

# Image Caption Generator for Chest X-Ray Using Deep Learning

*Shashank Pandey*

(20MIA1147)

*School of Computer Science and Engineering, Vellore Institute of Technology, Chennai 600127, Tamilnadu, India*

[shashank.pandey2020@vitstudent.ac.in](mailto:shashank.pandey2020@vitstudent.ac.in)

*Piyali Saha*

(20MIA1066)

*School of Computer Science and Engineering, Vellore Institute of Technology, Chennai 600127, Tamilnadu, India*

[piyali.saha2020@vitstudent.ac.in](mailto:piyali.saha2020@vitstudent.ac.in)

## ABSTRACT

In this paper, we present a deep learning model for generating captions for the Chest X-ray reports. Radiology reports are a critical aspect of medical diagnosis and treatment planning. Chest X-ray images are commonly used to diagnose various lung diseases, and their interpretation requires highly trained medical professionals. Generating accurate and comprehensive chest X-ray reports can be a time-consuming and expert-driven task. To address this issue, we developed a deep learning-based system for chest X-ray report generation. The system uses DenseNet a convolutional neural network (CNN) based network to analyse chest X-ray images and generate corresponding reports with captioning by help of GRU. Our model generates reports using the input chest X-ray images, and aims to provide a more efficient and cost-effective solution for healthcare professionals. The image caption generator extracts relevant features from the X-ray images and generate descriptive captions that capture important details such as abnormalities, disease progression, and treatment options. The model is trained on a large dataset of chest X-ray images and corresponding reports, and is implemented using a deep learning framework. We evaluate the performance of our model using standard metrics, and demonstrate its effectiveness in generating accurate and coherent reports.

**Keywords.** Chest X-ray, Deep learning, Image caption generator, Convolutional neural network, DenseNet, GRU

## 1. INTRODUCTION

Chest X-ray imaging is a critical tool used in the diagnosis and management of various respiratory and cardiovascular diseases. Radiologists and medical professionals spend a considerable amount of time analysing chest X-ray images to identify abnormalities and determine the appropriate treatment options. However, analysing and interpreting chest X-ray images can be a challenging and time-consuming task, even for experienced medical professionals. Furthermore, misinterpretation of these images can lead to inaccurate diagnoses and delays in treatment, which can negatively impact patient outcomes.

Recent advancements in machine learning and deep learning have shown great potential in automating various medical imaging tasks, including chest X-ray image analysis. One such application is the development of an image caption generator that can automatically generate accurate and descriptive captions for chest X-ray images. The Chest X-Ray Report Image Caption Generator is a project that aims to develop and evaluate the performance of such a system. The proposed system will utilize a deep learning-based approach that combines convolutional neural networks (CNNs) for feature extraction from X-ray images and GRU to generate descriptive captions. The CNN model will be trained on a large dataset of chest X-ray images, which will enable it to learn complex patterns and

relationships between various features present in the images. The GRU will then use these features to generate a caption that accurately describes the abnormalities and other important details present in the X-ray images.

The proposed Chest X-Ray Report Image Caption Generator project builds on this prior research and aims to develop a system that can generate accurate and descriptive captions for chest X-ray images. The system will be designed to be user-friendly and accessible to medical professionals, providing a valuable tool for clinical decision-making. One potential use case for this system is in the screening of chest X-ray images for abnormalities that may indicate the presence of lung cancer. Lung cancer is one of the most common types of cancer worldwide, and early detection is crucial for successful treatment. However, accurately identifying lung cancer from chest X-ray images can be difficult, even for experienced radiologists. By providing accurate and descriptive captions for these images, the Chest X-Ray Report Image Caption Generator can potentially improve the accuracy and speed of lung cancer detection, leading to better patient outcomes. The development of an automated image caption generator for chest X-ray images has several potential benefits. Firstly, it can reduce the workload of medical professionals, allowing them to spend more time on other important tasks such as patient care. Secondly, it can help to reduce the risk of misdiagnosis and improve patient outcomes by providing faster and more accurate diagnoses. Finally, it can potentially reduce healthcare costs by streamlining the diagnostic process and reducing the need for additional testing and follow-up visits.

In this paper, we will provide a detailed description of the proposed Chest X-Ray Report Image Caption Generator, including the architecture, implementation details, and evaluation metrics used to assess its performance. We will also compare the performance of our system with other state-of-the-art methods in the literature and discuss the potential limitations and future directions of this work.

## **2. LITERATURE REVIEW**

The authors [1] propose a method for automated radiology report captioning using a recurrent neural network (RNN). They present an approach to generate a natural language description of a radiology report by leveraging a sequence-to-sequence model based on an RNN architecture. The authors also compare their method to a baseline approach that uses a bag-of-words model and show that their RNN-based approach outperforms the baseline method in terms of caption quality and fluency. Overall, this paper provides an interesting approach to automated captioning of radiology reports using deep learning techniques, which has the potential to improve the efficiency and accuracy of radiology reporting.

