Jaypee Institute of Information Technology

Minor Project Report

Project Title:

Image Caption Generation BOT

Presented By:

Shivam Rajpoot
(17803004)

Ashutosh Singh
(17803010)

Shourya Saraogi
(17803019)

Supervisor:

Taj Alam Sir

Table of content:

DECLARATION BY THE SCHOLAR

“We hereby declare that the work reported in the B.Tech. project report entitled“Image Caption Generation Bot”submitted at Jaypee Institute of Information Technology, Noida, India, is an authentic record of our work carried out under the supervision of Dr. Taj Alam. We have not submitted this work elsewhere for any other degree or diploma. We are fully responsible for the contents of our B.Tech. project report.”

Ashutosh Singh

Department of Computer Science Engineering,

Jaypee Institute of Information Technology, Noida, India

Date: ....

Shivam Rajpoot

Department of Computer Science Engineering,

Jaypee Institute of Information Technology, Noida, India

Date: ....

Shourya Saraogi

Department of Computer Science Engineering,

Jaypee Institute of Information Technology, Noida, India

Date:.....................

SUPERVISOR’S CERTIFICATE

This is to certify that the work reported in the B.Tech. project report entitled “Image Caption Generation Bot”, submitted by Ashutosh Singh , Shivam Rajpoot, Shourya Saraogi at Jaypee Institute of Information Technology, Noida, India, is a bonafide record of their original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

(Signature of the Supervisor)

(Dr. Taj Alam)

Assistant Professor (Senior Grade),

CSE Department,

JIIT, Noida (U.P.)

INDIA

Date: ....

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We am highly indebted to Dr. Taj Alam (our project supervisor) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express my gratitude towards my parents & member of Coding Ninjas for their kind co-operation and encouragement which help me in completion of this project.

We would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

Out thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

Abstract :

This paper presents a model that generates captions or descriptions for images with the help of multimodal neural networks. The model consists of two subnetworks a convolution neural network that is utilized to extract the image characteristics and a recurrent neural network for the descriptions. These sub-networks are then aligned through a multimodal embedding to form the whole model. .As compared to previous models such as temporal convolution, the recurrent models are considered to be doubly deep. These recurrent neural networks can handle variable input/output. Thus, they can directly map a variable input (e.g. video) with a variable output (e.g. a caption/description, natural language text). The results distinctly show its advantage over state-of-the-art models that are used for generating descriptions or captioning images.

In the past few years, the problem of generating descriptive sentences automatically for images has garnered a rising interest in natural language processing and computer vision research. Image captioning is a fundamental task which requires semantic understanding of images and the ability of generating description sentences with proper and correct structure. In this study, the authors propose a hybrid system employing the use of multi-layer Convolutional Neural Network (CNN) to generate vocabulary describing the images and a Long Short Term Memory (LSTM) to accurately structure meaningful sentences using the generated keywords. The convolutional neural network compares the target image to a large dataset of training images, then generates an accurate description using the trained captions. We showcase the efficiency of our proposed model using the Flickr8K and Flickr30K datasets and show that their model gives superior results compared with the state-of-the-art models utilising the Bleu metric. The Bleu metric is an algorithm for evaluating the performance of a machine translation system by grading the quality of text translated from one natural language to another.

Synopsis :

The image captioning problem and its proposed solutions have existed since the advent of the Internet and its widespread adoption as a medium to share images. Numerous algorithms and techniques have been put forward by researchers from different perspectives. Krizhevsky et al. Implemented a neural network using non-saturating neurons and a very efficient a unique method GPU implementation of the convolution function. By employing a regularization method called dropout, they succeeded in reducing overfitting. Their neural network consisted of maxpooling layers and a final 1000-way softmax. Deng et al. Introduced a new database which they called ImageNet, an extensive collection of images built using the core of the WordNet structure. ImageNet organized the different classes of images in a densely populated semantic hierarchy. Karpathy and FeiFei made use of datasets of images and their sentence descriptions to learn about the inner correspondences visual data and language. Their work described a Multimodal Recurrent Neural Network architecture that utilises the inferred co-linear arrangement of features in order to learn how to generate novel descriptions of images. Yang et al. proposed a system for the automatic generation of a natural language description of an image, which will help immensely in furthering image understanding. The proposed multimodel neural network method, consisting of object detection and localization modules, is very similar to the human visual system which is able to learns how to describe the content of images automatically. In order to address the problem of LSTM units being complex and inherently sequential across time, Aneja et al. proposed a convolutional network model for machine translation and conditional image generation. Pan et. Al   experimented extensively with multiple network architectures on large datasets consisting of varying content styles, and proposed a unique model showing noteworthy improvement on captioning accuracy over the previously proposed models. Vinyals et al. presented a generative model consisting of a deep recurrent architecture that leverages machine translation and computer vision, used to generate natural descriptions of an image by ensuring highest probability of the generated sentence to accurately describe the target image. Xu et al. Introduced an attention based model that learned to describe the image regions automatically. The model was trained using standard backpropagation techniques by maximizing a variable lower bound. The model was able to automatically learn identify object boundaries while at the same time generate an accurate descriptive sentence.

CHAPTER -1

INTRODUCTION

1.1 About:

Caption generation is an interesting artificial intelligence problem where a descriptive sentence is generated for a given image. It involves the dual techniques from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. Image captioning has various applications such as recommendations in editing applications, usage in virtual assistants, for image indexing, for visually impaired persons, for social media, and several other natural language processing applications. Recently, deep learning methods have achieved state-ofthe-art results on examples of this problem. It has been demonstrated that deep learning models are able to achieve optimum results in the field of caption generation problems. Instead of requiring complex data preparation or a pipeline of specifically designed models, a single end-to-end model can be defined to predict a caption, given a photo. In order to evaluate our model, we measure its performance on the Flickr8K dataset using the BLEU standard metric. These results show that our proposed model performs better than standard models regarding image captioning in performance evaluation.

The limitations of neural networks are determined mostly by the amount of memory available on the GPUs used to train the network as well as the duration of training time it is allowed. Our network takes around seven days to train on GTX 1050 4GB and GTX 760 GPUs. According to our results, our results can be improved by utilising faster and larger GPUs and more exhaustive datasets.

This work is an attempt towards achieving the goal of generating apt descriptions of images. The primary challenge towards this goal is in the design of a model that is rich enough to simultaneously reason about contents of images and their representation in the domain of natural language. The model should be free of assumptions about specific hard-coded templates, rules or categories and instead rely on learning from the training data. The second, practical challenge is that datasets of image captions are available in large quantities on the internet [21, 16, 24], but these descriptions multiplex mentions of several entities whose locations in the images are unknown. To overcome these problems the deep neural network model is modeled in such a way that it infers the latent alignment between segments of sentences and the region of the image that they describe. The model associates the two modalities through a common, multimodal embedding space and a structured objective.

