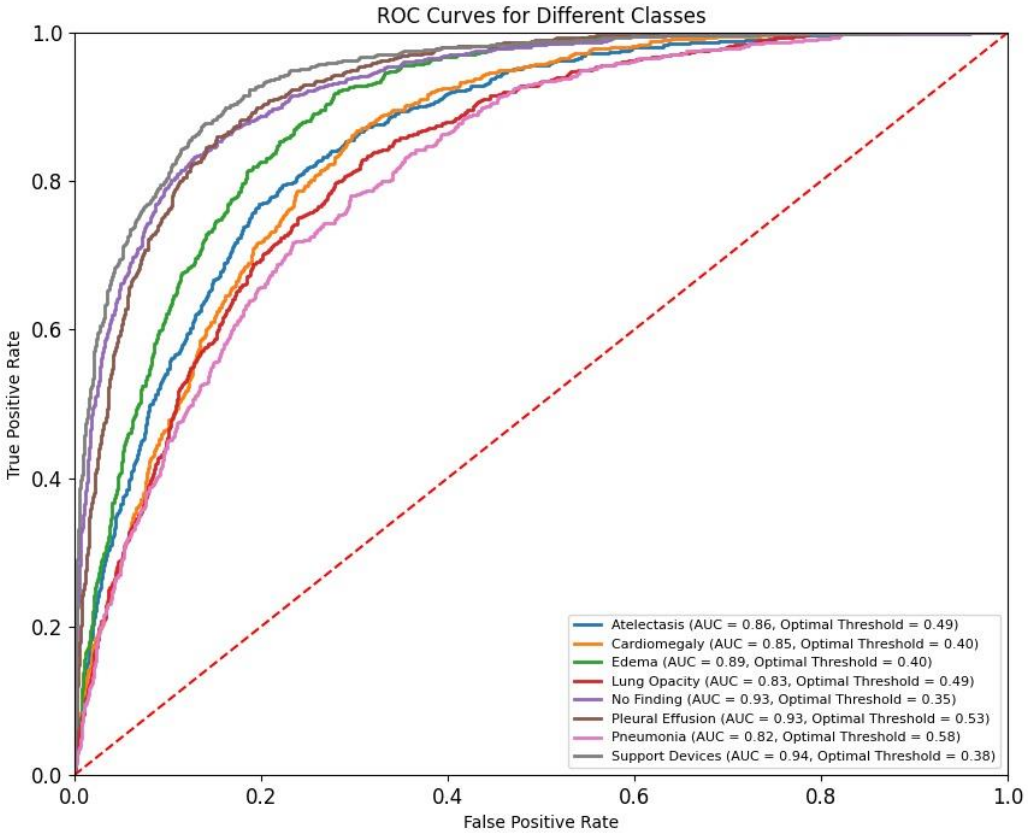


Figure 65 ROC Curve for Classifier

Label	Optimal Threshold
Atelectasis	0.279
Cardiomegaly	0.279
Edema	0.214
Lung Opacity	0.21
No Finding	0.35
Pleural Effusion	0.228
Pneumonia	0.135

Table 12 Optimal thresholds based on ROC Curve

Due to the severe imbalance in the data set using f1-score as a metric will be very sensitive to such problem so the main metric we used for the evaluation of our model is the AUC, but we have also run the evaluation for f1-sore and classification retics to just have intuition about the behavior of the classifier.

Class	Precision	Recall	F1 Score	FP	FN	TP	TN
Atelectasis	0.3767	0.6864	0.4865	134	37	81	448
Cardiomegaly	0.3065	0.6477	0.4161	129	31	57	483
Edema	0.3873	0.6875	0.4955	87	25	55	533
Lung Opacity	0.3849	0.6644	0.4874	155	49	97	399
No Finding	0.6824	0.8202	0.7450	121	57	260	262
Pleural Effusion	0.5692	0.8102	0.6687	84	26	111	479
Pneumonia	0.3415	0.5932	0.4334	135	48	70	447

Support Devices	0.5621	0.7167	0.6300	67	34	86	513

Saliency Map Results:

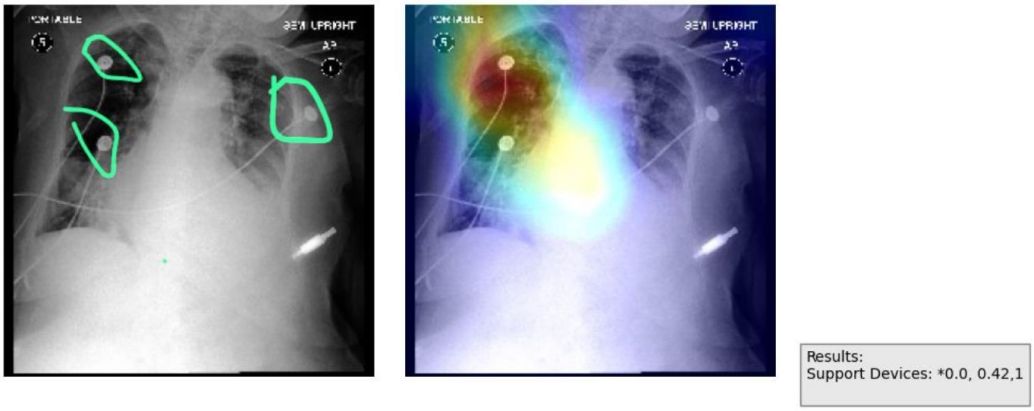


Figure 66 Support Devices localization saliency map with doctor's annotation

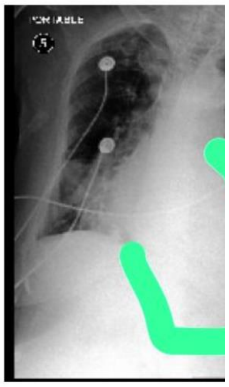


Table 13 Template Based Classifier Metric Results

Figure 67 Cardiomology localization saliency map with doctor's annotation

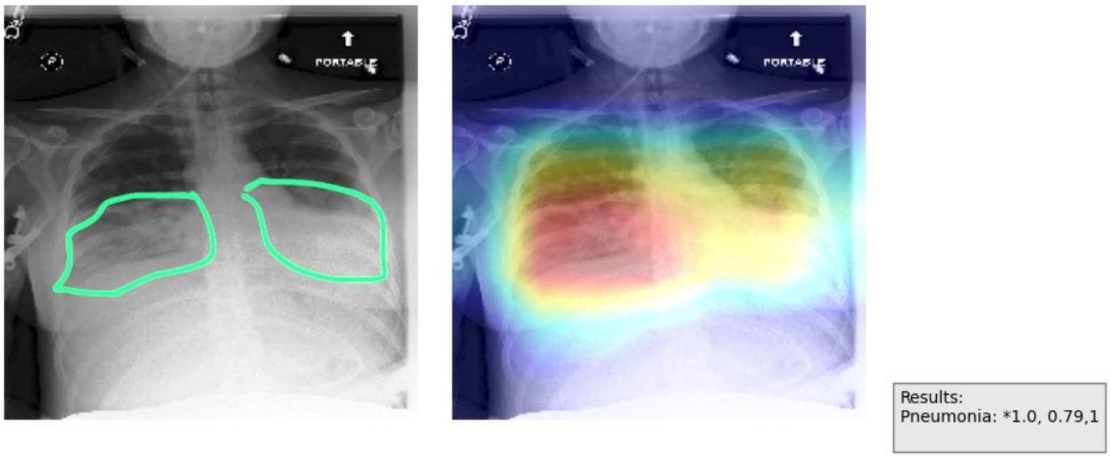


Figure 68 Peneuminia localization saliency map with doctor's annotation

6.2.2. Integration Testing

