CHEST X-RAY IMAGE BASED REPORT GENERATION USING DEEP LEARNING

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

T. MEGHANA	**(20KH1A05A6)**
N. PRAVALLIKA	**(20U41A0570)**
S. SYAM GANGA GANESH	**(20KH1A05A1)**
K. VENKATESWAR REDDY	**(20KH1A05C4)**

Under the Esteemed Guidance of

D. V. PADMAVATHI, M.Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA INSTITUTE OF TECHNOLOGY

(Approved by A.I.C.T.E & Affiliated to J.N.T.U Kakinada)

(An ISO Certified Institution & Accredited by NBA)

NARASARAOPETA-522601

(2020-2024)

NARASARAOPET INSTITUTE OF TECHNOLOGY

NARASARAOPET

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "CHEST X-RAY IMAGES BASED REPORT GENERATION USING DEEP LEARNING" is a bonafide project work carried out by the students T. MEGHANA (20KH1A05A6), N. PRAVALLIKA (20U41A0570), S. SYAM GANGA GANESH (20KH1A05A1), K. VENKATESWAR REDDY (20KH1A05C4), under the guidance and submitted in partial fulfillment for the award of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING from JNTUK KAKINADA, during the academic year 2023-2024.

Project Guide	**Head of the department**
D.V.Padmavathi., M.Tech	**Dr.R.SATHEESKUMAR., M.Tech.,Ph.D**
Assistant Professor	**Professor & Head**
Department of Computer Science and Engg.	**Department of Computer Science and Engg.**
Narasaraopeta Institute of Technology	**Narasaraopeta Institute of Technology**

Submitted for the Project Viva – Voice examination held on ____________

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We wish to express our thanks to various personalities who are responsible for the completion of the Project. We are extremely thankful to our beloved Chairman **Sri. M.V. Koteswara Rao** who took keen interest in using every effort throughout this course. We owe out our gratitude to our principal **Dr. P. Hari Krishna Prasad** for his kind attention and valuable guidance throughout the course.

We would like to express my sincere thanks to our Head of the Department **Dr. R. Satheeskumar,** Professor, Department of Computer Science and Engineering, for his valuable guidance and support in completing my Project.

We express our thanks to our Guide **D.V.Padmavathi,** Assistant Professor, Department of Computer Science and Engineering for her willingness and valuable guidance for successful completion of the Project.

We extend our sincere thanks to all other teaching and non-teaching staff of the Department for their cooperation and encouragement during our B.Tech course. We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

SUBMITTED BY

T. MEGHANA	**(20KH1A05A6)**
N. PRAVALLIKA	**(20U41A0570)**
S. SYAM GANGA GANESH	**(20KH1A05A1)**
K. VENKATESWAR REDDY	**(20KH1A05C4)**

DECLARATION

We (T. MEGHANA, N. PRAVALLIKA, S. SYAM GANGA GANESH, K. VENKATESWAR REDDY), the students of the NARASARAOPETA INSTITUTE OF TECHNOLOGY here by this project titled "CHEST X-RAY IMAGE BASED REPORT GENERATION USING DEEP LEARNING" being submitted to the Department of Computer Science and Engineering, NARASARAOPETA INSTITUTE OF TECHNOLOGY, Kotappakonda road, Yellamanda, Narasaraopeta, Guntur district. This project has not been submitted to any other University or Institute for the award of any degree.

Project Associates

T.MEGHANA	**(20KH1A05A6)**
N.PRAVALLIKA	**(20U41A0570)**
S.SYAM GANGA GANESH	**(20KH1A05A1)**
K.VENKATESWAR REDDY	**(20KH1A05C4)**

ABSTRACT

An adversarial reinforced report-generation framework for chest x-ray images is proposed. Previous medical-report-generation models are mostly trained by minimizing the cross-entropy loss or further optimizing the common image-captioning metrics, such as CIDEr, ignoring diagnostic accuracy, which should be the first consideration in this area. Inspired by the generative adversarial network, an adversarial reinforcement learning approach is proposed for report generation of chest x-ray images considering both diagnostic accuracy and language fluency. Specifically, an accuracy discriminator (AD) and fluency discriminator (FD) are built that serve as the evaluators by which a report based on these two aspects is scored. The FD checks how likely a report originates from a human expert, while the AD determines how much a report covers the key chest observations. The weighted score is viewed as a ‘‘reward’’ used for training the report generator via reinforcement learning, which solves the problem that the gradient cannot be passed back to the generative model when the output is discrete. Simultaneously, these two discriminators are optimized by maximum-likelihood estimation for better assessment ability. Additionally, a multi-type medical concept fused encoder followed by a hierarchical decoder is adopted as the report generator. Experiments on two large radiograph datasets demonstrate that the proposed model outperforms all methods to which it is compared.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
SVM	Support Vector Machine
VGG-16	Visual Geometric Group
X-RAY	X Radiation
EHR	Electronic Health Records
DWI	Driving While Impaired
DTA	Decision Tree Algorithm
AI	Artificial Intelligence
RELU	Rectified Linear Unit
API	Application Programming Interface

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Automatic radiology-report generation is a computer-aided diagnostic technology used for generating a free-text description of disease diagnosis or future treatment based on radiology images (such as chest x-rays). Compared with general disease diagnosis technology, it is closer to artificial intelligence (AI), for it can not only output a list of numbers corresponding to the probabilities of possible diseases but also ‘‘write’’ an easy-to-understand report with natural language. With this technology, patients can read the chest x-rays by themselves, and no longer have to queue up to consult doctors. Moreover, the workload of radiologists will be greatly lightened. Chest x-rays are the most common type of radiology image, which produces images of the heart, lungs, airways, blood vessels, and bones of the spine and chest, and is used. The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa M. Fouda. for diagnosis and treatment of chest diseases, such as pneumonia and pneumothorax. A chest x-ray report example is shown in Figure 1. Such a report includes two important parts: findings and impression. The former part describes in detailed the representations of different organs and regions and a determination of whether the patient has a certain or potential disease. The latter part is only the conclusion of the former part. Hence, the focus in this article is on generation of the findings. A similar study area is natural image captioning in computer vision and natural language processing because it has the same objective of mapping from images to text sequences. Hence, some common points exist between the two studies. First, encoder-decoder architecture is the basic architecture used to tackle these problems, in which the encoder, composed of a deep convolutional neural network (CNN), encodes images into a contextual vector, and the decoder, composed of long short-term memory (LSTM), decodes the contextual vector into a word sequence step by step**.**

Additionally, the entire model is usually trained by minimizing the cross-entropy loss or further retuned via reinforcement learning. However, an image-captioning approach cannot be directly applied to medical report generation for several significant reasons: compared to natural images, chest x-rays involve complex and abstract medical concepts ,e.g., ‘‘pulmonary atelectasis’’ and ‘‘cardiomegaly’’, which is difficult for a plain encoder to capture; a natural image caption mostly has one sentence, while a finding contains four, five, or even more sentences, and thus the basic decoder may struggle to learn such long-term dependencies; and for medical report generation, one should make diagnostic accuracy the top priority, rather than blindly seeking high text-relevance scores (such as BLEU score). Following, some improvements are made herein based on the original encoder-decoder approach. First, multi-type medical concepts are incorporated into the encoder. Detailed common chest observations and medical subject heading (MeSH) labels are adopted as two types of intermediate semantic information, which are predicted by a separate sub-network (called multi-label classification, or MLC) in the encoder. These predicted medical concepts will be embedded in the follow-up decoder along with the encoded image features. These two types of medical concepts have different semantic granularities, in which the observations cover generalized diseases, while the MeSH terms narrow their concepts to medical vocabulary. Second, a hierarchical LSTM is adopted as the decoder. The hierarchical decoder splits the decoding process into two stages: given encoded features, the sentence LSTM decodes topic vectors one by one, and then the word LSTM decodes a word sequence from a topic vector.

1.2 ARCHITECTURE OF X-RAY IMAGE ANALYSIS

A convolution is defined as an operation on two functions. In image analysis, one function consists of input values at a position in the image, and the second function is a filter each can be represented as array of numbers. Computing the dot product between the two functions gives an output. The filter is then shifted to the next position in the image as defined by the stride length. The computation is repeated until the entire image is covered, producing a feature map. This is a map of where the filter is strongly activated and 'sees' a feature such asa straight line, a dot, or a curved edge. If a photograph of a face was fed into a CNN, initially low-level features such as lines and edges are discovered by the filters. These build up to progressively higher features in subsequent layers, such as a nose, eye or ear, as the features maps become inputs for the next layer in the CNN architecture. Chest X-ray Images refer to radiographic images of the chest area that are used to detect and diagnose various lung diseases, including COVID-19 pneumonia. These images are captured using X-ray technology and can provide valuable information to aid in the screening, diagnosis, and monitoring of respiratory conditions.

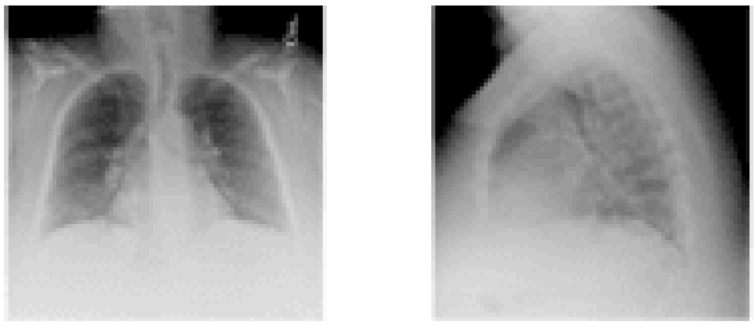


Fig 1.1 Medical image analysis Architecture

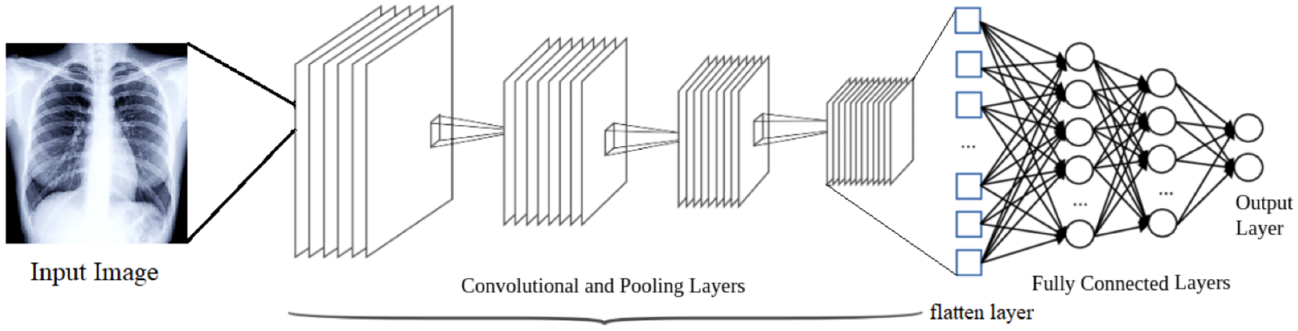


Fig: 1.2 Architecture of Convolutional Neural Network

1.3 PROBLEM DEFINITION

The main problem in this project is by using X-RAY scans it takes lot of time. Recently, deep learning techniques showed promising results towards improving accuracy of detection and classification of different type of tumors and abnormalities present in the chest from magnetic resonance imaging (X-RAY). Through Convolution Neural Network, the image is given to the machine, with the trained data it compares with the existing data in it and detect the type of tumor or abnormality in the chest.

1.4 OBJECTIVE

The objective of this project is to Automatic Report Generation for Chest X-Ray Images via Adversarial Reinforcement Learning.

CHAPTER 2
LITERATURE SURVEY

Title: X-Ray Medical Reports using Deep Learning
Year: 2020
Author: Islam Akef Ebeid

In this paper, Radiology is a medical discipline that uses medical imaging to diagnose and treat diseases. The job of a radiologist is to identify conditions from radiographic images and produce a report describing every image they examine. The field of radiomics aims at extracting a large number of features from radiographic images. The features are used to discover diseases automatically. Even though radiologists receive extensive training in radiomics in many cases, they have to look at many images containing a lot of variation in one sitting and produce a templated report describing abnormalities consisting of findings and an impressions section. Besides, the process itself can be time-consuming and prone to human error, especially in densely populated countries.

Title: Multi-Label Learning with Visual-Semantic Embedded Knowledge Graph for Diagnosis of Radiology Imaging
Year:2021

Author: Daibing Hou, Zijian Zhao, Sanyuan Hu

This project presents with the publishing of the Chest X-ray dataset, Wang et al. presented a unified weakly-supervised multi-label image classification framework by tailoring deep CNN (DCNN) architectures. Based on clinical taxonomies, Chen et al. presented a deep hierarchical multi-label classification (HMLC) approach for CXR CAD, and the abnormality labels were a constructed hierarchy from the PLCO dataset. Baltruschat et al. investigated a powerful network architecture, ResNet, in detail for multi-label CXR classification, and proved that X-ray-specific ResNet-38, integrating non-image data, yields the best overall results. Following the ML-GCN, Chen et al. proposed a similar architecture called CheXGCN that utilized the GCN as the classifier branch to explore the dependencies between pathologies for CXRs. Zhang et al. built a knowledge graph based on prior knowledge and applied the GCN to abnormal diagnosis for CXRs.

Title: Towards Surgical Tools Detection and Operative Skill Assessment Based on Deep Learning
Year:2022
Author: **Feng Li's**

Surgical tool detection and automatic operation skill assessment (AOSA) have important and extensive application scenarios in minimally invasive surgery (MIS). However, most of the deep learning methods currently used for surgical tool detection cannot achieve a good balance between speed and accuracy. We propose a new real-time detection algorithm.

Title: A framework based on the encoder-decoder architecture equipped with the multi-attention mechanism.
Year: 2022
Author: Kaur and Mittal

The idea is to compress the model without compromising its accuracy. First of all, the encoder is trained in a multi-task manner where they studied the effect of attending both visual and semantic features simultaneously. Given an input image I, we extract the global features using a features extractor f, the output is then fed into two branches, where a multilabel classification followed by a semantic attention module is applied in one branch. while, a visual attention module is applied in the second branch. the two outputs are then concatenated giving in a joint contect vector. The benefit of using both visual and semantic modules is to allow the language model to decide where and when to focus while generating the report.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

- The existing system of CXR image has the challenges, such as, handling variability in imaging conditions, dealing with rare or complex cases, and ensuring the generalization of models to diverse patient populations remain areas of active research.
- Difficult to categorize the type of disease in some unknown cases.

3.1.1 Disadvantages of Existing System

- Limited Accuracy
- Difficulty with Rare or Complex Cases
- Interpretability and Trustworthiness

3.2 PROPOSED SYSTEM

- To improve the quality of analyzing the CXR image, the Deep Learning Technology, such as CNN (Convolutional Neural Network) is used. The CNN effectively helps to segregate the image and deals with the complex and rare cases. The preprocessing, training, validation and deployment stages in the CNN, INCEPTION V3 and SVM technology will correctly diagnose the problem and shows the accurate results.

3.2.1 Advantages of Proposed System

- Improved Accuracy and Sensitivity
- Automation and Efficiency
- Consistency and Standardization

CHAPTER 4
SYSTEM SPECIFICATION

4.1 SOFTWARE SPECIFICATIONS

• OS	:	Windows
• Python IDE	:	Python 3.7.0

4.2 HARDWARE SPECIFICATIONS

• RAM	:	4GB and Higher
• Processor	:	Intel i3 and above
• Hard Disk	:	500GB: Minimum

4.3 SOFTWARE DESCRIPTION

Python:

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python. Python consistently ranks as one of the most popular programming languages.

