



***Dissertation on***

**“Image Captioning for Chest X-Ray Images”**

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

**Bachelor of Technology  
in  
Computer Science & Engineering**

***Submitted by:***

<b>Krishna Khurana</b>	<b>01FB16ECS482</b>
<b>Manmath Sahoo</b>	<b>01FB16ECS488</b>
<b>Shrey Jain</b>	<b>01FB16ECS490</b>

***Under the guidance of***

**Internal Guide**

**Rama Devi P**  
Professor,  
PES University

**External Guide**

**January – May 2020**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
FACULTY OF ENGINEERING  
PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)  
100ft Ring Road, Bengaluru – 560 085, Karnataka, India



## PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)  
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

### FACULTY OF ENGINEERING

## CERTIFICATE

*This is to certify that the dissertation entitled*

### **‘Image Captioning for Chest X-Ray Images’**

*is a bonafide work carried out by*

**Krishna Khurana**

**01FB16ECS482**

**Manmath Sahoo**

**01FB16ECS488**

**Shrey Jain**

**01FB16ECS490**

In partial fulfilment for the completion of eighth semester project work in the Program of Study Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2020 – May. 2020. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8<sup>th</sup> semester academic requirements in respect of project work.

Signature  
P Rama Devi  
Professor

Signature  
Dr. Shylaja S S  
Chairperson

Signature  
Dr. B K Keshavan  
Dean of Faculty

### External Viva

**Name of the Examiners**

**Signature with Date**

1. \_\_\_\_\_

\_\_\_\_\_

2. \_\_\_\_\_

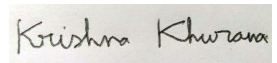
\_\_\_\_\_

## DECLARATION

We hereby declare that the project entitled “**Image Captioning for Chest X-Ray Images**” has been carried out by us under the guidance of Prof. P Rama Devi, and submitted in partial fulfillment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2020. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

**01FB16ECS482**

**Krishna Khurana**



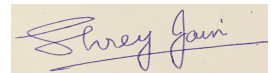
**01FB16ECS488**

**Manmath Sahoo**



**01FB16ECS490**

**Shrey Jain**



## **ACKNOWLEDGEMENT**

We would like to express my gratitude to Prof. Rama Devi P, Dept of Computer Science, PES University, for her continuous guidance, assistance and encouragement throughout the development of this project.

We are grateful to the project coordinators, Prof. Preet Kanwal and Prof. Sangeetha V I for organising, managing and helping out with the entire process.

We also take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

We are extremely grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing us various opportunities and enlightenment every step of the way.

## **ABSTRACT**

For medical images, radiologists have to write a description completely for the medical reports. Understanding these reports is a complex and tedious task which requires a lot of experience and time. Diagnosis done by inexperienced radiologists in rural areas can sometimes be error-prone and fatal. To have a less number of bugs and for saving the radiologist's time we need a system which is computer-aided for the generation of medical reports, which can further be used by radiologists to refer and diagnose better. So we propose an Encoder-Decoder architecture to annotate medical images. The input to the encoder is the medical image which we have to caption. The encoder consists of the Inception V3 model, in which we have removed the last layer of the model to encode the X-ray image. The encoded image from the CNN layer is used as the input to the RNN model, which consists of the Attention mechanism. The decoder will provide us with the caption for the images. We use various evaluation metrics to compare our model with other base models.

# TABLE OF CONTENTS

Chapter No.	Title	Page No.
<b>1.</b>	<b>INTRODUCTION</b>	<b>01</b>
<b>2.</b>	<b>PROBLEM DEFINITION</b>	<b>04</b>
<b>3.</b>	<b>LITERATURE SURVEY</b>	<b>05</b>
	3.1 A Survey on Biomedical Image Captioning	<b>05</b>
	3.1.1 Introduction	05
	3.1.2 Methods	05
	3.1.3 Evaluation	06
	3.2 From chest x-rays to Radiology reports: A multimodal machine learning approach	<b>06</b>
	3.2.1 Introduction	06
	3.2.2 Methods	07
	3.2.3 Evaluation	08
	3.3 On the automatic generation of Medical Imaging reports	<b>08</b>
	3.3.1 Introduction	08
	3.3.2 Methods	09
	3.3.3 Evaluation	09
<b>4.</b>	<b>PROJECT REQUIREMENTS SPECIFICATION</b>	<b>10</b>
	4.1 Sequence Diagram	<b>10</b>
	4.2 Constraints	<b>10</b>
	4.2.1 Design Constraints	10
	4.2.2 System Constraints	10
	4.3 Product Perspective	<b>11</b>
	4.3.1 User Characteristics	11
	4.3.2 General Constraints, Assumptions and Dependencies	11
	4.3.3 Risks	11
<b>5.</b>	<b>SYSTEM REQUIREMENT SPECIFICATIONS</b>	
	5.1 Functional Requirement	<b>12</b>
	5.1.1 Generation of Captions for Biomedical Images	12
	5.1.2 Use Case	12
	5.2 Non-Functional Requirement	<b>13</b>
	5.2.1 Dependencies	13
	5.2.2 Assumption	13
	5.3 Hardware Requirement	<b>13</b>
	5.4 Software Requirement	<b>13</b>

<b>6. SYSTEM DESIGN</b>	
6.1 Design Approach	<b>16</b>
6.1.1 Download the IU X-ray Dataset	16
6.1.2 Preprocess and tokenize the captions	16
6.1.3 Load pre-trained Inception V3 model	16
6.1.4 Model Building	17
6.1.5 Train the model	17
6.1.6 Generate captions	17
6.2 Image Captioning Model Architecture	<b>18</b>
<b>7. DETAILED DESIGN</b>	
7.1 Encoder Decoder	<b>19</b>
7.2 Attention Mechanism	<b>20</b>
7.3 Transfer Learning	<b>21</b>
<b>8. IMPLEMENTATION AND PSEUDOCODE</b>	
8.1 Download and Prepare IU X-Ray Dataset	<b>22</b>
8.2 Preprocess and tokenize the captions	<b>24</b>
8.3 Preprocess the images using Inception V3	<b>26</b>
8.4 Initialize Inception V3 and load the pre trained Imagenet weights	<b>26</b>
8.5 Model	<b>27</b>
8.5.1 Encoder	28
8.5.2 Decoder	29
8.5.3 Attention Mechanism	30
8.6 Training	<b>31</b>
<b>9. TESTING</b>	<b>34</b>
<b>10. RESULTS AND DISCUSSION</b>	<b>35</b>
10.1 Results	35
10.2 Discussions	36
<b>11. SNAPSHOTS</b>	<b>37</b>
<b>12. CONCLUSIONS</b>	<b>41</b>
<b>13. FUTURE WORKS</b>	<b>42</b>
<b>REFERENCES/BIBLIOGRAPHY</b>	<b>43</b>
<b>APPENDIX A DEFINITIONS, ACRONYMS &amp; ABBREVIATIONS</b>	<b>45</b>

## LIST OF TABLES

Table No.	Title	Page No
3.1	Evaluation metric for A survey on Biomedical Image Captioning	06
3.2	Result based on the generation of reports for medical radiology images( use of Glove and RadGlove).	08
3.3	Experimental result on generating radiology reports(Co-Attention)	09
10.1	Comparison of our model with other referred models	35

## LIST OF FIGURES

Figure No.	Title	Page No.
1.1	Introduction to Image Captioning	01
1.2	An image of Chest X-ray from IU X-RAY along with report provided by the radiology	02
3.1	A framework showing encoder-decoder for the generation of medical image's radiology reports along with LSTM recurrence iterations	07
4.1	Sequence Diagram	10
5.1	Use Case Diagram	12
6.1	Workflow of our Image Captioning model	16
6.2	Simple Architecture of the Captioning model	18
7.1	Encoder-Decoder Architecture	19
7.2	Attention Mechanism in our project	20



7.3	Transfer Learning Example	21
8.1	Download the IU-X ray dataset	22
8.2	Extracting Findings and Impressions	23
8.3	Train-Test Split	23
8.4	Loading the dataset	24
8.5	Text Cleaning	25
8.6	Preprocessing captions	25
8.7	Cleaned and pre-processed tokens	26
8.8	Preprocessing the Images	26
8.9	Loading the Inception V3 module	27
8.10	Using the Inception V3 as our Encoder	28
8.11	CNN Encoder	29
8.12	RNN Decoder	29
8.13	Bahdanau Attention	30
8.14	Attention mechanism in code	30
8.15	Training the model	31
8.16	Trained for approx 80 epochs	32
8.17	Loss Plot	32
8.18	Sample caption with attention	33
11.1	Code for ROGUE score calculation	37
11.2	Output for ROUGE average score	37
11.3	Code and Output for METEOR score	38
11.4	Output for BLEU score	38
11.5	Code to calculate BLUE score	39

11.6	Cosine Similarity code for evaluation of model	40
------	--	----

## Chapter-1

### INTRODUCTION

This decade, we have come to see major development in the field of medical science. Mostly the advancement has been in the field of technologies used to diagnose a disease. From manually doing checkups and going through the scan reports, to shifting to machine read reports and diagnosis, there has been a major shift, and it has all been possible due to the intelligent system built using techniques of machine learning and computer vision.

Humans can easily describe a thing after having a look at an object, same is the case with image captioning, but instead of humans it is the computer which does the captioning of images. It is an easy task to generate a caption for normal objects or scenes. The figure below shows “A road leading into the mountains”, or we can say “A beautiful view of mountain range from the road”. Also a caption can be something like “a road passing by the farm into the mountains”.

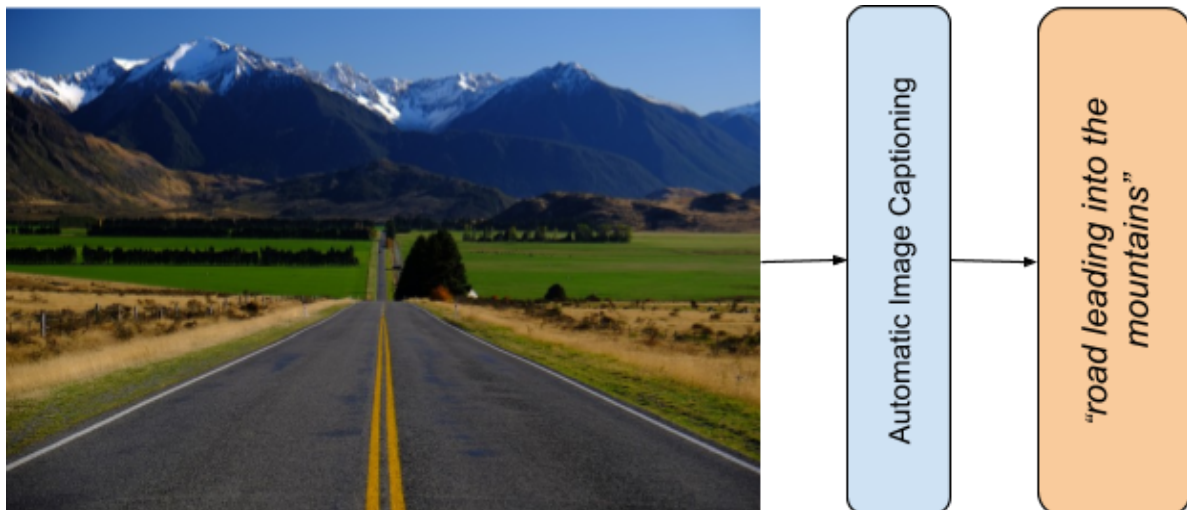


Fig 1.1: Introduction to Image Captioning

But the same is not true in the healthcare and medicine domain. Image captioning becomes extremely challenging when we process a medical image. We cannot have different captions for the same medical images. It is quite possible to miss out something when we look at objects, but such negligence can't be done in healthcare as a minor change in details can

result in wrong diagnosis of a patient. So it is very important to have the data and images which are genuine and preverified.