6.2.2.1 Custom Bounding Box Sentences

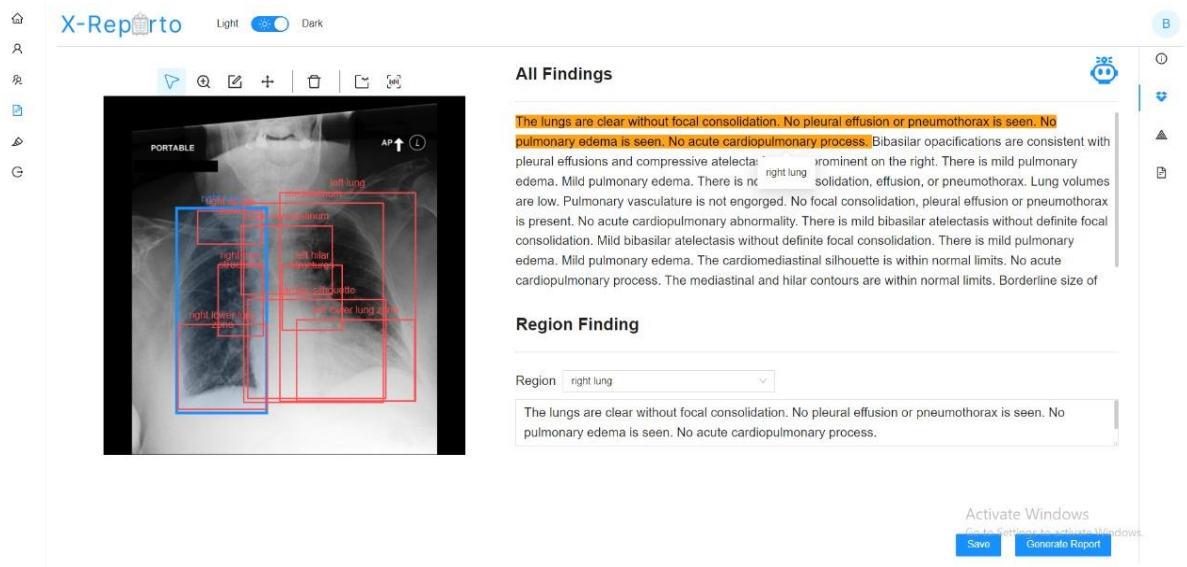


Figure 69 Custom Bounding box Sentences

After Generating report , Radiologist can choose any region sentence that exists on the image and modify it and save it without changing the original report.

6.2.2.2 Heat Map Image

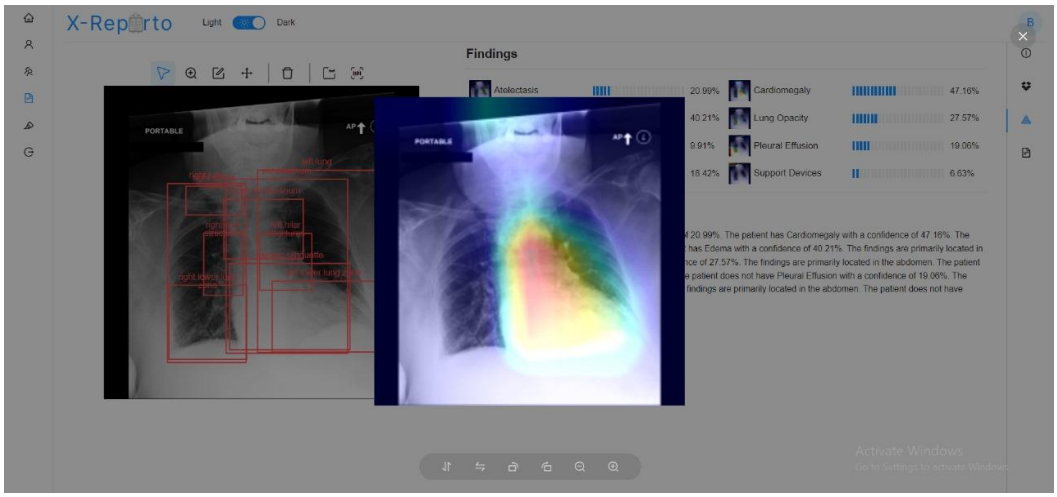


Figure 70 Heat Map Image

Radiologist Can get Location of each diseases in x-ray that template based report proved