Deep learning:

Deep learning is based on the branch of machine learning, which is a subset of artificial intelligence. Since neural networks imitate the human brain and so deep learning will do. In deep learning, nothing is programmed explicitly. Basically, it is a machine learning class that makes use of numerous nonlinear processing units so as to perform feature extraction as well as transformation. The output from each preceding layer is taken as input by each one of the successive layers. Deep learning models are capable enough to focus on the accurate features themselves by requiring a little guidance from the programmer and are very helpful in solving out the problem of dimensionality. Deep learning algorithms are used, especially when we have a huge no of inputs and outputs. Since deep learning has been evolved by the machine learning, which itself is a subset of artificial intelligence and as the idea behind the artificial intelligence is to mimic the human behavior, so same is "the idea of deep learning to build such algorithm that can mimic the brain". Deep learning is implemented with the help of Neural Networks, and the idea behind the motivation of Neural Network is the biological neurons, which is nothing but a brain cell.

Overview of Deep learning

Deep learning is a subfield of machine learning that deals with algorithms inspired by the structure and function of the brain. Deep learning is a subset of machine learning, which is a part of artificial intelligence (AI). Artificial intelligence is the ability of a machine to imitate intelligent human behavior.

CONVOLUTIONAL NEURAL NETWORK (CNN):

Convolutional neural network (CNN), a class of artificial neural networks that has become dominant in various computer vision tasks, is attracting interest across a variety of domains, including radiology. CNN is designed to automatically and adaptively learn spatial hierarchies of features through back propagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers. This review article offers a perspective on the basic concepts of CNN and its application to various radiological tasks, and discusses its challenges and future directions in the field of radiology. Two challenges in applying CNN to radiological tasks, small dataset and over fitting, will also be covered in this article, as well as techniques to minimize them. Being familiar with the concepts and advantages, as well as limitations, of CNN is essential to leverage its potential in diagnostic radiology, with the goal of augmenting the performance of radiologists and improving patient care.

Characteristics of CNN

CNN implementing the following features

- 3D volumes of neurons.
- The layers of a CNN have neurons arranged in 3 dimensions: width, height and depth.
- Where each neuron inside a convolutional layer is connected to only a small region of the layer before it, called a receptive field.

Overview of CNN

In neural networks, CNN is a unique family of deep learning models. CNN is a major artificial visual network for the identification of medical image patterns. The family of CNN primarily emerges from the information of the animal visual cortex. The major problem within a fully connected feed-forward neural network is that even for shallow architectures, the number of neurons may be very high, which makes them impractical to apply to image applications. The CNN is a method for reducing the number of parameters, allows a network to be deeper with fewer parameters. CNN’s are designed based on three architectural ideas that are shared weights, local receptive fields, and spatial sub-sampling . The essential element of CNN is the handling of unstructured data through the convolution operation. Convolution of the input signal $x(t)$ with filter signal $h(t)$ creates an output signal $y(t)$ that may reveal more information than the input signal itself. 1D convolution of a discrete signals $x(t)$ and $h(t)$ is the max-pooling method chooses the most superior invariant feature in a pooling region. The average pooling method selects the average of all the features in the pooling area. Thus, the max-pooling method holds texture information that can lead to faster convergence, average pooling method is called Keep background information. Spatial pyramid pooling, stochastic polling, Def-pooling, Multi activation pooling and detailed preserving pooling are different pooling techniques in the literature. A fully connected layer is used at the end of the CNN model. Fully connected layers perform like a traditional neural network. The input to this layer is a vector of numbers (output of the pooling layer) and outputs an N-dimensional vector (N number of classes). After the pooling layers, the feature of previous layer maps is flattened and connected to fully connected layers.

The first successful seven-layered LeNet-5 CNN was developed by Yann LeCunn in 1990 for handwritten digit recognition successfully. Krizhevsky et al. proposed AlexNet is a deep convolutional neural network composed of 5 convolutional and 3 fully-connected layers. In AlexNet changed the sigmoid activation function to a ReLU activation function to make model training easier. K. Simonyan and A. Zisserman invented the VGG-16 which has 13 convolutional and 3 fully connected layers. The Visual Geometric Group (VGG) research group released a series of CNN starting from VGG-11, VGG-13, VGG-16, and VGG-19. The main intention of the VGG group to understand how the depth of convolutional networks affects the accuracy of the models of image classification and recognition. Compared to the maximum VGG19, which has 16 convolutional layers and 3 fully connected layers, the minimum VGG11 has 8 convolutional layers and 3 fully connected layers. The last three fully connected layers are the same as the various variations of VGG. Szegedy et al. proposed an image classification network consisting of 22 different layers, which is Google Net. The main idea behind Google Net is the introduction of inception layers. Each inception layer convolves the input layers partially using different filter sizes. Kaiming He et al. proposed the ResNet architecture, which has 33 convolutional layers and one fully-connected layer. Many models introduced the principle of using multiple hidden layers and extremely deep neural networks, but then it was realized that such models suffered from the issue of vanishing or exploding gradients problem. For eliminating vanishing gradients' problem skip layers (shortcut connections) are introduced. Dense Net developed by Gao et al. consists of several dense blocks and transition blocks, which are placed between two adjacent dense blocks. The dense block consists of three layers of batch normalization, followed by a ReLU and a 3×3 convolution operation. The transition blocks are made of Batch Normalization, 1×1 convolution, and average Pooling. Compared to state-of-the-art handcrafted feature detectors, CNNs is an efficient technique for detecting features of an object and achieving good classification performance. There are drawbacks to CNNs, which are that unique relationships, size, perspective, and orientation of features are not taken into account. To overcome the loss of information in CNNs by pooling operation Capsule Networks (Caps Net) are used to obtain spatial information and most significant features. The special type of neurons, called capsules, can detect efficiently distinct information. The capsule network consists of four main components that are matrix multiplication, Scalar weighting of the input, dynamic routing algorithm, and squashing function.

INCEPTION V3:

Inception v3 is a convolutional neural network for assisting in image analysis and object detection, and got its start as a module for Google Net. It is the third edition of Google's Inception Convolutional Neural Network, originally introduced during the ImageNet Recognition Challenge. Just as Image Net can be thought of as a database of classified visual objects, Inception helps classification of objects[3] in the world of computer science. The Inceptionv3 architecture has been reused in many different applications, often used "pre-trained" from ImageNet. Inception v3 is a convolutional neural network (CNN) architecture designed for image recognition and classification tasks. It was developed by Google and released in 2015 as part of the Inception architecture series.

Characteristics of INCEPTION V3

- **Architecture**: Inception v3 utilizes a deep neural network architecture with multiple layers. It consists of a series of convolutional, pooling, and fully connected layers.
- **Inception Modules**: The key innovation of Inception v3 lies in its use of "Inception modules," which are modules containing multiple parallel convolutional layers of different filter sizes. This allows the network to capture features at different scales efficiently.
- **Factorization**: Inception v3 uses factorization techniques to reduce the computational cost of convolutions while maintaining expressive power. This includes techniques like factorizing convolutions into smaller convolutions, which helps in reducing parameters and computation.

- **Auxilary Classifiers**: Inception v3 introduces auxiliary classifiers at intermediate layers of the network during training. These auxiliary classifiers help in combating the vanishing gradient problem and provide additional regularization.
- **Batch Normalization**: Batch normalization is applied throughout the network, which helps in stabilizing and accelerating the training process by normalizing the inputs to each layer.
- **Global Average Pooling**: Instead of using fully connected layers at the end, Inception v3 employs global average pooling, which reduces overfitting and computation. This pooling operation computes the average of each feature map and directly feeds it into the softmax layer for classification.
- **Pre-Trained Model**: Inception v3 is often used as a pre-trained model for transfer learning tasks. Pre-trained models trained on large-scale datasets such as ImageNet are available, allowing users to fine-tune the network for specific tasks with smaller datasets.
- **Performance**: Inception v3 achieves high accuracy on various image classification benchmarks while being computationally efficient compared to earlier versions of the Inception architecture.

Overview of INCEPTION v3

Inception v3 builds upon the foundational concepts introduced in the Google net architecture, enhancing its capabilities in terms of accuracy and computational efficiency. Inception v3 is adept at assisting in image analysis tasks, including object detection and classification. Its architecture is optimized for detecting and recognizing various visual objects within images. Inception v3, like its predecessors, was developed and fine-tuned using large-scale datasets such as ImageNet, which contains labeled images spanning thousands of object categories. The network's performance was evaluated in competitions such as the ImageNet Challenge, where it demonstrated state-of-the-art accuracy in image classification tasks. Inception v3 models trained on datasets like ImageNet are often made available for reuse. These pre-trained models serve as a starting point for transfer learning, enabling developers and researchers to leverage the learned representations for their specific image recognition tasks with relatively small amounts of labeled data. Inception v3 has found widespread applications across various domains within computer vision, including but not limited to image classification, object detection, visual recognition in autonomous vehicles, medical image analysis, and more. Its versatility and performance make it a popular choice for diverse image-related tasks. Inception v3 incorporates innovative architectural elements, such as inception modules with factorized convolutions, auxiliary classifiers, batch normalization, and global average pooling. These design choices contribute to the network's ability to capture and analyze complex visual features effectively while maintaining computational efficiency. In summary, Inception v3 represents a significant milestone in the development of convolutional neural network architectures for image analysis and recognition tasks. Its versatility, efficiency, and high performance have cemented its status as a go-to solution for various computer vision applications.

SUPPORT VECTOR MACHINE (SVM):

Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships. SVM algorithms are very effective as we try to find the maximum separating hyperplane between the different classes available in the target feature. Support Vector Machine (SVM) is Supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it’s best suited for classification. The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Characteristics of SVM

- **Hyperplane**: In SVM, a hyperplane is a decision boundary that separates data points belonging to different classes. The goal of SVM is to find the hyperplane with the maximum margin, i.e., the maximum distance between the hyperplane and the nearest data points from each class.
- **Margin**: The margin is the distance between the hyperplane and the nearest data points from each class. SVM aims to maximize this margin to achieve better generalization and robustness to new data points.
- **Support Vectors**: Support vectors are the data points that lie closest to the decision boundary (hyperplane). These points are crucial in determining the optimal hyperplane because they define the margin.
- **Kernel Trick**: SVM can efficiently handle non-linearly separable data by mapping the input features into a higher-dimensional space using a kernel function. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid. The choice of kernel depends on the nature of the data and the problem at hand.
- **Regularization Parameter (C)**: SVM has a regularization parameter (C) that controls the trade-off between maximizing the margin and minimizing the classification error on the training data. A small value of C leads to a larger margin but may misclassify some training examples, while a large value of C may lead to overfitting by giving more importance to individual data points.
- **Binary Classifier**: SVM is inherently a binary classifier, meaning it classifies data into two classes. However, it can be extended to handle multi-class classification tasks using techniques like one-vs-one or one-vs-all strategies.
- **Loss Function**: SVM uses a hinge loss function, which penalizes misclassifications by an amount proportional to the distance from the margin. This loss function encourages the correct classification of data points while maximizing the margin.

Overview of SVM

SVM operates by finding the optimal hyperplane that best separates classes in the feature space. In a binary classification scenario, this hyperplane is the decision boundary that maximizes the margin between the closest data points from each class, known as support vectors. Initially designed for linearly separable data, SVM aims to find a hyperplane that linearly separates the classes. It's highly effective when the classes are well-separated in the feature space. SVM can be extended to handle non-linearly separable data by mapping the input features into a higher-dimensional space using a kernel function. This transformation allows SVM to learn complex decision boundaries. SVM optimizes by maximizing the margin, which is the distance between the decision boundary and the nearest data points from each class. Maximizing the margin promotes better generalization and helps prevent overfitting. SVM includes a regularization parameter (C) that controls the trade-off between maximizing the margin and minimizing the classification error. A smaller C value allows for a wider margin but may lead to misclassification, while a larger C value may lead to overfitting. SVM uses the hinge loss function, which penalizes misclassifications based on their distance from the decision boundary. This loss function encourages SVM to prioritize correctly classifying data points close to the decision boundary. SVM optimization aims to find a global solution rather than a local one, ensuring that the decision boundary is optimal for the entire dataset. While SVM is inherently a binary classifier, it can be extended for multi-class classification using strategies like one-vs-one or one-vs-all. SVM typically uses only a subset of training data points (support vectors) in the decision function, making it memory-efficient and suitable for large datasets. SVM has been successfully applied in various domains, including text classification, image recognition, bioinformatics, finance, and medical diagnosis, showcasing its versatility and effectiveness. Overall, SVM is a versatile and widely used algorithm known for its ability to handle linear and non-linear classification tasks, its robustness to overfitting, and its effectiveness in high-dimensional spaces.

METHODOLOGY

In general, the proposed model can be divided into three components: encoder, decoder, and reward module, as shown in Figure 2. The encoder is comprised of two separate branches (CNN and MLC) that extract visual and semantic features for the decoder separately. The MLC branch, serving as a multi-label classification task, predicts common observations and other medical concepts for given images. These predicted medical labels are embedded in vectors and then inserted into the decoder.

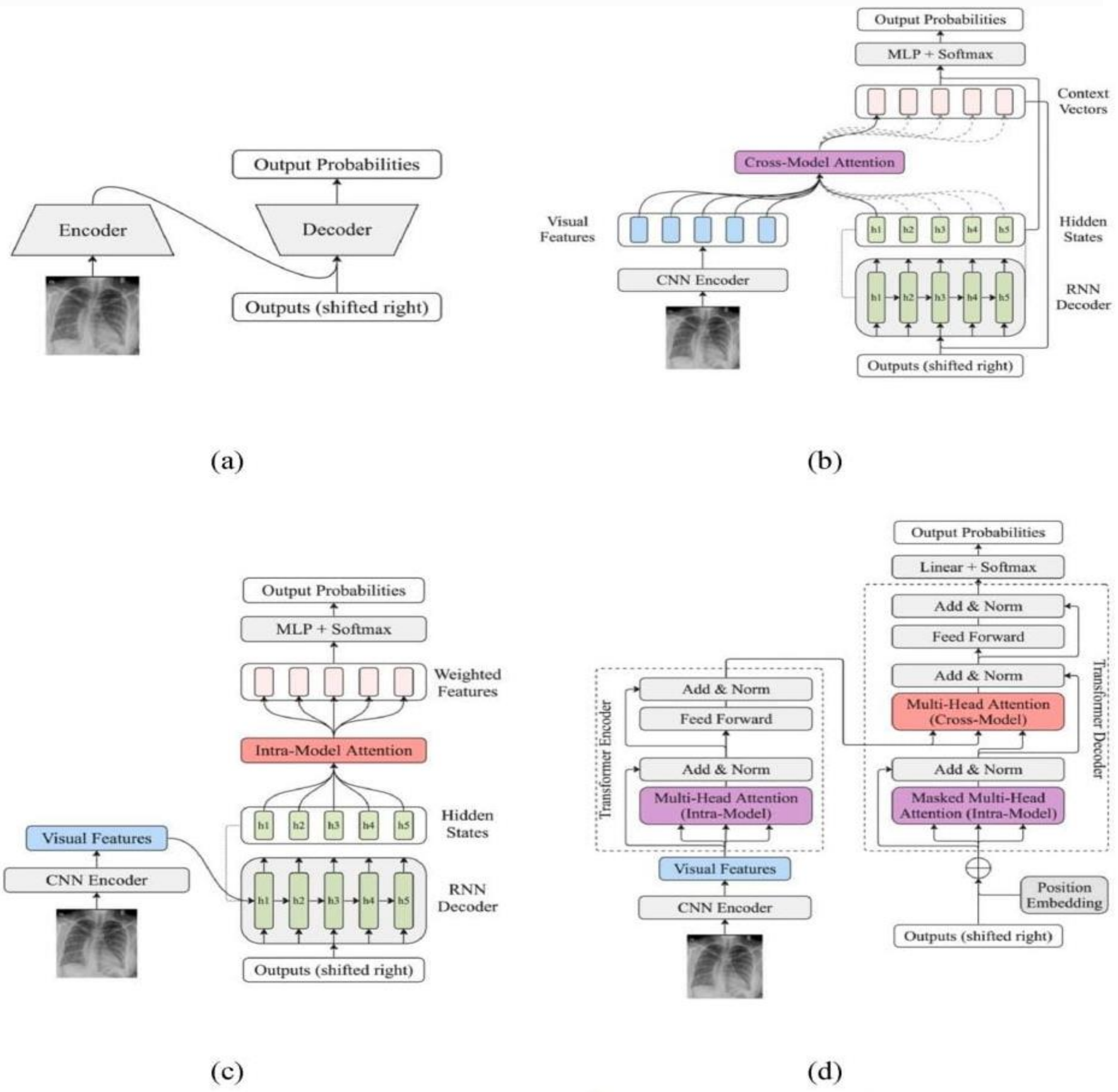


Fig 4.3.1 Encoding the x-ray images

Prior works have proved that incorporating semantic information can improve the performance of natural image captioning and medical-image-report generation. These semantic labels are predicted commonly by multi-label classification (MLC). In the studies of and, the extraction of visual and semantic features shares the same network, but in practice it is found that the MLC task is easier to learn than text generation; the shared network tends to learn the former task with higher priority and incurs lower loss, whereas the latter task performs worse.

