Title

Automated Image Captioning using CNN and RNN

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

ANUJ JHUNJHUNWALA

17BCE2149

Under the guidance of

Prof. Delhi Babu R.

School of Computer Science and Engineering

VIT, Vellore.

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

June, 2021

DECLARATION

I hereby declare that the thesis entitled "Automated Image Captioning using CNN and RNN" submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Prof. Delhi Babu R.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 09 June 2021

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled "Automated Image Captioning using CNN and RNN" submitted by **Anuj Jhunjhunwala (17BCE2149), School of Computer Science and Engineering**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him under my supervision during the period, 01. 12. 2018 to 30.04.2019, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meet the necessary standards for submission.

Place: Vellore

Date: 09 June 2021

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department

Computer Science and Engineering

ACKNOWLEDGEMENTS

I would like to thank the following people for granting me this opportunity to learn and guiding me throughout the project;

Prof. K.S. Sendhil Kumar, Associate Professor Grade 1, SCOPE, who has coordinated the whole capstone project virtually and seamlessly without any kind of obstacles for each and every student.

Prof. Delhi Babu R., SCOPE, for his vital supervision, constant guidance, and encouragement throughout the duration of my project, which helped me learn and improve my project.

At last, I would like to thank all the faculty members of SCOPE, VIT Vellore, who were always there to clear my doubts on any application and made my academic experience fruitful and fun.

Place: Vellore

Date: 09 June 2021 **Anuj Jhunjhunwala**

Executive Summary

Captioning the image means extracting the meaningful description of an image. This is a child's play for a literate human being, but not an easy task for a machine. So, basically, I am bound to search for a solution or a method in which the features of the image can be extracted easily and without any hassle. Further, the solution should be able to extract a comprehensible text from those features of the image.

In this project the aim is to create an automated image captioning model using Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to produce a series of texts that best describe the image. This model is trained on Flickr 8k dataset. The image captioning process will require two steps, first CNN is used to extract features from the given input image and secondly, RNN is used to convert the features into a simple and meaningful language. This project uses the encoder-decoder model, where CNN performs the feature extraction and the output of the encoder is fed to the decoder which processes the classified features into appropriate sentences. The feature extraction will be done by the latest Inception V3 module-50 technology with means of transfer learning so that I can modify the project specific to my application. The language model uses natural language toolkit for simple natural language processing and the architecture used for recurrent neural network is long short-term memory. Finally, I have tried to implement speech generation from the text generated using gTTS library.

List of Figures

List of Tables

List of Abbreviations

LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
NLP	Natural Language Processing
BLEU	Bilingual Evaluation Understudy

Symbols and Notations

x1	<START> vector
y1	Next word
x2	Word vector
xT	Final word
yT	End Token

1. INTRODUCTION

1.1 THEORETICAL BACKGROUND

When we see a picture and our cerebrum can without much of a stretch advise about the picture, yet can a computer determine what the picture is addressing? Computer vision analysts chipped away at this a ton and they thought of it as inconceivable as of recently! With the progression in Deep learning procedures, accessibility of tremendous datasets and computer power, we can fabricate models that can create subtitles for a picture. Captioning the image means extracting the meaningful description of an image. This is a child's play for a literate human being, but not an easy task for a machine. This automated task by the machine would include understanding what is there in the image as well as translating it into natural language. Now using deep learning, it has been proved that it is better of the traditional methods and have shown better results for the problem of automatically generating descriptions from images. Picture Captioning is the way toward producing literary depiction of a picture. It utilizes both NLP and CV to produce the inscriptions. The dataset will be in the structure a bunch of pictures and inscriptions. The dataset comprises of input pictures and their relating yield inscriptions.

1.2 MOTIVATION

According to the statistics, around 1.5% of India’s population comprises of blind people. My project will ultimately help the blind person to observe his surroundings independently. Also, with the advancements in technology and ease of computation of extensive data, it is now possible for us to easily apply deep learning in several projects using our personal workstation.

For a specific photo, a caption can be "A white canine in a green region", or "White canine with earthy colored spots" but it may also be "A canine on grass and some pink blossoms". Unquestionably these subtitles are significant for this picture and there might be some others too.

We should initially see how significant this issue is to genuine situations. How about we see not many applications where an answer for this issue can be valuable.

Self-driving vehicles — Automatic driving is probably the greatest test and on the off chance that we can appropriately subtitle the scene around the vehicle, it can give a lift to oneself driving framework.

Help to the visually impaired — This is the application that I have personally looked upon in this project. So, a blind person would be able to walk freely and independently without any external human support. He will be easy able to know what is going on in his surroundings.

CCTV cameras are wherever today, however alongside survey the world, assuming we can likewise create pertinent inscriptions, we can raise cautions when there is some noxious action going on some place. This could likely assistance lessen some wrongdoing and additionally mishaps.

Image captioning can also help the search engines work more efficiently. When a search engine takes a photo, it makes it difficult for the internal algorithm to process it efficiently. Whereas, if it takes an image as input and converts it into caption, the search would become extremely easy.

1.3 AIM OF THE PROPOSED WORK

Picture captioning is an interesting issue of picture comprehension, and it is made out of two common parts ("look" and "language articulation") which compare to the two most significant fields of man-made consciousness ("computer vision" and "natural language processing"). With the improvement of profound neural organizations and better naming information base, the picture inscribing procedures have grown rapidly. In this project, the picture inscribing approaches and upgrades dependent on profound neural organization are presented, including the attributes of the particular strategies. The early picture captioning approach dependent on profound neural organization is the recovery-based technique. The recovery strategy utilizes a looking through method to track down a proper picture portrayal. The layout-based technique isolates the picture inscribing measure into object location and sentence age. As of late, start to finish learning-based picture subtitling strategy has been checked successful at picture inscribing. The start to finish learning procedures can create more adaptable and familiar sentence. In this project, the picture subtitling technique is used in favor of blind people. According to the statistics, around 1.5% of India’s population comprises of blind people. My project will ultimately help the blind person to observe his surroundings independently. Also, with the advancements in technology and ease of computation of extensive data, it is now possible for us to easily apply deep learning in several projects using our personal workstation. The ultimate goal is to help the blind person to observe his surrounding independently and without any kind of external human support.

1.4 OBJECTIVES OF THE PROPOSED WORK

So, basically, I am bound to search for a solution or a method in which the features of the image can be extracted easily and without any hassle. Further, the solution should be able to extract a comprehensible text from those features of the image. This will make extensive use of computer vision and NLP. Further, the generated text will be converted to speech so that the blind can hear the description of what is there in his surroundings. So, the main objectives to achieve the goal are:

- Extract features from the captured image.
- Generate a caption for the extracted features
- Convert the caption (text) to speech so that the user can hear

2. LITERATURE SURVEY

2.1 SURVEY OF EXISTING MODELS/WORK

1.

Title Image Captioning by Incorporating Affective Concepts Learned from Both Visual and Textual Components
Author/s Jufeng Yanga, Yan Suna, Jie Lianga, Bo Rena, Shang-Hong Laib
Date of Publication August, 2018
Dataset Used SentiCap dataset
Application and Methodology This paper proposes a model to upgrade the feeling expressibility of text portrayals by creating suitable passionate ideas to sentences, which is determined from huge scope visual and printed storehouses by learning both substance and semantic modules. The visual and text based portrayals are extricated, trailed by consolidating the inactive codes of the two segments into a low-dimensional subspace. From that point onward, the joined idle portrayals are decoded finally creating the emotional picture subtitles. Roused by the captivated comment from the SentiCap, it tends to be viably used to create sentences for unbiased pictures with very surprising feelings. For every one of the unbiased pictures, it changes their enthusiastic likelihood vectors and produce sentences with various slants.
Conclusion To measure the consequences of passionate portrayals, both programmed and human sourcing metric were embraced for assessment. In view of the age aftereffects of EmoCap, the need of passionate depictions is examined. An appropriate edge is extricated for the assessment of feelings and afterward, the proposed strategy is applied to produce good/negative depictions for nonpartisan pictures. Investigations show that the EmoCap can create sentences with wanted feelings for unbiased pictures, which gives inconsistent comprehension to a given picture.

2.

Title A Neural Image Captioning Model with Caption-to-Images Semantic Constructor
Author/s Jinsong Su, Jialong Tang, Ziyao Lu, Xianpei Han, Haiying Zhang
Date of Publication August, 2019
Dataset Used Microsoft COCO Caption Data Set, COCO online testing server
Application and Methodology This paper proposed a technique in which the traditional CNN-LSTM picture subtitling model is stretched out by acquainting a cation-with pictures semantic re-constructor, which helps in recreating the semantic portrayals of the info picture and its comparative pictures from covered up conditions of the decoder. The re-development score of the semantic re-constructor is joined with the probability to refine model preparing. Henceforth thusly, the semantics of the information picture can be all the more viably moved to the decoder and be completely misused to create better subtitles. The re-development score is additionally utilized alongside the log probability to choose better inscription through re-positioning.
Conclusion This paper acquaints an augmentation with the standard CNN-LSTM picture inscribing model by acquainting an inscription with picture semantic re-constructor that empowers the model to all the while abuse bidirectional conditions for subtitle age. The utilization of semantic re-constructor not just refines model preparing by expanding target work with re-development score yet in addition benefits model testing with the use of re-positioning. Exploratory outcomes demonstrate that the proposed model fundamentally improves the nature of the produced subtitles and beats a regular picture inscribing model, LSTM-A5.

3.

Title

Image Captioning via Semantic Element Embedding

Author/s

Xiaodan Zhang, Shengfeng He, Xinhang Song, Rynson W. H. Lau, Jianbin Jiao, Qixiang Ye

Date of Publication

July, 2019

Dataset Used

Microsoft COCO Caption Data Set, Flickr8k, Flickr30k

Application and Methodology

With the momentous advancement of profound element learning, picture inscribing is an exceptionally respected point in the fields of PC vision and neuro registering networks. This paper proposes a philosophy to advance the semantic portrayals of pictures and update the language model by proposing semantic component inserting. For the semantic component revelation, an item recognition module is utilized to foresee the districts of the picture, and an inscribing model, Long Short-Term Memory (LSTM) is utilized to create neighborhood portrayals for these locales. The anticipated portrayals and classifications are utilized to create the semantic element, which contains point by point data as well as offers a word space with depictions, and along these lines connects the methodology hole between visual pictures and semantic inscriptions. Further, the CNN highlight is incorporated alongside the semantic component into the proposed Element Embedding LSTM (EE-LSTM) model to foresee a language depiction.

Conclusion

Examination results on the comparing dataset exhibits that the proposed approach beats traditional inscription techniques and is adaptable to consolidate with standard models to accomplish unrivaled execution. The proposed technique outflanks the LSTM strategy with a huge edge, showing the adequacy of item component inserting for picture understanding.

4.

Title

Multilayer Dense Attention Model for Image Caption

Author/s

ERIC KE WANG , XUN ZHANG, FAN WANG , TSU-YANG WU , AND CHIEN-MING CHEN

Date of Publication

May 20, 2019

Dataset Used

Microsoft COCO Caption Data Set, Flickr dataset, AI challenge Chinese dataset.

Application and Methodology

The picture subtitle is an innovation that empowers us to comprehend the substance and create clear content, of pictures utilizing machines. Numerous functional applications need the picture inscription innovation. In the wake of snapping a photo, clients can utilize this innovation to coordinate with the suitable content, which can supplant the client's manual loading up with text. Comparative undertakings incorporate video subtitle, where the info is a video and the yield are its portrayal. This paper proposes a multi-facet thick consideration model for picture subtitle consolidating the consideration LSTM with phonetic LSTM.A quicker repetitive convolutional neural organizations (Faster R-CNN) is utilized to extricate picture highlights as the coding layer, the long momentary memory (LSTM)- go to is utilized to unravel the multi-facet thick consideration model, and the portrayal text is created. The model boundaries are streamlined utilizing methodology inclination enhancement in support learning. The exploratory outcomes in the field of general pictures approve the model's acceptable capacity to get pictures and producing text.

Conclusion

The creator proposes a multi-facet thick consideration model for picture subtitling task. A quicker RCNN is utilized to separate the picture highlights as the coding layer, LSTM-Attend is utilized to interpret the multi-facet thick consideration model, and the portrayal text is produced. The trial results show that our model has a decent capacity of picture comprehension and text age. For future work the proposed approach can be stretched out for task-explicit subtitle age.

5.

Title

Integration of textual cues for fine-grained image captioning using deep CNN and LSTM

Author/s

Neeraj Gupta, Anand Singh Jalal

Date of Publication

9 October 2019

Dataset Used

Data set used are Flickr dataset 8k & 30k.

Application and Methodology

The programmed portrayal of a characteristic scene is a significant quality in man-made consciousness that joins PC vision and normal language handling. Subtitle age is a difficult assignment in scene understanding. In this paper, the creator has proposed a model which consolidates a profound convolutional neural organization and long transient memory to support the precision of picture subtitling by combining text include accessible in a picture with the visual highlights separated in cutting edge techniques. The proposed model comprises of three modules. The principal module of the model is an extraction of picture visual highlights utilizing visual math bunch (VGG) net, a profound convolution neural organization (CNN) model. The subsequent module is word spotting, in which a book saliency model is utilized to extricate the remarkable content as a component from a picture. The third module is the combination of visual highlights and printed highlights utilizing a long momentary memory (LSTM) which creates the last subtitle of a picture.

Conclusion

The creator examined those visual highlights alone are lacking for producing an inscription for a scene. The printed data separated as a striking content from a picture is used for producing more significant subtitle for that picture. The element combination is performed by infusing the content component vector into the underlying layer of LSTM. Later on, the proposed approach can be reached out for task-explicit subtitle age.

6.

Title

A Neural Network Framework to Generate Caption from Images

Author/s

Ayan Ghosh, Debarati Dutta and Tiyasa Moitra

Date of Publication

17 July 2019

Dataset Used

Data set used are Flickr dataset 8k

Application and Methodology

In this paper, the creator zeroed in on one of the visual acknowledgment aspects of PC vision, i.e., picture subtitling. The point of the paper was to prepare convolutional neural organizations (CNN) with a few many hyper-boundaries and apply it on an enormous dataset of pictures (ImageNet) and consolidate the consequences of this picture classifier with a repetitive neural organization to make a subtitle for the characterized picture. In this paper, the creator methodicallly examined a profound neural organizations based picture inscription age strategy. With a picture as the information, the technique can yield an English sentence portraying the substance in the picture. The creator dissected three segments of the strategy: convolutional neural organization (CNN), repetitive neural organization (RNN) and sentence age. In this paper, the creator accomplished a BLEU score of 56 on the Flickr8k dataset while the cutting edge results lay at 66 on the dataset.

Conclusion

Examination results on the comparing dataset exhibits that the proposed approach beats traditional inscription techniques and is adaptable to consolidate with standard models to accomplish unrivaled execution. The proposed technique outflanks the LSTM strategy with a huge edge, showing the adequacy of item component inserting for picture understanding. The author has also used the concept of BLEU to check the accuracy score.

7.

Title Neural Image Captioning
Author/s Lakshay Sharma, Elaina Tan
Date of Publication July, 2019
Dataset Used Microsoft COCO (Common Objects in Context) (Lin et al. [2014]) Captioning Dataset (2014)
Application and Methodology As of late, the greatest advances in significant Computer Vision errands, for example, object acknowledgment, manually written digit distinguishing proof, facial acknowledgment, and numerous others., have all gotten using Convolutional Neural Networks (CNNs). These individual advances have enormously profited assignments even at the convergence of NLP and Computer Vision, and motivated by this achievement, the creator has concentrated some current neural picture inscribing models that have demonstrated to function admirably. In this work, they concentrate some current subtitling models that give close to best in class exhibitions, and attempt to upgrade one such model. The creator utilizes a straightforward picture inscribing model that utilizes a CNN, a LSTM, and the shaft search1 calculation, and study its exhibition dependent on different subjective and quantitative measurements. This paper has been composed exclusively with the end goal of study and investigation.
Conclusion The creator has proposed two model structures: initial one is an upgrade on a present status of-the-workmanship execution and the other is a straightforward CNN/LSTM model that showed the capacity of a conventional model to produce lucid picture inscriptions. Trials were performed, and execution varieties were evaluated with varieties in pillar search, input highlights created, and the quantity of LSTM layers. This was trailed by a k-closest neighbors power/generalizability check. The blunders created by the model were arranged into authoritative classifications.

8.

Title

Image Captioning using Google’s Inception-resnetv2 and Recurrent Neural Network

Author/s

Yajurv Bhatia, Aman Bajpayee, Deepanshu Raghuvanshi, Himanshu Mittal

Date of Publication

August, 2019

Dataset Used

GloVe dataset for sentence semantics, set of google images

Application and Methodology

Given a photo as info, this paper takes care of the issue of encountering a conceivable inscription of the photo. The model finds out about the relationships among's language and pictures from the gave informational collection of named pictures. It proposes a completely programmed approach through a blend of Convolutional Neural Network and a Recurrent Neural Network. The encoder is liable for understanding the highlights present in the inputted picture that are valuable in the end delivering a clarification. The model efforts to create inscriptions for both the articles and the districts present in the picture. Regarding language as a major mark space, the task produces forecasts for the different locales of the picture and afterward join them together.

Conclusion

When the model was prepared over a bunch of shaded photographs, it was tried out on the organization by taking care of it for certain pictures. The outcomes were blended. From the outcomes, it was seen that albeit a few outcomes were very acceptable, some created subtitles will in general be syntactically mistaken. For instance, one of the photographs utilized predicts "the man is climbing rock climbing". The organization neglects to perceive the action word thing connection and consequently predicts an action word pointlessly. Generally, shortsighted inscriptions that portray the pictures to some degree enigmatically are delivered without mistakes, however when the organization attempts to investigate more complexities in the picture it neglects to depict them appropriately.

2.2 SUMMARY OF THE LITERATURE SURVEY

The research papers that I have surveyed contains the basis of my project implementation. The deep learning models that use CNN and RNN are itself used in the projects according to the respective articles. Most of the papers have used the same flickr8k dataset to get the results. Some have also used real time images for testing but have got mixed results. Some papers have also compared the performance using various architectures on the dataset. In addition to the previous work done on this domain and application, I am also generating the text-to-speech audio which will make the user independent and he can move around observing the environment without any help.

3. Overview of the Proposed System

3.1 INTRODUCTION AND RELATED CONCEPTS

Picture inscribing is a piece of cake for any person yet it is anything but a simple errand to compose a PC program that accepts a picture as info and produces an applicable inscription as yield. The fundamental objective here is to assembled CNN-RNN to make a programmed picture subtitling model that takes in a picture as info and yields a grouping of text that depicts the picture lastly, convert that text to discourse so that the outwardly debilitated can tune in to Image inscriptions are utilized in an assortment of uses. For example:

- It can be used for blind people to make them aware about their surroundings and help them feel independent.
- To make the content of a web page more readable, it is generally advised to label the images. Here, the image captioning model will be a very useful tool.
- Now a days, as educational videos have become very important in a students life, it will be really helpful if they are captioned in real time.

