Table of Contents

Automatic Impression Generation from Medical Imaging Report

Process of generating textual description from medical report – end-to-end Deep learning model

1 Business Problem

The problem statement here is to find the impression from the given chest X-Ray images. These images are in two types Frontal and Lateral view of the chest. With these two types of images as input we need to find the impression for given X-Ray.

To achieve this problem, we will be building a predictive model which involves both image and text processing to build a deep learning model. Automatically describing the content of the given image is one of the recent artificial intelligence models that connects both computer vision and natural language processing.

2 Introduction about the Dataset

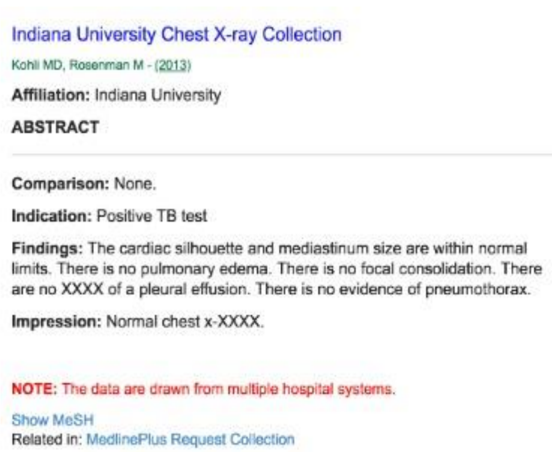
Open-i chest X-ray collection from Indiana University

This dataset is about 7,470 chest x-rays with 3,955 radiology reports for the chest x-ray images from Indiana university hospital network. - Images are downloaded as png format - Reports are downloaded as xml format.

Each xml is the report for corresponding patient. To identify images associated with the reports we need to check the xml tag <parentImages id="image-id"> id attribute in the id we have the image name corresponding to the png images. More than one images could be associated with one report or xml.

Original data source: https://openeni.nlm.nih.gov/
Other Resources: https://www.kaggle.com/raddar/chest-xrays-indiana-university

Sample Data point:


Indiana University Chest X-ray Collection

Kohli MD, Rosenman M - (2013)

Affiliation: Indiana University

ABSTRACT

Comparison: None.

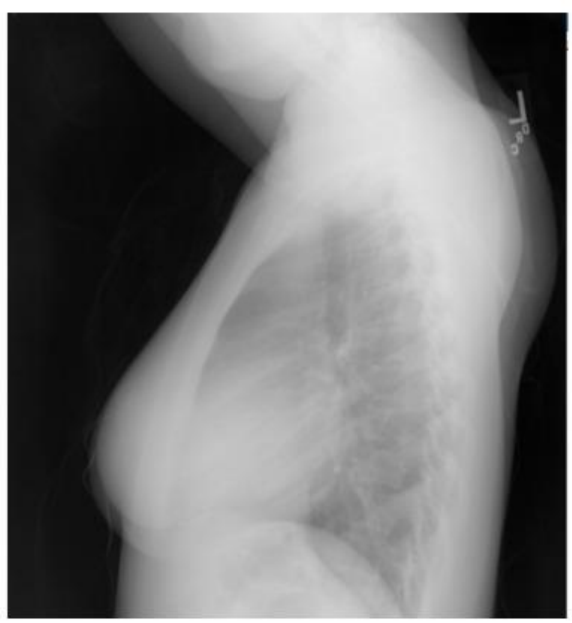
Indication: Positive TB test

Findings: The cardiac silhouette and mediastinum size are within normal limits. There is no pulmonary edema. There is no focal consolidation. There are no XXXX of a pleural effusion. There is no evidence of pneumothorax.

Impression: Normal chest x-XXXX.

NOTE: The data are drawn from multiple hospital systems.

Show MeSH
Related in: MedlinePlus Request Collection

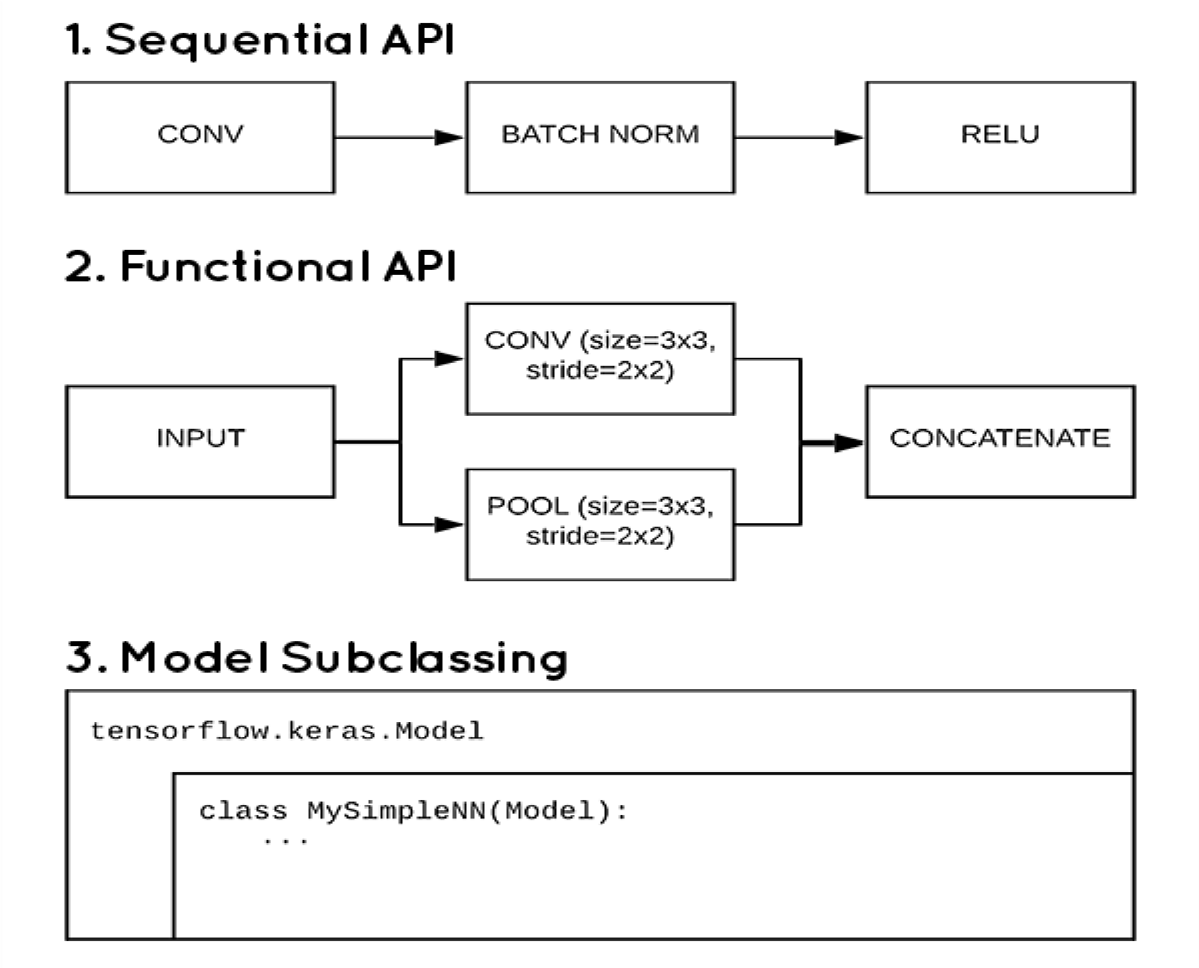


3 Prerequisite

Before we go through deep on this work, I assume that you are familiar with the following deep learning concepts and python libraries.

Convolution Neural Network, Recurrent Neural Network, LSTM, Transfer learning, Activation functions, Optimization techniques like SGD, Adam. Loss functions like categorical cross entropy, sparse categorical cross entropy. Finally, TensorBoard for performance visualization and debugging

Python, tensorflow, Keras, tokenizer, Pandas, numpy, Matplotlib. Understanding concept of Sequential Api, Functional Api and model subclass type keras model implementation. The reason I have chosen the subclasse model is, it is **fully-customizable** and enables you to **implement your own custom forward-pass** of the model. Also we can have control over every nuance of the network and training process.



Below I have mentioned import blogs and tutorials.

1. https://www.tensorflow.org/tutorials/text/nmt_with_attention - TensorFlow Tutorial
2. https://www.tensorflow.org/tutorials/text/image_captioning - TensorFlow Tutorial
3. https://becominghuman.ai/transfer-learning-retraining-inception-v3-for-custom-image-classification-2820f653c557 - Transfer Learning tutorial
4. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202 - InceptionV3 model tutorial

5. https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/ - why ImageNet – why InceptionV3
6. https://www.pyimagesearch.com/2019/10/28/3-ways-to-create-a-keras-model-with-tensorflow-2-0-sequential-functional-and-model-subclassing/ - 3 ways Keras model implementation
7. https://www.tensorflow.org/tensorboard/get_started - TensorBoard Tutorial

4 Existing Research-Papers/Solutions

This work is inspired from the below research and Blog:

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

In the mentioned paper they have used encoder and decoder model with attention mechanism. In the encoder part they have used CNN to extract feature from images. In decoder they use a long short-term memory (LSTM) network that produces a caption by generating one word at every time step conditioned on a context vector, the previous hidden state and the previously generated words. They have used BLEU score to measure the performance of the model.

Few other Blogs i have referenced.

1. https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2
2. https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/

5 My Approach – Solution

Initially I will be doing the Exploratory Data Analysis part I both image input and text output with EDA I could find the data imbalance, Images availability per patient, Type of images associated for each patient. After the EDA I will be implementing deep learning model with two different approach to find the improvement on one another.

1. The basic model:

   A simple encoder and decoder architecture. In encoder part it will have the CNN single fully connected layer to get the feature vector of images from pretrained InceptionV3 model. Decoder part will be having LSTM layer where it takes two inputs one is image feature vector and the sequence of text to word in each time step.

2. Main Model:

   I will be using encoder-decoder architecture to generate the impression from the chest X-ray. The encoder will output the image feature vectors. The feature vectors are then passed to decoder with attention mechanism this will generate the next word for the content of the image. With same model approach from basic model I will be creating a new architecture which is implemented using the research paper Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification

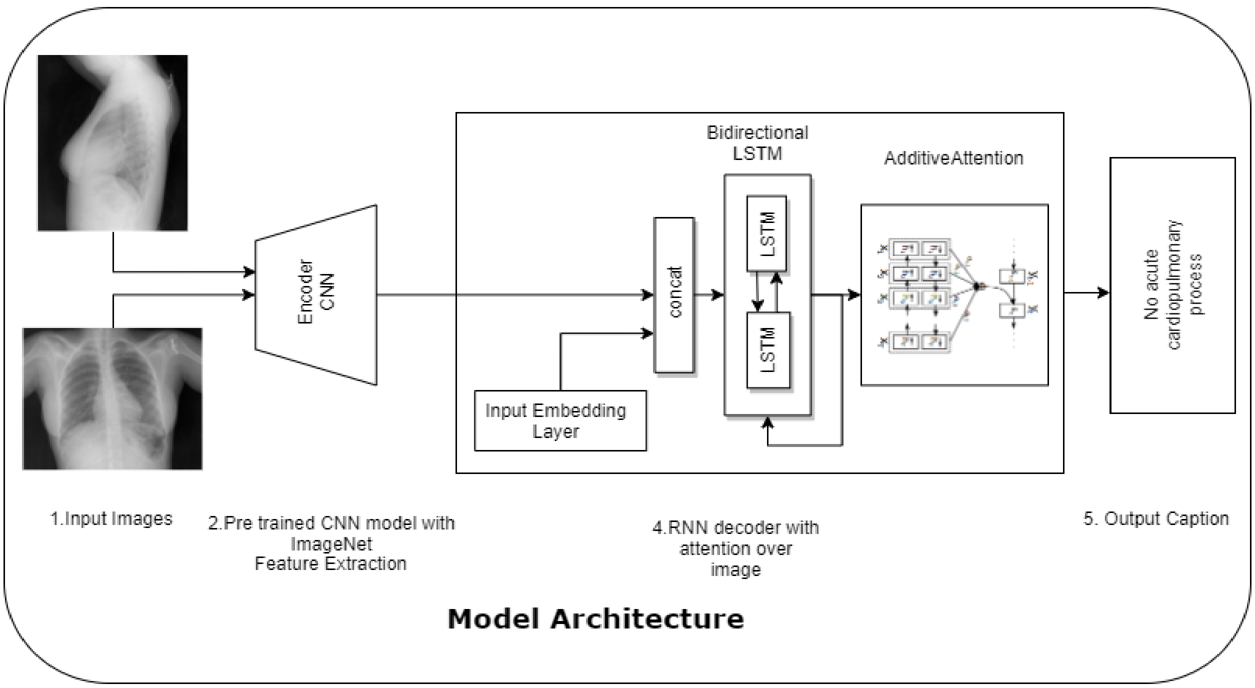
   As initial step I will do an image classification using InceptionV3 model over this dataset https://www.kaggle.com/yash612/covidnet-mini-and-gan-enerated-chest-xray. With this classification model I will save the weights over this training and use this weight in Encoder feature extraction by loading the saved weights to InceptionV3.

Encoder:

The encoder is a single fully connected linear model. The input image is given to InceptionV3 to extract the features. this extracted feature of two images are added and input to the FC layer to get the output vector. This last hidden state of the Encoder is connected to the Decoder.

Decoder:

The Decoder is a have a Bidirectional LSTM layer which does language modelling up to the word level. The first-time step receives the encoded output from the encoder and the <start> vector. This input passed to 2 stage Bidirectional LSTM layer with attention mechanism. The output vector is two vector one is predicted label and other is the previous hidden state of decoder this fed back again to decoder on each time step. Detailed Architecture is mentioned below.



6 XML Parsing Creating Data Points

In this section we will see how the raw xml data is parsed and structured as data points, Then the data points are stored in csv files for future model requirements.

Raw XML Tree View:

eCitation

- meta : @type=rr
- uId : @id=CXR1
- pmcId : @id=1

- docSource : CXR
- IUXRId :
  @id=1
- licenseType : open-access
- licenseURL : http://creativecommons.org/licenses/by-nc-nd/4.0/
- ccLicense : byncnd
- articleURL :
- articleDate : 2013-08-01
- articleType : XR
- publisher : Indiana University
- title : Indiana University Chest X-ray Collection
- note : The data are drawn from multiple hospital systems.
- specialty : pulmonary diseases
- subset : CXR
- MedlineCitation @Owner=Indiana University @Status=supplied by publisher
  - Article @PubModel=Electronic
    - Journal
      - JournalIssue
        - PubDate
          - Year : 2013
          - Month : 08
          - Day : 01
    - ArticleTitle : Indiana University Chest X-ray Collection
    - **Abstract**
      - **AbstractText : None.**
        @Label=COMPARISON
      - **AbstractText : Positive TB test**
        @Label=INDICATION
      - **AbstractText : The cardiac silhouette and mediastinum size are within normal limits. There is no pulmonary edema. There is no focal consolidation. There are no XXXX of a pleural effusion. There is no evidence of pneumothorax.**
        @Label=FINDINGS
      - **AbstractText : Normal chest x-XXXX.**
        @Label=IMPRESSION
    - Affiliation : Indiana University
    - AuthorList @CompleteYN=Y
      - Author @ValidYN=Y
        - LastName : Kohli
        - ForeName : Marc
        - Initials : MD
      - Author @ValidYN=Y
        - LastName : Rosenman
        - ForeName : Marc

                - Initials : M
            - Language : eng
            - PublicationTypeList
                - PublicationType : Radiology Report
            - ArticleDate
                - Year : 2013
                - Month : 08
                - Day : 01
    - EssieArticleTitle : Indiana University Chest X-ray Collection
    - IMedAuthor : Marc David Kohli MD
    - IMedAuthor : Marc Rosenman M
- MeSH
    - major : normal
- **parentImage** **@id=CXR1_1_IM-0001-3001**
    - figureId : F1
    - caption : Xray Chest PA and Lateral
    - panel @type=single
        - url : /hadoop/storage/radiology/extract/CXR1_1_IM-0001-3001.jpg
        - imgModality : 7
        - region @type=panel
            - globalImageFeatures
                - CEDD : f2p0k1352
                - ColorLayout : f1p0k36
                - EdgeHistogram : f0p0k969
                - FCTH : f4p0k2423
                - SemanticContext60 : f3p0k305
- **parentImage** **@id=CXR1_1_IM-0001-4001**
    - figureId : F2
    - caption : Xray Chest PA and Lateral
    - panel @type=single
        - url : /hadoop/storage/radiology/extract/CXR1_1_IM-0001-4001.jpg
        - imgModality : 7
        - region @type=panel
            - globalImageFeatures
                - CEDD : f2p0k1013
                - ColorLayout : f1p0k36
                - EdgeHistogram : f0p0k184
                - FCTH : f4p0k1133
                - SemanticContext60 : f3p0k277

From the xml file we will be extracting the Abstract and ParentImage Nodes. In this we have the Impression and image file name as below.