1) VISUAL FEATURE EXTRACTION

The CNN branch is responsible for visual feature encoding. For one patient, there may exist one or more radiology images, representing different views. Yuan et al. conducted a multi-view fusion on deep features after the CNN backbone. However, an implicit and simpler approach is adopted here; that is, different views are treated as different input channels and it is assumed that the deep CNN can learn how to perform effective feature fusion during training.

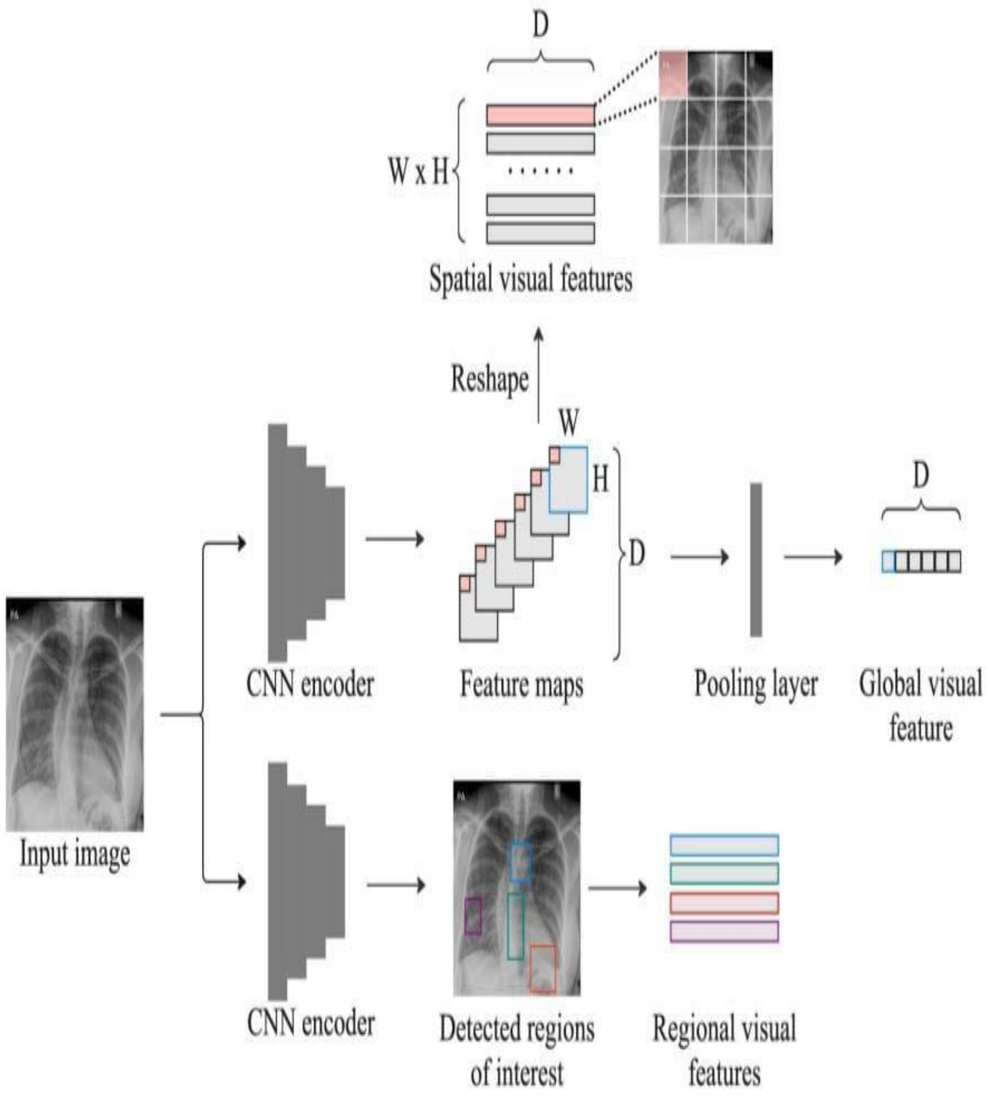


Fig 4.3.2 Layers in the encoding image

CHAPTER 5

SYSTEM DESIGN

5.1. SYSTEM ARCHITECTURE

A convolution is defined as an operation on two functions. In image analysis, one function consists of input values at a position in the image, and the second function is a filter each can be represented as array of numbers. Computing the dot product between the two functions gives an output. The filter is then shifted to the next position in the image as defined by the stride length. The computation is repeated until the entire image is covered, producing a feature map. This is a map of where the filter is strongly activated and 'sees' a feature such asa straight line, a dot, or a curved edge. If a photograph of a face was fed into a CNN, initially low-level features such as lines and edges are discovered by the filters. These build up to progressively higher features in subsequent layers, such as a nose, eye or ear, as the features maps become inputs for the next layer in the CNN architecture.

.

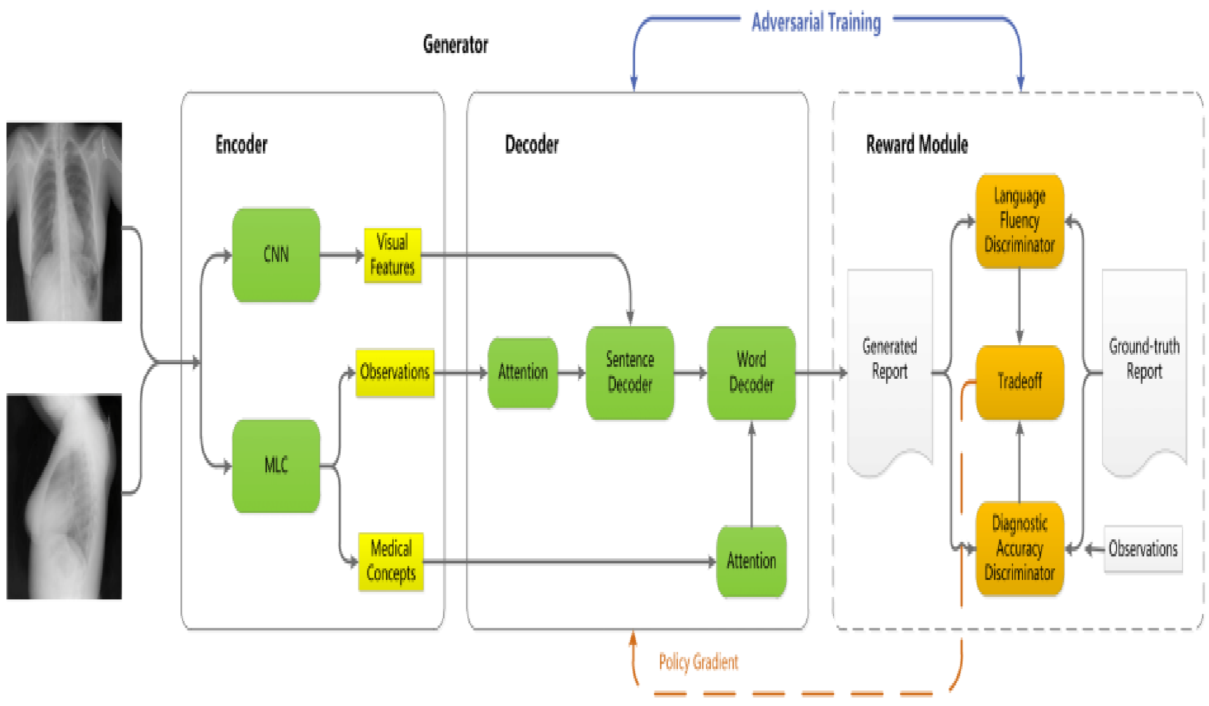


Fig 5.1 System architecture

5.2 FLOW DIAGRAM

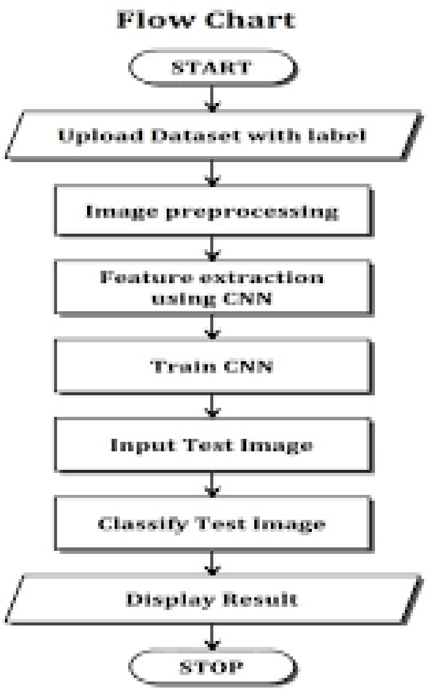


Fig 5.2.1 Flow chart diagram for report generation

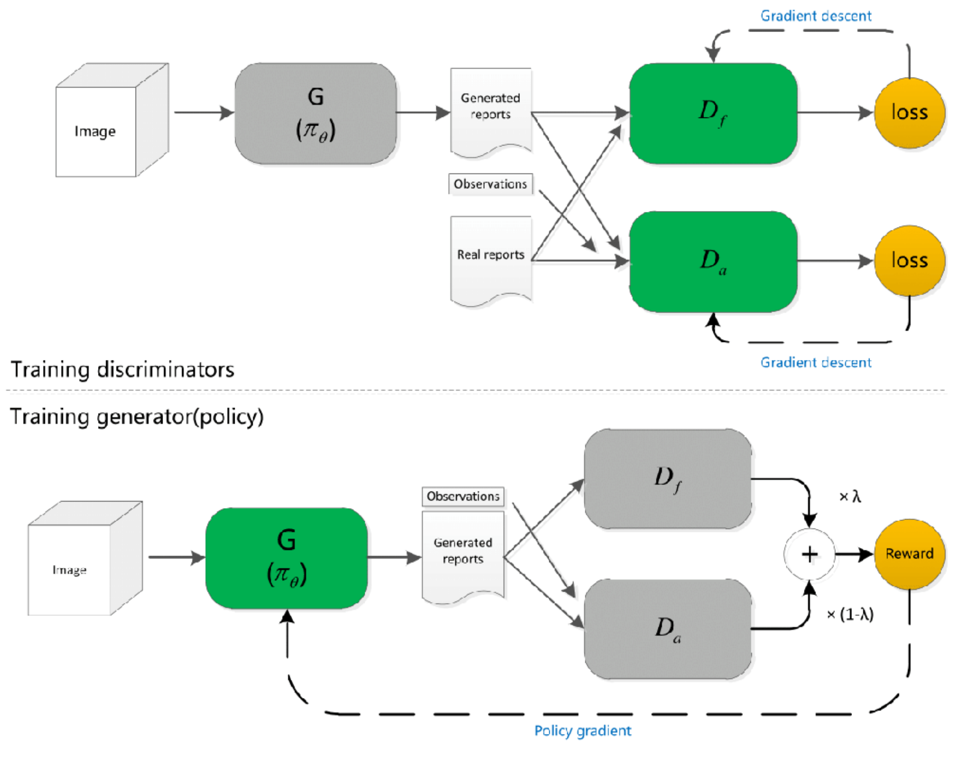


Fig 5.2.2 Training and loss function

5.3 USE CASE DIAGRAM

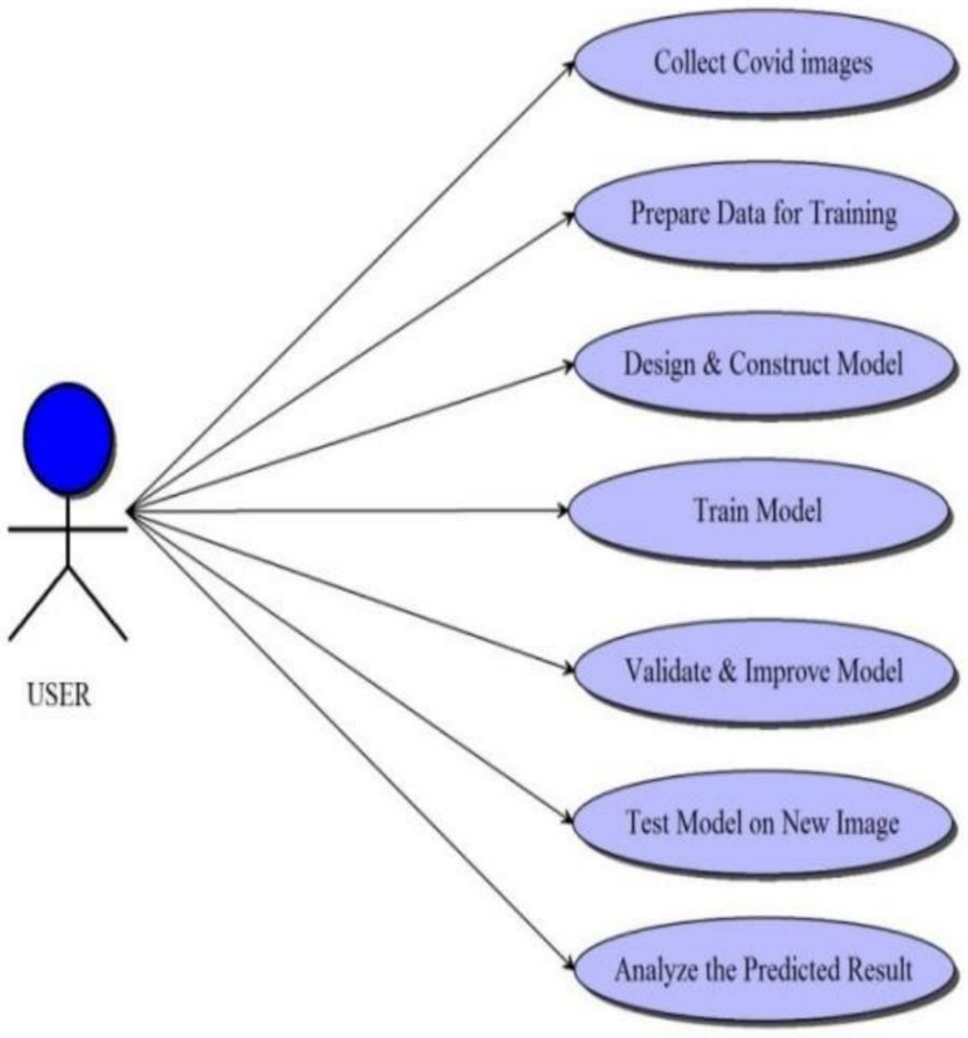


Fig 5.3.1 Use case diagram for report generation

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. In our use case diagram first Node deployment, Route discovery, Sleep/wake implementation, information collection and analysis.

5.4 SEQUENCE DIARGAM

Above Diagram tells us about the different sequence we are following to make a network formation to Performance evaluation. A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order.