3.2 ARCHITECTURE OF THE PROPOSED SYSTEM

INCEPTION V3 ARCHITECTURE

Pictures are only information (X) to our model. We should know the importance of a vector which is the input to the model. We need to change over each picture into a fixed estimated vector which would then be able to be taken care of as contribution to the neural organization. For this reason, I have chosen to use the InceptionV3 model (CNN) made by the research team of Google. One thousand different photos were taken from the imagenet dataset and picture order was performed. Be that as it may, our motivation here isn't to order the picture however sort out length educational vector for each picture. This interaction is called programmed highlight designing. Hence the softmax layer at the last is dropped off and a vector measuring two thousand and forty eight length is extracted which denotes the bottleneck features:

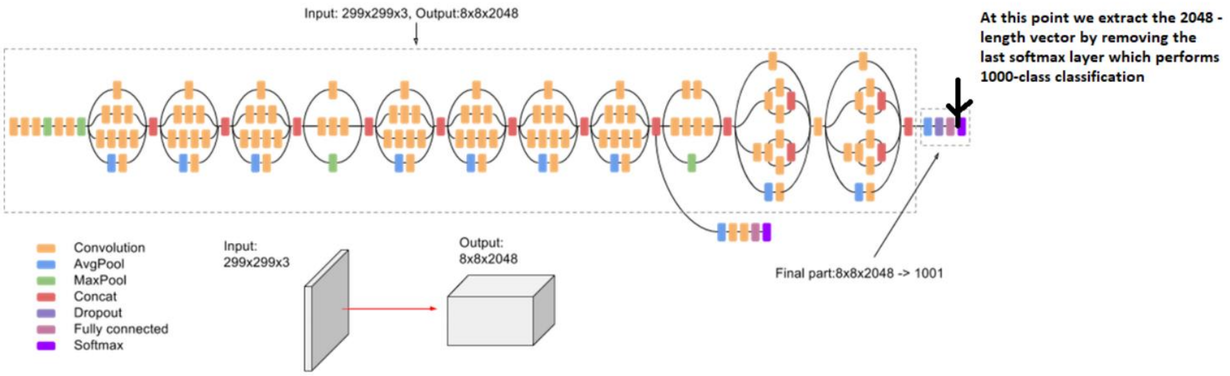

At this point we extract the 2048 - length vector by removing the last softmax layer which performs 1000-class classification

Figure 1: InceptionV3 architecture

I have saved all the narrowing train highlights in a key-value based data structure in python and saved it on the disk utilizing “Pickle File”, to be specific "encoded_train_images.pkl" whose keys is associated with picture title and qualities measuring two thousand and forty eight length include vector. This interaction took around three to four hours since I don't have a very good quality laptop. Likewise, to avoid any kind of time delay in subsequent runs, these records are saved as "encoded_test_images.pkl".

LSTM ARCHITECTURE

The working of LSTM can be imagined by understanding the working of a news channel's group covering a homicide story. Presently, a report is worked around realities, proof and explanations of numerous individuals. At whatever point another occasion happens you make both of the three strides.

Suppose, we were accepting that the homicide was finished by 'harming' the person in question, however the examination report that just came in said that the reason for death was 'an effect on the head'. Being a piece of this news group what do you do? You quickly fail to remember the past reason for death and all accounts that were woven around this reality.

What, if an altogether new suspect is brought into the image. An individual who had feelings of resentment with the person in question and could be the killer? You input this data into your news channel, isn't that so?

Presently all these messed up snippets of data can't be served on traditional press. Thus, after a specific time span, you need to sum up this data and yield the pertinent things to your crowd. Possibly as "XYZ ends up being the excellent suspect.".

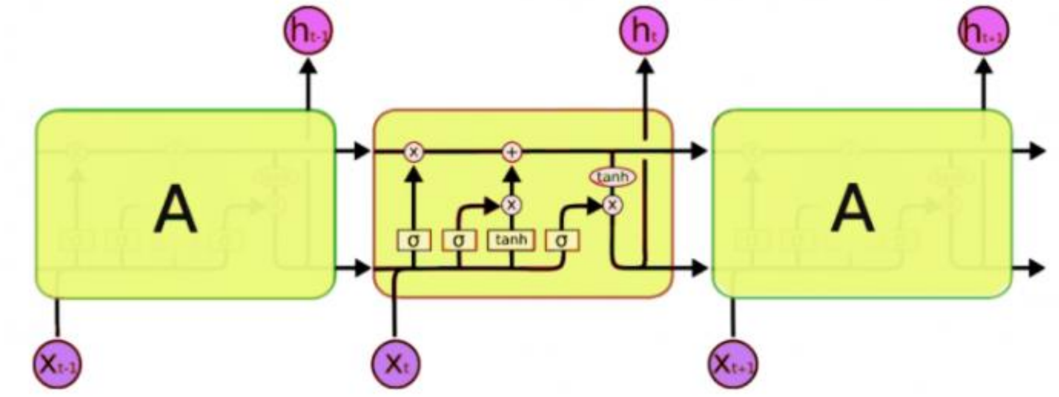


Figure 2: LSTM Architecture

Presently, this is no place near the worked on form which we saw previously, yet let me walk you through it. An average LSTM network is contained distinctive memory blocks called cells (the square shapes that we find in the picture). There are two expresses that are being moved to the following cell; the cell state and the secret state. The memory blocks are liable for recalling things and controls to this memory is done through three significant systems, called entryways. Every one of them is being talked about underneath.

3.3 MODEL ARCHITECTURE

Since we know that the input has got two sub divisions, that is, the input image and the partial descriptions for those images. We will not be able to make good use of the sequential API which is proposed by the Keras Library. Hence, the "functional API" would allow us to make use of the un-splitted models. Following is the short design which contains the significant level sub-modules:

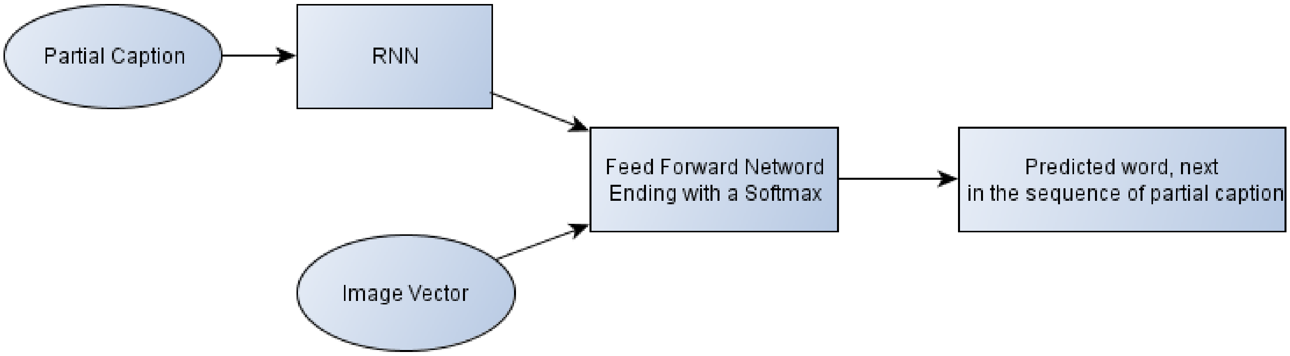
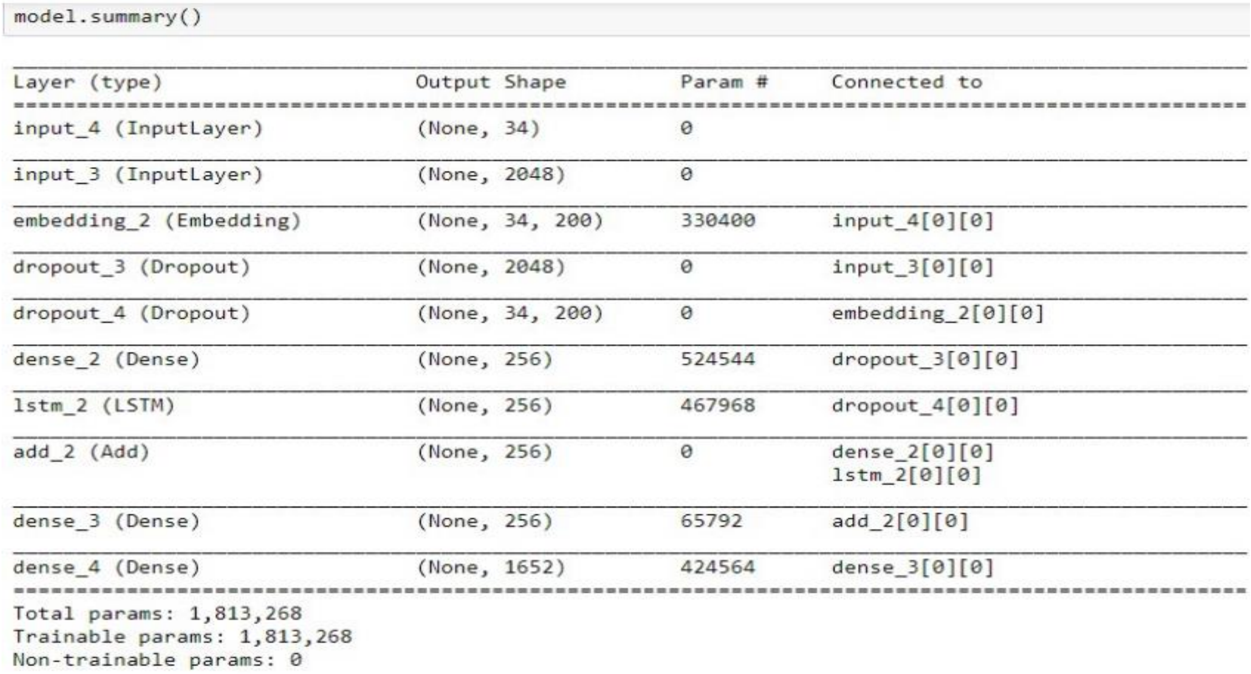


Figure 3: Model Architecture



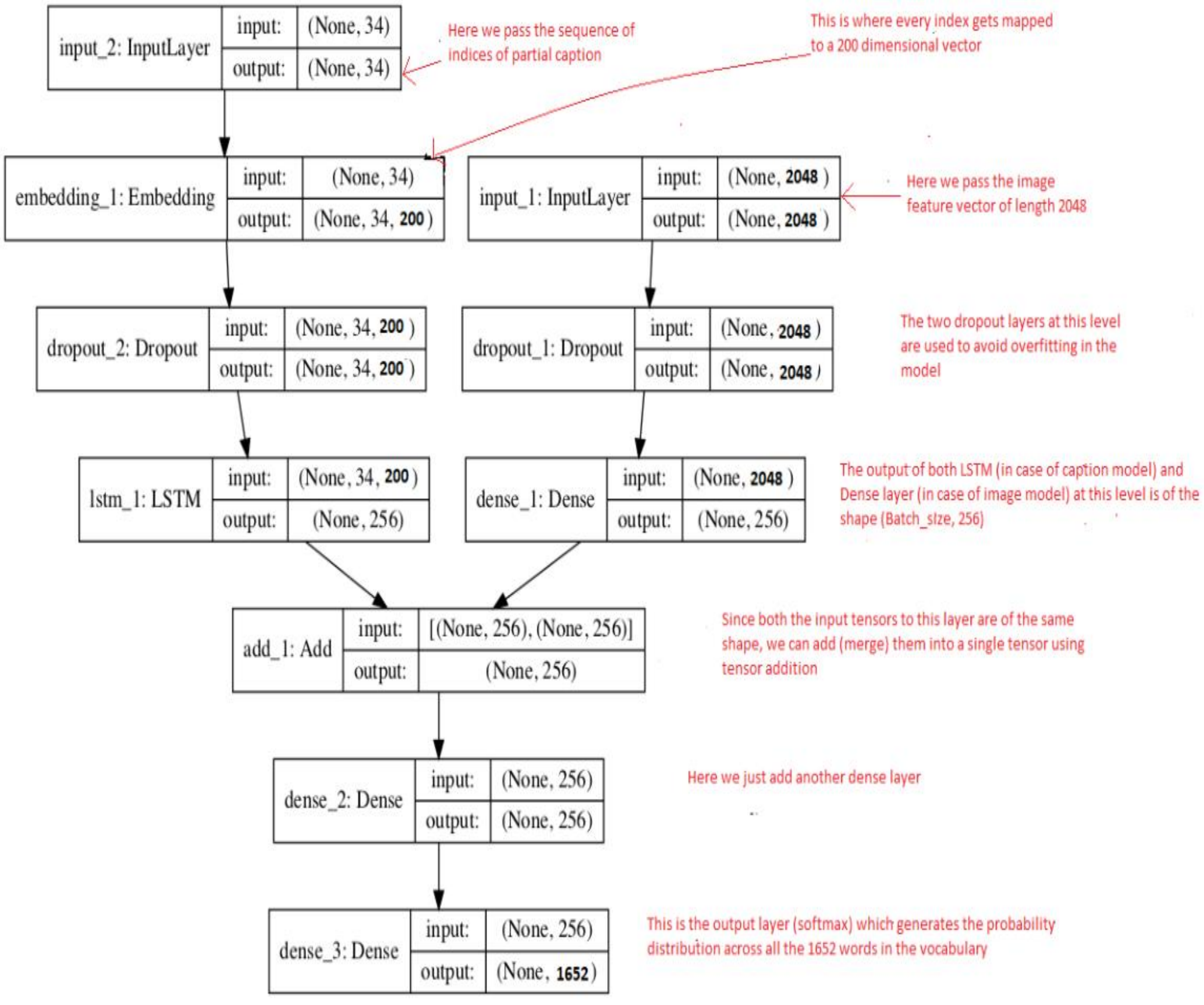
```
model.summary()
```

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_4 (InputLayer)             (None, 34)            0
____________________________________________________________________________________________________
input_3 (InputLayer)             (None, 2048)          0
____________________________________________________________________________________________________
embedding_2 (Embedding)          (None, 34, 200)       330400      input_4[0][0]
____________________________________________________________________________________________________
dropout_3 (Dropout)              (None, 2048)          0           input_3[0][0]
____________________________________________________________________________________________________
dropout_4 (Dropout)              (None, 34, 200)       0           embedding_2[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 256)           524544      dropout_3[0][0]
____________________________________________________________________________________________________
lstm_2 (LSTM)                    (None, 256)           467968      dropout_4[0][0]
____________________________________________________________________________________________________
add_2 (Add)                      (None, 256)           0           dense_2[0][0]
                                                                   lstm_2[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 256)           65792       add_2[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 1652)          424564      dense_3[0][0]
====================================================================================================
Total params: 1,813,268
Trainable params: 1,813,268
Non-trainable params: 0
```

Summary of the parameters in the model

Table 1: Different parameter of the model and its summery

The following flowchart give the description of how the two inputs get merged and how it is processed further:



Here we pass the sequence of indices of partial caption

This is where every index gets mapped to a 200 dimensional vector

Here we pass the image feature vector of length 2048

The two dropout layers at this level are used to avoid overfitting in the model

The output of both LSTM (in case of caption model) and Dense layer (in case of image model) at this level is of the shape (Batch_size, 256)

Since both the input tensors to this layer are of the same shape, we can add (merge) them into a single tensor using tensor addition

Here we just add another dense layer

This is the output layer (softmax) which generates the probability distribution across all the 1652 words in the vocabulary

Flowchart of the architecture

Figure 4: Flowchart

The highlighted words are for understanding the information planning part of the model engineering. The LSTM layer is only a particular “Recurrent Neural Network” to handle the arrangement input (fractional inscriptions for our situation). At that point we incorporate the installed framework from a pre-prepared “GloVe Model” prior to beginning the preparation interaction which we have recently made. Since we are utilizing a pre-prepared inserting layer, we need to make it unchangeable, prior to preparing the model, with the goal that it doesn't get refreshed when some backtracking is performed. At long last we incorporate the model utilizing the “adam” streamlining agent.

First Input - > “Partial Caption”

Second Input - > “Image highlight vector”

Resultant - > A suitable word, next in the arrangement of halfway subtitle gave in the input_1 (or in likelihood terms we say molded on picture vector and the fractional inscription)

Time Duration: It required almost 4 hours to prepare the model as per my framework arrangement.

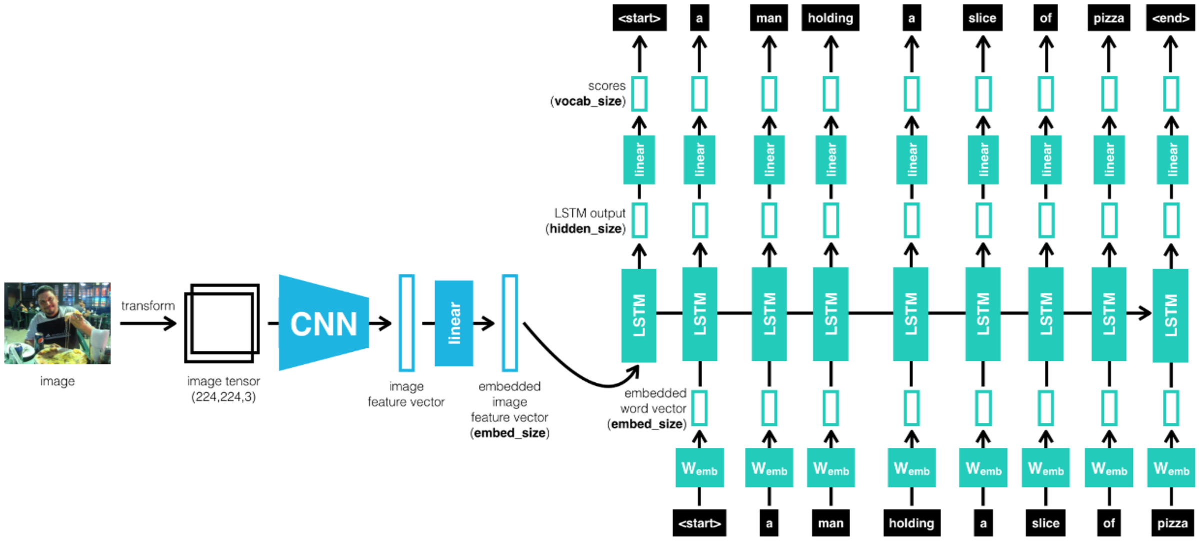


Figure 5: Neural Diagram

Dataset

What is the reason to choose Flickr8k dataset?

The first and foremost reason to choose flickr8k dataset is that it suffices the need of general images which is much needed in my project's application. The blind people miss out on seeing the daily life. This dataset contains all the daily life images that a normal person sees in his general life.

Secondly, it is small in size and contains only eight thousand images. So, it is very easy to train the model over it without the need of a high-end machine.

Third, the data is properly labelled. Every image comes with five different captions.

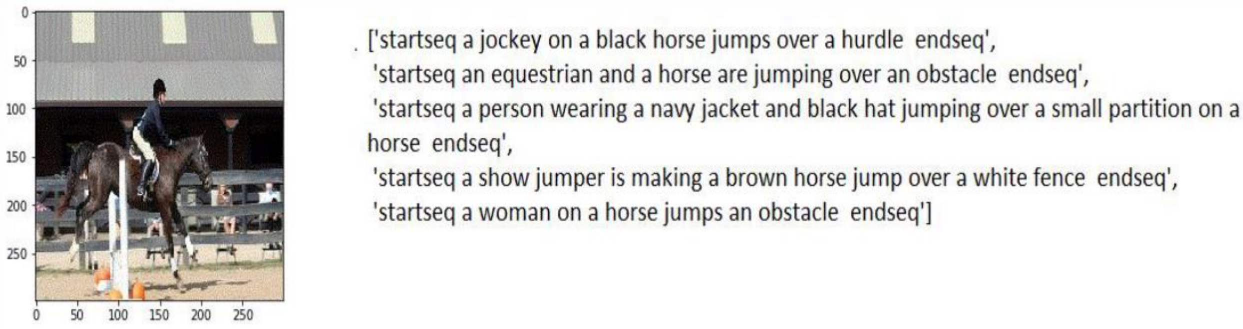
Last but not the least, this dataset is available for free by the author. So, there is no charge to access the database for experiments.

Information pre-preparing and cleaning is a significant piece of the entire model structure measure. Understanding the information assists us with building more precise models.

In the wake of removing compress records, I discovered underneath envelopes

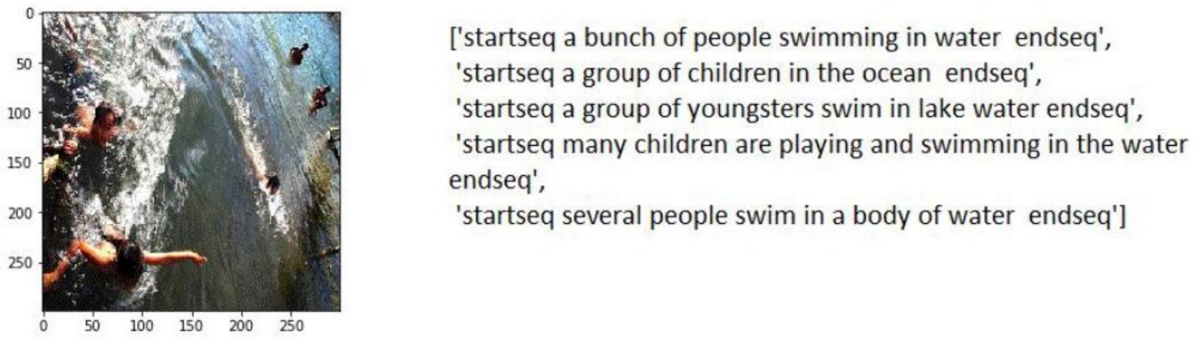
"Flickr8k_Dataset" : This is a freely available dataset which consists of eight thousand and ninety two images where six thousand are for training, one thousand for testing and the rest for validation

"Flickr8k_text": Contains text documents portraying "train_set", "test_set". "Flickr8k.token.txt" contains 5 inscriptions for each picture for example absolute 40460 subtitles.



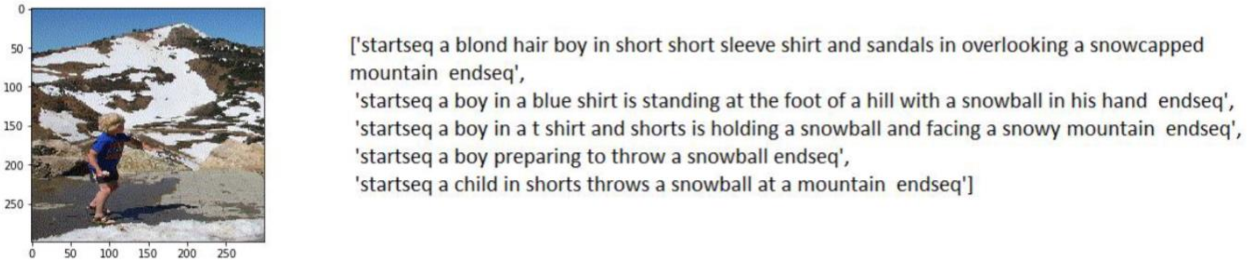
```
['startseq a jockey on a black horse jumps over a hurdle  endseq',
 'startseq an equestrian and a horse are jumping over an obstacle  endseq',
 'startseq a person wearing a navy jacket and black hat jumping over a small partition on a
horse  endseq',
 'startseq a show jumper is making a brown horse jump over a white fence  endseq',
 'startseq a woman on a horse jumps an obstacle  endseq']
```

Figure 6: Image with captions Sample 1



