Image Captioning Bot

Major Project Report

Submitted by:

Shubham Gupta (9917102219)

Sarthak Samarth (9917102112)

Nimish Sahdev (9917102117)

Under the supervision of:

Dr. Kapil Dev Tyagi

Department of ECE

Jaypee Institute of Information Technology University, Noida

December 2020

CERTIFICATE

It is to certify that this Major Project Report titled as " IMAGE CAPTIONING BOT ", submitted by **Shubham Gupta, Sarthak Samarth & Nimish Sahdev** in realization of the requirements for the award of degree of Bachelor of Technology in **Electronics & Communication Engineering** from **Jaypee Institute of Information Technology, Noida** in their final year, is a genuine work accomplished by them under my supervision. The set of submissions in the form of this report is distinct & has not been tendered for the award of any other degree.

Name of the Supervisor: DR. KAPIL DEV TYAGI
ECE Department
JIIT, Sec-128
Noida-201304

ACKNOWLEDGEMENT

The fulfillment of this inter-disciplinary project depends upon combined & coordinated efforts of various knowledge resource. We are thankful to **Dr. Kapil Dev Tyagi** for his preparedness to give us precious counsel throughout. We would also like to greet our college authorities for allowing us to continue with such a project in such an emerging field & helped us to accomplish it.

DECLARATION

We preconize that this project submission reflect our own thoughts and findings. And, wherever external thoughts and findings are included, the related source/references are adequately adduced. We also preconize that we have adhered to academic honesty, integrity & have not indulged in any fabrication or misrepresentation of the submissions made in the form of this report.

Shubham Gupta (9917102219)

Sarthak Samarth (9917102112)

Nimish Sahdev (9917102117)

INDEX OF CONTENT

INDEX OF FIGURES

INDEX OF TABLES

ABSTRACT

In the times we are in, we can see how much advancement and improvement there has been in the field of AI & ML. Image captioning has attracted the researchers within the field of AI & has emerged as a fascinating & onerous domain. Image Captioning, automatically produced natural language descriptions in step with content examined in a picture, is an essential part of scenic know-how, which basically merges the expertise of computational vision & natural language processing. The application of image captioning is tremendous & significant, for instance, the realization of human-computation interaction. This submission summarizes the associated techniques & focuses on Transfer Learning, which plays important function in computational vision & is vastly used in image captioning - generation tasks, especially in recent times. Further, the benefits & the lacking of these strategies are discussed, employing the commonly used datasets & evaluation criteria in this area.

This vast field of technology is advancing, improving and being explored to be used to enhance our lifestyle. In this project, we've attempted to develop a bot that can do "image captioning tasks". If their accuracy is good enough, it can manage massive amount of media data, & generate descriptions in the language perceivable to humans.

In this project, we go deep into the concepts and domain of image captioning, especially dense image captioning. We will present the technological foundations of a model as the attempt is to better and advance such such tasks. Concretely, a structure of DenseCap & Neural Image Captioning will be executed in detail.

CHAPTER-1
PROBLEM ANALYSIS

1.1 INTRODUCTION

Analysis of problem includes detailed examination of the problems/lackings to be addressed and the advancements needed. It gives the identification of the gap b/w the present state & wanted (desired) state of a procedure or product. The first move of fixing a problem/lacking is knowing the situation, that can be carried out by this analysis.

Problem-statements are vastly utilized by various industries to develop or/and improve solutions, processes or/and procedures. A simplified & nicely described problem-statement is used by the professionals to recognize the problem/lacking(s) & work towards the required solution/development/improvement. It also helps the administrators and decision-makers with sharp insights to help them to make suitable choices and decisions. As such, this problem statement must be perfect & unambiguous.

It is significant to be aware that this analysis do not outline the strategies or solutions for accomplishing the desired goal. This analysis without doubt gives the identification and examination of the gap b/w the present & wanted states. It can be stated that "a problem properly said is 1/2 of solved." However, there are multiple feasible answers to a problem/lacking. Only after this analysis is written & agreed upon, the solution(s) to be identified and the path forward be decided.

1.2 PROBLEM IDENTIFIED

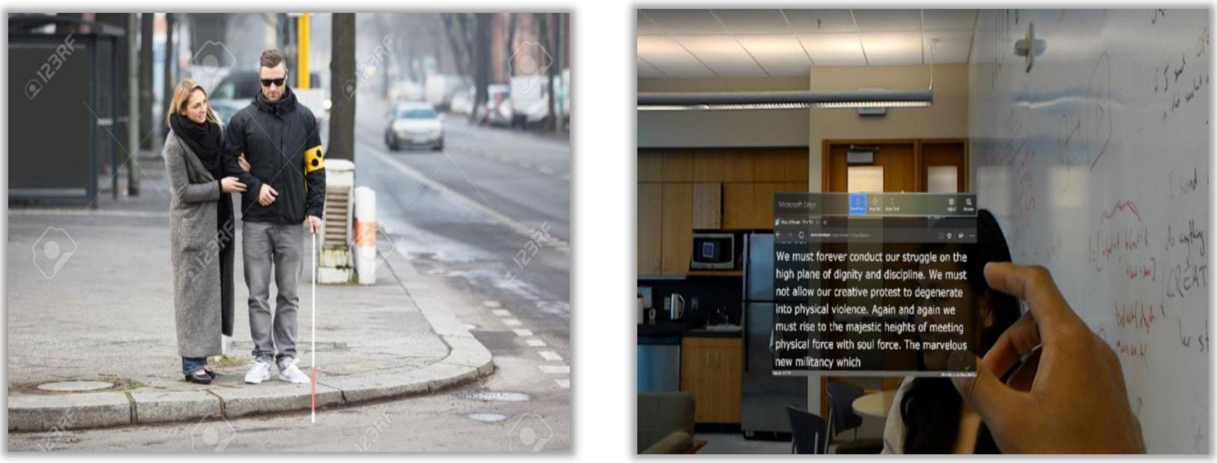
In the times we are in, one of the biggest challenges we face is the large information that exists and is growing rapidly. Most of this information is distinct from conventional information & the media or digital type occupies the major portions. These mostly arise from internet platforms including social networks, digital media, etc.

Apart from the reality that humans can work upon and utilize these media/digital information, what and the extent to which the computational devices can presently acquire from them is limited & their possible assistance in the work and taking it forward is also limited. "Image captioning applications", if they are accurate enough, they can manage big amounts of media/digital information & generate descriptions in human perceivable language. The machine will be able to better assist human beings to use these information materials to do better and more. We must understand how important this problem is to real-world occurences. Now we will see few applications where such solution will be significantly useful.