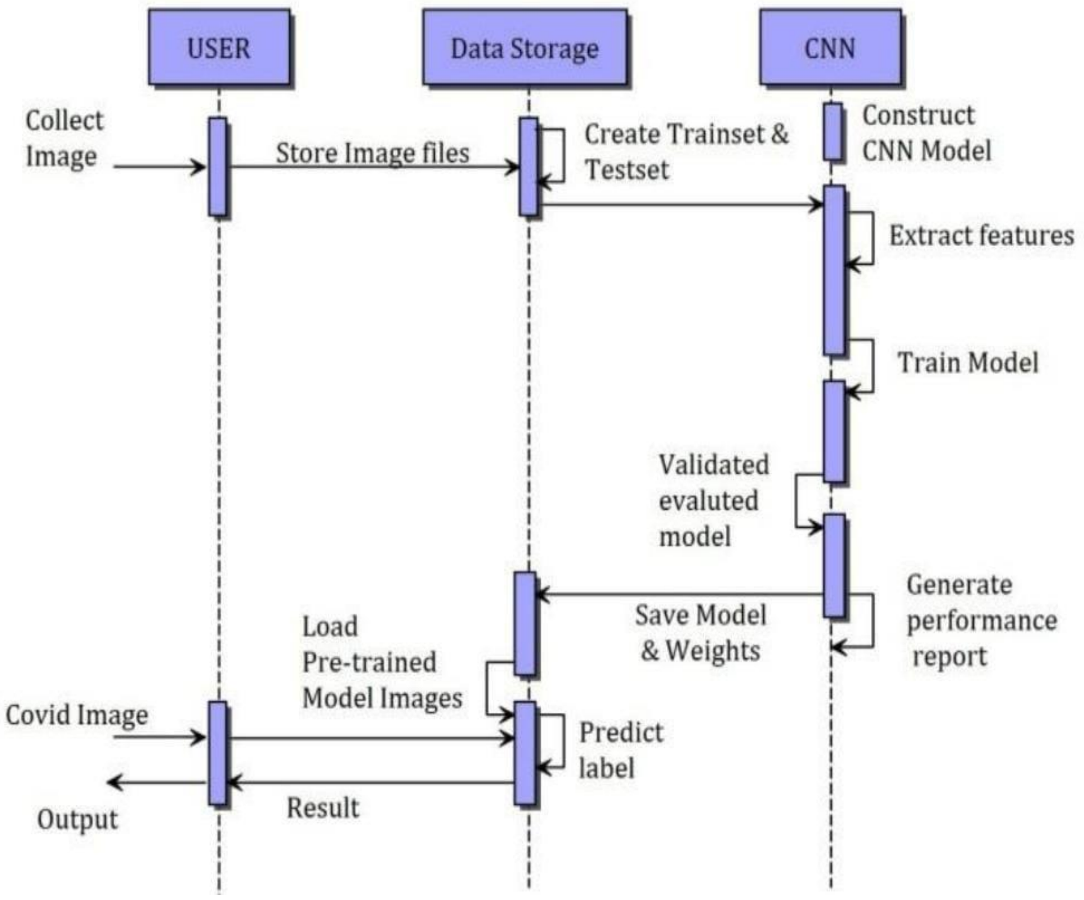


Fig 5.4.1 Sequence diagram for report generation

Class Activation Mapping

In ConvNet-based image classification models, class activation maps (CAM) are used to highlight the most relevant or discriminative portions of an image that are used by a model to identify disorders in chest X-rays. A class activation map serves as a visual explanation of a ConvNet model that can assist radiologists in determining whether the decisions made by the model are based on the processing of the correct features of chest X-ray images. Class activation maps can also assist in the detection of data bias.

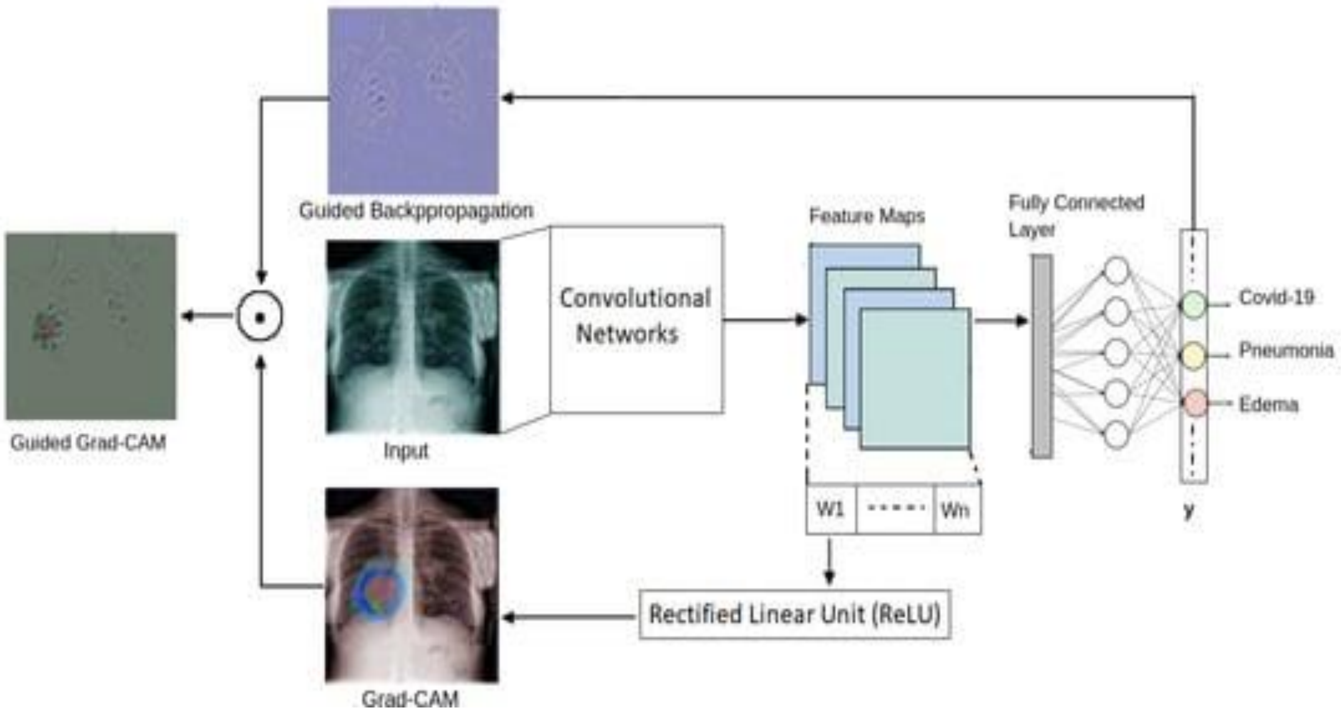


Fig 5.4.2 Activation mapping of layers

Attention-Based Explanation

In the field of deep learning, the concept of attention has attracted a lot of interest due to its powerful influence on the learning ability of deep neural networks. Studies have been conducted on developing attention-based models that can explain decisions made by neural network models, allowing humans to trust these decisions.

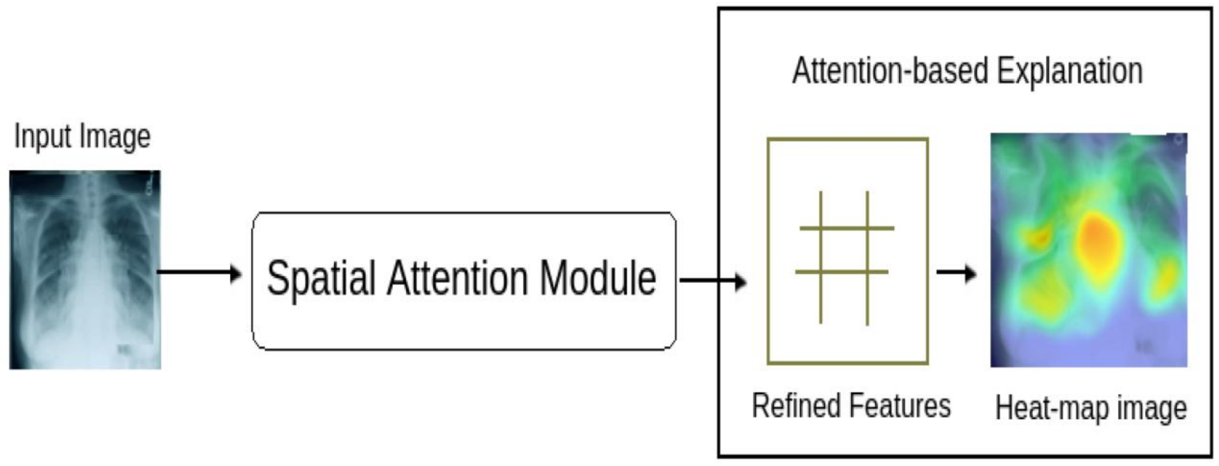


Fig 5.4.3 Attention based explanation of heat map image

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 MODULES

1. Upload X-RAY image dataset
2. Preprocessing & Feature Extraction
3. Generate Train & Test Model
4. Generate Deep Learning CNN Model
5. Get Drive HQ Images

6.1.1 Upload X-RAY image dataset

Using this module, we are uploading X-RAY train images and then application read all images and convert them grey format. This scan designed to support medical research and advancements in the field of health care. This dataset is a valuable resource for researchers, clinicians, and developers interested in leveraging scan data for various applications.

6.1.2 Preprocessing & Feature Extraction

Preprocessing and feature extraction are iterative processes and often require domain knowledge and experimentation to determine the most effective techniques for a given dataset and modeling task. Additionally, the choice of preprocessing and feature extraction techniques can significantly impact the performance of machine learning models.

6.1.3 Generate Train & Test Model

Using this module, we will build array of pixels with all images features and then split dataset into train and test model to calculate accuracy using test images by applying train model on it. Training and testing a machine learning model involves several steps, typically performed using a labelled dataset.

6.1.4 Generate Deep Learning Models

6.1.4.1 Generate Deep Learning CNN Model

Using this module, we will build array of pixels with all images features and then split dataset into train and test model to calculate accuracy using test images by applying train model on it**.**

6.1.4.2 Generate Deep Learning SVM Model

Using this module, we will build array of pixels with all images features and then split dataset into train and test model to calculate accuracy using test images by applying train model on it**.**

6.1.4.3 Generate Deep Learning INCEPTION V3 Model

Using this module, we will build array of pixels with all images features and then split dataset into train and test model to calculate accuracy using test images by applying train model on it**.**

6.1.5 Get Drive HQ Images

Using this module will input train and test data to auto stack CNN model to build training classifier.

CHAPTER 7

RESULTS AND DISCUSSION

7.1 INTRODUCTION:

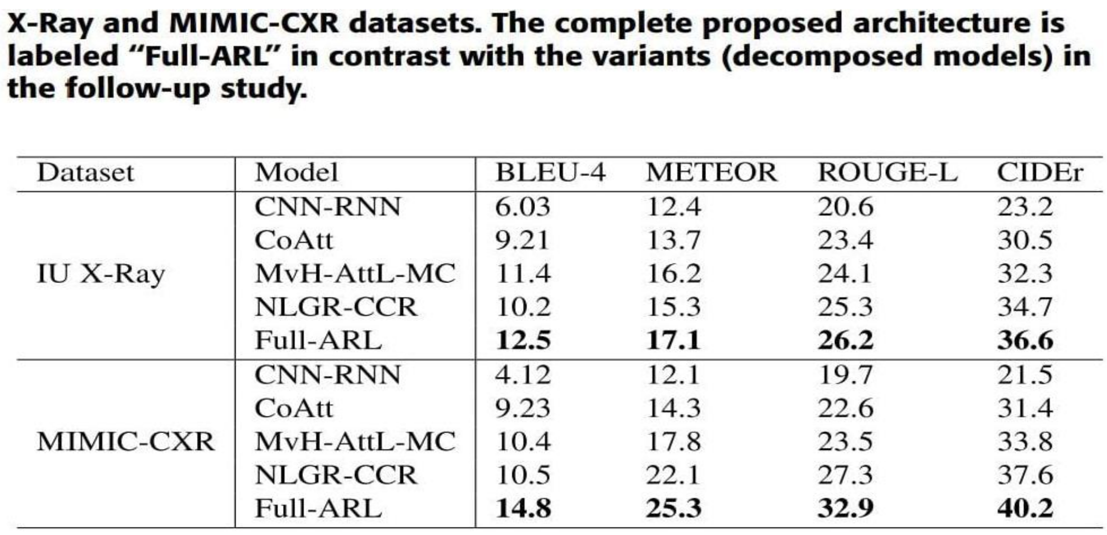
Introduced to optimize the generator. In adversarial training, the report generator is trained by RL viewing the overall score as the reward, and simultaneously the discriminators improve their judgment through maximum-likelihood estimation. The discriminators are the language fluency discriminator (FD) and diagnostic accuracy discriminator (AD), allowing for readability and accuracy, respectively. The FD checks how likely a report originates from a human expert, while the AD determines how much a report covers the ground-truth observations. To the best of our knowledge, this is the first introduction of ARL to report generation for medical images. In short, the main contributions of the proposed framework are summarized as follows.

7.2 PERFORMANCE MEASURES:

In this section, first all the models are separately trained on the two datasets. The training sets are used for parameter learning, validation sets for early stopping, and the test sets are invisible during training for conducting experiments. To ensure the comparability of experimental results, all the hyper-parameters are kept as consistent as possible.

7.3 PERFORMANCE EVALUTION:

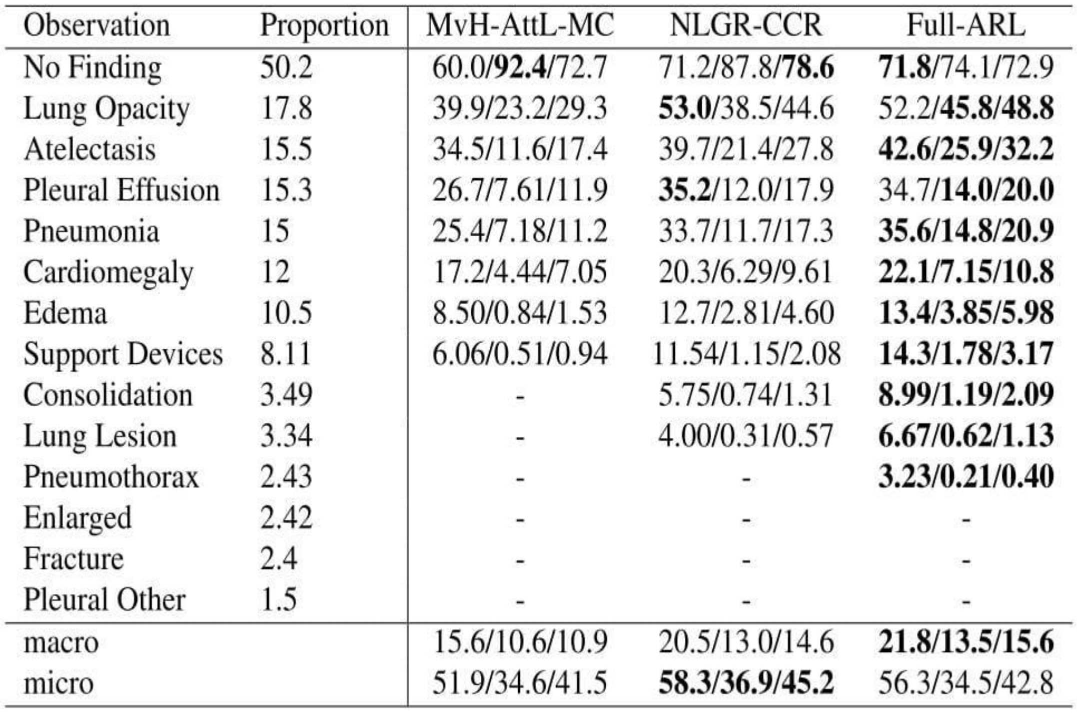
To fully evaluate the performance of the proposed model, several different metrics considering both language fluency and diagnostic accuracy are adopted. First, popular natural language generation (NLG) evaluation metrics, including BLEU-4, METEOR, ROUGE-L, and CIDEr, are adopted, which are used to measure the statistical correlation between two text sequences. In the experiments, the automatic tool3 is used to calculate these metrics. Second, to measure the diagnostic accuracy of generated reports, the CheXpert labeler is applied to the generated reports, and the precision, recall, and F1 scores are calculated.