6.2.2.3 Generated Report

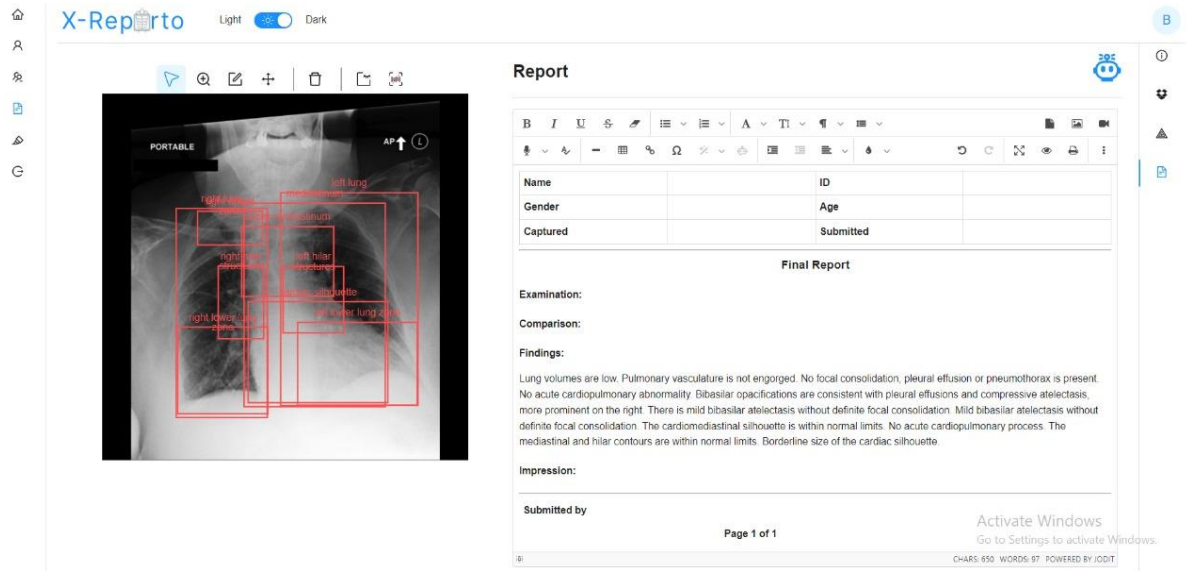


Figure 71 Generate AI report and doctor can edit it

Radiologist Can get generated report and put it in template format to be saved and modified and printed

6.2.2.3 Template Based Report

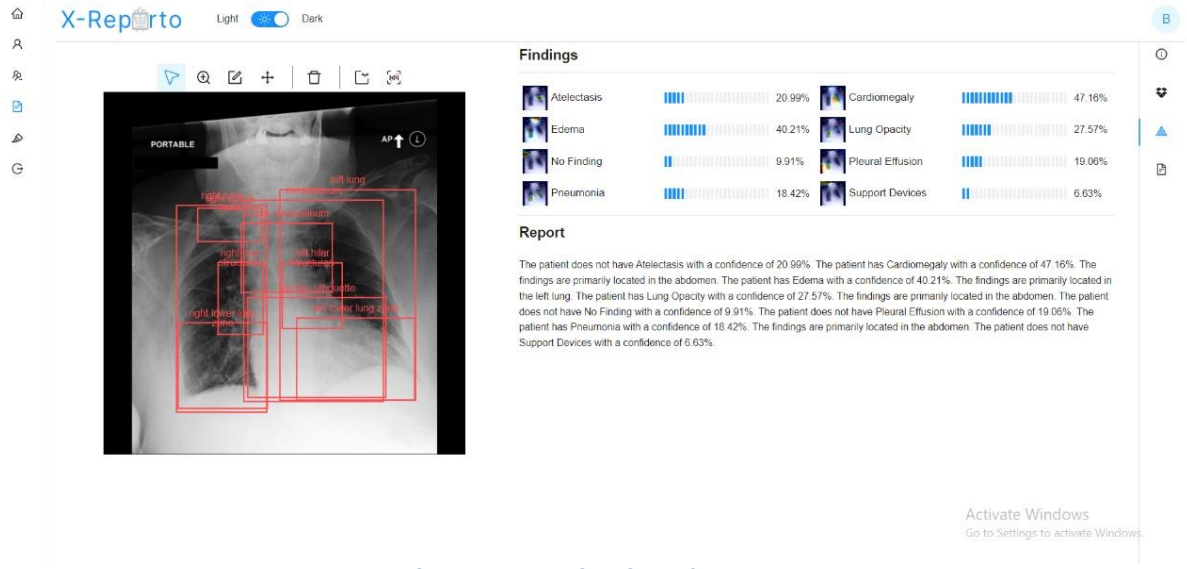


Figure 72 Template based Report

Radiologist Can also get template-based report by probabilities of existence of each disease in this chest x-ray ant location on image

6.3. Testing Schedule

we start testing the modules of X-reporto in February and try to enhance them until july

6.4. Comparative Results to Previous Work

we compare our results of medical reports with other approaches mentioned above

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L
CNN-RNN (Vinyals et al., 2015)	0.316	0.211	0.140	0.095	0.267
R2Gen (2020)	0.323	0.218	0.145	0.103	0.277
X-Reporto (2024)	0.313	0.233	0.18	0.121	0.33

Table 14 Comparative Study

Chapter 7: Conclusions and Future Work

7.1. Faced Challenges

7.1.1 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

7.1.2 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

7.1.3 Custom Language model

How to deal with visual features generated by object detectors:
We designed our visual Encoder model that applies transformation from image space features to text space features.
How to integrate visual features information in our language model:
We designed our custom masked self-attention block that adds visual features keys along with input to generate output scores.
How to train LLM with limited resources:

- Apply gradient checkpointing mechanism
- Apply gradient accumulating while training.

How to deal with redundant sentences:
Apply sentence similarity using pre-trained language model to remove very high similar sentences.
How to deal with hallucination:
We designed a regex that searches for English words or repeated characters or repeated continuous words, then removes these sentences.

7.2. Gained Experience

7.2.1 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

7.2.2 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

7.2.3 Language Model

- Learned Transformer Architecture
- How to train Language Model
- how to customize language model upon your problem
- How to optimize model resources and chose best parameters
- How to generate sentences using trained model

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.