This paper proposes a framework for generating high-quality chest X-ray reports from radiological images. [2] The proposed method involves two main steps: first, the images are pre-processed using intensity normalization techniques to improve the quality of the images. Then, an adversarial training method is used to generate the reports from the images. The authors demonstrate the effectiveness of their method by evaluating it on a dataset of chest X-ray images and comparing the generated reports with those written by radiologists. The results show that the generated reports are of high quality and have a high degree of similarity to the radiologist-written reports. This paper presents a promising approach to automating the generation of chest X-ray reports, which could potentially save time and improve the accuracy of medical diagnosis.

The paper [3] proposes an automated system for generating chest X-ray reports using deep learning techniques. The authors use a convolutional neural network (CNN) to extract features from the X-ray images, and a long short-term memory (LSTM) network to generate the text reports. The system is trained on a dataset of over 100,000 chest X-ray images and their corresponding reports. The authors report an accuracy of 80% in generating reports that are judged to be of "reasonable quality" by human radiologists. They also propose a method for evaluating the quality of the generated reports using a combination of automatic metrics and human evaluation. The paper demonstrates the

potential of deep learning techniques for automating the time-consuming and error-prone process of generating radiology reports. However, further work is needed to improve the quality and accuracy of the generated reports, and to ensure that they are clinically useful for radiologists.

The paper [4] discusses the use of a Transformer-based deep learning model for automatically generating reports for chest X-ray images. The authors explain that generating accurate reports for medical images is time-consuming and requires expertise, and therefore an automated approach can be beneficial for improving efficiency and accuracy. The proposed deep learning model uses a Transformer architecture, which is commonly used in natural language processing tasks, to generate text-based reports from input chest X-ray images. The authors evaluate the performance of the model using a dataset of chest X-ray images and corresponding reports, and report promising results. The paper concludes that the proposed Transformer-based deep learning model can be used for automated report generation for chest X-ray images, and has the potential to improve the efficiency and accuracy of the reporting process. The research is significant for the field of medical image analysis and highlights the potential of deep learning approaches for improving healthcare outcomes.

The paper [5] proposes a system that automatically generates chest X-ray reports using convolutional neural networks (CNNs). The authors use a pre-trained CNN to extract features from the input chest X-ray image, which is then passed through a sequence-to-sequence model for text generation. The system was evaluated on a dataset of 1128 chest X-ray images and corresponding reports, achieving an accuracy of 79.2% in generating the correct report for a given image. The authors conclude that their system can potentially save time and effort for radiologists in generating reports and providing diagnoses.

### **3. IMPLEMENTATION AND ANALYSIS**

#### **3.1. DATASET DESCRIPTION**

The Chest X-rays Indiana University dataset, available on Kaggle, is a collection of chest X-ray images and their corresponding reports, collected from Indiana University hospitals. The dataset consists of a total of 8,047 chest X-ray images, taken from 7,470 unique patients, with each image having an associated text report describing the findings of the radiologist.

The images in the dataset are in DICOM format, which is a standard format used for storing medical images. The images are in grayscale and have varying dimensions, with the largest image being 3,600 x 3,600 pixels and the smallest being 1,250 x 1,000 pixels. The text reports associated with the images are in plain text format and contain a detailed description of the findings of the radiologist who analysed the chest X-ray image. The reports typically include information such as the presence of abnormalities, the location of the abnormalities, and the severity of the condition. The reports also


include a conclusion section, where the radiologist provides their diagnosis and recommendation for further treatment.

## Chest X-rays (Indiana University)

Data CardCode (16)Discussion (0)

67New NotebookDownload (14 GB)

1000\_IM-0003-1001.dcm.png (2.14 MB)



Data Explorer

Version 2 (14.19 GB)

images

images\_normalized

1000\_IM-0003-1001000\_IM-0003-2001000\_IM-0003-3001001\_IM-0004-1001001\_IM-0004-1001002\_IM-0004-1001002\_IM-0004-2001003\_IM-0005-2001004\_IM-0005-1001004\_IM-0005-2001005\_IM-0006-1001005\_IM-0006-3001006\_IM-0007-1001006\_IM-0007-3001007\_IM-0008-1001007\_IM-0008-2001007\_IM-0008-3001008\_IM-0009-2001008\_IM-0009-4001009\_IM-0010-1001009\_IM-0010-2001009\_IM-0010-300

Summary

7472 files


11 columns

## Chest X-rays (Indiana University)

Data CardCode (16)Discussion (0)

67New NotebookDownload (14 GB)

indiana\_reports.csv (1.68 MB)



DetailCompactColumn

8 of 8 columns

uid	MeSH	Problems	image	Indication	compa
1	normal	36%	normal	36%	Xray Chest PA and Lateral
2	Cardiomegaly;borderline;Pulmonary Artery/enlarged	Cardiomegaly;Pulmonary Artery	Chest, 2 views, frontal and lateral	Preop bariatric surgery.	None.
3	normal	normal	Xray Chest PA and Lateral	rib pain after a XXXX, XXXX XXXX steps this XXXX. Pain to R back, R elbow and R rib XXXX, no previou...	None.
4	Pulmonary Disease, Chronic Obstructive;Bullous Emphysema;Pulmonary Fibrosis/interstitia	Pulmonary Disease, Chronic Obstructive;Bullous Emphysema;Pulmonary Fibrosis;Cicatrix;Op	PA and lateral views of the chest XXXX, XXXX at XXXX hours	XXXX-year-old XXXX with XXXX.	None av

