

# IMAGE PARAGRAPH CAPTIONING USING DEEP LEARNING AND NLP TECHNIQUES

Harihar Panda (umds20003)

2020-2022

Under the guidance of Prof. Arif Ahmed Sekh

## 1 Introduction

Image caption generator is a task that involves computer vision and natural language processing concepts to recognize the context of an image and describe them in a natural language like English.

### 1.1 Problem Statement

The main objective of the project is to describe the visual content in the form of text using image paragraph captioning by generating the textual description for the given input image using deep learning and Natural language processing techniques.

### 1.2 Approach to the problem statement

In this Python project, we will be implementing the caption generator using CNN (Convolutional Neural Networks) and LSTM (Long short term memory). The image features will be extracted from Xception which is a CNN model trained on the imagenet dataset and then we feed the features into the LSTM model which will be responsible for generating the image captions.

### 1.3 Understanding the Dataset

For the image caption generator, we will be using the Flickr8K dataset. There are also other big datasets like Flickr30K and MSCOCO dataset but it can take weeks just to train the network so we will be using a small Flickr8k dataset.

## 2 Relevance of AI-powered Image Captioning

The AI-powered image captioning model is an automated tool that generates concise and meaningful captions for prodigious volumes of images efficiently.

The model employs techniques from computer vision and Natural Language Processing (NLP) to extract comprehensive textual information about the given images.

## **2.1 Recommendations in Editing Applications**

The image captioning model automates and accelerates the close captioning process for digital content production, editing, delivery, and archival. Well-trained models replace manual efforts for generating quality captions for images as well as videos.

## **2.2 Assistance for Visually Impaired**

The advent of machine learning solutions like image captioning is a boon for visually impaired people who are unable to comprehend visuals. With AI-powered image caption generator, image descriptions can be read out to visually impaired, enabling them to get a better sense of their surroundings.

## **2.3 Media and Publishing Houses**

The media and public relations industry circulate tens of thousands of visual data across borders in the form of newsletters, emails, etc. The image captioning model accelerates subtitle creation and enables executives to focus on more important tasks.

## **2.4 Social Media Posts**

For social media, artificial intelligence is moving from discussion rooms to underlying mechanisms for identifying and describing terabytes of media files. It enables community administrators to monitor interactions and analysts to formulate business strategies.

# **3 Literature Survey**

There are many methods which are used for image paragraph captioning. These methods aim to generate simple sentences for an image.

## **3.1 Sequence Level Training with Neural Network**

It generates a syntactically and semantically correct sequence of consecutive words to form a sentence. Many natural language processing applications use language models to generate text. These models are typically trained to predict the next word in a sequence, given the previous words and some context such as an image. However, at test time the model is expected to generate the entire sequence from scratch. This discrepancy makes generation brittle, as errors may accumulate along the way. We address this issue by proposing a novel sequence

level training algorithm that directly optimizes the metric used at test time, such as BLEU or ROUGE. On three different tasks, our approach outperforms several strong baselines for greedy generation. The method is also competitive when these baselines employ beam search, while being several times faster. Lstm, Data As Demonstrated to end backpropagation are used here.

### **3.2 Bottom up and top down for image captioning and visual answering question**

Problems combining image and language understanding such as image captioning and visual question answering (VQA) continue to inspire considerable research at the boundary of computer vision and natural language processing. In both these tasks it is often necessary to perform some fine-grained visual processing, or even multiple steps of reasoning to generate high quality outputs. These mechanisms improve performance by learning to focus on the regions of the image that are salient and are currently based on deep neural network architectures. Top-down attention mechanism, Faster R-CNN Lstm are used to achieve this.

### **3.3 Self-Critical Sequence Training For Image Captioning**

In this paper we consider the problem of optimizing image captioning systems using reinforcement learning, and show that by carefully optimizing our systems using the test metrics of the MSCOCO task, significant gains in performance can be realized. Our systems are built using a new optimization approach that we call self-critical sequence training (SCST).

### **3.4 Deep Visual Semantic Alignments For Generating Image Descriptions**

We present a model that generates natural language descriptions of images and their regions. Our approach leverages datasets of images and their sentence descriptions to learn about the inter-modal correspondences between language and visual data. Our alignment model is based on a novel combination of Convolutional Neural Networks over image regions, bidirectional Recurrent Neural Networks over sentences, and a structured objective that aligns the two modalities through a multimodal embedding. This model provides state of the art performance on image-sentence ranking experiments. This describes a Multimodal RNN architecture that generates descriptions of visual data.