The proposed multimodal Recurrent Neural Networks (mRNN) model carries out two tasks. Firstly, it generates novel descriptions for the images. Secondly it also carries out the task of image and sentence retrieval. This m-RNN architecture contains an image part, a language model part and a multimodal part. The image part uses a deep Convolution Neural Network (CNN) to extract the image characteristics while the language model part learns the dense feature embedding for each word in the dictionary and maintains the semantic temporal context in recurrent layers. The structured objective or the multimodal part connects the language model and the deep CNN together by a one-layer representation. This m-RNN model is learned using a perplexity based cost function. The model parameters are updated by back-propagating the error to the three layers of the m-RNN model.In the experiments, the model is validated on two benchmark datasets: Flickr 8K [12], and Flickr 30K . The model has potential extensions and it can further be improved by integrating more powerful neural networks .

1.2 MOTIVATION

We must first understand how important this problem is to real world scenarios. Let’s see few applications where a solution to this problem can be very useful.

- Self driving cars — Automatic driving is one of the biggest challenges and if we can properly caption the scene around the car, it can give a boost to the self driving system.

- Aid to the blind — We can create a product for the blind which will guide them travelling on the roads without the support of anyone else. We can do this by first converting the scene into text and then the text to voice. Both are now famous applications of Deep Learning.

CCTV cameras are everywhere today, but along with viewing the world, if we can also generate relevant captions, then we can raise alarms as soon as there is some malicious activity going on somewhere. This could probably help reduce some crime and/or accidents.

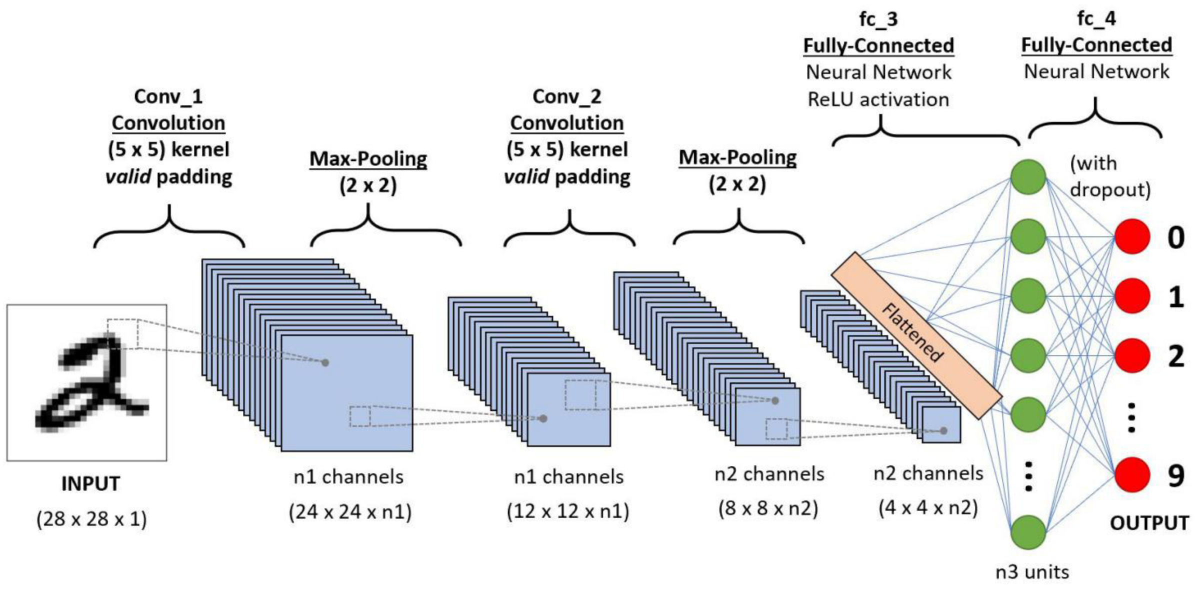
Automatic Captioning can help, make Google Image Search as good as Google Search, as then every image could be first converted into a caption and then search can be performed based on the caption.

1.3 PREREQUISITES:

This post assumes familiarity with basic Deep Learning concepts like Multi-layered Perceptrons, Convolution Neural Networks, Recurrent Neural Networks, Transfer Learning, Gradient Descent, Backpropagation, Overfitting, Probability, Text Processing, Python syntax and data structures, Keras library, etc.

1.3.1 INTRODUCTION TO CNN:

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used. CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.



1.3.2 INTRODUCTION TO RNN:

Recurrent Neural Networks or RNN as they are called in short, are a very important variant of neural networks heavily used in Natural Language Processing. In a general neural network, an input is processed through a number of layers and an output is produced, with an assumption that two successive inputs are independent of each other.

This assumption is however not true in a number of real-life scenarios. For instance, if one wants to predict the price of a stock at a given time or wants to predict the next word in a sequence it is imperative that dependence on previous observations is considered.

CHAPTER – 2

DATA COLLECTION AND CLEANING

2.1 DATA COLLECTION :

There are many open source datasets available for this problem, like Flickr 8k (containing8k images), Flickr 30k (containing 30k images), MS COCO (containing 180k images), etc.
But for the purpose of this case study, I have used the Flickr 30k dataset.
This dataset contains 30000 images each with 5 captions (as we have already seen in the Introduction section that an image can have multiple captions, all being relevant simultaneously).
These images are bifurcated as follows:

- Training Set — 29000 images
- Test Set — 1000 images
- Result.csv file

Why Flickr30k dataset…?

1. It is small in size. So, the model can be trained easily on low-end laptops/desktops...

2. Data is properly labelled. For each image 5 captions are provided.

3. The dataset is available for free.

Data pre-processing and cleaning is an important part of the whole model building process. Understanding the data helps us to build more accurate models.

After extracting zip files you will find below folders...

1. Flickr30k_Dataset: Contains a total of 30000 images in JPEG format with different shapes and sizes. Of which 29000 are used for training, 1000 for test.

2. Result.csv : Contains text files describing train set ,test set. Flickr30k.token.txt contains 5 captions for each image i.e. total 158916 captions.