Data Explorer

Version 2 (14.19 GB)

images

images\_normalized

1000\_IM-0003-1001000\_IM-0003-2001000\_IM-0003-3001001\_IM-0004-1001001\_IM-0004-1001002\_IM-0004-1001002\_IM-0004-2001003\_IM-0005-2001004\_IM-0005-1001004\_IM-0005-2001005\_IM-0006-1001005\_IM-0006-3001006\_IM-0007-1001006\_IM-0007-3001007\_IM-0008-1001007\_IM-0008-2001007\_IM-0008-3001008\_IM-0009-2001008\_IM-0009-4001009\_IM-0010-1001009\_IM-0010-2001009\_IM-0010-300

Summary

7472 files

11 columns

## Chest X-rays (Indiana University)

Data Card Code (16) Discussion (0)

67

New Notebook

Download (14 GB)

indiana\_projections.csv (289.4 kB)

Download View

Version 2 (14.19 GB)

Detail Compact Column

3 of 3 columns

### About this file

Images classified to Frontal/Lateral

uid	filename	projection	
7466 unique values			
1	1_IM-0001-4001.dcm.png	Frontal	
1	1_IM-0001-3001.dcm.png	Lateral	
2	2_IM-0652-1001.dcm.png	Frontal	
2	2_IM-0652-2001.dcm.png	Lateral	
3	3_IM-1384-1001.dcm.png	Frontal	
3	3_IM-1384-2001.dcm.png	Lateral	
4	4_IM-2050-1001.dcm.png	Frontal	
4	4_IM-2050-	Lateral	

images  
indiana\_projections.csv  
indiana\_reports.csv

### Summary

7472 files  
11 columns

Fig 1: Chest Xray dataset from Kaggle

Fig 1 includes a wide range of chest X-ray abnormalities, including pneumonia, tuberculosis, emphysema, and lung cancer. The images were collected over a period of several years, from a diverse patient population, and were analysed by multiple radiologists, providing a rich and diverse set of data for developing and evaluating automated image analysis algorithms.

The dataset is split into training, validation, and testing sets, with approximately 80% of the data used for training, 10% for validation, and 10% for testing. The dataset also includes metadata for each image, including the patient's age, gender, and the date the X-ray was taken. This metadata can be used to perform additional analysis and stratification of the data based on patient demographics or time period.

## 3.2. METHODOLOGY

We developed a deep learning-based system that uses a CNN to extract relevant features from the chest X-ray images and a natural language processing (NLP) model to generate corresponding reports. We used a modified version of the DensNet architecture for our CNN model. The NLP model was trained using a combination of encoder-decoder models and attention mechanisms. We used a large dataset of total of 8,047 chest X-ray images from 7,470 unique patients to train and evaluate our model. We evaluated our model's performance using the BLEU score, which measures the similarity between the generated report and the ground truth report.

### 3.2.1. Data Pre-processing

## Text Pre-processing

Text pre-processing refers to the various steps that are performed on raw text data to transform it into a more structured and analysable format. Text data often contains noise, irrelevant information, and inconsistencies that can negatively impact the results of any analysis or machine learning task. The following are some of the most common text pre-processing techniques used in our paper Tokenization, Lowercasing, Stopword removal, Punctuation removal, Numerical removal, Lemmatization, Stemming.

```
def remove_special_chars(text):
    re1 = re.compile(r' +')
    x1 = text.lower().replace('#39;', '').replace('amp;', '&').replace('#146;', '').replace(
        'nbsp;', ' ').replace('#36;', '$').replace('\n', '\n').replace('quot;', '').replace(
        '<br />', '\n').replace('\\"', '').replace('<unk>', 'u_n').replace('@.@ ', '.').replace(
        '@-@ ', '-').replace('\\"', '\\ ')
    return re1.sub(' ', html.unescape(x1))
```

- `remove_special_chars(text)`: This function removes special characters, such as HTML tags and escape sequences, from the text. It uses regular expressions to replace multiple spaces with a single space, unescape HTML characters, and remove unwanted characters.

```
def remove_non_ascii(text):
    """Remove non-ASCII characters from list of tokenized words"""
    return unicodedata.normalize('NFKD', text).encode('ascii', 'ignore').decode('utf-8', 'ignore')
```

- `remove_non_ascii(text)`: This function removes non-ASCII characters from the text. It uses the `unicodedata` module to normalize the text to the NFKD (Normalization Form Compatibility Decomposition) form, which separates characters into their basic components. It then removes all non-ASCII characters by encoding the text to ASCII and ignoring any errors.

```
def to_lowercase(text):
    return text.lower()
```

