

AI IMAGE CAPTIONING

Major project report submitted in partial fulfillment of the requirement for
the degree of Bachelor of Technology

in

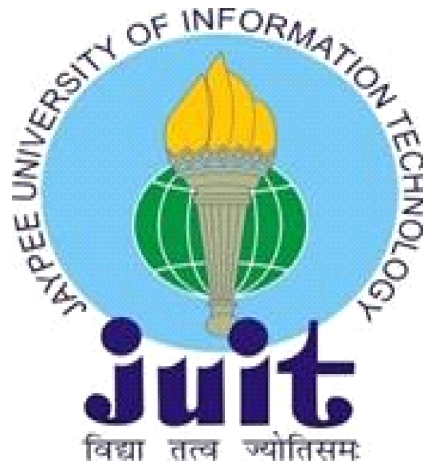
Computer Science and Engineering

By

PARTH BALI (181354)

UNDER THE SUPERVISION OF

Mr. Amit Kumar



Department of Computer Science & Engineering and
Information Technology

**Jaypee University of Information Technology, Wagnaghat,
173234, Himachal Pradesh, INDIA**

(I) CERTIFICATE

This is to certify that the work which is being presented in the project report titled “**AI Image Captioning**” in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering and submitted to the Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “Parth Bali (181354)” during the period from January 2021 to May 2021 under the supervision of **Dr. Amit Kumar**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Parth Bali (181354)

The above statement made is correct to the best of my knowledge.



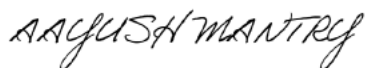
Dr. Amit Kumar Assistant

Professor (SG)

Computer Science & Engineering and Information

Technology Jaypee University of Information Technology,

Waknaghat



Mr. Mantry Aayush

DC Senior Product Specialist

Hashedin by Deloitte,

Bangalore

(II) ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for his divine blessing makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Amit Kumar, Assistant Professor (SG)**, Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of “**Data Science**” to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Amit Kumar**, Department of CSE, for her kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educated and non- instructed, who have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Parth Bali (181354)

(III) TABLE OF CONTENT

| | |
|----------------------|------|
| Certificate | (I) |
| Abstract | (II) |
| List of figures | (IV) |
| Introduction | 7 |
| Literature Survey | 12 |
| System Development | 17 |
| Performance Analysis | 33 |
| Conclusions | |
| References | |

(III) ABSTRACT

Past years have definitely been very crucial to the development of artificial intelligence and scene recognition has become the most contributed field of Computer Vision, and it has attracted the attention of many young researchers. Generating natural language descriptions automatically according to the content observed in a picture, is an important part of scene understanding, which combines the knowledge of computer vision along with natural language processing. The application of image captioning is vast and profound. Tremendous progress in scene recognition problems is credited to the availability of large databases like MS COCO, Places, ImageNet, etc, and the development of CNNs (Convolutional Neural Networks) for gathering high-level features. This paper aims to describe the methodology used and possible improvements.

Chapter 01: INTRODUCTION

- **Intro**



Think of a caption to this

- One would say “A dog in a grassy area”, Others might think as “A dog with brown spots” and a few would be saying different things
- But, are you capable of writing a worm that asks for an input of image, gives an accurate caption as output?

- **Objective**

This problem was unthinkable by the best in the field of computer science before current DNN research. This problem appears to be readily solved using Deep Learning.

- **Need**

- **Blind Help** — This project makes us prepared to develop a product for the blind that will allow them to navigate the highways without the assistance of others. We're prepared to accomplish so by turning the view to words and then the words to sound. These Deep Learning applications are now well-known.

- Self-driving vehicles — One of the most important challenges is self-driving cars, and if the area around the car is properly captioned, it can help the self-driving system.

Project Description:

The purpose of this project is to understand (in simpler words) how Deep Learning often wants to solve the matter in the generation of a caption in a given image, thus it is called Image Captioning.

- **Languages Used**

RNN:

A recurrent neural network (RNN) is a type of artificial neural network in which the connections between nodes generate a directed or undirected graph over time. As a result, it can show temporal dynamic behaviour. RNNs, which are derived from feedforward neural networks, can process variable length sequences of inputs by utilising their internal state (memory).

CNN:

A convolutional neural network (CNN, or ConvNet) is a type of artificial neural network used to research visual imagery in deep learning. Shift invariant or space invariant artificial neural networks (SIANN) are supported by a shared-weight architecture of convolution kernels or filters that slide along input features and offer translation equivariant responses in the form of feature maps. Surprisingly, most convolutional neural networks are only equivariant to translation as opposition invariant.

Keras:

Keras is a human-centric API, not a machine-centric one. Keras uses best practises to reduce cognitive burden, such as providing simple and consistent APIs, decreasing the number of user steps required for typical use cases, and providing clear and actionable error signals. There's also a lot of documentation and developer instructions.

- **Technical Requirements (Hardware)**

Screen: 14 inches Mouse: 3

Button scroll Keyboard: 108

keys

Chapter 02: LITERATURE SURVEY

- **Feasibility Study:**

For the past five years, NLPs have been studied. The advent of computing has greatly improved the model's performance. The results are still far from adequate. It will always be a difficulty since machines cannot mimic human minds or analyse in the same way that humans do. It is getting increasingly challenging to keep up with the most recent research and outcomes in the field of picture captioning due to the enormous number of information on the subject. During this research, a thorough Systematic Literature Review(SLR) gives some short review related to photo captioning advancements in the 4 years.

- **SLR Methodology:**

- SLR has made many things easier in the data progressed world these days, having a lot of data growth.
- This can be hard to ingest every currently available info later diving for a certain block. For the situation, after discussing image captioning and, as previously stated, having the most meaning during this assignment, it was discovered that there is a great deal of material that is difficult to describe and thus stay up to speed with the most recent achievements.
- Researchers spent time studying most image captioning articles in depth—digital libraries, which house the

majority of the articles, were identified, search questions were formulated, all articles found were analysed, and the results were presented along with key challenges identified during the review process.

2.1. Sources

- The investigation was primarily carried out using three separate digital libraries:
 1. Science Web
 2. IEEE Xplore
 3. ArXiv

Because there has been a lot of study for the field in picture captioning, I limited our literary conclusions to studies published in the last four years (2019 to 2021).

2.2. Questions

- It's critical to have specific questions that need to be answered after you've gone over all of the literature.
- The queries were carefully crafted after multiple attempts to ensure that the results received after each query were accurate, free of excessive noise, and did not contain any non-required articles.

- We respond to four questions in this paper:
 - 🌍 What methods were employed in the creation of image captions?
 - 🌍 What are the difficulties in captioning images?
 - 🌍 How does the incorporation of semantics and innovative picture descriptions enhance the model's progress for AI image credits?
 - 🌍 Most recent picture captioning studies?

The questions were carefully chosen to cover the most important motives of the report: for highlighting most commonly utilised strategies of picture captioning during last four years, and to find the most significant obstacles that the researchers have faced. We also wanted to summarise results from recent publications in a good comparison for lateron papers, thus it added some broad view the image captioning but segragating out articles published in 2018. We would like to read only the most current articles because image captioning questions should be too expanding, and we would have a large focus on introducing readers for most recent successes.

2.3. Query

- To have a better understanding of the issue of "picture caption creation," we first ran through a quick survey of related papers.
- We gained an understanding of the concepts and technologies that are widely used in this field, allowing us to conduct research that is both relevant and accurate.
- Furthermore, we would not reduce the query into minute features to provide adequate outcomes and a suitable no. of keywords shown just as they were supplied for the search query.
- In Tables 1–4, the query questions are listed joined along with a no. of information found in each library. The query was first searched in ArXiv, then in IEEE Xplore, and finally in WOS, in the order stated in the table. If an article was located that had already been previewed in a prior query, it was not added to the total number of relevant articles, but was instead identified in brackets.

Table 1. Results from the search after “Techniques image caption generation” search query.

| 1Q | ArXiv | IEEE | WOS |
|----------|-------|-------|-------|
| Found | 29 | 9 | 18 |
| Relevant | 11 | 6 (1) | 10(6) |

Table 2. Results from the search after “Challenges image caption generation” search query.

| 2Q | ArXiv | IEEE | WOS |
|----------|--------|-------|---------|
| Found | 48 | 19 | 78 |
| Relevant | 16 (3) | 3 (6) | 12 (11) |

Table 3. Results from the search after “Novel semantic image captioning” search query.

| 3Q | ArXiv | IEEE | WOS |
|----------|-------|-------|--------|
| Found | 26 | 28 | 42 |
| Relevant | 6 (4) | 8 (6) | 2 (10) |

Table 4. Results from the search after “Image captioning” search query with filtered date to 2019 as w aimed for the newest articles published.

| 4Q | ArXiv | IEEE | WOS |
|----------|-------|-------|-------|
| Found | 38 | 20 | 25 |
| Relevant | 8 (3) | 5 (3) | 7 (3) |

-
- One is self-evident that Service Web produces most results, albeit with the minimal %age of relevant articles for the required topic. ArXiv had been most accurate and possessed the most effective ratio comparatively accuracy and other features, according to this study's findings.