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

Fig 1.1 : Self Driving car recognizing objects & generating a caption which is fed back into it to take actions accordingly

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



We must forever conduct our struggle on the high plane of dignity and discipline. We must not allow our creative protest to degenerate into physical violence. Again and again we must rise to the majestic heights of meeting physical force with soul force. The marvelous new militancy which

Fig 1.2 : Blind Person can't see but pic captioning can understand what is happening in front so it will generate a caption for it read it loud in ears of the blind person

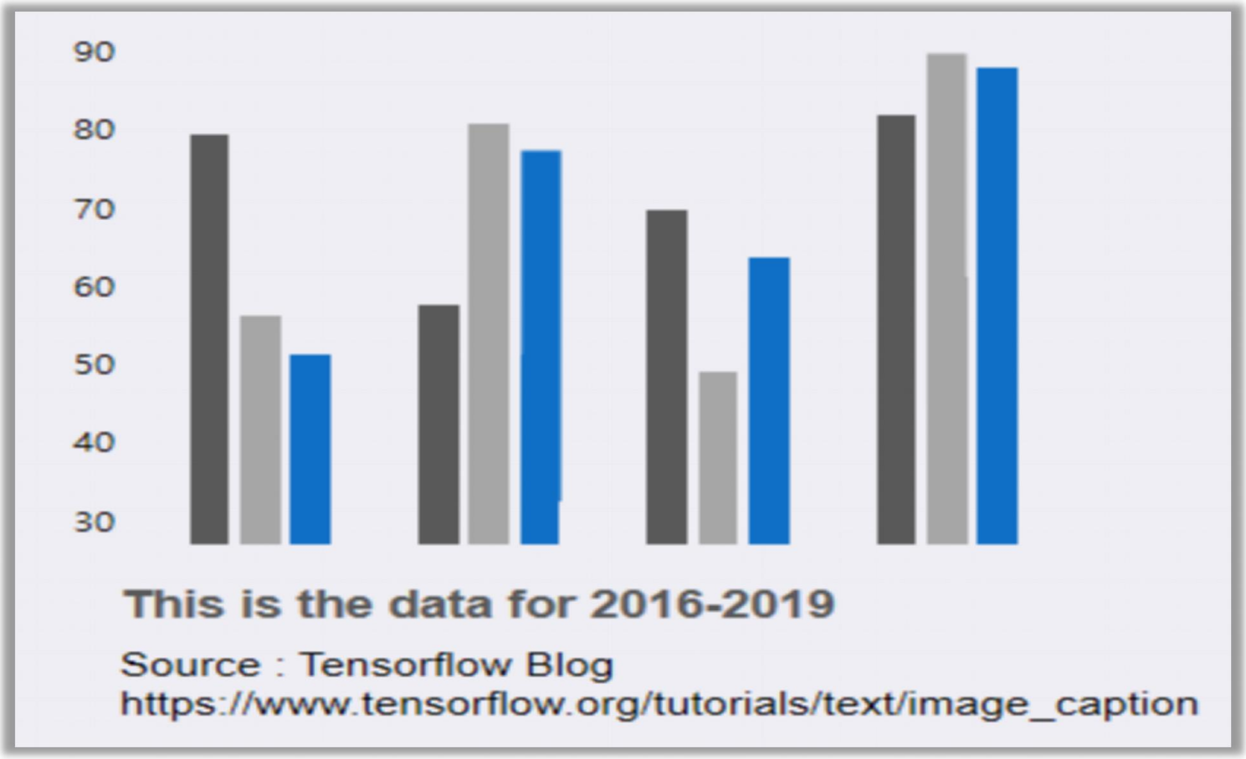
CCTV cameras are present everywhere, but along with keeping eye on this world, if we can also generate pertinent description, then we can raise alarms as soon as there is some unwanted activity going on somewhere. This could probably help reduce some crime and/or accidents.

Automatic Captioning can help, make pic Search as good as Google Search since after that every pic could be first allowed to generate a caption, & then the search can be performed based on the caption.

Robots with emotion, This project is base of various opportunities like Robotic emotion Development **e.g. Sophia the first humanoid who got citizenship in Saudi Arabia.**

1.3 STATISTICS

[1] A Comprehensive Survey of Deep Learning for pic captioning



This is the data for 2016-2019

Source : Tensorflow Blog
https://www.tensorflow.org/tutorials/text/image_caption

86%

Over 86% of car crashes are caused by driver error. There would be fewer user errors & fewer mistakes on the roads if all vehicles became driverless. Drunk & drugged drivers would also be a thing of the past, & passengers might even sleep without risking safety. & Driving fatigue & getting lost would be things of the past.

84%

With 84% accuracy, Mars Exploration Rover can detect the weather & climatic situation.
Also with almost the same accuracy +<5, it can detect the metal, non-metals, minerals present on the surface of Mars.

89%

89% accuracy achievement for the first robot to receive Saudi Arabian citizenship - Sophia, to detect the scenario & plattering the thoughts about it

1.4 WHAT IS MISSING?

We have already made our API ready to be deployed. We can take our pic captioning bot on Cloud but somehow Cloud Pricing is too high for student to match.

1.5 CONCLUSION

Problem statements are reached upon after thorough study of the various aspects & no. of problems which can be solved by pic Captioning Bot. & Then, missing points are formulated.

CHAPTER-2
SOLUTION & SYSTEM ANALYSIS

2.1 INTRODUCTION

A solution is to identify goals & purposes, & create systems & procedures that will efficiently achieve them. A way to look at system evaluation is as an analytic methodology that includes breaking down a process into its rudimentary portions for assessing how good these portions work & have interaction, to achieve the desired outcome/purpose.

The area of system evaluation relates to evaluation of requisites or operations research. It is "an absolute formal inquiry carried out to help a choice-maker analyze a better course of actions & accomplish better decision than he might otherwise have made."

System analysis is acclimated in every domain, wherever something is developed. Analysis can be a sequence of parts & pieces that carry out important roles together, like engineering of systems. The interdisciplinary area of engineering, that puts emphasis on how complicated engineering missions have to be designed & managed, is System Engineering

2.2 OUR SOLUTION

We implemented a deep recurrent architecture that automatically make a short description for pics. Our models use a CNN, which was pre-trained on ImageNet to obtain pic features. We then feed these features into an LSTM network to generate a description of the pic in simple English. After this, we will cover the feature extraction process using CNN.

After the feature extraction process, we come towards the Recurrent NN where we will use embedding matrix LSTM & glove vector. Then we will use Google's Pre Trained model i.e. Resnet-50 (picnet).
Let us see how we are going to perform all this step by step:

2.2.1 Collection of Dataset

There are multiple open-supply datasets present for such problems, like COCO (containing 180k pics), Flickr8k (carries 8,000 pics), Flickr30k (contain 30k pics), etc. However for this project, we have used the Flickr8k dataset, which includes 8000 pics. Each picture has five captions (like we previously saw in the Introduction segment that a picture can have numerous captions, all being pertinent side by side).