- `to_lowercase(text)`: This function converts all text to lowercase, which is a common pre-processing step to normalize the text.

```
def remove_punctuation(text):
    """Remove punctuation from list of tokenized words"""
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator)
```

- `remove_punctuation(text)`: This function removes all punctuation from the text. It uses the `string` module to obtain a string of all punctuation characters and then removes them using the `translate()` method.

```
def replace_numbers(text):
    """Replace all integer occurrences in list of tokenized words with textual representation"""
    return re.sub(r'\d+', '', text)
```

- `replace_numbers(text)`: This function replaces all integers in the text with an empty string. This is often done to prevent the model from learning spurious associations between numbers and labels.

```
def remove_whitespace(text):
    return text.strip()
```

- `remove_whitespace(text)`: This function removes any leading or trailing whitespaces in the text.

```
def remove_stopwords(words, stop_words):
    """
    :param words:
    :type words:
    :param stop_words: from sklearn.feature_extraction.stop_words import ENGLISH_STOP_WORDS
    or
    from spacy.lang.en.stop_words import STOP_WORDS
    :type stop_words:
    :return:
    :rtype:
    """
    return [word for word in words if word not in stop_words]
```

- `remove_stopwords(words, stop_words)`: This function removes stop words from the text. Stop words are commonly occurring words in a language that do not carry much meaning, such as "the" and "and". This function takes a list of words and a set of stop words as input, and returns a new list of words with the stop words removed.

```
def stem_words(words):
    """Stem words in text"""
    stemmer = PorterStemmer()
    return [stemmer.stem(word) for word in words]
```

- `stem_words(words)`: This function stems the words in the text. Stemming is the process of reducing a word to its root or base form, which can help to reduce the vocabulary size and improve model performance. This function uses the Porter stemming algorithm, which is a widely used and effective stemming algorithm.

```
def lemmatize_words(words):
    """Lemmatize words in text"""
    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(word) for word in words]
```

- `lemmatize_words(words)`: This function lemmatizes the words in the text. Lemmatization is the process of reducing a word to its base form or lemma, which can also help to reduce the



vocabulary size and improve model performance. This function uses the WordNet lemmatization algorithm, which is based on a large lexical database of English words.

```
def lemmatize_verbs(words):  
    """Lemmatize verbs in text"""  
  
    lemmatizer = WordNetLemmatizer()  
    return ' '.join([lemmatizer.lemmatize(word, pos='v') for word in words])
```

- `lemmatize_verbs(words)`: This function lemmatizes only the verbs in the text. This can be useful in cases where the model needs to distinguish between different parts of speech, such as in natural language generation tasks.

```
def text2words(text):  
    return word_tokenize(text)
```

- `text2words(text)`: This function tokenizes the text into individual words. It uses the `word_tokenize()` function from the `nlTK` module, which splits the text into words based on whitespace and punctuation.

```
def normalize_text(text):  
    text = remove_special_chars(text)  
    text = remove_non_ascii(text)  
    text = remove_punctuation(text)  
    text = to_lowercase(text)  
    text = replace_numbers(text)  
    #words = text2words(text)  
    #stop_words = stopwords.words('english')  
    #words = remove_stopwords(words, stop_words)  
    #words = stem_words(words) # Either stem or word lemmatize  
    #words = lemmatize_words(words)  
    #words = lemmatize_verbs(words)  
  
    return text
```

- `normalize_text(text)`: This function applies all of the above pre-processing steps to the text in sequence. It takes a string of text as input and returns the pre-processed text as a string.

```
def normalize_corpus(corpus):  
    return [normalize_text(t) for t in corpus]
```

- `normalize_corpus(corpus)`: This function applies the `normalize_text()` function to each document in the corpus, which is a collection of texts. It returns a list of pre-processed documents, where each document is a string. This function can be used to pre-process a large corpus of text data efficiently.



```
[17]: df['report'] = df[df.columns[1:]].apply(
      lambda x: ','.join(x.astype(str)),
      axis=1
    )
    df['report'].head()

[17]: 0    normal,normal,Xray Chest PA and Lateral,Positi...
      1    Cardiomegaly/borderline;Pulmonary Artery/enlar...
      2    normal,normal,Xray Chest PA and Lateral,rib pa...
      3    Pulmonary Disease, Chronic Obstructive;Bullous...
      4    Osteophyte/thoracic vertebrae/multiple/small;T...
      Name: report, dtype: object

[18]: df['report'] = df['report'].apply(normalize_text)

[19]: df['report'] = '<start> '+df['report']+' <end>'
```

The above pre-processing steps will help to prepare the data for the image-captioning model. The cleaned and normalized text data will be used as inputs to the model along with the image data. The model will learn the relationship between the image and text data and generate accurate captions for the chest X-ray images.