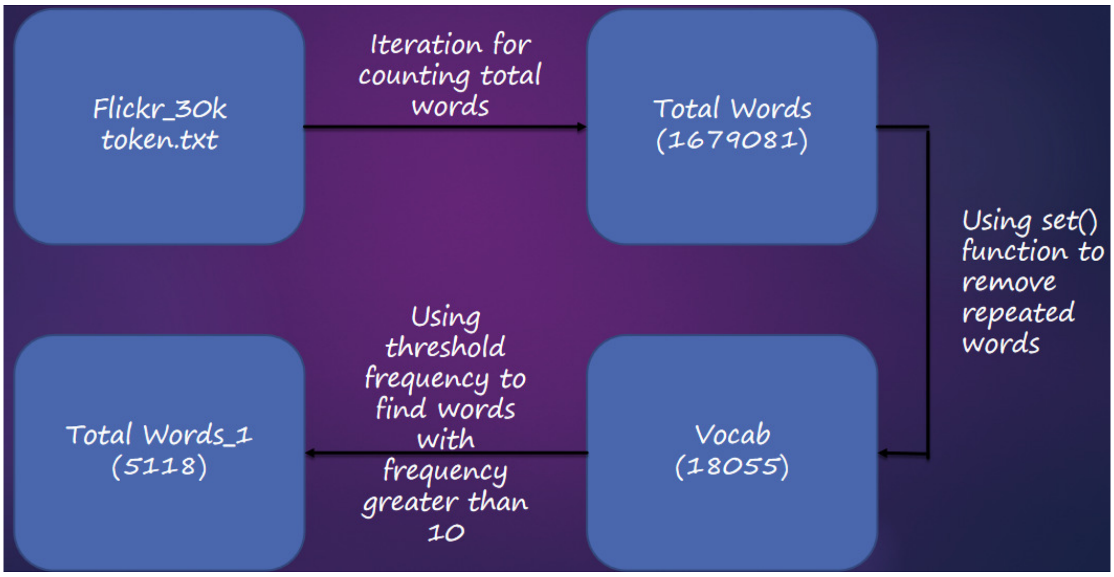
We have mainly two types of data.

1. Images
2. Captions (Text)

2.2 DATA CLEANING

For data cleaning first we have downloaded dataset in that dataset we have flicker30k_token.txt file we will use that and after that we have make a function and that function will take input a sentence and remove all special character , numbers and return only those words which belongs to a to z and after that we will convert all words to small character. And then we will save them into a file after converting into json string file as I have done and named "**descriptions_1**" text file. Now we don't want repeated term so we will remove them using **set()** function it's a data structure and in most of the cases it's complexity is **O(1).** Actually it removes repeated words so we get something around **"18055"** words in dictionary named **"vocab"** in this project**.**

After this step now many words are just coming once or twice so we don't want them so for that we will count every words frequency and for that we will make a dictionary . After making dictionary we will set threshold frequency of count 10 because we don't want that words which is coming less than 10 times in the total words and we get total 5118 words whose frequency is more than 10 and we will store in **total_words** variable.


Flickr_30k token.txt
Iteration for counting total words
Total Words (1679081)
Using set() function to remove repeated words
Vocab (18055)
Using threshold frequency to find words with frequency greater than 10
Total Words_1 (5118)

Chapter 3

Data Processing

Introduction:

Since we have already cleaned our caption folder and created a dictionary of “1845” word whose frequency is greater than 10 so now we cannot directly use these word for model training so first we have to add two important words “start sequence” and “end sequence” into our dictionary because of smooth training of our model. After that we will convert every word into vector form and for that we will use “glove.6b.50d”.

GLOVE VECTOR:

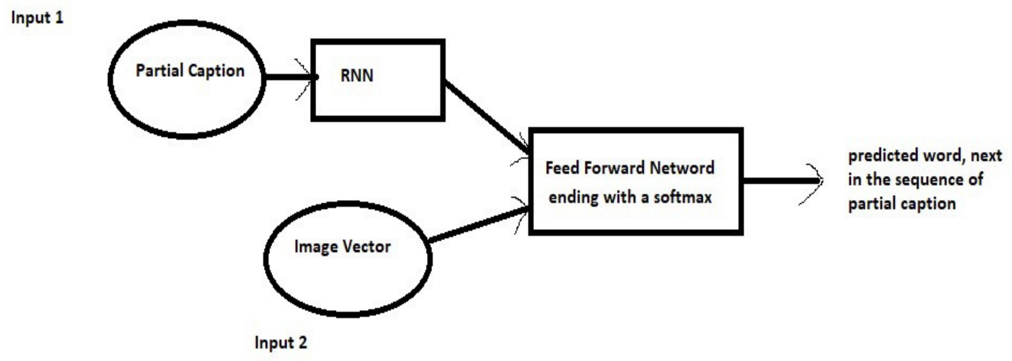
The similarity metrics used for nearest neighbour evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, *man* may be regarded as similar to *woman* in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

In order to capture in a quantitative way the nuance necessary to distinguish *man* from *woman*, it is necessary for a model to associate more than a single number to the word pair. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two word vectors. Glove is designed in order that such vector differences capture as much as possible the meaning specified by the juxtaposition of two words.

Representation of glove vector of apple:

```
[array([ 0.52042 , -0.8314  ,  0.49961 ,  1.2893  ,  0.1151  ,  0.
057521,
      -1.3753  , -0.97313 ,  0.18346 ,  0.47672 , -0.15112 ,  0.3
5532 ,
       0.25912 , -0.77857 ,  0.52181 ,  0.47695 , -1.4251  ,  0.8
58   ,
       0.59821 , -1.0903  ,  0.33574 , -0.60891 ,  0.41742 ,  0.2
1569 ,
      -0.07417 , -0.5822  , -0.4502  ,  0.17253 ,  0.16448 , -0.3
8413 ,
       2.3283  , -0.66682 , -0.58181 ,  0.74389 ,  0.095015, -0.4
7865 ,
      -0.84591 ,  0.38704 ,  0.23693 , -1.5523  ,  0.64802 , -0.1
6521 ,
      -1.4719  , -0.16224 ,  0.79857 ,  0.97391 ,  0.40027 , -0.2
1912 ,
      -0.30938 ,  0.26581 ])]
```

After adding “startseq” and “endseq” in both the dictionary (word_to_idx and idx_to_word). Now we will calculate caption having maximum length so that our model knows where to start and where to end . We have already preserve zeroth element of total_words_1 because if captions length is less than maximum caption length we will append zeroth element to each caption .


Input 1
Partial Caption
RNN
Feed Forward Netword ending with a softmax
predicted word, next in the sequence of partial caption
Image Vector
Input 2

We must note that captions are something that we want to predict. So during the training period, captions will be the target variables (Y) that the model is learning to predict.

But the prediction of the entire caption, given the image does not happen at once. We will predict the caption **word by word**. Thus, we need to encode each word into a fixed sized vector. However, this part will be seen later when we look at the model design, but for now we will create two Python Dictionaries namely “wordtoix” (pronounced — word to index) and “ixtoword” (pronounced — index to word).

Stating simply, we will represent every unique word in the vocabulary by an integer (index). As seen above, we have 1845 unique words in the corpus and thus each word will be represented by an integer index between 1 to 1845.