Result:

We were able to get the essential understanding of the key features of picture captioning after reading all of the articles and being motivated by another SLR. To provide the summary in a convenient manner, a full comparison table of all publications found with the methodology utilised, as well as the findings on the most widely used datasets for testing, was developed.

- The following is the structure:

🌐 Year

- 2016
 - 2017
 - 2018
 - 2019;
- GoogleNet (which includes all nine Inception models), VGG-16 Net, DenseNet, AlexNet, and ResNet are feature extractors.
- CNN, RNN, LSTM, TPGN, and cGRU are examples of language models.
- Encoder-decoder; Attention mechanism; Novel objects; Semantics are some of the methods used.

🌐 MS COCO; Flickr 8k; MS COCO; MS COCO; MS COCO; MS COCO; MS COCO; MS COCO; MS COCO; MS

🌐 METEOR; CIDEr; BLEU-4; BLEU-1; BLEU-2; BLEU-2; BLEU-3

If an aspect was mentioned in the article, an x was written next to it in each column. The following columns show the findings of calculation metrics for 2 sets of data: Flickr30k and MS COCO. The cells were left empty in case no testing had been applied on any of the given data sets chosen. A brief note was supplied if a different data set or calculation metric was applied in its publication (Appl. Sci. 2018, 6, 2021 5 of 19).

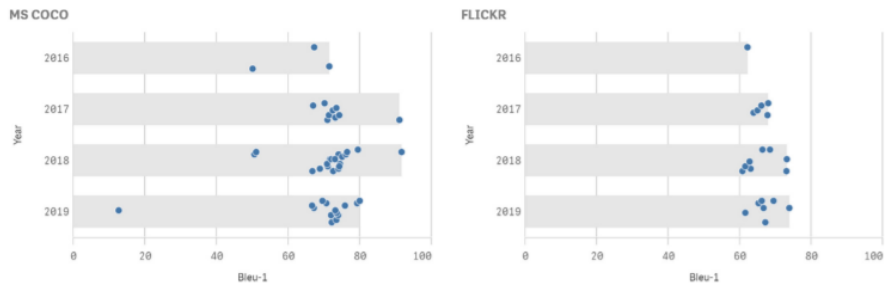


Figure 1. Bleu-1 results by year on MSCOCO (left) and Flickr30k (right).

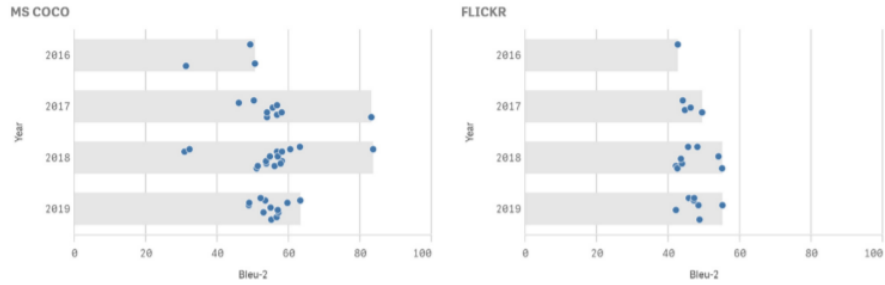


Figure 2. Bleu-2 results by year on MSCOCO (left) and Flickr30k (right).

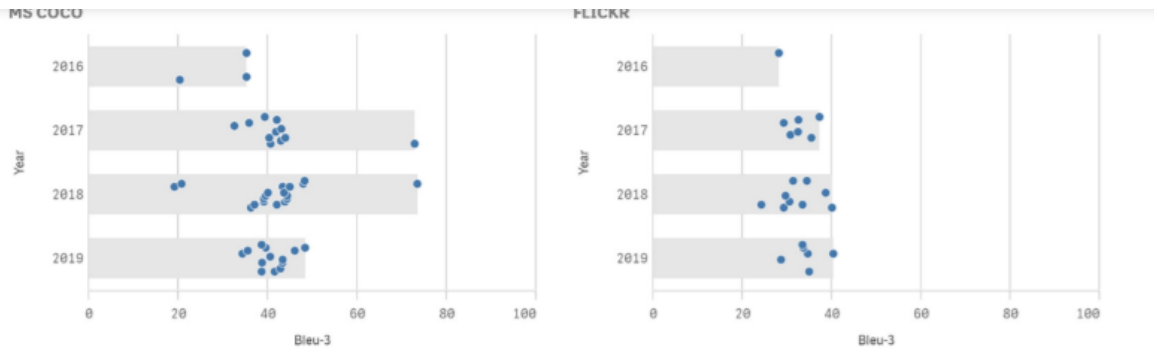


Figure 3. Bleu-3 results by year on MSCOCO (left) and Flickr30k (right).

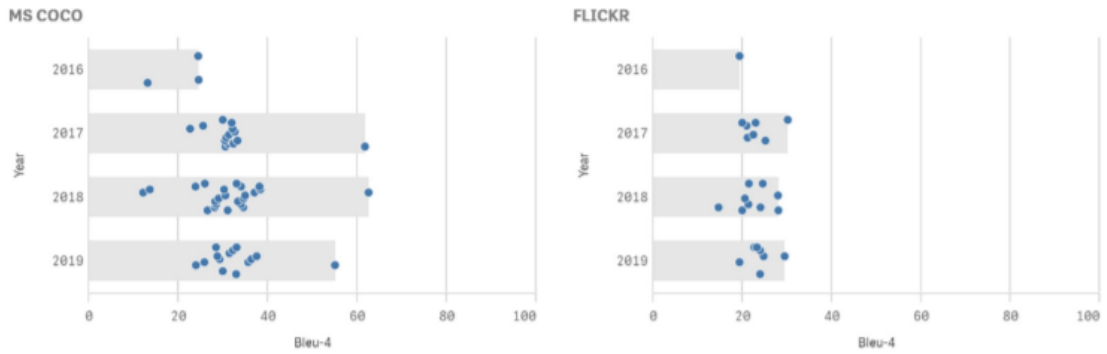


Figure 4. Bleu-4 results by year on MSCOCO (left) and Flickr30k (right).

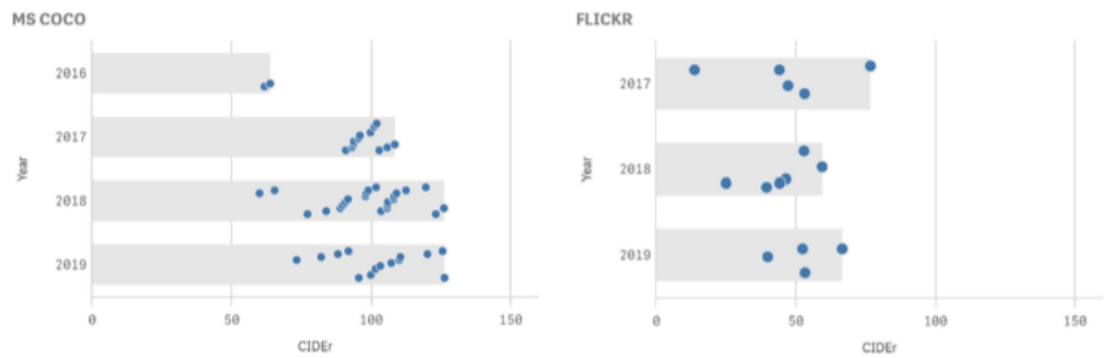


Figure 5. CIDEr results by year on MSCOCO (left) and Flickr30k (right).

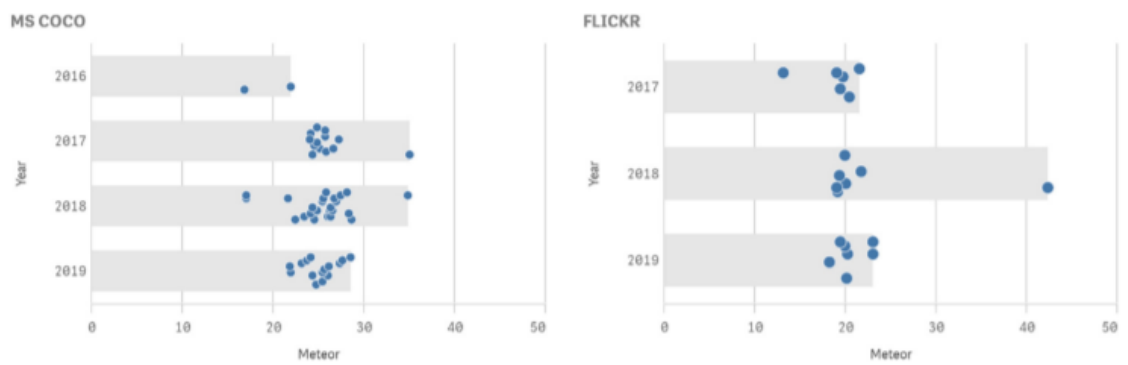


Figure 6. Meteor results by year on MSCOCO (left) and Flickr30k (right).