### **3.5 Recurrent Topic Transition GAN For Visual Paragraph Generation**

A natural image usually conveys rich semantic content and can be viewed from different angles. Existing image description methods are largely restricted by

smallsets of biased visual paragraph annotations, and fail to cover rich underlying semantics. This paper investigates a semi-supervised paragraph generative framework that is able to synthesize diverse and semantically coherent paragraph descriptions by reasoning over local semantic regions and exploiting linguistic knowledge. Sentence Discriminator, Topic Transition Discriminator, Adversarial Training Scheme methods are used for desired results.

## 4 What is CNN ?

Convolutional Neural networks are specialized deep neural networks which can process the data that has input shape like a 2D matrix. Images are easily represented as a 2D matrix and CNN is very useful in working with images. CNN is basically used for image classifications and identifying if an image is a bird, a plane or Superman, etc. It scans images from left to right and top to bottom to pull out important features from the image and combines the feature to classify images. It can handle the images that have been translated, rotated, scaled and changes in perspective.

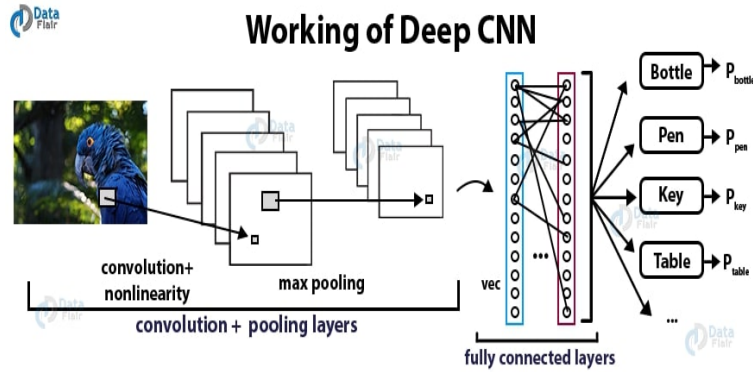


Figure 1: working-of-Deep-CNN-Python-project

## 5 What is LSTM ?

LSTM stands for Long short term memory, they are a type of RNN (recurrent neural network) which is well suited for sequence prediction problems. Based on the previous text, we can predict what the next word will be. It has proven itself effective from the traditional RNN by overcoming the limitations of RNN which had short term memory. LSTM can carry out relevant information throughout the processing of inputs and with a forget gate, it discards non-relevant information.

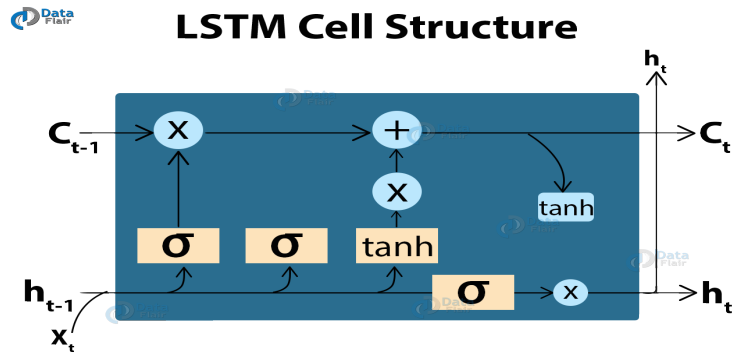


Figure 2: LSTM Cell Structure

## 6 Image Caption Generator Model

So, to make our image caption generator model, we will be merging these architectures. It is also called a CNN-RNN model.

- CNN is used for extracting features from the image. We will use the pre-trained model Xception.
- LSTM will use the information from CNN to help generate a description of the image.

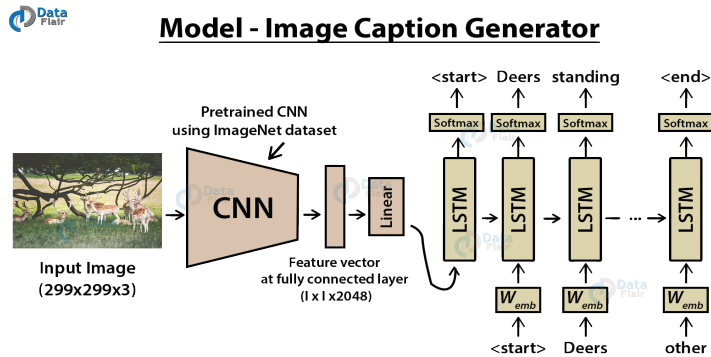


Figure 3: LSTM Cell Structure

## 7 Project Plan

We are using the Flickr8kDataset. The dataset contains two directories:

- Flickr8kDataset: Contains 8092 photographs in JPEG format.
- Flickr8ktext: Contains a number of files containing different sources of
  - descriptions for the photographs. Flickr8ktext folder contains file - Flickr8k.token which is the main file of our dataset that contains image
  - name and their respective captions separated by newline.