```
['startseq a bunch of people swimming in water  endseq',
 'startseq a group of children in the ocean  endseq',
 'startseq a group of youngsters swim in lake water endseq',
 'startseq many children are playing and swimming in the water
endseq',
 'startseq several people swim in a body of water  endseq']
```

Figure 7: Image with captions Sample 2



```
['startseq a blond hair boy in short short sleeve shirt and sandals in overlooking a snowcapped
mountain  endseq',
 'startseq a boy in a blue shirt is standing at the foot of a hill with a snowball in his hand  endseq',
 'startseq a boy in a t shirt and shorts is holding a snowball and facing a snowy mountain  endseq',
 'startseq a boy preparing to throw a snowball endseq',
 'startseq a child in shorts throws a snowball at a mountain  endseq']
```

Figure 8: Image with captions Sample 3

METHODOLOGY

1. Data Cleaning

This is the most basic process which takes place where some description related problem is involved. We need to remove all the unnecessary punctuation marks and stop words (eg – is, the, a, and, etc) which does not add much value to the output of the model. Suppose some word only comes once in the entire text, then that word is also not adding any value to the text as a whole. This basically makes the model robust and more efficient towards its accuracy and dealing with outliers.

2. Loading the training set

There are in total of six thousand images in the training set. These images comes with five captions each and we have to add the following sequences before and after the descriptions respectively:

‘startseq’ -> “This is a start sequence token which will be added at the start of every caption.”

‘endseq’ -> “This is an end sequence token which will be added at the end of every caption.”

3. Data Preprocessing on Images

So the question here is that what will be fed to the model. The answer is simple, i.e. images denoted by “X”. This input is also given in the form of a vector of fixed size and then further fed to the neural network. To solve this purpose, I have used Inception V3 model which uses transfer learning in its internal working. This was created by the team of google researchers. Lastly I just have to drop the “softmax” layer and then obtain the vector which is of length two thousand and forty eight. This is done for each and every image. Now, to save time in each run, it is a good practice to save these in a Pickle file. If this is not done, it will take same amount of time to train the model in each run which is very time consuming.

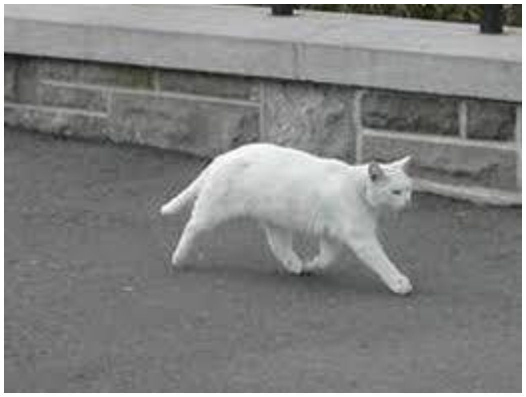
4. Data Preprocessing on Captions

Our main aim here is to guess what will be the most suitable description for the given photo. Therefore, for the model, the resultant will be captions that is the “target variable” (Y). We will not be able to generate the whole caption at once, but step wise. So, we will encrypt each word to some vector whose measure is fixed. I have created two “Python Dictionaries” namely “wordtoix” (word mapped to index) and “ixtoword” (index mapped to word).

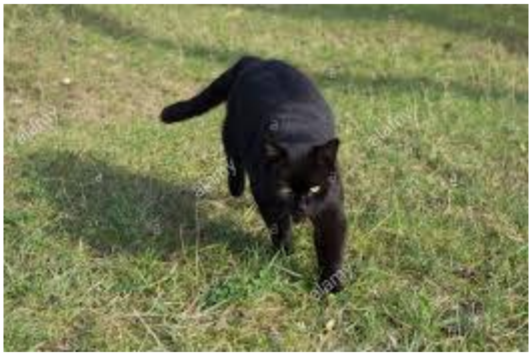
5. Data Preparation using Generator Function

These are the three images, first two for training and the third one for testing which I will use in the subsequent pages for explanation:

First image for training -> “The black cat sat on grass”



Second Image for training -> “The white cat is walking on road



Testing image -> “The black cat is walking on grass”

Figure 9: Train and Test Images

In the subsequent pages, I have used the first two photos from the training set and the third one to test it using the model.

The first and foremost step is to switch the training images in to a feature vector which measures two thousand and forty eight in length.

Secondly, I build the bag of words for the training the descriptions and by adding “startseq” before the description and “endseq” after the description:

First Caption -> “startseq the black cat sat on grass endseq”

Second Caption -> “startseq the white cat is walking on road endseq”

vocab = {“black”, “cat”, “endseq”, “grass”, “is”, “on”, “road”, “sat”, “startseq”, “the”, “walking”, “white”}

Further, in the bag of words we assign an “index” to each one of them:

“black” -1, “cat” -2, “endseq” -3, “grass” -4, “is” -5, “on” -6, “road” -7, “sat” -8, “startseq” -9, “the” -10, “walking” -11, “white” -12

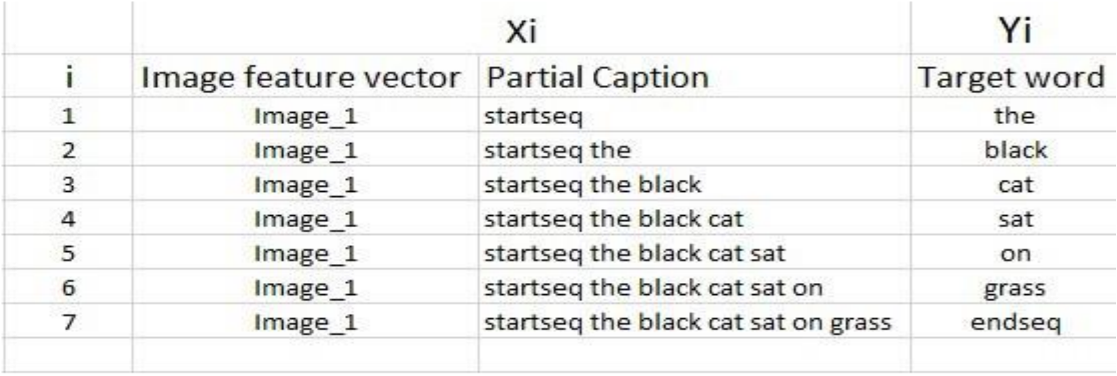
Firstly, I will provide the vectors of the image and the first set of character as input to the model and then subsequently it will guess what the next word is:

“Image_1” + “startseq”; Resultant = “the”

Then I provide vectors of the image and the two words in the beginning and then it will predict the next word, i.e.:

“Image_1” + “startseq the”; Resultant = “cat”

Now, the table containing the matrix will look as follows:

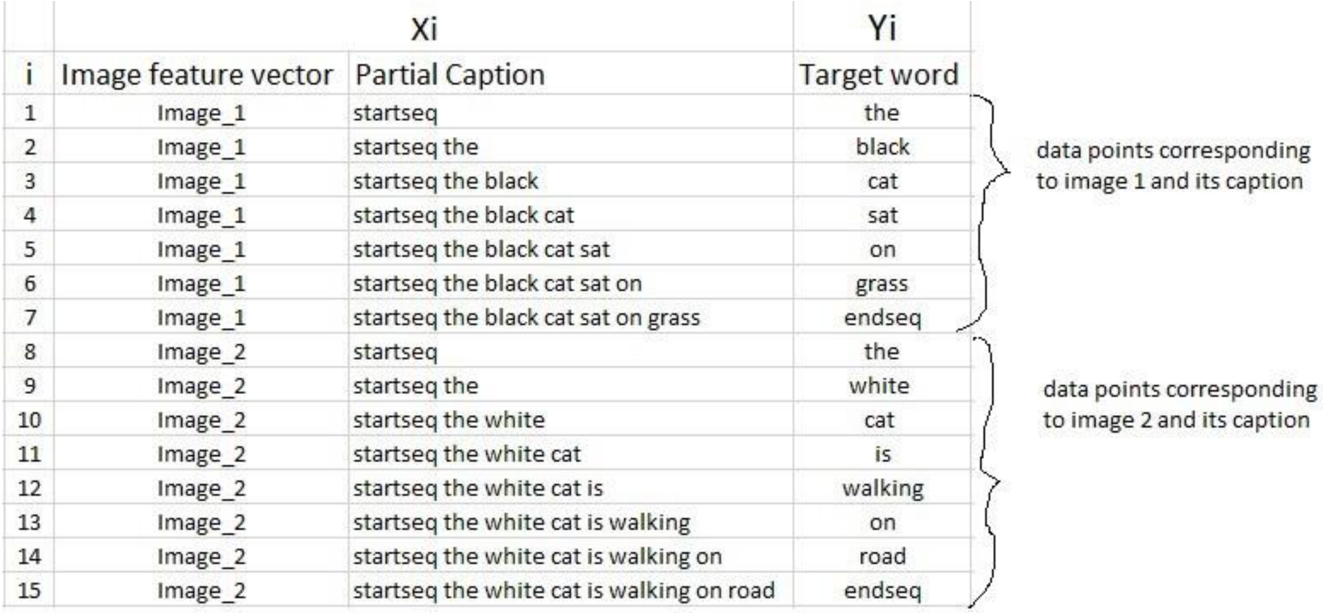


	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	startseq	the
2	Image_1	startseq the	black
3	Image_1	startseq the black	cat
4	Image_1	startseq the black cat	sat
5	Image_1	startseq the black cat sat	on
6	Image_1	startseq the black cat sat on	grass
7	Image_1	startseq the black cat sat on grass	endseq

Table 2: Caption Table with one image

A data point is not a set of one caption and one photo. They can be considered as more than one point of data which varies with the measure of characters in the description.

In the same way, when two photos are considered together, the table looks like this:



	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	startseq	the
2	Image_1	startseq the	black
3	Image_1	startseq the black	cat
4	Image_1	startseq the black cat	sat
5	Image_1	startseq the black cat sat	on
6	Image_1	startseq the black cat sat on	grass
7	Image_1	startseq the black cat sat on grass	endseq
8	Image_2	startseq	the
9	Image_2	startseq the	white
10	Image_2	startseq the white	cat
11	Image_2	startseq the white cat	is
12	Image_2	startseq the white cat is	walking
13	Image_2	startseq the white cat is walking	on
14	Image_2	startseq the white cat is walking on	road
15	Image_2	startseq the white cat is walking on road	endseq

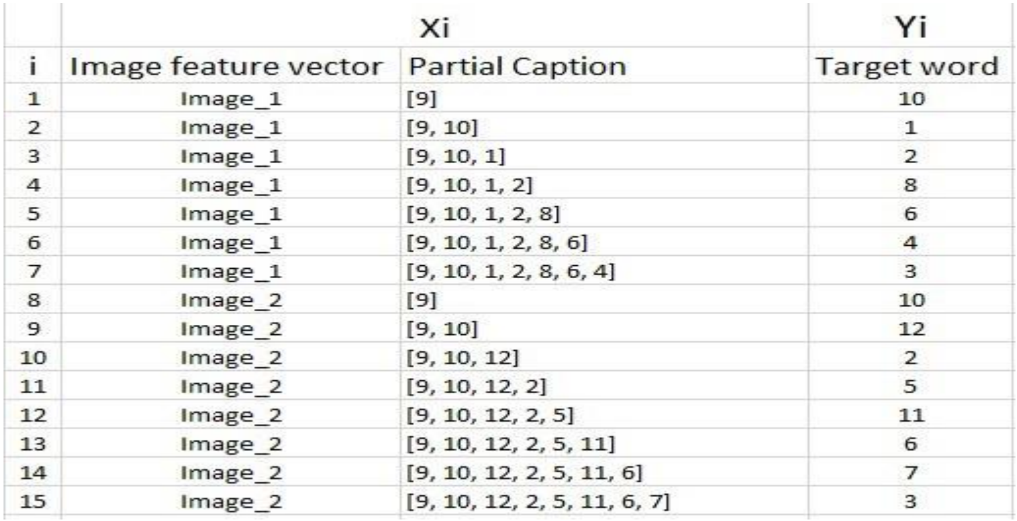
data points corresponding to image 1 and its caption

data points corresponding to image 2 and its caption

Table 3: Caption Table with two image

In each point of data, both the photo features and an incomplete set of description goes as an input so that the model can predict the next set of characters in a serial way. Here, RNN is the best model to be used since we are dealing with serial processing.

Now, the “indices” are processed sequentially where each one of them is mapped to a different set of characters. After that the table with the matrix will look as below:

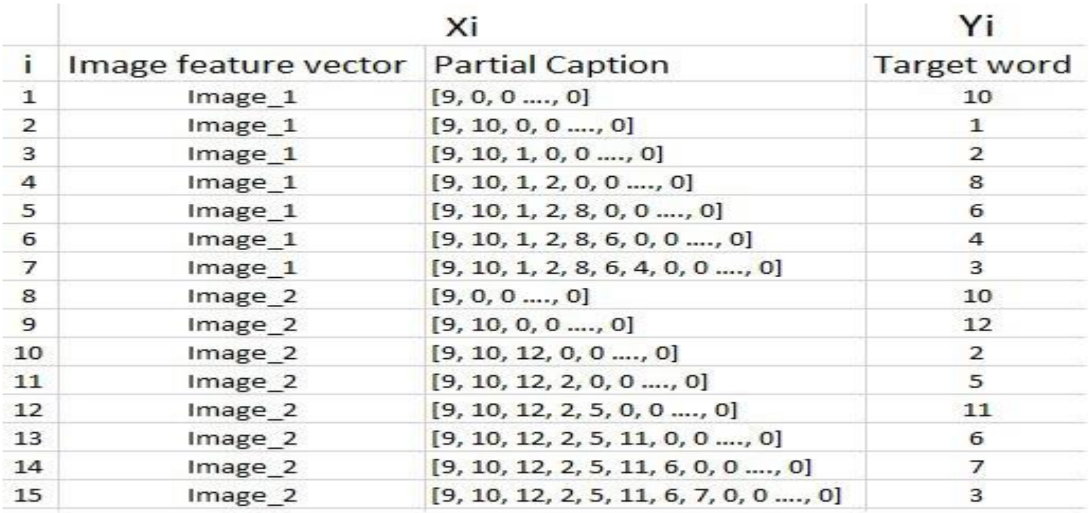


	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	[9]	10
2	Image_1	[9, 10]	1
3	Image_1	[9, 10, 1]	2
4	Image_1	[9, 10, 1, 2]	8
5	Image_1	[9, 10, 1, 2, 8]	6
6	Image_1	[9, 10, 1, 2, 8, 6]	4
7	Image_1	[9, 10, 1, 2, 8, 6, 4]	3
8	Image_2	[9]	10
9	Image_2	[9, 10]	12
10	Image_2	[9, 10, 12]	2
11	Image_2	[9, 10, 12, 2]	5
12	Image_2	[9, 10, 12, 2, 5]	11
13	Image_2	[9, 10, 12, 2, 5, 11]	6
14	Image_2	[9, 10, 12, 2, 5, 11, 6]	7
15	Image_2	[9, 10, 12, 2, 5, 11, 6, 7]	3

Table 4: Caption Table converted to Feature vectors

I have utilized group handling thus, I need to ensure that each succession is of equivalent length. Consequently I need to affix 0's (zero cushioning) toward the finish of each grouping. I attach those many number of zeros which will prompt each succession having greatest length.

After the padding process, the table will look this way:



	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	[9, 0, 0 …., 0]	10
2	Image_1	[9, 10, 0, 0 …., 0]	1
3	Image_1	[9, 10, 1, 0, 0 …., 0]	2
4	Image_1	[9, 10, 1, 2, 0, 0 …., 0]	8
5	Image_1	[9, 10, 1, 2, 8, 0, 0 …., 0]	6
6	Image_1	[9, 10, 1, 2, 8, 6, 0, 0 …., 0]	4
7	Image_1	[9, 10, 1, 2, 8, 6, 4, 0, 0 …., 0]	3