X-Ray and MIMIC-CXR datasets. The complete proposed architecture is labeled "Full-ARL" in contrast with the variants (decomposed models) in the follow-up study.

Dataset	Model	BLEU-4	METEOR	ROUGE-L	CIDEr
IU X-Ray	CNN-RNN	6.03	12.4	20.6	23.2
	CoAtt	9.21	13.7	23.4	30.5
	MvH-AttL-MC	11.4	16.2	24.1	32.3
	NLGR-CCR	10.2	15.3	25.3	34.7
	Full-ARL	**12.5**	**17.1**	**26.2**	**36.6**
MIMIC-CXR	CNN-RNN	4.12	12.1	19.7	21.5
	CoAtt	9.23	14.3	22.6	31.4
	MvH-AttL-MC	10.4	17.8	23.5	33.8
	NLGR-CCR	10.5	22.1	27.3	37.6
	Full-ARL	**14.8**	**25.3**	**32.9**	**40.2**

Fig 7.3.1 CXR datasets

7.3.2 TABLE 2: Diagnostic accuracy evaluation results of different methods on MIMIC-CXR dataset. Second column lists the proportion (%) of each observation, and starting from the next column, it exhibits precision/recall/F1 (%) for different methods across all observations, where ''–'' denotes that precision and recall equal zero, and thus F1 score is undefined. Note that the word ''Enlarged'' in the observation column is short for ''Enlarged cardio mediastinum'' since this phrase is too long to display the data



Observation	Proportion	MvH-AttL-MC	NLGR-CCR	Full-ARL
No Finding	50.2	60.0/**92.4**/72.7	71.2/87.8/**78.6**	**71.8**/74.1/72.9
Lung Opacity	17.8	39.9/23.2/29.3	**53.0**/38.5/44.6	52.2/**45.8/48.8**
Atelectasis	15.5	34.5/11.6/17.4	39.7/21.4/27.8	**42.6/25.9/32.2**
Pleural Effusion	15.3	26.7/7.61/11.9	**35.2**/12.0/17.9	34.7/**14.0/20.0**
Pneumonia	15	25.4/7.18/11.2	33.7/11.7/17.3	**35.6/14.8/20.9**
Cardiomegaly	12	17.2/4.44/7.05	20.3/6.29/9.61	**22.1/7.15/10.8**
Edema	10.5	8.50/0.84/1.53	12.7/2.81/4.60	**13.4/3.85/5.98**
Support Devices	8.11	6.06/0.51/0.94	11.54/1.15/2.08	**14.3/1.78/3.17**
Consolidation	3.49	-	5.75/0.74/1.31	**8.99/1.19/2.09**
Lung Lesion	3.34	-	4.00/0.31/0.57	**6.67/0.62/1.13**
Pneumothorax	2.43	-	-	**3.23/0.21/0.40**
Enlarged	2.42	-	-	-
Fracture	2.4	-	-	-
Pleural Other	1.5	-	-	-
macro		15.6/10.6/10.9	20.5/13.0/14.6	**21.8/13.5/15.6**
micro		51.9/34.6/41.5	**58.3/36.9/45.2**	56.3/34.5/42.8

Fig 7.3.2 Diagnostic accuracy evaluation results of CXR

Effect of CNN and MLC backbones on model performance.

CNN	MLC	AUC	CIDEr	F1
ResNet-152	VGG-16	**64.6**	**31.5**	**11.2**
VGG-16	VGG-16	**64.6**	29.7	10.4
ResNet-152	ResNet-152	62.1	30.4	9.8
VGG-16	ResNet-152	62.1	28.8	8.7

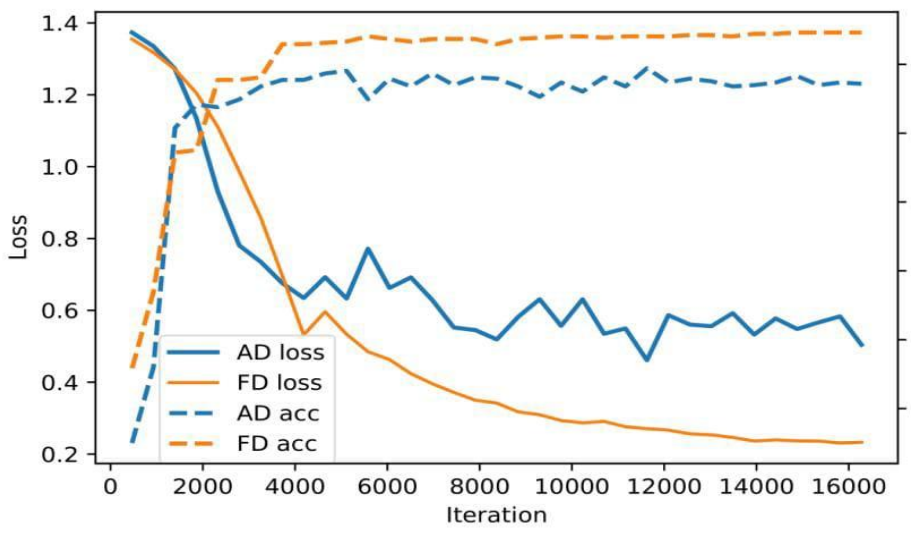
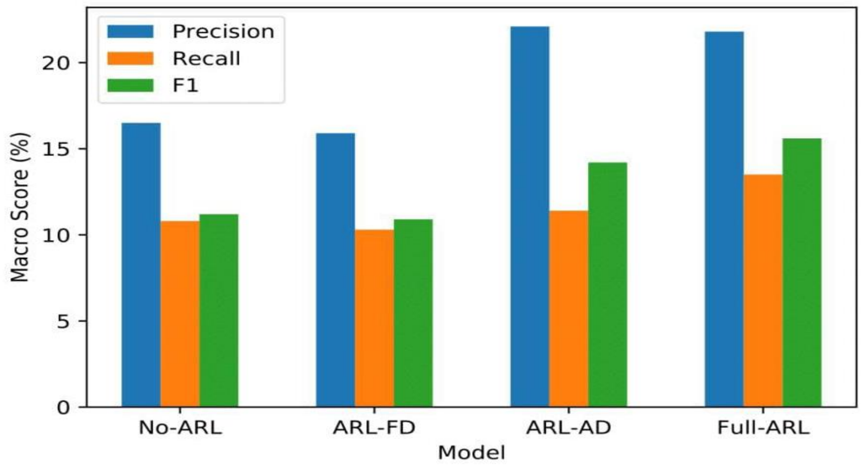
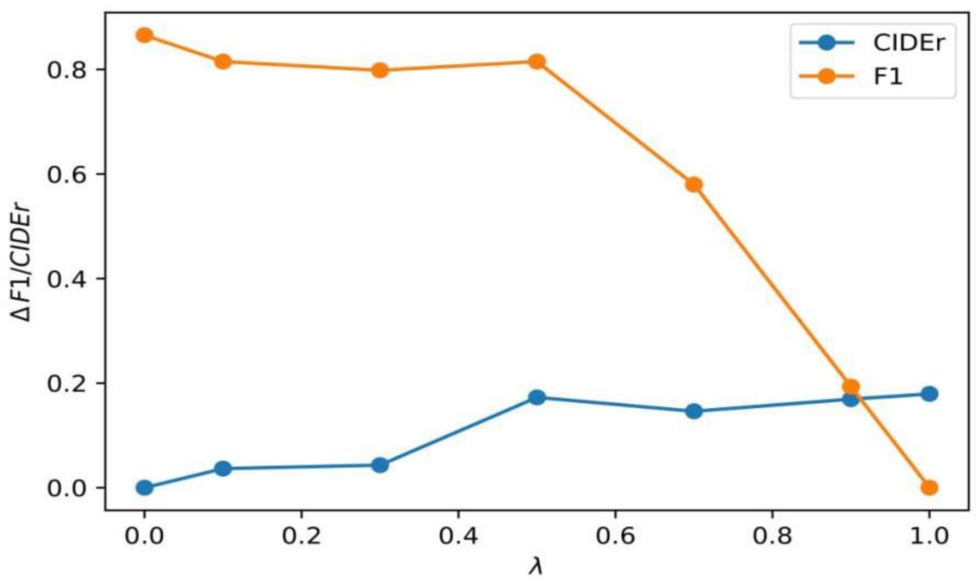


Fig 7.3.3 The other two metrics (CIDEr and F1) are for the generator



Macro precision/recall/F1 scores (%) of different variants of proposed model for observations. Full-ARL results are consistent with those in last row of Table 4.

Fig 7.3.4 Bar graph of the observations



Relative change rate of F1 and CIDEr with respect to λ on MIMIC-CXR dataset.

Fig 7.3.5 CIDEr AND F1 graph

7.4 CHALLENGES IN PNEUMONIA CLASSIFICATION

Developing an automatic report generation system for chest X-ray images via adversarial reinforcement learning (ARL) presents several challenges. One significant challenge is ensuring the accuracy and reliability of the generated reports. The model must accurately interpret complex medical images and translate them into clinically relevant and accurate textual descriptions. Achieving this requires addressing variations in image quality, anatomical structures, and pathology presentations, which can be highly diverse and nuanced. Another challenge is balancing between language fluency and diagnostic accuracy. While it's essential for the generated reports to be linguistically fluent and coherent, they must also convey precise and clinically relevant information. Balancing these objectives within the ARL framework requires careful design and optimization to avoid sacrificing one aspect for the other.

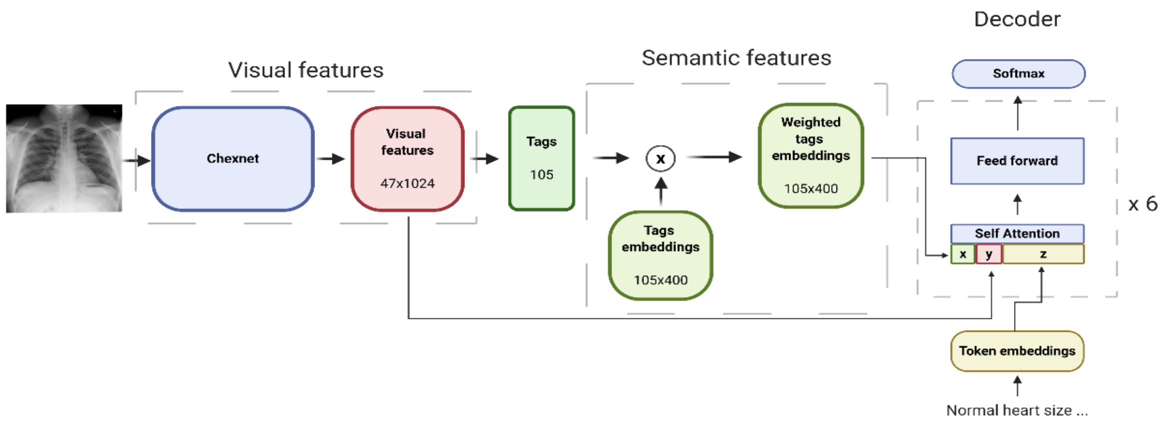
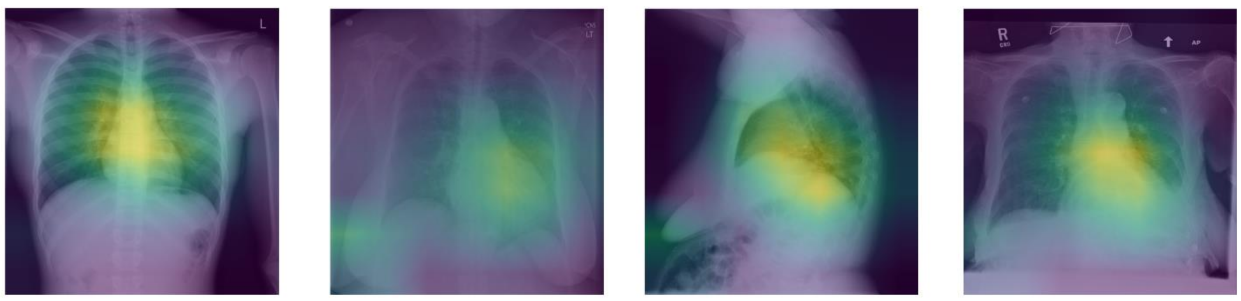
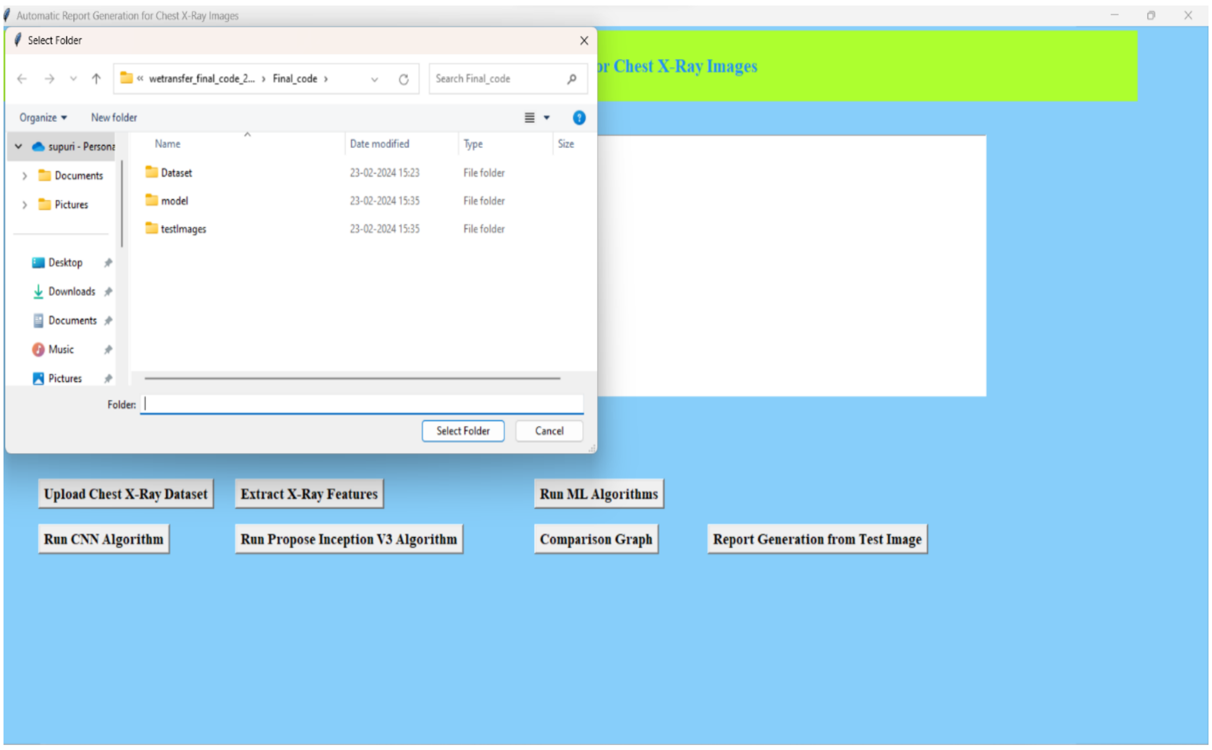


Fig: GPT2-Chest-X-Ray-Report-Generation (CDGPT2)