The image dataset is divided into 6000 images for training, 1000 images for validation and 1000 images for testing.

Here, we will break down the module into following sections for better understanding:

- Preprocessing of Image
- Creating the vocabulary for the image
- Train the set
- Evaluating the model
- Testing on individual image

## 8 Tools and Technologies

### 8.1 Software Requirements

Programming language: PYTHON Operating system: Windows Front end: Html Tools: Anaconda Navigator /TensorFlow

### 8.2 Introduction to Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guidovan Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

### 8.3 Hardware requirements

Processor: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz RAM: 8GB or above Hard disk: 500 GB or above

### 8.4 User Interface

The user have to give an image.

### 8.5 Hardware Interface:

MONITOR: The output is displayed on the monitor screen

## 8.6 Software Interface:

Python is used as a programming language which will support the machine learning algorithm for predicting accuracy.

## 9 Implementation

1. First, we import all the necessary packages :-

```
import string
import numpy as np
from PIL import Image
import os
from pickle import dump, load
import numpy as np
from keras.applications.xception import Xception, preprocess_input
from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from keras.layers.merge import add
from keras.models import Model, load_model
from keras.layers import Input, Dense, LSTM, Embedding, Dropout
# small library for seeing the progress of loops.
from tqdm.notebook import tqdm
```

2. Getting and performing data cleaning :- The main text file which contains all image captions is Flickr8k.token in our Flickr8ktext folder. Have a look at the file –

```
File Edit Format Run Options Window Help
1000268201_e93b08cb0e.jpg#0 A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_e93b08cb0e.jpg#1 A girl going into a wooden building .
1000268201_e93b08cb0e.jpg#2 A little girl climbing into a wooden playhouse .
1000268201_e93b08cb0e.jpg#3 A little girl climbing the stairs to her playhouse .
1000268201_e93b08cb0e.jpg#4 A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg#0 A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg#1 A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg#2 A black dog and a white dog with brown spots are staring at each other in the
1001773457_577c3a7d70.jpg#3 Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg#4 Two dogs on pavement moving toward each other .
1002674143_lb742ab4b8.jpg#0 A little girl covered in paint sits in front of a painted rainbow with her han
1002674143_lb742ab4b8.jpg#1 A little girl is sitting in front of a large painted rainbow .
1002674143_lb742ab4b8.jpg#2 A small girl in the grass plays with fingerpaints in front of a white canvas w
1002674143_lb742ab4b8.jpg#3 There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_lb742ab4b8.jpg#4 Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg#0 A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg#1 A man lays on the bench to which a white dog is also tied .
1003163366_44323f5815.jpg#2 a man sleeping on a bench outside with a white and black dog sitting next to h
1003163366_44323f5815.jpg#3 A shirtless man lies on a park bench with his dog .
1003163366_44323f5815.jpg#4 man laying on bench holding leash of dog sitting on ground
1007129816_e794419615.jpg#0 A man in an orange hat starrng at something .
1007129816_e794419615.jpg#1 A man wears an orange hat and glasses .
1007129816_e794419615.jpg#2 A man with gauges and glasses is wearing a Blitz hat .
1007129816_e794419615.jpg#3 A man with glasses is wearing a beer can crocheted hat .
1007129816_e794419615.jpg#4 The man with pierced ears is wearing glasses and an orange hat .
1007320043_627395c3d8.jpg#0 A child playing on a rope net .
```

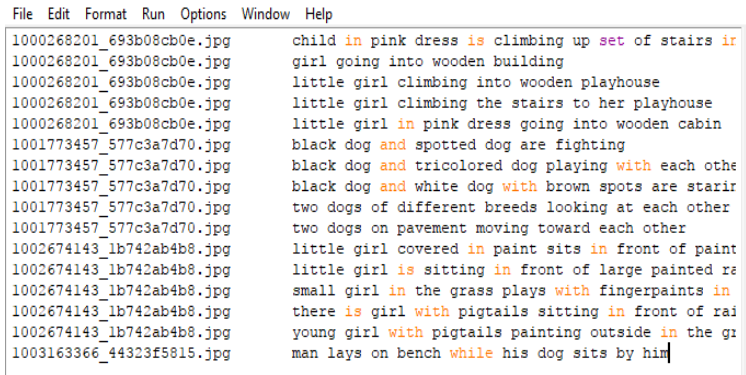
The format of our file is image and caption separated by a new line.