8	Image_2	[9, 0, 0 …., 0]	10
9	Image_2	[9, 10, 0, 0 …., 0]	12
10	Image_2	[9, 10, 12, 0, 0 …., 0]	2
11	Image_2	[9, 10, 12, 2, 0, 0 …., 0]	5
12	Image_2	[9, 10, 12, 2, 5, 0, 0 …., 0]	11
13	Image_2	[9, 10, 12, 2, 5, 11, 0, 0 …., 0]	6
14	Image_2	[9, 10, 12, 2, 5, 11, 6, 0, 0 …., 0]	7
15	Image_2	[9, 10, 12, 2, 5, 11, 6, 7, 0, 0 …., 0]	3

Table 5: Padded Feature vectors

6. Word Embeddings

These are basically the comprehensible text which is switched to corresponding numbers. But, it may happen that there is different representations of numbers of the same string. Here, I am using the “GloVe” Model to map every comprehensible set of characters to a vector of size two hundred.

7. Conversion of text to speech

Here we use gTTS module to convert the generated caption into a speech. We will use a loop to iteratively accept the new images as an input and keep saving it in our machine

4. Proposed System Analysis and Design

4.1 Introduction

For this section, I shall analyze the different types of requirements for the project. This will include the following requirements:

Functional Requirements – The requirements that are much needed by the user which is regarded as the very basic necessity of the system. These functionalities should be necessarily be incorporated in the system without fail.

Non-functional Requirements – These are fundamentally the quality imperatives that the framework should fulfill as indicated by the venture contract. The need or degree to which these components are executed changes from one undertaking to other. They are additionally called non-social necessities.

System Requirements – These include the requirements for hardware and software to run the project. This includes the resources that the proposed system will consume for its smooth functioning.

4.2 Requirement Analysis

4.2.1 Functional Requirements

4.2.1.1 Product Perspective

This product is basically a sort of an aid to the blind people who are unable to consume a glimpse of this beautiful world. It will describe him the scenery which comes in front of the blind person (user) through speech or voice notes.

4.2.1.2 Product features

This includes the traits and attributes of the product. The product will take the real-life images as an input and give a description of the image as speech as an output.

4.2.1.3 User characteristics

The product will be useful for a vast number of people as this is a new product which the blind people can use and they will not feel that they are different from normal people. They can use it whenever they feel that they should go out for a walk. This will also make them independent.

4.2.1.4 Assumption & Dependencies

This product is specifically for blind person only. This product is not for the person who has hearing deficiency.

4.2.1.5 Domain Requirements

The domain requirement for this product would be that it should be able to capture the image, extract features from the image and generate caption from the images that it captures.

4.2.1.6 User Requirements

The user requirement for this product would be that the blind person should be able to observe the surroundings independently without any external human support. This will be done by generating a comprehensible caption for the user.

4.2.2 Non-functional Requirements

4.2.2.1 Product Requirements

4.2.2.1.1 Efficiency

The initial time requirements are for time required to train the model. After that when a new image is given as input, it takes no time for the model to generate a suitable caption. The size of the dataset used is just over a gigabyte.

4.2.2.1.2 Reliability

The system is predicted to be of good reliability as an accuracy of more than 85% is expected.

4.2.2.1.3 Portability

The system is highly portable because now a days people wear cameras on the chest or headgear while going for an adventure trip and to shoot vlogs. This camera setup will be used by the user to capture real life images.

4.2.2.1.4 Usability

The product will be useful for a vast number of people as this is a new product which the blind people can use and they will not feel that they are different from normal people. They can use it whenever they feel that they should go out for a walk. This will also make them independent.

4.2.2.2 Organizational Requirements

4.2.2.2.1 Implementation Requirements (in terms of deployment)

IDE – Jupyter Notebook, Dataset of Images and Captions (Training and Testing) – Flickr8k, Flickr30k, Microsoft COCO Dataset, Real Time Images, Knowledge of Python.

4.2.2.2.2 ENGINEERING STANDARD REQUIREMENTS

Our solution should be in accordance with all the latest metrics in the standards list set by the reputable regulatory authority. Standards are distributed records made to guarantee the unwavering quality of the materials, items, strategies, as well as administrations. They build up prerequisites, particulars, rules, qualities, or potentially methods planned. Ordinarily, they are created through an agreement interaction and endorsed by different public and global offices, proficient social orders, or industry associations. Standards are the negligibly acknowledged proficient practice and additionally quality that should be noticed.

We have gone through a basic overview of the IEEE and ISO software engineering standards and have not found any of our practices violating their clauses.

4.2.2.2 OPERATIONAL REQUIREMENTS

ECONOMIC

Software engineering economics aspects is a theme that tends to the components of programming project costs assessment and examination and undertaking advantage cost proportion investigation. Too these expenses, and the advantages of consuming those costs produce unmistakable and ordinarily the immaterial worth. The time-staged parts of creating programming for cash, implies we need to comprehend the planning parts of delivering this worth. Each of the three factors in the worldview of programming advancement for cash - time, cost, and worth - are irregular factors. This arbitrariness comes from the hidden vulnerabilities in the cycles found in the advancement of the product. These vulnerabilities are consistently there, they never disappear, they are permanent.

Our solution does not require users to bear any extra expensive additional costs that has implications on their business plan. Since we are not going for the extensive machine learning based approach, therefore we save our users extra expenses that would be otherwise required on a powerful processer or bigger memory. Our code is lightweight and therefore easily compatible with normal systems. No external hardware is required.

ENVIRONMENTAL

Our project is entirely software based and also avoids the requirement of extensive hardware and processing capabilities. Therefore our carbon footprint is negligible.

SOCIAL

Our solution is basically for social welfare. Now a days many old age parents who are blind feel lonely because their kids cannot afford to meet the expenses. These blind people will be taken care of as they will be engaged in observing the world with virtual eyes.

Political

If we see in today’s world cyber-attacks are very often, every country wants to get data from another country and use them against their enemy. If they can’t get inside military cyber-attacks are the bets options. So, our project can help the authorities to at least stop the sql injection from enemies and protect the data. Hence, it forms a political requirement that these types of securities are must. This could be a good factor for a ruling party to depict their achievements that they saved their country. If they fail to stop that then the opposition party could use it as leverage so our system could help them also for their seat and country protection.

Ethical

The project we are doing is ethical in way of securing the boundaries of any organization to protect their rights and showing the outsiders that we are secure and we can save our assets. Our model show that we care about the security from hackers and also provide with an easy but reliable system.