Tables:

Table 5. Top 3 results in 2016 (sorted by CIDEr result on MSCOCO).

| Year | Citation | Image Encoder | | | | | Image Decoder | | | | | Method | | | MS COCO | | | | Flickr30k | | | | | | | |
|------|----------|---------------|--------|-----------|--------|----------|---------------|-----|-----|------|------|-----------------|-----------|---------------|-----------|--------|--------|--------|-----------|-------|--------|--------|--------|--------|--------|-------|
| | | AlexNet | VGGNet | GoogLeNet | ResNet | DenseNet | LSTM | RNN | CNN | cGRU | TPGN | Encoder-Decoder | Attention | Novel objects | Semantics | Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 | CIDEr | Meteor | Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 | CIDEr |
| 2016 | [13] | | x | | | x | | | | | x | x | | | 71.4 | 50.5 | 35.2 | 24.5 | 63.8 | 21.9 | | | | | | |
| 2016 | [12] | | | x | | x | | | | | x | | x | | 50.0 | 31.2 | 20.3 | 13.1 | 61.8 | 16.8 | | | | | | |
| 2016 | [11] | x | x | | | x | | | | | x | | | | 67.2 | 49.2 | 35.2 | 24.4 | | | 62.1 | 42.6 | 28.1 | 19.3 | | |

Table 6. Top 5 results in 2017 (sorted by CIDEr result on MSCOCO).

| Year | Citation | Image Encoder | | | | | Image Decoder | | | | | Method | | | MS COCO | | | | Flickr30k | | | | | | | |
|------|----------|---------------|--------|-----------|--------|----------|---------------|-----|-----|------|------|-----------------|-----------|---------------|-----------|--------|--------|--------|-----------|-------|--------|--------|--------|--------|--------|-------|
| | | AlexNet | VGGNet | GoogLeNet | ResNet | DenseNet | LSTM | RNN | CNN | cGRU | TPGN | Encoder-Decoder | Attention | Novel Objects | Semantics | Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 | CIDEr | Meteor | Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 | CIDEr |
| 2017 | [34] | | | | x | x | | | | | x | x | | | 74.2 | 58 | 43.9 | 33.2 | 108.5 | 26.6 | 67.7 | 49.4 | 35.4 | 25.1 | 53.1 | 20.4 |
| 2017 | [33] | | | | x | x | | | | | x | x | | | 73.1 | 56.7 | 42.9 | 32.3 | 105.8 | 25.8 | | | | | | |
| 2017 | [32] | | x | | x | x | x | | x | | x | x | | | 91 | 83.1 | 72.8 | 61.7 | 102.9 | 35 | | | | | | |
| 2017 | [31] | | x | | x | x | | | | | x | | | | | | 39.3 | 29.9 | 102 | 24.8 | | | 37.2 | 30.1 | 76.7 | 21.5 |
| 2017 | [30] | | x | | x | x | | | | | | | | | | | 42 | 31.9 | 101.1 | 25.7 | | | 32.5 | 22.9 | 44.1 | 19 |

Table 7. Top 5 results in 2018 (sorted by CIDEr result on MSCOCO).

| Year | Citation | Image Encoder | | | | | Image Decoder | | | | | Method | | | MS COCO | | | | Flickr30k | | | | | | | |
|------|----------|---------------|--------|-----------|--------|----------|---------------|-----|-----|------|------|-----------------|-----------|---------------|-----------|--------|--------|--------|-----------|-------|--------|--------|--------|--------|--------|-------|
| | | AlexNet | VGGNet | GoogLeNet | ResNet | DenseNet | LSTM | RNN | CNN | cGRU | TPGN | Encoder-Decoder | Attention | Novel Objects | Semantics | Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 | CIDEr | Meteor | Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 | CIDEr |
| 2018 | [70] | | | | x | x | | | | | x | x | | | | | | | 38.1 | 126.1 | 28.3 | | | | | |
| 2018 | [69] | | | | x | x | | | | | x | x | | | | | | | 38.3 | 123.2 | 28.6 | | | | | |
| 2018 | [68] | | | | x | x | x | | | | x | x | x | x | 79.4 | 63.1 | 48.2 | 36.1 | 119.6 | 28.1 | 72.1 | 48.7 | 36.9 | 21.2 | 53.6 | 20.5 |
| 2018 | [67] | | x | | x | x | | | | | x | x | x | | 76.4 | 60.4 | 47.9 | 37 | 112.5 | 27.4 | | | | | | |
| 2018 | [66] | | | | x | x | | | | | x | x | | x | 76.1 | 58.1 | 44.9 | 34.9 | 109.1 | 26.7 | 73.1 | 54 | 38.6 | 27.9 | 59.4 | 21.7 |

Table 8. Top 5 results in 2019 (sorted by CIDEr result on MSCOCO).

| Year | Citation | Image Encoder | | | | | Image Decoder | | | | | Method | | | MS COCO | | | | Flickr30k | | | | | | | | |
|------|----------|---------------|--------|-----------|--------|----------|---------------|-----|-----|------|------|-----------------|-----------|---------------|-----------|--------|--------|--------|-----------|-------|--------|--------|--------|--------|--------|-------|--------|
| | | AlexNet | VGGNet | GoogLeNet | ResNet | DenseNet | LSTM | RNN | CNN | cGRU | TPGN | Encoder-Decoder | Attention | Novel Objects | Semantics | Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 | CIDEr | Meteor | Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 | CIDEr | Meteor |
| 2019 | [87] | | | | x | x | | | | | x | x | x | x | | | | | 38.6 | 28.3 | 126.3 | | | | | | |
| 2019 | [86] | | | | x | x | | | | | x | x | x | x | 79.9 | | | | 37.5 | 125.6 | 28.5 | 73.8 | 55.1 | 40.3 | 29.4 | 66.6 | 23 |
| 2019 | [85] | | | | x | x | | | | | x | x | x | x | 79.2 | 63.2 | 48.3 | 36.3 | 120.2 | 27.6 | | | | | | | |
| 2019 | [84] | | | | x | x | | | | | | | | x | 75.8 | 59.6 | 46 | 35.6 | 110.5 | 27.3 | | | | | | | |
| 2019 | [83] | | x | | x | x | | | | | x | x | x | x | | | | | 55 | 110.1 | 26.1 | | | | | | |

Conclusions

- Image captioning is a fun activity that generates a lot of competitiveness among scholars.
- Many scientists are opting to research this field of study, resulting in a steady increase in knowledge.

- Despite the fact that there are dozens of fresh articles with even better results and new ideas for development, it was observed that the results are commonly compared to previous articles.
- Comparison of prior papers leads to a misunderstanding of the crucial concept of result improvement—in most cases, significantly larger results have already been attained but are not mentioned in the report.
- If new ideas are not diligently sought for, they can easily be lost.
- This comprehensive literature review integrates all of the most recent papers and their findings in one location, preventing the loss of useful ideas and promoting fair competition among the new models developed.
- Furthermore, it is yet unknown if the MS_COCO & Flickr 8k Datasets are acceptable for model evaluation & whether they produce enough results in a range of settings.

Chapter 03: SYSTEM DEVELOPMENT

- **Model Development**

1. Collection of Data:

- Many open source datasets are available for such problems , such as Flickr 8k , Flickr 30k , MS COCO
- In our project, we've utilised the already known Flickr8KDataset, having 8K images.
- Each image contains 5 captions linked to it.
- These pictures are divided into 3 categories:
 - Training containing 6001 pictures
 - Dev containing 1001 pictures
 - Testing containing 1001 pictures

2. Knowing Our Dataset:

- The file name "Flickr 8k.token.txt" has the name of the picture and the captions linked to it. It will be read in a following manner:
 - `F_name = '/Datasets/TxtFiles/Flickr_8k.token.txt'`
 - `F = open(fname, 'r')`
 - `Docum = f.read()`
- The txt file is:

```

1  101654506_8eb26cfb60.jpg#0    A brown and white dog is running through the snow .
2  101654506_8eb26cfb60.jpg#1    A dog is running in the snow
3  101654506_8eb26cfb60.jpg#2    A dog running through snow .
4  101654506_8eb26cfb60.jpg#3    a white and brown dog is running through a snow covered field .
5  101654506_8eb26cfb60.jpg#4    The white and brown dog is running over the surface of the snow
6
7  1000268201_693b08cb0e.jpg#0    A child in a pink dress is climbing up a set of stairs in an ent
8  1000268201_693b08cb0e.jpg#1    A girl going into a wooden building .
9  1000268201_693b08cb0e.jpg#2    A little girl climbing into a wooden playhouse .
10 1000268201_693b08cb0e.jpg#3    A little girl climbing the stairs to her playhouse .
11 1000268201_693b08cb0e.jpg#4    A little girl in a pink dress going into a wooden cabin .