### Merge Images Path

```
[24]: df2 = df2[df2['projection']=='Frontal']

[25]: df = pd.merge(df, df2, on=['uid'])

[26]: df.head()
```

	uid	MeSH	Problems	image	indication	comparison	findings	impression	report	filename	projection
0	1	normal	normal	Xray Chest PA and Lateral	Positive TB test	None	The cardiac silhouette and mediastinum size ar...	Normal chest x-XXXX.	<start> normalnormalxray chest pa and lateral...	1_IM-0001-4001.dcm.png	Frontal
1	2	Cardiomegaly/borderline;Pulmonary Artery/enlarged	Cardiomegaly;Pulmonary Artery	Chest, 2 views, frontal and lateral	Preop bariatric surgery.	None	Borderline cardiomegaly. Midline sternotomy XX...	No acute pulmonary findings.	<start> cardiomegaly/borderlinepulmonary artery...	2_IM-0692-1001.dcm.png	Frontal
2	3	normal	normal	Xray Chest PA and Lateral	rib pain after a XXXXX, XXXXX XXXX steps this XCL.	NaN	NaN	No displaced rib fractures, pneumothorax, or p...	<start> normalnormalxray chest pa and lateral...	3_IM-1384-1001.dcm.png	Frontal
3	4	Pulmonary Disease, Chronic Obstructive;Bullous...	Pulmonary Disease, Chronic Obstructive;Bullous...	PA and lateral views of the chest XXXXX, XXXXX a...	XXXXX-year-old XXXXX with XXXXX.	None available	There are diffuse bilateral interstitial and a...	1. Bullous emphysema and interstitial fibrosis...	<start> pulmonary disease chronic obstructive...	4_IM-2050-1001.dcm.png	Frontal
4	5	Osteophyte/thoracic vertebrae/multiple/small;T...	Osteophyte;thickening;Lung	Xray Chest PA and Lateral	Chest and nasal congestion.	NaN	The cardiomeastinal silhouette and pulmonary...	No acute cardiopulmonary abnormality.	<start> osteophyte/thoracic vertebraemultiplesm...	5_IM-2117-1003032.dcm.png	Frontal

Now, we are merging both the csv files i.e., Indiana\_projection.csv and Indiana\_reports.csv into one data frame so that we can easily get the report findings and impressions for the chest Xray using one data frame and the above figure shows the merged data frame for our chest Xray impression csv.

### 3.2.2. Model Building

```

# Input layers
img_input = layers.Input(shape= (256,256,3))
report_input= layers.Input(shape= (max_len-1,))

# Encoder

Densenet_model = tf.keras.applications.DenseNet121(
    include_top=False,
    weights=None,
    input_shape=(256,256,3),
)
number_of_encoder_layers= len(Densenet_model.layers)

encoder_output = Densenet_model(img_input)
encoder_output = layers.Flatten()(encoder_output)
encoder_output = layers.Dropout(0.2)(encoder_output)
encoder_output = layers.Dense(512,activation='relu')(encoder_output)

#decoder

#layers
gru_layer = layers.GRU(512, return_sequences=True)
dense_layer= layers.Dense(vocab_size,activation='softmax')
embedding_layer = layers.Embedding(vocab_size, 300, mask_zero=True)
dropout = layers.Dropout(0.2)

# decoder model
embedding_output = embedding_layer(report_input)
gru_output = gru_layer(embedding_output, initial_state=encoder_output )
gru_output = dropout(gru_output)
output = dense_layer(gru_output)
model = Model([img_input,report_input ],output)

#Inference models

#encoder_inference model
encoder_model = Model(img_input,encoder_output)

#decoder_inference model
prev_hidden_state= layers.Input(shape= (512))
report_input2 = layers.Input(shape= (1,))
embedding_output2= embedding_layer(report_input2)
gru_output2 = gru_layer(embedding_output2, initial_state=prev_hidden_state )
gru_output2 = dropout(gru_output2)
output2 = dense_layer(gru_output2)
decoder_model = Model([report_input2,prev_hidden_state],[output2,gru_output2])

```

This implementation of a deep learning model for generating captions for chest X-ray reports. The model has an encoder-decoder architecture, where the encoder is a pre-trained DenseNet121 model that extracts relevant features from chest X-ray images. The encoder output is passed through a flattening layer, dropout layer, and a fully connected layer to reduce the feature dimensionality and improve the model's generalization capability. The decoder consists of an embedding layer, a GRU layer, a dropout layer, and a dense layer that maps the decoder output to the vocabulary size of the caption. The decoder takes as input the encoder output and the caption's initial hidden state, which is fed to the GRU layer for generating the caption. Two inference models are also defined: the encoder inference model that takes a chest X-ray image as input and outputs the corresponding encoder output, and the decoder inference model that takes the previous hidden state and the last generated word as inputs and outputs the next word and the current hidden state.



```
model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 256, 256, 3)]	0	[]
densenet121 (Functional)	(None, 8, 8, 1024)	7037504	['input_1[0][0]']
flatten (Flatten)	(None, 65536)	0	['densenet121[0][0]']
input_2 (InputLayer)	[(None, 259)]	0	[]
dropout (Dropout)	(None, 65536)	0	['flatten[0][0]']
embedding (Embedding)	multiple	3000000	['input_2[0][0]']
dense (Dense)	(None, 512)	33554944	['dropout[0][0]']
gru (GRU)	multiple	1250304	['embedding[0][0]', 'dense[0][0]']
dropout_1 (Dropout)	multiple	0	['gru[0][0]']
dense_1 (Dense)	multiple	5130000	['dropout_1[0][0]']

=====  
Total params: 49,972,752  
Trainable params: 49,889,104  
Non-trainable params: 83,648  
=====