These pictures are forked as follows:

Dev Folder— 1000 Snaps

Training Folder — 6000 Snaps

Test Folder — 1000 Snaps

2.2.2 Data Manipulation

When we download this dataset on Kaggle, then, other than the pictures, we will also get number of files containing the text related to the pics. So, one file is "Flickr8k.token.txt" carrying name of every pic along with its five captions. We may have the file as shown:

Therefore every row in file carry the picture name & n-descriptions, where 0≤n≤4. This includes pic name, description number (0 to 4), & real description.

Now, make a dictionary with the title "descriptions" carrying name of pictures (with removed .jpg) as keys & 5 captions range for corresponding pictures as values.

2.2.3 Filtering and Cleansing Data

Every time working with text require cleaning like changing words into lower case (otherwise"major" & "MAJOR" will be considered as 2 different words), deleting spl. words (like '^', '*', '!', etc.), removing alphanumeric words (e.g. 'Martin1909').

Creating vocab of all different words across 8K*5 = 40K pic-descriptions in the dataset :

We have 8,763 distinct words across all 40k pic-captions. We write these descriptions with pics in file titled - *"descriptions.txt" &* import this in the drive.

Various words occurs less times, say twice or thrice. Because we need to create a model that can predict, we might never desire to play with all diff. words within vocab. The words with higher likelihood of occurence are given preference. The model hence become stronger, more accurate & less prone to errors.

Therefore we consider those words which **occurs minimum ten times** in whole text file. So, now we've 1,651 distinct words in the vocab. Although, we'll append zer0's.

Hence Total words = 1,651+1 = **1,652** (1 -> index for Zer0).

2.2.4 Loading of Training Set

Document named "Flickr_8k.trainpics.txt" carries name of pics that belongs to file of training . Therefore, we put these names in -> "train". Hence we've put the 6K training pictures in a catalogue named "train". Now, load the captions of these pics from dict. "descriptions.txt" (in hard drive) in dict. "train_descriptions".

Whenever the captions are loaded, the following 2 tokens are added in each of the captions:

'**startseq**' -> It is token of start-sequence that will be put to the beginning of each description.

'**endseq**' -> It is that token of the sequence which will be put at the ending of each description.

2.2.5 Image Preprocessing

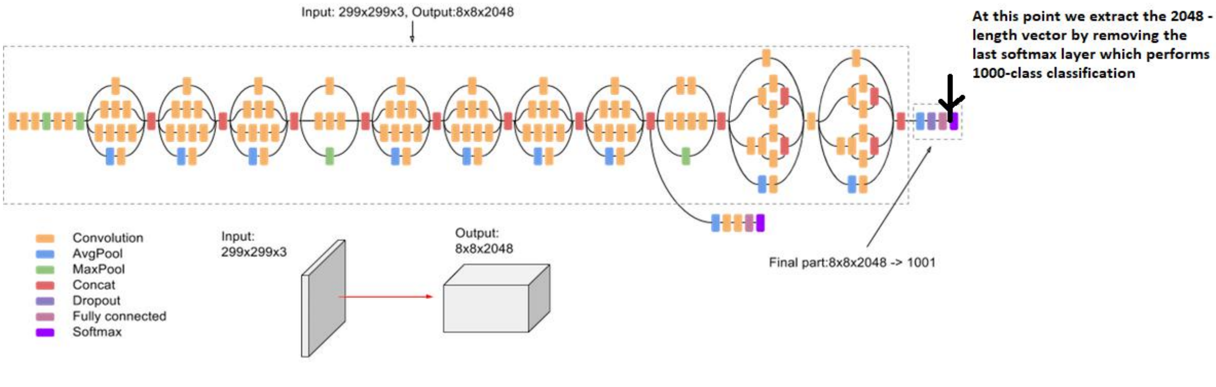
Pictures are just i/p to the model. Note that any i/p has to be in the form of vector to a model for both training & testing.

We want to transform each pic into a vector of fixed size which is fed as i/p to NN(Neural-Network). For this motive only, we choose **transfer-learning** by employing Inception-V3 (CNN model) made with the aid of Google-Research.

This NN-model (Inception) was trained over dataset named Imagenet, where classification of pics was done over 1K diff. classes of pics. But our motive is not the image-classification rather to get a vector of fixed length for every pic. Hence it is known as **automatic feature engineering.**

Therefore, we eliminate the soft-max layer (last-layer) in the NN(Neural-Network) model & extract a 2,048 length vector for each pic as follows:

Fig 2.1 : Feature Vector Extraction (Feature Engineering) from InceptionV3 (Googlenet)


Input: 299x299x3, Output:8x8x2048
At this point we extract the 2048 - length vector by removing the last softmax layer which performs 1000-class classification
Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax
Input: 299x299x3
Output: 8x8x2048
Final part:8x8x2048 -> 1001

We saved all features of train file in a py dict. & saved these on hard drive using the .pkl file (Pickle), named "**encoded_train_pics.pkl**", which has keys as names of the pics & values correspond to the feature-vector of size 2,048. Likewise, test pics are encoded & we save these on hard drive using the file, "**encoded_test_pics.pkl**”.

2.2.6 Description Preprocessing

We know that brief description of picture is something that we need the model to form and give as o/p. So, at the time of training, descriptions are the goal/targeted variables (Y) that model is studying & learning to be able to predict.

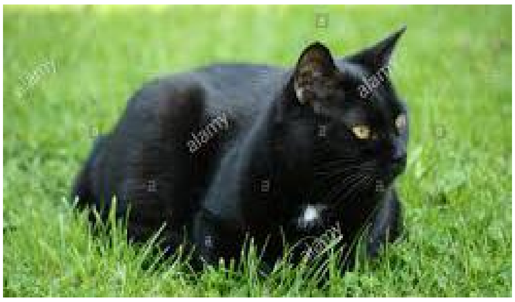
Description will be predicted **word by word**. Hence, we have to encode every word into a vector of fixed size. After that we’ll make 2 Py Dictionaries namely "wordtoidx" (pronounced as word-to-index) & "idxtoword" (pronounced ad index-to-word).

2.2.7 Data Loader & Generator:

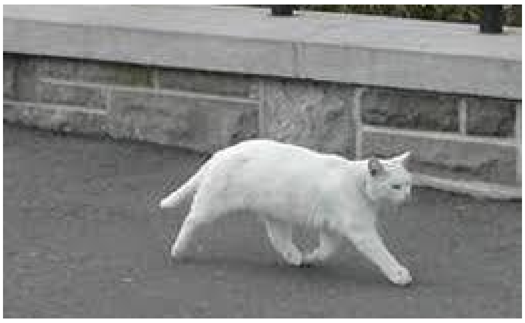
Here we’ll see how assembling of the data took place in a way which will be a suitable method to be supplied to the deep learning model as i/p and that will eventually lead to training of data.