```

- i.
- ii. Each line has #j, in which $1 \leq j \leq 5$ that is, name of the picture, no. of images (1 to 5) and moreover it contains the real caption.
- iii. Now, we will make a dictionary containing picture names (not

```

1  descriptions = dict()
2  for line in doc.split('\n'):
3      # split line by white space
4      tokens = line.split()
5
6      # take the first token as image id, the rest as description
7      image_id, image_desc = tokens[0], tokens[1:]
8
9      # extract filename from image id
10     image_id = image_id.split('.')[0]
11
12     # convert description tokens back to string
13     image_desc = ' '.join(image_desc)
14     if image_id not in mapping:
15         descriptions[image_id] = list()
16     descriptions[image_id].append(image_desc)

```

having its extension) as keys which will be linked to 5 captions related to it.

3. Data Cleaning:

- a. We will do the basic text cleaning, such as all words will be converted to lower-case letters, special characters will be removed , all the words containing numbers will be eliminated.

```

1 # prepare translation table for removing punctuation
2 table = str.maketrans('', '', string.punctuation)
3 for key, desc_list in descriptions.items():
4     for i in range(len(desc_list)):
5         desc = desc_list[i]
6         # tokenize
7         desc = desc.split()
8         # convert to lower case
9         desc = [word.lower() for word in desc]
10        # remove punctuation from each token
11        desc = [w.translate(table) for w in desc]
12        # remove hanging 's' and 'a'
13        desc = [word for word in desc if len(word)>1]
14        # remove tokens with numbers in them
15        desc = [word for word in desc if word.isalpha()]
16        # store as string
17        desc_list[i] = ' '.join(desc)

```

b.

c. A new vocabulary is created, containing the unique words; 8200*4 image captions (corpus) within the data set :

- Vocab=Set_()
- FOR k in desc.keys():
- [vocab..update(l.split()) for l in desc[keys]]
- cout(“Actual Vocab Size: %d”: % len(vocab))
- Actual Vocab Size = 8814

d. In a nutshell, we got 8814 unique __ words over the total image captions that are present. A new file named ‘descriptions.txt’ contains all the captions

e. However, many words would occur only a small amount of times, say I, II or III times. Since we are making a predictive model, we'd not wish present in our vocabulary but the words which are more likely to occur or which are common.

- f. Hence only those words occur a minimum of ten times within the total outcome. Having code below:

```
1 # Create a list of all the training captions
2 all_train_captions = []
3 for key, val in train_descriptions.items():
4     for cap in val:
5         all_train_captions.append(cap)
6
7 # Consider only words which occur at least 10 times in the corpus
8 word_count_threshold = 10
9 word_counts = {}
10 nsents = 0
11 for sent in all_train_captions:
12     nsents += 1
13     for w in sent.split(' '):
14         word_counts[w] = word_counts.get(w, 0) + 1
15
16 vocab = [w for w in word_counts if word_counts[w] >= word_count_threshold]
17
18 print('preprocessed words %d ' % len(vocab))
19 # preprocessed words 1651
```

i.

- ii. In total , we have 1651 unique words .

4. TrainingSet Loading

The txt file “Flickr_8k.trainImages.txt” has images of training set. The names are loaded to the list “train”.

```
filename = 'dataset/TextFiles/Flickr_8k.trainImages.txt'
doc = load_doc(filename)
train = list()
for line in doc.split('\n'):
    identifier = line.split('.')[0]
    train.append(identifier)
print('Dataset: %d' % len(train))
Dataset: 6000
```

There are 6000 images in the training set.

Now, we load the descriptions of those images from “descriptions.txt” within the Python dictionary “train_descriptions”.

However, after we load them, we are going to add two tokens in every caption as follows (significance explained later):

‘end seq’ -> this can be an end sequence token which can be added at the top of each caption.

```
doc = load_doc('descriptions.txt')
train_descriptions = dict()
for line in doc.split('\n'):
    # split line by white space
    tokens = line.split()

    # split id from description
    image_id, image_desc = tokens[0], tokens[1:]

    # skip images not in the set
    if image_id in dataset:
        if image_id not in descriptions:
            train_descriptions[image_id] = list()

        # wrap description in tokens
        desc = 'startseq ' + ' '.join(image_desc) + ' endseq'

        # store
        train_descriptions[image_id].append(desc)

    print('Descriptions: train=%d' % len(train_descriptions))
# Descriptions: train=6000
```

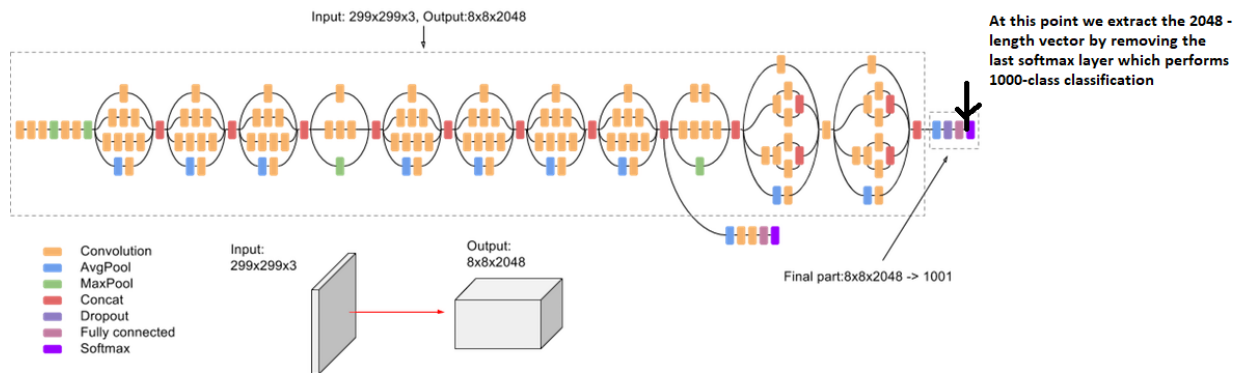
5. Data Preprocessing — Images

Images are input (X) to the model. As you'll already know that any input to a model must run within the type of a vector.

Every picture will be converted to vector type. We are using transfer learning by using the InceptionV3 model (Convolutional Neural Network) developed by Google.

This model performs image classification on 1,000 various classes of pictures and was trained on Imagenet dataset. However, the purpose of the model isn't to classify the image but just get a fixed-length informative vector for each picture. The procedure is called automatic feature engineering.

Therefore, we will delete the last softmax layer of the model and extract 2048 length vector (bottleneck features) for every image:



Feature Vector Extraction (Feature Engineering) from InceptionV3

The code is as given below:

```
# Get the InceptionV3 model trained on imagenet data
model = InceptionV3(weights='imagenet')
# Remove the last layer (output softmax layer) from the inception v3
model_new = Model(model.input, model.layers[-2].output)
```

After this , each image is passed to the present model and induced to the corresponding 2048 length feature vector as follows:

```
# Convert all the images to size 299x299 as expected by the
# inception v3 model
img = image.load_img(image_path, target_size=(299, 299))
# Convert PIL image to numpy array of 3-dimensions
x = image.img_to_array(img)
# Add one more dimension
x = np.expand_dims(x, axis=0)
# preprocess images using preprocess_input() from inception module
x = preprocess_input(x)
# reshape from (1, 2048) to (2048, )
x = np.reshape(x, x.shape[1])
```

Picture classification is the process of finding and labelling groups of pixels or vectors inside an image based on a set of rules. The classification law can be based on one or more spectral or textural qualities. Unsupervised and supervised classification methods are the two types of classification procedures.

NOTE: This process takes around 1-2 hours.

Similarly we do this procedure with all the test images

Data Preprocessing — Captions

The model will predict the captions to the iamges. So during the training period, captions will be (Y) i.e. target variables , that the model is learning to predict.

However, predicting the whole caption based on the image is not instantaneous. We'll try to guess the caption word for word. As a result, each word will be encoded into a vector of a specific size. Except for now, we'll be making two Python Dictionaries called "wordtoix" and "ixtoward." This section would have to be seen once we look at the model design, but for now, we'll be constructing two Python Dictionaries called "wordtoix" and "ixtoward."

In general, we'll use an integer to represent each individual word in the lexicon (index). As previously said, we have a total of 1652 unique words, and each word will be represented by a number ranging from 1 to 1652.

We will be using 2 python dictionaries:

1. `word_to_ix['abc']` -> Gives idx of a word 'xyz'
2. `ix_to_word[k]` -> Gives word that has idx as 'i'

The following is a list of the code:

```
ixtoward = {}
wordtoix = {}

ix = 1
for w in vocab:
    wordtoix[w] = ix
    ixtoward[ix] = w
    ix += 1
```

Another thing we will be calculating is the Maximum length of a caption and the code for it is mentioned below:

```
# convert a dictionary of clean descriptions to a list of
descriptions
def to_lines(descriptions):
    all_desc = list()
    for key in descriptions.keys():
        [all_desc.append(d) for d in descriptions[key]]
    return all_desc

# calculate the length of the description with the most words
def max_length(descriptions):
    lines = to_lines(descriptions)
    return max(len(d.split()) for d in lines)

# determine the maximum sequence length
max_length = max_length(train_descriptions)
print('Max Description Length: %d' % max_length)
Max Description Length: 34
```

Maximum Description Length = 34

Therefore, the maximum length of a caption = 34.