Each image has 5 captions and we can see that (0 to 5) number is assigned for each caption.

We will define 5 functions:

- loaddoc( filename ) – For loading the document file and reading the contents inside the file into a string.

- `allimgcaptions( filename )` – This function will create a descriptions dictionary that maps images with a list of 5 captions. The descriptions dictionary will look something like this:



The screenshot shows a text editor window with a menu bar (File, Edit, Format, Run, Options, Window, Help) and a list of image filenames on the left, each followed by a descriptive caption on the right. The captions are generated by the `allimgcaptions` function and contain various words highlighted in different colors (pink, blue, green, orange, red) to represent different features or classes. The list of image filenames and their corresponding captions is as follows:

Image Filename	Description
1000268201_693b08cb0e.jpg	child in pink dress is climbing up set of stairs in
1000268201_693b08cb0e.jpg	girl going into wooden building
1000268201_693b08cb0e.jpg	little girl climbing into wooden playhouse
1000268201_693b08cb0e.jpg	little girl climbing the stairs to her playhouse
1000268201_693b08cb0e.jpg	little girl in pink dress going into wooden cabin
1001773457_577c3a7d70.jpg	black dog and spotted dog are fighting
1001773457_577c3a7d70.jpg	black dog and tricolored dog playing with each other
1001773457_577c3a7d70.jpg	black dog and white dog with brown spots are staring
1001773457_577c3a7d70.jpg	two dogs of different breeds looking at each other
1001773457_577c3a7d70.jpg	two dogs on pavement moving toward each other
1002674143_lb742ab4b8.jpg	little girl covered in paint sits in front of paint
1002674143_lb742ab4b8.jpg	little girl is sitting in front of large painted re
1002674143_lb742ab4b8.jpg	small girl in the grass plays with fingerpaints in
1002674143_lb742ab4b8.jpg	there is girl with pigtails sitting in front of rail
1002674143_lb742ab4b8.jpg	young girl with pigtails painting outside in the gr
1003163366_44323f5815.jpg	man lays on bench while his dog sits by him

- `cleaningtext( descriptions )` – This function takes all descriptions and performs data cleaning. This is an important step when we work with textual data, according to our goal, we decide what type of cleaning we want to perform on the text. In our case, we will be removing punctuations, converting all text to lowercase and removing words that contain numbers.
  - So, a caption like “A man riding on a three-wheeled wheelchair” will be transformed into “man riding on three wheeled wheelchair”
  - `textvocabulary( descriptions )` – This is a simple function that will separate all the unique words and create the vocabulary from all the descriptions.
  - `savedescriptions( descriptions, filename )` – This function will create a list of all the descriptions that have been preprocessed and store them into a file. We will create a `descriptions.txt` file to store all the captions. It will look something like this:
3. Extracting the feature vector from all images :- This technique is also called transfer learning, we don’t have to do everything on our own, we use the pre-trained model that have been already trained on large datasets and extract the features from these models and use them for our tasks. We are using the Xception model which has been trained on imagenet dataset that had 1000 different classes to classify. We can directly import this model from the `keras.applications` . Make sure you are connected to the internet as the weights get automatically downloaded. Since the Xception model was originally built for imagenet, we will do little changes for integrating with our model. One thing to notice is that the Xception model takes 299\*299\*3 image size as input. We will remove the last classification layer and get the 2048 feature vector.



```

File Edit Format Run Options Window Help
1000268201_693b08cb0e.jpg#0 A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg#1 A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2 A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg#3 A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg#4 A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg#0 A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg#1 A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg#2 A black dog and a white dog with brown spots are staring at each other in the
1001773457_577c3a7d70.jpg#3 Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg#4 Two dogs on pavement moving toward each other .
1002674143_lb742ab4b8.jpg#0 A little girl covered in paint sits in front of a painted rainbow with her han
1002674143_lb742ab4b8.jpg#1 A little girl is sitting in front of a large painted rainbow .
1002674143_lb742ab4b8.jpg#2 A small girl in the grass plays with fingerpaints in front of a white canvas w
1002674143_lb742ab4b8.jpg#3 There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_lb742ab4b8.jpg#4 Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg#0 A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg#1 A man lays on the bench to which a white dog is also tied .
1003163366_44323f5815.jpg#2 a man sleeping on a bench outside with a white and black dog sitting next to h
1003163366_44323f5815.jpg#3 A shirtless man lies on a park bench with his dog .
1003163366_44323f5815.jpg#4 man laying on bench holding leash of dog sitting on ground
1007129816_e794419615.jpg#0 A man in an orange hat staring at something .
1007129816_e794419615.jpg#1 A man wears an orange hat and glasses .
1007129816_e794419615.jpg#2 A man with gauges and glasses is wearing a Blitz hat .
1007129816_e794419615.jpg#3 A man with glasses is wearing a beer can crocheted hat .
1007129816_e794419615.jpg#4 The man with pierced ears is wearing glasses and an orange hat .
1007320049_627395c3d8.jpg#0 A child playing on a rope net .