Impression level:

We will retrieve the Abstract text values

- Abstract
  - **AbstractText : None.**
    @Label=COMPARISON
  - **AbstractText : Positive TB test**
    @Label=INDICATION
  - **AbstractText : The cardiac silhouette and mediastinum size are within normal limits. There is no pulm onary edema. There is no focal consolidation. There are no XXXX of a pleural effusion. There is no eviden ce of pneumothorax.**
    @Label=FINDINGS
  - **AbstractText : Normal chest x-XXXX.**
    @Label=IMPRESSION

Image File name:

Image file name available in the id attribute. We can ignore other details because the data are not relevant for our report. As we can see there are two parentImage nodes we have two image for this report.

- **parentImage** **@id=CXR1_1_IM-0001-3001**
  - figureId : F1
  - caption : Xray Chest PA and Lateral
  - panel @type=single
    - url : /hadoop/storage/radiology/extract/CXR1_1_IM-0001-3001.jpg
    - imgModality : 7
    - region @type=panel
      - globalImageFeatures
        - CEDD : f2p0k1352
        - ColorLayout : f1p0k36
        - EdgeHistogram : f0p0k969
        - FCTH : f4p0k2423
        - SemanticContext60 : f3p0k305
- **parentImage** **@id=CXR1_1_IM-0001-4001**
  - figureId : F2
  - caption : Xray Chest PA and Lateral
  - panel @type=single
    - url : /hadoop/storage/radiology/extract/CXR1_1_IM-0001-4001.jpg
    - imgModality : 7
    - region @type=panel
      - globalImageFeatures
        - CEDD : f2p0k1013

- ColorLayout : f1p0k36
- EdgeHistogram : f0p0k184
- FCTH : f4p0k1133
- SemanticContext60 : f3p0k277

XML parser code to retrieve the details mentioned above.

```
columns = ["image_name", "image_caption", "comparison", "indication", "findings", "impression"]
dataframe = pd.DataFrame(columns = columns)
#list files from Directory
for file in tqdm(os.listdir("ecgen-radiology/")):
    #find files ends with .xml only
    if file.endswith(".xml"):
        #parse the xml file
        tree = ET.parse("ecgen-radiology/"+file)
        #find images in each parentImage tag
        img_list = set()
        cap_list = set()
        for parent in tree.findall("parentImage"):
            img = parent.attrib['id']+".png"
            #for each image iterate and add the corresponding report
                #reading hight and width for image
            h = mpimg.imread("img/"+img).shape[0]
            w = mpimg.imread("img/"+img).shape[1]
            cap_list.add('' if parent.find('caption').text is None else parent.find('caption').text)
            img_list.add(img)
        # finding root element
        tree = ET.parse("ecgen-radiology/"+file)
        comparision = tree.find(".//AbstractText[@Label='COMPARISON']").text
        indication = tree.find(".//AbstractText[@Label='INDICATION']").text
        findings = tree.find(".//AbstractText[@Label='FINDINGS']").text
        impression = tree.find(".//AbstractText[@Label='IMPRESSION']").text
        text_mesh = ""
        i = 1
        for child in tree.find("MeSH"):
            if len(tree.find("MeSH")) == i:
                text_mesh += child.text
            else:
                text_mesh += child.text+" "
            i+=1
        # add reports and image details to dataframe
        dataframe = dataframe.append(pd.Series([',',join(img_list), ','.join(cap_list), comparision, indication, findings, impression],
                                    index = columns), ignore_index = True)
```

After extraction we have 3955 rows, data in dataframe view,

```
dataframe.head()
```

	image_name	image_caption	comparison	indication	findings	impression
0	CXR1_1_IM-0001-3001.png,CXR1_1_IM-0001-4001.png	Xray Chest PA and Lateral	None.	Positive TB test	The cardiac silhouette and mediastinum size ar...	Normal chest x-XXXX.
1	CXR10_IM-0002-1001.png,CXR10_IM-0002-2001.png	PA and lateral chest x-XXXX XXXX.	Chest radiographs XXXX.	XXXX-year-old male, chest pain.	The cardiomediastinal silhouette is within nor...	No acute cardiopulmonary process.
2	CXR100_IM-0002-1001.png,CXR100_IM-0002-2001.png	CHEST 2V FRONTAL/LATERAL XXXX, XXXX XXXX PM	None.	None	Both lungs are clear and expanded. Heart and m...	No active disease.
3	CXR1000_IM-0003-2001.png,CXR1000_IM-0003-1001....	PA and lateral chest x-XXXX XXXX.	XXXX PA and lateral chest radiographs	XXXX-year-old male, XXXX.	There is XXXX increased opacity within the rig...	1. Increased opacity in the right upper lobe w...
4	CXR1001_IM-0004-1002.png,CXR1001_IM-0004-1001.png	CHEST 2V FRONTAL/LATERAL XXXX, XXXX XXXX PM	None	dyspnea, subjective fevers, arthritis, immigra...	Interstitial markings are diffusely prominent ...	Diffuse fibrosis. No visible focal acute disease.

7 Data Preprocessing

In this phase the text data are preprocessed to remove unwanted tags, texts, punctuation and numbers. We will also check for the empty cell or NaN values.

- If there any empty cells in image name column we will drop those cells.
- If there any empty or NaN value in text data we will replace it with "No <Column Name>" (ex: No Impression)

- Each text column word counts are calculated and added to the dataframe column.

```python
#remove HTML from the Text column and save in the Text column only
def preprocess_text(data, isCaption):
    # Combining all the above stundents
    preprocessed_reviews_eng = []

    # tqdm is for printing the status bar
    for sentance in tqdm(data.values):
        sentance = sentance.lower()
        sentance = re.sub(r"http\S+", "", sentance)
        sentance = BeautifulSoup(sentance, 'lxml').get_text()
        sentance = re.sub(r",", " ", sentance)
        sentance = re.sub(r"xxxx", "", sentance)
        sentance = re.sub(r"xxxxx", "", sentance)
        sentance = re.sub(r'[0-9]',"",sentance)
        sentance = re.sub(r"[-()\"#/@;:<>{}`+=~|.!?$%^&*'/+\[\]_]+", "", sentance)
        sentance = re.sub(r"yearold", "", sentance)
        sentance = re.sub('\s+',' ',sentance)
        #if not isCaption:
            #sentance = '<start> ' + sentance + ' <end>'
        preprocessed_reviews_eng.append(sentance.strip())
    return preprocessed_reviews_eng
```

After the data preprocessing missing value handling below is the dataframe view and we have total of 3851 rows present in the final data points.

	image_name	image_caption	comparison	indication	findings	impression	findings_count	impression_count	image_count
0	CXR1_1_IM-0001-3001.png,CXR1_1_IM-0001-4001.png	xray chest pa and lateral	none	positive tb test	the cardiac silhouette and mediastinum size ar...	normal chest x	33	3	2
1	CXR10_IM-0002-1001.png,CXR10_IM-0002-2001.png	pa and lateral chest x	chest radiographs	male chest pain	the cardiomediastinal silhouette is within nor...	no acute cardiopulmonary process	38	4	2
2	CXR100_IM-0002-1001.png,CXR100_IM-0002-2001.png	chest v frontallateral pm	none	no indication	both lungs are clear and expanded heart and me...	no active disease	10	3	2
3	CXR1000_IM-0003-2001.png,CXR1000_IM-0003-1001....	pa and lateral chest x	pa and lateral chest radiographs	male	there is increased opacity within the right up...	increased opacity in the right upper lobe with...	52	36	3
4	CXR1001_IM-0004-1002.png,CXR1001_IM-0004-1001.png	chest v frontallateral pm	none	dyspnea subjective fevers arthritis immigrant ...	interstitial markings are diffusely prominent ...	diffuse fibrosis no visible focal acute disease	14	7	2

```python
print("Shape of the dataframe ", data.shape)
```

```
Shape of the dataframe  (3851, 9)
```

- Total number of unique Images 3851
- Total number of unique Caption 402
- Total number of unique Comparison 281
- Total number of unique Indication 2098
- Total number of unique Findings 2545
- Total number of unique Impression 1692

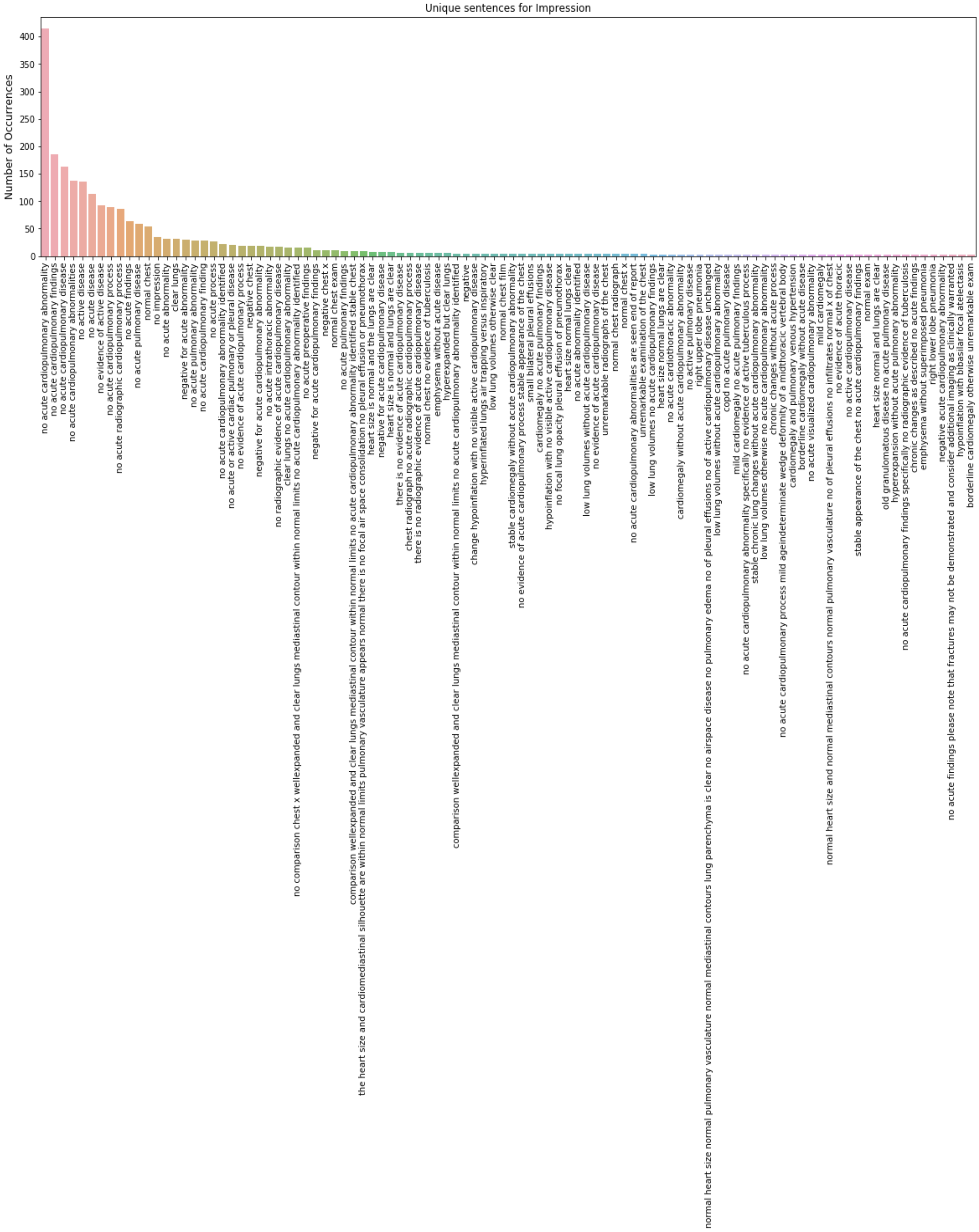
8 Exploratory Data Analysis

In this section we will see different approaches to analyze the data set by summarizing and visualizing their main characteristics.

8.1 EDA on Text data

In the text analysis we will be taking the impression column target variable. With below visualization we could see the top 100 most occurring sentences.