6. Generator Func

This is a very important step of our model. Here we'll understand the way to prepare the info in a manner which is able to be convenient to incline as input to the deep learning model.

Hereafter, we will explain this step with the help of an example:

Consider we've 3 pictures and the 3 corresponding captions to the images as mentioned below:



(Training Picture - 1) Caption = The black cat sat on grass



(Training Picture 2) Caption = The white cat is walking on road



(Testing Picture) Caption = The black cat is walking on grass

Here, First 2 images used to train the model, and third one is the testing image.

Now some questions that arises:

- What makes this a difficulty for supervised learning?
- What information do we have at our disposal?
- How does the data matrix appear?

Firstly, Both photos will be converted to their corresponding feature vectors. of 2048 length. Let's call the feature vectors of the primary two photos "Training Picture 1" and "Training Picture 2," respectively.

Second, we'll combine the two tokens "start seq" and "end seq" in Image 1 and Image 2 to develop a vocab for the two main (train) captions:

Cptn 1 = “startseq the black cat sat on grass endseq”

Cptn 2 = “startseq the white cat is walking on road endseq”

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

Each word is given an index within vocab as follows:

- black = 1
- cat = 2
- endseq = 3
- grass = 4
- is = 5
- on = 6
- road = 7
- sat = 8
- start seq = 9
- the = 10
- walking = 11
- white = 12

Consider it as a supervised learning problem with a set of knowledge points $D = \{ (X_i, Y_i) \}$, where X_i (feature vector of knowledge point i) and Y_i (the target variable).

Consider the primary picture vector Image 1 as well as the caption that goes with it: "start seq the fisher cat sat on grass end seq." Remember, the input is the image vector, and We want to predict what the caption for the image is. However, here's how we think the caption will turn out:

For the first time, we provide the picture vector (and hence the first word) as input and try to predict the second word, i.e.

Input -> Image1 + ' startseq '

Output -> ' the '

Then we offer an image vector as well as the first two words as input, and check out to predict the third word, i.e.

Output = ' cat '

Input = Image_1 + ' startseq the '

Thereafter,

At the end , the information matrix for one image and its title will be described 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 |
| | | | |

image_1 and its caption are represented by data points.

It is worth noting that image_1 + caption isn't only one datum, but rather a collection of points of data based on the caption's length.

In a similar manner, Our data matrix will look like this if both the photos and their descriptions are taken into account:

| | Xi | | Yi | |
|----|----------------------|---|-------------|--|
| i | Image feature vector | Partial Caption | Target word | |
| 1 | Image_1 | startseq | the | data points corresponding to image 1 and its caption |
| 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 | data points corresponding to image 2 and its caption |
| 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 Matrix for Image_1 and Image_2 with their captions

We must now recognise that with every piece of data, not only the image, but also a partial caption, assists in the prediction of the next word in the sequence, acts as an input to the system.

We'll use a Recurrent Neural Network to interpret these incomplete captions because we're dealing with sequences (more on this later).

We've already established that we won't be passing the caption's unique English text; instead, we'll transmit a sequence of indices, each of which indicates a single word.

Let's examine what the information matrix would look like if we replaced the words with their indices now that we've created an index for each word:

| | | 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 |

After replacing the words with their indices, create a data matrix.

We want to ensure that all of the sequences are of equal length because we might be executing instruction execution (described later). As a result, at the conclusion of each sequence, we'd like to append 0's (zero padding). But, in each sequence, what percentage of zeros should we append?

This might explain why we set the maximum length of a caption at 34 characters (if you remember). As a result, we'll append those many zeros that can make every sequence to be 34 characters long.

The data matrix that we obtain looks like this:

| | | X_i | Y_i |
|-----|----------------------|--|-------------|
| 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 |

Appending zeros to each sequence to make them all of same length 34

What is the need of data generator:

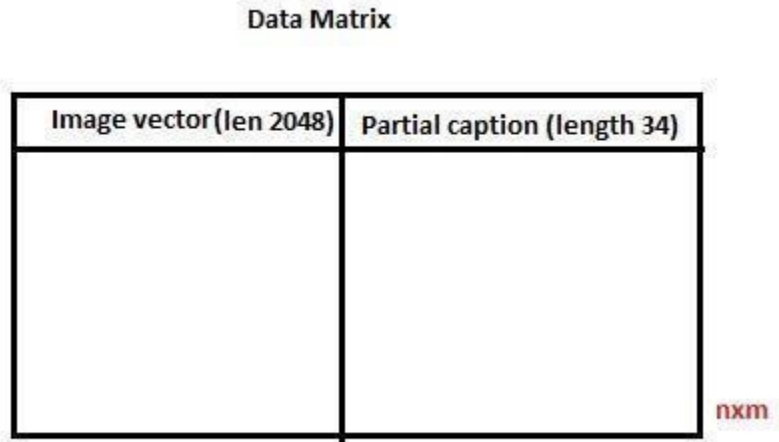
This, I believe, offers you a good idea of how we'll prepare the dataset for this topic. There is, however, a huge catch.

I've just considered two images and captions in the given example, yielding fifteen data points.

However, we have 6000 photos in our actual training dataset, each with five captions. This brings the total number of photos and captions to 30000.

Even if each caption on the median is only seven words long, the total number of data points will be 30000×7 , or 210000.

Computing the size of the data matrix is as follows:



data matrix size = $n \times m$

Where n stands for the no. of data points (assumed to be 210000), and m stands for the length of every data point.

Clearly, $m = \text{picture vector length (2048)} + \text{partial caption length (x)}$.

2048 Plus $x = m$

What's the value of x , though?

You might believe it's 34, But wait a minute, that's incorrect.

Through one of the word embedding approaches, each word (or index) will be mapped (embedded) to a higher dimensional space.

Every word/index is mapped to a 200-long vector using a pre-trained GLOVE word embedding model later on during the model building stage.

Each sequence now has 34 indices, each of which can be a 200-dimensional vector. As a result, $x = 34 * 200 = 6,800$.

As a result, $m = 2,048 + 6,800 = 8,848$

As a result, the size of the data matrix is equal to $210000 * 8848$, which equals 1858080000 blocks.

Whether we assume that one block occupies two bytes or not, we'll need at least 3 GB of main RAM to hold this data matrix.

This is a really large need, and whether or not If we can fit this much data into RAM, the system will be extremely sluggish.

This is why data generators are frequently used in deep learning. The data generators are a Python feature that's available out of the box. The Keras API's ImageDataGenerator class is nothing more than a Python application of the Generator Function.

So, how can employing a generator function help with this issue?

\

CHAPTER 04: PERFORMANCE ANALYSIS

1. Word Embeddings

As written above, we'll map every word (index) to a 200-long vector and for this purpose, we are using a pre-trained GLOVE Model:

```
# Load Glove vectors
glove_dir = 'dataset/glove'
embeddings_index = {} # empty dictionary
f = open(os.path.join(glove_dir, 'glove.6B.200d.txt'), encoding="utf-8")

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

We now generate an embedding matrix for each of the 1652 unique words in our lexicon, which can be fed into the model prior to training.

```
embedding_dim = 200

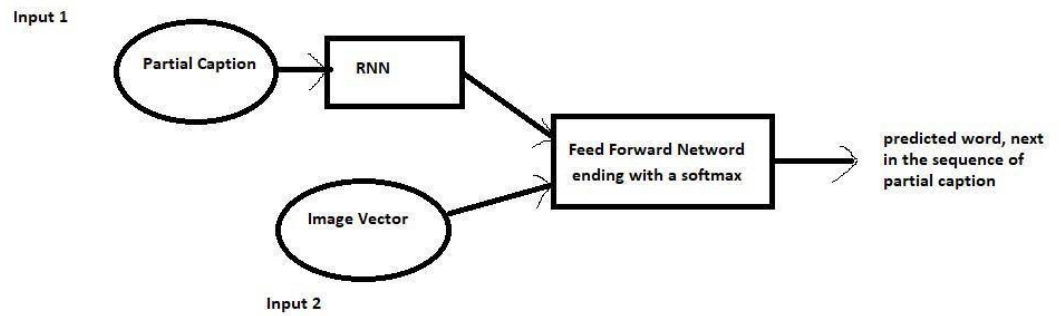
# Get 200-dim dense vector for each of the 10000 words in out
vocabulary
embedding_matrix = np.zeros((vocab_size, embedding_dim))

for word, i in wordtoix.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
```

2. Architecture of Model

We can't utilise the Sequential API supplied by the Keras library since the input consists of two parts: a picture vector and a partial caption. As a result, we employ the Functional API, which enables us to create Merge Models.

Let's start with a quick overview of the architecture, which includes the high-level sub-modules:



The model is given below:

```

1 # image feature extractor model
2 inputs1 = Input(shape=(2048,))
3 fe1 = Dropout(0.5)(inputs1)
4 fe2 = Dense(256, activation='relu')(fe1)
5
6 # partial caption sequence model
7 inputs2 = Input(shape=(max_length,))
8 se1 = Embedding(vocab_size, embedding_dim, mask_zero=True)(inputs2)
9 se2 = Dropout(0.5)(se1)
10 se3 = LSTM(256)(se2)
11
12 # decoder (feed forward) model
13 decoder1 = add([fe2, se3])
14 decoder2 = Dense(256, activation='relu')(decoder1)
15 outputs = Dense(vocab_size, activation='softmax')(decoder2)
16
17 # merge the two input models
18 model = Model(inputs=[inputs1, inputs2], outputs=outputs)
  
```

Code for defining the Model

Model Summary is mentioned below:

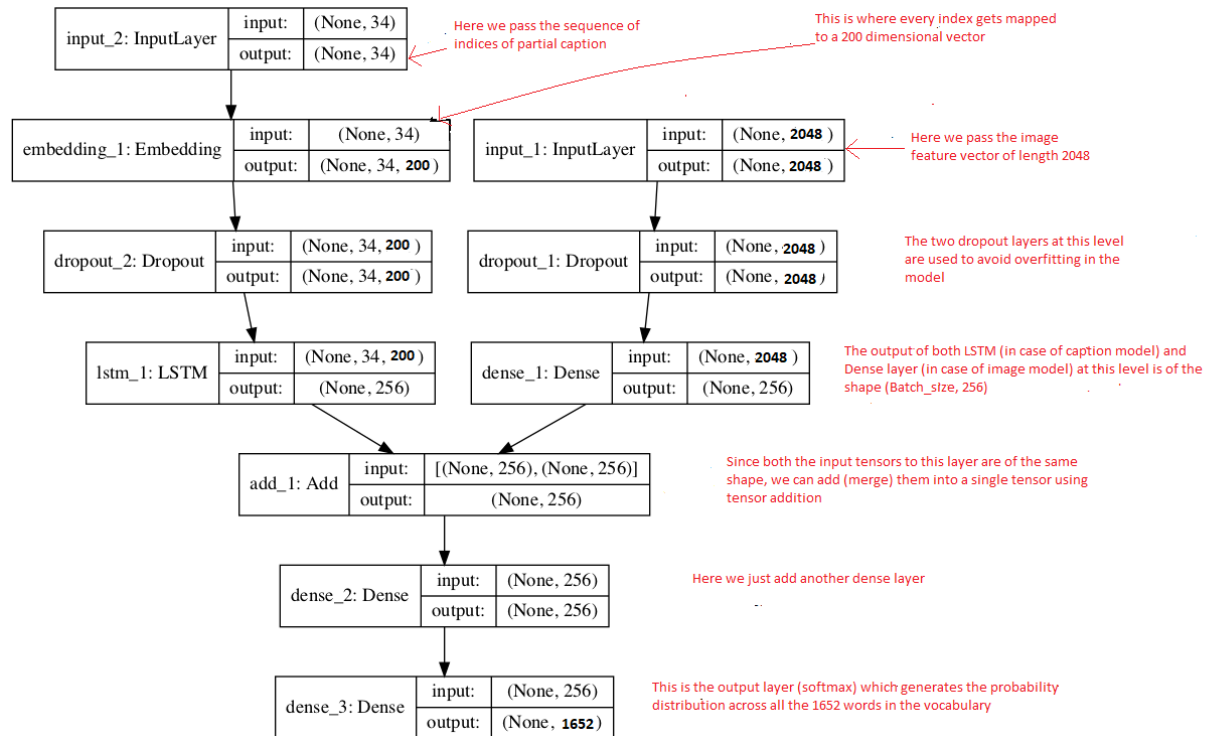
```
model.summary()
```

| 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

Model Summary

The below plot illustrates the network's topology and provides a clearer visual representation of two input streams:



Architecture Flowchart

The red coloured text are the comments for better understanding.

The long-short term memory layer (LSTM) is a proprietary recurrent neural network that analyses sequence input in the model, including incomplete captions.

Comments are written to understand the architecture in a better way.

Before beginning the training, we created an embedding matrix from an already trained Glove model, that one wish for incorporate into the model.:

```
model__layers.st__wght([ebdg__mtrx])
```

```
model__layers.trainable = False
```

Because we're implementing some already trained embedding layer, we have to freeze that before its training (`trainable = False`) to prevent it from being changed during backpropagation.

Finally, we use the Adam optimizer to assemble the model.