These two Python dictionaries can be used as follows:

wordtoix[‘abc’] -> returns index of the word ‘abc’

ixtoword[k] -> returns the word whose index is ‘k’

The code used is as below:

#here we are going to maping number with words and we know that each line can be of different length so to make length and

#later we will append zeroth index to make length same.

```
word_to_idx = {}

idx_to_word = {}

for i,word in enumerate(total_words):

    word_to_idx[word] = i+1 # here we are using i+1 to reserve index
0

    idx_to_word[i+1] = word
```

we will also assign every idx for every word:

```
#two special words that is "startseq" and "endseq" that i have to add
in my dictionary



word_to_idx["startseq"] = 1846

idx_to_word[1846] = "Startseq"



word_to_idx["endseq"] = 1847

idx_to_word[1847] = "endseq"



vocab_size = len(word_to_idx) + 1

print("vocab_size:" , vocab_size)
```

There is one more parameter that we need to calculate, i.e., the maximum length of a caption and we do it as below:

```
max_len = 0
for key in train_descriptions.keys():
    for cap in train_descriptions[key]:
        max_len = max(max_len,len(cap.split()))

print(max_len)
```

So the maximum length of any caption is 74.

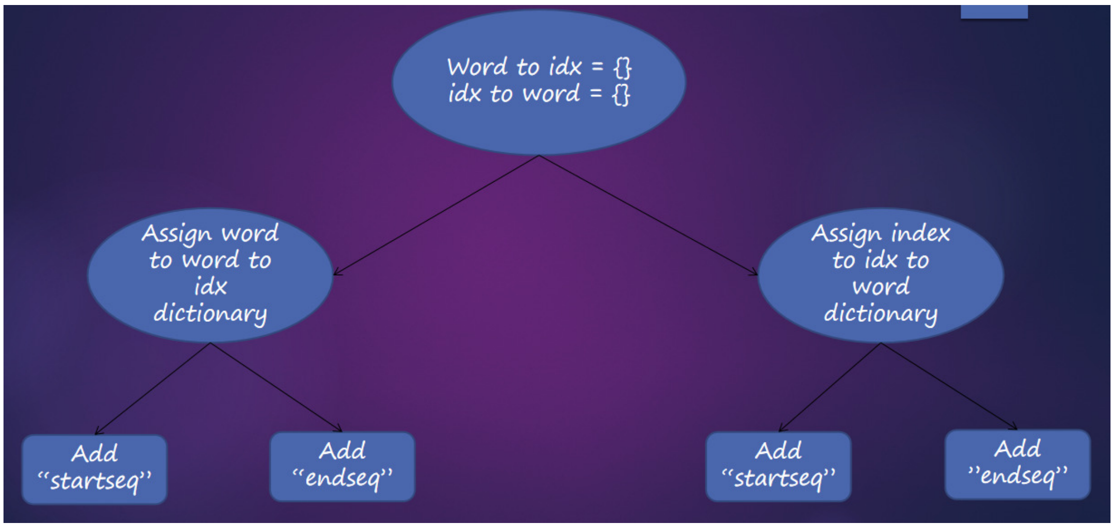
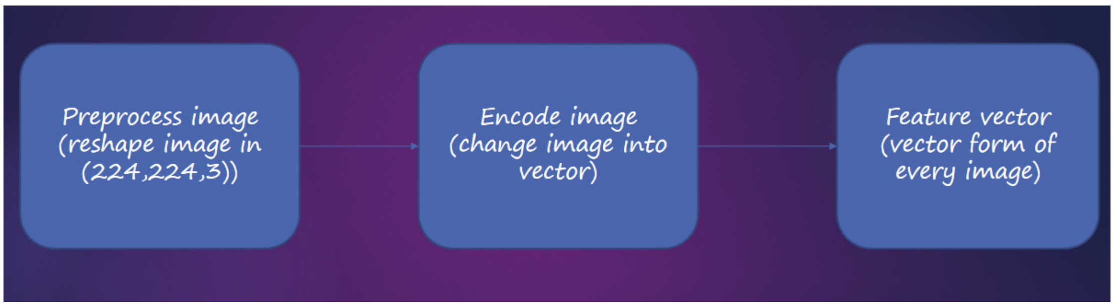


Image Processing :


Preprocess image (reshape image in (224,224,3))
Encode image (change image into vector)
Feature vector (vector form of every image)