In the field of healthcare, one of the data sources which is fastest and largest growing is medical imaging . It comprises the maximum number of medical data in the present day healthcare industry. Even after an increase in the number of medical images, a research study by Signify shows that the count of radiologists per million people is gradually becoming less world wide. Also, in a country like India which has over a billion people , there are approximately 1-2 radiologists for every one lakh people . If we compare this with the United States , where we have a ratio of one radiologist for every ten thousand people , India is significantly behind. That's the primary reason behind why the clinical practices have now become much more busy, since we need to have the examinations for the diagnosis for such a large number of cases, which is a matter of grave concern.

Even though there has been a shift in the way doctors do their work, the major hurdles for new research in the healthcare domain is due to limited expertise, privacy and ethical issues. The medical data are not easily available for general works, also if they are there is a need to have a safe environment to work, as there is a risk of leaking of private datas. But the positive side is that a lot of medical data is stored in hospitals' Private Archival and Communication System (PACS). Also some universities have started collecting data, which can be found publicly.



**Comparison:**  
Chest dated xxxx  
**Indication:**  
xx-year old female with chest pain  
**Findings:**  
The lungs are clear without evidence of focal airspace disease.  
There is no evidence of pneumothorax or large pleural effusion.  
The cardiac and mediastinal contours are within normal limits.  
There are degenerative changes throughout the thoracic spine.  
**Impression:**  
No radiographic evidence of acute cardiopulmonary disease.

Fig-1.2: An image of Chest X-ray from IU X-RAY along with report provided by the radiology

The report provided by the radiologist is used for documentation and communication of important observations in the study of radiology. An usual report provided by the radiologist normally contains a background check which has the description of all the examinations and information of the patient, another section is of findings, and another section is of the impression. Usually the steps that these radiologists follow is that they first do the dictation of the findings in detail for the medical report, and then they have to find the overall findings which were salient and put it into the section of Impression which is more concise which is also based on the patient's condition. It is evident from Fig-2 that this task is one of the tasks that are very challenging, and there is a need for a system which is able to understand the chest X-ray's content comprehensively and behave as a bridge for the x-ray image and the x-ray image's description. Since human readings can sometimes be error prone while reading the images, we intend to build an image captioning model which will automatically generate the summary of findings.

## CHAPTER-2

### PROBLEM DEFINITION

In clinics and hospitals, the commonly used images are medical images of pathology and radiology, which are used for the diagnosis and also for treating a large number of diseases. Interpretations and readings are generally done by specialized medical professionals. We plan on building a model which can be used to diagnose general problems, and thus allowing doctors to focus more on the patients with much dangerous problems.

We aim to develop an image captioning model, which will take in Chest X-ray images as the input and automatically generate a caption which will contain the summary of findings in the image.

The model can help tackle issues such as-

1. In rural areas the quality of healthcare is relatively low and there are some less experienced radiologists and pathologists. For them writing correct reports can be very demanding. Using the help of our model, these Radiologists can do their job with confidence.
2. In nations with large populations like India, a radiologist might sometimes have to go through a large number of x-ray images on a daily basis. Typing the findings for all such reports can be tedious and also be error prone. Automatically generating reports can help reduce human errors and also make the work easier, helping to diagnose more number of patients.
3. The model can be used in case of emergency as well. If in case there are no attendings or radiologists present during the emergency trauma, the model can help generate a result, which can help the residents to give proper painkillers or medication to buy some more time until the attendings arrive.
4. Teleradiology is used nowadays to send medical images to doctors in metropolitan cities or remote areas. Along with it a computer generated report can be sent which in return can help doctors in remote locations diagnose a patient more quickly.

---

## CHAPTER-3

### LITERATURE SURVEY

#### 3.1 A Survey on Biomedical Image Captioning

##### 3.1.1 Introduction

The survey is the very first overview of ways (methods) used for the image captioning of the biomedical images, datasets and evaluation measures. It illustrates the problem that the radiologists and other physicians are facing in their day to day life, examining many biomedical images on a daily basis for example positron emission tomography (PET) or Computerized tomography (CT) scans. Examination of the x-ray images and then the work of writing their observation for the medical reports is a tedious task for the radiologists. Methods which can assist radiologists to focus on important regions of the image to describe their findings can reduce workload of the radiologists and can also result in less medical errors and benefit the department of the medical science by reducing the cost per exam.

Despite having the significance of biomedical image captioning, the datasets related to this field are not available, obstructing the emergence of new techniques. There are only three datasets available publicly, IU X-RAY, which contains images of x-ray, PEIR GROSS which contains clinical photographs and ICLEF CAPTION which contains a mix of x-ray images and clinical photographs. This is also a factor which is limiting the growth of the image captioning in the field of medical science.

##### 3.1.2 Methods

This paper proposes the two methods that they have used for the generation of captions for the biomedical images. The first one is based on the frequency and the second one is based on the nearest neighbour. The frequency based method uses the count of each and every word that is present in the captions of the training dataset. The word which has the highest number of occurrence in the training dataset captions, is always chosen to be the predicted caption's first word and in the same way, the following word which is chosen for the second word is the next frequent word of the training dataset caption. The average of the length of the training dataset captions gives the count of the words that will be there in the predicted caption. The nearest neighbour method follows the methodology of similar biomedical images should

have similar captions generated. In this they have used the RESNET-18 for encoding the images of the training dataset and then used cosine similarity methods to recognize the images that are the same in a way. The caption that will be generated for the given input image will be generated from the caption of the image which is retrieved as the most similar image with respect to the new image. This also describes why the similar image retrieval system executes efficiently for the biomedical image captioning.

### 3.1.3 Evaluation

The BLUE score was high for the evaluation of the frequency based method and the reason behind this was that these are dependent on the processing of unigrams. The method which is frequency based, follows this process of the words which are most common in the training dataset. The nearest neighbour method outperforms the frequency based method in all evaluation measures for the pair gross dataset but it has low accuracy when the IU-Xray dataset is considered. The reason behind the low accuracy was probably because the IU-Xray has x-ray images and the RESNET-18 which is used for encoding was unsuccessful in handling the images properly. This paper also proposes to use a better encoder to implement and train on X-ray images so that the accuracy can be improved.

Methods	BLEU	METEOR	ROUGE	WMS
Frequency	0.442	0.176	0.187	0.302
Nearest Neighbour	0.281	0.125	0.209	0.26

Table: 3.1 Evaluation metric for A survey on Biomedical Image Captioning

## 3.2 From chest x-rays to Radiology reports: A multimodal machine learning approach

### 3.2.1 Introduction

The paper starts with referring to how the medical images have been the major source of medical data present in the current day healthcare system. Also it talks about the percentage of radiologists present in the different parts of the world, how there is the scarcity of radiologists which needs to be addressed. It has been discussed how hard it is to read medical images and the difficulties faced by the current day radiologist to write error-free reports.



The paper proposes a model which is using the technique of multimodal machine learning which helps in the generation of medical image's radiology reports. It further claims that the previous text-generation model which used the Convolutional Neural Network-Recurrent Neural Network which uses the Long Short Term Memory which is comparatively shallow in depth. Based on a research project for the detection of objects and classification of images, it has been claimed that in terms of learning methods which are deep and hierarchical are much better than shallow methods. Henceforth, a multi stacked model which uses Convolutional Neural Network-Recurrent Neural Network has the capability for the generation of reports for chest X-ray images which are semantically rich.

### 3.2.2 Methods

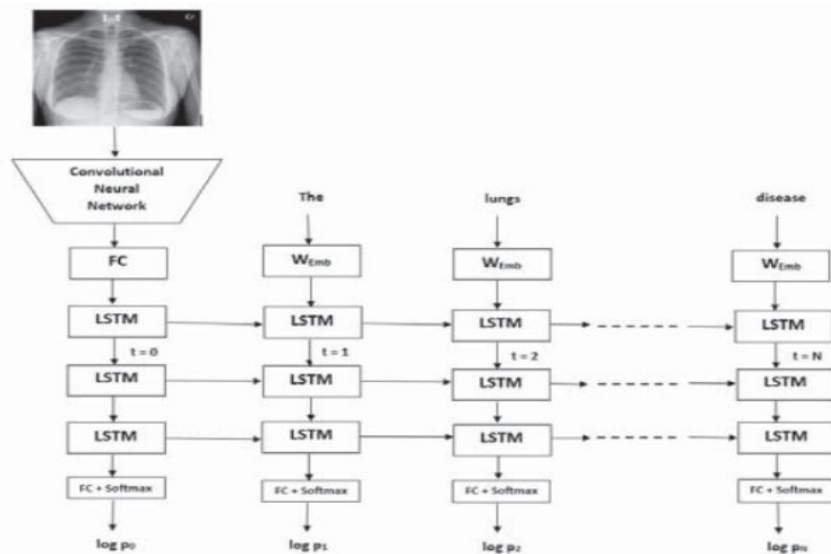


Fig-3.1 A framework showing encoder-decoder for the generation of medical image's radiology reports along with LSTM recurrence iterations.

The encoder-decoder model used is an extended version of “neural image captioner” by Vinyals. This model has an encoder framework which helps in conversion of the image into a vector form of representation and also it has a decoder that helps us in transformation of the vector form of image into a words sequence form. They have used the inception-V3 model which is provided by Google and this has an encoder so that it can have a significantly deeper structure and because of the inception blocks it gets to learn semantically rich representation of the images. On the validation set of the “ImageNet Large Scale Visual Recognition Challenge”, the V3-Inception model scored 3.58% top-5 error as compared to the VGG-16 model which merely achieved 6.8% top 5 error. And for the decoder the paper has used LSTM as the RNN since they are usually trained and this training is done using the process of

back propagation , which can propagate back through the time, that can help in solving the issues of disappearing the gradient problems.

The model also uses the Transfer Learning technique since it has an important use in the domain of Medical Imaging, where it is very difficult in getting the dataset having the image along with labels. To enable knowledge transfer, the CNN was initialized with the weights which were pre-trained and they were already trained and this was done on a large ImageNet dataset. Also, they used Glove word embeddings on the text side. Also the paper used RadGlove which has been trained at Stanford University and this has been done on 4,500,000 reports for the radiology images.

### 3.2.3 Evaluation

The project has been performed on IU-Xray Dataset which has been made publicly available by Indiana University. They have compared their model with a baseline model which used Vanilla CNN-RNN in terms of BLEU, ROUGE and METEOR. The model was able to outperform the base model and shows a qualitative result of how the RadGlove helps in getting a better result.

Methods	BLEU	METEOR	ROUGE
Baseline	28.91	13.79	26.57
Depth-3 with Glove	36.39	15.98	30.21
Depth-3 with RadGlove	37.40	16.35	30.76

Table-3.2 :Result based on the generation of reports for medical radiology images( use of Glove and RadGlove).