```
model__compile(loss='ctgrical__crossentropy', optmyzr='')
```

Finally, using a backpropagation technique, the model's weights will be adjusted, moreover it would learn to opt any word that is given a picture advantage vector and a half caption. At conclusion, results are:

Input_1 -> Half Caption

Input_2 -> Img Advantage vector

Following partial caption sequence in input 1, output -> A acceptable sentence (or in probability terms papers are saying conditioned on image vector and also half caption)

Hyper parameters during training:

After that, the model was trained for 30 epochs at a learning rate of 0.001 and three images each batch (batch size). The training rate was dropped to 0.0001 after 20 epochs, and the model was trained on 6 photos per batch.

This is generally wise because, as the model approaches convergence, we must lower the educational rate to take smaller steps towards the minima during the final stages of coaching. Gradually increasing the batch size improves the power of your gradient updates.

3. Inference

So far, we've seen how to prepare the data and construct the model. We'll learn how to test (infer) our model by passing in fresh images in the final phase of this series, i.e. how to write a caption for a replacement test image.

Remember that we only utilised the primary two images and their captions in the example where we demonstrated how to prepare the information. Let's take a look at the third image and see how we'd like the caption to be generated.

The third image vector and caption were as follows:



Testing Picture

A BLACK CAT is wandering through the green grass

Following is the vocab from ex we used:

Vocabulary: CAT, BLACK, end seq,is, grass, on, sat, start
seq,road, white, the

The caption will be created in phases, with each word in each step
being generated iteratively.

Step 1:

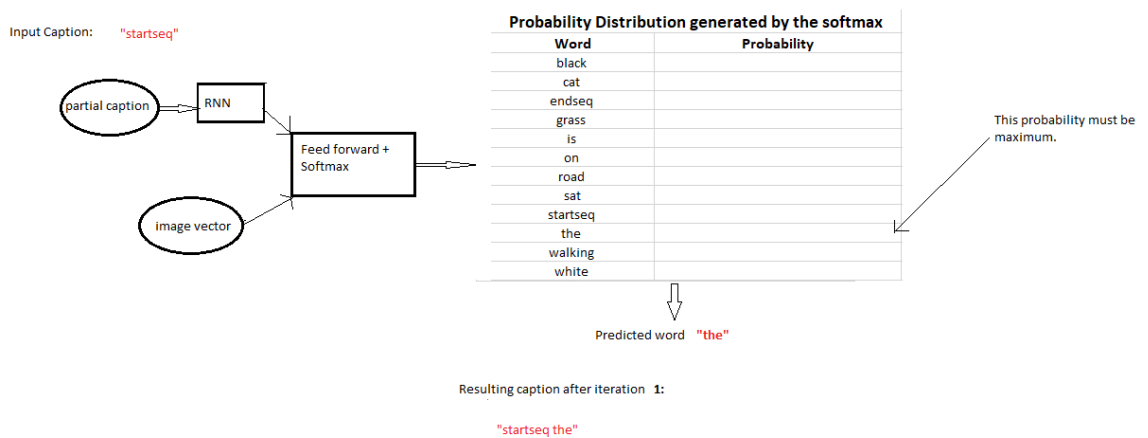
$I_{pt} = \text{Img vector} + \text{"startseq"}$

Predicted word = "THE"

(The token 'start seq' is extremely important during inference because it is utilised as any image with a first partial caption.)

But, it creates a 12-long vector (a 1641-long vector of original ex) that may be a distribution for most of vocab items. As a result, we greedily select a word with the highest likelihood given the advantage vector and half-baked caption.

We should anticipate that prob of word "the" will have to have the highest if the Model is well-trained:



Step I

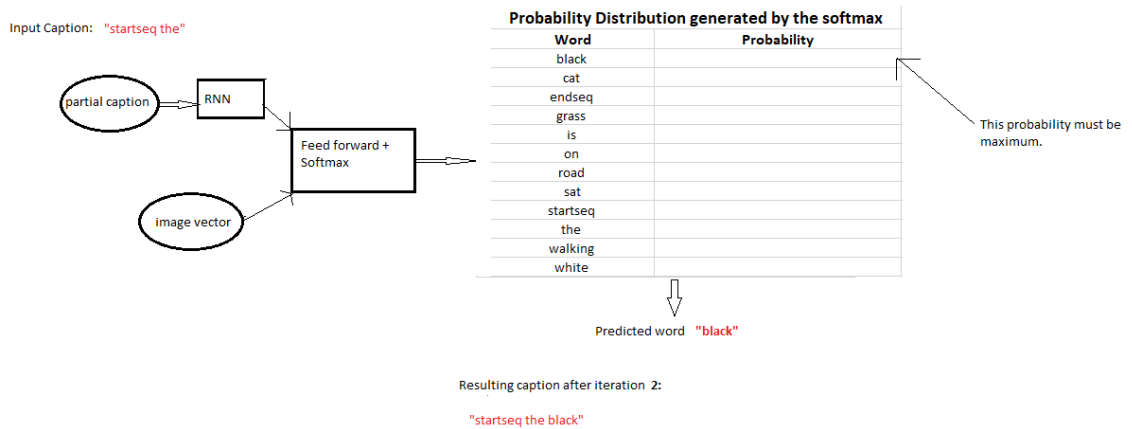
Maximum Likelihood Estimation is the name of the method (MLE).

Greedy Search is another term for it.

Step 2:

$I_{pt} = \text{Img vector} + \text{"start the"}$

Predicted word = "BLACK"

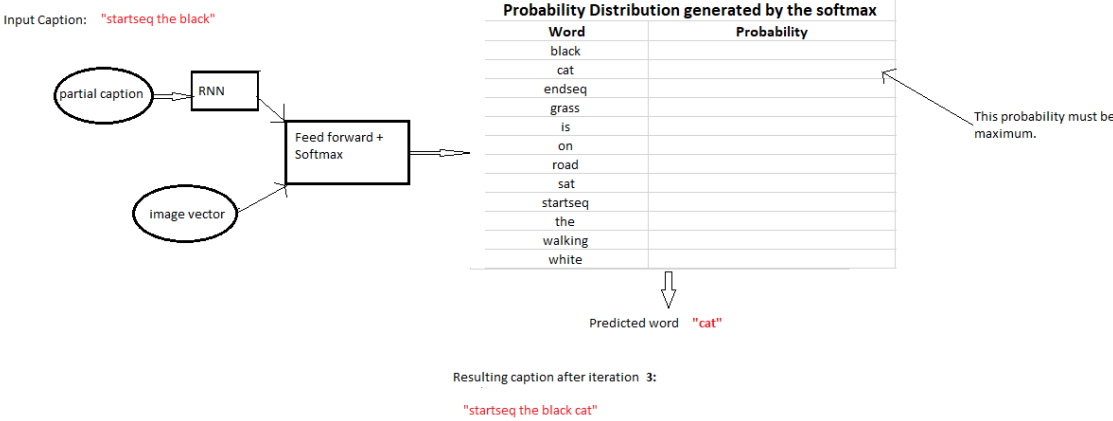


Step II

Step 3:

$I_{pt} = \text{Img vector} + \text{"start black"}$