```
plot_model(model)
```

[37...

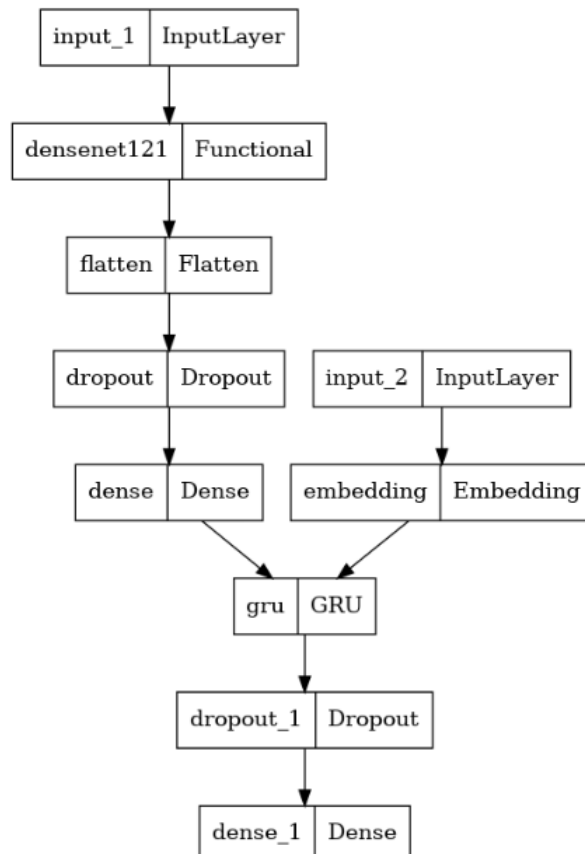


Fig: Architecture of Model

The model architecture consists of an encoder-decoder framework for generating captions for chest X-ray reports. The encoder uses a pre-trained DenseNet121 model to extract features from the input chest X-ray images. The encoder output is passed through several layers to reduce the feature

dimensionality and improve the model's generalization capability. The decoder consists of an embedding layer, a GRU layer, a dropout layer, and a dense layer. The embedding layer maps the input report sequence to a dense vector space. The GRU layer processes the input sequence and generates the corresponding caption one word at a time. The dense layer maps the decoder output to the vocabulary size of the caption. The decoder takes as input the encoder output and the caption's initial hidden state, which is fed to the GRU layer for generating the caption. Two inference models are also defined, the encoder inference model, and the decoder inference model. The encoder inference model takes a chest X-ray image as input and outputs the corresponding encoder output, while the decoder inference model takes the previous hidden state and the last generated word as inputs and outputs the next word and the current hidden state.

```
import numpy as np
from nltk.translate.bleu_score import corpus_bleu

def tokens_to_text(tokens, tok, end_token='end'):
    sentence=""
    for token in tokens:
        if token ==0:
            break
        word = tok.index_word[token]
        if word==end_token:
            break
        sentence+= word+" "
    sentence = sentence.strip()
    return sentence

def greedy_inference(input_img, tok, encoder_model, decoder_model, max_len, start_token="start", end_token='end', decoder_type="GRU"):
    if decoder_type=='LSTM':
        a0,c0 =encoder_model(np.expand_dims(input_img,axis=0))
    elif decoder_type=='GRU':
        hidden_layer =encoder_model(np.expand_dims(input_img,axis=0))

    word = tok.word_index[start_token]
    words = []
    for index in range(max_len):
        if decoder_type=='LSTM':
            word_probs , a0,c0 = decoder_model.predict([np.array([word]),a0,c0])
        elif decoder_type=='GRU':
            word_probs , hidden_layer = decoder_model.predict([np.array([word]),hidden_layer])
            hidden_layer=hidden_layer[0]
        word = np.argmax(word_probs)
        try:
            if tok.index_word[word]==end_token:
                break
        except:
            pass

        words.append(word)
    words = tokens_to_text(words,tok)
    return words
```

The above code defines two functions. The first function 'tokens\_to\_text' takes a list of integer tokens, a tokenizer, and end token as inputs. It converts the integer tokens back to a string of words using the provided tokenizer, stopping at the end token or the maximum length of the report.

The second function 'greedy\_inference' takes an input image, a tokenizer, an encoder model, a decoder model, the maximum length of the report, and start and end tokens as inputs. It first passes the input image through the encoder model to obtain either the final hidden state or both the final hidden state and cell state, depending on the decoder type. It then initializes the word variable to the index of the start token in the tokenizer's word\_index attribute.

The function generates a report for the input image using a greedy inference algorithm. It initializes the decoder with the start token and repeatedly predicts the next word in the report until the end

token is predicted or the maximum length is reached. The function uses the encoder model to obtain the encoder output, which is then used to initialize the hidden state of the decoder. The function returns the generated report as a string. The function supports two types of decoders: LSTM and GRU.