## 3.3 On the automatic generation of Medical Imaging reports

### 3.3.1 Introduction

The paper describes the usefulness of medical images and how they have been used to diagnose various diseases. Also the paper tells about the problems faced by the radiologists, and how generating reports manually can be tedious and tiresome. It also mentions the skills required by the radiologist to read a medical image. After covering the use cases and study,

the paper proposed the model they have used and how it can be achieved. They divided the work into 3 different sections to generate the automated report.

### 3.3.2 Methods

First and foremost, the paper describes a multi task framework. The model treats the prediction of tags as a multi label classification and also treats the production of long descriptions as a Text-generation work. From the last layer of convolutional network the visual features of the VGG-19 model are extracted since they are a classification model and the last two layers of VGG-19 are used for MLC.

Then, a co-attention mechanism is used to localize attachment and image-regions to the right description, which simultaneously attends the predicted tags and images. It has been found that visual attention is unable to provide sufficient high level semantic data. So the tags can be used which will provide the much required high level information, which can help visualize the image properly and find the problem much more efficiently.

Lastly the descriptions in medical imaging reports are often very long, therefore containing multiple sentences. As long sentences generally tend to be non trivial, so instead of using a single layer LSTM model which is less capable of modelling they have tried using a hierarchical LSTM to generate long texts, which when combined with a co-attention mechanism, usually generates high level topics.

### 3.3.3 Evaluation

The paper evaluates the model with different types of evaluation metrics such as METEOR, BLEU, ROUGE and CIDER. They have compared their work with four other works which include models such as Vanilla CNN-RNN, Soft-Attention and ATT-RK. Their work did way better than their baseline model, mentioned in Table-2 below.

Methods	BLEU	METEOR	ROUGE	CIDER
Vanilla CNN-RNN	0.316	0.159	0.267	0.111
Co-Attention	0.517	0.217	0.447	0.327

Table 3.3 Experimental result on generating radiology reports(Co-Attention)

## CHAPTER-4

### PROJECT REQUIREMENTS SPECIFICATION

#### 4.1 Sequence Diagram

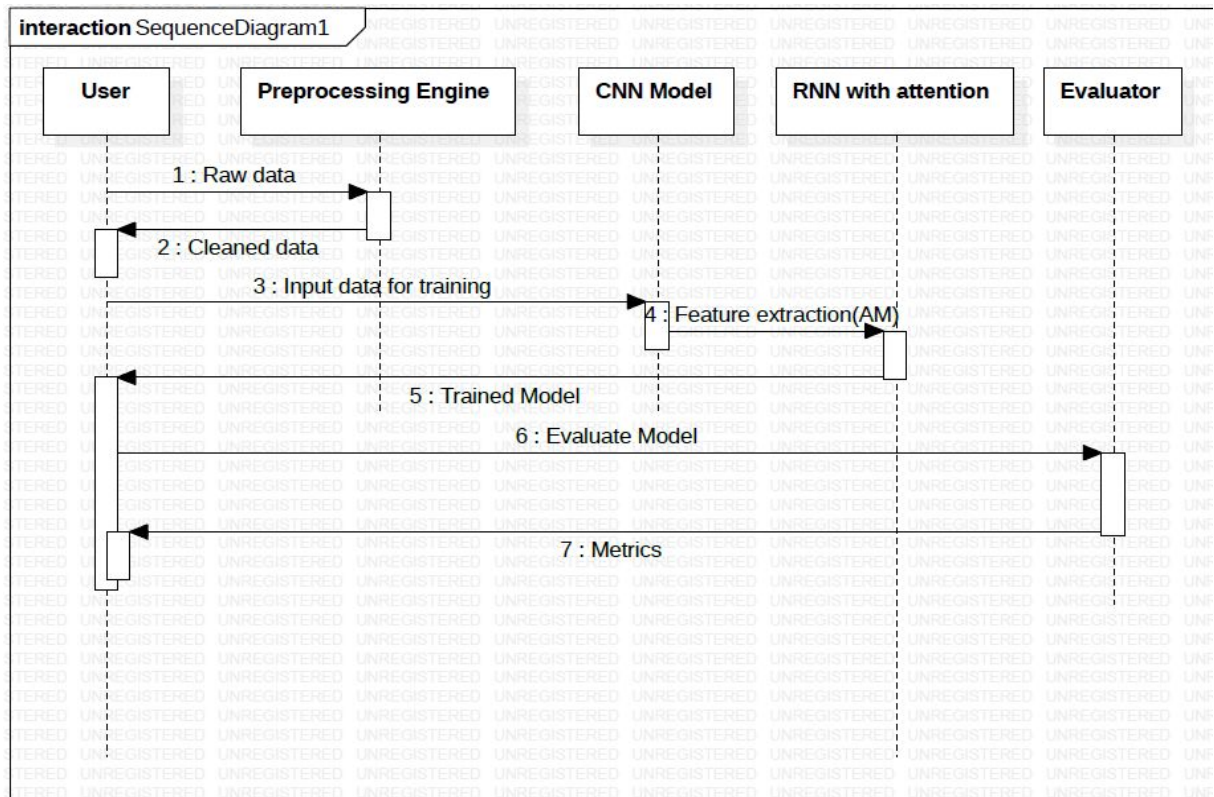


Fig 4.1: Sequence Diagram

#### 4.2 Constraints

##### 4.2.1 Design Constraints

1. The model is being designed for the IU-Xray dataset only.
2. The model is being targeted for the biomedical x-ray images.

##### 4.2.2 System Constraints

1. The image caption generation is supported for Windows 10 and up and is supported for 64-bit ABI.
2. Linux, MacOS X, Windows are the operating systems that the image caption generation model can work on.
3. Generation of image captions for the biomedical images requires software packages - Python  $\geq 3.7.4$  , NumPy  $\geq 1.16.5$  , xml.etree.ElementTree  $\geq 1.3.0$  , os  $\geq 1.16.0$  ,

sklearn >=0.21.3 , PIL >6.2.0 , Pandas >=0.25.1 , Matplotlib >=3.1.1 , TensorFlow  
>=2.1.0

### 4.3 Product Perspective

The platform used for the development of this project are:

1. Windows 10, 64 bit
2. Jupyter Notebook anaconda3
3. TensorFlow

#### 4.3.1 User Characteristics

The radiologists will be benefited from this image captioning for x-ray images. Their tedious task of writing the findings for the x-ray image will be easy as they can save their precious time not typing their findings and also they will be able to focus on important regions of the image to describe their findings can reduce workload and can also result in less medical errors and benefit the department of the medical science by reducing the cost per exam.

#### 4.3.2 General Constraints, Assumptions and Dependencies

1. Restrictions to unauthorized users, since it can be used in an unethical manner to make money.
2. The images along with the reports received in the dataset are genuine and true.
3. Various modules which run on previous models of keras, can cause version problems.
4. The application should be supported by various web browsers.
5. High computation power is needed to train the model.

#### 4.3.3 Risks

1. The dataset cannot be made public, since it is of medical use. This limits the growth of new technology.
2. Only certified radiologists will be given the access to use the product to generate the caption.

## CHAPTER-5

### SYSTEM REQUIREMENT SPECIFICATIONS

#### 5.1 Functional Requirement

##### 5.1.1 Generation of Captions for Biomedical images

As at the very first stage the dataset is loaded and then the captions are preprocessed to have a cleaner description by formatting all the characters to lowercase and then removing all the punctuations and characters like 'a' and 's'. The model is then trained using InceptionV3 as encoder and then using RNN as decoder. The model takes an image as input and then it will generate the caption for the input image, and then it is evaluated for calculating the accuracy.

##### 5.1.2 Use Case

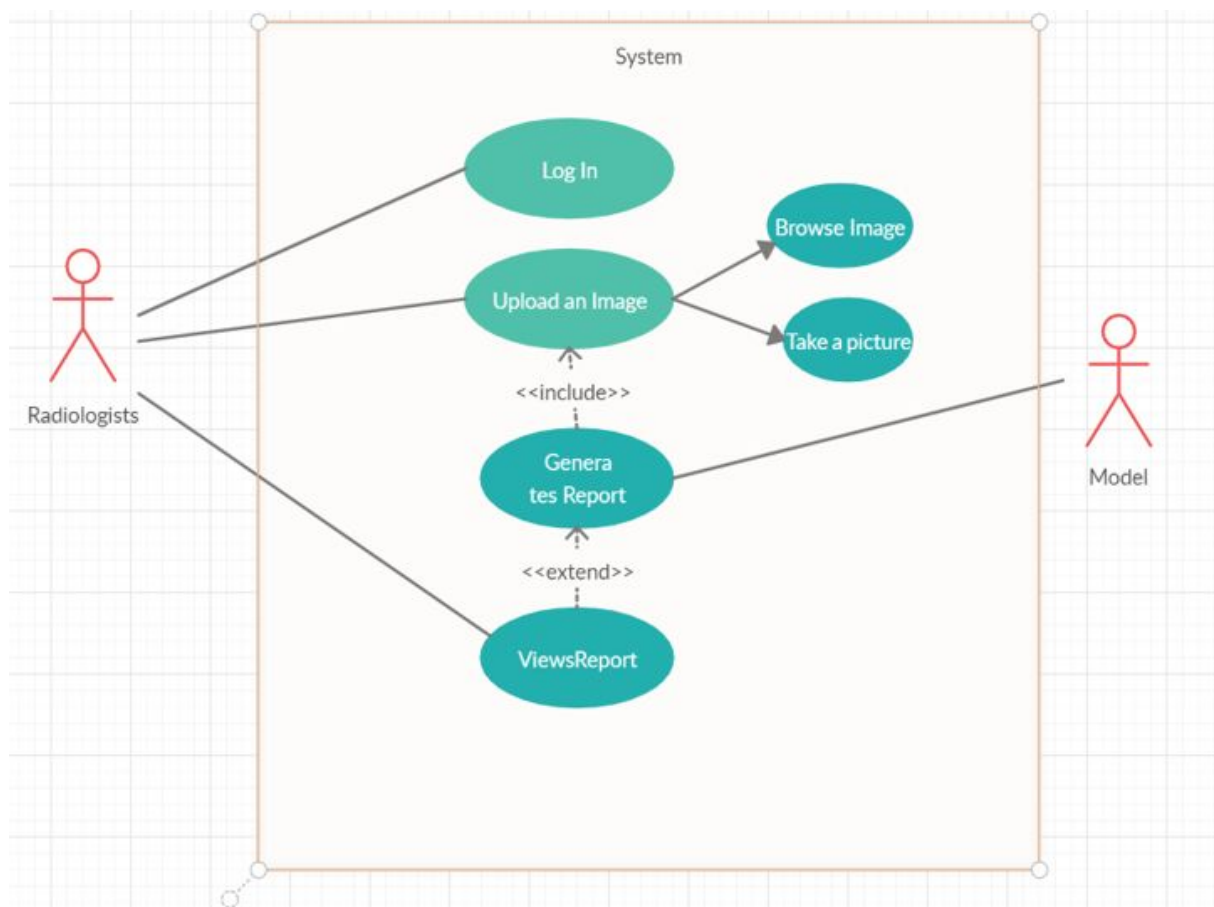


Fig 5.1: Use case Diagram

## 5.2 Non-Functional Requirement

### 5.2.1 Dependencies

1. Various modules which run on previous models of keras, can cause version problems.
2. The application should be supported by various web browsers.
3. High computation power is needed to train the model.