Predicted word = "CAT"

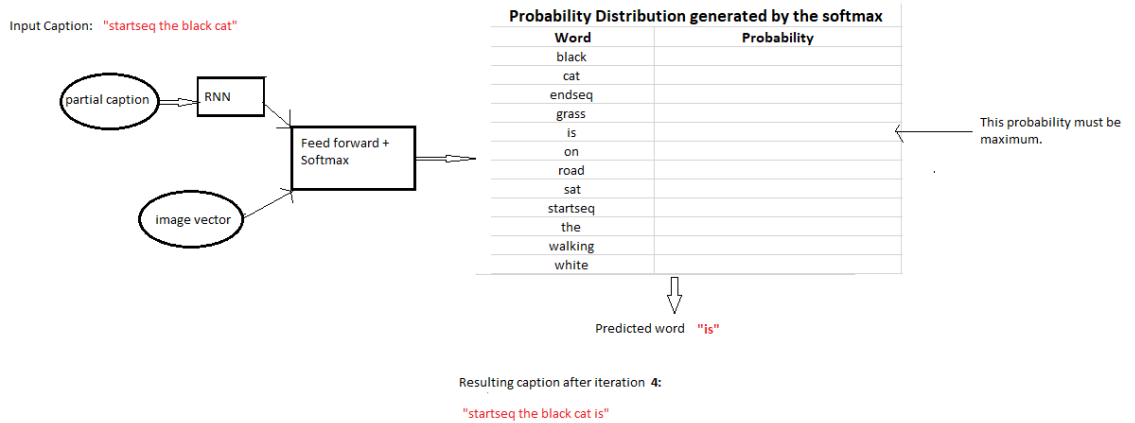


Iteration Step III

Step 4:

$I_{pt} = \text{Img vector} + \text{"startcat"}$

Predicted word: "IS"

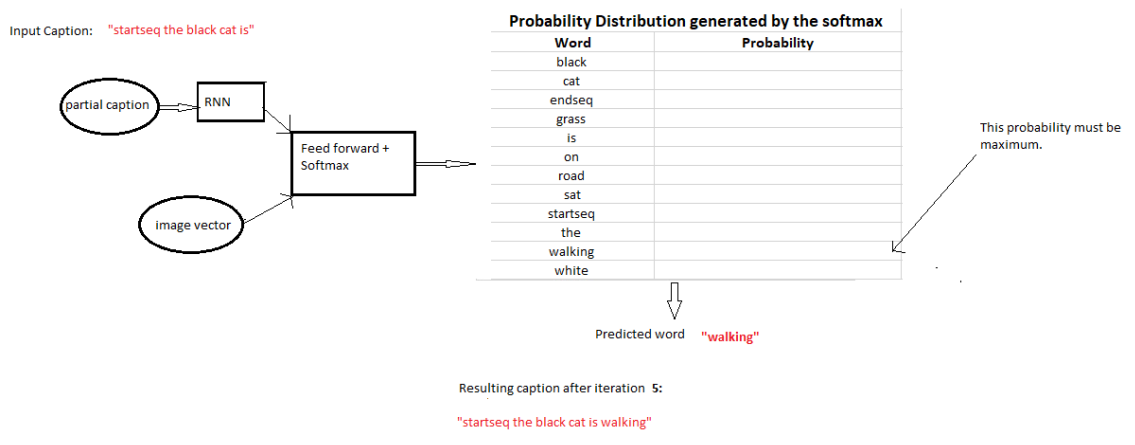


Step IV

Step 5:

$I_{pt} = \text{Img vector} + \text{"startseq black cat is"}$

Predicted word: "WALKING"

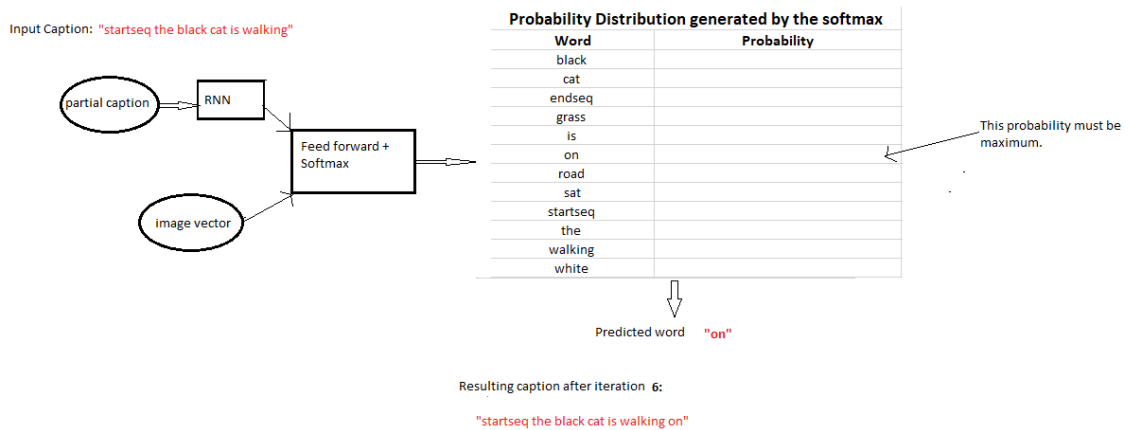


Step V

Step 6:

$I_{pt} = \text{Img vector} + \text{"startseq is walking"}$

Predicted word = "ON"

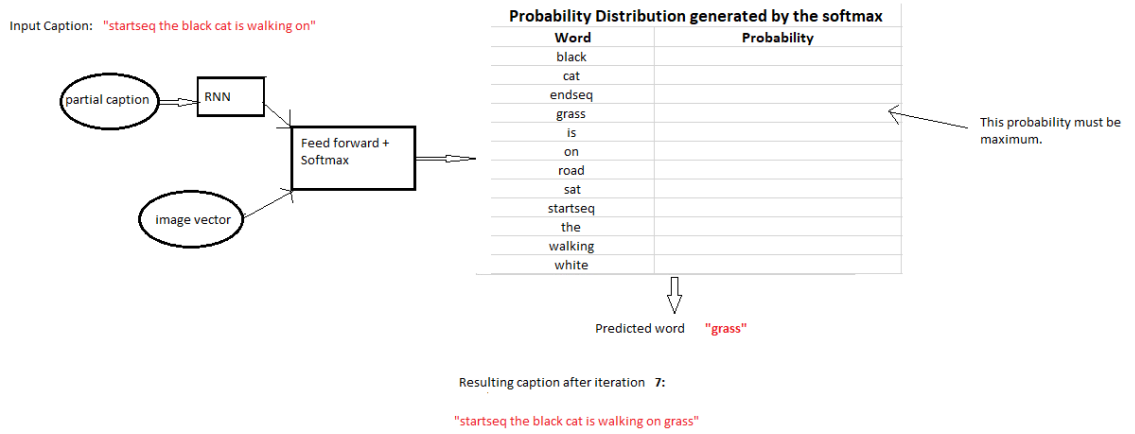


Step VI

Step 7:

$I_{pt} = \text{Img vector} + \text{"cat is walking on"}$

Predicted word = "GRASS"



Iteration Step 7
 Predicted Caption = “ walking on grass”

4. Evaluation

Some of the OUTPUT'S of our model is:

Ex-1

man in red shirt is standing in front of crowd



Ex-2

man in red shirt and black mask is standing in front of crowd



Ex-3

brown dog is running on the grass

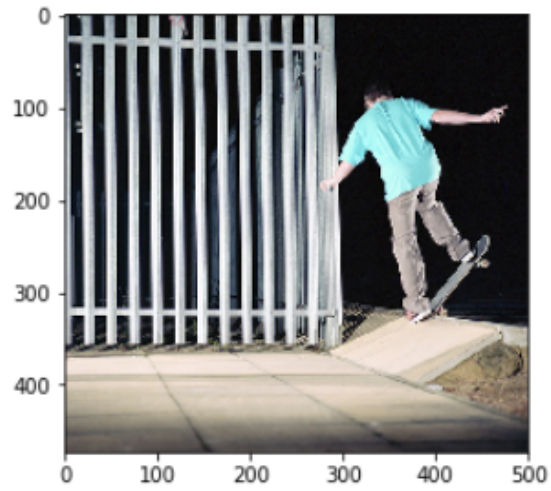


Ex - 4

woman in blue shirt is standing in front of some bushes

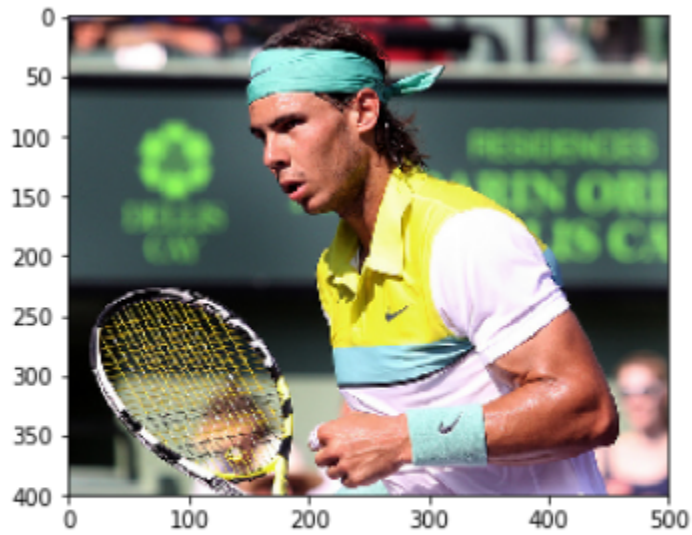


Ex-5



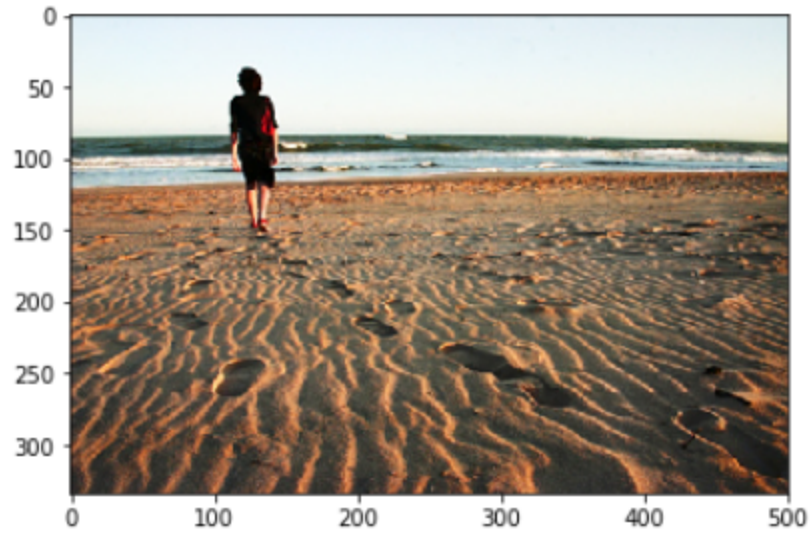
Greedy: man in black shirt is skateboarding down ramp

Ex-6

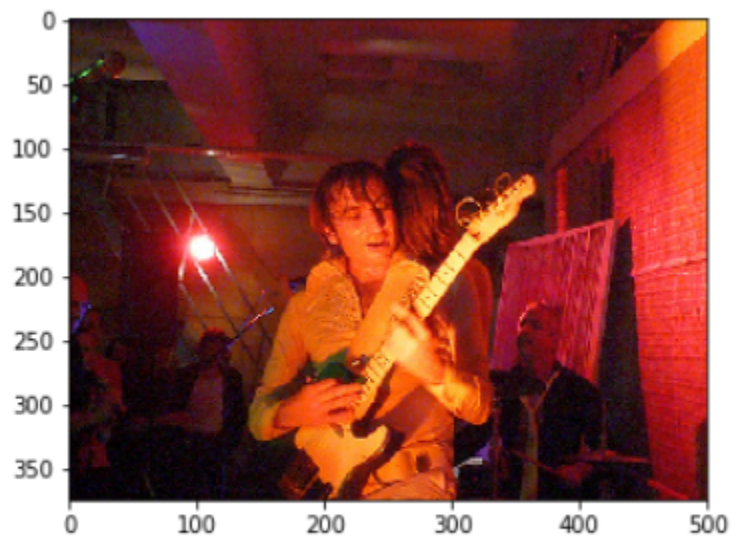


Greedy: a woman in a tennis racket on the court .

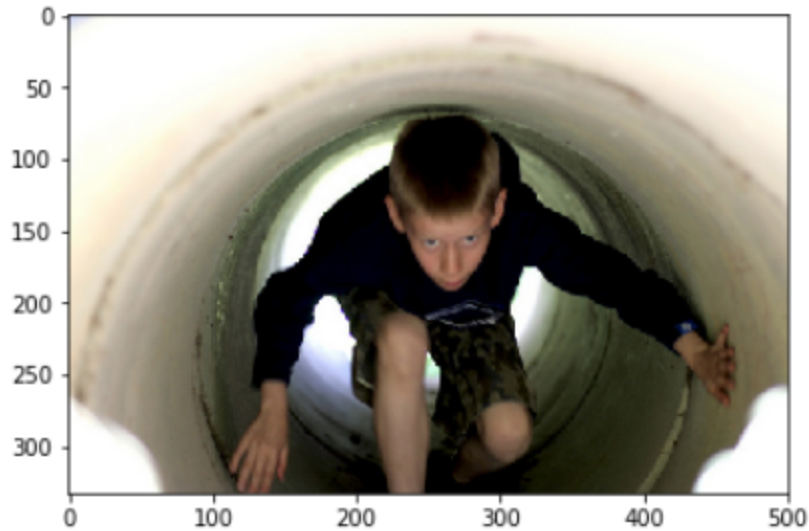
Ex-7



Greedy: a boy is walking on the beach with the ocean .
Ex— 8



Greedy: a man in a guitar for a .
Ex-9



Greedy: a little boy is on the water on the floor with a over the floor

Ex-10

These were some of our project's outcomes.

Important Point:

It should be noted that the images used for testing should be associated with the training images. As a hypothesis, should we make our model on the photographs in plants and animals, one shouldn't test on photographs related to cars and humans which will lead to inaccuracy.

CHAPTER 05: CONCLUSIONS

- **PROJECT LIMITATIONS:**

Modifications we could use are:

- Having a larger set for data.
- Using code during object oriented manner in order so it is easier to duplicate
- Using cross validation and understanding overfitting
- Architecture model changed
- Doing a more extensive tuning of parameter

References:

- **Andrej Karpathy. Deep Visual-Semantic Alignments,** <https://cs.stanford.edu/people/karpathy/cvpr2015.pdf>
- **Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan. Show and Tell: A Neural Image Caption Generator,** <https://arxiv.org/abs/1411.4555>
- **Marc Tanti. Where to put the Image in an Image Caption Generator,** <https://arxiv.org/abs/1703.09137>
- **Albert Gatt. What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?,** <https://arxiv.org/abs/1708.02043>
- **Jason Brownlee. How to Develop a Deep Learning Photo Caption Generator from Scratch,** <https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/>