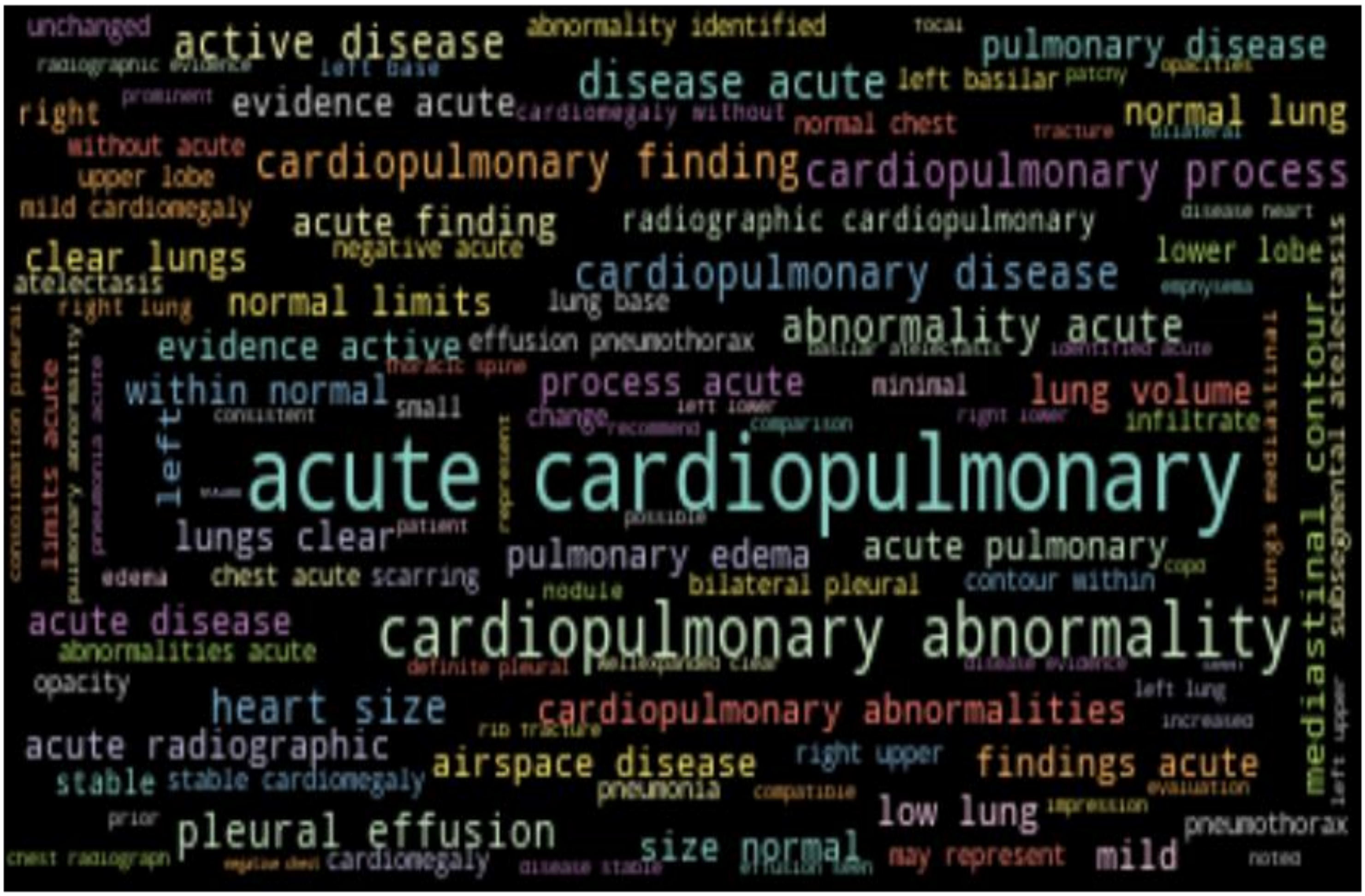
Sentence occurrences for Impression


Unique sentences for Impression
Number of Occurrences
0
50
100
150
200
250
300
350
400
no acute cardiopulmonary abnormality
no acute cardiopulmonary findings
no acute cardiopulmonary disease
no acute cardiopulmonary abnormalities
no active disease
no acute disease
no evidence of active disease
no acute cardiopulmonary process
no acute radiographic cardiopulmonary process
no acute findings
no acute pulmonary disease
normal chest
no impression
no acute abnormality
clear lungs
negative for acute abnormality
no acute pulmonary abnormality
no acute cardiopulmonary finding
no acute process
no acute cardiopulmonary abnormality identified
no acute or active cardiac pulmonary or pleural disease
no evidence of acute cardiopulmonary process
negative chest
negative for acute cardiopulmonary abnormality
no acute intrathoracic abnormality
no radiographic evidence of acute cardiopulmonary disease
clear lungs no acute cardiopulmonary abnormality
no comparison chest x wellexpanded and clear lungs mediastinal contour within normal limits no acute cardiopulmonary abnormality identified
no acute preoperative findings
negative for acute cardiopulmonary findings
negative chest x
normal chest exam
no acute pulmonary findings
comparison wellexpanded and clear lungs mediastinal contour within normal limits no acute cardiopulmonary abnormality identified stable chest
the heart size and cardiomediastinal silhouette are within normal limits pulmonary vasculature appears normal there is no focal air space consolidation no pleural effusion or pneumothorax
heart size is normal and the lungs are clear
negative for acute cardiopulmonary disease
heart size is normal and lungs are clear
there is no evidence of acute cardiopulmonary disease
chest radiograph no acute radiographic cardiopulmonary process
there is no radiographic evidence of acute cardiopulmonary disease
normal chest no evidence of tuberculosis
emphysema without acute disease
hyperexpanded but clear lungs
comparison wellexpanded and clear lungs mediastinal contour within normal limits no acute cardiopulmonary abnormality identified
negative
change hypoinflation with no visible active cardiopulmonary disease
hyperinflated lungs air trapping versus inspiratory
low lung volumes otherwise clear
normal chest film
stable cardiomegaly without acute cardiopulmonary abnormality
no evidence of acute cardiopulmonary process stable appearance of the chest
small bilateral pleural effusions
cardiomegaly no acute pulmonary findings
hypoinflation with no visible active cardiopulmonary disease
no focal lung opacity pleural effusion of pneumothorax
heart size normal lungs clear
no acute abnormality identified
low lung volumes without acute cardiopulmonary disease
no evidence of acute cardiopulmonary disease
unremarkable radiographs of the chest
normal chest radiograph
normal chest x
no acute cardiopulmonary abnormalities are seen end of report
unremarkable examination of the chest
low lung volumes no acute cardiopulmonary findings
heart size normal lungs are clear
no acute cardiothoracic abnormality
cardiomegaly without acute cardiopulmonary abnormality
no active pulmonary disease
right upper lobe pneumonia
normal heart size normal pulmonary vasculature normal mediastinal contours lung parenchyma is clear no airspace disease no pulmonary edema no of pleural effusions no of active cardiopulmonary disease unchanged
low lung volumes without acute cardiopulmonary abnormality
copd no acute pulmonary disease
mild cardiomegaly no acute pulmonary findings
no acute cardiopulmonary abnormality specifically no evidence of active tuberculous process
stable chronic lung changes without acute cardiopulmonary abnormality
low lung volumes otherwise no acute cardiopulmonary abnormality
chronic changes without acute process
no acute cardiopulmonary process mild ageindeterminate wedge deformity of a midthoracic vertebral body
cardiomegaly and pulmonary venous hypertension
borderline cardiomegaly without acute disease
no acute visualized cardiopulmonary abnormality
mild cardiomegaly
normal heart size and normal mediastinal contours normal pulmonary vasculature no of pleural effusions no infiltrates normal x of chest
no evidence of acute thoracic
no active cardiopulmonary disease
stable appearance of the chest no acute cardiopulmonary findings
normal exam
heart size normal and lungs are clear
old granulomatous disease no acute pulmonary disease
hyperexpansion without acute pulmonary abnormality
no acute cardiopulmonary findings specifically no radiographic evidence of tuberculosis
chronic changes as described no acute findings
emphysema without superimposed pneumonia
right lower lobe pneumonia
negative acute cardiopulmonary abnormality
no acute findings please note that fractures may not be demonstrated and consider additional imaging as clinically warranted
hypoinflation with bibasilar focal atelectasis
borderline cardiomegaly otherwise unremarkable exam

- From above visualization we can see that "No acute cardiopulmonary abnormality" occurred almost 600 times.
- Mostly longer sentences are occurred less than are equal to 10 times

Word occurrences for Impression

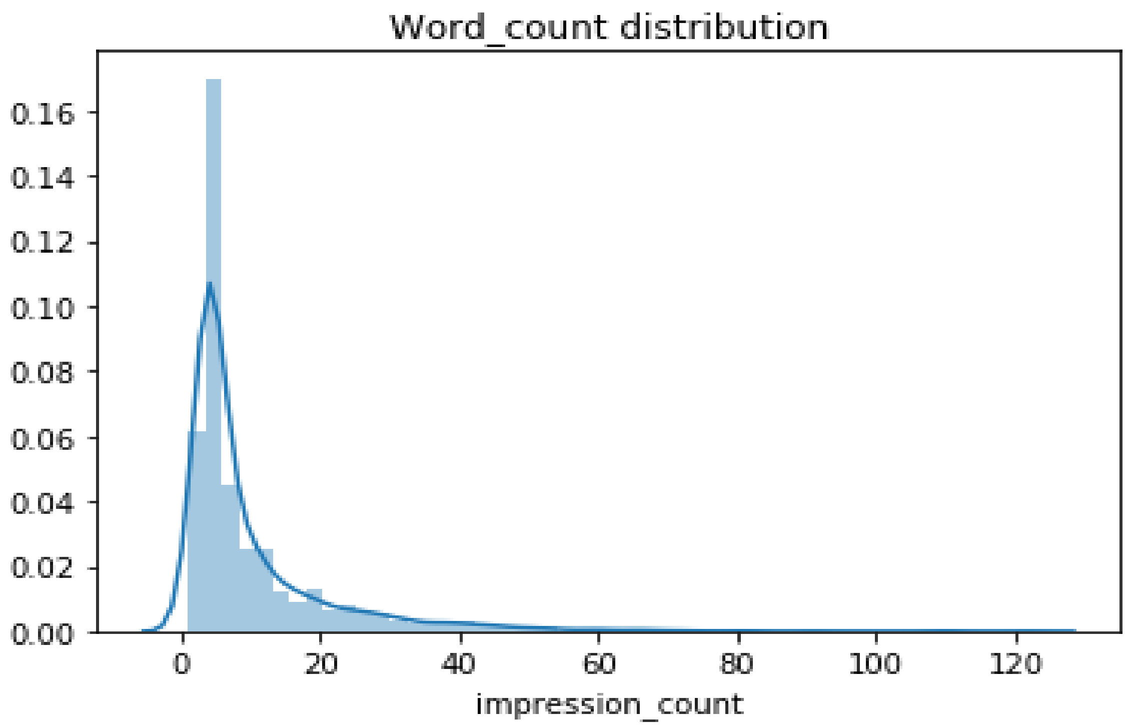
We will see the word wise occurrence using the word cloud for impression column



- Above word cloud are generated on the top 1000 max occurrence words.
- Acute, cardiopulmonary, abnormality, disease, pleural, effusion, active these are the highlighted words from above visualization.

Word count distribution

Let’s see the word count distribution on the impression column as we have already calculated the word count in impression_count column we see the distribution like below.

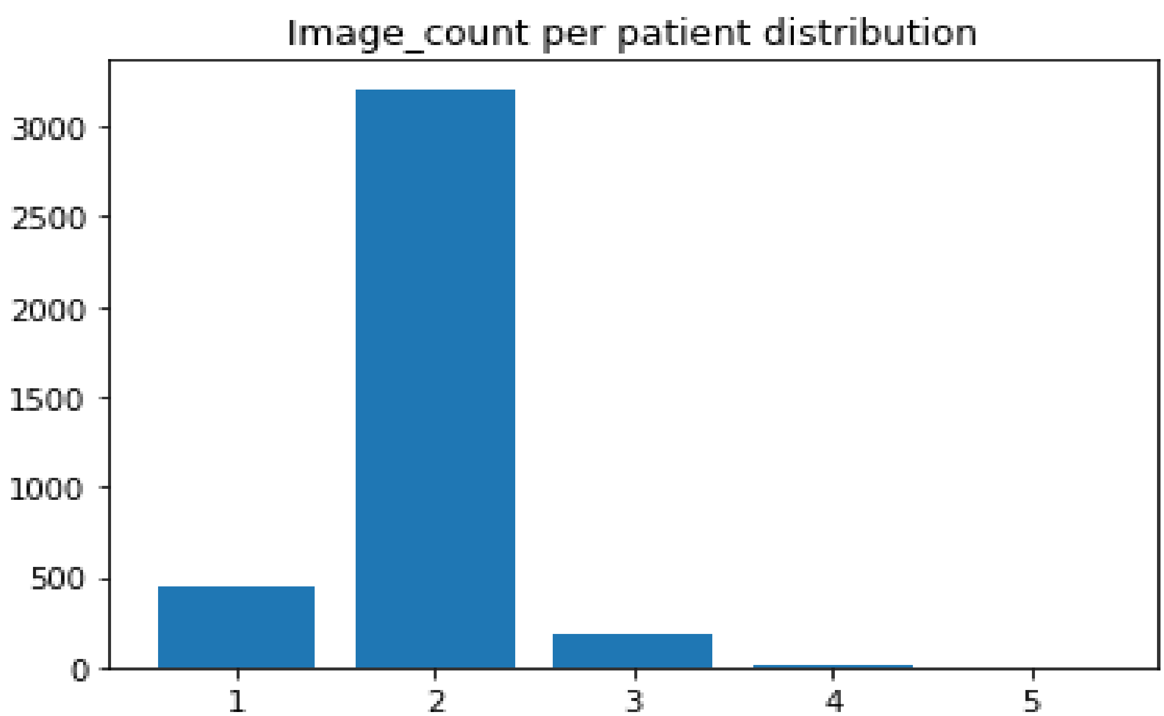


Minimum word count is 1 - Maximum word count is 122 - Median word count is 5.0

- We can see the maximum and minimum word count from this distribution.
- Word count that maximum occurrence is mostly 5
- Most often word count is between 5 to 10 only.

8.2 EDA on Image data

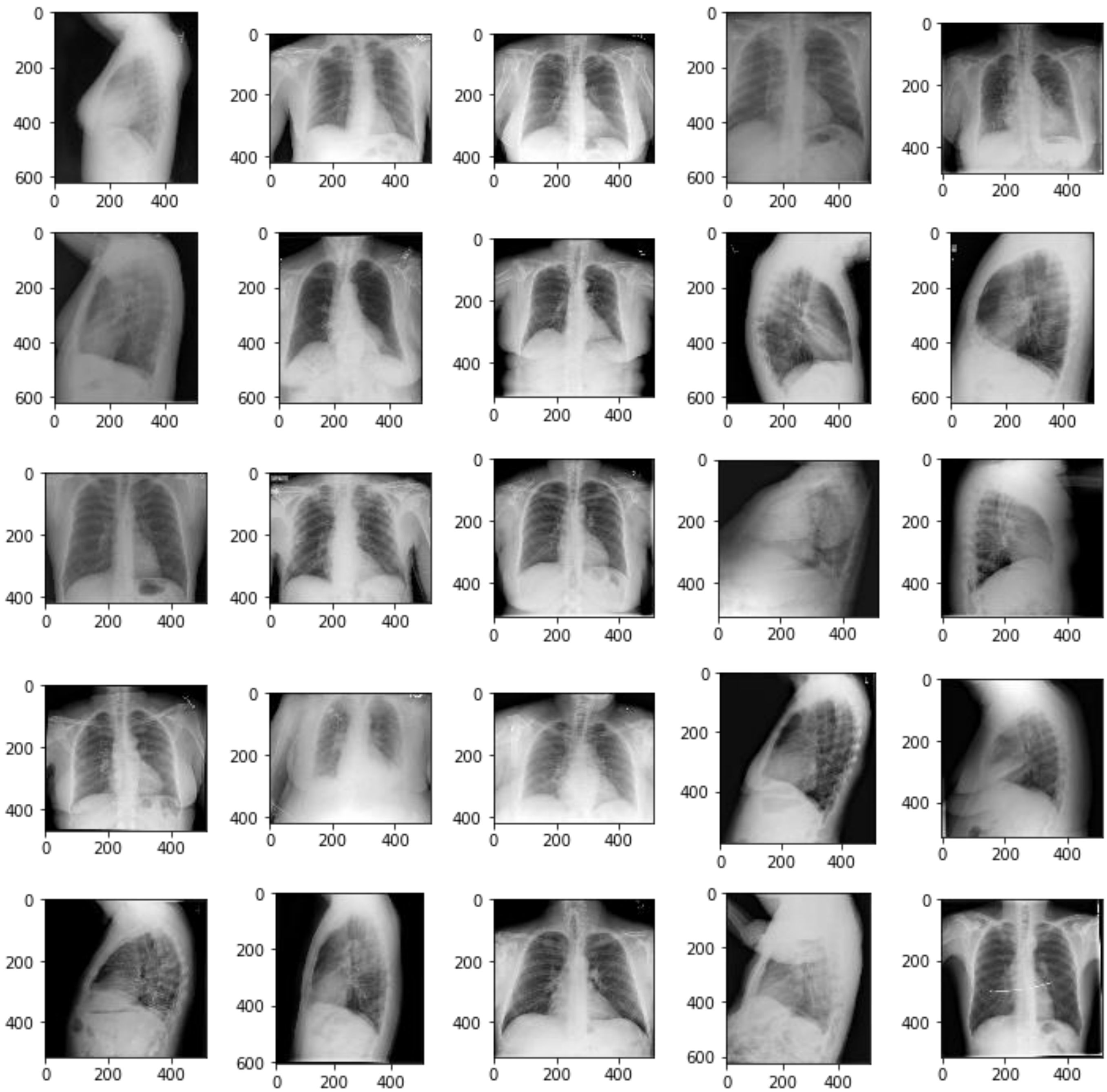
Lets analyze the total image present per data point or report.



Minimum Image count is 1 - Maximum Image count is 5 - Median Image count is 2.0

- Most frequent images count per record is 2.
- Second frequent is single image.
- We do have 5 images per records too.

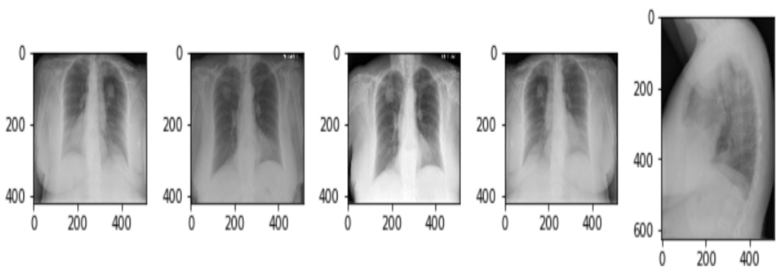
Displaying random 25 patient X-Ray



As we have seen the images are in both Frontal and Lateral view. And each patient have one or more than 2 images associated with it. Let see some random data points with its images.

Sample data point


```
Total Images present for this patient 3
====================================================================================================
Findings: Total No of words 41
trachea is midline the cardiomediastinal silhouette is normal the lungs are clear without evidence of acute infiltrate or effusion there is no pneumothorax the visualized bony structures show
no acute abnormalities lateral view reveals mild degenerative changes of the thoracic spine
====================================================================================================
Impression: Total No of words 4
no acute cardiopulmonary abnormalities
====================================================================================================
```




```
Total Images present for this patient 5
====================================================================================================
Findings: Total No of words 75
in the interval a cm uncalcified mass has developed in the posterior segment of the right upper lobe in addition on the pa view an mm opacity is adjacent to the left of the heart this opacity
cannot be well identified on the lateral view it may be artifactual but another mass on the left cannot be excluded mediastinum is normal with no evidence for adenopathy heart size normal not
e of an unchanged hiatal hernia
====================================================================================================
Impression: Total No of words 19
right upper lobe mass suspicious for neoplasm ct of chest abdomen and head would be helpful for further evaluation
====================================================================================================
```

8.3 EDA Findings

- All the raw texts from xml files are parsed and created the dataset.
- Each patient have multiple x-rays associated with them.
- Major finding is that how the images are in sequence or number of images associated with each record.
- We have mostly of 2 images per record frontal and lateral. and also, we have 1, 3, 4, 5 images associated with each record.

- There is no missing files. We have total of 3955 records and 3 additional features (Comparison, Indication and Findings) which we will not use in this model and 1 Impression target variable.
- Most occurring words:
    - Impression: acute cardiopulmonary
- Images are in different shapes.
- All the X-Ray images are human upper body particularly about Chest part.
- In text features there are some unknown values like XXXX XXXXX these are replaced with empty string.

8.4 Data Conflicts

There are only two image types Frontal and Lateral, but we have 1, 3, 4, and 5 images associated for each datapoints. we have met a conflict here that how we provide the data points to the model we build. Because of this conflict we need to come up with an idea to handle how we input the data to the model. Before building our model, we see some data point structuring methods that could help use handle this case.

9 Data point construction

As we have more than 2 image or some case less than 2 images associated with each data point. If we have no images, we dropped those data point.

Lets handle the data point which are having 1,3,4,5 images. Below is the data point counts with number of image sets.

Data point having 2 images is 3208

Data point having 1 images is 446

Data point having 3 images is 181

Data point having 4 images is 15

Data point having 1 images is 1

Total data point is 3851 data points

Approach,

Limiting the data point to 2 images per data point, if we have 5 images, its 4+1 (all image + last image) so make it as 4 data points as below.