CHAPTER – 4

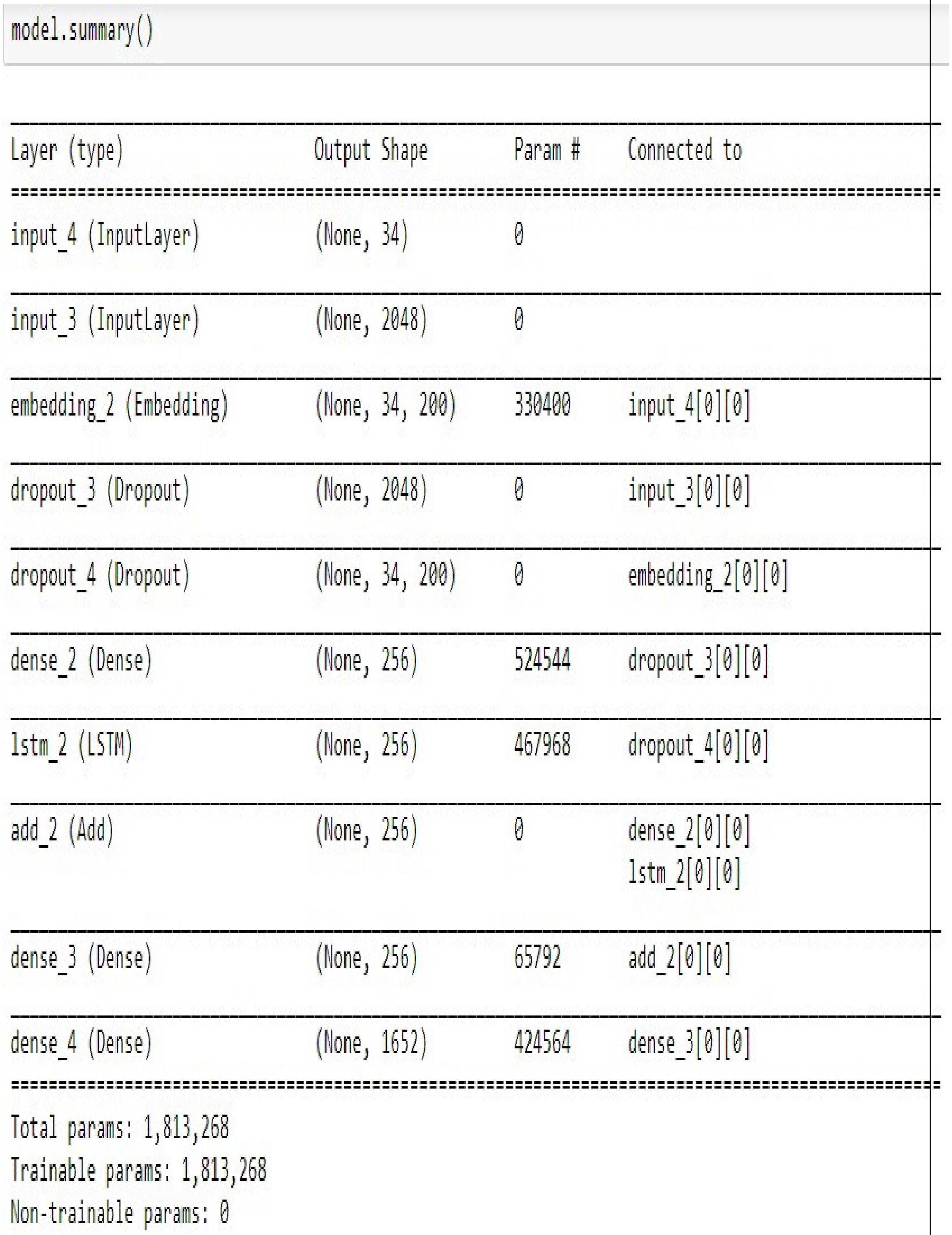
MODEL ARCHITECTURE

INTRODUCTION

Since the input consists of two parts, an image vector and a partial caption, we cannot use the Sequential API provided by the Keras library. For this reason, we use the Functional API which allows us to create Merge Models.

First, let's look at the brief architecture which contains the high level sub-modules:

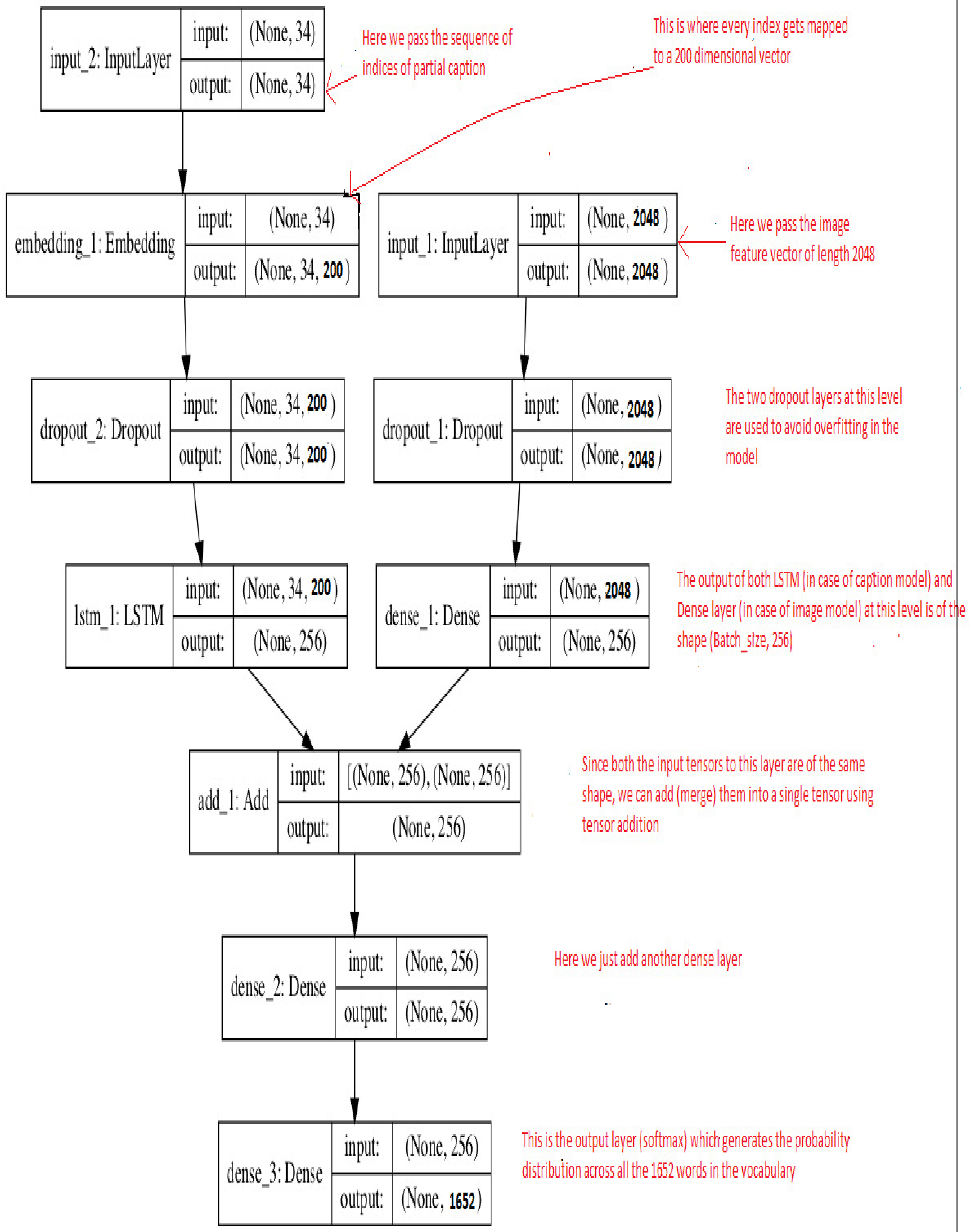
MODEL SUMMARY:



```
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
```

The below plot helps to visualize the structure of the network and better understand the two streams of input:


input_2: InputLayer
input: (None, 34)
output: (None, 34)
Here we pass the sequence of indices of partial caption
This is where every index gets mapped to a 200 dimensional vector
embedding_1: Embedding
input: (None, 34)
output: (None, 34, 200)
input_1: InputLayer
input: (None, 2048)
output: (None, 2048)
Here we pass the image feature vector of length 2048
dropout_2: Dropout
input: (None, 34, 200)
output: (None, 34, 200)
dropout_1: Dropout
input: (None, 2048)
output: (None, 2048)
The two dropout layers at this level are used to avoid overfitting in the model
lstm_1: LSTM
input: (None, 34, 200)
output: (None, 256)
dense_1: Dense
input: (None, 2048)
output: (None, 256)
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)
add_1: Add
input: [(None, 256), (None, 256)]
output: (None, 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
dense_2: Dense
input: (None, 256)
output: (None, 256)
Here we just add another dense layer
dense_3: Dense
input: (None, 256)
output: (None, 1652)
This is the output layer (softmax) which generates the probability distribution across all the 1652 words in the vocabulary

The text in red on the right side are the comments provided for you to map your understanding of the data preparation to model architecture.