```

```
model = Xception( includetop=False, pooling='avg' )
```

The function `extractfeatures()` will extract features for all images and we will map image names with their respective feature array. Then we will dump the features dictionary into a “features.p” pickle file.

4. Loading dataset for Training the model :- In our Flickr8ktest folder, we have Flickr8k.trainImages.txt file that contains a list of 6000 image names that we will use for training.

For loading the training dataset, we need more functions:

- `loadphotos( filename )` – This will load the text file in a string and will return the list of image names.
- `loadcleandescriptions( filename, photos )` – This function will create a dictionary that contains captions for each photo from the list of photos. We also append the `jstarti` and `jendi` identifier for each caption. We need this so that our LSTM model can identify the starting and ending of the caption.
- `loadfeatures(photos)` – This function will give us the dictionary for image names and their feature vector which we have previously extracted from the Xception model.

5. Tokenizing the vocabulary :- Computers don’t understand English words, for computers, we will have to represent them with numbers. So, we will map each word of the vocabulary with a unique index value. Keras library provides us with the tokenizer function that we will use to create tokens from our vocabulary and save them to a “tokenizer.p” pickle file.

Our vocabulary contains 7577 words.

We calculate the maximum length of the descriptions. This is important for deciding the model structure parameters. Maxlength of description is 32.

6. Create Data generator :- Let us first see how the input and output of our model will look like. To make this task into a supervised learning task, we have to provide input and output to the model for training. We have to train our model on 6000 images and each image will contain 2048 length feature vector and caption is also represented as numbers. This amount of data for 6000 images is not possible to hold into memory so we will be using a generator method that will yield batches.

The generator will yield the input and output sequence.

For example:

The input to our model is  $[x1, x2]$  and the output will be  $y$ , where  $x1$  is the 2048 feature vector of that image,  $x2$  is the input text sequence and  $y$  is the output text sequence that the model has to predict.

$x1$ (feature vector)	$x2$ (Text sequence)	$y$ (word to predict)
feature	start,	two
feature	start, two	dogs
feature	start, two, dogs	drink
feature	start, two, dogs, drink	water
feature	start, two, dogs, drink, water	end

7. Defining the CNN-RNN model :- To define the structure of the model, we will be using the Keras Model from Functional API. It will consist of three major parts:
- Feature Extractor – The feature extracted from the image has a size of 2048, with a dense layer, we will reduce the dimensions to 256 nodes.
  - Sequence Processor – An embedding layer will handle the textual input, followed by the LSTM layer.
  - Decoder – By merging the output from the above two layers, we will process by the dense layer to make the final prediction. The final layer will contain the number of nodes equal to our vocabulary size.
8. Training the model :- To train the model, we will be using the 6000 training images by generating the input and output sequences in batches and fitting them to the model using `model.fit()` method. We also save the model to our models folder. This will take some time depending on your system capability.