### 5.2.2 Assumption

1. Restrictions to unauthorized users, since it can be used in an unethical manner to make money.
2. The images along with the reports received in the dataset are genuine and true.

## 5.3 Hardware Requirement

The model for the biomedical image captioning can be run on all the operating systems that can run Jupyter notebook, so all the hardware requirements that can host these operating systems can support the biomedical image captioning model. However a sufficient amount of RAM preferably 8 GB to train the model is advised.

## 5.4 Software Requirement

- Python -

Python is designed by the very famous programmer Guido van Rossum in the year 1991. It is a highly interpreted, high-level, general purpose coding/programming language.

Version - 3.7.4

- NumPy - One of the most basic and fundamental modules which helps in performing scientific computations is the NumPy in python. It is a python library that gives us several derived objects , multidimensional array object and a variety of routines for a fast and quick operations on arrays, which

include shape ,manipulation, logical, mathematical, selecting, I/O , DFT's ,  
sorting, statistical operations, some basic linear algebra and much more.

Version:'1.16.5'

- xml.etree.ElementTree - (Lightweight XML support for Python) . This module implements an elegant and efficient API which is used for parsing and creating XML data.

Version = '1.3.0'

- os - It is an OS routine for Posix or NOT depending on what system we are using.

This module provides us very portable methods of using OS dependent functionality.

Version = '1.16.0'

- Sklearn - ( Scikit-learn ) is a module in Python which combines the machine learning (ML) algorithms in the inseparable world of scientific Python packages ( matplotlib,numpy,scipy). The package aims to provide elegant and efficient solutions to the learning problems that are approachable to everybody and reusable in several contexts.

Version = ' 0.21.3 '

- Pickle - Serialization & deserialization of an object in Python is generally done using the Pickle module. Any object in Python language can be pickled so that we can save it on our disk. The pickle module “serializes” the object first beforehand writing it to the file.

- PIL - Pillow (Fork of the Python Imaging Library) - It is a free library for the Python language which adds support for manipulating, opening, and saving several image file formats.

Version = '6.2.0'



- Pandas - Python provides pandas as a software library for performing data manipulation & analysis. It provides us with data structures & operations that help in manipulating time series and numerical tables. It is a free software which was released under the 3-clause BSD license. It provides a flexible, fast, and expressive data structures devised to make working with "labeled" or "relational" data that is both easy and intuitive.

Version = '0.25.1'

- Matplotlib - It is a plotting library for Python. It provides us with an object-oriented API that can be used for embedding plots into several applications.

Version = '3.1.1'

- Tensorflow - It is an open source library for high performance numerical computation. Its flexible architecture helps in easy deployment of computation.

Version = '2.1.0'

## CHAPTER-6

### SYSTEM DESIGN

#### 6.1 Design Approach



Fig 6.1 Workflow of our Image Captioning model

##### 6.1.1 Download the IU X-Ray Dataset

There was no publicly readymade dataset available , so we had to perform web scraping on the Open Access Biomedical Image Search Engine. We downloaded the images and also the reports associated with them to get the captions. We have used the Findings and Impression part of the X-ray report for building the caption.

##### 6.1.2 Preprocessing of captions

The captions were prepared from the X-Ray reports, which had to be cleaned and converted to a format which could be fed to a deep learning model.

1. Firstly, we tokenized the captions This presents us with a vocabulary of all unique words in the data.
2. Next, we had to reduce the vocabulary size to a maximum of 3,000 words .
3. We generated word-to-index & index-to-word mappings.
4. Lastly, we padded all sequences to be of exact same length as the longest.

##### 6.1.3 Load the pre-trained InceptionV3

We then made a keras model where our final output layer is the last convolutional layer in the InceptionV3 architecture. The shape of output of this layer was found to be 8x8x2048. So we use the last convolutional layer since we are using the attention mechanism here. This

initialization is not performed during training because there is a possibility of becoming a bottleneck.

#### **6.1.4 Model Building**

Extract the features from lower convolutional layer of InceptionV3 giving a vector.

We squash it to the shape of (64, 2048).

1. This vector is passed through the Convolutional Neural Network Encoder.
2. The Gated Recurrent Network decoder attends over image to predict next word.

#### **6.1.5 Train our model**

1. Extract features stored in .npy files and afterwards pass them through our encoder.
2. The encoder output, hidden state and the decoder input is passed to our CNN decoder.
3. The CNN decoder gives back model predictions and the hidden state of the decoder.
4. This decoder hidden state is made to pass back into our model, after which the predictions are used to calculate the loss.
5. Teacher forcing has been employed to decide the next input for our decoder.
6. The last step will calculate the gradients and apply those gradients to optimizer and then backpropagate.

#### **6.1.6 Produce captions**

Evaluate function is very much the same as the training loop, the only difference is that we have not used teacher forcing. The input to decoder at every time step is the previous predictions, hidden state and the encoder output. Lastly, we stop to predict when the model produces the end token, and stores attention weights for each step.

## 6.2 Image Captioning Model Architecture

### Image Captioning Model Architecture

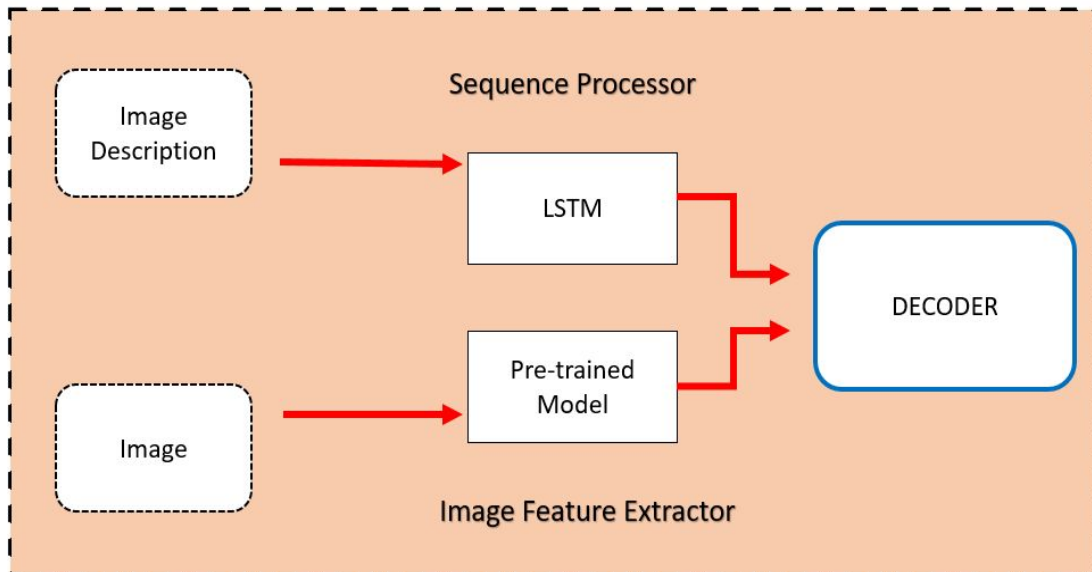


Fig 6.2: Simple Architecture of the Captioning model

#### Encoder-Decoder used for Image Captioning.

A pre-trained CNN model has been employed to bring out the features of the image in a dense representation. The image description has to be transformed to a n-dimensional vector using our word-to-id map. Now, both our Sequences and Image Features are merged and supplied into our Decoder.

## CHAPTER-7

### DETAILED DESIGN

#### 7.1 Encoder Decoder

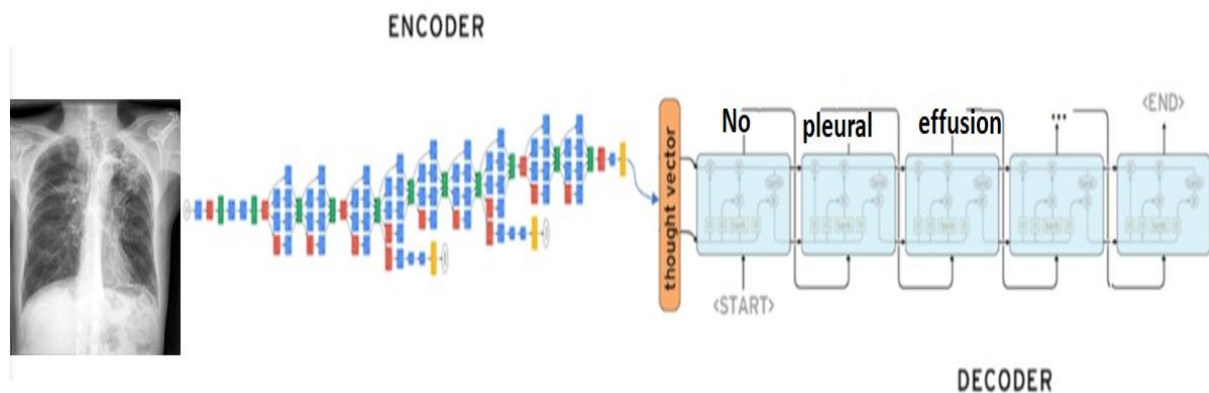


Fig 7.1: Encoder-Decoder Architecture

The job of image captioning is thought of as being broken down into two modules – one being **image based model** (which draws out the features and other nuances out of the image), and the second one is **language based model** – (which is used to transform the features, objects given by first model into a natural language text).

Now for the image based model (Encoder) – we have deployed a Convolutional Neural Network(CNN) model. As for the language based model (Decoder) – Recurrent Neural Network is used. The image above here summarizes the approach.

A pre trained CNN draws out the features from the given input image. The feature vector is linearly transformed and is made sure to have same dimension as input dimension of our RNN/LSTM network. This network is trained as language model on the feature vectors we got.