This will be applied to during prediction as well.

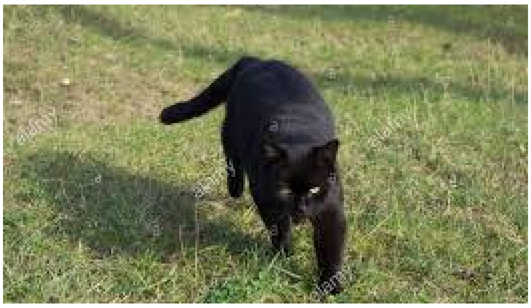
Assume we have 3 pics & 3 corresponding descriptions as shown below:



(Training pic-1)
Description -> Black coloured cat relaxing over grass



(Training pic-2)
Description -> White coloured cat roaming across the road



(Validation pic)
Description -> Black coloured cat is roaming on grass

Assume we use initial **2 pics** & the corresponding descriptions for model **training** & likewise **3rd pic** for **validation** of our model.

Firstly, convert pics into feature vector of 2,048 length as discussed previously. Let us say "**pic_1**" & "**pic_2**" be the vectors of starting 2 pics respectively.

Secondly, make vocab for training pics descriptions by putting the word token "startseq" & "endseq" in description of 2 pics: (supposing we've done the sentence/token cleansing already)

Description_1 : "startseq Black cat sat on the grass endseq"

Description_2 : "startseq White cat is walking on the road endseq"

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

Give a idx to every token in the vocab:

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

Let's look at list of word token D = {Xi, Yi}, where Yi is corresponding target variable & Xi is vector of 'i'.

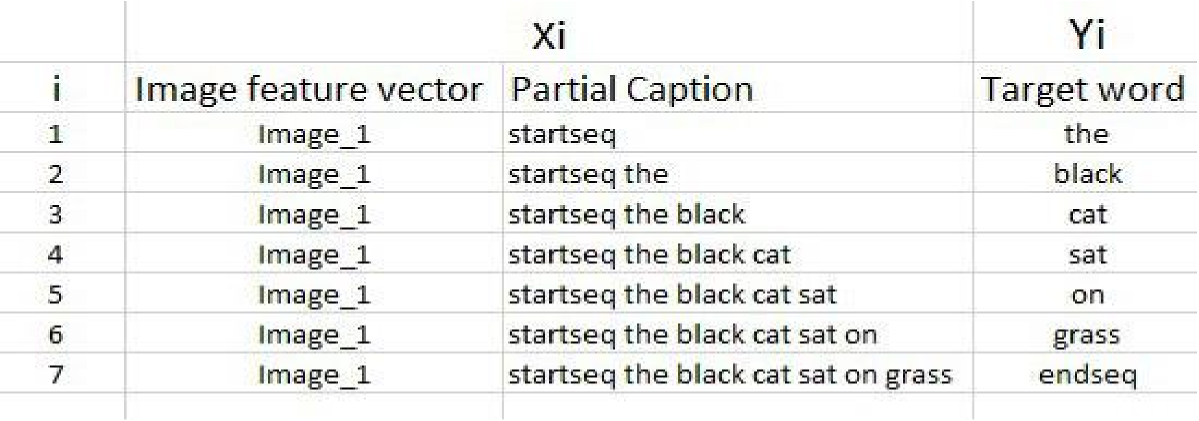
Let us look at **pic_1** feature vector & its description "**startseq the black cat sat on grass endseq**". Remember pic vector is i/p & the description is o/p which model is going to frame on its own. But the method we frame description is as follows:

Initially, we see pic-vector & starting word as I/p & let model frame the 2nd word : I/p = pic_1 + 'startseq'; O/p = 'the'

& This continues until we send all word of sentence for training

Summary for the data matrix for one pic & its corresponding description is as follows:

Table 2.2.7.1 Word/token point for 1 pic & framing its each predicted description



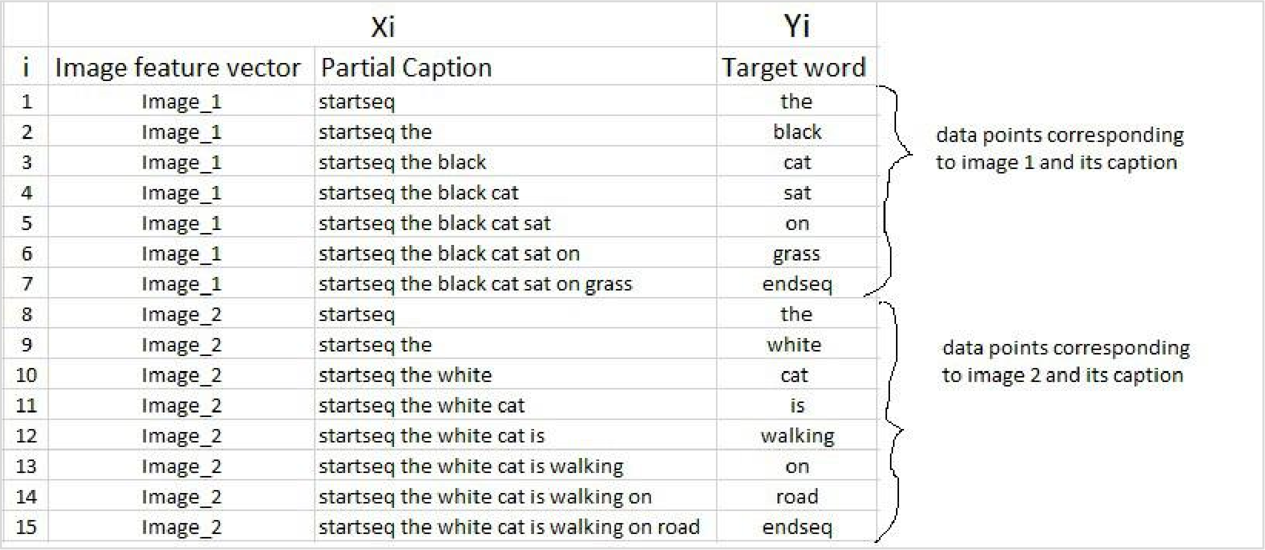
	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	startseq	the
2	Image_1	startseq the	black
3	Image_1	startseq the black	cat
4	Image_1	startseq the black cat	sat
5	Image_1	startseq the black cat sat	on
6	Image_1	startseq the black cat sat on	grass
7	Image_1	startseq the black cat sat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

[4] Guided Open Vocabulary pic Captioning

It should be marked that 1 pic + description is **not just considered as one data point** but are many data points based on description length.