Here last image should be Lateral if we have frontal as remaining images.

if i have 5 images then, here 5th image is Lateral other or frontal,

1st image + 5th image => Frontal + Lateral

2nd image + 5th image => Frontal + Lateral

3rd image + 5th image => Frontal + Lateral

4th image + 5th image => Frontal + Lateral

Increased to 4 data point from this single data point likewise, for other data point,

if i have 4 images then,

1st (Frontal) + 4th (Lateral)

2nd (Frontal) + 4th (Lateral)

3rd (Frontal) + 4th (Lateral)

Increase to 3 data point from 1 data point

if i have 3 images then,

1st (Frontal) + 3rd (Lateral)

2nd (Frontal) + 3rd (Lateral)

Increased to 2 data point from this single data point

If we have only one image then,

1st images either (Frontal or Lateral) + Duplicate 1st image

Same data point count. We need to make sure this duplicating data point should be equally split among the train test and validation sets. If we don't have Lateral images, then keep the frontal as last image data points.

So with this data constructing method we could also increase the data point and come up with fine input data points. Code for the above explained data structuring.

```
columns = ["image_1", "image_2", "impression"]
df = pd.DataFrame(columns = columns)
columns = ["image_1", "image_2", "impression"]
df_dup = pd.DataFrame(columns = columns)
no_lateral = 0
for item in tqdm(data.iterrows()):
    l = item[1]['image_name'].split(',')
    if len(l) > 2:
        li, last_img = find_Fr_la(l)
        if last_img == "":
            no_lateral +=1
            li, last_img = li[:-1], li[-1]
        for i in li:
            image_1 = i
            image_2 = last_img
            df = df.append(pd.Series([image_1, image_2, item[1]['impression']], index = columns), ignore_index = True)
    elif len(l) == 2:
        image_1 = l[0]
        image_2 = l[1]
        df = df.append(pd.Series([image_1, image_2, item[1]['impression']], index = columns), ignore_index = True)
    elif len(l) == 1:
        #creating duplicate dataframe separately to keep it in all dataset train test validate
        df_dup = df_dup.append(pd.Series([l[0], l[0], item[1]['impression']], index = columns), ignore_index = True)
print("Total Report without Lateral images {}".format(no_lateral))
```

```
3851it [00:13, 283.67it/s]
Total Report without Lateral images 1
```

After constructing the data point we will add the <start> and <end> token to text data.

Final datapoints,

	image_1	image_2	impression
0	CXR1_1_IM-0001-3001.png	CXR1_1_IM-0001-4001.png	<start> normal chest x <end>
1	CXR10_IM-0002-1001.png	CXR10_IM-0002-2001.png	<start> no acute cardiopulmonary process <end>
2	CXR100_IM-0002-1001.png	CXR100_IM-0002-2001.png	<start> no active disease <end>
3	CXR1000_IM-0003-1001.png	CXR1000_IM-0003-2001.png	<start> increased opacity in the right upper l...
4	CXR1000_IM-0003-3001.png	CXR1000_IM-0003-2001.png	<start> increased opacity in the right upper l...

10 Train Test and Validation split

We have a separate data one is without duplicate data points other is with duplicate data points. We need to split the data points as the duplicate data points are equally available in all three splits.

```
i_train, input_test, o_train, output_test = train_test_split(df[['image_1','image_2']].values, df['impression'].values, test_size=0.1, random_state=15)
input_train, input_val, output_train, output_val = train_test_split(i_train, o_train, test_size=0.2, random_state=15)
input_train.shape, output_train.shape, input_val.shape, output_val.shape, input_test.shape, output_test.shape
```

```
((2542, 2), (2542,), (636, 2), (636,), (354, 2), (354,))
```

- Train test and validation split for duplicate dataframe

```
i_train_dup, input_test_dup, o_train_dup, output_test_dup = train_test_split(df_dup[['image_1','image_2']].values, df_dup['impression'].values, test_size=0.1, random_state=15)
input_train_dup, input_val_dup, output_train_dup, output_val_dup = train_test_split(i_train_dup, o_train_dup, test_size=0.2, random_state=15)
input_train_dup.shape, output_train_dup.shape, input_val_dup.shape, output_val_dup.shape, input_test_dup.shape, output_test_dup.shape
```

```
((320, 2), (320,), (81, 2), (81,), (45, 2), (45,))
```

After taking the two different data set we need to concatenate the dataset equally.

```
in_train = np.append(input_train, input_train_dup, axis=0)
out_train = np.append(output_train, output_train_dup, axis=0)
in_val = np.append(input_val, input_val_dup, axis=0)
out_val = np.append(output_val, output_val_dup, axis=0)
in_test = np.append(input_test, input_test_dup, axis=0)
out_test = np.append(output_test, output_test_dup, axis=0)
print("===== Final data point shape =====")
in_train.shape, out_train.shape, in_val.shape, out_val.shape, in_test.shape, out_test.shape
```

```
===== Final data point shape =====
((2862, 2), (2862,), (717, 2), (717,), (399, 2), (399,))
```

We get the final data point shape as above.

11 Tokenization and Dataset preparation

11.1 Tokenization

We cannot feed raw text to our deep learning model. Text data need to be encoded as numbers and then used in both machine learning and deep learning models. The Keras deep learning library provides some basic tools to perform this operation.

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_len_output = 60

tokenizer = Tokenizer(oov_token="<unk>", filters='!"#$%&()*+.,-/:;=?@[\]^_`{|}~ ')
tokenizer.fit_on_texts(out_train)
text_train = tokenizer.texts_to_sequences(out_train)
text_test = tokenizer.texts_to_sequences(out_test)
text_val = tokenizer.texts_to_sequences(out_val)
dictionary = tokenizer.word_index

word2idx = {}
idx2word = {}
for k, v in dictionary.items():
    word2idx[k] = v
    idx2word[v] = k
```

```
vocab_size = len(word2idx)+1
vocab_size
```

```
1339
```

Total vocabulary present is 1339 and maximum length of the output sentence is taken as 60.

11.2 Dataset Preparation

For the dataset preparation we will be using the transfer learning method for image to feature vector conversion and text data tokenization.

Please refer this blog on why I have chosen the inception model over others
https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

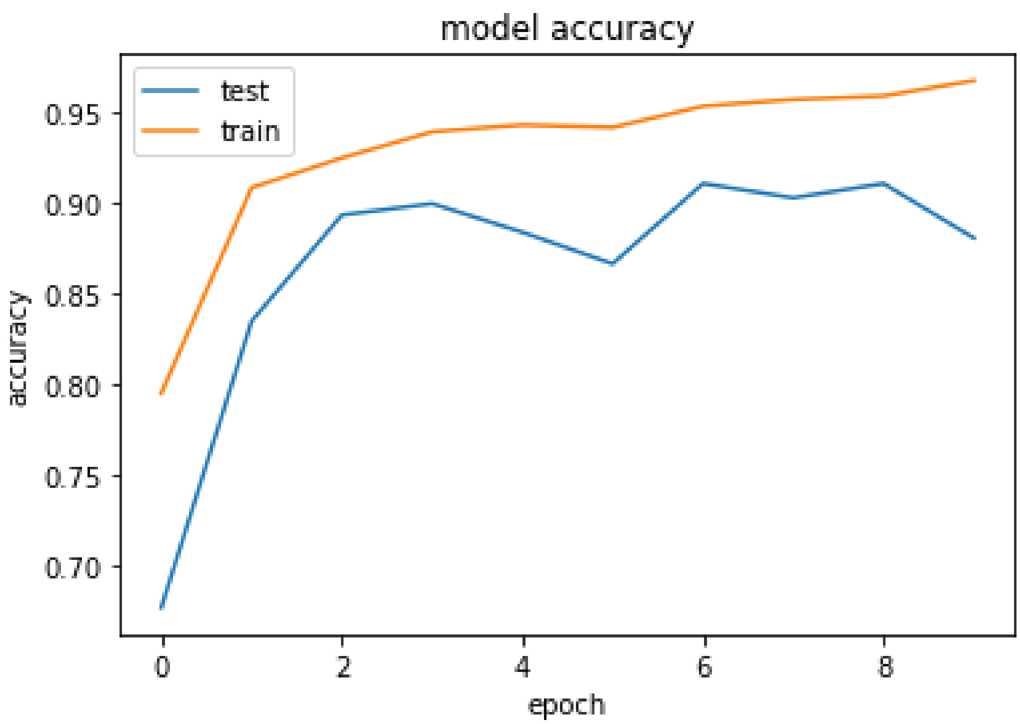
I will be using the InceptionV3 model trained on ImageNet dataset. Initially I will be doing a xray classification task using below mentioned dataset. https://www.kaggle.com/yash612/covidnet-mini-and-gan-enerated-chest-xray. It's a three-class classification task where we need to classify the whether the xray of the patient is belongs to one of these 3 class Corona or Normal or Pneumonia.

Once the classification is done, I will save the weights of the trained model and use this model with removed top layer of shape(1, 2048) as feature vector for our model and prepare the dataset.

Below is the model architecture for this classification task.

```
InceptionV3_model = InceptionV3(include_top=False, weights='imagenet', pooling='avg', input_shape=(299,299,3))
InceptionV3_model.input, InceptionV3_model.output
x=tf.keras.layers.Dense(256, activation='relu')(InceptionV3_model.output)
x=tf.keras.layers.Dense(64, activation='relu')(x)
output_layer = tf.keras.layers.Dense(3, activation='softmax')(x)
model = tf.keras.Model(InceptionV3_model.input, output_layer)
model.summary()
```

Accuracy plot on the classification task.




```
inception_model = InceptionV3(include_top=False, weights='imagenet', pooling='avg', input_shape=(299,299,3))
for i, layer in enumerate(inception_model.layers):
    layer.set_weights(model.layers[i].get_weights())
```

```
inception_model.save_weights("trained_weights-07-0.9102.hdf5")
```

Model weights are saved for future use as hdf5 file.

I have trained this model using ImageNet weights and without ImageNet weights with image net weights performed well in this classification.

With the trained weights I will use like below for feature extraction for our image data.

```
#loads the pretrained weights
image_features_model = InceptionV3(include_top=False, weights='imagenet', pooling='avg', input_shape=(299,299,3))
image_features_model.load_weights("trained_weights-07-0.9102.hdf5")
```

I will create a image tensor for all available images using the inception feature vectorization like below.

```
img_tensor = []
#creates image feature vector
for img in tqdm(image_name):
    img = tf.io.read_file(image_path + str(img))
    img = tf.image.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, (299, 299))
    img = preprocess_input(img)
    img_features = image_features_model(tf.constant(img)[None, :])
    img_tensor.append(img_features)
```

These image tensors are used in TensorFlow dataset preparation basically I am doing a caching mechanism here for future use.

Create TensorFlow using tf.data

Refer the link for further reading on tf.data: https://www.tensorflow.org/guide/data

Tutorials to read: https://adventuresinmachinelearning.com/tensorflow-dataset-tutorial/

Now that we have our image tensor and text vectors we can build the tf.data dataset

```
dataset_train = tf.data.Dataset.from_tensor_slices((in_train, text_output_train))

# Use map to load the numpy files in parallel
dataset_train = dataset_train.map(lambda item1, item2: tf.numpy_function(
          multi_image, [item1, item2], [tf.float32, tf.int32]),
          num_parallel_calls=tf.data.experimental.AUTOTUNE)

dataset_val = tf.data.Dataset.from_tensor_slices((in_val, text_output_val))

# Use map to load the numpy files in parallel
dataset_val = dataset_val.map(lambda item1, item2: tf.numpy_function(
          multi_image, [item1, item2], [tf.float32, tf.int32]),
          num_parallel_calls=tf.data.experimental.AUTOTUNE)


BATCH_SIZE = 32
BUFFER_SIZE = 1000
embedding_dim = 256
units = 128

# Shuffle and batch Train
dataset_train = dataset_train.shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
dataset_train = dataset_train.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
# Shuffle and batch Validation
dataset_val = dataset_val.shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
dataset_val = dataset_val.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
```

Multi_image() function converts the two-input tensor of shape (1,2048) & (1,2048) to (2,1,2048). Batch_size, embedding dimension, and units size are mentioned these are the hyperparameters that we can tune according to our model.

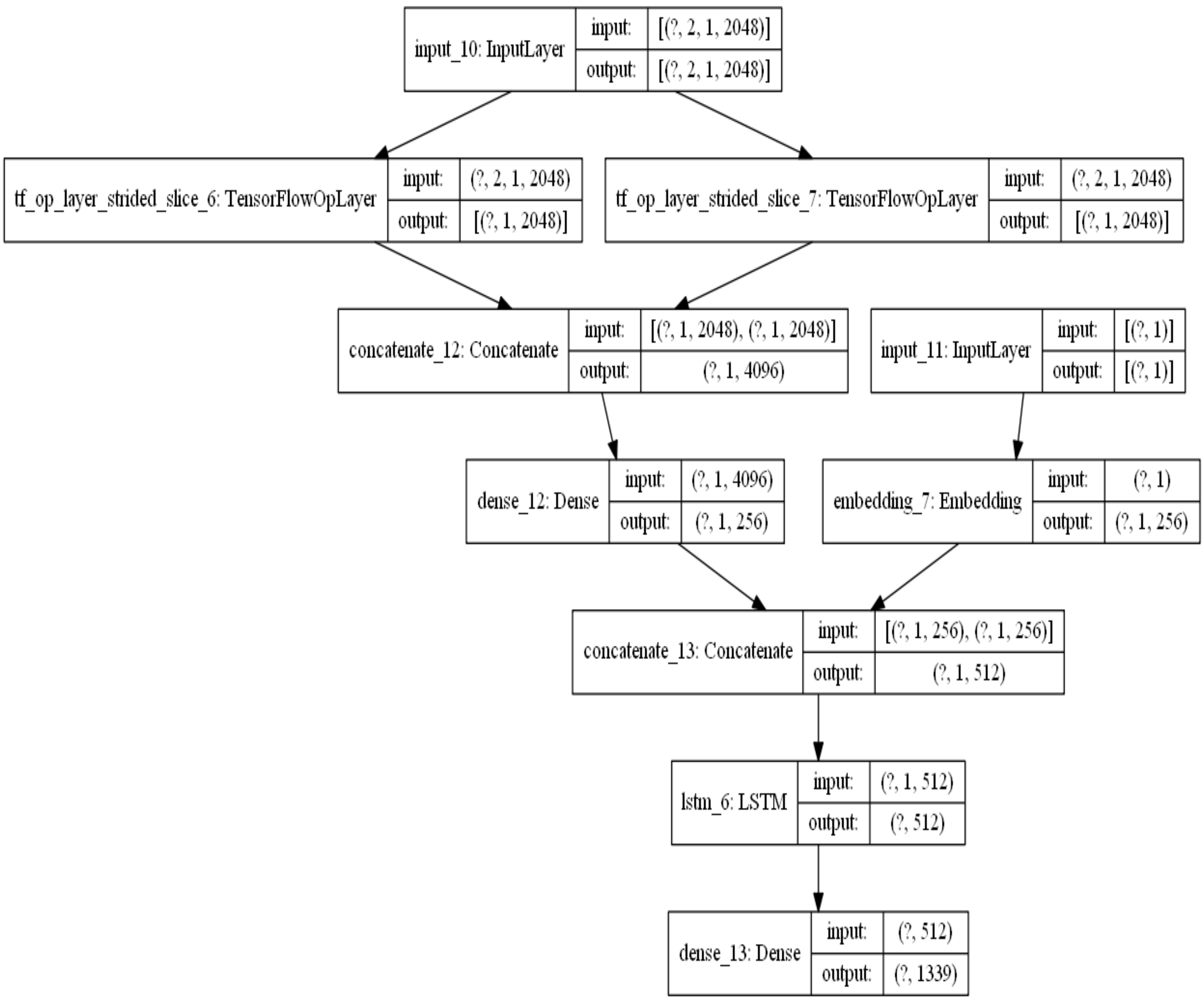
So we have done the feature extraction and tokenization for our model to work, And we have the tf.data dataset now lets build the required model.

12 Basic Model

12.1 Model Architecture

As I have already explained about the subclass model. I will directly jump into the model architecture.

I have built the functional Api model for checking the model architecture



12.1.1 Encoder architecture:

Have single fully connected layer linear output. Before we pass to the FC layer, we add the two image tensor and pass to FC layer. This layer outputs shape of (batch_size, 1, embedding_dimension)

```python
class Encoder(tf.keras.Model):
    def __init__(self, embedding_dim):
        super(Encoder, self).__init__()
        self.fc = tf.keras.layers.Dense(embedding_dim, kernel_initializer=tf.keras.initializers.glorot_uniform(seed=45),
                  name="encoder_output_layer")

    def call(self, x):
        x = tf.reshape(x, [x.shape[0], x.shape[1], x.shape[3]])
        encoder_concat = tf.keras.layers.concatenate([x[:,0], x[:,1]])
        x = self.fc(encoder_concat)
        x = tf.nn.relu(x)
        return x
```

12.1.2 Decoder Architecture:

In this part we have an embedding layer LSTM layer and dense layer which outputs shape (batch_size, vocab_size)

LSTM layer is Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies.

To know more about LSTM refer this link: [Understanding LSTM Networks]