## 7.2 Attention Mechanism

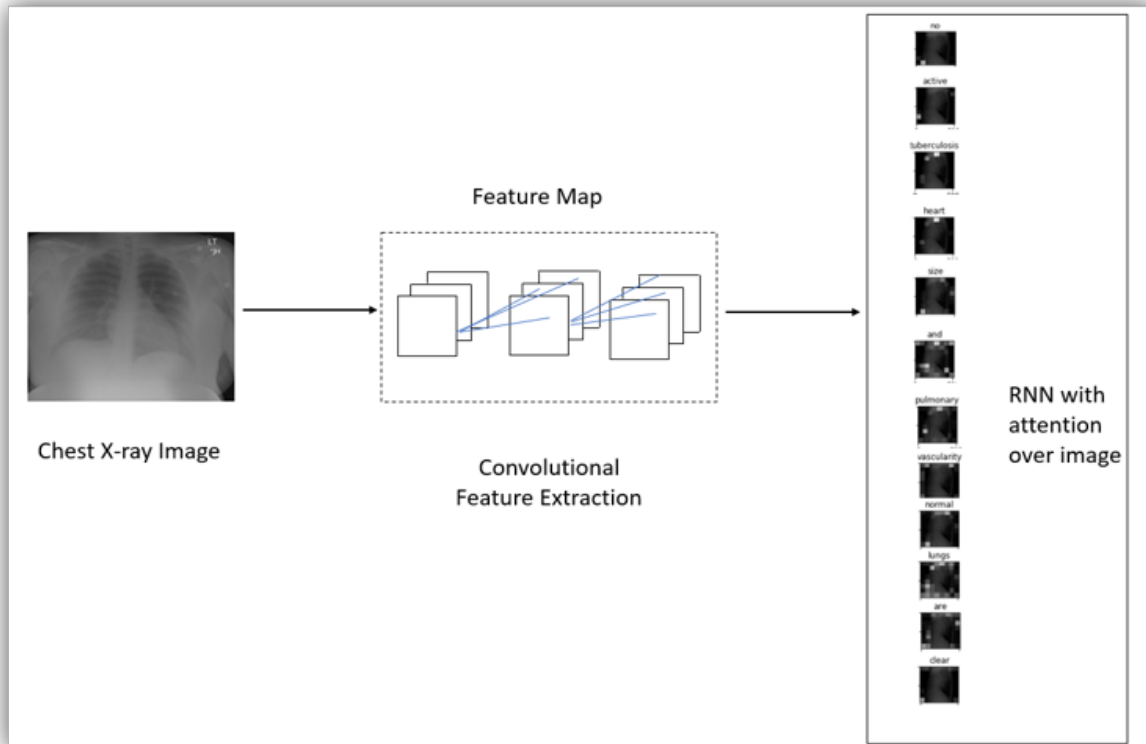


Fig 7.2- : Attention Mechanism used in our project

The Main problem with plain vanilla Encoder-Decoder method is, when model is trying to generate the next word of caption, the word is actually describing just a part of the image. It is not able to capture the essence of our full input image. The reason behind that using a whole representation of our image  $h$  to condition the generation of every word could not be done efficiently to produce different words for different parts of our image. This is where an Attention mechanism is provided for great help. With the help of Attention mechanism, the image is firstly divided into many  $n$  parts, and we then compute with CNN representations of each part for e.g.  $h1, \dots, hn$ . When the RNN is producing a new word every time, the attention mechanism aims to focus on the relevant part of our image, so the decoder only has to use some specific parts of the image.

## 7.3 Transfer Learning

Many Deep learning models are layered architectures that tend to learn from different features at different layers. These layers are finally connected to the last layer to get our final output. The layered architecture allows researchers to employ a pre-trained network (for e.g Inception V3) without the final layer as a fixed feature extractor for our very own task.

Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.

Assumes that  $D_S = D_T$

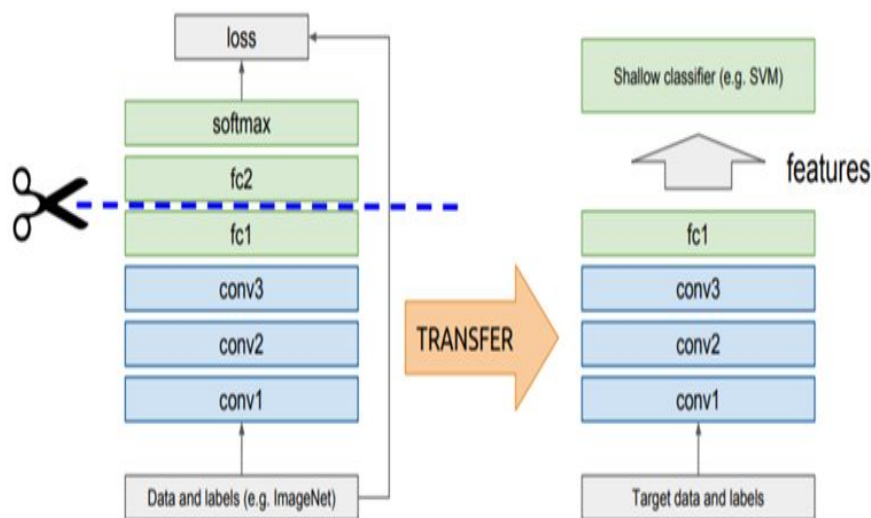


Fig 7.3 : Transfer Learning Example

Consider here, images are nothing but just an input to our model. Since any input to a deep learning model has to be given in a vector format. So we have to convert every image in our model to a fixed size vector which can be fed as an input to our neural network. For this purpose, we go for **transfer learning** by employing InceptionV3 model.

## CHAPTER-8

### IMPLEMENTATION AND CODE

#### 8.1 Download and Prepare IU X-Ray Dataset

There was no publicly readymade dataset available , so we had to perform web scraping on the Open Access Biomedical Image Search Engine. We downloaded the images and also the reports associated with them to get the captions. We have used the Findings and Impression part of the X-ray report for building the caption. The dataset was divided into training and testing with 90-10 split.

```
1  # PROJECT TITLE - "Image Captioning" for CHEST X-RAY Images
2
3  import os
4  from shutil import rmtree
5  import xml.etree.ElementTree as ET
6  import random
7  import numpy
8  import json
9
10
11 def split_cases(reports_images, reports_text, keys, filename):
12     new_images = {}
13
14     for key in keys:
15         for image in reports_images[key]:
16             new_images[image] = reports_text[key]
17
18     with open(filename, "w") as output_file:
19         for new_image in new_images:
20             output_file.write(new_image + "\t" + new_images[new_image])
21             output_file.write("\n")
22
23
24 # create dataset folder
25 try:
26     rmtree("iu_xray/")
27 except BaseException:
28     pass
29 os.makedirs("iu_xray/")
30
31 # download PNG images
32 os.system("wget -P iu_xray/ https://openi.nlm.nih.gov/imgs/collections/NLMCXR_png.tgz")
33
34 # download reports
35 os.system("wget -P iu_xray/ https://openi.nlm.nih.gov/imgs/collections/NLMCXR_reports.tgz")
36
37 # create folder for images
38 os.makedirs("iu_xray/iu_xray_images/")
39
```

Fig 8.1: Download the IU-X Ray dataset



```

40 # unzip
41 os.system("tar -xzf ./iu_xray/NLMCXR_png.tgz -C iu_xray/iu_xray_images/")
42 os.system("tar -xzf ./iu_xray/NLMCXR_reports.tgz -C iu_xray/")
43
44 # read the reports xml files and create the dataset tsv
45 reports_path = "iu_xray/ecgen-radiology"
46
47 reports = os.listdir(reports_path)
48
49 reports.sort()
50
51 reports_with_no_image = []
52 reports_with_empty_sections = []
53 reports_with_no_impression = []
54 reports_with_no_findings = []
55
56 images_captions = {}
57 images_major_tags = {}
58 images_auto_tags = {}
59 reports_with_images = {}
60 text_of_reports = {}
61
62 for report in reports:
63     tree = ET.parse(os.path.join(reports_path, report))
64     root = tree.getroot()
65     img_ids = []
66     # find the images of the report
67     images = root.findall("parentImage")
68     # if there aren't any ignore the report
69     if len(images) == 0:
70         reports_with_no_image.append(report)
71     else:
72         sections = root.find("MedlineCitation").find("Article").find("Abstract").findall("AbstractText")
73         # find impression and findings sections
74         for section in sections:
75             if section.get("Label") == "FINDINGS":
76                 findings = section.text
77             if section.get("Label") == "IMPRESSION":
78                 impression = section.text

```

Fig 8.2: Extracting Findings and Impressions

```

115 print("Collected", len(images_captions), "image-caption pairs")
116
117 with open("iu_xray/iu_xray.tsv", "w") as output_file:
118     for image_caption in images_captions:
119         output_file.write(image_caption + "\t" + images_captions[image_caption])
120         output_file.write("\n")
121
122 # Safer JSON storing
123 with open("iu_xray/iu_xray_captions.json", "w") as output_file:
124     output_file.write(json.dumps(images_captions))
125 with open("iu_xray/iu_xray_major_tags.json", "w") as output_file:
126     output_file.write(json.dumps(images_major_tags))
127 with open("iu_xray/iu_xray_auto_tags.json", "w") as output_file:
128     output_file.write(json.dumps(images_auto_tags))
129
130 # perform a case based split
131 random.seed(42)
132 keys = list(reports_with_images.keys())
133 random.shuffle(keys)
134
135 train_split = int(numpy.floor(len(reports_with_images) * 0.9))
136
137 train_keys = keys[:train_split]
138 test_keys = keys[train_split:]
139
140 train_path = "iu_xray/train_images.tsv"
141 test_path = "iu_xray/test_images.tsv"
142
143 split_cases(reports_with_images, text_of_reports, train_keys, train_path)
144 split_cases(reports_with_images, text_of_reports, test_keys, test_path)
145

```

Fig 8.3: Train-Test Split

## 8.2 Preprocess and tokenize the captions

This dataset contains several descriptions for every X-Ray and the caption of the X-Ray requires some minimal text cleaning.

First of all, we will load the file containing all of the captions.

```
In [3]: 1 # Load doc into memory
        2 def load_doc(filename):
        3     # open the file as read only
        4     file = open(filename, 'r')
        5     # read all text
        6     text = file.read()
        7     # close the file
        8     file.close()
        9     return text
        10
        11 filename = "token.tsv"
        12 # Load descriptions
        13 doc = load_doc(filename)
        14 print(doc[:300])

CXRI1597_IM-0388-1001.png      No acute abnormality. Heart and mediastinum within normal limits. Negative for focal pulmonary
consolidation, pleural effusion, or pneumothorax.
CXRI1597_IM-0388-2001.png      No acute abnormality. Heart and mediastinum within normal limits. Negative for focal pulmonary
consolidat
```

Fig 8.4: Loading the dataset

We will perform some text cleaning just to decrease the size of our vocabulary of words. This is done in several steps :

- First we convert all the words into lowercase.
- We then remove all punctuations.
- Then we take out all words that are of one character or less in length.
- Lastly we remove words with numbers inside them.

Below the `clean_descriptions()` function is defined that, given a dictionary of image identifiers to captions, processed through each and every description and performs cleaning on the text.

```
In [7]: 1 def clean_descriptions(descriptions):
2         # prepare translation table for removing punctuation
3         table = str.maketrans('', '', string.punctuation)
4         for key, desc_list in descriptions.items():
5             for i in range(len(desc_list)):
6                 desc = desc_list[i]
7                 # tokenize
8                 desc = desc.split()
9                 # convert to lower case
10                desc = [word.lower() for word in desc]
11                # remove punctuation from each token
12                desc = [w.translate(table) for w in desc]
13                # remove hanging 's' and 'a'
14                desc = [word for word in desc if len(word)>1]
15                # remove tokens with numbers in them
16                desc = [word for word in desc if word.isalpha()]
17                # store as string
18                desc_list[i] = ' '.join(desc)
19
20        # clean descriptions
21        clean_descriptions(descriptions)
```

Fig 8.5: Text Cleaning

```
1 # Find the maximum length of any caption in our dataset
2 def calc_max_length(tensor):
3     return max(len(t) for t in tensor)

1 # Choose the top 3000 words from the vocabulary
2 top_k = 3000
3 tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=top_k,
4                                                    oov_token="<unk>",
5                                                    filters='!"#$%&()*+.,-/:;=?@[\]^_`{|}~ ')
6 tokenizer.fit_on_texts(train_captions)
7 train_seqs = tokenizer.texts_to_sequences(train_captions)

1 tokenizer
<keras_preprocessing.text.Tokenizer at 0x19672e6a888>

1 tokenizer.word_index['<pad>'] = 0
2 tokenizer.index_word[0] = '<pad>'

1 # Create the tokenized vectors
2 train_seqs = tokenizer.texts_to_sequences(train_captions)

1 # Pad each vector to the max_length of the captions
2 # If you do not provide a max_length value, pad_sequences calculates it automatically
3 cap_vector = tf.keras.preprocessing.sequence.pad_sequences(train_seqs, padding='post')

1 # Calculates the max_length, which is used to store the attention weights
2 max_length = calc_max_length(train_seqs)
```