The **LSTM (Long Short Term Memory)** layer is nothing but a specialized Recurrent Neural Network to process the sequence input (partial captions in our case).

If you have followed the previous section, I think reading these comments should help you to understand the model architecture in a straight forward manner.

Recall that we had created an embedding matrix from a pre-trained Glove model which we need to include in the model before starting the training:

```
model.layers[2].set_weights([embedding_matrix])
model.layers[2].trainable = False
```

Notice that since we are using a pre-trained embedding layer, we need to **freeze** it (trainable = False), before training the model, so that it does not get updated during the backpropagation.

Finally we compile the model using the adam optimizer

```
model.compile(loss='categorical_crossentropy',
optimizer='adam')
```

Finally the weights of the model will be updated through backpropagation algorithm and the model will learn to output a word, given an image feature vector and a partial caption. So in summary, we have:

Input_1 -> Partial Caption

Input_2 -> Image feature vector

Output -> An appropriate word, next in the sequence of partial caption provided in the input_1 (or in probability terms we say **conditioned** on image vector and the partial caption)

Hyper parameters during training:

The model was then trained for 30 epochs with the initial learning rate of 0.001 and 3 pictures per batch (batch size). However after 20 epochs, the learning rate was reduced to 0.0001 and the model was trained on 6 pictures per batch.

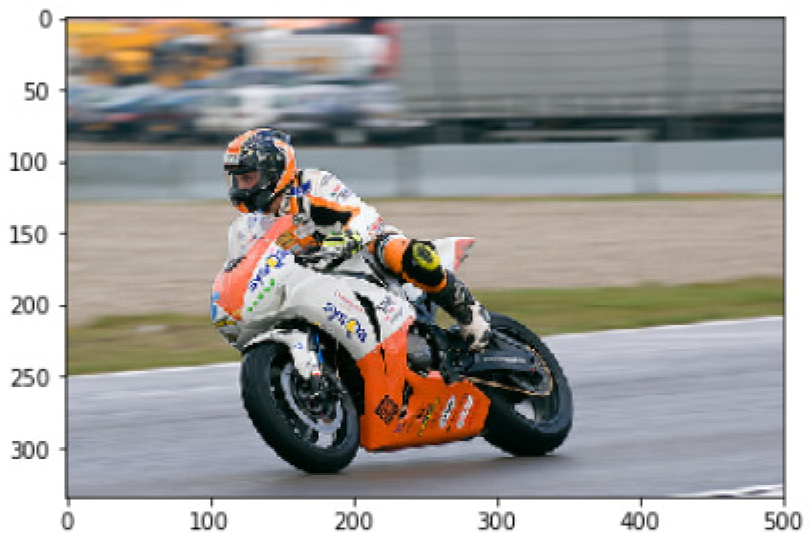
This generally makes sense because during the later stages of training, since the model is moving towards convergence, we must lower the learning rate so that we take smaller steps towards the minima. Also increasing the batch size over time helps your gradient updates to be more powerful.

Time Taken: I used the GPU+ Gradient Notebook hence it took me approximately an hour to train the model. However if you train it on a PC without GPU, it could take anywhere from 8 to 16 hours depending on the configuration of your system.

CHAPTER – 5

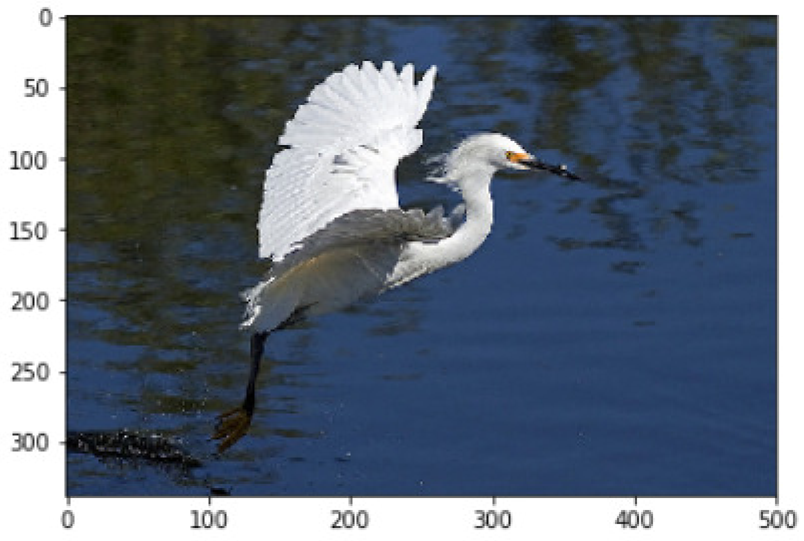
Conclusion

To understand how good the model is, let’s try to generate captions on images from the test dataset (i.e. the images which the model did not see during the training).

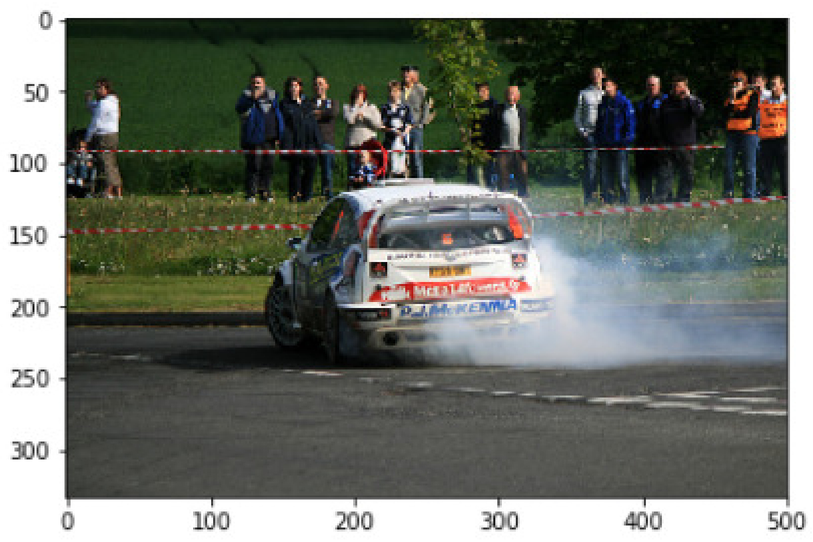


```
Greedy: motorcyclist is riding an orange motorcycle
```