```
def get_predictions_from_data_loader(data_loader, tok, encoder_model, decoder_model, max_len, start_token="start",
                                     end_token="end", inference_type='greedy', decoder_type='GRU'):

    data_loader_iterator = data_loader.__iter__()

    pred_sentences = []
    Gt_sentences = []
    for index, (X,Y) in enumerate(data_loader_iterator):
        for img,_, sample_y in zip(X[0],X[1],Y):

            if inference_type=='greedy':
                pred_sentence = greedy_inference(img, tok, encoder_model, decoder_model, max_len,
                                                start_token="start", end_token="end", decoder_type='GRU')

            GT_sentence = tokens_to_text(sample_y, tok)

            pred_sentences.append(pred_sentence)
            Gt_sentences.append(GT_sentence)

        if index == data_loader.nb_iteration -1:
            break
        print("Done with batch number: {} ".format(index))

    return Gt_sentences, pred_sentences
```

Here in the above given code defines a function called `get_predictions_from_data_loader` which takes a data loader object, tokenizer object, encoder model, decoder model, max length of sequence, start token, end token, inference type, and decoder type as input parameters.

The function iterates over the data loader object to retrieve batches of image and sequence data. For each image in the batch, the function generates a predicted sentence using the `greedy_inference` function defined earlier in the code. It also retrieves the ground truth sentence from the sequence data using the `tokens_to_text` function. Both the predicted and ground truth sentences are appended to separate lists.

The function returns two lists, one containing the ground truth sentences and the other containing the predicted sentences for all the images in the data loader. Additionally, the function prints the progress of processing batches of data during execution.

```
def calculate_bleu_evaluation(Gt_sentences, predicted_sentences):
    BLEU_1 = corpus_bleu(Gt_sentences, predicted_sentences, weights=(1.0, 0, 0, 0))
    BLEU_2 = corpus_bleu(Gt_sentences, predicted_sentences, weights=(0.5, 0.5, 0, 0))
    BLEU_3 = corpus_bleu(Gt_sentences, predicted_sentences, weights=(0.3, 0.3, 0.3, 0))
    BLEU_4 = corpus_bleu(Gt_sentences, predicted_sentences, weights=(0.25, 0.25, 0.25, 0.25))

    return BLEU_1, BLEU_2, BLEU_3, BLEU_4

+ Code + Markdown

[44]: def evaluate_from_data_loader(data_loader, tok, encoder_model, decoder_model, max_len, start_token="start", end_token="end", inference_type='greedy', decoder_type="GRU"):
    Gt_sentences, pred_sentences = get_predictions_from_data_loader(data_loader, tok, encoder_model, decoder_model, max_len, start_token=start_token, end_token=end_token, inference_type=inference_type, decoder_type=decoder_type)
    BLEU_1, BLEU_2, BLEU_3, BLEU_4 = calculate_bleu_evaluation(Gt_sentences, pred_sentences)

    return BLEU_1, BLEU_2, BLEU_3, BLEU_4

BLEU_1, BLEU_2, BLEU_3, BLEU_4 = evaluate_from_data_loader(val_data_loader, tok, encoder_model, decoder_model, max_len)
print(BLEU_1)
print('-----')
print(BLEU_2)
print('-----')
print(BLEU_3)
print('-----')
print(BLEU_4)

1/1 [=====] - 2s 25/step
1/1 [=====] - 0s 22ms/step
1/1 [=====] - 0s 23ms/step
1/1 [=====] - 0s 24ms/step
1/1 [=====] - 0s 23ms/step
1/1 [=====] - 0s 23ms/step
```

Now we have defined a function `calculate_bleu_evaluation` uses the `corpus_bleu` function from the `nlk.translate.bleu_score` module to calculate the BLEU-1, BLEU-2, BLEU-3 and BLEU-4 scores for the predicted sentences with respect to the ground truth sentences. The BLEU scores are calculated with

different weights for n-grams up to length 4. Finally, the function returns the BLEU scores as a tuple of four floats.