Fig 8.6: Preprocessing captions

```
In [34]: 1 all_captions
<start> no acute pulmonary abnormality the lungs and pleural spaces show no acute abnormality heart size and pulmonary vasc
ularity within normal limits <end>',
'<start> no acute pulmonary abnormality the lungs and pleural spaces show no acute abnormality heart size and pulmonary vasc
ularity within normal limits <end>',
'<start> no active disease both lungs are clear and expanded heart and mediastinum normal <end>',
'<start> no active disease both lungs are clear and expanded heart and mediastinum normal <end>',
'<start> borderline heart size no acute pulmonary findings heart size borderline enlarged no focal alveolar consolidation no
definite pleural effusion seen no typical findings of pulmonary edema dense nodule in the right lower lobe suggests previous
granulomatous process <end>',
'<start> borderline heart size no acute pulmonary findings heart size borderline enlarged no focal alveolar consolidation no
definite pleural effusion seen no typical findings of pulmonary edema dense nodule in the right lower lobe suggests previous
granulomatous process <end>',
'<start> normal chest heart size normal lungs are clear xxxx are normal no pneumonia effusions edema pneumothorax adenopathy
nodules or masses <end>',
'<start> normal chest heart size normal lungs are clear xxxx are normal no pneumonia effusions edema pneumothorax adenopathy
nodules or masses <end>',
'<start> comparison xxxx xxxx no suspicious appearing lung nodules wellexpanded and clear lungs mediastinal contour within n
ormal limits no acute cardiopulmonary abnormality identified stable chest <end>',
'<start> comparison xxxx xxxx no suspicious appearing lung nodules wellexpanded and clear lungs mediastinal contour within n
ormal limits no acute cardiopulmonary abnormality identified stable chest <end>',
'<start> endoste right-sided hemothorax as detailed above probable lateral right rib fracture endotracheal tube at the

In [35]: 1 train_captions
Out[35]: ['<start> no acute cardiopulmonary disease lungs appear clear heart and pulmonary xxxx appear normal pleural spaces are clear
mediastinal contours are normal no pneumothorax <end>',
'<start> no acute cardiopulmonary disease lungs appear clear heart and pulmonary xxxx appear normal pleural spaces are clear
mediastinal contours are normal no pneumothorax <end>',
'<start> no acute or active cardiac pulmonary or pleural disease frontal and lateral views of the chest with overlying exter
nal cardiac monitor leads show an unchanged cardiomeastinal silhouette no xxxx focal airspace consolidation or pleural effu
sion <end>',
'<start> no acute or active cardiac pulmonary or pleural disease frontal and lateral views of the chest with overlying exter
nal cardiac monitor leads show an unchanged cardiomeastinal silhouette no xxxx focal airspace consolidation or pleural effu
sion <end>',
'<start> no acute disease the heart is normal in size the mediastinum is unremarkable xxxx xxxx opacities in right mid lung
the lungs are otherwise grossly clear <end>',
```

Fig 8.7: Cleaned and pre-processed tokens

## 8.3 Preprocess the images using InceptionV3

Now, we use the InceptionV3 model to classify every image. We extract features from our last convolutional layer, to get a dense representation.

For doing that, we first converted the images into InceptionV3's presumed format. This is done by:

- First of all we resize the image to a size of 299px X 299px.
- After that we preprocess our images using our very own preprocess\_input function that normalizes our image so that all the images contain pixels only in the range of  $[-1, 1]$  which matches with the expected format of what InceptionV3 module uses to train.

```
1 def load_image(image_path):
2     img = tf.io.read_file(image_path)
3     img = tf.image.decode_jpeg(img, channels=3)
4     img = tf.image.resize(img, (299, 299))
5     img = tf.keras.applications.inception_v3.preprocess_input(img)
6     return img, image_path
```

Fig 8.8: Preprocessing the Images



## 8.4 Initialize InceptionV3 module & load its pre-trained weights

We just have to convert each image into an allotted sized vector which can then be supplied as an input to our neural network. For this, we take help of **transfer learning** by employing the InceptionV3 module.

We then create a tf.keras model where our output layer is the last CNN layer in the architecture of InceptionV3. The shape of this output layer is '8x8x2048'. We have employed the last CNN layer since we are using attention here.

- We pass on each X-Ray image through the network & save the resultant vector in a dictionary in this format {image\_name : feature\_vector}.
- When all of the images progress through the network, we then pickle our dictionary and save it in our disk

```
1 image_model = tf.keras.applications.InceptionV3(include_top=False,  
2                                             weights='imagenet')  
3 new_input = image_model.input  
4 hidden_layer = image_model.layers[-1].output  
5  
6 image_features_extract_model = tf.keras.Model(new_input, hidden_layer)
```

Fig 8.9: Loading the Inception V3 module

Inception V3 model has been trained on the Imagenet dataset to carry out image classification task on over a 1000 different classes of images. However, in our use case we don't have to classify the image, rather we need to get a fixed-length dense vector for which features of the image are compressed. This whole method is known as **automatic feature engineering**.

So, we just take off the last softmax layer from Inception V3 model and, extract a 2048 length vector for each image as shown in the diagram:

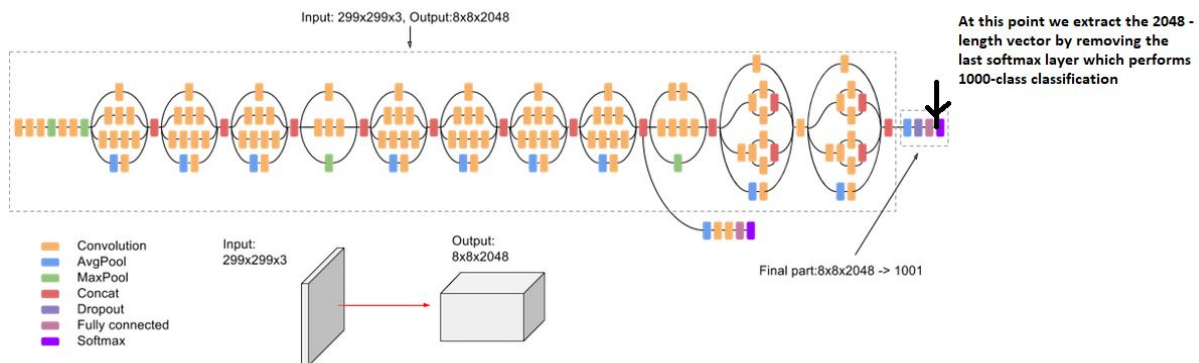


Fig 8.10: Using the Inception V3 as our Encoder

## 8.5 Model

In our task we need to predict the captions. So for the training phase, our target variable will be the caption that the model has to learn to predict.

But we have to know that the prediction of the whole caption, given the image does not happen all at once. The model is built in such a way that it predicts the caption word by word. So, for that we encode every word into a fixed size vector.

- Features are extracted from the lower CNN layer of InceptionV3 module giving a vector of shape [8, 8, 2048].
- It is then squashed to a shape of [64, 2048].
- This vector is passed through CNN Encoder.
- The RNN attends over each image to predict the next consecutive word.

### 8.5.1 Encoder

The Convolutional Neural Network(CNN) can be looked at as an encoder. Every input image is supplied to our CNN which extracts the features. The last hidden state of CNN module is linked to the Decoder.

```
class CNN_Encoder(tf.keras.Model):
    # Since you have already extracted the features and dumped it using pickle
    # This encoder passes those features through a Fully connected Layer
    def __init__(self, embedding_dim):
        super(CNN_Encoder, self).__init__()
        # shape after fc == (batch_size, 64, embedding_dim)
        self.fc = tf.keras.layers.Dense(embedding_dim)

    def call(self, x):
        x = self.fc(x)
        x = tf.nn.relu(x)
        return x
```

Fig 8.11: CNN Encoder

## 8.5.2 Decoder

The Decoder here is a Recurrent Neural Network which does the task of modelling the language to the word level. The 1st time step gets the output from the encoder and also the <start> token.

```
class RNN_Decoder(tf.keras.Model):
    def __init__(self, embedding_dim, units, vocab_size):
        super(RNN_Decoder, self).__init__()
        self.units = units

        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(self.units,
                                        return_sequences=True,
                                        return_state=True,
                                        recurrent_initializer='glorot_uniform')

        self.fc1 = tf.keras.layers.Dense(self.units)
        self.fc2 = tf.keras.layers.Dense(vocab_size)

        self.attention = BahdanauAttention(self.units)

    def call(self, x, features, hidden):
        # defining attention as a separate model
        context_vector, attention_weights = self.attention(features, hidden)

        # x shape after passing through embedding == (batch_size, 1, embedding_dim)
        x = self.embedding(x)

        # x shape after concatenation == (batch_size, 1, embedding_dim + hidden_size)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

        # passing the concatenated vector to the GRU
        output, state = self.gru(x)

        # shape == (batch_size, max_length, hidden_size)
        x = self.fc1(output)

        # x shape == (batch_size * max_length, hidden_size)
        x = tf.reshape(x, (-1, x.shape[2]))

        # output shape == (batch_size * max_length, vocab)
        x = self.fc2(x)

        return x, state, attention_weights

    def reset_state(self, batch_size):
        return tf.zeros((batch_size, self.units))
```

Fig 8.12: RNN Decoder

### 8.5.3 Attention Mechanism

The Attention mechanism revolutionised the way NLP models were created and is right now a standard fixture in most of the state-of-the-art NLP models. The reason for this is that it helped the model to “recall” all the words given in input and also focus on some specific words when generating a valid response.