Legality

It’s completely legal to save our assets. Laws are an important factor in cyber securities. There are laws that we can use any softwares to maintain our assets and locate and mark the person who is trying to compromise the security area and making him pay for it according to rules and laws. To manage security we should be able to remember the type of injection they are using so we can do it in our system by law.

Health and Safety

As our product doesn’t match a individual health or safety but it looks after the organizational health and safety. If it is compromised then the company might go in data breach and they can lost their customers and their company value health and assets health might go bad and customers might not feel confidence about company and their data safety. Our project tries in a way that the health and safety of the organization remain intact by using some algorithmic thinking.

Sustainability

Data in today’s world has become an integral part of society. Many companies thrive because of them. Data is being sold on network mostly by OTT organizations. So, for the companies’ data has become a sustainable part of them. They are sustained because of them. Their working, their assets and their work depends on data and hence saving data for them is very important and they can’t make any mistake in it, if they make, they might not sustain in data world fight and hence for them our system will add an extra feature in their sustainable cycle.

Inspectability

Inspectability in other word is publicly inspected or observe. So, our system in any log will send a log whenever there is a request send to server. It depends that the request is valid or not but we can see what the system does and behave when there is need to. Inspectability helps admin to reduce their work to analyze the error that network has encountered upon a request. Our system has the feature of Inspectability so that the maintenance is reduced and more business work is done by people sitting in the admin or network park to inspect their organizational assets.

4.2.3 System Requirements

4.2.3.1 H/W Requirements

Since the dataset is large and that too it comprises of images, a heavy configuration machine is required to train the model.

RAM - 16GB, CUDA enabled GPU with at least 4 GB Memory

4.2.3.2 S/W Requirements

An Integrated Development Environment to run deep learning modules on Python, Windows/Linux OS

INFERENCE

Now we are done with the step of preparing the data and building the model. Now what if a new image is introduced to the model. For example-

We take this image as a reference and see how is the flow of generation of a caption:

Caption -> “the black cat is walking on grass”

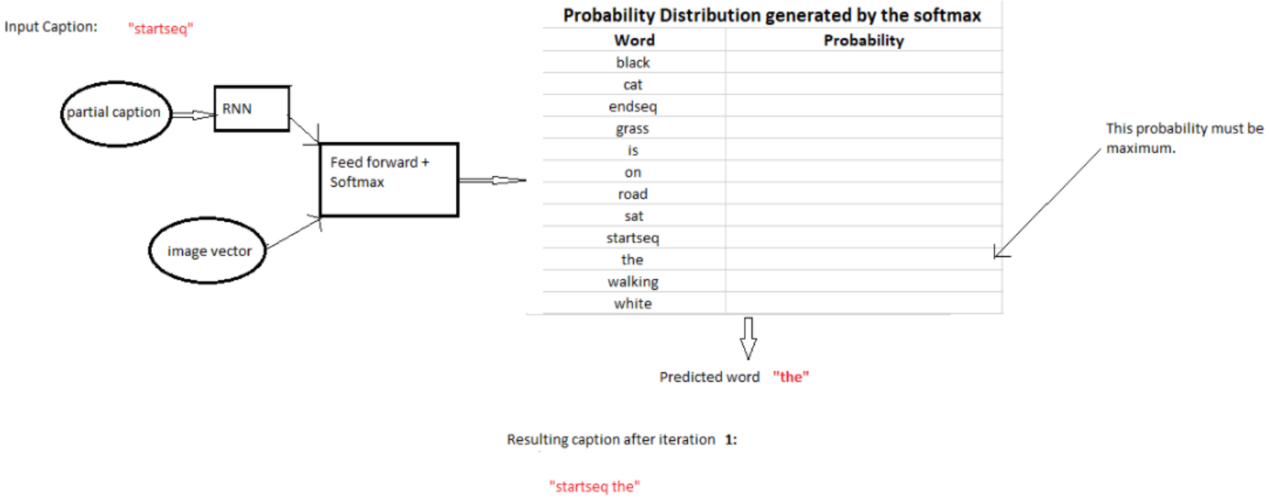
The bag of words after cleaning the data would be:

vocab = {“black”, “cat”, “endseq”, “grass”, “is”, “on”, “road”, “sat”, “startseq”, “the”, “walking”, “white”}

First Iteration:

Input: “Image vector” + “startseq” (not the full caption)

Resultant word: “the”


Input Caption: "startseq"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "the"
Resulting caption after iteration 1:
"startseq the"

Probability Distribution generated by the softmax

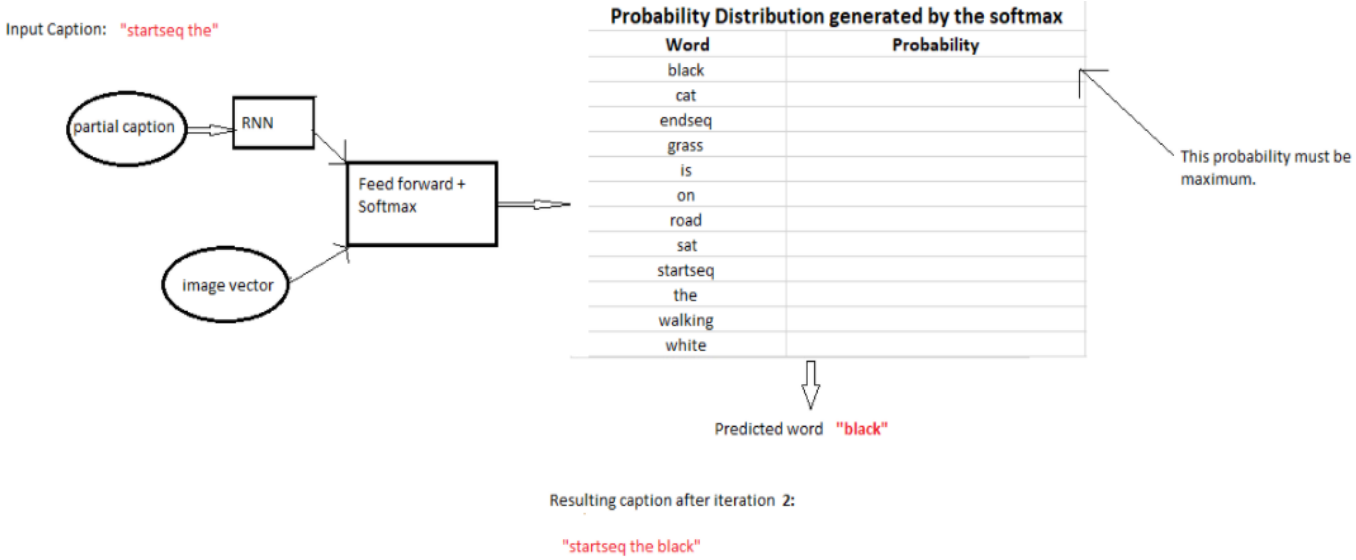
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 10: Result after first iteration

Second Iteration:

Input: “Image vector” + “startseq the”

Resultant word: “black”


Input Caption: "startseq the"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "black"
Resulting caption after iteration 2:
"startseq the black"

Probability Distribution generated by the softmax

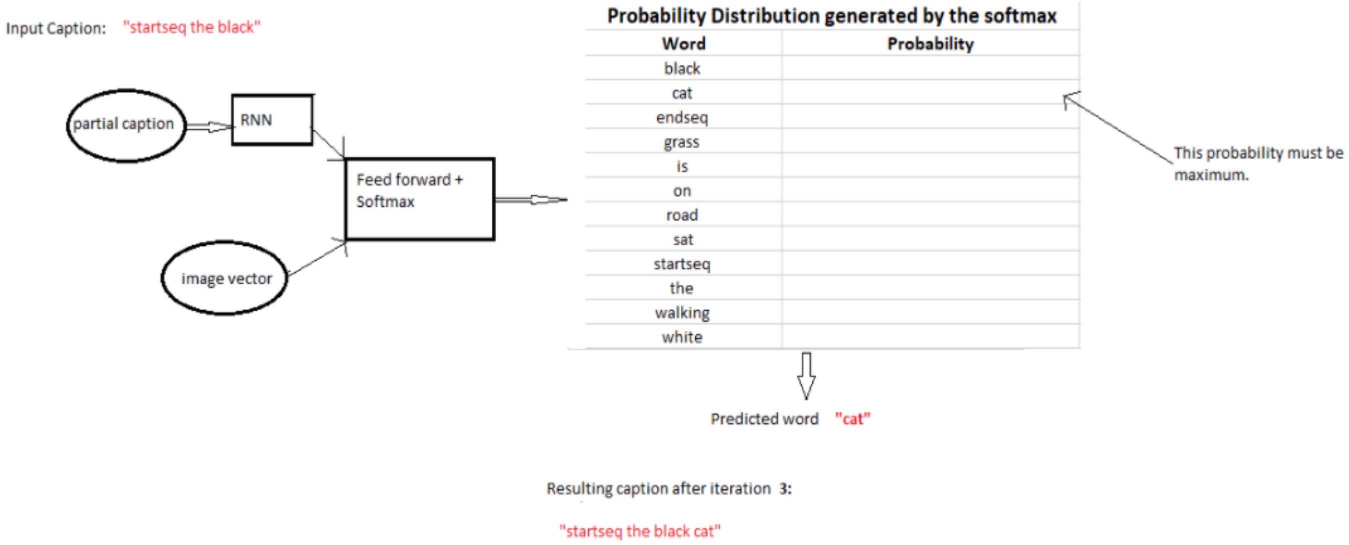
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 11: Result after second iteration

Third Iteration:

Input: “Image vector” + “startseq the black”

Resultant word: “cat”


Input Caption: "startseq the black"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "cat"
Resulting caption after iteration 3:
"startseq the black cat"

Probability Distribution generated by the softmax

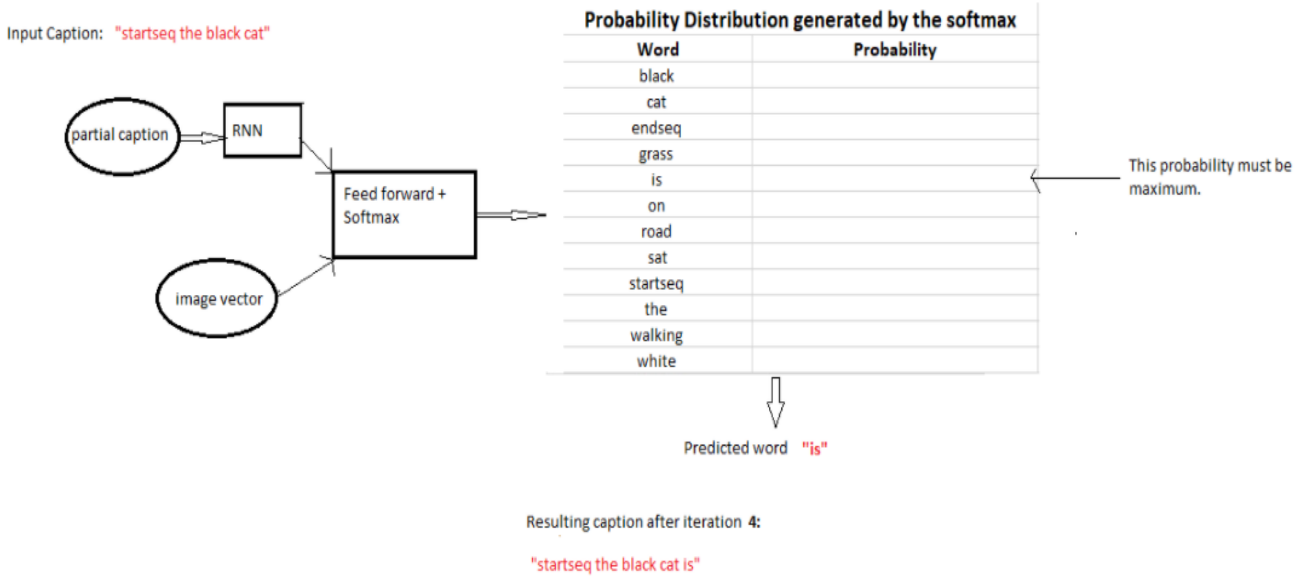
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 12: Result after third iteration

Forth Iteration:

Input: “Image vector” + “startseq the black cat”

Resultant word: “is”


Input Caption: "startseq the black cat"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "is"
Resulting caption after iteration 4:
"startseq the black cat is"

Probability Distribution generated by the softmax

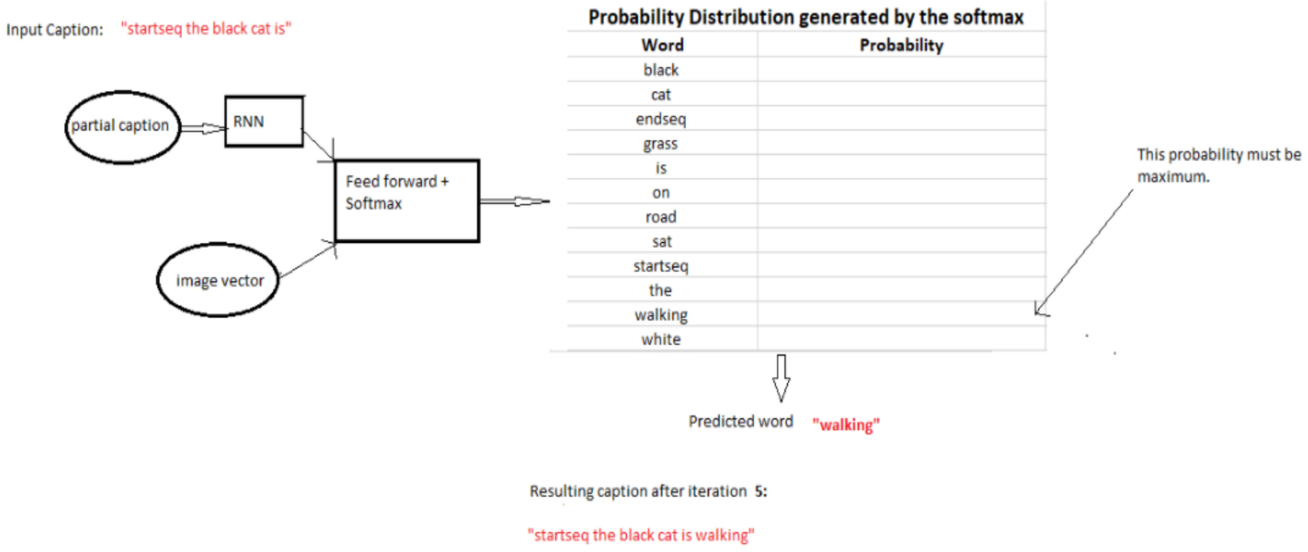
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 13: Result after forth iteration

Fifth Iteration:

Input: “Image vector” + “startseq the black cat is”

Resultant word: “walking”


Input Caption: "startseq the black cat is"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "walking"
Resulting caption after iteration 5:
"startseq the black cat is walking"

Probability Distribution generated by the softmax

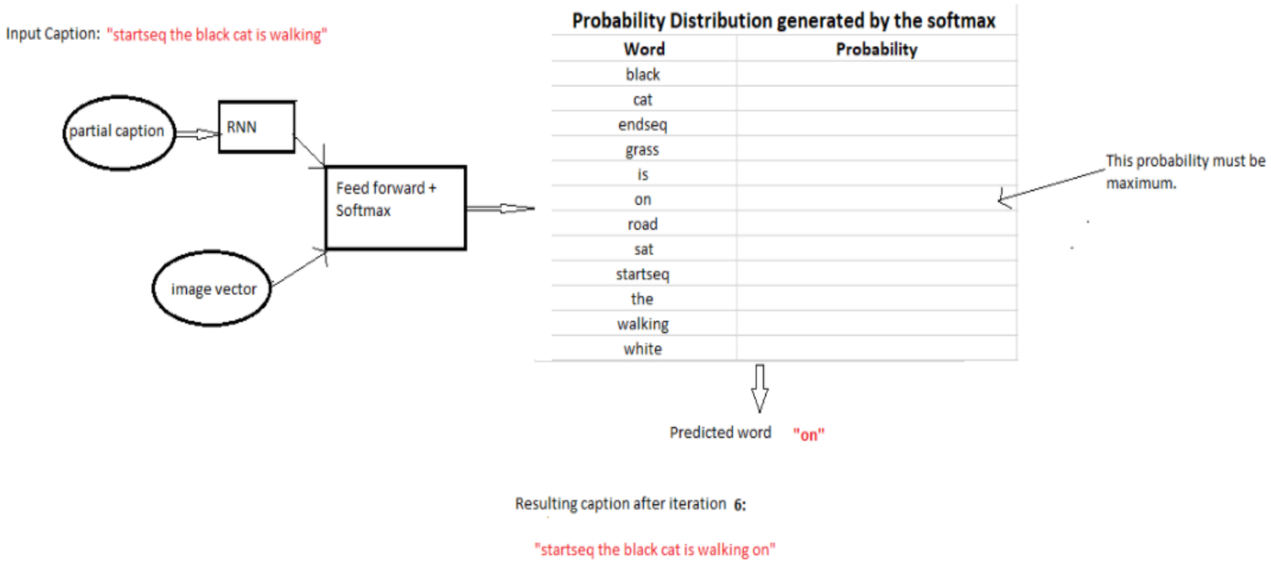
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 14: Result after fifth iteration

Sixth Iteration:

Input: “Image vector” + “startseq the black cat is walking”

Resultant word: “on”


Input Caption: "startseq the black cat is walking"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "on"
Resulting caption after iteration 6:
"startseq the black cat is walking on"

Probability Distribution generated by the softmax

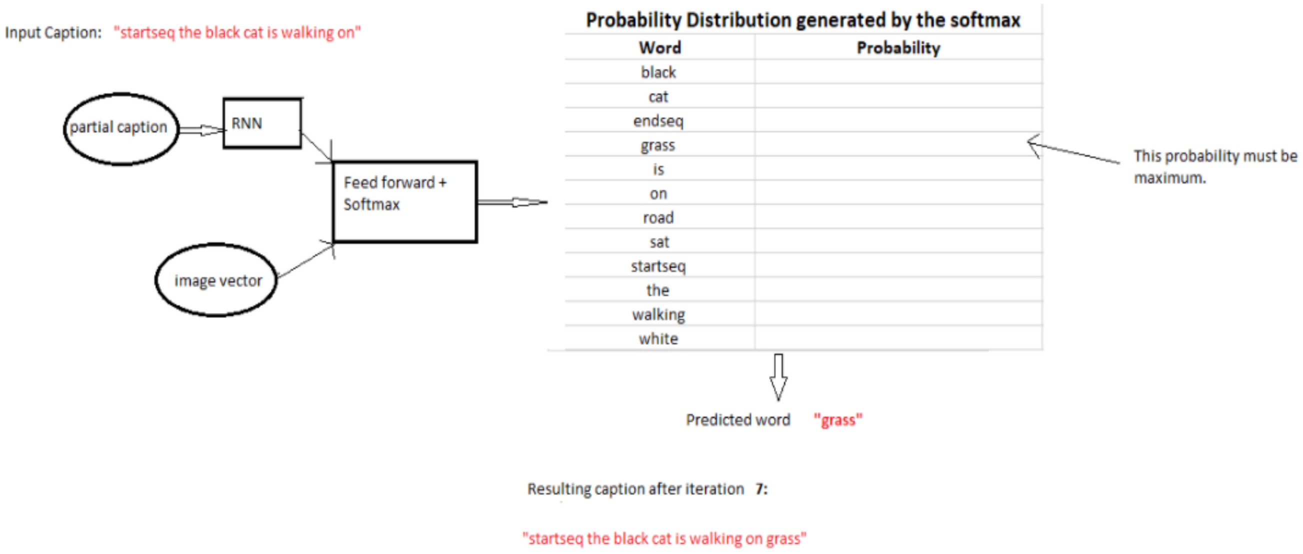
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 15: Result after sixth iteration

Seventh Iteration:

Input: “Image vector” + “startseq the black cat is walking on”

Resultant word: “grass”


Input Caption: "startseq the black cat is walking on"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "grass"
Resulting caption after iteration 7:
"startseq the black cat is walking on grass"

Probability Distribution generated by the softmax

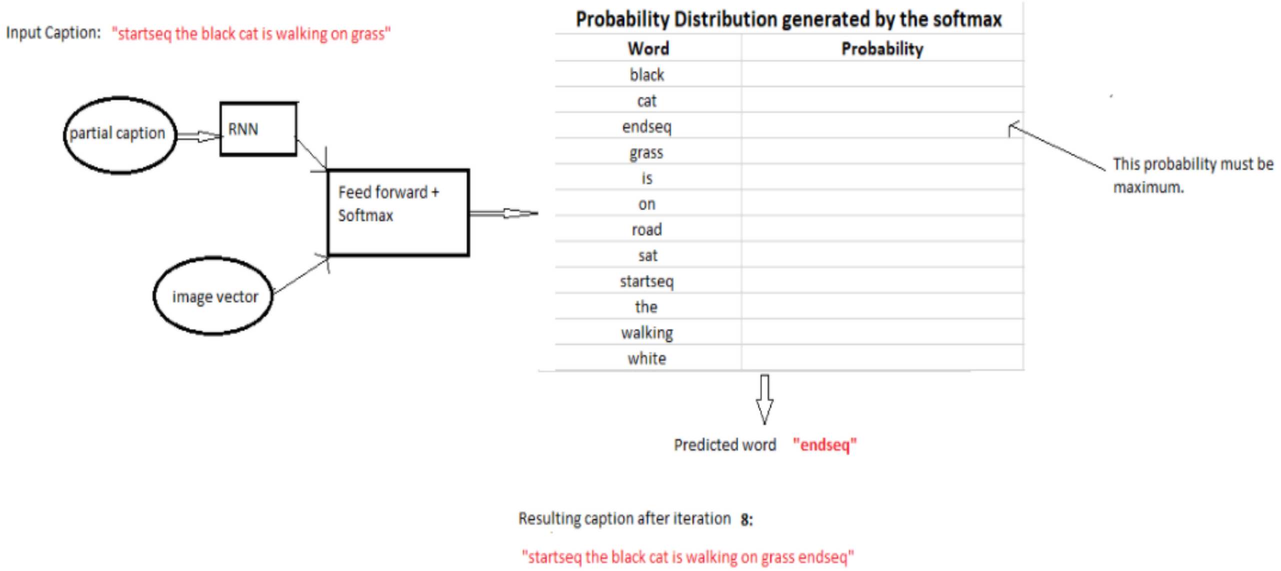
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 16: Result after seventh iteration

Eighth Iteration:

Input: “Image vector” + “startseq the black cat is walking on grass”

Resultant word: “endseq”


Input Caption: "startseq the black cat is walking on grass"
partial caption
RNN
image vector
Feed forward + Softmax
This probability must be maximum.
Predicted word "endseq"
Resulting caption after iteration 8:
"startseq the black cat is walking on grass endseq"

Probability Distribution generated by the softmax

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 17: Result after iteration 8

Hence, we come to the end of the iterations. When the program encounters an “endseq” this means that the iterations need to be stopped. Then after removing these tokens, we get the final output of the model.

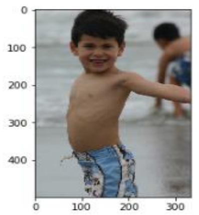
5. Results and Discussion

Dataset Used: Flickr 8k

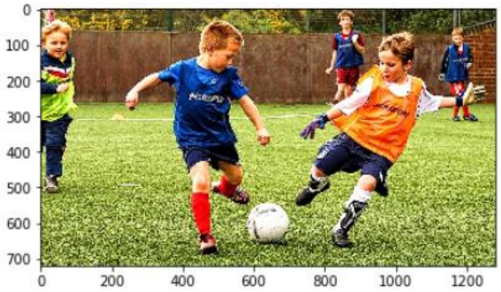
Dataset Author: Hsankesara

I have been able to achieve a fair accuracy of 70-80% for the model. This is based on the assumptions and output form the model. There can be more improvements in the accuracy if larger dataset is used such as Microsoft COCO or Flickr30k. I think this is a very strong project with high capabilities and applications starting with feature extraction, then converting it into a comprehensible text and then speech. While our gauge rivals in exactness more intricate best in class models, I show that its benefits can't be effortlessly parlayed into enhancements for such errands as picture sentence recovery, consequently underlining the limits of current strategies and the degree for development. Next, I will also try to test my model with real-time images to make my model application ready.

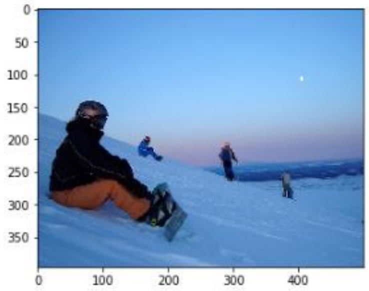
Tools used: jupyter notebook, TensorFlow, keras



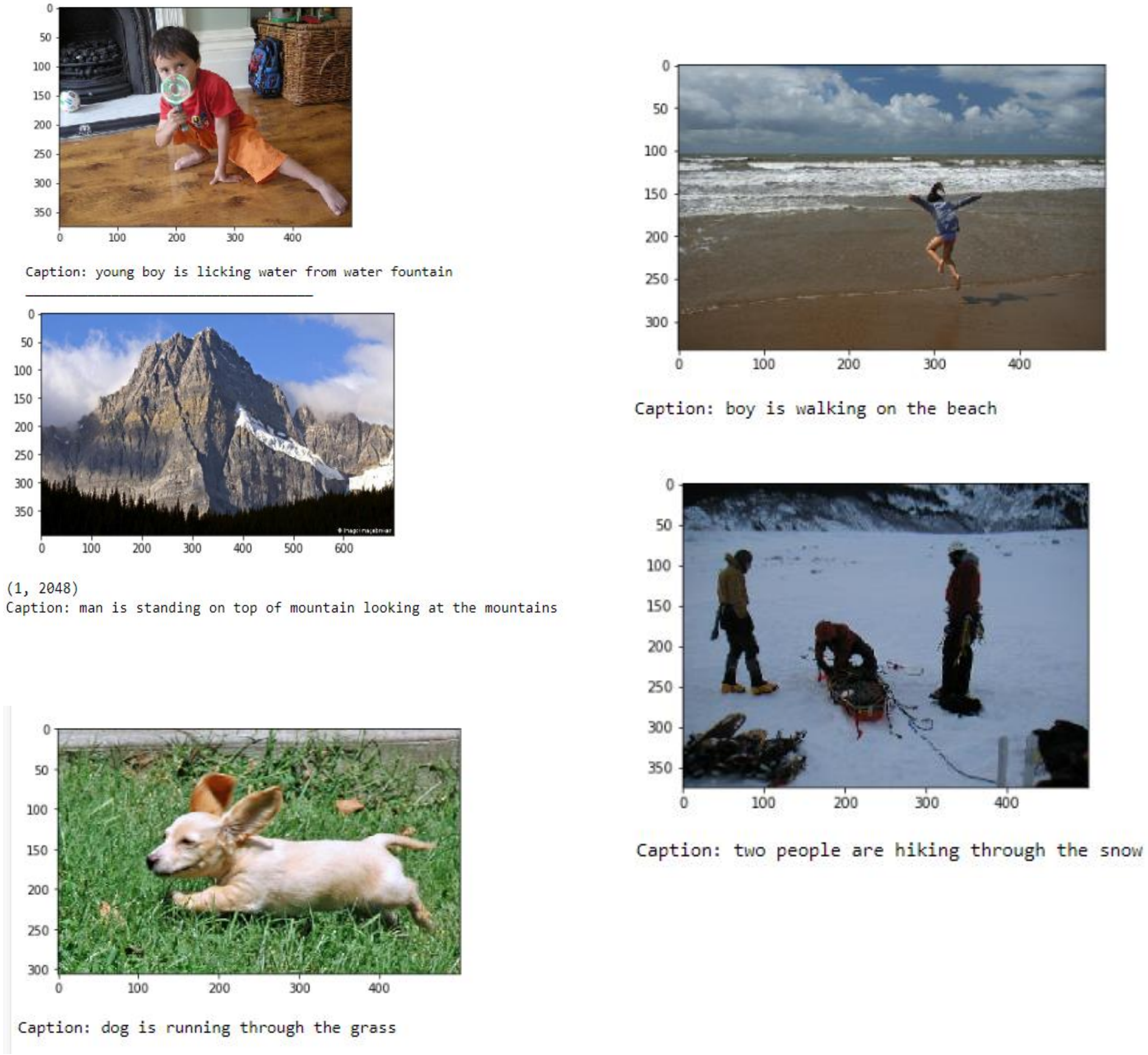
```
Caption: small boy wades in the water
```