7.4. Future Work

7.4.1 Denoiser Deep Learning:

- Try to Mix types of noise and train modules on this data to be able handle many types of defects.
- Use Resnet on Perceptual Loss instead of Vgg for improving results.

7.4.2 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

7.4.3 Object Detector Deep Learning

- Try to use YOLO and get the feature from external module by crop the part in bounding box and send it to feature extractor module
- Validate the input is a valid chest X-ray image before sending it to the object detector
- Use efficient net instead of resnet50 as backbone which may enhance the feature extractor

7.4.4 Language Model

- Try to use bigger model sizes in order to generate more complex sentences and reports.
- Improve Visual Encoder Model to enhance transformation from image space to text space.

7.4.5 Template Based Report Generator Model

- Since we are now using the x-Ray as the input which makes it a challenge task to identify all findings without patient history which is a very difficult task

for the radiologist themselves, they need to read the history patient so we may integration the patient history and use a multi-model input.

- We can try other reason approaches like grid-CAM for better localization

References

[1] O'shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint

[2] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.

[3] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. InMedical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 2015 (pp. 234-241). Springer International Publishing.

[4] Ahmadi, Mehdi, et al. "Reproducing AmbientGAN: Generative models from lossy measurements." arXiv preprint arXiv:1810.10108 (2018).

[5] Menon, Sumeet, et al. "Generating realistic covid-19 x-rays with a mean teacher+ transfer learning gan." 2020 IEEE International Conference on Big Data (Big Data). IEEE, 2020.

[6] Liu Z, Bicer T, Kettimuthu R, Gursoy D, De Carlo F, Foster I. TomoGAN: low-dose synchrotron x-ray tomography with generative adversarial networks: discussion. JOSA A. 2020 Mar 1;37(3):422-34.

[7] Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019):

[8] Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 2097-2106).

[9] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 4700-4708).

[10] Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning deep features for discriminative localization. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 2921-2929).

[11] M. Lin, Q. Chen, and S. Yan. Network in network. International Conference on Learning Representations, 2014.

[12] 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. Hardware Platforms

This section provides a detailed description of the hardware platforms and environments utilized during the development and testing of the project. Each platform is essential for supporting various aspects of the project, from model training to deployment.

A.1.1. Google Colab

Google Colab is a cloud-based platform that provides free access to powerful GPUs and TPUs, making it an ideal environment for training machine learning models. In this project, Colab was used for downloading datasets and uploading to google drive storage. Colab ease of access to datasets stored in Google Drive significantly streamlined the data management process.

A.1.2. CMP Server

The CMP Server is a dedicated hardware from our department designed for intensive computational tasks, equipped with high-performance GPUs optimized for deep learning workloads. This server was utilized for experiment initial ai models and validating working of models before actually training to see it’s work or need to be fixed.

A.1.3. Development Server on Cloud

The Development Server on Cloud is a remote server utilized for training AI models throughout the project. This cloud-based infrastructure offers scalable computational resources, making it ideal for handling the extensive processing demands of deep learning tasks. By leveraging this server, we were able to efficiently train complex models on large datasets, ensuring optimal performance and reliability. The flexibility of the cloud environment also allowed for easy adjustments to computational resources as needed, facilitating an agile development workflow and supporting ongoing model enhancements and evaluations.

A.2. Software Tools

A.2.1. 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.2.2. 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.2.3. 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.2.4. 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.2.5. 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.

A.2.6. FastAPI

FastAPI is a modern, high-performance web framework for building APIs with Python. It was employed in the project to develop the backend services, providing a robust and efficient architecture for handling requests between the frontend and AI models. FastAPI's asynchronous capabilities and automatic generation of interactive API documentation made it an ideal choice for this application.

A.2.7. Postman

Postman is a powerful tool for testing and interacting with APIs. It was utilized during the development phase to facilitate the testing of the FastAPI endpoints and ensure proper functionality of the backend services. Postman allowed for easy simulation of various API requests and validation of responses, streamlining the debugging process.

A.2.8. PostgreSQL

PostgreSQL is a powerful open-source relational database management system. In this project, PostgreSQL was used to store and manage the data associated with patients, studies, results, and templates. Its robustness, support for complex queries, and scalability made it a suitable choice for managing the backend data efficiently.