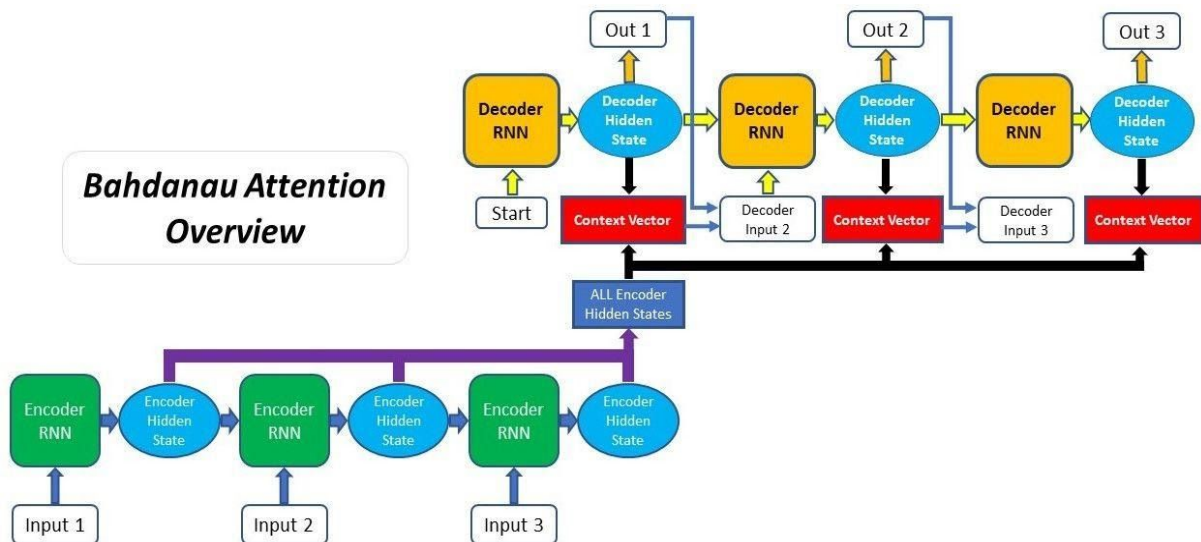


Fig 8.13: Bahdanau Attention

```
class BahdanauAttention(tf.keras.Model):
    def __init__(self, units):
        super(BahdanauAttention, self).__init__()
        self.W1 = tf.keras.layers.Dense(units)
        self.W2 = tf.keras.layers.Dense(units)
        self.V = tf.keras.layers.Dense(1)

    def call(self, features, hidden):
        # features(CNN_encoder output) shape == (batch_size, 64, embedding_dim)

        # hidden shape == (batch_size, hidden_size)
        # hidden_with_time_axis shape == (batch_size, 1, hidden_size)
        hidden_with_time_axis = tf.expand_dims(hidden, 1)

        # score shape == (batch_size, 64, hidden_size)
        score = tf.nn.tanh(self.W1(features) + self.W2(hidden_with_time_axis))

        # attention_weights shape == (batch_size, 64, 1)
        # you get 1 at the last axis because you are applying score to self.V
        attention_weights = tf.nn.softmax(self.V(score), axis=1)

        # context_vector shape after sum == (batch_size, hidden_size)
        context_vector = attention_weights * features
        context_vector = tf.reduce_sum(context_vector, axis=1)

        return context_vector, attention_weights
```

Fig 8.14: Attention mechanism in code



## 8.6 Training Phase

1. We extract features stored in the .npy files and these are then passed to the encoder.
2. The encoder output, hidden state and the decoder input is passed to our very own decoder.
3. The decoder gives back the predictions and also the hidden state of decoder.
4. This hidden state is passed back into our model which makes predictions, and uses them to calculate the loss.
5. We have employed teacher forcing to determine the next input passed to the decoder.
6. Teacher forcing is the method where the target word is usually progressed as the next input to our decoder.
7. Our final step will be to calculate the gradients and use it by applying them to the optimizer and backpropagate them again.

```
@tf.function
def train_step(img_tensor, target):
    loss = 0

    # initializing the hidden state for each batch
    # because the captions are not related from image to image
    hidden = decoder.reset_state(batch_size=target.shape[0])

    dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * target.shape[0], 1)

    with tf.GradientTape() as tape:
        features = encoder(img_tensor)

        for i in range(1, target.shape[1]):
            # passing the features through the decoder
            predictions, hidden, _ = decoder(dec_input, features, hidden)

            loss += loss_function(target[:, i], predictions)

            # using teacher forcing
            dec_input = tf.expand_dims(target[:, i], 1)

    total_loss = (loss / int(target.shape[1]))

    trainable_variables = encoder.trainable_variables + decoder.trainable_variables
    gradients = tape.gradient(loss, trainable_variables)
    optimizer.apply_gradients(zip(gradients, trainable_variables))

    return loss, total_loss
```

Fig 8.15: Training the model

We trained the model progressively for about 80 epochs.

## Image Captioning for Chest X-Ray Images

```
In [125]: 1 EPOCHS = 80
2
3 for epoch in range(start_epoch, EPOCHS):
4     start = time.time()
5     total_loss = 0
6
7     for (batch, (img_tensor, target)) in enumerate(dataset):
8         batch_loss, t_loss = train_step(img_tensor, target)
9         total_loss += t_loss
10
11         if batch % 100 == 0:
12             print('Epoch {} Batch {} Loss {:.4f}'.format(
13                 epoch + 1, batch, batch_loss.numpy() / int(target.shape[1])))
14         # storing the epoch end loss value to plot later
15         loss_plot.append(total_loss / num_steps)
16
17     if epoch % 5 == 0:
18         ckpt_manager.save()
19
20     print('Epoch {} Loss {:.6f}'.format(epoch + 1,
21                                         total_loss/num_steps))
22     print('Time taken for 1 epoch {} sec\n'.format(time.time() - start))
```

```
Epoch 6 Loss 0.028914
Time taken for 1 epoch 1255.580708026886 sec

Epoch 7 Batch 0 Loss 0.0292
Epoch 7 Loss 0.033883
Time taken for 1 epoch 1162.1868648529053 sec

Epoch 8 Batch 0 Loss 0.0399
Epoch 8 Loss 0.034274
Time taken for 1 epoch 1315.1560904979706 sec

Epoch 9 Batch 0 Loss 0.0303
Epoch 9 Loss 0.032386
Time taken for 1 epoch 1225.5860965251923 sec

Epoch 10 Batch 0 Loss 0.0284
Epoch 10 Loss 0.027001
Time taken for 1 epoch 1109.3273606300354 sec
```

Fig 8.16: Trained for approx 80 epochs

Here is the loss plot for our model, showing Loss(Y-axis) vs Number of Epochs(X-axis)

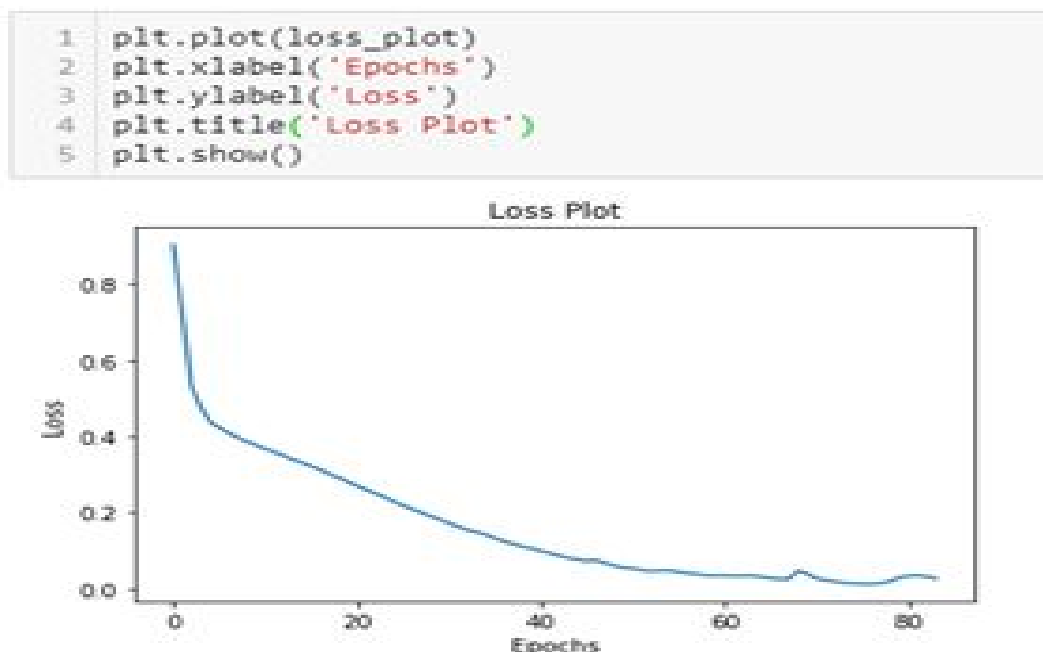


Fig 8.17: Loss plot

Here is a sample caption generated from this particular photo, with attention mechanism attending over specific parts of the image.

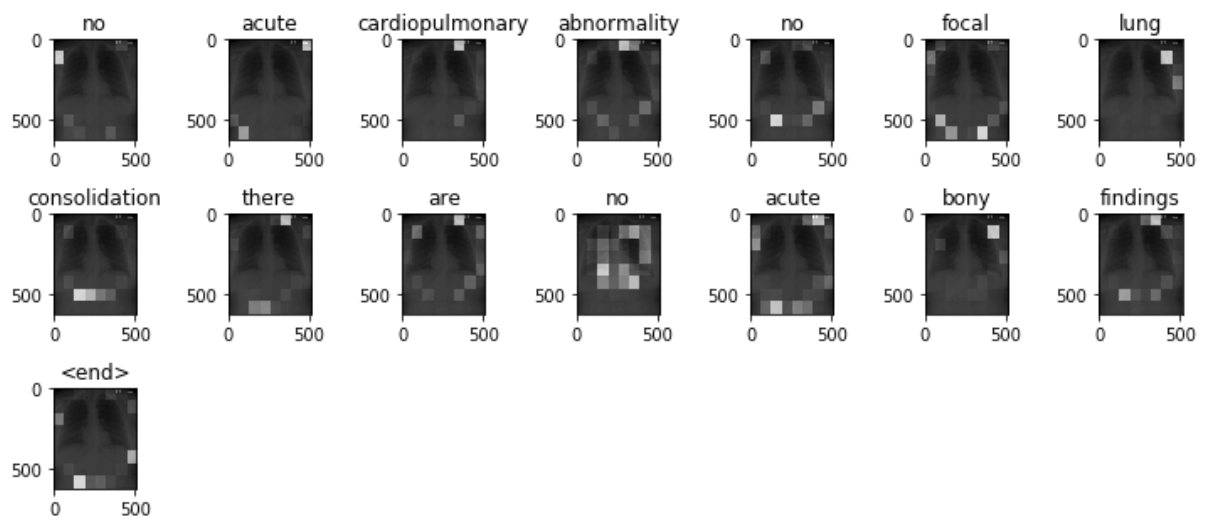


Fig 8.18 Sample caption with attention

## CHAPTER-9

### TESTING

Now after our model was ready , and was generating captions, we had to plan for evaluation of our model. The evaluation metrics we used specifically designed for calculating accuracy of machine generated text.

We used BLEU, METEOR , ROGUE scores for testing our model.

We also utilized cosine-similarity to measure similarity between real and predicted caption.

## CHAPTER-10

### RESULTS AND DISCUSSION

#### 10.1 Results

We have evaluated our model on the basis of four metrics. They are Cosine Similarity, BLEU score, ROUGE and METEOR. Also we have compared our model with different reference papers which have used the same IU-Xray dataset for their project. The results which we generated are-

Reference Paper	Cosine Similarity	BLEU	METEOR	ROUGE
Frequency Model	NA	0.442	0.176	0.187
Nearest Neighbour	NA	0.281	0.125	0.209
CNN-RNN(Vinyals et al, 2015)	NA	0.316	0.159	0.267
LRCN(Donahue et al, 2015)	NA	0.369	0.155	0.278
Soft Att( Xu et al, 2015)	NA	0.399	0.167	0.323
ATT-RK(You et al, 2016)	NA	0.369	0.171	0.323
CNN-LSTM(Sonit Singh, 2019)	NA	0.374	0.163	0.307
Co-Attention(Jing et al, 2018)	NA	0.369	0.217	0.447
<b>Our Model</b>	<b>0.325</b>	<b>0.452</b>	<b>0.201</b>	<b>P-0.308</b> <b>R-0.328</b> <b>F-0.267</b>

Table 10.1: Comparison of our model with other referred models

## 10.2 Discussions

The outcome of our model seems encouraging, but there are certain issues which need to be addressed before deploying the automated report generation into the picture. The reasons being-

1. The current recurrent neural networks are not suitable for long pathology or radiology reports, because they cannot save the long range dependencies.
2. There is a dire need for separate evaluation metrics to evaluate system generated radiology reports. BLEU score works on the basis of overlap of words, which can calculate wrong accuracy. For a real caption like *“There is no evidence of pleural effusion, cardiopulmonary abnormality and pneumothorax”* and the generated *“There is pneumothorax and cardiopulmonary abnormality . No evidence of pleural effusion”* we will get a BLEU score of 100% even though the predicted caption is opposite of the real.
3. Medical datasets generally lack manually annotated bounding boxes for abnormal and normal regions, which makes it difficult for a Convolutional Neural Network to get fine-grained features separated out.