```python
class Decoder(tf.keras.Model):
    def __init__(self, embedding_dim, units, vocab_size):
        super(Decoder, self).__init__()
        self.units = units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.lstm = tf.keras.layers.LSTM(self.units,
                                         return_sequences=True,
                                         return_state=True,
                                         recurrent_initializer=tf.keras.initializers.glorot_uniform(seed=45))
        self.dense = tf.keras.layers.Dense(vocab_size, kernel_initializer=tf.keras.initializers.glorot_uniform(seed=45))

    def call(self, x, features):
        #input x = input word teach forcing
        #input features = encoder image features
        x = self.embedding(x)
        x = tf.concat([x, tf.expand_dims(features,1)], axis=-1)
        output, state, _ = self.lstm(x)
        x = self.dense(output)
        return x
```

12.2 Model metric and optimizer Initialization

```
optimizer = tf.keras.optimizers.Adam()
loss_obj = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True, reduction='none')

acc_obj = tf.keras.metrics.SparseCategoricalAccuracy()

def loss_func(real, pred):
    loss_f = loss_obj(real, pred)
    return tf.reduce_mean(loss_f)


def acc_func(real, pred):
    acc_f = acc_obj(real, pred)
    return tf.reduce_mean(acc_f)
```

12.3 Model Training

For the training phase we use the Teacher forcing. Teacher forcing is a strategy for training recurrent neural networks that uses model output from a prior time step as an input.

In the Training, a “start-of-sequence” token can be used to start the process and the generated word in the output sequence is used as input on the subsequent time step, perhaps along with other input like an image or a source text.

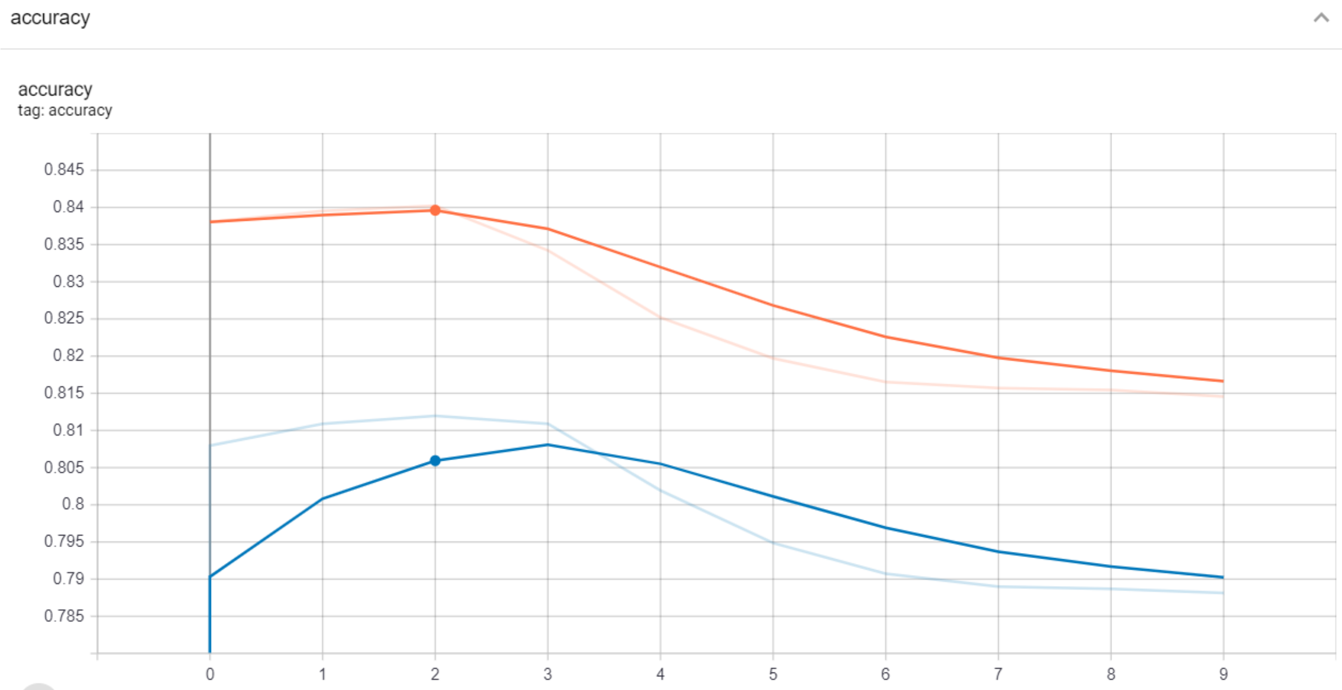
This same recursive output-as-input process is used till the model converge to better result. Below I have mentioned the source.

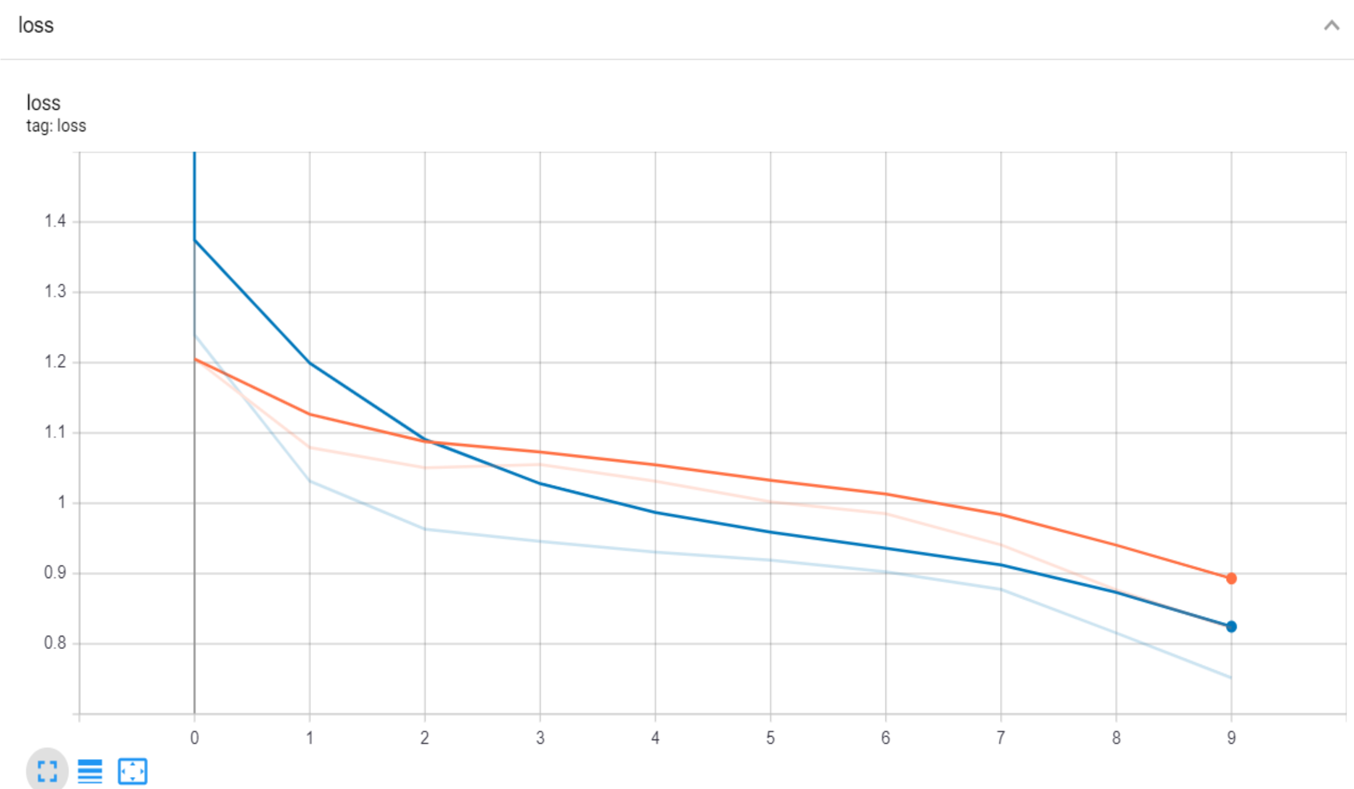
Further readings about teacher forcing: link to [Teacher forcing]

```
@tf.function
def train_step(tensor, target):
    loss = 0
    accuracy = 0
    dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * target.shape[0], 1)
    with tf.GradientTape() as tape:
        features = encoder(tensor)
        for i in range(1, target.shape[1]):
            # passing the features through the decoder
            predictions = decoder(dec_input, features)
            loss += loss_func(target[:, i], predictions)
            accuracy += acc_func(target[:, i], predictions)

            # using teacher forcing
            dec_input = tf.expand_dims(target[:, i],1)
            #print("decoder input teacher", dec_input.shape)
    total_loss = (loss / int(target.shape[1]))
    total_acc = (accuracy / 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, total_acc
```

12.4 Model Performance visualized in TensorBoard

We have logged the loss and accuracy using tf.summary





12.5 Model Evaluation

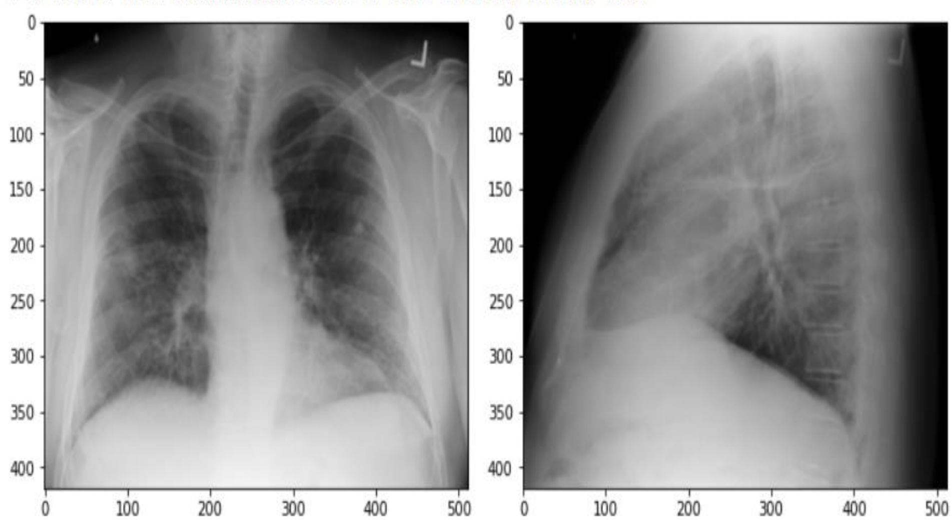
In the evaluation or testing stage I have used the argmax search based teacher forcing to find the output sentence. In time step t we generated a word using <start> token and the predicted word again fed back to the next step and it become the input of the decoder in time t+1. Code for argmax search is mentioned below.

```
def evaluate(img_name):
    img_tensor = tf.convert_to_tensor([get_img_tensor("img/",img_name[0], image_features_model),
                                      get_img_tensor("img/",img_name[1], image_features_model)])
    img_features = tf.constant(img_tensor)[None, :]
    features_val = encoder(img_features)
    dec_input = tf.expand_dims([tokenizer.word_index['<start>']], 1)
    result = []
    text = ""
    for i in range(max_len_output):

        predictions = decoder(dec_input, features_val)
        predictions = tf.reshape(predictions, [predictions.shape[0],predictions.shape[2]])
        predicted_id = tf.argmax(predictions, axis=1)[0].numpy()
        result.append(tokenizer.index_word[predicted_id])
        text += " " + tokenizer.index_word[predicted_id]
        if tokenizer.index_word[predicted_id] == '<end>':
            return result, text