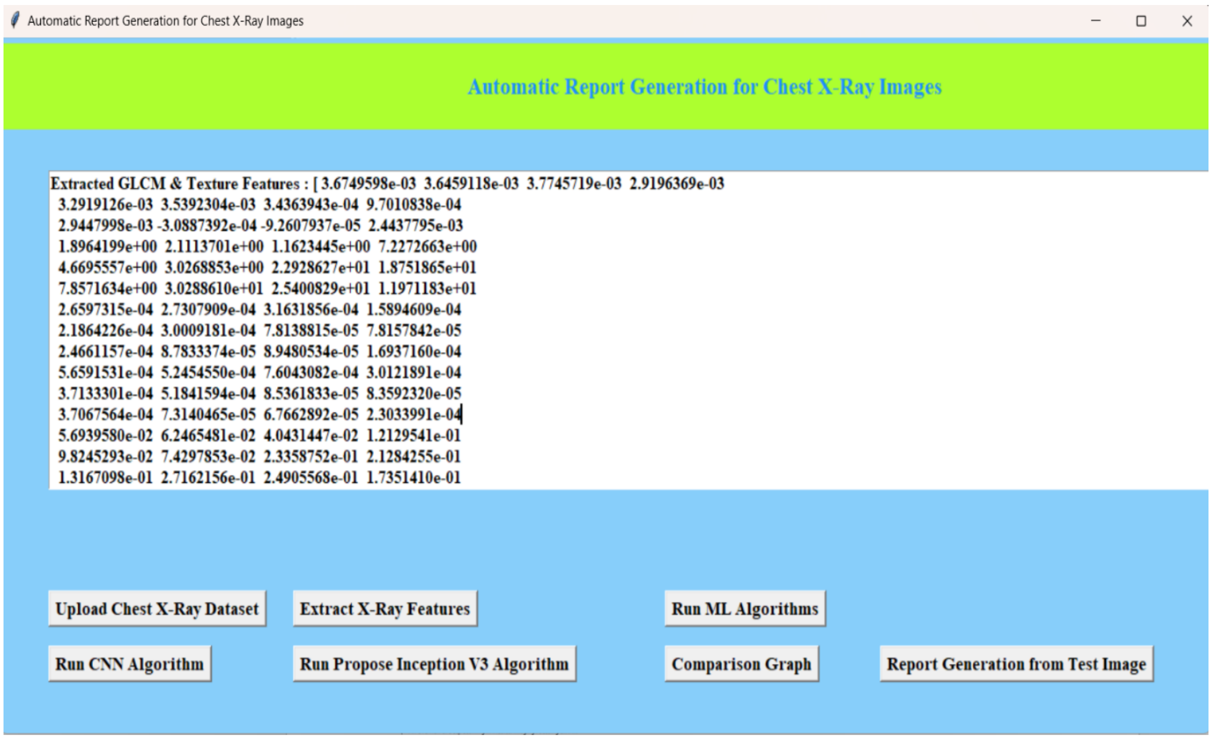
	Normal	Accurate	Missing Details	False
Ground Truth	Negative. The cardiomediastinal silhouette is normal in size and contour. No focal consolidation, pneumothorax or large pleural effusion. normal.	Stable cardiomegaly with mild pulmonary interstitial edema. Unchanged cardiomegaly. negative for pneumothorax or focal consolidation. No large effusion. Mildly prominent interstitial opacities.	Minimal left basilar atelectasis versus infiltrate. Low lung volumes. Normal cardiomediastinal contours. Low lung volumes with minimal left basilar opacities. No pneumothorax or pleural effusions.	Prominent hiatal hernia. left basilar opacity compatible pleural effusion and atelectasis. Right pleural effusion. no pulmonary edema / overt chf identified. stable senescent mediastinal contour.
Prediction	No evidence of active Disease. The heart size and pulmonary vascularity appear within normal limits. The lungs are free of focal airspace disease. No pleural effusion or pneumothorax is seen.	Cardiomegaly and mild interstitial pulmonary edema. Moderate cardiomegaly. bibasilar and perihilar interstitial opacities. No pneumothorax	No acute cardiopulmonary abnormality. The lungs are clear, and without focal air space opacity. The cardiomediastinal silhouette is normal in size and contour. There is no mediastinal contour, large pleural effusion, pneumothorax, or focal airspace consolidation.	Interval left subclavian central venous catheter with tip approximating the high svc. No evidence of pneumothorax. Generalized heart size and mediastinal contours appear within normal limits. Atherosclerotic changes of the aorta. Moderate degenerative changes of the thoracic spinea.

Fig: Sample predictions



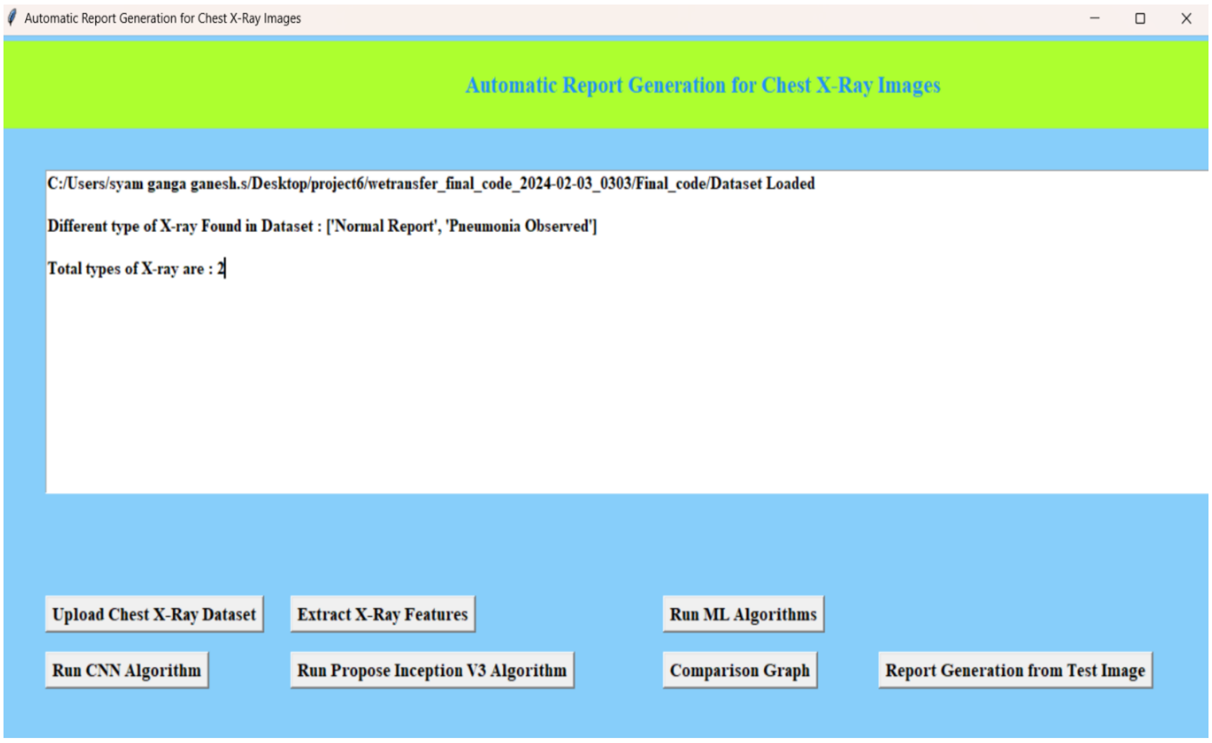
Name	Date modified	Type	Size
Dataset	23-02-2024 15:23	File folder	
model	23-02-2024 15:35	File folder	
testImages	23-02-2024 15:35	File folder	

Test Images


Automatic Report Generation for Chest X-Ray Images
Automatic Report Generation for Chest X-Ray Images
Upload Chest X-Ray Dataset
Extract X-Ray Features
Run ML Algorithms
Run CNN Algorithm
Run Propose Inception V3 Algorithm
Comparison Graph
Report Generation from Test Image

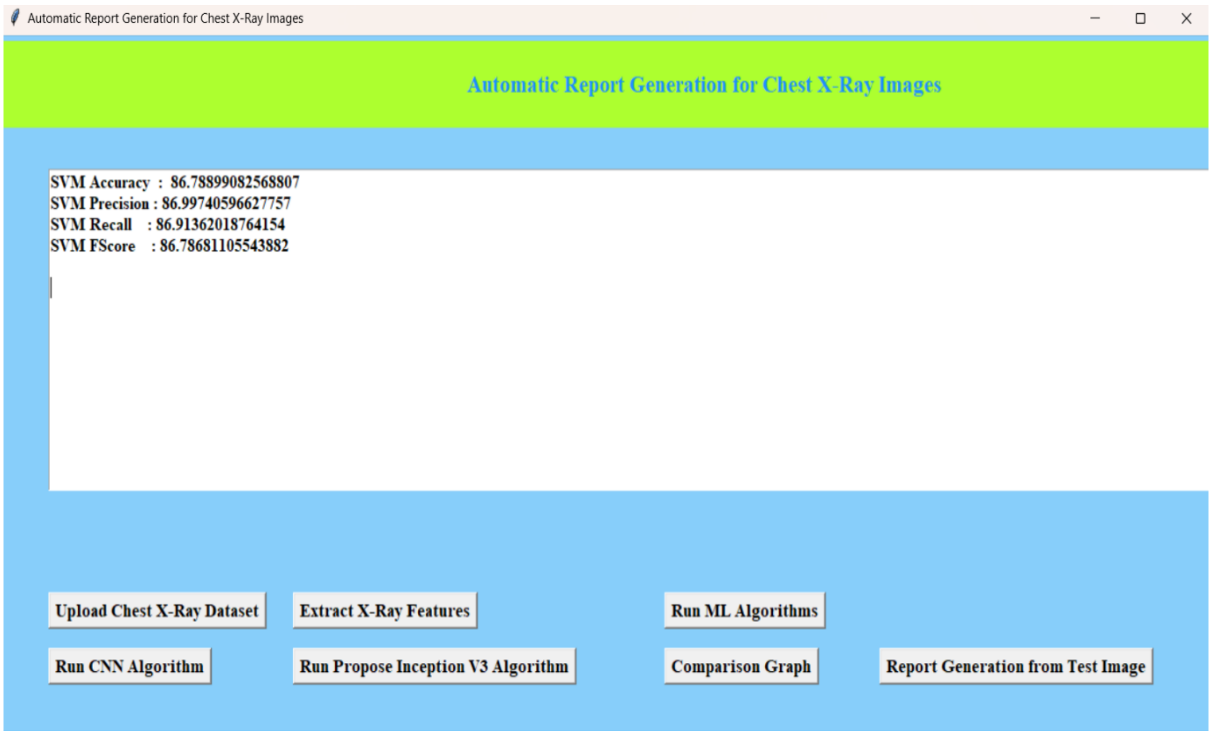
```
Extracted GLCM & Texture Features : [ 3.6749598e-03  3.6459118e-03  3.7745719e-03  2.9196369e-03
  3.2919126e-03  3.5392304e-03  3.4363943e-04  9.7010838e-04
  2.9447998e-03 -3.0887392e-04 -9.2607937e-05  2.4437795e-03
  1.8964199e+00  2.1113701e+00  1.1623445e+00  7.2272663e+00
  4.6695557e+00  3.0268853e+00  2.2928627e+01  1.8751865e+01
  7.8571634e+00  3.0288610e+01  2.5400829e+01  1.1971183e+01
  2.6597315e-04  2.7307909e-04  3.1631856e-04  1.5894609e-04
  2.1864226e-04  3.0009181e-04  7.8138815e-05  7.8157842e-05
  2.4661157e-04  8.7833374e-05  8.9480534e-05  1.6937160e-04
  5.6591531e-04  5.2454550e-04  7.6043082e-04  3.0121891e-04
  3.7133301e-04  5.1841594e-04  8.5361833e-05  8.3592320e-05
  3.7067564e-04  7.3140465e-05  6.7662892e-05  2.3033991e-04
  5.6939580e-02  6.2465481e-02  4.0431447e-02  1.2129541e-01
  9.8245293e-02  7.4297853e-02  2.3358752e-01  2.1284255e-01
  1.3167098e-01  2.7162156e-01  2.4905568e-01  1.7351410e-01
```

Upload Data sets


Automatic Report Generation for Chest X-Ray Images
Automatic Report Generation for Chest X-Ray Images
Upload Chest X-Ray Dataset
Extract X-Ray Features
Run ML Algorithms
Run CNN Algorithm
Run Propose Inception V3 Algorithm
Comparison Graph
Report Generation from Test Image

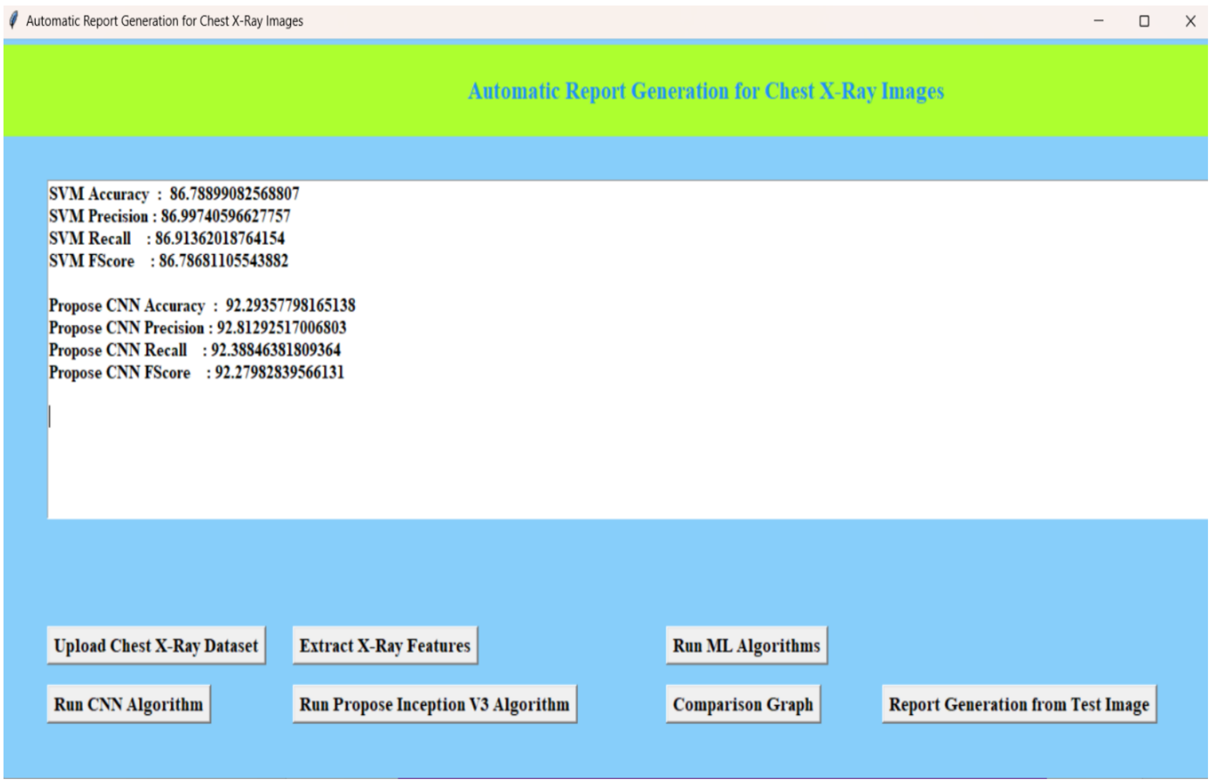
```
C:/Users/syam ganga ganesh.s/Desktop/project6/wetransfer_final_code_2024-02-03_0303/Final_code/Dataset Loaded

Different type of X-ray Found in Dataset : ['Normal Report', 'Pneumonia Observed']

Total types of X-ray are : 2
```

Extracting X-Ray Features


Automatic Report Generation for Chest X-Ray Images
Automatic Report Generation for Chest X-Ray Images
Upload Chest X-Ray Dataset
Extract X-Ray Features
Run ML Algorithms
Run CNN Algorithm
Run Propose Inception V3 Algorithm
Comparison Graph
Report Generation from Test Image