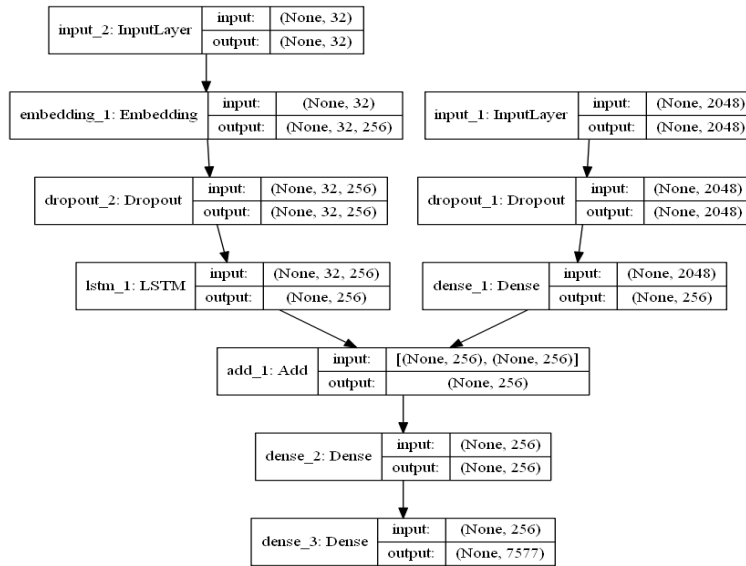
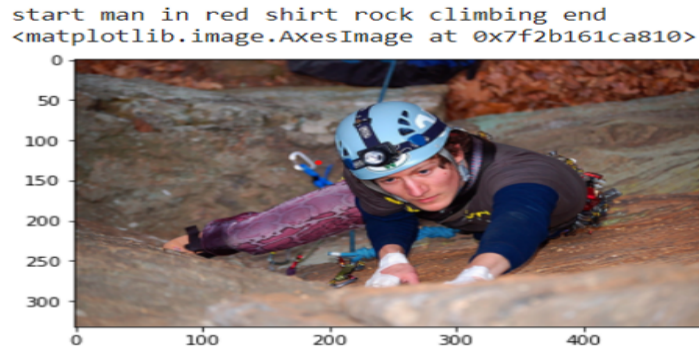


Figure 4: Visual representation of the final model

9. Testing the model :- The model has been trained, now, we will make a separate file `testingcaptiongenerator.py` which will load the model and generate predictions. The predictions contain the max length of index values so we will use the same `tokenizer.p` pickle file to get the words from their index values.
10. Results :-



## 10 EXPERIMENTAL ANALYSIS (BLEU)

BLEU is a quality metric score for MT systems that attempts to measure the correspondence between a machine translation output and a human translation. The central idea behind BLEU is that the closer a machine translation is to a professional human translation, the better it is.

BLEU scores only reflect how a system performs on the specific set of source sentences and the translations selected for the test. As the selected translation for each segment may not be the only correct one, it is often possible to score good translations poorly. As a result, the scores don't always reflect the actual potential performance of a system, especially on content that differs from the specific test material.

### BLEU

- N-gram overlap between machine translation output and reference translation
- Compute precision for n-grams of size 1 to 4
- Add brevity penalty (for too short translations)

$$\text{BLEU} = \min \left( 1, \frac{\text{output-length}}{\text{reference-length}} \right) \left( \prod_{i=1}^4 \text{precision}_i \right)^{\frac{1}{4}}$$

- Typically computed over the entire corpus, not single sentences

### The BLEU Evaluation

- The BLEU metric ranges from 0 to 1
- 1 is very rare: only for perfect match
- The more, the better
- Human translation score 0.3468 against four references and scored 0.2571 against two references
- Table 1: 5 systems against two reference

Table 1: BLEU on 500 sentences

S1	S2	S3	H1	H2
0.0527	0.0829	0.0930	0.1934	0.2571

## 11 CONCLUSION AND FUTURE WORK

In this advanced Python project, we have implemented a CNN-RNN model by building an image caption generator. Some key points to note are that our model depends on the data, so, it cannot predict the words that are out of its

vocabulary. We used a small dataset consisting of 8000 images. For production-level models, we need to train on datasets larger than 100,000 images which can produce better accuracy models.

In the future, image captioning can be improved in three aspects. The first aspect is to enhance the adaptability of the network, so that the model can pay attention to the specific situations and concerns, and generate targeted descriptions according to different situations and concerns. The second aspect is to optimize the evaluation algorithm to evaluate the quality of the output sequence of the model more accurately. The third aspect is to enhance the robustness of the model and avoid the influence of interference characteristics on the model output. We have implemented a CNN-LSTM model for building an Image Caption Generator. A CNN-LSTM architecture has wide-ranging applications which include use cases in Computer Vision and Natural Language Processing domains.