```
(1, 2048)
Caption: two boys play soccer on field
```



```
Caption: man in red suit is standing on snow covered hill
```



```
Caption: young boy is licking water from water fountain
```

```
(1, 2048)
Caption: man is standing on top of mountain looking at the mountains
```

```
Caption: dog is running through the grass
```

```
Caption: boy is walking on the beach
```

```
Caption: two people are hiking through the snow
```

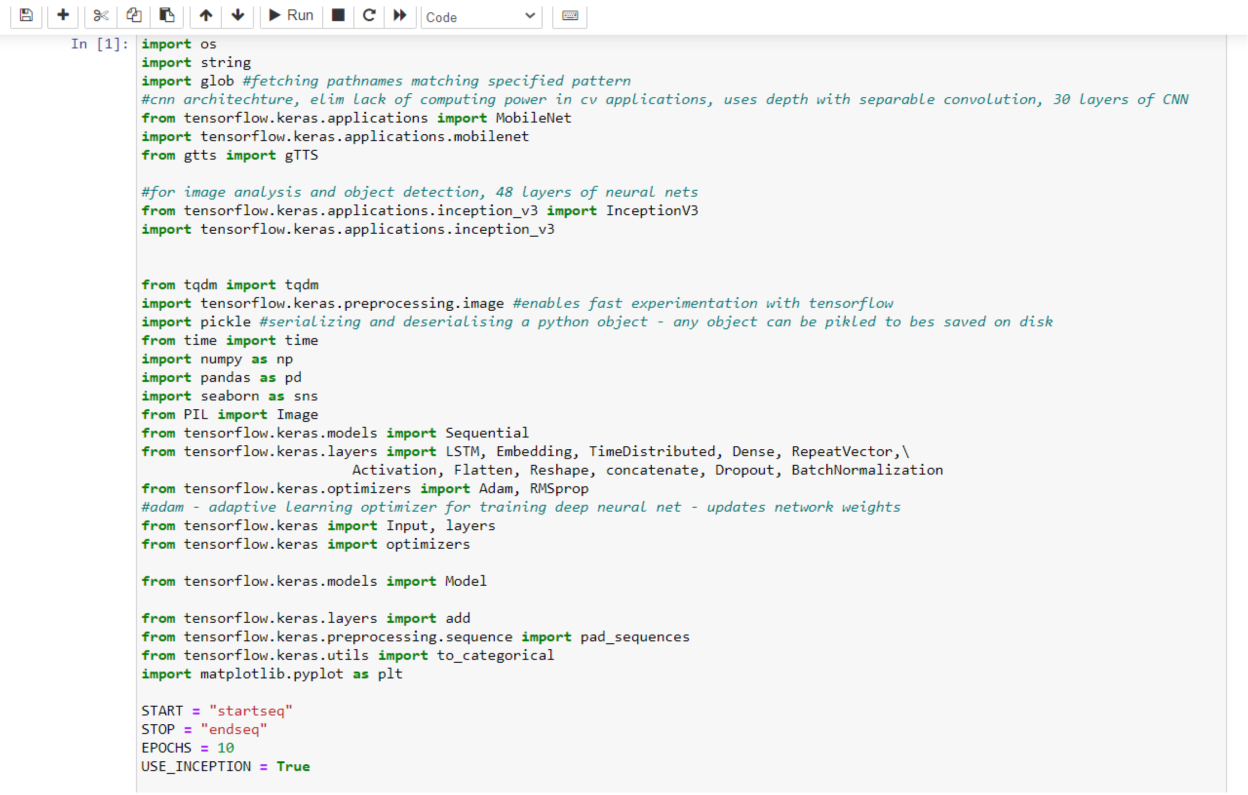
Figure 18: Test Images with Captions

6. References

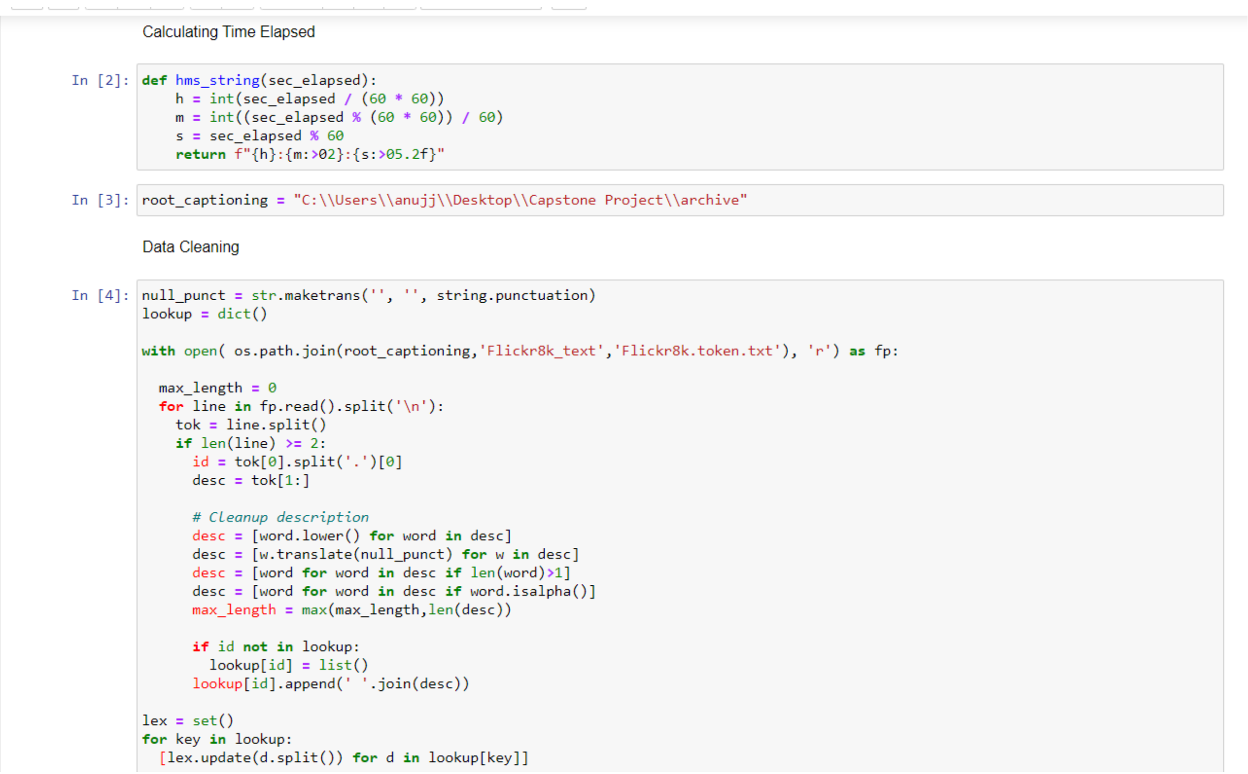
- Wang, E. K., Zhang, X., Wang, F., Wu, T. Y., & Chen, C. M. (2019). Multilayer dense attention model for image caption. *IEEE Access*, *7*, 66358-66368.
- Gupta, N., & Jalal, A. S. (2019). Integration of textual cues for fine-grained image captioning using deep CNN and LSTM. *Neural Computing and Applications*, 1-10.
- Ghosh, A., Dutta, D., & Moitra, T. (2020). A Neural Network Framework to Generate Caption from Images. In *Emerging Technology in Modelling and Graphics* (pp. 171-180). Springer, Singapore.
- Bhatia, Y., Bajpayee, A., Raghuvanshi, D., & Mittal, H. (2019, August). Image Captioning using Google's Inception-resnet-v2 and Recurrent Neural Network. In *2019 Twelfth International Conference on Contemporary Computing (IC3)* (pp. 1-6). IEEE.
- Yang, J., Sun, Y., Liang, J., Ren, B., & Lai, S. H. (2019). Image captioning by incorporating affective concepts learned from both visual and textual components. *Neurocomputing*, *328*, 56-68.
- Su, J., Tang, J., Lu, Z., Han, X., & Zhang, H. (2019). A neural image captioning model with caption-to-images semantic constructor. *Neurocomputing*, *367*, 144-151.
- Zhang, X., He, S., Song, X., Lau, R. W., Jiao, J., & Ye, Q. (2019). Image captioning via semantic element embedding. *Neurocomputing*.
- Xu, N., Liu, A. A., Liu, J., Nie, W., & Su, Y. (2019). Scene graph captioner: Image captioning based on structural visual representation. *Journal of Visual Communication and Image Representation*, *58*, 477-485.

Appendix A

Sample Code



```
In [1]: import os
        import string
        import glob #fetching pathnames matching specified pattern
        #cnn architechture, elim lack of computing power in cv applications, uses depth with separable convolution, 30 layers of CNN
        from tensorflow.keras.applications import MobileNet
        import tensorflow.keras.applications.mobilenet
        from gtts import gTTS

        #for image analysis and object detection, 48 layers of neural nets
        from tensorflow.keras.applications.inception_v3 import InceptionV3
        import tensorflow.keras.applications.inception_v3


        from tqdm import tqdm
        import tensorflow.keras.preprocessing.image #enables fast experimentation with tensorflow
        import pickle #serializing and deserializing a python object - any object can be pikled to bes saved on disk
        from time import time
        import numpy as np
        import pandas as pd
        import seaborn as sns
        from PIL import Image
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import LSTM, Embedding, TimeDistributed, Dense, RepeatVector,\
                                 Activation, Flatten, Reshape, concatenate, Dropout, BatchNormalization
        from tensorflow.keras.optimizers import Adam, RMSprop
        #adam - adaptive learning optimizer for training deep neural net - updates network weights
        from tensorflow.keras import Input, layers
        from tensorflow.keras import optimizers

        from tensorflow.keras.models import Model

        from tensorflow.keras.layers import add
        from tensorflow.keras.preprocessing.sequence import pad_sequences
        from tensorflow.keras.utils import to_categorical
        import matplotlib.pyplot as plt

        START = "startseq"
        STOP = "endseq"
        EPOCHS = 10
        USE_INCEPTION = True
```



Calculating Time Elapsed

```
In [2]: def hms_string(sec_elapsed):
            h = int(sec_elapsed / (60 * 60))
            m = int((sec_elapsed % (60 * 60)) / 60)
            s = sec_elapsed % 60
            return f"{h}:{m:>02}:{s:>05.2f}"
```

```
In [3]: root_captioning = "C:\\Users\\anujj\\Desktop\\Capstone Project\\archive"
```

Data Cleaning

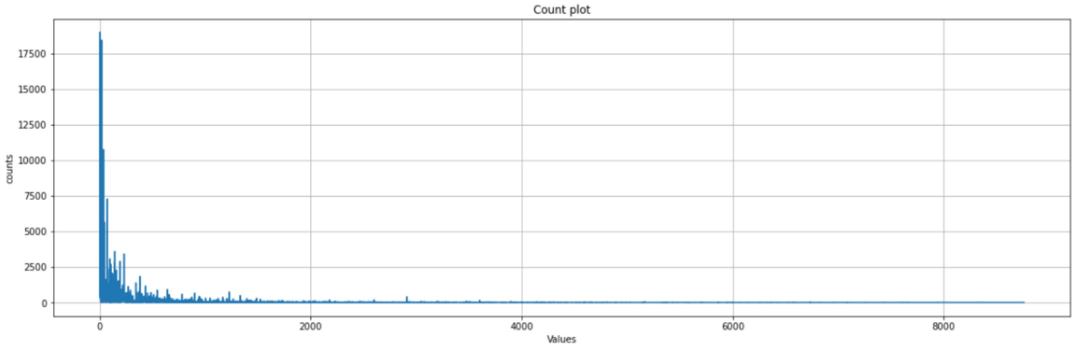
```
In [4]: null_punct = str.maketrans('', '', string.punctuation)
        lookup = dict()

        with open( os.path.join(root_captioning,'Flickr8k_text','Flickr8k.token.txt'), 'r') as fp:

          max_length = 0
          for line in fp.read().split('\n'):
            tok = line.split()
            if len(line) >= 2:
              id = tok[0].split('.')[0]
              desc = tok[1:]

              # Cleanup description
              desc = [word.lower() for word in desc]
              desc = [w.translate(null_punct) for w in desc]
              desc = [word for word in desc if len(word)>1]
              desc = [word for word in desc if word.isalpha()]
              max_length = max(max_length,len(desc))

              if id not in lookup:
                lookup[id] = list()
              lookup[id].append(' '.join(desc))

        lex = set()
        for key in lookup:
          [lex.update(d.split()) for d in lookup[key]]
```

```
In [5]: print(len(lookup)) # How many unique words
        print(len(lex)) # The dictionary
        print(max_length) # Maximum length of a caption (in words)
```

```
8092
8763
32
```

Distribution of word count

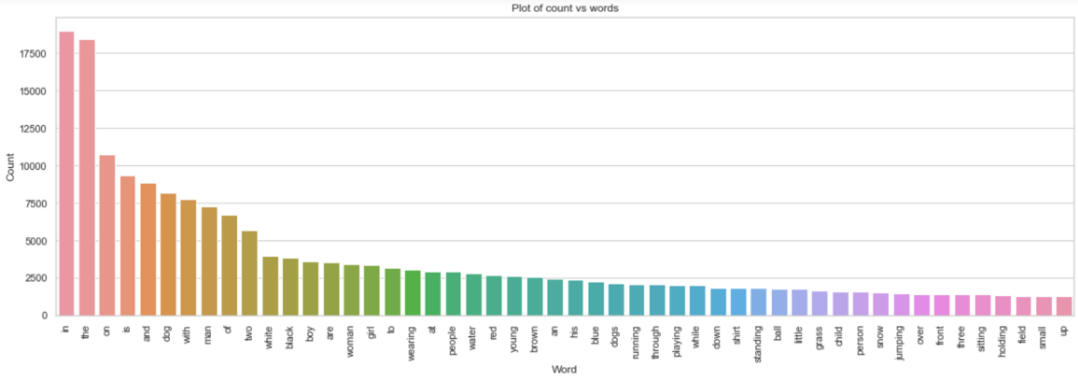
```
In [6]: wordfreq = {}
        temlst=[]
        for raw_word in lookup:
            words = temlst.extend(lookup[raw_word])
        temlst1=[]
        for raw_words in temlst:
            x=raw_words.split()
            for i in x:
                if i!='startseq' and i!='endseq':
                    if i not in wordfreq:
                        wordfreq[i]=0
                    wordfreq[i]+=1

        df=pd.DataFrame(wordfreq.items(), columns=['Word', 'Count'])
        df["Count"].plot(grid=True,figsize=(20,6))
        plt.title("Count plot")
        plt.xlabel("Values")

        plt.ylabel("counts")
        plt.show()
```



Distribution of top 50 words

```
In [7]: df=df.sort_values(by='Count',ascending=False)
        sns.set(style="whitegrid")
        plt.figure(figsize=(20,6))
        sns.barplot(x="Word",y="Count",data=df[:50])
        plt.xticks(rotation="vertical")
        plt.title("Plot of count vs words")
```

```
Out[7]: Text(0.5, 1.0, 'Plot of count vs words')
```



In [8]:
```
# Just rerun if len(img) = 0
img = glob.glob(os.path.join(root_captioning,'Images', '*.jpg'))
```

In [9]:
```
len(img)
```

Out[9]:
```
8091
```

Importing captions for Training and Testing Images

In [10]:
```
train_images_path = os.path.join(root_captioning,'Flickr8k_text','Flickr_8k.trainImages.txt')
train_images = set(open(train_images_path, 'r').read().strip().split('\n'))
test_images_path = os.path.join(root_captioning,'Flickr8k_text','Flickr_8k.testImages.txt')
test_images = set(open(test_images_path, 'r').read().strip().split('\n'))

train_img = []
test_img = []