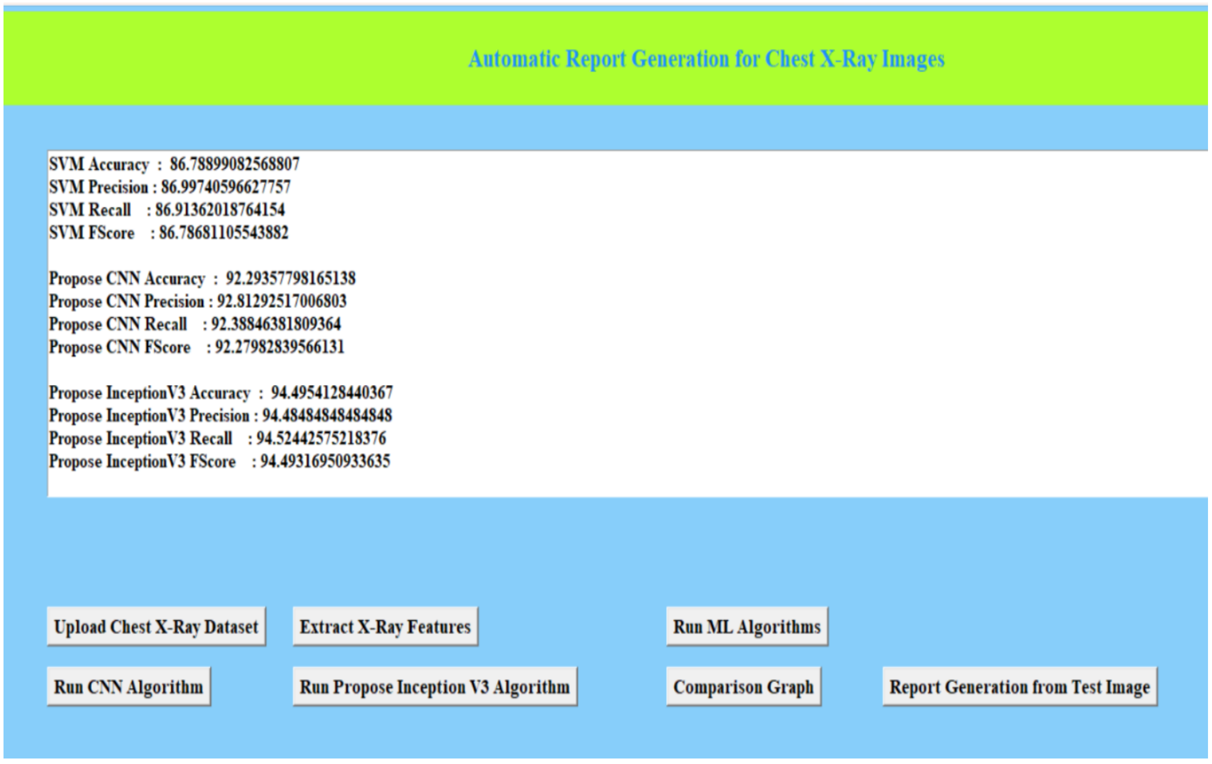
```
SVM Accuracy  :  86.78899082568807
SVM Precision : 86.99740596627757
SVM Recall    : 86.91362018764154
SVM FScore    : 86.78681105543882
```

Generate deep learning SVM model


Automatic Report Generation for Chest X-Ray Images
Automatic Report Generation for Chest X-Ray Images
Upload Chest X-Ray Dataset
Extract X-Ray Features
Run ML Algorithms
Run CNN Algorithm
Run Propose Inception V3 Algorithm
Comparison Graph
Report Generation from Test Image

```
SVM Accuracy  :  86.78899082568807
SVM Precision : 86.99740596627757
SVM Recall    : 86.91362018764154
SVM FScore    : 86.78681105543882

Propose CNN Accuracy  :  92.29357798165138
Propose CNN Precision : 92.81292517006803
Propose CNN Recall    : 92.38846381809364
Propose CNN FScore    : 92.27982839566131
```

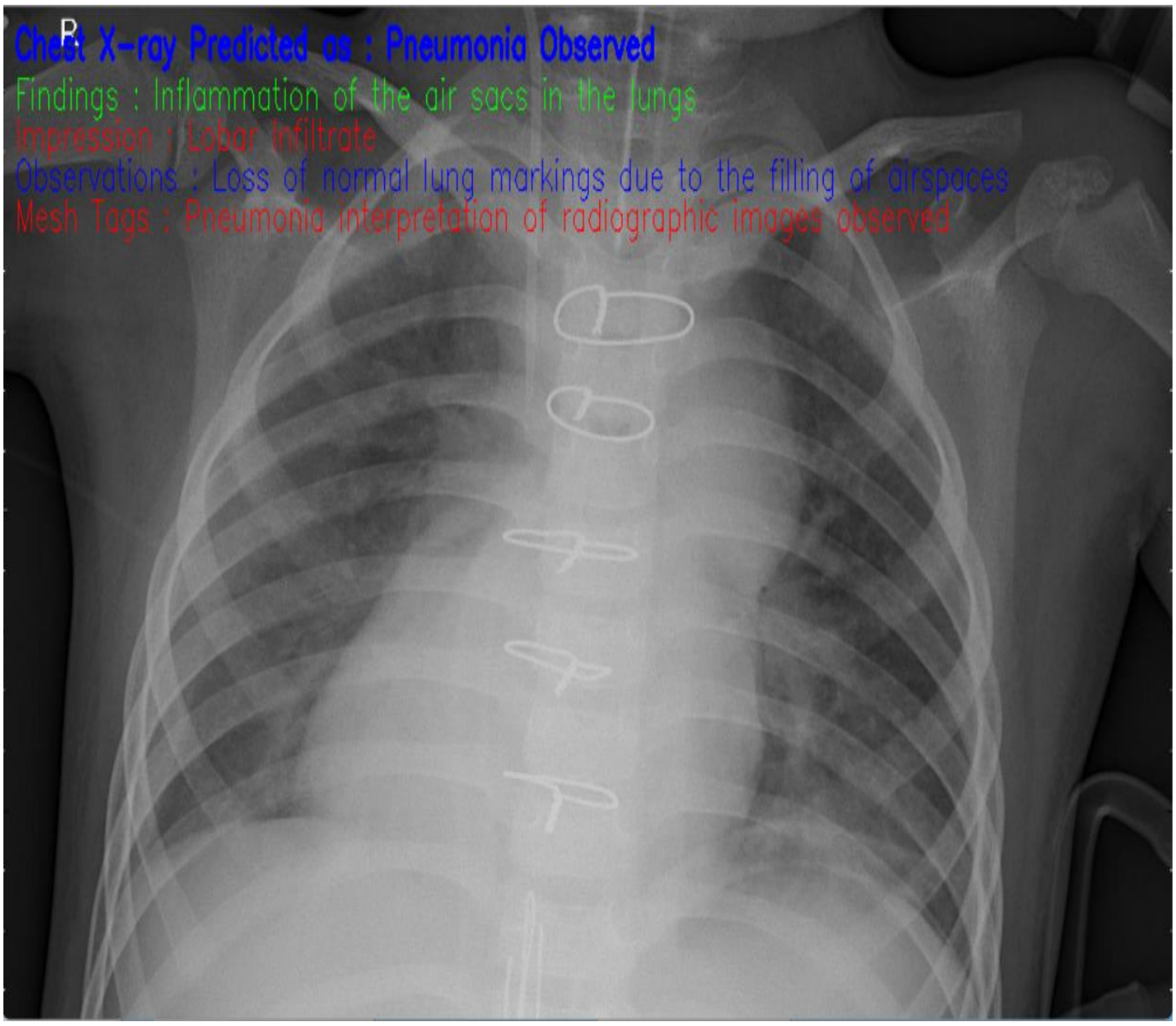
Generate deep learning CNN model


Automatic Report Generation for Chest X-Ray Images
Upload Chest X-Ray Dataset
Extract X-Ray Features
Run ML Algorithms
Run CNN Algorithm
Run Propose Inception V3 Algorithm
Comparison Graph
Report Generation from Test Image

```
SVM Accuracy  :  86.78899082568807
SVM Precision : 86.99740596627757
SVM Recall    : 86.91362018764154
SVM FScore    : 86.78681105543882

Propose CNN Accuracy  :  92.29357798165138
Propose CNN Precision : 92.81292517006803
Propose CNN Recall    : 92.38846381809364
Propose CNN FScore    : 92.27982839566131

Propose InceptionV3 Accuracy  :  94.4954128440367
Propose InceptionV3 Precision : 94.48484848484848
Propose InceptionV3 Recall    : 94.52442575218376
Propose InceptionV3 FScore    : 94.49316950933635
```

Generate deep learning INCEPTION V3

OUTPUT



Chest X-ray Predicted as : Pneumonia Observed
Findings : Inflammation of the air sacs in the lungs
Impression : Lobar Infiltrate
Observations : Loss of normal lung markings due to the filling of airspaces
Mesh Tags : Pneumonia interpretation of radiographic images observed

Generated output of the CXR image

CHAPTER 8
SYSTEM TESTING

8.1. TESTING OF PRODUCT

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to variety of tests-on-line response, Volume Street, recovery and security and usability test. A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

- **Unit Testing**

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’. The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module.

➢ **Integration Testing**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance.

➢ **Validation Testing**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, but a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

➢ **White Box Testing**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we derived test cases that guarantee that all independent paths within a module have been exercised at least once.

➢ **Black Box Testing**

- Black box testing is done to find incorrect or missing function
- Interface error
- Errors in external database access
- Performance errors
- Initialization and termination errors

In 'functional testing', is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called 'black box testing'. It tests the external behavior of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

- **Output testing:**

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

8.2. SOFTWARE IMPLEMENTATION

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. The people are not sure that the software is meant to make their job easier.

The active user must be aware of the benefits of using the system.

Their confidence in the software built up.

Proper guidance is impaired to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not running on the server, the actual processes will not take place.

- **User Training**

To achieve the objectives and benefits expected from the proposed system it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for education and training is more and more important. Education is complementary to training. It brings life to formal training by explaining the background to the resources for them. Education involves creating the right atmosphere and motivating user staff. Education information can make training more interesting and more understandable.

- **Training On the Application Software**

After providing the necessary basic training on the computer awareness, the users will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design, type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the data entered. This training may be different across different user groups and across different levels of hierarchy.

8.3 SOFTWARE TESTING

- **Acceptance Testing:**

Acceptance testing is a formal type of software testing that is performed by end user when the features have been delivered by developers. The aim of this testing is to check if the software confirms to their business needs and to the requirements provided earlier. Acceptance tests are normally documented at the beginning of the sprint (in agile) and is a means for testers and developers to work towards a common understanding and shared business domain knowledge.

- **API Testing:**

API testing is a type of testing that is similar to unit testing. Each of the Software APIs are tested as per API specification. API testing is mostly done by testing team unless APIs to be tested or complex and needs extensive coding. API testing requires understanding both API functionality and possessing good coding skills.

- **Beta Testing:**

This is a formal type of software testing that is carried out by end customers before releasing or handing over software to end users. Successful completion of Beta testing means customer acceptance of the software.

- **Black Box testing:**

Black box testing is a software testing method where in testers are not required to know coding or internal structure of the software. Black box testing method relies on testing software with various inputs and validating results against expected output.

CHAPTER 9
CONCLUSION

9.1 CONCLUSION

In this project, a novel medical report generation framework is proposed that considers both language fluency and diagnostic accuracy. More importantly, adversarial reinforcement learning (ARL) is introduced into the training procedure of medical report generation. The encoder-decoder is viewed as a generator and the reward modules as discriminators. In training iterations, discriminators are optimized by maximum-likelihood estimation, whereas the generator is trained by reinforcement learning. Finally, the reward modules give highly accurate rewards and the generator generates better reports. In experiments, first the high performance of the proposed full model is proved by performance comparison with several classical or recently proposed models from different aspects on two large chest X-ray datasets. Ablation studies are then conducted to verify the effectiveness of the language fluency discriminator (FD) and the diagnostic accuracy discriminator (AD), followed by trade-off parameter analysis and qualitative analysis. All of the experimental results demonstrate that the proposed fully learnable ARL architecture that combines AD and FD is superior to purely traditional optimization by cross-entropy alone, or to additional RL with manually designed reward functions.

9.2 FUTURE WORK

This involves fine-tuning parameters and exploring alternative model architectures to improve both language fluency and diagnostic accuracy. Extending the framework to other medical imaging modalities beyond chest X-rays, such as X-RAY or CT scans, would broaden its applicability. Additionally, integrating additional clinical data sources, such as patient demographics or medical history, could enrich contextual understanding and improve report accuracy. Real-world deployment and evaluation in clinical workflows are crucial for assessing practical utility and impact on patient care. Enhancing interpretability and explainability of generated reports is vital for gaining trust and acceptance from healthcare professionals. Addressing ethical concerns surrounding patient privacy, data security, and algorithmic bias is imperative for responsible development and deployment. Collaboration with regulatory bodies and adherence to established guidelines and standards for medical AI systems will ensure patient safety and regulatory complaints.

SAMPLE CODING

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import os
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import seaborn as sns
import pickle

import cv2
from skimage import color
from skimage.feature import greycomatrix, greycoprops
import scipy.stats as stats
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix

from keras.utils.np_utils import to_categorical
from keras.layers import  MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D
from keras.models import Sequential
from keras.models import model_from_json

global filename
global X,Y
accuracy = []
precision = []
recall = []
fscore = []
global X_train, X_test, y_train, y_test
global cnn
global labels

labels = ['Normal Report','Pneumonia Observed']
Findings= ['lung fields appears clear and symmetrical','Inflammation of the air sacs in the
lungs']
Impression=['No evidence of significant abnormalities or pathological changes is
observed','Lobar Infiltrate']
Observations=['Absence of nodules, masses, or cavities','Loss of normal lung markings due to
the filling of airspaces']
MeSH=['Normal interpretation of radiographic images','Pneumonia interpretation of
radiographic images observed']

with open('model/model.json', "r") as json_file:
    loaded_model_json = json_file.read()
    cnn_classifier = model_from_json(loaded_model_json)
json_file.close()
cnn_classifier.load_weights("model/model_weights.h5")
cnn_classifier._make_predict_function()
```

```
main = tkinter.Tk()
main.title("Automatic Report Generation for Chest X-Ray Images") #designing main screen
main.geometry("1300x1200")

def remove_green_pixels(image):
  # Transform from (256,256,3) to (3,256,256)
  channels_first = channels_first_transform(image)

  r_channel = channels_first[0]
  g_channel = channels_first[1]
  b_channel = channels_first[2]

  # Set those pixels where green value is larger than both blue and red to 0
  mask = False == np.multiply(g_channel > r_channel, g_channel > b_channel)
  channels_first = np.multiply(channels_first, mask)

  # Transfrom from (3,256,256) back to (256,256,3)
  image = channels_first.transpose(1, 2, 0)
  return image
def rgb2lab(image):
  return color.rgb2lab(image)

def rgb2gray(image):
  return np.array(color.rgb2gray(image) * 255, dtype=np.uint8)

def glcm(image, offsets=[1], angles=[0], squeeze=False): #extract glcm features
  single_channel_image = image if len(image.shape) == 2 else rgb2gray(image)
  gclm = greycomatrix(single_channel_image, offsets, angles)
  return np.squeeze(gclm) if squeeze else gclm

def histogram_features_bucket_count(image): #texture features will be extracted using
histogram
```