Likewise, look at both pics & their descriptions, our data matrix will then appear as:

Table 2.2.7.1 Data Matrix (Data points) for both the pics & captions



	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	

In training-dataset, we've 6K pics, each having five descriptions. This implies 30k descriptions in totality. If we count in every caption, these are just 7 words long on avg., which will 30k*7 i.e. 210k data points.

We employ Data Loader & Generators massively for this reason: Data Loader plays crucial role that is natively created in Python. Keras API provide the Data Generator module which is nothing more than the application of the generator function .

2.2.8 Embedding Matrix & Glove Vector

As said previously, we'll map each index [word] to the vector of length 200 & for same motive, we used a GLOVE Model which is pretrained.

For each of the 1,652 distinct words in the vocab, build a matrix [embedding] that was introduced into the NN-model before this training .

2.2.9 Model Architecture

Because i/p consists of 2 parts, a pic-vector & a description, don't use Keras App. Program Interface-Sequential. We can use the Functional API which permit us to Merge the Models for this reason.

Firstly, let us see model carrying high-degree sub-process implementation:

So, This is how Descriptions and pic vectors will be fed to NN for training

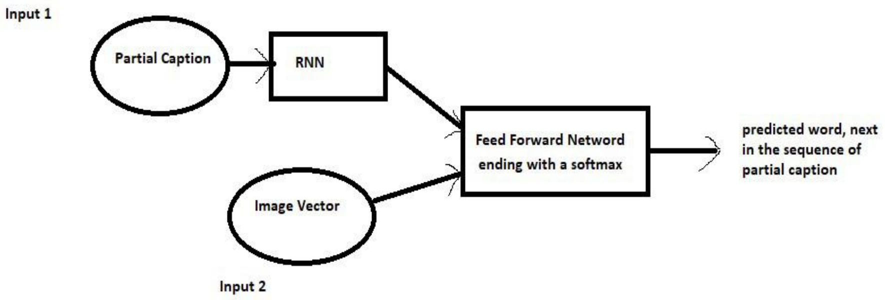
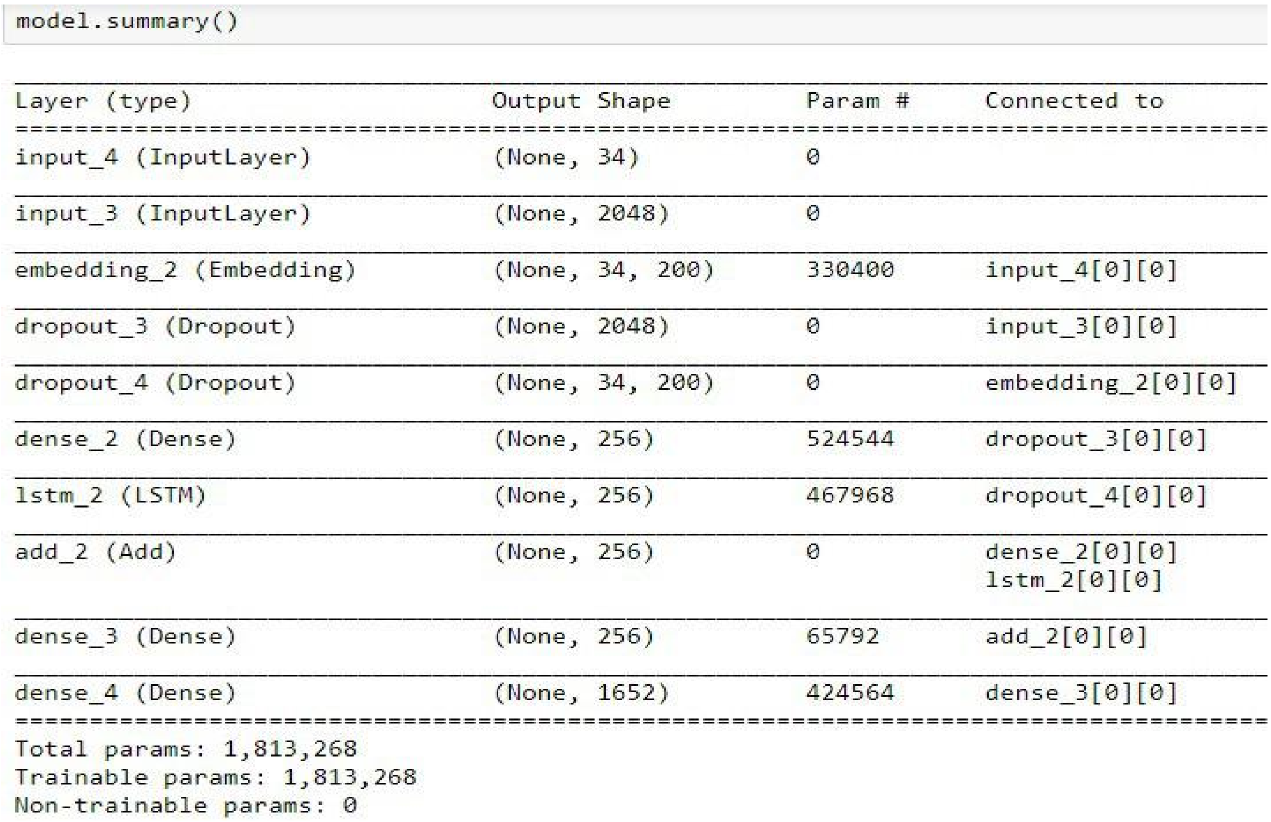

Input 1
Partial Caption
RNN
Feed Forward Netword ending with a softmax
predicted word, next in the sequence of partial caption
Image Vector
Input 2

Table 2.2.9.1 Here we define the model summary:

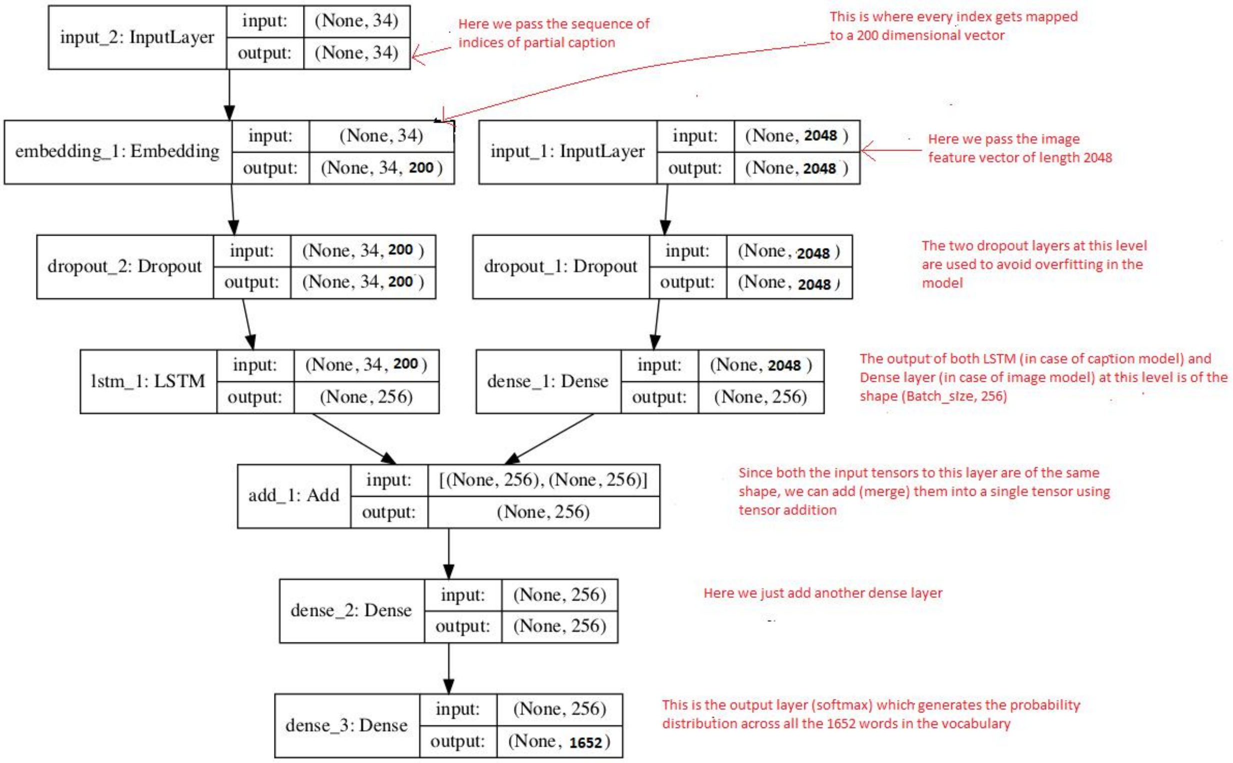


```
model.summary()
```

```
________________________________________________________________________________________
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
```

Here you can see how CNN layers are applied and how parameters are learned - LSTM, dropouts, fully connected layers

Table 2.2.9.2 Summary: the parameters within the model in layer-wise feeding



Here we pass the sequence of indices of partial caption

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

Here we pass the image feature vector of length 2048

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

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

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

Here we just add another dense layer

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

The pink textual write-up towards right are feedback commentary given to understand preparation of the data/information for model architecture.

The **LSTM** is a specialised RNN [Recurent] to prepare the entire series i/p sequence (partial descriptions in our case).

Since a pre-trained embedding layer is being used by us, we require to falsify the training for last two layers, before model training, so as to prevent updation of other previous layers during backpropogation.

Finally, compilation of model is being done using adam optimizer. **Backpropagation algorithm** will lead to updating of weights & having a pic

[3]Deep Reinforcement Learning-based pic Captioning with Embedding

Feature-vector & a description ,framing a word will be learned by model, so here is summary:

I/p_one : Partial Description

I/p_two: Pic feature Vector

O/p : An suitable word, subsequent within the series of partial description provided in the I/p_1 (or in probability terms, **conditioned** on the partial description & vector of pic)

2.2.10 Parameters Training:

Training of model is done within 30 epochs where learning rate will be 1/1000 & there will be 3 pics in each group group. Hence After twenty epochs, we see reduction of 1/10000 in learning rate & the model gets ready by being trained for 6 pics per group.

2.2.11 Inference

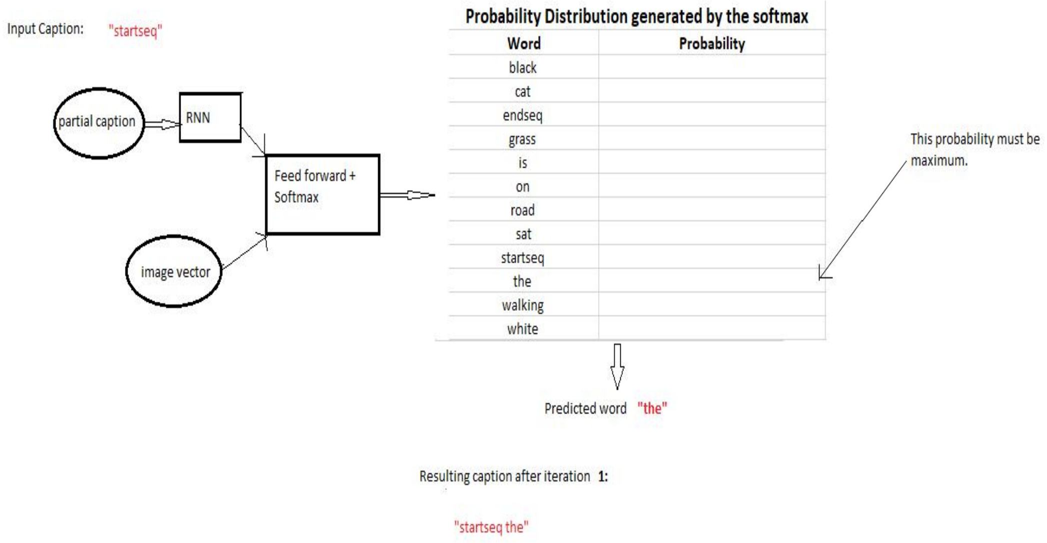
Until now, we've shown how data pre-processing took place & also constructively helps in making model. As last step, model will generate a description for validation/test pic.

In previous instance , we look over pre-process data, where only first 2 pics & their descriptions were used. Now we will work with 3rd pic & understand how should description to be framed. The 3rd pic-description & vector were as shown:

Description -> Black coloured cat is roaming over grass field

The description will be generated iteratively and so will the model work, one word at an instance as shown:

Itr 1: I/p : pic_feature_vector +“startseq” **Expected O/p :** “the”


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

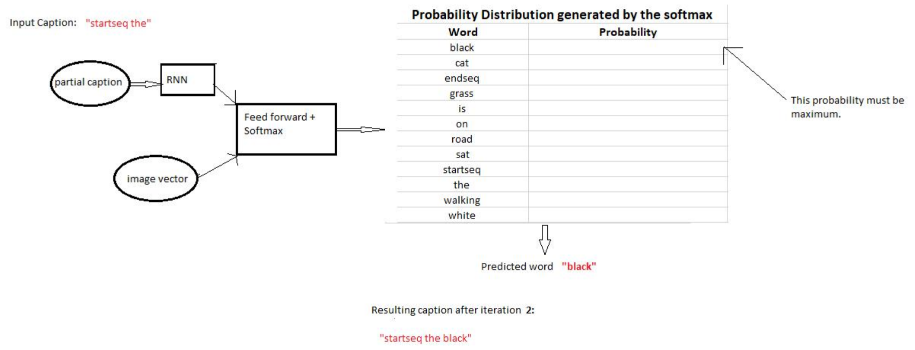
Probability Distribution generated by the softmax

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

That is regarded in terms of **Maximum Likelihood Estimation,** where model pick phrase which is most common acc. to the architecture for the given i/p.