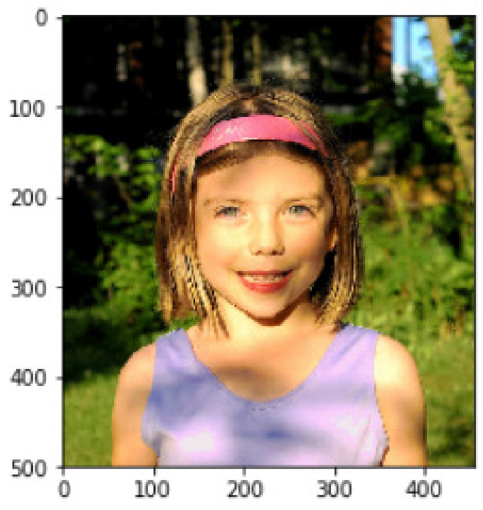
We must appreciate how the model is able to identify the colours precisely.



```
Greedy: white crane with black begins to take flight from the water
```

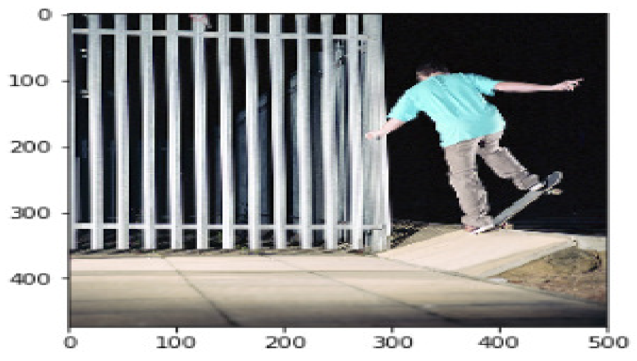


```
15 Greedy: race car spins down the road as spectators watch
```



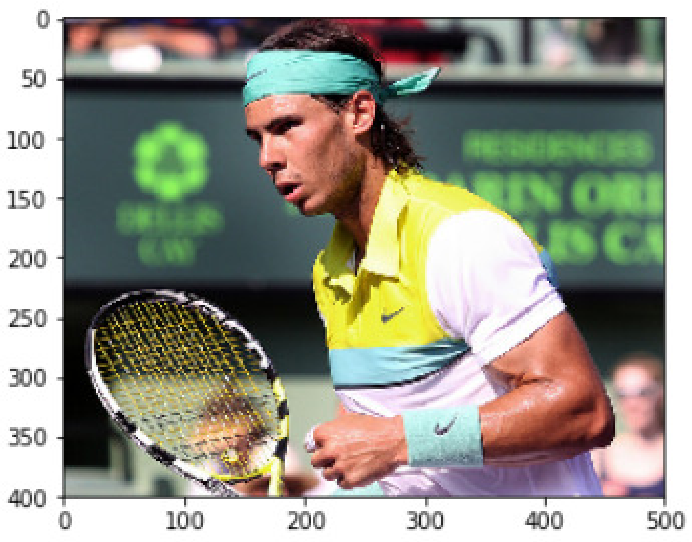
```
Greedy: girl in pink shirt is smiling whilst standing in front of tree
```

Of course, I would be fooling you if I only showed you the appropriate captions. No model in the world is ever perfect and this model also makes mistakes. Let's look at some examples where the captions are not very relevant and sometimes even irrelevant.



```
Greedy: man in black shirt is skateboarding down ramp
```

Probably the colour of the shirt got mixed with the colour in the background in above image.



```
Greedy: a woman in a tennis racket on the court .
```

Why does the model classify the famous Rafael Nadal as a woman :-) ? Probably because of the long hair.

Important Point:

We must understand that the images used for testing must be semantically related to those used for training the model. For example, if we train our model on the images of cats, dogs, etc. we must not test it on images of air planes, waterfalls, etc. This is an example where the distribution of the train and test sets will be very different and in such cases no Machine Learning model in the world will give good performance.

Conclusion and Future work

Note that due to the stochastic nature of the models, the captions generated by you (if you try to replicate the code) may not be exactly similar to those generated in my case.

Of course this is just a first-cut solution and a lot of modifications can be made to improve this solution like:

- Using a **larger** dataset.
- Changing the model architecture, e.g. include an **attention** module.
- Doing more **hyper parameter tuning** (learning rate, batch size, number of layers, number of units, dropout rate, batch normalization etc.).
- Use the cross validation set to understand **overfitting**.
- Using **Beam Search** instead of Greedy Search during Inference.
- Writing the code in a proper object oriented way so that it becomes easier for others to replicate :-)

Furture Work Advanced loss function: The original softmax loss function can cause problems. It can produce force negative. For example, if we input a the test picture with caption A man is riding a horse, the produced caption A horse is carrying a horse will produce high loss, but actually these two caption all correctly describe the picture. On the other hand the model can also produce force negative. For example, if the previous test picture produces a caption A man is carryinga horse, the loss will be small, but this is actually a wrong description of the picture. Sharper attention: From the result we notice that the attention coefficient are evenly distributed, which means that the model takes the whole picture information to generate the next time step hidden layer via LSTM. But we expect that we can highlight specific part of the picture related to the certain word. To achieve this goal we can use hard attention, which restricts information extraction from image as whole. We can also use a harper activation function instead of softmax to produce a suitable attention distribution. Moreover, we can label more detailed captions to force the mode to attend smaller parts. Language model: Since the model will produce a probability distribution of the vocabulary on every time step, we can use language model to generate natural sentences based on these vocabulary probability distributions. Further more, we can build a markov random field like hidden markov model on the top of the the softmax output layer. We can try different architecture and especially layer of CNN encoder to get a better feature map level.

References

1->BLEU: a Method for Automatic Evaluation of Machine Translation Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu IBM T. J. Watson Research Center Yorktown Heights, NY 10598, USA

2->Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, Yoshua Bengio, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, [Online] Available: https://arxiv.org/pdf/1502.03044.pdf [9] M. Hodosh, P. Young and J. Hockenmaier (2013) ”Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics”, Journal of Artificial Intelligence Research, Volume 47, pages 853-899

3->Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan, Show and Tell: A Neural Image Caption Generator, [Online] Available: https://arxiv.org/pdf/1411.4555.pdf

4->Jia-Yu Pan, Hyung-Jeong Yang, Pinar Duygulu, Automatic Image Captioning, Conference: Conference: Multimedia and Expo, 2004. ICME ’04. 2004 IEEE International Conference on, Volume: 3

5->Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database

6->A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks” NIPS, pages 1097–1105, 2012.

7->T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. arXiv preprint arXiv:1405.0312, 2014.

8->Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329, 2014.

9->Andrej Karpathy, Li Fei-Fei, “Deep Visual-Semantic Alignments for Generating Image Descriptions” IEEE, 05 August 2016.

10->M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: data, models and evaluation metrics. Journal of Artificial Intelligence Research, 2013.

11->T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. arXiv preprint arXiv:1405.0312, 2014

12->Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, et al .“From Captions to Visual Concepts and Back” IEEE Conf. on Computer Vision and Pattern Recognition, 2015.