        dec_input = tf.expand_dims([predicted_id], 1)
    return result, text
```

Sample outputs are shown below

Lets try a longer sentence word first

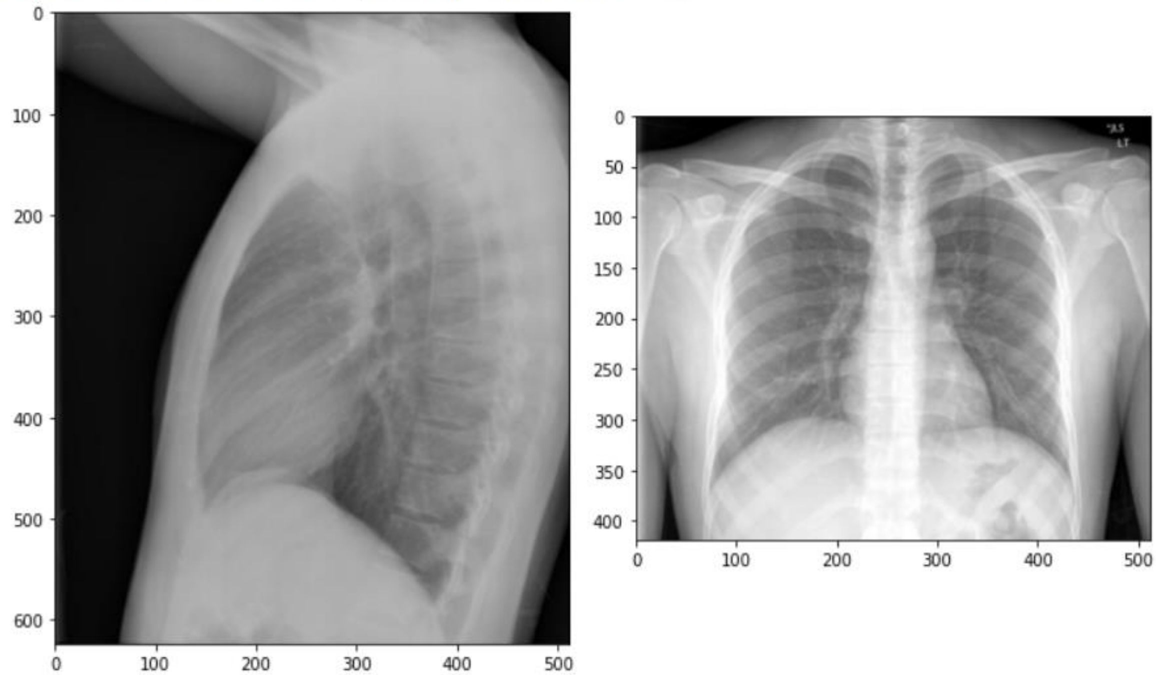
Actual <start> round density within the anterior segment of the right upper lobe this may represent pulmonary nodule the primordial was employed to notify the referring physicians of this critical finding <end>



Predicted: negative loculation heart size persistent infiltrate <end>

Prediction is not perfect in the longer sentence lets see the shorter sentence.

Actual <start> no acute cardiopulmonary abnormalities <end>



Predicted: no evidence of be focus base opacity <end>

Even in the short sentence model not performing well.

12.6 Basic Model Conclusion

- This model is built on a simple encoder decoder with LSTM.
- getting not perfect or not worst predictions
- validation accuracy is not improving much but loss is converging
- we could even fine tune this model for perform well.

We will see a better performing and modified architecture having bidirectional LSTM layer with Additive Attention mechanism.

13 Main Model

13.1 Model Architecture

The Model Architecture is reimplemented using one of the research paper I came across [Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification]

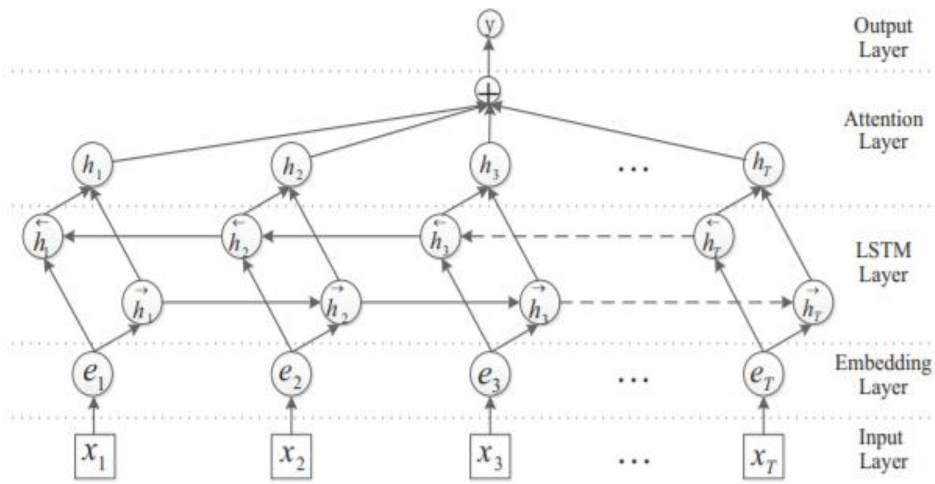
Please refer the paper before we understand the model architecture of this model.

In this paper they proposes Attention-BLSTM model in detail.

As shown in below figure, the model proposed in this paper contains five components:

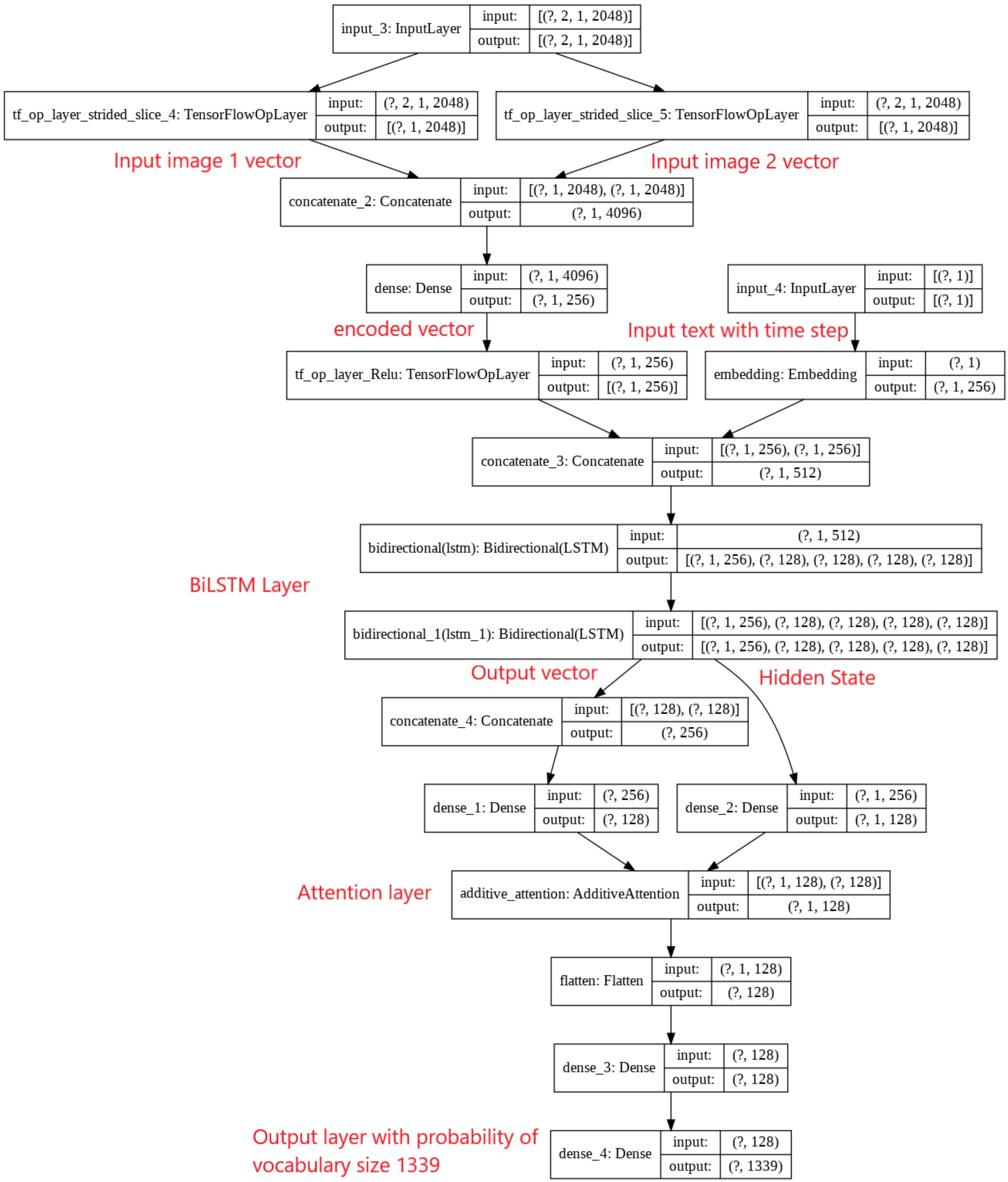
- Input layer: input sentence to this model summed with the image feature vectors
- Embedding layer: map each word into a low dimension vector
- LSTM layer: utilize BLSTM to get high level features from step (2) this BLSTM layers is repeated twice for more in depth feature understanding

- Attention layer: produce a weight vector, and merge word-level features from each time step into a sentence-level feature vector, by multiplying the weight vector
- Output layer: the sentence-level feature vector is finally used for relation classification. These components will be presented in detail functional view in coming sections.



Lets see the model functional layers using Functional Api. In this model the hyper parameters are same as the basic model only change is with the maximum sentence length is take as 80.

The same below functional model will be implemented using model subclass as I have already mentioned the subclass model is easier while debugging your architecture. We can have the control over each layer.


input_3: InputLayer
input: [(?, 2, 1, 2048)]
output: [(?, 2, 1, 2048)]
tf_op_layer_strided_slice_4: TensorFlowOpLayer
input: (?, 2, 1, 2048)
output: [(?, 1, 2048)]
tf_op_layer_strided_slice_5: TensorFlowOpLayer
input: (?, 2, 1, 2048)
output: [(?, 1, 2048)]
Input image 1 vector
Input image 2 vector
concatenate_2: Concatenate
input: [(?, 1, 2048), (?, 1, 2048)]
output: (?, 1, 4096)
dense: Dense
input: (?, 1, 4096)
output: (?, 1, 256)
input_4: InputLayer
input: [(?, 1)]
output: [(?, 1)]
encoded vector
Input text with time step
tf_op_layer_Relu: TensorFlowOpLayer
input: (?, 1, 256)
output: [(?, 1, 256)]
embedding: Embedding
input: (?, 1)
output: (?, 1, 256)
concatenate_3: Concatenate
input: [(?, 1, 256), (?, 1, 256)]
output: (?, 1, 512)
bidirectional(lstm): Bidirectional(LSTM)
input: (?, 1, 512)
output: [(?, 1, 256), (?, 128), (?, 128), (?, 128), (?, 128)]
BiLSTM Layer
bidirectional_1(lstm_1): Bidirectional(LSTM)
input: [(?, 1, 256), (?, 128), (?, 128), (?, 128), (?, 128)]
output: [(?, 1, 256), (?, 128), (?, 128), (?, 128), (?, 128)]
Output vector
Hidden State
concatenate_4: Concatenate
input: [(?, 128), (?, 128)]
output: (?, 256)
dense_1: Dense
input: (?, 256)
output: (?, 128)
dense_2: Dense
input: (?, 1, 256)
output: (?, 1, 128)
Attention layer
additive_attention: AdditiveAttention
input: [(?, 1, 128), (?, 128)]
output: (?, 1, 128)
flatten: Flatten
input: (?, 1, 128)
output: (?, 128)
dense_3: Dense
input: (?, 128)
output: (?, 128)
Output layer with probability of
vocabulary size 1339
dense_4: Dense
input: (?, 128)
output: (?, 1339)

Now we can see how this above architecture implemented using model subclass. With separate encoder and decoder part.

13.1.1 Encoder Architecture:

In the encoder part it is same as the basic model architecture summed image vector with single fully connected layer.

```
class Encoder(tf.keras.Model):
    def __init__(self, embedding_dim):
        super(Encoder, self).__init__()
        self.dense = tf.keras.layers.Dense(embedding_dim, activation='relu', kernel_initializer=tf.keras.initializers.glorot_uniform(seed=45))
        self.concat = tf.keras.layers.Concatenate()
    def call(self, x):
        # CNN two input Images concatenate to get single vector
        # Concatenating 2 images
        # Input x shape (batch_size, 2,None,2048)
        # x1 shape (batch_size, None, 2048)
        # x2 shape (batch_size, None, 2048)
        encoder_concat = self.concat([x[:,0], x[:,1]])
        x = self.dense(encoder_concat)
        x = tf.nn.relu(x)
        return x
```

13.1.2 Decoder Architecture:

Similar architecture as the mentioned paper and I have modified with one additional layer of BiLSTM for better feature representation. Attention mechanism is used. Take look at the quick overview on attention mechanism in this link. Further readings on attention mechanism is mentioned in the reference section (Attention is all you need)

In our model I have used the tensorflow AdditiveAttention it is nothing but the Bahdanau-style attention. Please refer the implementation details in the reference section.

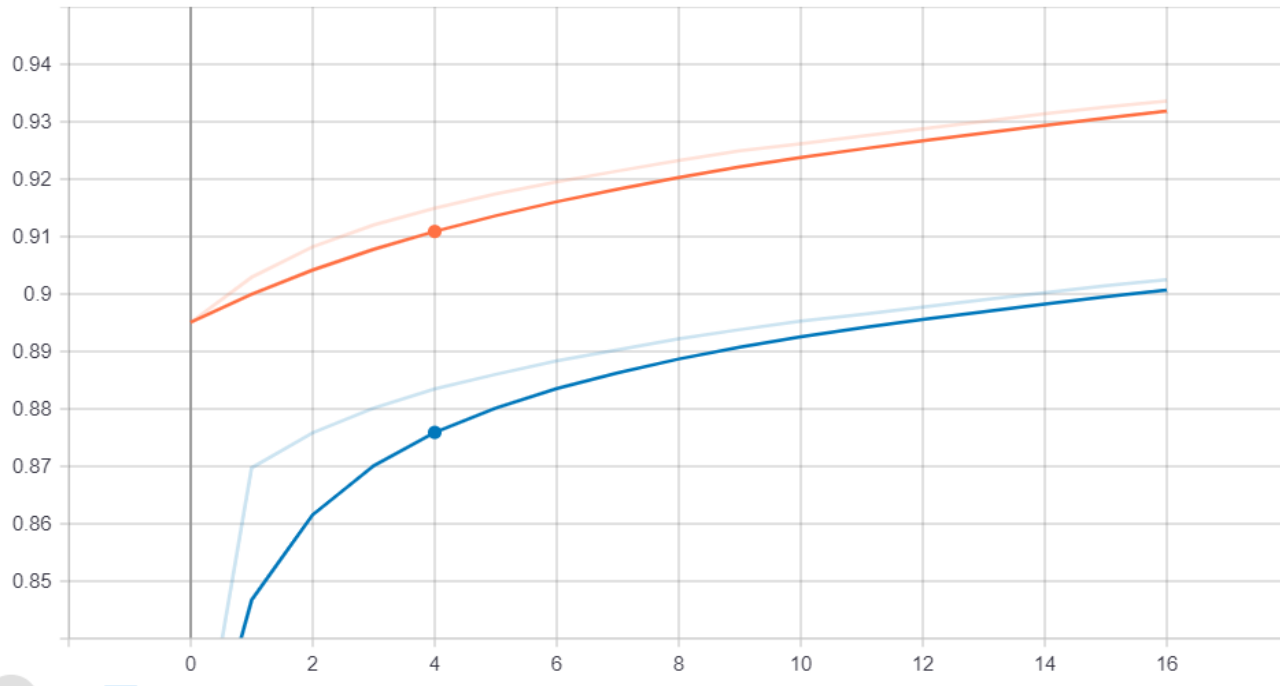
```python
class Decoder(tf.keras.Model):
    def __init__(self, embedding_dim, units, vocab_size):
        super(Decoder, self).__init__()
        self.units = units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.w_1 = tf.keras.layers.Dense(units, activation='relu')
        self.w_2 = tf.keras.layers.Dense(units, activation='relu')
        # Bidirectional LSTM
        self.bilstm_1 = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM \
                              (self.units, dropout=0.3, return_sequences=True, return_state=True, \
                               #kernel_regularizer=l2(0.001), recurrent_regularizer=l2(0.001), \
                               recurrent_activation='relu', recurrent_initializer= \
                               tf.keras.initializers.glorot_uniform(seed=26)))
        self.bilstm_2 = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM\
                              (self.units, dropout=0.2, return_sequences=True, return_state=True, \
                               #kernel_regularizer=l2(0.001), recurrent_regularizer=l2(0.001), \
                               recurrent_activation='relu', recurrent_initializer= \
                               tf.keras.initializers.glorot_uniform(seed=26)))
        self.dense_1 = tf.keras.layers.Dense(self.units, activation='relu', kernel_initializer=tf.keras.initializers.glorot_uniform(seed=45))
        self.dense_2 = tf.keras.layers.Dense(vocab_size, kernel_initializer=tf.keras.initializers.glorot_uniform(seed=45))
        #Additive Attention
        self.attention = tf.keras.layers.AdditiveAttention(self.units)
        self.concat = tf.keras.layers.Concatenate()
        self.flatten = tf.keras.layers.Flatten()
    def call(self, x):
        # x = [dec_input, features, hidden] [decoder_input_word_tensor, encoder_output, hidden_state_previous]
        embedded_layer = self.embedding(x[0])
        x_con = self.concat([embedded_layer, x[1]])
        bi_lstm = self.bilstm_1(x_con)
        lstm, forward_h, forward_c, backward_h, backward_c = self.bilstm_2(bi_lstm)
        state = self.concat([forward_h, backward_h])
        state = self.concat([state,x[2]])
        state = self.w_1(state)
        lstm = self.w_2(lstm)
        additive = self.attention([lstm,state])
        #decoder_1_1/additive_attention_1/Identity_1:0, shape=(None, max_len, 128)
        #Reshaping to (None, max_len * units)
        output = self.flatten(additive)
        output = self.dense_1(output)
        output = self.dense_2(output)
        # output will be (None, 1339)
        return output, state
```

Brief explanation and implementation of model Metric and Optimization initializer, Model trainings are mentioned basic model section same is used here in the main model.

13.2 Model Performance visualized in TensorBoard

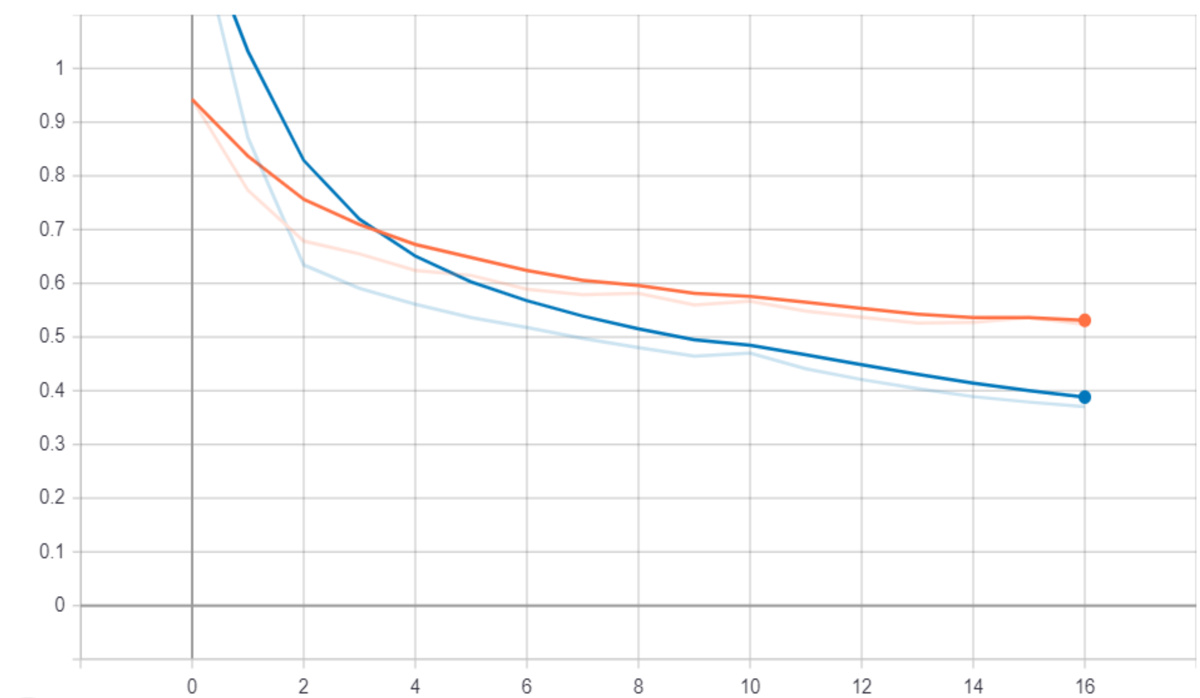
We have logged the loss and accuracy using tf.summary accuracy

accuracy
tag: accuracy



loss

loss
tag: loss



13.3 Model Evaluation

In the evaluation or testing stage I have used the Beam search-based teacher forcing to find the output sentence. As we have already seen Teacher forcing in brief lets move to the implementation part.

Bleu score metric:

I have used the Bleu (bilingual evaluation understudy) score as the metric for find the quality of the machine translated word to actual word. Take a quick look at wiki Blue here

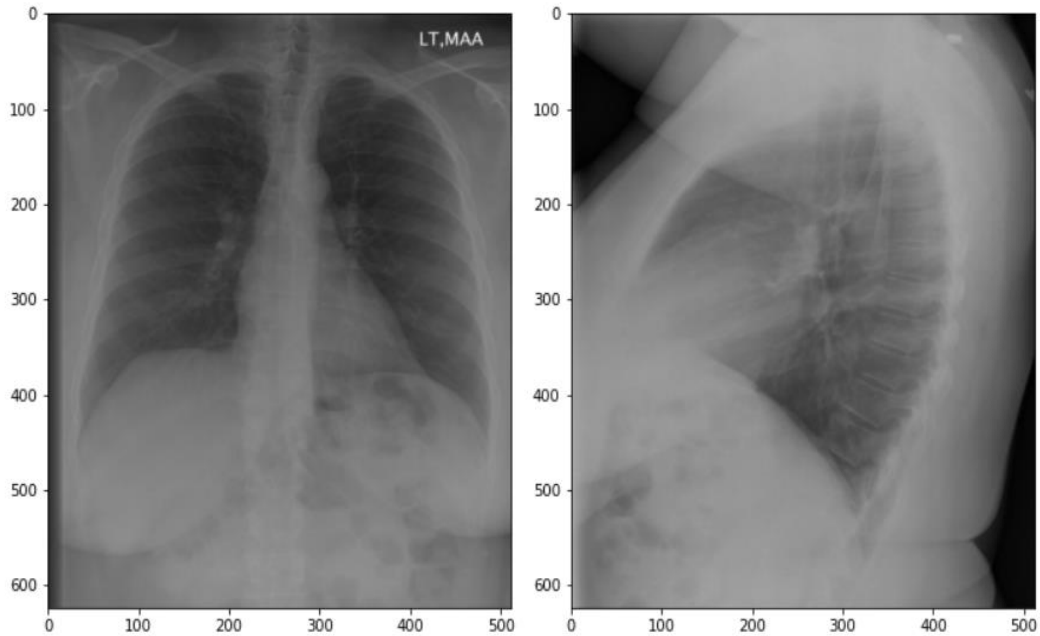
Beam Search:

Instead of greedily (usual choosing single highest probalbility word) choosing the most likely next step as the sequence is constructed, the beam search expands all possible next steps and keeps the *k* most likely k is the beam index in other words, where *k* is a user-specified parameter and controls the number of beams or parallel searches through the sequence of probabilities. Take a quick look on this source link.