A.2.9. Docker

Docker is a containerization platform that enables the creation and management of lightweight, portable containers for applications. In this project, Docker was employed to package the entire application, including the backend and AI models, ensuring consistent deployment across different environments. This streamlined the development workflow and simplified the deployment process, allowing for easier updates and maintenance of 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

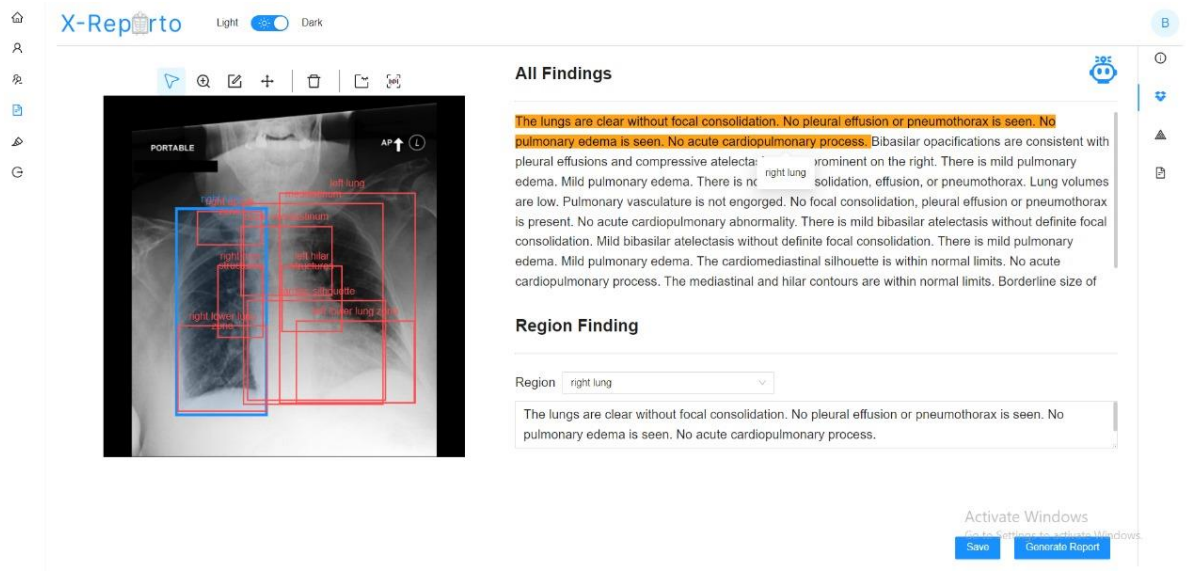


Figure 73 generate AI boxes in abnormal region

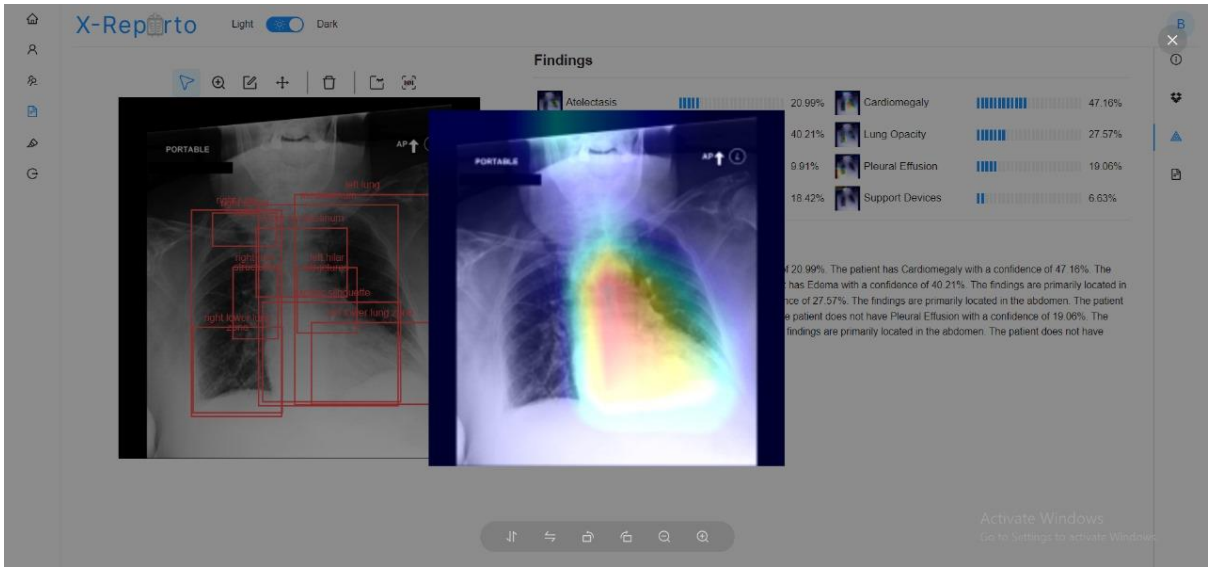


Figure 74 generate heatmap for each disease

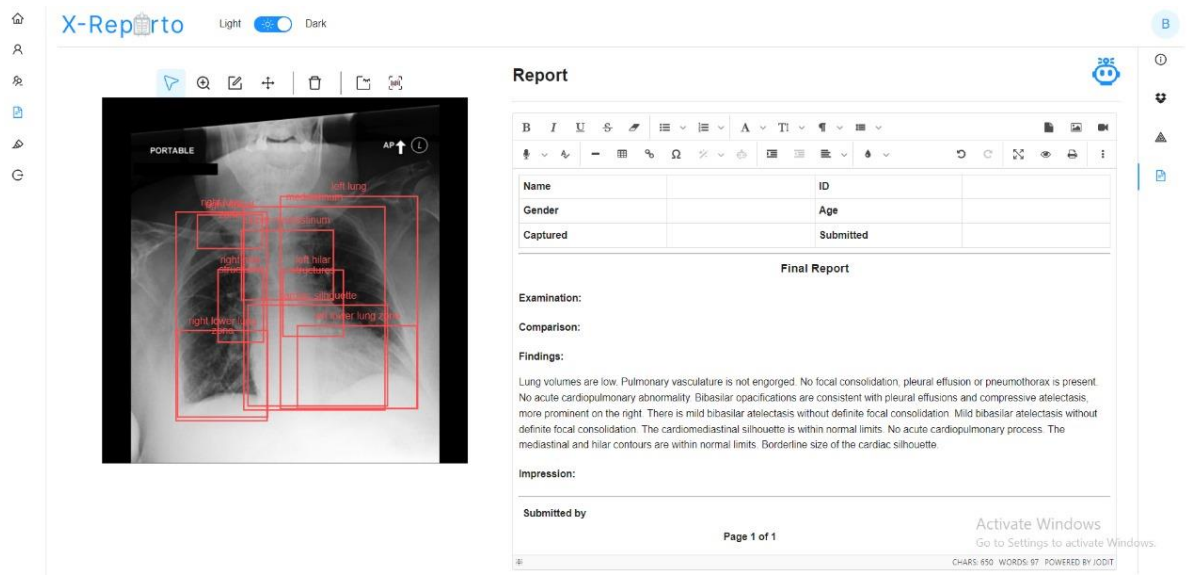


Figure 75 generate AI report and doctor can edit it

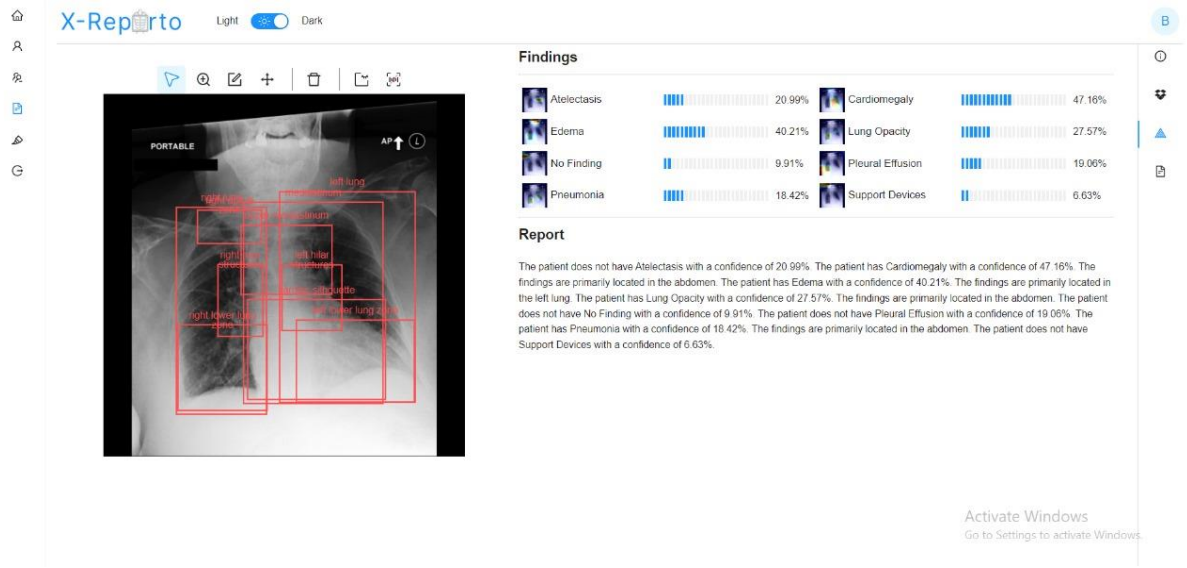


Figure 76 generate each disease probability and template report

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

Appendix D: Project Progress Log

Meet (1): 21 May 2024 (Online)