APPENDIX

a. Dataset preprocessing result
b. Final model result
c. Result comparison

APPENDIX A: DATASET PREPROCESSING RESULT:

Appendix 1.1: image preprocessing :

Resnet50:

In image preprocessing we have used resnet 50 model and converted our image pixels from (224,224) to (,2048). After that we applied encode image function to assign the vector value for all the images and encode them as one of the ouput as follows:

```
array([0.04154464, 0.17075905, 0.29276785, ..., 0.05094026, 0.3
131964 ,
       0.8600726 ], dtype=float32)
```

Size of above array is (,2048).

Appendix 1.2:caption preprocessing :

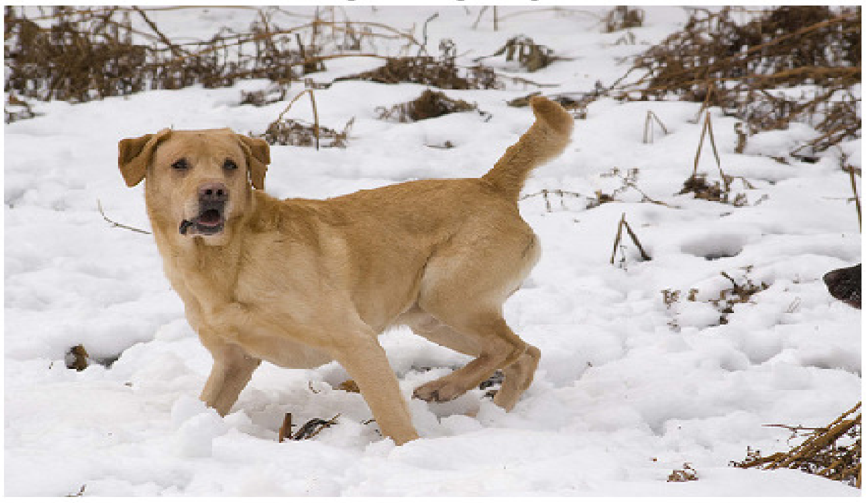
Glove.6b.50d:

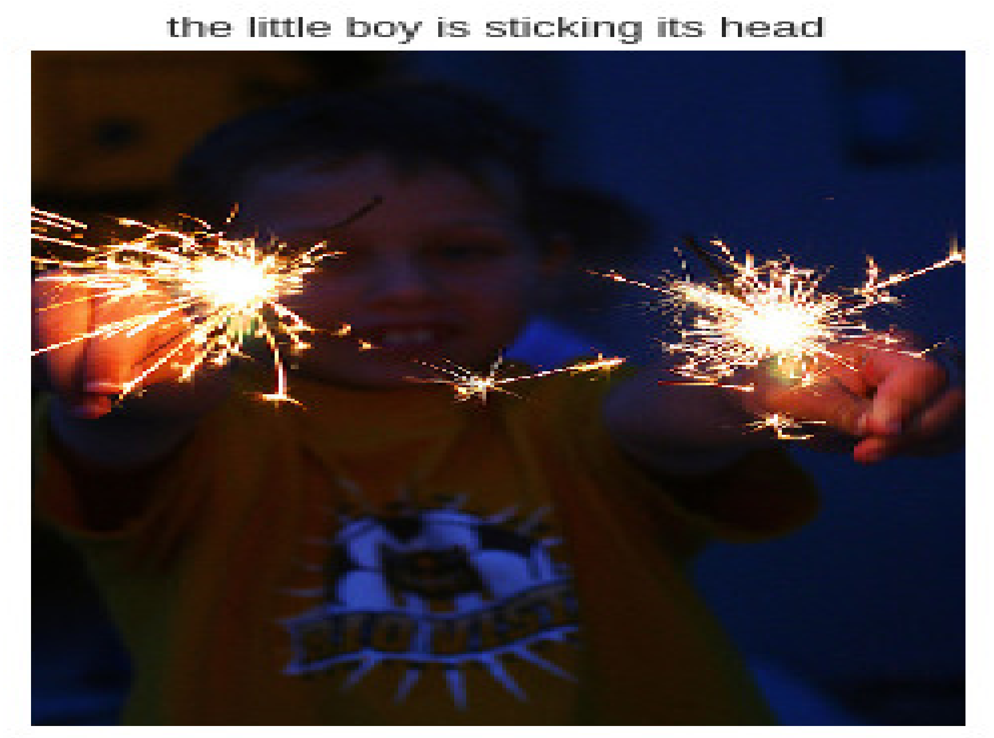
In caption preprocessing we have used glove vector so that we could convert every word to vector form and for that we had used glove vector and we applied it into our word dictionary. Example of one word as follows:

```
embedding_index['apple']


array([ 0.52042 , -0.8314 , 0.49961 , 1.2893 , 0.1151 , 0.057521,
 -1.3753 , -0.97313 , 0.18346 , 0.47672 , -0.15112 , 0.35532 , 0.
25912 , -0.77857 , 0.52181 , 0.47695 , -1.4251 , 0.858 , 0.59821 ,
 -1.0903 , 0.33574 , -0.60891 , 0.41742 , 0.21569 , -0.07417 , -0.
5822 , -0.4502 , 0.17253 , 0.16448 , -0.38413 , 2.3283 , -0.6668
2 , -0.58181 , 0.74389 , 0.095015, -0.47865 , -0.84591 , 0.38704
 , 0.23693 , -1.5523 , 0.64802 , -0.16521 , -1.4719 , -0.16224 ,
0.79857 , 0.97391 , 0.40027 , -0.21912 , -0.30938 , 0.26581 ])
```

APPENDIX B : FINAL MODEL RESULT:

brown dog is running through snow





APPENDIX C: RESULT COMPARISON:

Previous work:

The image captioning model was implemented and we were able to generate moderately comparable captions with compared to human generated captions. The VGG net model first assigns probabilities to all the objects that are possibly present in the image. The model converts the image into word vector. This word vector is provided as input to LSTM cells which will then form sentence from this word vector. Generated sentence are black dog runs into the ocean next to a rock, while actual human generated sentences are black dog runs into the ocean next to a pile of seaweed., black dog runs into the ocean, a black dog runs into the ball, a black dog runs to a ball. This results in a BLEU score of 57 for this image.

Proposed work:

The image captioning model was implemented and we were able to generate moderately comparable captions with compared to human generated captions. The RESNET 50 model first assigns probabilities to all the objects that are possibly present in the image. The model converts the image into word vector. This word vector is provided as input to LSTM cells which will then form sentence from this word vector. Generated sentence are black dog runs into the ocean next to a rock, while actual human generated sentences are black dog runs into the ocean next to a pile of seaweed., black dog runs into the ocean, a black dog runs into the ball, a black dog runs to a ball. This results in a BLEU score of 66 for this image.