```python
def calculate_score(x):
    """Calculates the cumulative score for the length of sentence"""
    return x[1]/len(x[0])

def beam_search(img_name, beam_index = 3):
    """Beam search implementaion takes images as input"""
    hidden =  tf.zeros((1, units))
    img_tensor = tf.convert_to_tensor([get_img_tensor("img/",img_name[0], image_features_model),
                                      get_img_tensor("img/",img_name[1], image_features_model)])
    img_features = tf.constant(img_tensor)[None, :]
    features_val = encoder(img_features)
    start = [tokenizer.word_index["<start>"]]
    dec_word = [[start, 0.0]]
    while len(dec_word[0][0]) < max_len_output:
        temp = []
        for s in dec_word:
            predictions, hidden = decoder([tf.cast(tf.expand_dims([s[0][-1]], 0), tf.float32), features_val, hidden])

            word_preds = np.argsort(predictions[0])[-beam_index:]
            # Getting the top <beam_index>(n) predictions and creating a
            # new list so as to put them via the model again
            for w in word_preds:
                next_cap, prob = s[0][:], s[1]
                next_cap.append(w)
                prob += predictions[0][w]
                temp.append([next_cap, prob.numpy()])
        dec_word = temp
        # Sorting according to the probabilities scores
        dec_word = sorted(dec_word, reverse=False, key=calculate_score)
        # Getting the top words
        dec_word = dec_word[-beam_index:]
    dec_word = dec_word[-1][0]
    impression = [tokenizer.index_word[i] for i in dec_word if i !=0]
    result = []

    for i in impression:
        if i != '<end>':
            result.append(i)
        else:
            break

    text = ' '.join(result[1:])
    return result, text
```

Sample outputs are shown below with bleu score in both cumulative and N gram scores.

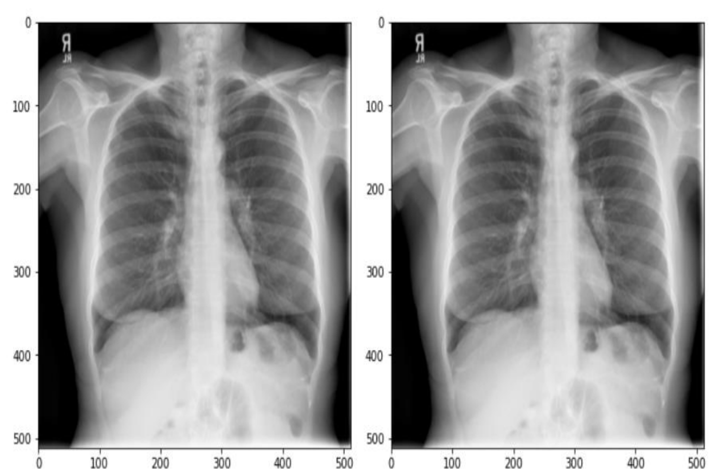
Short sentences




```
Beam Search, index= 3
==================================================
Actual <start> no acute cardiopulmonary process <end>
Predicted: no cardiopulmonary abnormalities
==================================================
Individual 1-gram: 0.4777 Cumulative 1-gram: 0.4777
Individual 2-gram: 0.7165 Cumulative 2-gram: 0.5850
Individual 3-gram: 0.7165 Cumulative 3-gram: 0.6268
Individual 4-gram: 0.7165 Cumulative 4-gram: 0.6475
```

As we can see the predicted output is good. We can also see it from the bleu score.

Longer sentence



```
Beam Search, index= 3
==================================================
Actual <start> comparison no suspicious appearing lung nodules identified wellexpanded and clear lungs mediastinal contour within normal limits no acute cardiopulmonary abnormality identified
<end>
Predicted: no evidence for disease
==================================================
Individual 1-gram: 0.0036 Cumulative 1-gram: 0.0036
Individual 2-gram: 0.0143 Cumulative 2-gram: 0.0071
Individual 3-gram: 0.0143 Cumulative 3-gram: 0.0090
Individual 4-gram: 0.0143 Cumulative 4-gram: 0.0101
```

Not a good prediction using Bleu score, but we can see there are some similarity in the meaning of the two sentence both explains there is no disease for this record. Theses are on the major problem with the Bleu score doesn't consider the meaning of the word.

Lets try another long sentence


```
Beam Search, index= 3
================================================
Actual <start> no active cardiopulmonary disease left humeral head is positioned anterior and inferior to the glenoid concerning for anterior shoulder subluxation this is related to the muscu
lar dystrophy and decreased shoulder muscles support postoperative changes from the spinal placement <end>
Predicted: no acute cardiopulmonary disease
================================================
Individual 1-gram: 0.0002 Cumulative 1-gram: 0.0002
Individual 2-gram: 0.0001 Cumulative 2-gram: 0.0001
Individual 3-gram: 0.0002 Cumulative 3-gram: 0.0001
Individual 4-gram: 0.0002 Cumulative 4-gram: 0.0001
```

This one also like the previous prediction. First 4 words exactly matches the predicted word but still we are not having the good Bleu score because the word count is high.

13.4 Model Conclusion

- The model build on Bidirectional LSTM with attention seems performing better than basic model.
- As per the Model Architecture mentioned in the source Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification it worked well in classification task.
- Loss is converged to 0.3 with accuracy of 89% train and 92% validation from the result we can see there is similarity between each predicted and actual output.

14 Conclusion

- Compared with normal ImageNet trained InceptionV3 model, we could able improve the model performance with training the same InceptionV3 in X-Ray classification and using that weights to extract the image feature.
- The BiLSTM architecture gives the good result, even in very less Bleu score values we could able to see the meaning of true sentence in the predicted sentence.
- We could perform an Error analysis on the predictions to identify the whether the issue is with model or the data point.

15 Error Analysis

In this section we will see the error analysis it is an analysis of what is causing this error and use that findings to improve the mode. we will be looking into low Bleu score data points which is error in this case and how to make sense of it. After error identification we will see how to improve the model using this.

Error in the model can be reducible or irreducible we will work on the reducible error. After training the model we check the validation set to find the error and do the analysis. Once we find the error if it is reducible error then we fix those in our future training of the model in this way the model will be improved than your previous one.

Lets find out how I have worked on this error analysis.

Validation set with Bleu score,

	idx	image_1	image_2	actual	predicted	score
0	0	CXR2978_IM-1367-4001.png	CXR2978_IM-1367-1001.png	no acute findings	no evidence of prior excluded could be identified	0.125000
1	1	CXR58_IM-2177-2001.png	CXR58_IM-2177-1001.png	no acute disease	no cardiopulmonary abnormalities	0.333333
2	2	CXR3953_IM-2021-1002.png	CXR3953_IM-2021-1001.png	no acute cardiopulmonary abnormality	no evidence of pleural disease	0.200000
3	3	CXR3227_IM-1525-1001.png	CXR3227_IM-1525-2001.png	no acute cardiopulmonary process	no cardiopulmonary abnormalities	0.477688
4	4	CXR2820_IM-1244-1001.png	CXR2820_IM-1244-2001.png	no acute disease	no cardiopulmonary disease	0.666667
5	5	CXR1029_IM-0022-1001.png	CXR1029_IM-0022-1001.png	no pneumonia heart size normal scoliosis	no evidence for pulmonary nodules of the bone ...	0.041667
6	6	CXR1510_IM-0331-2001.png	CXR1510_IM-0331-1001.png	no acute cardiopulmonary abnormality	no acute abnormalities	0.477688
7	7	CXR979_IM-2466-2001.png	CXR979_IM-2466-1001.png	negative for acute abnormality	no evidence for consolidation	0.250000
8	8	CXR3662_IM-1821-1001.png	CXR3662_IM-1821-2001.png	chest radiograph no acute radiographic cardiop...	no evidence of primordial	0.118092
9	9	CXR1303_IM-0199-2001-0001.png	CXR1303_IM-0199-2001-0002.png	right upper lobe mass suspicious for neoplasm ...	no cardiopulmonary abnormalities	0.000000
10	10	CXR1418_IM-0267-1001.png	CXR1418_IM-0267-2002.png	no comparison chest x wellexpanded and clear l...	no evidence for disease	0.007549
11	11	CXR1005_IM-0006-1001.png	CXR1005_IM-0006-3003.png	no acute findings	no acute findings	1.000000
12	12	CXR3477_IM-1690-3001.png	CXR3477_IM-1690-2001.png	no acute disease	no acute cardiopulmonary primordial	0.500000
13	13	CXR3427_IM-1657-1001.png	CXR3427_IM-1657-2001.png	there is no evidence of acute cardiopulmonary ...	no acute findings	0.012210
14	14	CXR1067_IM-0048-2001.png	CXR1067_IM-0048-1001.png	no radiographic evidence of acute cardiopulmon...	no cardiopulmonary abnormalities	0.175731
15	15	CXR1953_IM-0621-2001.png	CXR1953_IM-0621-1001.png	no radiographic evidence of acute cardiopulmon...	no acute cardiopulmonary process	0.354275
16	16	CXR1469_IM-0303-1001.png	CXR1469_IM-0303-2001.png	no acute cardiopulmonary abnormality	no acute abnormalities are identified	0.400000
17	17	CXR886_IM-2400-1002.png	CXR886_IM-2400-1001.png	no acute disease	no cardiopulmonary abnormalities	0.333333
18	18	CXR1701_IM-0462-2001.png	CXR1701_IM-0462-1001.png	no acute findings	low lung effusion	0.000000
19	19	CXR2679_IM-1153-1001.png	CXR2679_IM-1153-2001.png	normal heart size and normal mediastinal conto...	no cardiopulmonary disease	0.000156

In the initial step we will take the score which are lesser than 0.08 Bleu and check the data point.

We can also ignore the duplicate data point.We have used the duplicate data point in our model. Which is having the same image in both input we consider this as noise point check the 9. Data point construction section to find the details of data construction.

	idx	image_1	image_2	actual	predicted	score	duplicate
5	5	CXR1029_IM-0022-1001.png	CXR1029_IM-0022-1001.png	no pneumonia heart size normal scoliosis	no evidence for pulmonary nodules of the bone ...	0.041667	1
9	9	CXR1303_IM-0199-2001-0001.png	CXR1303_IM-0199-2001-0002.png	right upper lobe mass suspicious for neoplasm ...	no cardiopulmonary abnormalities	0.000000	0
10	10	CXR1418_IM-0267-1001.png	CXR1418_IM-0267-2002.png	no comparison chest x wellexpanded and clear l...	no evidence for disease	0.007549	0
13	13	CXR3427_IM-1657-1001.png	CXR3427_IM-1657-2001.png	there is no evidence of acute cardiopulmonary ...	no acute findings	0.012210	0
18	18	CXR1701_IM-0462-2001.png	CXR1701_IM-0462-1001.png	no acute findings	low lung effusion	0.000000	0

From total of 139 poor Bleu score dataset we have 22 duplicate dataset.

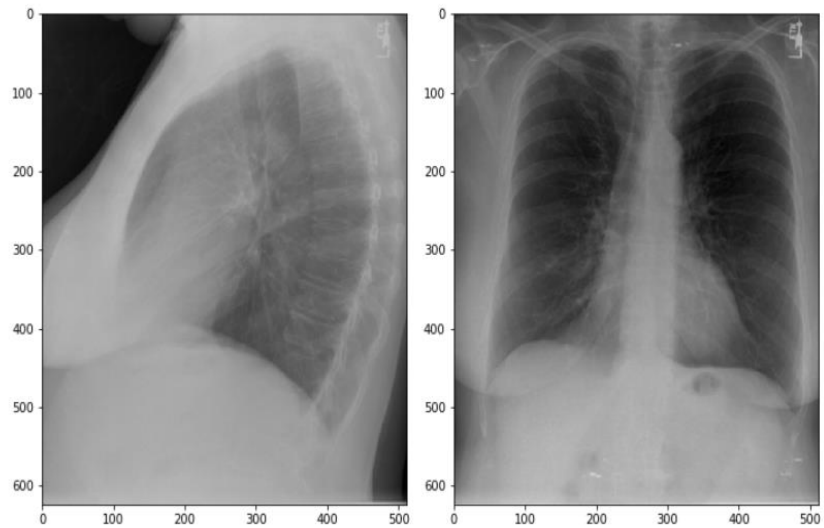
Final data point,

	idx	image_1	image_2	actual	predicted	score	actual_count
301	341	CXR1562_IM-0367-2001.png	CXR1562_IM-0367-1001.png	negative for acute cardiopulmonary abnormality	no acute subsegmental streaky airways pulmonar...	0.071429	5
26	28	CXR219_IM-0799-2001.png	CXR219_IM-0799-1001.png	no x evidence of pulmonary metastatic disease ...	no evidence for consolidation	0.067668	12
54	62	CXR1485_IM-0313-1001.png	CXR1485_IM-0313-2001.png	unchanged platelike bibasilar opacities most r...	low lung characterized within the body acute f...	0.067032	14
46	53	CXR594_IM-2187-1001.png	CXR594_IM-2187-2001.png	borderline cardiomegaly ageindeterminate chron...	low lung sequela of the heart this is within n...	0.066667	9
304	344	CXR300_IM-1385-1001.png	CXR300_IM-1385-1002.png	changes of chronic lung disease with no acute ...	no acute abnormalities	0.064648	10

Lets take each data point and do the analysis,

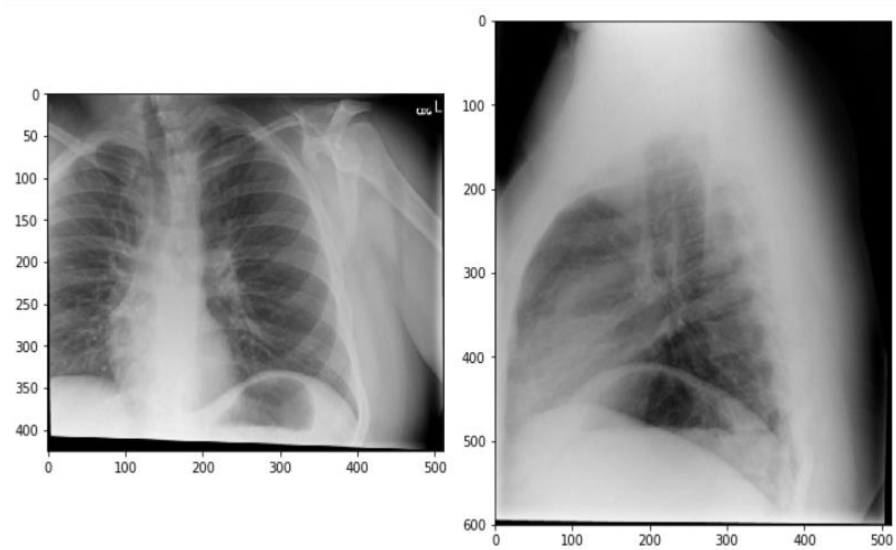

```
Score: 0.0004681758116527773
Actual: stable normal cardiac size and contour unremarkable mediastinal silhouette normal pulmonary and interstitium lungs clear no airspace disease pleural effus
ion or pneumothorax no activeacute cardiopulmonary disease
Predicted: no cardiopulmonary disease
word count: 26
```

- word length is 26 and there is a word overlap "activeacute" in actual value could not find any image issue
- Predicted word gives the partial meaning from the actual not a poor prediction.
- we get the poor value because Bleu score does not account the meaning




```
Score: 0.0
Actual: small right juxtahilar opacity may represent infiltrate in the setting of followup chest x is recommended at an appropriate interval following treatment t
o document
Predicted: no acute abnormalities
word count: 24
```

- - word count is 24, No error in actual word. still not able to find any image wise pattern issue
- - predicted word is poor did not give any similar meanings