for i in img:
  f = os.path.split(i)[-1]
  if f in train_images:
    train_img.append(f)
  elif f in test_images:
    test_img.append(f)
```

In [11]:
```
print(len(train_images))
print(len(test_images))
```

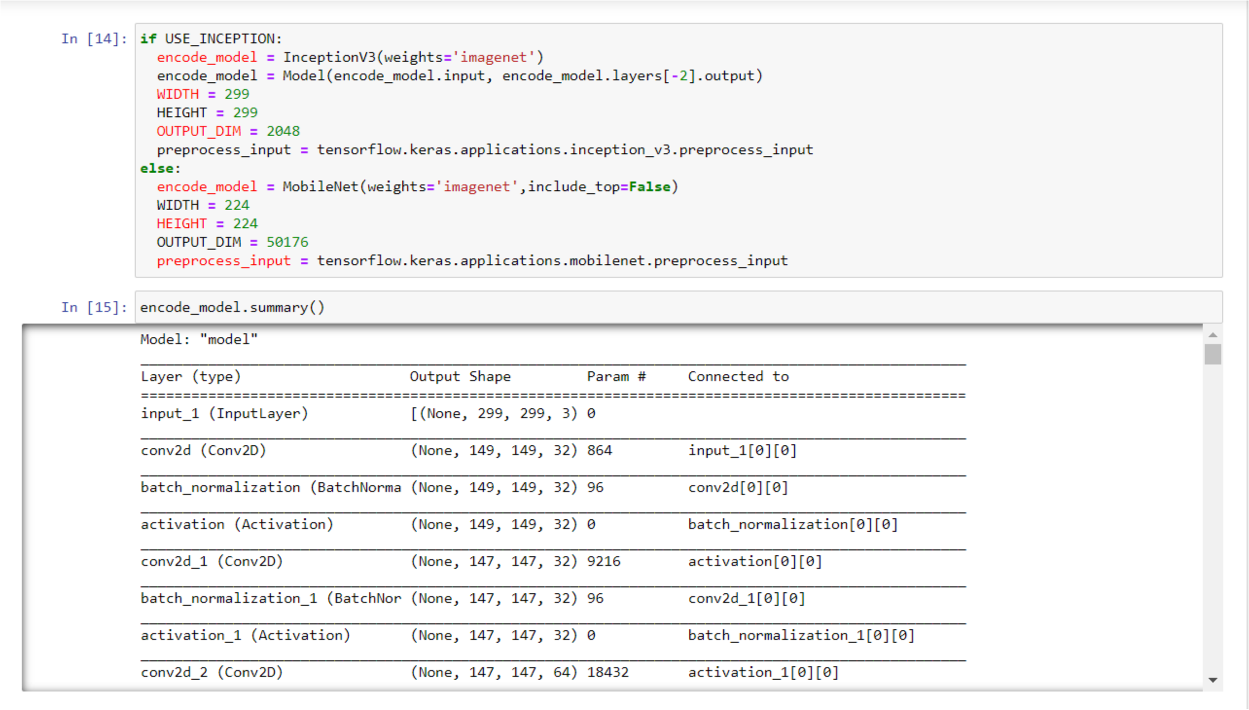
```
6000
1000
```

Dictionary with images as key and 5 captions as values

In [12]:
```
train_descriptions = {k:v for k,v in lookup.items() if f'{k}.jpg' in train_images}
for n,v in train_descriptions.items():
  for d in range(len(v)):
    v[d] = f'{START} {v[d]} {STOP}'
```

In [13]:
```
len(train_descriptions)
```

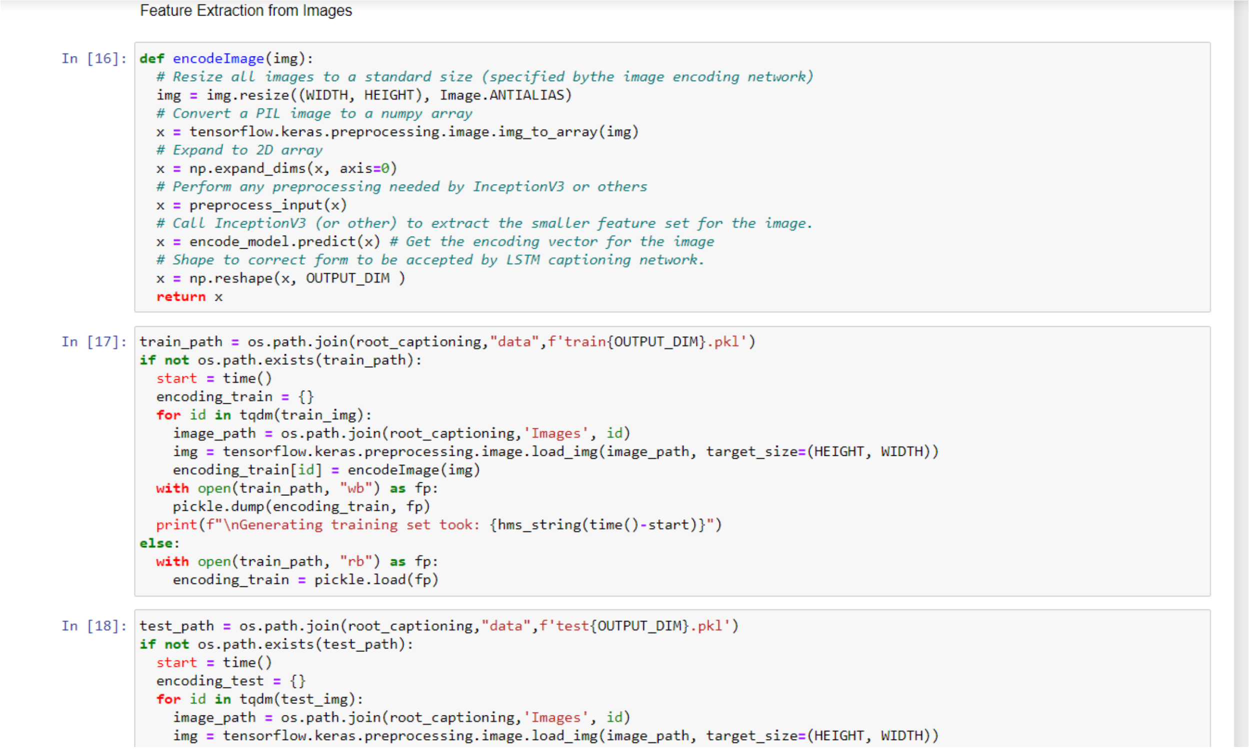
Out[13]:
```
6000
```



In [14]:
```
if USE_INCEPTION:
  encode_model = InceptionV3(weights='imagenet')
  encode_model = Model(encode_model.input, encode_model.layers[-2].output)
  WIDTH = 299
  HEIGHT = 299
  OUTPUT_DIM = 2048
  preprocess_input = tensorflow.keras.applications.inception_v3.preprocess_input
else:
  encode_model = MobileNet(weights='imagenet',include_top=False)
  WIDTH = 224
  HEIGHT = 224
  OUTPUT_DIM = 50176
  preprocess_input = tensorflow.keras.applications.mobilenet.preprocess_input
```

In [15]:
```
encode_model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 299, 299, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 149, 149, 32) 864         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 149, 149, 32) 96          conv2d[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 149, 149, 32) 0           batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 147, 147, 32) 9216        activation[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 147, 147, 32) 96          conv2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 147, 147, 32) 0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 147, 147, 64) 18432       activation_1[0][0]
```

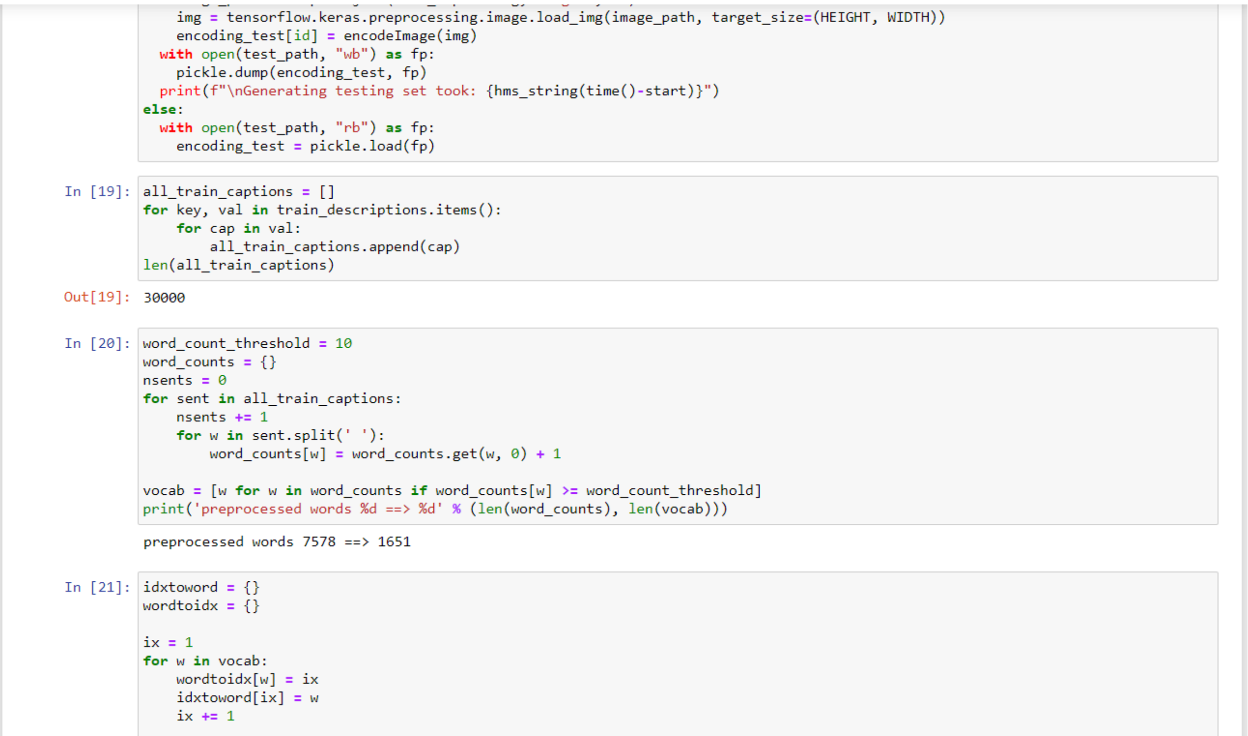


Feature Extraction from Images

In [16]:
```
def encodeImage(img):
  # Resize all images to a standard size (specified bythe image encoding network)
  img = img.resize((WIDTH, HEIGHT), Image.ANTIALIAS)
  # Convert a PIL image to a numpy array
  x = tensorflow.keras.preprocessing.image.img_to_array(img)
  # Expand to 2D array
  x = np.expand_dims(x, axis=0)
  # Perform any preprocessing needed by InceptionV3 or others
  x = preprocess_input(x)
  # Call InceptionV3 (or other) to extract the smaller feature set for the image.
  x = encode_model.predict(x) # Get the encoding vector for the image
  # Shape to correct form to be accepted by LSTM captioning network.
  x = np.reshape(x, OUTPUT_DIM )
  return x
```

In [17]:
```
train_path = os.path.join(root_captioning,"data",f'train{OUTPUT_DIM}.pkl')
if not os.path.exists(train_path):
  start = time()
  encoding_train = {}
  for id in tqdm(train_img):
    image_path = os.path.join(root_captioning,'Images', id)
    img = tensorflow.keras.preprocessing.image.load_img(image_path, target_size=(HEIGHT, WIDTH))
    encoding_train[id] = encodeImage(img)
  with open(train_path, "wb") as fp:
    pickle.dump(encoding_train, fp)
  print(f"\nGenerating training set took: {hms_string(time()-start)}")
else:
  with open(train_path, "rb") as fp:
    encoding_train = pickle.load(fp)
```

In [18]:
```
test_path = os.path.join(root_captioning,"data",f'test{OUTPUT_DIM}.pkl')
if not os.path.exists(test_path):
  start = time()
  encoding_test = {}
  for id in tqdm(test_img):
    image_path = os.path.join(root_captioning,'Images', id)
    img = tensorflow.keras.preprocessing.image.load_img(image_path, target_size=(HEIGHT, WIDTH))
```



```
    img = tensorflow.keras.preprocessing.image.load_img(image_path, target_size=(HEIGHT, WIDTH))
    encoding_test[id] = encodeImage(img)
  with open(test_path, "wb") as fp:
    pickle.dump(encoding_test, fp)
  print(f"\nGenerating testing set took: {hms_string(time()-start)}")
else:
  with open(test_path, "rb") as fp:
    encoding_test = pickle.load(fp)
```

In [19]:

```
all_train_captions = []
for key, val in train_descriptions.items():
    for cap in val:
        all_train_captions.append(cap)
len(all_train_captions)
```

Out[19]:

```
30000
```

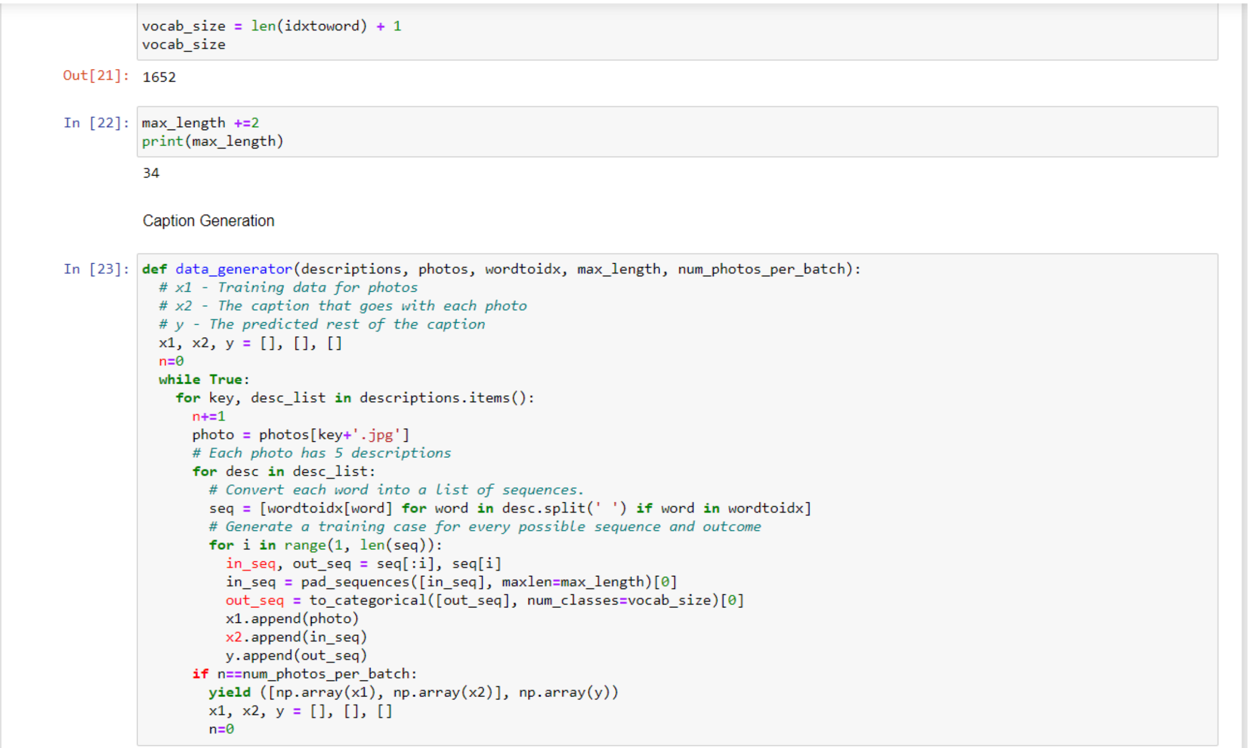
In [20]:

```
word_count_threshold = 10
word_counts = {}
nsents = 0
for sent in all_train_captions:
    nsents += 1
    for w in sent.split(' '):
        word_counts[w] = word_counts.get(w, 0) + 1

vocab = [w for w in word_counts if word_counts[w] >= word_count_threshold]
print('preprocessed words %d ==> %d' % (len(word_counts), len(vocab)))
```

```
preprocessed words 7578 ==> 1651
```

In [21]:

```
idxtoword = {}
wordtoidx = {}

ix = 1
for w in vocab:
    wordtoidx[w] = ix
    idxtoword[ix] = w
    ix += 1
```



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

Out[21]:

```
1652
```

In [22]:

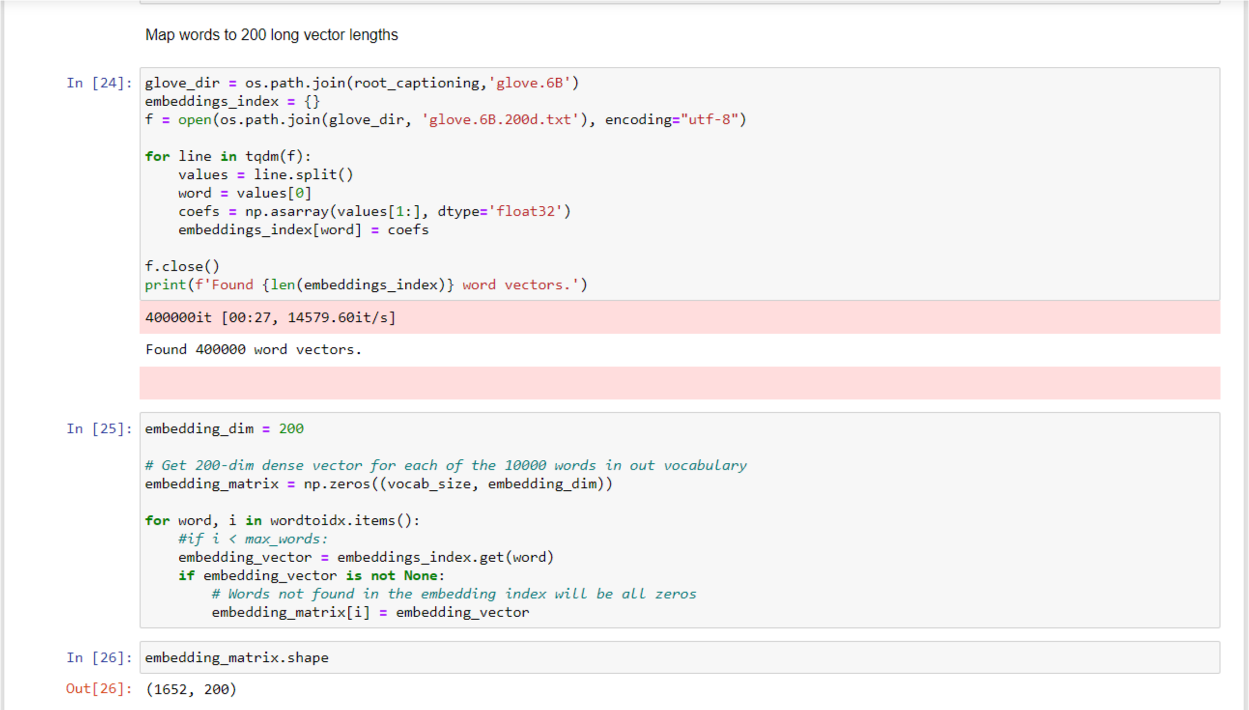
```
max_length +=2
print(max_length)
```

```
34
```

Caption Generation

In [23]:

```
def data_generator(descriptions, photos, wordtoidx, max_length, num_photos_per_batch):
  # x1 - Training data for photos
  # x2 - The caption that goes with each photo
  # y - The predicted rest of the caption
  x1, x2, y = [], [], []
  n=0
  while True:
    for key, desc_list in descriptions.items():
      n+=1
      photo = photos[key+'.jpg']
      # Each photo has 5 descriptions
      for desc in desc_list:
        # Convert each word into a list of sequences.
        seq = [wordtoidx[word] for word in desc.split(' ') if word in wordtoidx]
        # Generate a training case for every possible sequence and outcome
        for i in range(1, len(seq)):
          in_seq, out_seq = seq[:i], seq[i]
          in_seq = pad_sequences([in_seq], maxlen=max_length)[0]
          out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]
          x1.append(photo)
          x2.append(in_seq)
          y.append(out_seq)
      if n==num_photos_per_batch:
        yield ([np.array(x1), np.array(x2)], np.array(y))
        x1, x2, y = [], [], []
        n=0
```



Map words to 200 long vector lengths

In [24]:
```
glove_dir = os.path.join(root_captioning,'glove.6B')
embeddings_index = {}
f = open(os.path.join(glove_dir, 'glove.6B.200d.txt'), encoding="utf-8")

for line in tqdm(f):
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs

f.close()
print(f'Found {len(embeddings_index)} word vectors.')
```

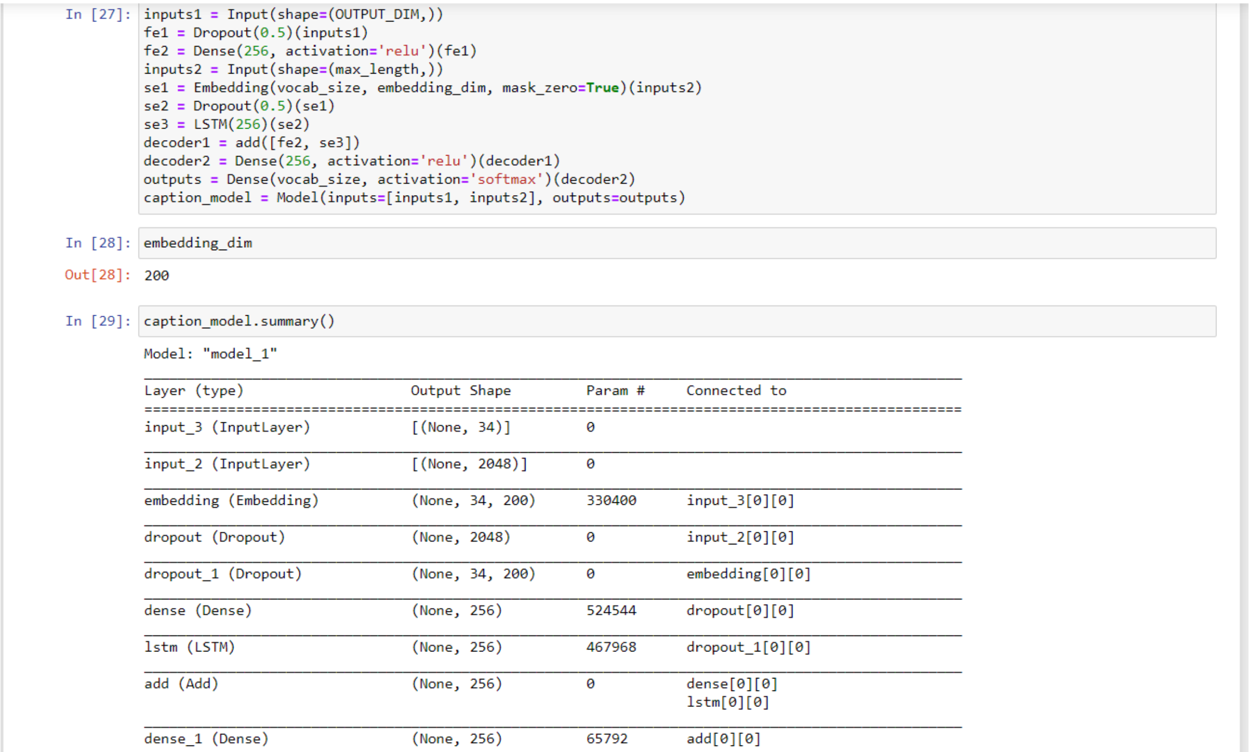
```
400000it [00:27, 14579.60it/s]
```