This kind of framing of phrase with usage of iteration is regarded as **Greedy Search**, as we greedily choose the phrase with most probability.

Itr 2: **I/p :** pic_vector + "startseq the" **Expc. O/p :** "black"



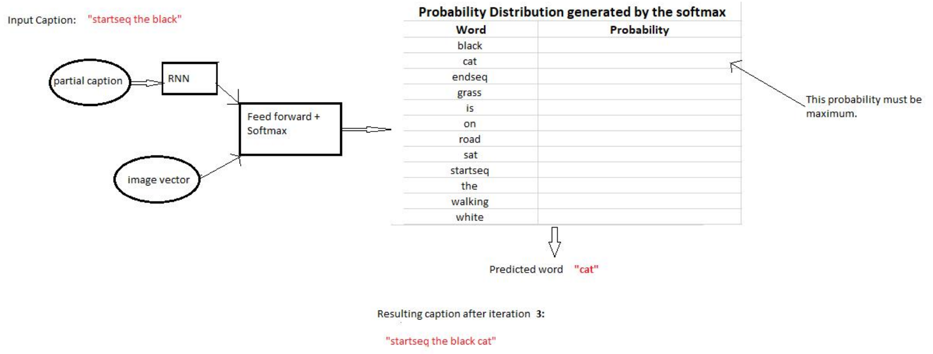
Probability Distribution generated by the softmax

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

Resulting caption after iteration 2:

"startseq the black"

Itr 3: **I/p :** pic_vector + "startseq the black" **Expc. O/p :** "cat"



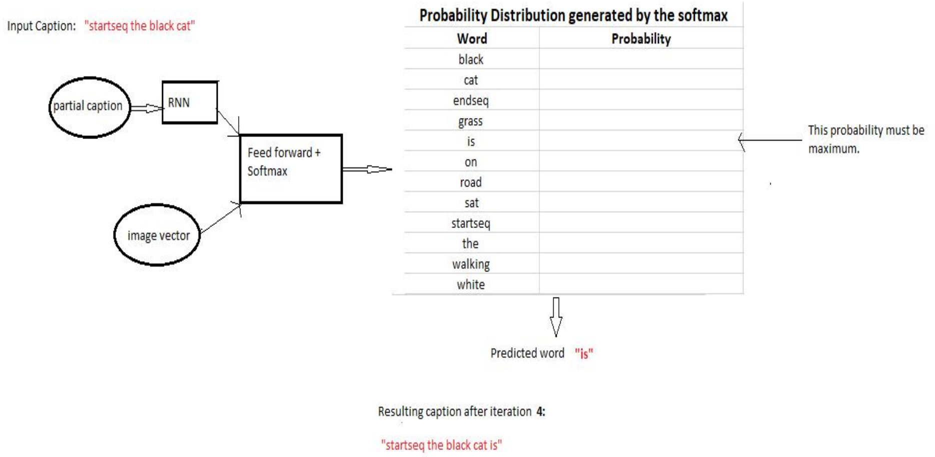
Probability Distribution generated by the softmax

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

Resulting caption after iteration 3:

"startseq the black cat"

Itr 4: **I/p :** pic vector + "startseq the black cat" **Expc. O/p :** "is"


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

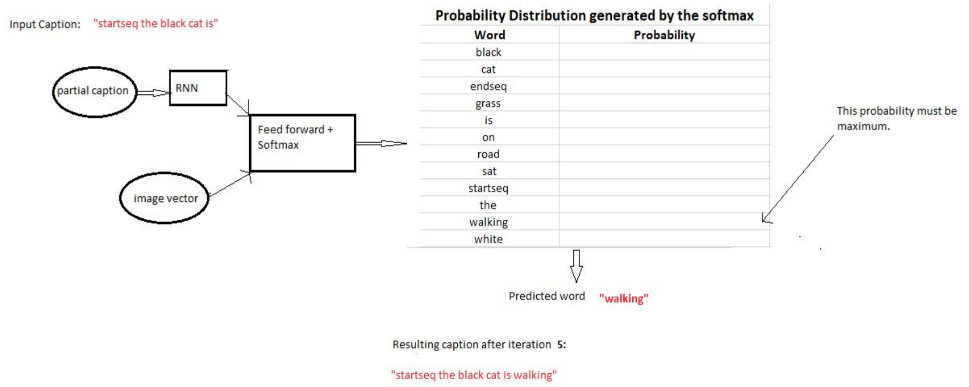
Probability Distribution generated by the softmax

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

Resulting caption after iteration 4:

"startseq the black cat is"

Itr 5: **I/p :** pic_vector +“startseq the black cat is” **Expc. O/p :** “walking”


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

Probability Distribution generated by the softmax

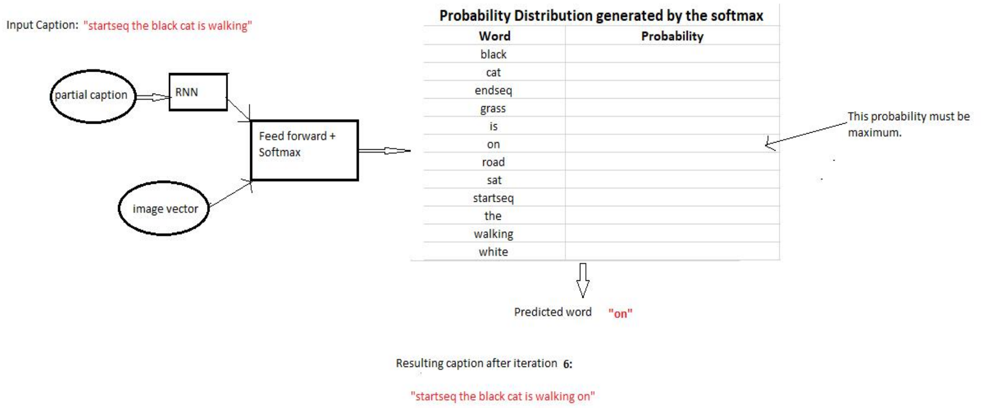
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Predicted word "walking"

Resulting caption after iteration 5:

"startseq the black cat is walking"

Itr 6: **I/p :** pic_vector+“startseq the black cat is walking” **Expc. O/p :** “on”