```
Score: 0.04632230081520103
Actual: chest no visible active cardiopulmonary disease left hip advanced posttraumatic osteoarthritis
Predicted: no cardiopulmonary abnormalities
word count: 11
```

- word count is 11, No error in actual word, images is not perfectly captured when we compare with other.
- Prediction gives same meaning, issue with the Bleu score and images.

As we can see the bleu score which is having value greater than 0 gives the partial meaning of actual which considered as good prediction lets take bleu score which are 0.

Another finding is when we have word more than 20 word give 0 values. which shows that our model did not perform well for longer sentence. lets consider word lesser than 20.

Final data, having 62 data point.

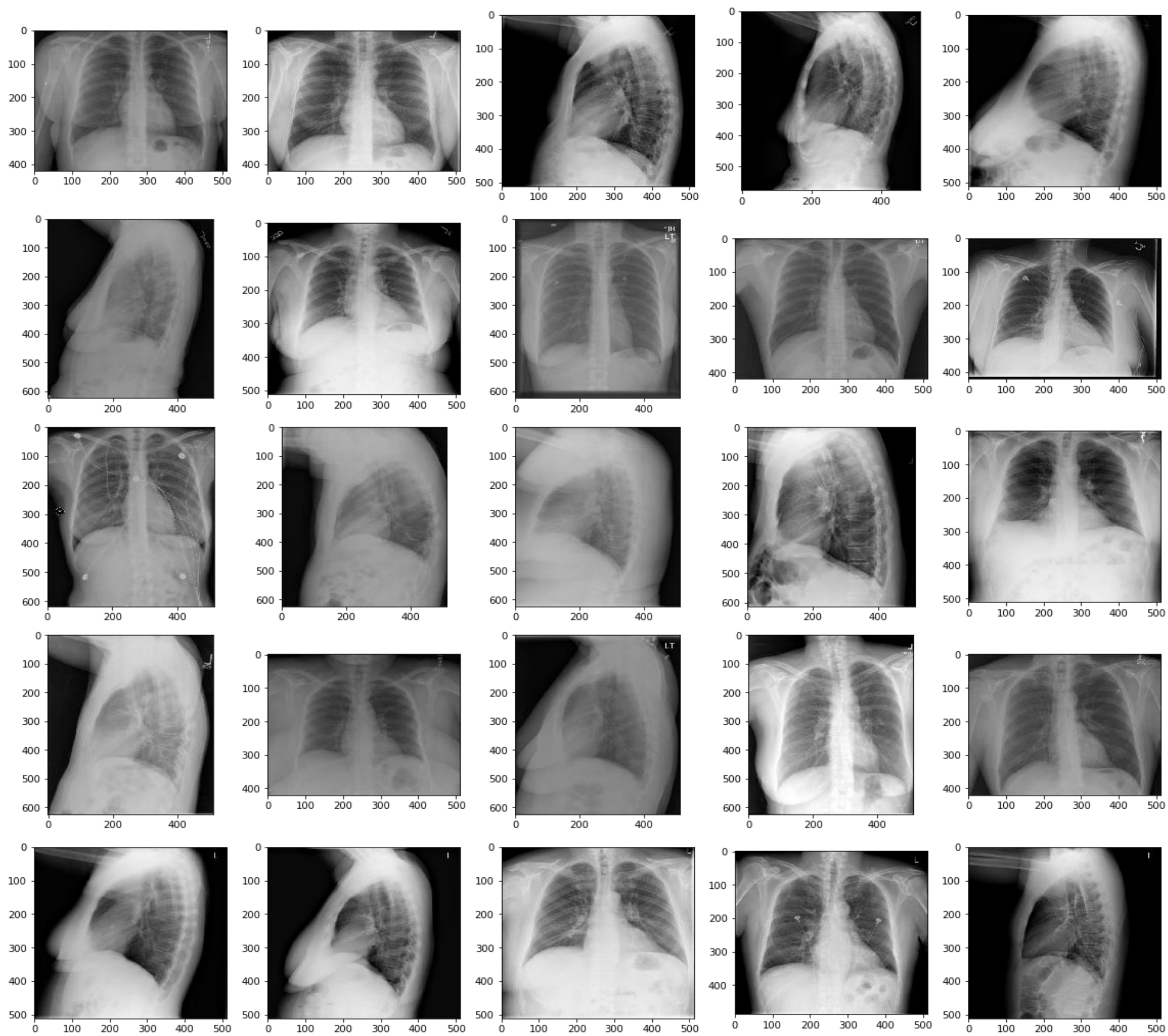
	idx	image_1	image_2	actual	predicted	score	actual_count
49	56	CXR2716_IM-1181-1001.png	CXR2716_IM-1181-2001.png	right lower lobe airspace disease with bilater...	no cardiopulmonary abnormalities	0.0	9
352	397	CXR1013_IM-0013-1001.png	CXR1013_IM-0013-2001.png	stable mild cardiomegaly without acute cardiop...	no evidence of pulmonary venous hypertension	0.0	7
184	211	CXR1304_IM-0199-2001.png	CXR1304_IM-0199-1001.png	normal chest	no acute findings	0.0	2
208	236	CXR2922_IM-1325-12012.png	CXR2922_IM-1325-1001.png	hyperinflated lungs air trapping versus inspir...	low lung features are elevation	0.0	6
8	9	CXR1303_IM-0199-2001-0001.png	CXR1303_IM-0199-2001-0002.png	right upper lobe mass suspicious for neoplasm ...	no cardiopulmonary abnormalities	0.0	19

```
df_poor_zero.shape
```

```
(62, 7)
```

As we have already separated best and worst case data points lets visualize and look for the patterns

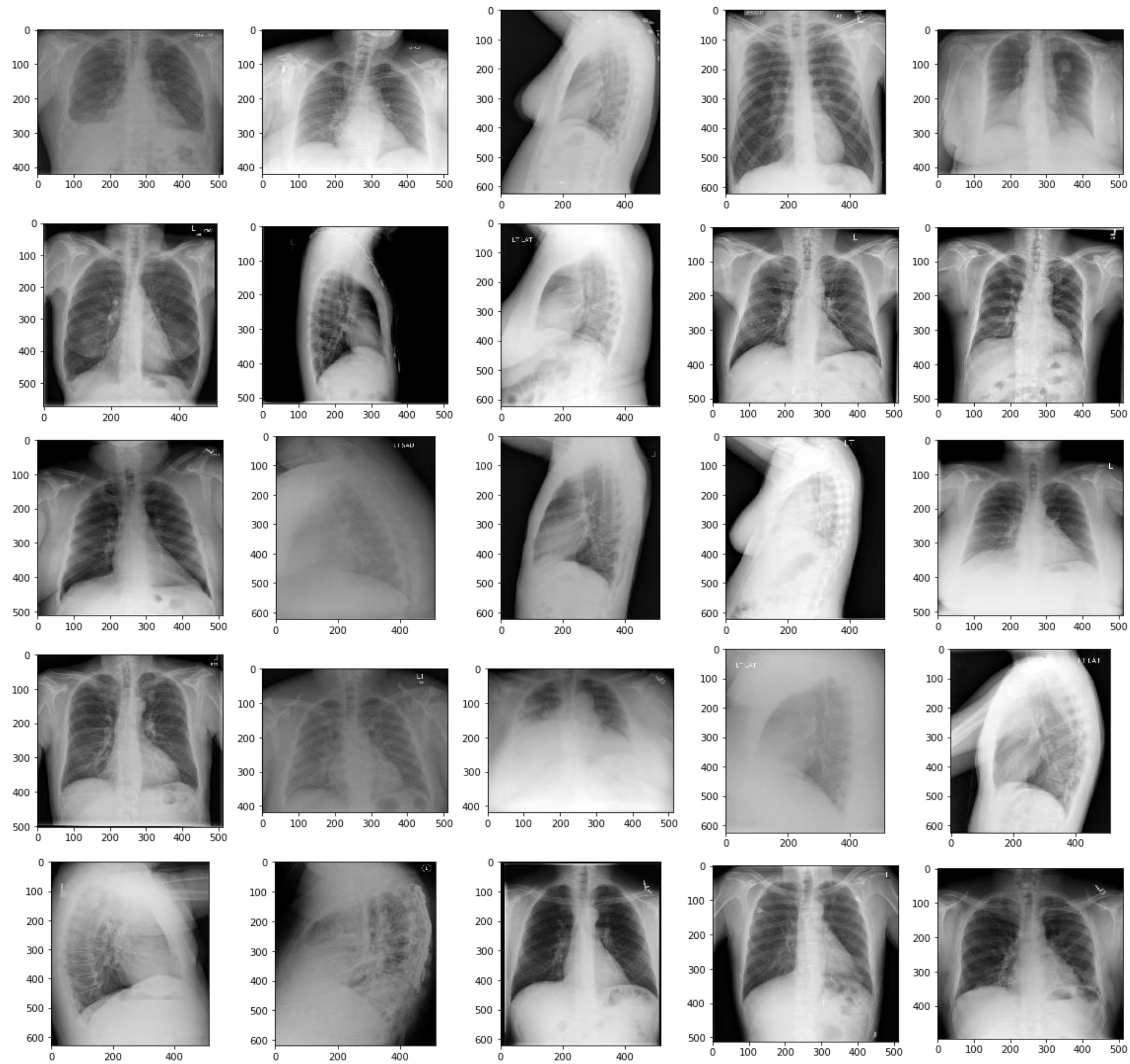
Below is the best result data points.



Points to take in best result images

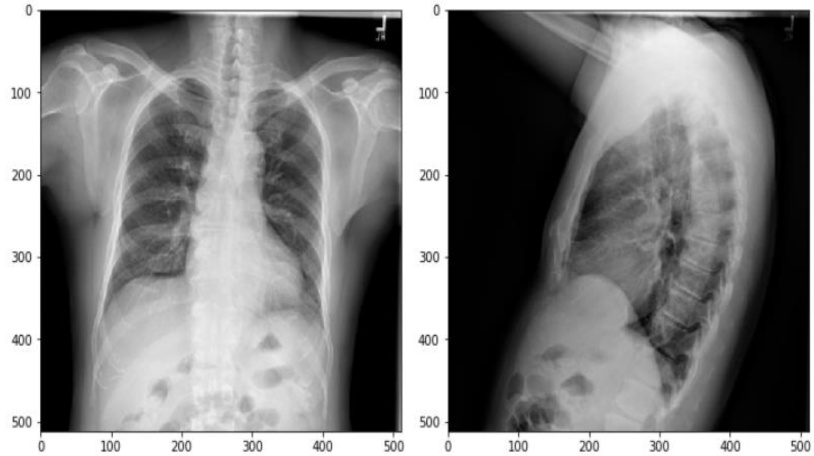
- - proper alignment of images
- - brighter view of chest bones
- - does not have any additional dark line
- - Even in the dull images we could clearly see the chest bones

Below is the poor result data points



points to take in poor result

- - images are shadowed in some case (row, column) (3,2),(3,4),(4,2),(4,3),(4,4)
- - images are too bright in some case (1,2),(3,4),(5,3)
- - Lets see both images in a data point to check whether least one image have those above issue.

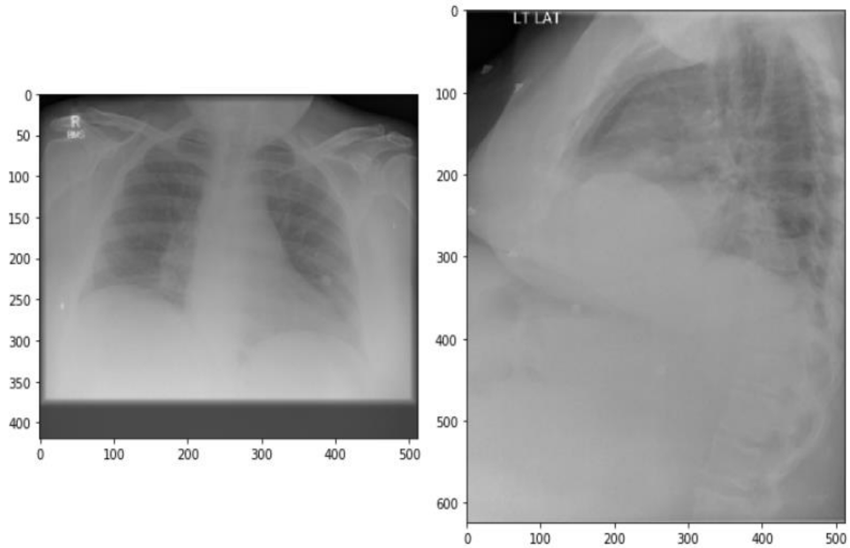



```
Score: 0.0
Actual: no evidence of active disease
Predicted: low lung features
word count: 5
```

- In this data point we see the second image is not properly taken. there is fingerprints visible in the bottom of the picture, major error.

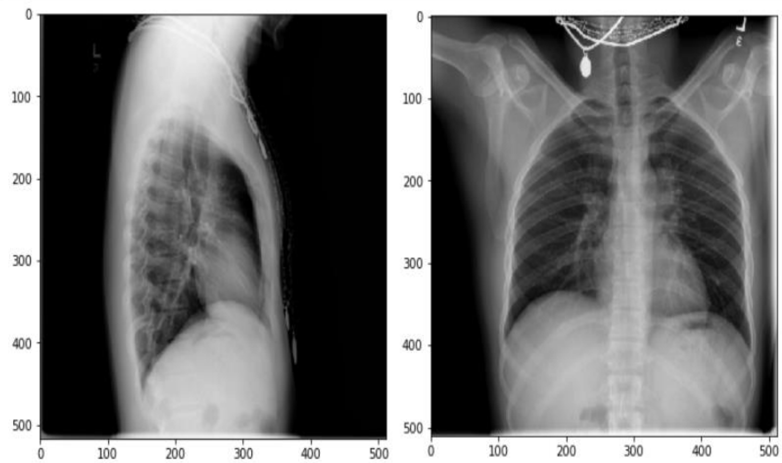

```
Score:  0.0
Actual:  bilateral small pleural effusions and associated atelectasis stable right upper mediastinal opacity consistent with goiter
Predicted:  no acute cardiopulmonary abnormality
```

- Clear view of poor xray capturing also covered the hands
- Right side image have addition dark stripes in the lower left edge



```
Score:  0.0
Actual:  mild central vascular prominence congestion heart size at the upper limits of normal
Predicted:  no acute cardiopulmonary abnormality
```

- Clear view of Poor image quality in both images.



```
Score:  0.0
Actual:  no acute findings
Predicted:  low lung effusion
```

- Clear view of poor image quality. x-ray with Jewelry in both images this is not found in any x-rays even in the x-ray classification task dataset.

15.1 Conclusion

- From the above analysis we have found that the quality of the images is plays major role. Mostly the error data points are with poor images quality poor chest bone view this is the primary take away.
- We have also seen some fingerprints, jewelry of the patient clearly visible in the image.
- The model works well on the clear visible chest bones. we have already seen this and compared in the best and worst case images.
- There are images which are brighter those cases model fails. we have also seen the best result images where we does not have the brighter images. brighter means higher white pixels.

- Another finding is that our model did not perform well in the case where we have more than 20 words. we could able to improve this by changing the architecture. better than this but our model does not show that its poor model. error are 62 out of 399 which is almost 15% of the data. does not show it is poor prediction.
- Some case where we have incorrect words in true sentence.
- We could ignore these errors in our future work to get the better performance. And these are the reducible error in error analysis.

Source Code for this blog **GitHub**

16 Future work

- We can also modify whole architecture with state-of-the-art BERT Transformer instead of att-BiLSTM. This can be achieved by sending the image feature and text input as single vector in time step to predict the next sentence. This is one the method using transformer. Few other references below
  - Unified Vision-Language Pre-Training for Image Captioning and VQA
  - https://papers.nips.cc/paper/9293-image-captioning-transforming-objects-into-words.pdf
  - https://arxiv.org/pdf/2004.08070v2.pdf - Entity-Aware News Image Captioning using Transformer
  - http://papers.nips.cc/paper/8297-vilbert-pretraining-task-agnostic-visiolinguistic-representations-for-vision-and-language-tasks.pdf - Pretraining Task-Agnostic Visio linguistic Representations for Vision-and-Language Tasks
- We can further increase the Encoder CNN layer to deep layer for improvements.
- Image can be trained on Different ImageNet based keras model like I have implemented training phase to classify the X-ray and using that weight to extract the feature.
- From Error analysis we have found that there are some data points which are in poor quality and bad capturing of image led us in poor performance. We can eliminate these issues in our future work.

17 References

1. https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/ - How to Prepare Text Data for Deep Learning
2. https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c - Teacher forcing further readings
3. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification – BiLSTM Architecture
4. Attention is all you need
5. CNN+CNN: Convolutional Decoders for Image Captioning
6. Review Networks for Caption Generation
7. AdditiveAttention – Bahdanau style attention
8. https://yashk2810.github.io/Image-Captioning-using-InceptionV3-and-Beam-Search/ - Beam search tutorial
9. Evaluating text output in NLP using Bleu – Bleu Tutorial
10. AppliedAICourse