```
1/1 [=====] - 0s 23ms/step
1/1 [=====] - 0s 22ms/step
1/1 [=====] - 0s 22ms/step
1/1 [=====] - 0s 23ms/step
/opt/conda/lib/python3.7/site-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)
0.0902628941484851
-----
0.300437837411477
-----
0.48601847268849496
-----
0.5481221008237827
```

And we have finally got Bleu score for our model i.e., 0.09026, 0.30043, 0.48601, 0.5481221 and BLEU score is a metric used to evaluate the quality of machine-translated text against one or more reference translations. In our case, the BLEU score of your chest x-ray project model is not very high. A BLEU score of 1.0 is considered a perfect match, and a higher score indicates a better match between the generated text and the reference text. The BLEU-1 score of your model is relatively low, indicating that the model struggles to generate accurate individual words. The higher BLEU-2, BLEU-3, and BLEU-4 scores suggest that the model performs better at generating longer phrases and sentences, but still not satisfactory. But it can be improved easily by increasing the epochs and layer of our model as in the lack of resources to run that many epochs and layers and we can use data augmentation techniques to generate better image captions and we got these results which is satisfactory but not that great.

#### 4. CONCLUSION

The Chest X-ray Image Caption Generator project aims to automate the process of generating a textual description of chest X-ray images. The successful implementation of the deep learning model, coupled with the data pre-processing techniques used in the project, can serve as a baseline for future research in the field. This project has potential applications in the healthcare industry, where radiologists and physicians can benefit from an automated report generation system. The proposed Chest X-ray Image Caption Generator has demonstrated promising results in generating accurate and meaningful captions for radiological images. This tool can assist medical professionals in quickly interpreting and communicating the findings of Chest X-ray images, which can lead to better and faster diagnoses.

#### 5. FUTURE WORK

The Chest X-ray Image Caption Generator can be enhanced firstly, by more extensive data augmentation techniques to improve model generalization and robustness. This can include adding noise or applying deformations to the images during training. Secondly, integrating external knowledge sources such as clinical reports or electronic health records can potentially improve the accuracy and specificity of generated captions. Thirdly, incorporating medical ontologies into the model to improve the quality and accuracy of the generated captions. This would allow the model to better understand and interpret the medical terminology used in X-ray reports and generate more accurate and informative captions. Finally, developing a web-based or mobile application to facilitate the use of the model by radiologists and other healthcare professionals can be an interesting avenue

for future work. This would enable them to generate captions for chest X-ray images more efficiently and accurately, thus saving time and improving patient care.

## **6. LIMITATIONS**

- **Limited dataset:**

The quality and quantity of data are crucial factors that determine the accuracy and effectiveness of a machine learning model. The dataset used in this project may not represent all possible variations and abnormalities present in chest X-ray images, and diversity to produce highly accurate and generalized results. There is a need for more extensive and diverse datasets to enhance the model's performance.

- **Lack of domain-specific knowledge:**

Although the deep learning models used in this project can automatically learn features from the data, they may not capture the full extent of domain-specific knowledge required for accurate report generation. Incorporating additional medical knowledge, such as clinical findings and patient history, may improve the accuracy of the generated reports.

- **Interpretability:**

Deep learning models are often considered black boxes, which means that it is challenging to understand how the model makes decisions. The generated reports may not be easily interpretable by medical professionals, which can limit their usefulness in clinical settings.

- **Limited evaluation metrics:**

The project used standard evaluation metrics such as BLEU score to evaluate the quality of generated reports. However, these metrics may not fully capture the semantic and grammatical accuracy of the generated reports, and alternative evaluation metrics that consider the clinical relevance of the generated reports may be more suitable.

- **Limited generalization:**

The model was trained and evaluated on a specific dataset from a single source. Its ability to generalize to other datasets or domains may be limited, which would require retraining or fine-tuning the model for different datasets.

- **Ethical considerations:**

The use of machine learning algorithms in healthcare raises ethical concerns related to privacy, bias, and accountability. The model may inadvertently perpetuate bias if the training data is not diverse enough, and it may not be clear who is responsible for the accuracy and safety of the generated reports. Further research and development of ethical guidelines for the use of machine learning in healthcare will be essential to mitigate these concerns.

- **Technical limitations:**

The model used in this project was based on deep learning techniques, which require significant computational resources and may take a long time to train. In addition, the model may require specialized hardware such as GPUs, which may not be readily available to all researchers or healthcare organizations.



## REFERENCES

1. Liao, R., Gao, Y., Jiang, T., & Xu, K. (2017). On the Automatic Generation of Medical Imaging Reports. arXiv preprint arXiv:1711.08195.
2. Motamed S, Rogalla P, Khalvati F. Data augmentation using Generative Adversarial Networks (GANs) for GAN-based detection of Pneumonia and COVID-19 in chest X-ray images. *Inform Med Unlocked*. 2021;27:100779. doi: 10.1016/j.imu.2021.100779. Epub 2021 Nov 22. PMID: 34841040; PMCID: PMC8607740.
3. K. P. Panda, P. R. Bana, O. Kiselychnyk, J. Wang and G. Panda, "A Single-Source Switched-Capacitor-Based Step-Up Multilevel Inverter with Reduced Components," in *IEEE Transactions on Industry Applications*, vol. 57, no. 4, pp. 3801-3811, July-Aug. 2021, doi: 10.1109/TIA.2021.3068076.
4. A. B. Amjoud and M. Amrouch, "Automatic Generation of Chest X-ray Reports Using a Transformer-based Deep Learning Model," 2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS), Fez, Morocco, 2021, pp. 1-5, doi: 10.1109/ICDS53782.2021.9626725.
5. S. Khoshabi Nobar, M. H. Ahmed, Y. Morgan and S. Mahmoud, "Joint Channel Assignment and Occupancy Time Optimization in Frame-Based Listen-Before-Talk," in *IEEE Communications Letters*, vol. 24, no. 3, pp. 695-699, March 2020, doi: 10.1109/LCOMM.2019.2959525.