❖ Dr Notes:

1. Don’t Focus on Accuracy mainly
2. Deploy on mobile + offline ML models
3. Private data for rendering for other models [Medical + Secrecy for Data]
4. Dr Will check for open source
5. The game of the doctor to learn languages
6. Prototypes
7. GPUs
8. Kaggle
9. Nasa Space Apps https://www.spaceappschallenge.org/2023/challenges/
10. benefits for products.
11. can be used in mobile in small size.
12. research not improving numbers that reached before "too risky as it's competition".
13. Federated Learning is a good idea specially in Medical field for data secure.
14. Always not depend on Accuracy
15. Always explore and try prototypes
16. take Hardware into accounts
17. take time of implementation into accounts

Meet (2): 29-9-2023 (Online)

❖ Main Topic: Discussing Our new ideas.

❖ Dr Notes:

1. Word
    1. Facebook Similar Model (large) the doctor will check in meta.ai. research
    2. What is Our Target??
1. Student APP

2. Question generation & Auto grader → gives me the part from the question was generated.
3. Fake Problem: We won't find data
4. Lost Children
5. Need to search on the other contributions.
6. Fashion
    1. 360 scanning
    2. ML
    3. ML to scan
    4. How are the clothes being predicted
7. Ads by eye gazing
8. Steal mobile.
9. Recommendation system with eye gaze
    1. Past view of the place
    2. Translation
10. Decoration:
    a. Add feature to know old furniture + recommend new and view

Meet (3): 3-10-2023 (On Campus)

- Main Topic: Discussing Regulations of the Sponsor Cacao Systems.
- Dr. Notes: Sponsorship approved by the doctor.

Meet (4): 12-10-2023 (On Campus)

- Main Topic: New Sponsor Voyance Medicals
- Our ideas:
    1. X- Report Generation
    2. Medical Assistance App [Full Assistance]
- Dr. Notes:
    1. Check the Dataset availability.
    2. The Dr Preferred the Report idea

Meet (5): 19-12-2023 (On Campus)

- Rehearsal for the presentation

Meet (6): 19-12-2023 (On Campus)

- Main Topic:
- Dr. Notes:
    1. Fine till now.
- Done Till now:
    1. Done Obejctect Detector

Meet (7): 25-3-2024 (On Campus)

- Main Topic: Problem in (Language Model - HeatMap) and verify load of project
- Details:
  1. In Language Model generates same sentences for all images
  2. Heat Map Problem
- Dr. Notes:
  1. Model can be not learning from images
  2. Load is fine

Meet (8): 19-4-2024 (On Campus)

- Main Topic: Start training Language Model
- Details:
  1. We have resource problem
- Dr. Notes:
  1. Try to contact to doctors to get resources

Meet (9): 15-5-2024 (On Campus)

- Main Topic:
- Dr. Notes: fine till now.

Done Till now:

- ✓ Done gpt Trainer on half epoch
- ✓ We have a problem with server version
- ✓ Train heat map on 5 subsets on colab and get results and Confirmed Result of an Example from a Dr.
- ✓ We begin in Denoiser approaches.

Next Meet Tasks:

- ✓ UI/UX initial Design
- ✓ Fix Problem of gpt and complete train

Meet (10): 22 June 2024 (Online):

- Agenda:
  1. Present final UI Screens for the tools
  2. Present results for training GPT (phase (2))
  3. Present results of Denoiser (Deep Approach)
  4. Present initial results of classical approach for Denoiser
  5.
- Dr. Notes:

Meet (15): 1 Jul 2024 (Online):

- Agenda:
  1. Asking Object Detector classical approach results
  2. Asking Denoiser classical approach results

Meet (16): 6 Jul 2024 (Online):

- Agenda:
  1. Present Object Detector classical approach results
  2. Present Denoiser classical approach results
  3. Present completed pages for the website
  4. Present some of the results of the heatmap module

- Dr. Notes:
  1. General Guidelines about writing the book
  2. We need to prepare a Demo next time for the Website

Meet (17): 11 Jul 2024 (Online):

- Main Topic:
  1. Discuss Workload
  2. Present Timeline Documentation file
  3. Present initial parts done so far in the GP book
  4. Demo for the complete website

Meet (18): 13 Jul 2024 (Online):

- Main Topic:
  1. Show Book to doctor
  2. Take Note About Book
  3. Make initial Presentation