```
Found 400000 word vectors.
```

In [25]:
```
embedding_dim = 200

# Get 200-dim dense vector for each of the 10000 words in out vocabulary
embedding_matrix = np.zeros((vocab_size, embedding_dim))

for word, i in wordtoidx.items():
    #if i < max_words:
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # Words not found in the embedding index will be all zeros
        embedding_matrix[i] = embedding_vector
```

In [26]:
```
embedding_matrix.shape
```

Out[26]:
```
(1652, 200)
```



In [27]:
```
inputs1 = Input(shape=(OUTPUT_DIM,))
fe1 = Dropout(0.5)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)
inputs2 = Input(shape=(max_length,))
se1 = Embedding(vocab_size, embedding_dim, mask_zero=True)(inputs2)
se2 = Dropout(0.5)(se1)
se3 = LSTM(256)(se2)
decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)
caption_model = Model(inputs=[inputs1, inputs2], outputs=outputs)
```

In [28]:
```
embedding_dim
```

Out[28]:
```
200
```

In [29]:
```
caption_model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 34)]         0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 2048)]       0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 34, 200)      330400      input_3[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 2048)         0           input_2[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 34, 200)      0           embedding[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          524544      dropout[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 256)          467968      dropout_1[0][0]
__________________________________________________________________________________________________
add (Add)                       (None, 256)          0           dense[0][0]
                                                                 lstm[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          65792       add[0][0]
```

```
dense_1 (Dense)                 (None, 256)          65792       add[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 1652)         424564      dense_1[0][0]
==================================================================================================
Total params: 1,813,268
Trainable params: 1,813,268
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [30]:

```
caption_model.layers[2].set_weights([embedding_matrix])
caption_model.layers[2].trainable = False
caption_model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
```

In [31]:

```
number_pics_per_bath = 3
steps = len(train_descriptions)//number_pics_per_bath
```

In [32]:

```
model_path = os.path.join(root_captioning,"data",f'caption-model.hdf5')
start = time()
if not os.path.exists(model_path):
  for i in tqdm(range(EPOCHS*2)):
      generator = data_generator(train_descriptions, encoding_train, wordtoidx, max_length, number_pics_per_bath)
      caption_model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)

  caption_model.optimizer.lr = 1e-4
  number_pics_per_bath = 6
  steps = len(train_descriptions)//number_pics_per_bath

  for i in range(EPOCHS):
      generator = data_generator(train_descriptions, encoding_train, wordtoidx, max_length, number_pics_per_bath)
      caption_model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
  caption_model.save_weights(model_path)
  print(f"\Training took: {hms_string(time()-start)}")
else:
  caption_model.load_weights(model_path)
```

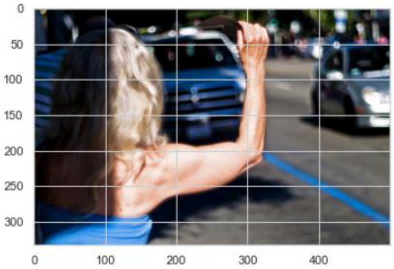
In [33]:

```
def generateCaption(photo):
    in_text = START
    for i in range(max_length):
        sequence = [wordtoidx[w] for w in in_text.split() if w in wordtoidx]
        sequence = pad_sequences([sequence], maxlen=max_length)
        yhat = caption_model.predict([photo,sequence], verbose=0)
        yhat = np.argmax(yhat)
        word = idxtoword[yhat]
        in_text += ' ' + word
        if word == STOP:
            break
    final = in_text.split()
    final = final[1:-1]
    final = ' '.join(final)
    return final
```

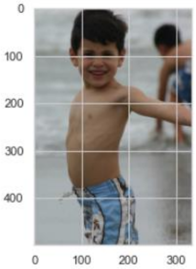
In [34]:

```
for z in range(5):
  pic = list(encoding_test.keys())[z]
  image = encoding_test[pic].reshape((1,OUTPUT_DIM))
  print(os.path.join(root_captioning,'Images', pic))
  x=plt.imread(os.path.join(root_captioning,'Images', pic))
  plt.imshow(x)
  plt.show()
  print("Caption:",generateCaption(image))
  print("_____________________________________")
```

```
C:\Users\anujj\Desktop\Capstone Project\archive\Images\1056338697_4f7d7ce270.jpg
```



```
Caption: man in blue shirt and jeans is standing next to an
_____________________________________
C:\Users\anujj\Desktop\Capstone Project\archive\Images\106490881_5a2dd9b7bd.jpg
```



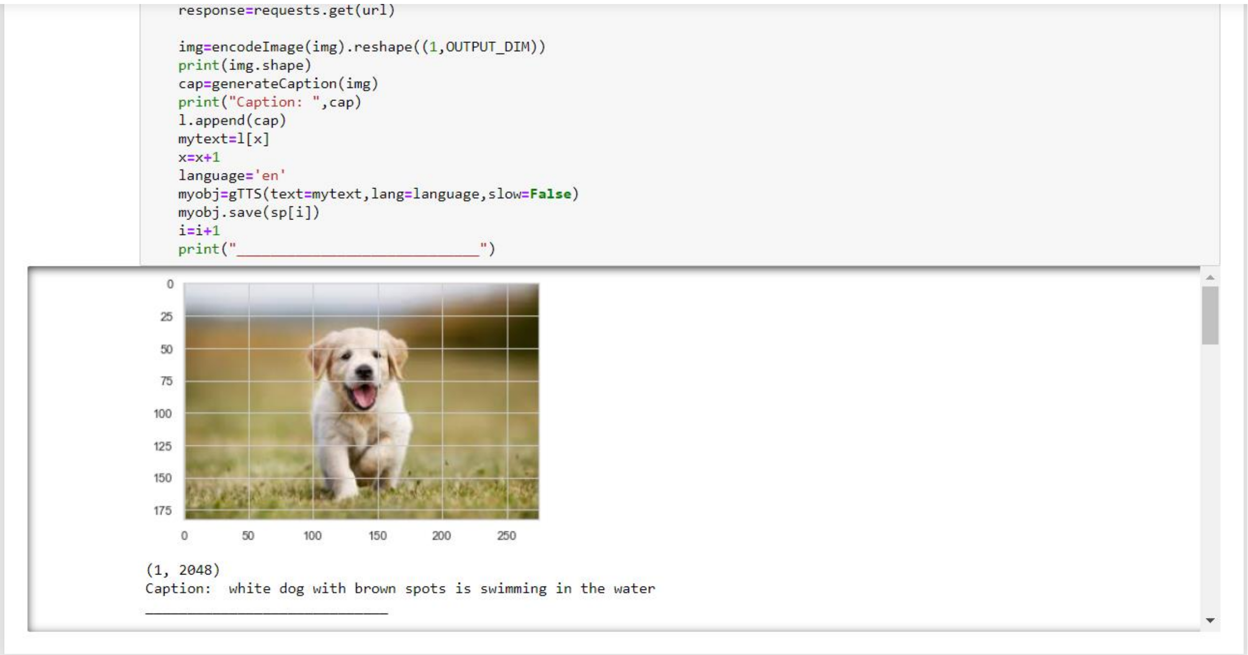
```
Caption: boy in blue shorts is running through the water
```

In [35]:

```python
from PIL import Image,ImageFile
from matplotlib.pyplot import imshow
import requests
from io import BytesIO
import numpy as np
import io

%matplotlib inline

urls=[
    "https://github.com/anujjhunjhunwala98/capstone/blob/main/1.jpg?raw=true",
    "https://github.com/anujjhunjhunwala98/capstone/blob/main/2.jpg?raw=true",
    "https://github.com/anujjhunjhunwala98/capstone/blob/main/3.jpg?raw=true",
    "https://github.com/anujjhunjhunwala98/capstone/blob/main/4.jpg?raw=true",
    "https://github.com/anujjhunjhunwala98/capstone/blob/main/5.jpg?raw=true",
    "https://github.com/anujjhunjhunwala98/capstone/blob/main/6.jpg?raw=true"
]
```

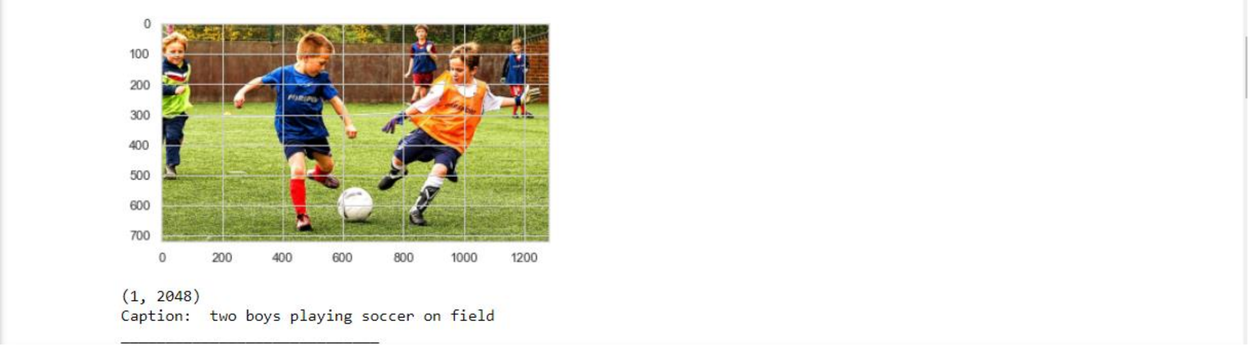
In [36]:

```python
l=[]
x=0
sp=["speech1.mp3","speech2.mp3","speech3.mp3","speech4.mp3","speech5.mp3","speech6.mp3"]
i=0
for url in urls:
    response=requests.get(url)
    img=Image.open(BytesIO(response.content))
    img.load()

    plt.imshow(img)
    plt.show()

    response=requests.get(url)

    img=encodeImage(img).reshape((1,OUTPUT_DIM))
    print(img.shape)
    cap=generateCaption(img)
    print("Caption: ",cap)
    l.append(cap)
    mytext=l[x]
```

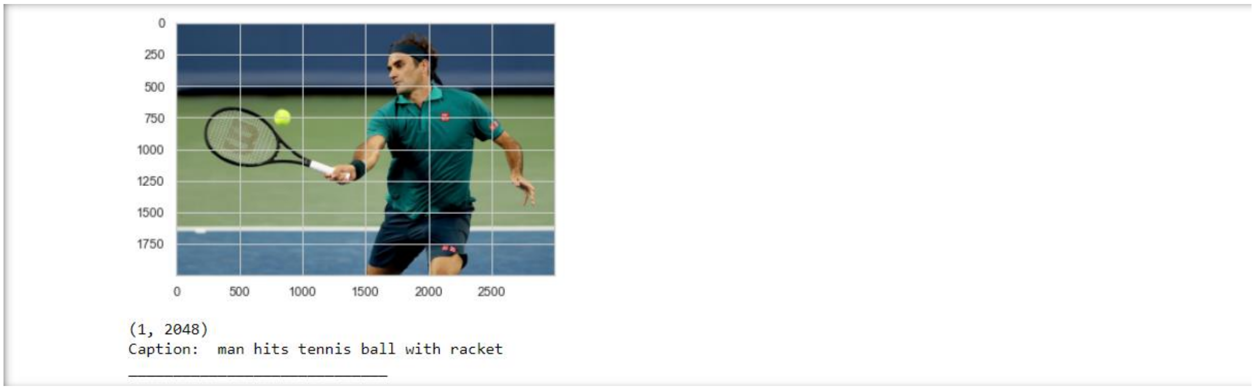


```
    response=requests.get(url)

    img=encodeImage(img).reshape((1,OUTPUT_DIM))
    print(img.shape)
    cap=generateCaption(img)
    print("Caption: ",cap)
    l.append(cap)
    mytext=l[x]
    x=x+1
    language='en'
    myobj=gTTS(text=mytext,lang=language,slow=False)
    myobj.save(sp[i])
    i=i+1
    print("_____________________________")
```

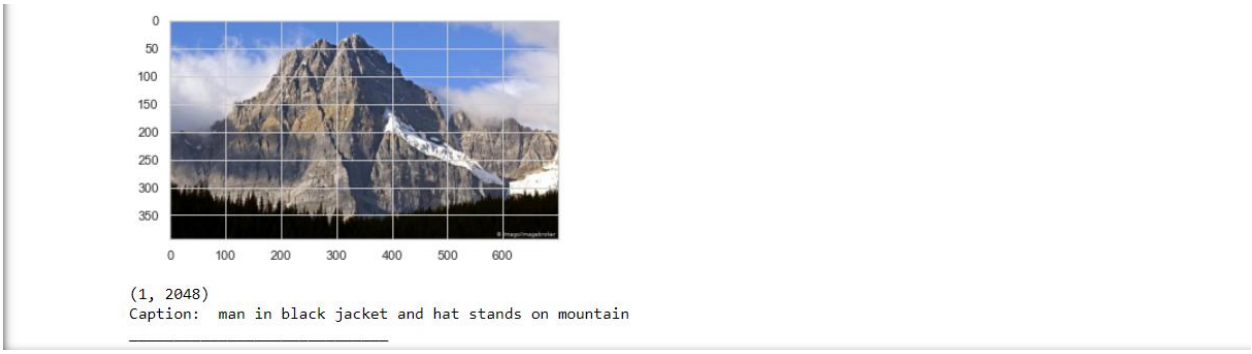
```
(1, 2048)
Caption:  white dog with brown spots is swimming in the water
_____________________________
```



```
(1, 2048)
Caption:  two boys playing soccer on field
_____________________________
```

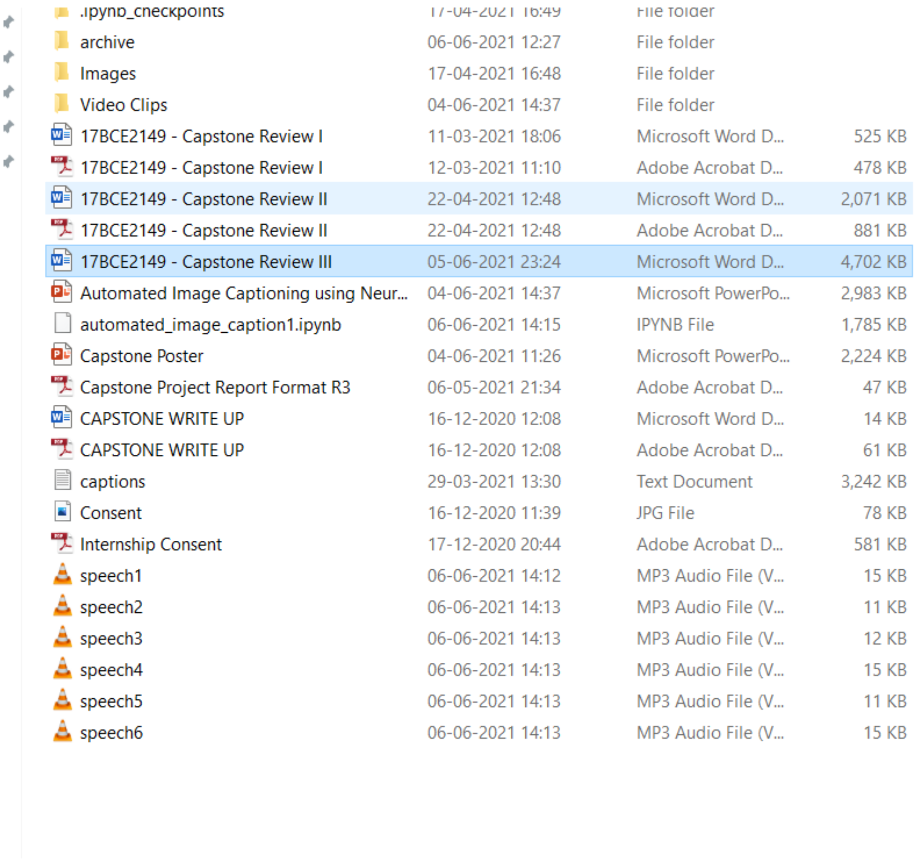


```
(1, 2048)
Caption:  man hits tennis ball with racket
_____________________________
```



```
(1, 2048)
Caption:  man in black jacket and hat stands on mountain
```

Speech Stored as MP3 files on disk



.ipynb_checkpoints	17-04-2021 16:49	File folder	
archive	06-06-2021 12:27	File folder	
Images	17-04-2021 16:48	File folder	
Video Clips	04-06-2021 14:37	File folder	
17BCE2149 - Capstone Review I	11-03-2021 18:06	Microsoft Word D...	525 KB
17BCE2149 - Capstone Review I	12-03-2021 11:10	Adobe Acrobat D...	478 KB
17BCE2149 - Capstone Review II	22-04-2021 12:48	Microsoft Word D...	2,071 KB
17BCE2149 - Capstone Review II	22-04-2021 12:48	Adobe Acrobat D...	881 KB
17BCE2149 - Capstone Review III	05-06-2021 23:24	Microsoft Word D...	4,702 KB
Automated Image Captioning using Neur...	04-06-2021 14:37	Microsoft PowerPo...	2,983 KB
automated_image_caption1.ipynb	06-06-2021 14:15	IPYNB File	1,785 KB
Capstone Poster	04-06-2021 11:26	Microsoft PowerPo...	2,224 KB
Capstone Project Report Format R3	06-05-2021 21:34	Adobe Acrobat D...	47 KB
CAPSTONE WRITE UP	16-12-2020 12:08	Microsoft Word D...	14 KB
CAPSTONE WRITE UP	16-12-2020 12:08	Adobe Acrobat D...	61 KB
captions	29-03-2021 13:30	Text Document	3,242 KB
Consent	16-12-2020 11:39	JPG File	78 KB
Internship Consent	17-12-2020 20:44	Adobe Acrobat D...	581 KB
speech1	06-06-2021 14:12	MP3 Audio File (V...	15 KB
speech2	06-06-2021 14:13	MP3 Audio File (V...	11 KB
speech3	06-06-2021 14:13	MP3 Audio File (V...	12 KB
speech4	06-06-2021 14:13	MP3 Audio File (V...	15 KB
speech5	06-06-2021 14:13	MP3 Audio File (V...	11 KB
speech6	06-06-2021 14:13	MP3 Audio File (V...	15 KB