## Chapter-11

### SNAPSHOTS

```

import rouge
import nltk
nltk.download('punkt')

def prepare_results(p, r, f):
    return '\t{}\t{}: {:.2f}\t{}: {:.2f}'.format(metric, 'P', 100.0 * p, 'R', 100.0 * r, 'F1', 100.0 * f)

for aggregator in ['Avg', 'Individual']:
    print('Evaluation with {}'.format(aggregator))
    apply_avg = aggregator == 'Avg'

    evaluator = rouge.Rouge(metrics=['rouge-n', 'rouge-l', 'rouge-w'],
                             max_n=4,
                             limit_length=True,
                             length_limit=100,
                             length_limit_type='words',
                             apply_avg=apply_avg,
                             # apply_best=apply_best,
                             alpha=0.5, # Default F1_score
                             weight_factor=1.2,
                             stemming=True)

    scores = evaluator.get_scores(realValue, predictedValue)

    for metric, results in sorted(scores.items(), key=lambda x: x[0]):
        if not apply_avg: # value is a type of list as we evaluate each summary vs each reference
            for hypothesis_id, results_per_ref in enumerate(results):
                nb_references = len(results_per_ref['p'])
                # for reference_id in range(nb_references):
                #     print('\tHypothesis #{} & Reference #{}: '.format(hypothesis_id, reference_id))
                #     print('\t' + prepare_results(results_per_ref['p'][reference_id], results_per_ref['r'][reference_id], results_per_ref['f'][reference_id]))
            print()
        else:
            print(prepare_results(results['p'], results['r'], results['f']))
    print()

```

Fig-11.1 Code for ROGUE score calculation

```

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
Evaluation with Avg
rouge-1:      P: 30.81      R: 32.85      F1: 26.36
rouge-2:      P:  9.70      R:  9.40      F1:  7.95
rouge-3:      P:  4.10      R:  3.77      F1:  3.27
rouge-4:      P:  1.61      R:  1.44      F1:  1.25
rouge-l:      P: 29.26      R: 31.99      F1: 26.45
rouge-w:      P: 18.63      R: 12.11      F1: 11.15

```

Fig- 11.2 Output for ROUGE average score

```
[ ] avg_ms=0
score_list=[]
list1m=[]
list2m=[]
list3m=[]
list4m=[]
list5m=[]
list6m=[]
for i in range(0,len(realValue)):
    accuracy=(round(single_meteor_score(predictedValue[i],realValue[i]),4))
    if(accuracy>=0.6):
        list1m.append(imageName[i])
    elif(accuracy>=0.5 and accuracy<0.6):
        list2m.append(imageName[i])
    elif(accuracy>=0.4 and accuracy<0.5):
        list3m.append(imageName[i])
    elif(accuracy>=0.3 and accuracy<0.4):
        list4m.append(imageName[i])
    elif(accuracy>=0.2 and accuracy<0.3):
        list5m.append(imageName[i])
    else:
        list6m.append(imageName[i])
    score_list.append(accuracy)
avg_ms=(0.65*3) + (0.55*8) +(0.45*43) + (0.35*125) + (0.25*255)+ (0.15*901)
print("Meteor Score is - ->",avg_ms/1335)
print(score_list)
```

```
☞ Meteor Score is - -> 0.20101123595505618
[0.1812, 0.1712, 0.1441, 0.1183, 0.1115, 0.1068, 0.3453, 0.1792, 0.0496, 0.1351, 0
```

Fig- 11.3 Code and Output for METEOR score

```
☞ /usr/local/lib/python3.6/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 3-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
    warnings.warn(_msg)
/usr/local/lib/python3.6/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 4-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
    warnings.warn(_msg)
/usr/local/lib/python3.6/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 2-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
    warnings.warn(_msg)
0.4762248034116116
```

Fig-11.4 Output for BLEU score



```
m=0
for i in range(0,len(realValue)):
    real_list=realValue[i].split()
    predicted_list=predictedValue[i].split()
    real_new=[]
    predicted_new=[]
    for j in real_list:
        if j not in word_list:
            real_new.append(j)
    for j in predicted_list:
        if j not in word_list:
            predicted_new.append(j)
    real_string=""
    predicted_string=""
    for ele in real_new:
        real_string=real_string+ele+" "
    for ele in predicted_new:
        predicted_string=predicted_string+ele+" "
    documents=[real_string,predicted_string]
    BLEUScore = nltk.translate.bleu_score.sentence_bleu([real_new],predicted_new,weights = (1,0,0,0))
    accuracy=BLEUScore
    if (accuracy>=0.1 and accuracy<0.2):
        list1b.append(imageName[i])
    elif (accuracy>=0.2 and accuracy<0.3):
        list2b.append(imageName[i])
    elif (accuracy>=0.3 and accuracy<0.4):
        list3b.append(imageName[i])
    elif (accuracy>=0.4):
        list4b.append(imageName[i])
    else:
        list5b.append(imageName[i])
    if (BLEUScore>m):
        m=BLEUScore
print(m)
```

Fig- 11.5 Code to calculate BLEU score.

```
[ ] for i in range(0,len(realValue)):
    real_list=realValue[i].split()
    predicted_list=predictedValue[i].split()
    real_new=[]
    predicted_new=[]
    for j in real_list:
        if j not in word_list:
            real_new.append(j)
    for j in predicted_list:
        if j not in word_list:
            predicted_new.append(j)
    real_string=""
    predicted_string=""
    for ele in real_new:
        real_string=real_string+ele+" "
    for ele in predicted_new:
        predicted_string=predicted_string+ele+" "
    documents=[real_string,predicted_string]
    count_vectorizer = CountVectorizer(stop_words='english')
    count_vectorizer = CountVectorizer()
    sparse_matrix = count_vectorizer.fit_transform(documents)
    doc_term_matrix = sparse_matrix.todense()
    df = pd.DataFrame(doc_term_matrix,
                      columns=count_vectorizer.get_feature_names(),
                      index=['real', 'predict'])
    accuracy=cosine_similarity(df,df)[0][1]
    if(accuracy>=0.5 and accuracy<0.6):
        list1.append(imageName[i])
    elif(accuracy>=0.6 and accuracy<0.7):
        list2.append(imageName[i])
    elif(accuracy>=0.7 and accuracy<0.8):
        list3.append(imageName[i])
    elif(accuracy>=0.8):
        list4.append(imageName[i])
    else:
        list5.append(imageName[i])
```

Fig- 11.6 Cosine Similarity code for evaluation of model.

## Chapter- 12

### CONCLUSIONS

In this project we have tried to automatically generate the radiology reports for chest X-ray images, with the main agenda to help radiologists and medical professionals to produce reports more accurately, efficiently and in a faster way. We have used the Indiana University publicly available dataset IU-Xray which contains 7470 images along with medical reports for each image. We proposed a solution which uses Inception-V3 and Attention mechanism to help generate the caption for medical image. Also we have used the techniques of Transfer learning and Teacher Forcing in our model.

Seeing the evaluation result, our project work seems to be very encouraging. But working in the field of medical domain tells us that automatic radiology report generation is a challenging task and is very much different from the normal image captioning. Therefore, there is a need of more advanced RNN models which will help for better caption prediction

## Chapter- 13

### FUTURE WORK

As from the evaluation result it is very much clear that the model using Co- Attention with hierarchical LSTM generates a better result for the radiology reports, we would like to use it in our model, and replace our single attention model. But the other model has used VGG-19 as their encoder, which has more error rate than Inception-V3. We believe implementing Co-Attention with V3 models can help us get a better result and surpass all the other models present.

Also there needs to be more research done on the evaluation of an Image Caption Model, since the evaluation metrics out there do not take medical context into account. This metric has to be worked with the assistance from the medical field . We could possibly try to use Rad-Glove for generating vector representation of captions.

---

## REFERENCES / BIBLIOGRAPHY

- [1] Learning to Summarize Radiology Findings , Daisy Yi Ding, Yuhao Zhang, Tianpei Qian,  
Curtis P. Langlotz, Christopher D. Manning. Published in Louhi@EMNLP 2018.
- [2] From Chest X-rays to Radiology Reports: A Multimodal Machine Learning Approach.  
Sarvnaz Karimi, Sonit Singh, Len Hamey, Kevin Ho-Shon, Department of Computing,  
Macquarie University, Sydney, Australia. Macquarie University Health Sciences Centre,  
Sydney, Australia Published 2019 Digital Image Computing: Techniques and Applications  
(DICTA)
- [3] A Survey on Biomedical Image Captioning. John Pavlopoulos, Vasiliki Kougia, Ion  
Androutsopoulos. Department of Informatics, Athens University of Economics and Business,  
Greece.
- [4] On the Automatic Generation of Medical Imaging Reports. Pengtao Xie, Baoyu Jing, Eric  
Xing.
- [5] A Comprehensive Survey of Deep Learning for Image Captioning.  
Ferdous Sohel, Md. Zakir Hossan , Hamid Laga, Mohd Fairuz Shiratuddin
- [6] Medical image captioning: learning to describe medical, image findings using  
multi-task-loss CNN. Eli Sason, Pavel Kisilev, Sharbell Hashoul, Ella Barkanl. IBM Haifa  
Research Lab, Haifa, Israel, Carmel Medical Center, Haifa, Israel

## REFERENCES / BIBLIOGRAPHY

- <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- <https://towardsdatascience.com/evaluating-text-output-in-nlp-bleu-at-your-own-risk-e8609665a213>

## APPENDIX-A

### DEFINITIONS, ACRONYMS AND ABBREVIATIONS

The keywords given below are used in the document unambiguously. The meaning of these keywords remain as defined here, unless indicated otherwise :

- **RNN:** Recurrent Neural Network. Are a network of neuron-like nodes organized into successive layers. Each and every node in a given layer are connected with a one-way(directed) connection to the every other node in the next layer below or the successive layer.
- **CNN:** Convolutional Neural Network. It is a deep learning algo which takes in an image as an input, assigns important ( biases and learnable weights) to various objects in the image and is able to differentiate one another.
- **LSTM:** Long-Short Term Memory. It has a very similar control flow as a (RNN) recurrent neural network. It processes data which passes on information as it propagates forward. The differences are between the operations within the LSTM's gates and cells.
- **VGG:** Visual Geometry Group , refers to a deep convolutional neural network for recognition of objects trained and developed by Oxford's renowned group, which achieved extremely good performance on the ImageNet dataset.
- **TensorFlow:** TensorFlow is an open source library for large-scale and numerical computation machine learning. TensorFlow bundles together a heap of machine learning and deep learning (also known as neural networking) algorithms and models and makes them useful by a common technique application.
- **Teacher Forcing:** Teacher forcing is the method where the target word is usually passed in as the next input to the decoder.