```
  image = channels_first_transform(image).reshape(3,-1)

  r_channel = image[0]
  g_channel = image[1]
  b_channel = image[2]

  r_hist = np.histogram(r_channel, bins = 26, range=(0,255))[0]
  g_hist = np.histogram(g_channel, bins = 26, range=(0,255))[0]
  b_hist = np.histogram(b_channel, bins = 26, range=(0,255))[0]

  return np.concatenate((r_hist, g_hist, b_hist))

def histogram_features(image):
  color_histogram = np.histogram(image.flatten(), bins = 255, range=(0,255))[0]
  return np.array([
    np.mean(color_histogram),
    np.std(color_histogram),
    stats.entropy(color_histogram),
    stats.kurtosis(color_histogram),
    stats.skew(color_histogram),
    np.sqrt(np.mean(np.square(color_histogram)))
  ])

def texture_features(full_image, offsets=[1], angles=[0], remove_green = True):
  image = remove_green_pixels(full_image) if remove_green else full_image
  gray_image = rgb2gray(image)
  glcmatrix = glcm(gray_image, offsets=offsets, angles=angles)
  return glcm_features(glcmatrix)

def glcm_features(glcm):
  return np.array([
    greycoprops(glcm, 'correlation'),
    greycoprops(glcm, 'contrast'),
    greycoprops(glcm, 'energy'),
    greycoprops(glcm, 'homogeneity'),
    greycoprops(glcm, 'dissimilarity'),
  ]).flatten()

def channels_first_transform(image):
  return image.transpose((2,0,1))

def extract_features(image):
  offsets=[1,3,10,20]
  angles=[0, np.pi/4, np.pi/2]
  channels_first = channels_first_transform(image)
  return np.concatenate((
      texture_features(image, offsets=offsets, angles=angles),
      texture_features(image, offsets=offsets, angles=angles, remove_green=False),
      histogram_features_bucket_count(image),
      histogram_features(channels_first[0]),
      histogram_features(channels_first[1]),
      histogram_features(channels_first[2]),
      ))

def getID(name):
    index = 0
    for i in range(len(labels)):
        if labels[i] == name:
            index = i
            break
    return index

def uploadDataset():
    global filename
    filename = filedialog.askdirectory(initialdir = ".")
    text.delete('1.0', END)
    text.insert(END,filename+' Loaded\n\n')
    text.insert(END,"Different type of X-ray Found in Dataset : "+str(labels)+"\n\n")
    text.insert(END,"Total types of X-ray are : "+str(len(labels)))

def featuresExtraction():
    global filename
    global X,Y
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    if os.path.exists("model/X.npy"):
        X = np.load('model/X.npy')
        Y = np.load('model/Y.npy')
    else:
        X = []
        Y = []
        for root, dirs, directory in os.walk(filename):
            for j in range(len(directory)):
                name = os.path.basename(root)
                if 'Thumbs.db' not in directory[j]:
                    img = cv2.imread(root+"/"+directory[j])
                    img = cv2.resize(img, (64,64))
                    class_label = getID(name)
                    features = extract_features(img)
                    Y.append(class_label)
                    X.append(features)
                    print(name+" "+root+"/"+directory[j]+" "+str(features.shape)+"
"+str(class_label))
        X = np.asarray(X)
        Y = np.asarray(Y)
        np.save("model/X",X)
        np.save("model/Y",Y)
    X = X.astype('float32')
    X = X/255 #features normalization
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
    text.insert(END,"Extracted GLCM & Texture Features : "+str(X[0])+"\n\n")
    text.insert(END,"Total images found in dataset : "+str(X.shape[0])+"\n\n")
    text.insert(END,"Dataset train & test split. 80% dataset images used for training and 20%
for testing\n\n")
    text.insert(END,"80% training images : "+str(X_train.shape[0])+"\n\n")
    text.insert(END,"20% training images : "+str(X_test.shape[0])+"\n\n")

def calculateMetrics(algorithm, predict, y_test):
    a = accuracy_score(y_test,predict)*100
    p = precision_score(y_test, predict,average='macro') * 100
    r = recall_score(y_test, predict,average='macro') * 100
    f = f1_score(y_test, predict,average='macro') * 100
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
    text.insert(END,algorithm+" Accuracy  :  "+str(a)+"\n")
    text.insert(END,algorithm+" Precision : "+str(p)+"\n")
    text.insert(END,algorithm+" Recall    : "+str(r)+"\n")
    text.insert(END,algorithm+" FScore    : "+str(f)+"\n\n")
    conf_matrix = confusion_matrix(y_test, predict)
    plt.figure(figsize =(6, 3))
    ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True,
cmap="viridis" ,fmt ="g");
```

```
    ax.set_ylim([0,len(labels)])
    plt.title(algorithm+" Confusion matrix")
    plt.xticks(rotation=90)
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.tight_layout()
    plt.show()

def runSVM():
    global X_train, X_test, y_train, y_test, X, Y
    global accuracy, precision,recall, fscore
    accuracy.clear()
    precision.clear()
    recall.clear()
    fscore.clear()
    text.delete('1.0', END)

    if os.path.exists('model/svm.txt'):
        with open('model/svm.txt', 'rb') as file:
            svm_cls = pickle.load(file)
        file.close()
    else:
        svm_cls = svm.SVC()
        svm_cls.fit(X, Y)
        with open('model/svm.txt', 'wb') as file:
            pickle.dump(svm_cls, file)
        file.close()
    predict = svm_cls.predict(X_test)
    calculateMetrics("SVM", predict, y_test)

def runCNN():
    global X_train, X_test, y_train, y_test, X, Y, cnn
    global accuracy, precision,recall, fscore
    Y1 = to_categorical(Y)
    XX = np.reshape(X, (X.shape[0], X.shape[1], 1, 1))
    X_train, X_test, y_train, y_test = train_test_split(XX, Y1, test_size=0.2)
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            cnn = model_from_json(loaded_model_json)
        json_file.close()
        cnn.load_weights("model/model_weights.h5")
        cnn._make_predict_function()
    else:
        cnn = Sequential()
        cnn.add(Convolution2D(32, 1, 1, input_shape = (XX.shape[1], XX.shape[2],
XX.shape[3]), activation = 'relu'))
        cnn.add(MaxPooling2D(pool_size = (1, 1)))
        cnn.add(Convolution2D(32, 1, 1, activation = 'relu'))
        cnn.add(MaxPooling2D(pool_size = (1, 1)))
        cnn.add(Flatten())
        cnn.add(Dense(output_dim = 256, activation = 'relu'))
        cnn.add(Dense(output_dim = Y1.shape[1], activation = 'softmax'))
        cnn.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
        hist = cnn.fit(XX, Y1, batch_size=12, epochs=10, shuffle=True, verbose=2)
        cnn.save_weights('model/model_weights.h5')
        model_json = cnn.to_json()
        with open("model/model.json", "w") as json_file:
            json_file.write(model_json)
        json_file.close()
        f = open('model/history.pckl', 'wb')
        pickle.dump(hist.history, f)
        f.close()
    print(cnn.summary())
```

```
    predict = cnn.predict(X_test)
    predict = np.argmax(predict, axis=1)
    y_test = np.argmax(y_test, axis=1)
    calculateMetrics("Propose CNN", predict, y_test)



def runInceptionV3():
    global X_train, X_test, y_train, y_test, X, Y, cnn
    global accuracy, precision,recall, fscore
    Y1 = to_categorical(Y)
    XX = np.reshape(X, (X.shape[0], X.shape[1], 1, 1))
    X_train, X_test, y_train, y_test = train_test_split(XX, Y1, test_size=0.2)
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            inceptionv3 = model_from_json(loaded_model_json)
        json_file.close()
        inceptionv3.load_weights("model/inceptionv3_model_weights.h5")
        inceptionv3._make_predict_function()
        model_json = cnn.to_json()
        with open("model/model.json", "w") as json_file:
            json_file.write(model_json)
        json_file.close()
        f = open('model/inceptionv3_history.pckl', 'wb')
        #pickle.dump(hist.history, f)
        #f.close()
    predict = inceptionv3.predict(X_test)
    predict = np.argmax(predict, axis=1)
    y_test = np.argmax(y_test, axis=1)
    calculateMetrics("Propose InceptionV3", predict, y_test)

def graph():
    data = [['SVM', 'Accuracy', accuracy[0]],
        ['SVM', 'Precision', precision[0]],
        ['SVM', 'Recall', recall[0]],
        ['SVM', 'FScore', fscore[0]],
        ['Propose CNN', 'Accuracy', accuracy[1]],
        ['Propose CNN', 'Precision', precision[1]],
        ['Propose CNN', 'Recall', recall[1]],
        ['Propose CNN', 'FScore', fscore[1]],
        ['Propose Inception-V3', 'Accuracy', accuracy[2]],
        ['Propose Inception-V3', 'Precision', precision[2]],
        ['Propose Inception-V3', 'Recall', recall[2]],
        ['Propose Inception-V3', 'FScore', fscore[2]]]

    df = pd.DataFrame(data, columns=['Algorithms', 'Parameters', 'Value'])
    df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')
    plt.show()

def predict():
    global cnn
    filename = filedialog.askopenfilename(initialdir="testImages")
    img = cv2.imread(filename)
    test = []
    img = cv2.resize(img, (64,64))
    features = extract_features(img)
    test.append(features)
    test = np.asarray(test)
    test = test.astype('float32')
    test = test/255
    test = np.reshape(test, (test.shape[0], test.shape[1], 1, 1))
    predict = cnn.predict(test)
    predict = np.argmax(predict)
    img = cv2.imread(filename)
    img = cv2.resize(img, (1000,500))
    cv2.putText(img, 'Chest X-ray Predicted as : '+labels[predict], (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (255, 0, 0), 2)
    cv2.putText(img, 'Findings : '+Findings[predict], (10, 50),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 0), 1)
    cv2.putText(img, 'Impression : '+Impression[predict], (10, 70),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 0, 255), 1)
    cv2.putText(img, 'Observations : '+Observations[predict], (10, 90),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (255, 0, 0), 1)
    cv2.putText(img, 'Mesh Tags : '+MeSH[predict], (10, 110),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 0, 255), 1)
    cv2.imshow('Chest X-ray Predicted as : '+labels[predict], img)
    cv2.waitKey(0)

font = ('times', 16, 'bold')
title = Label(main, text='Automatic Report Generation for Chest X-Ray Images ')
title.config(bg='greenyellow', fg='dodger blue')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 12, 'bold')
text=Text(main,height=15,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)


font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload Chest X-Ray Dataset",
command=uploadDataset)
uploadButton.place(x=50,y=500)
uploadButton.config(font=font1)

featuresButton = Button(main, text="Extract X-Ray Features", command=featuresExtraction)
featuresButton.place(x=300,y=500)
featuresButton.config(font=font1)

svmButton = Button(main, text="Run ML Algorithms", command=runSVM)
svmButton.place(x=680,y=500)
svmButton.config(font=font1)

cnnButton = Button(main, text="Run CNN Algorithm", command=runCNN)
cnnButton.place(x=50,y=550)
cnnButton.config(font=font1)

cnnButton = Button(main, text="Run Propose Inception V3 Algorithm",
command=runInceptionV3)
cnnButton.place(x=300,y=550)
cnnButton.config(font=font1)

graphButton = Button(main, text="Comparison Graph", command=graph)
graphButton.place(x=680,y=550)
graphButton.config(font=font1)


predictButton = Button(main, text="Report Generation from Test Image", command=predict)
predictButton.place(x=900,y=550)
predictButton.config(font=font1)

main.config(bg='LightSkyBlue')
main.mainloop()
```

REFERENCES

1. S. Hochreiter and J. Schmidhuber, ‘‘Long short-term memory,’’ *Neural Comput.*, vol. 9 no. 8, pp. 1735–1780, 1997.
2. B. Jing, P. Xie, and E. Xing, ‘‘On the automatic generation of medical imaging reports,’’ 2017, *arXiv:1711.08195*. [Online]. Available: http://arxiv. org/abs/1711.08195
3. G. Liu, T.-M. Harry Hsu, M. McDermott, W. Boag, W.-H. Weng, P. Szolovits, and M. Ghassemi, ‘‘Clinically accurate chest X-ray report generation,’’ 2019, *arXiv:1904.02633*. [Online]. Available: http://arxiv. org/abs/1904.02633
4. Y. Xue and X. Huang, ‘‘Improved disease classification in chest X-rays with transferred features from report generation,’’ in *Proc. Int. Conf. Inf. Process. Med. Imag.* Cham, Switzerland: Springer, 2019, pp. 125–138.
5. J. Yuan, H. Liao, R. Luo, and J. Luo, ‘‘Automatic radiology report generation based on multi-view image fusion and medical concept enrichment,’’ in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2019, pp. 721–729.
6. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ‘‘Attention is all you need,’’ in Proc. Adv.Neural Inf. Process. Syst., 2017, pp. 5998–6008.
7. K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, ‘‘Show, attend and tell: Neural image caption generation with visual attention,’’ in Proc. Int. Conf. Mach. Learn., 2015, pp. 2048–2057.
8. J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, ‘‘Adversarial learning for neural dialogue generation,’’ 2017, arXiv:1701.06547. [Online]. Available: http://arxiv.org/abs/1701.06547
9. L. Yu, W. Zhang, J. Wang, and Y. Yu, ‘‘SeqGAN: Sequence generative adversarial nets with policy gradient,’’ in Proc. 31st AAAI Conf. Artif. Intell., 2017, pp. 2852–2858.
10. O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, ‘‘Show and tell: A neural image caption generator,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 3156–3164.

11. Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, ‘‘Image captioning with semantic attention,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 4651–4659.
12. J. Krause, J. Johnson, R. Krishna, and L. Fei-Fei, ‘‘A hierarchical approach for generating descriptive image paragraphs,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 317–325.
13. M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, ‘‘Sequence level training with recurrent neural networks,’’ 2015, arXiv:1511.06732. [Online]. Available: http://arxiv.org/abs/1511.06732
14. S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, ‘‘Self-critical sequence training for image captioning,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 7008–7024.
15. Y. Xue, T. Xu, L. R. Long, Z. Xue, S. Antani, G. R. Thoma, and X. Huang, ‘‘Multimodal recurrent model with attention for automated radiology report generation,’’ in Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervent. Cham, Switzerland: Springer, 2018, pp. 457–466.
16. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, ‘‘Generative adversarial nets,’’ in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 2672–2680.
17. A. Radford, L. Metz, and S. Chintala, ‘‘Unsupervised representation learning with deep convolutional generative adversarial networks,’’ 2015, arXiv:1511.06434. [Online]. Available: http://arxiv.org/abs/1511.06434
18. H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, ‘‘StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks,’’ in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 5907–5915.
19. X. Wang, Y. Peng, L. Lu, Z. Lu, and R. M. Summers, ‘‘TieNet: Textimage embedding network for common thorax disease classification and reporting in chest X-rays,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 9049–9058.
20. W. Wei, L. Cheng, X. Mao, G. Zhou, and F. Zhu, ‘‘Stack-VS: Stacked visual-semantic attention for image caption generation,’’ 2019,arXiv:1909.02489. [Online]. Available: http://arxiv.org/abs/1909.02489

21. T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei, ‘‘Boosting image captioning with attributes,’’ in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 4894–4902.
22. Z. Gan, C. Gan, X. He, Y. Pu, K. Tran, J. Gao, L. Carin, and L. Deng, ‘‘Semantic compositional networks for visual captioning,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 5630–5639.
23. K. He, X. Zhang, S. Ren, and J. Sun, ‘‘Deep residual learning for image recognition,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.
24. J. Irvin, P. Rajpurkar, M. Ko, and Y. Yu, ‘‘CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison,’’ in Proc. 33rd AAAI Conf. Artif. Intell., 2019, pp. 590–597.
25. K. Simonyan and A. Zisserman, ‘‘Very deep convolutional networks for large-scale image recognition,’’ 2014, arXiv:1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556
26. X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, ‘‘ChestXray8: Hospital-scale chest X-ray database and benchmarks on weaklysupervised classification and localization of common thorax diseases,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017,pp. 2097–2106.
27. Q. Guan, Y. Wang, B. Ping, D. Li, J. Du, Y. Qin, H. Lu, X. Wan, andJ. Xiang, ‘‘Deep convolutional neural network vgg-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images:A pilot study,’’ J. Cancer, vol. 10, no. 20, p. 4876, 2019.