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

Probability Distribution generated by the softmax

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

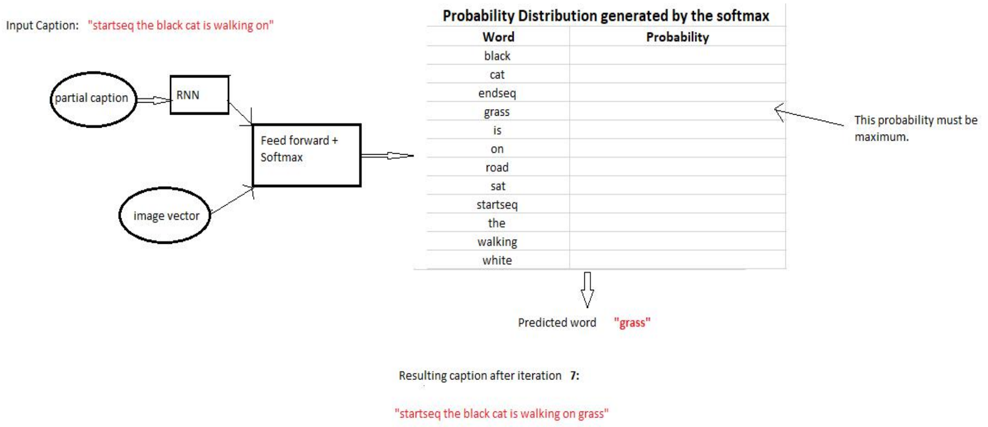
Predicted word "on"

Resulting caption after iteration 6:

"startseq the black cat is walking on"

Itr 7: **I/p :** pic_vector+“startseq the black cat is walking on”

Expc. O/p : “grass”


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

Probability Distribution generated by the softmax

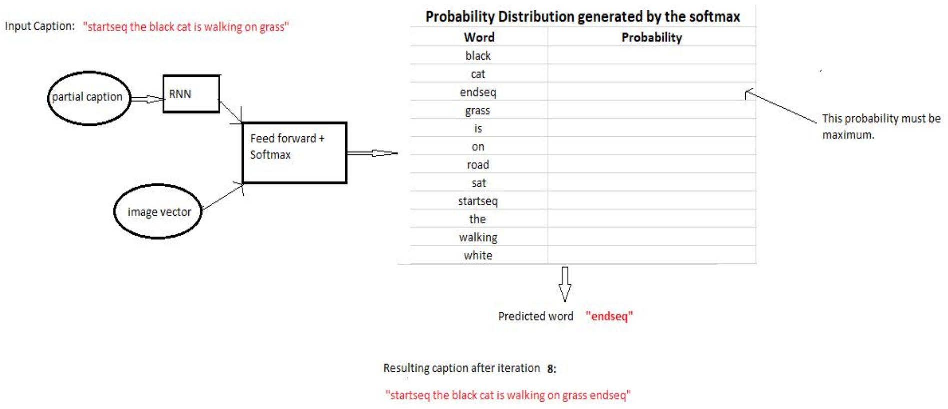
Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Predicted word "grass"

Resulting caption after iteration 7:

"startseq the black cat is walking on grass"

Itr 8: **I/p :** pic vector+“startseq the black cat is walking on grass” **Expc. O/p :** “endseq”


Input Caption: "startseq the black cat is walking on grass"
partial caption
RNN
Feed forward + Softmax
image vector
Probability Distribution generated by the softmax
Word
Probability
black
cat
endseq
grass
is
on
road
sat
startseq
the
walking
white
This probability must be maximum.
Predicted word "endseq"
Resulting caption after iteration 8:
"startseq the black cat is walking on grass endseq"

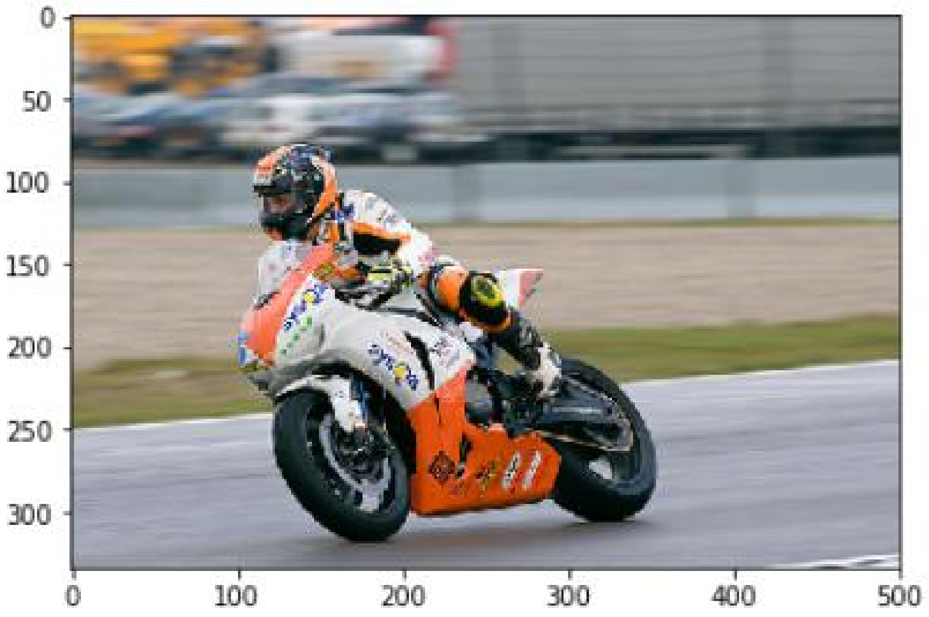
Here iterating **stops** when either of 2 conditions met:

We come across an ‘**endseq**’ token because of this ending of description is learnt by model. (You must now recognize the significance of the ‘endseq’ word) or We attain the max range of phrases formed by the model.

If any of the above written state is met, we cut the iteration & send the formed description as o/p of model for the pic under consideration.

2.2.12 Evaluations

To apprehend how precise the model is, let’s attempt to frame descriptions on pics from validation file (i.e. the pics which the model not looked over throughout the training)



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

Fig 2.2.12.1 o/p — 1

We must recognize how the model is aware of color and identified it precisely.

2.2.13 Web Application

We already have developed a model that can precisely predict the caption for the pics not too different from those part of the used flickr8k-dataset. And now time has come to take this model to work at backend so as to create a Web Application

We need to get this model web-ready so that anyone can upload his or her pic & get a desired computerized caption for the pic. We will use HTML/CSS, Bootstrap, & flask & convert this model into Web-API.

What is Model Deployment?

In a typical machine learning & deep learning project, we tend to begin by shaping the problem statement followed by data collection, preparation & pre-processing, understanding the data & model building, right? But, in the end, we wish our model to be accessible for the end-users such that it can help others too. Deployment of model is last stage of almost all machine learning project & are often little tricky. How will you get your project/machine learning model to your stakeholder? What are various things you need to watch out when sending your model into production? & how are you even able to begin to deploy such model?

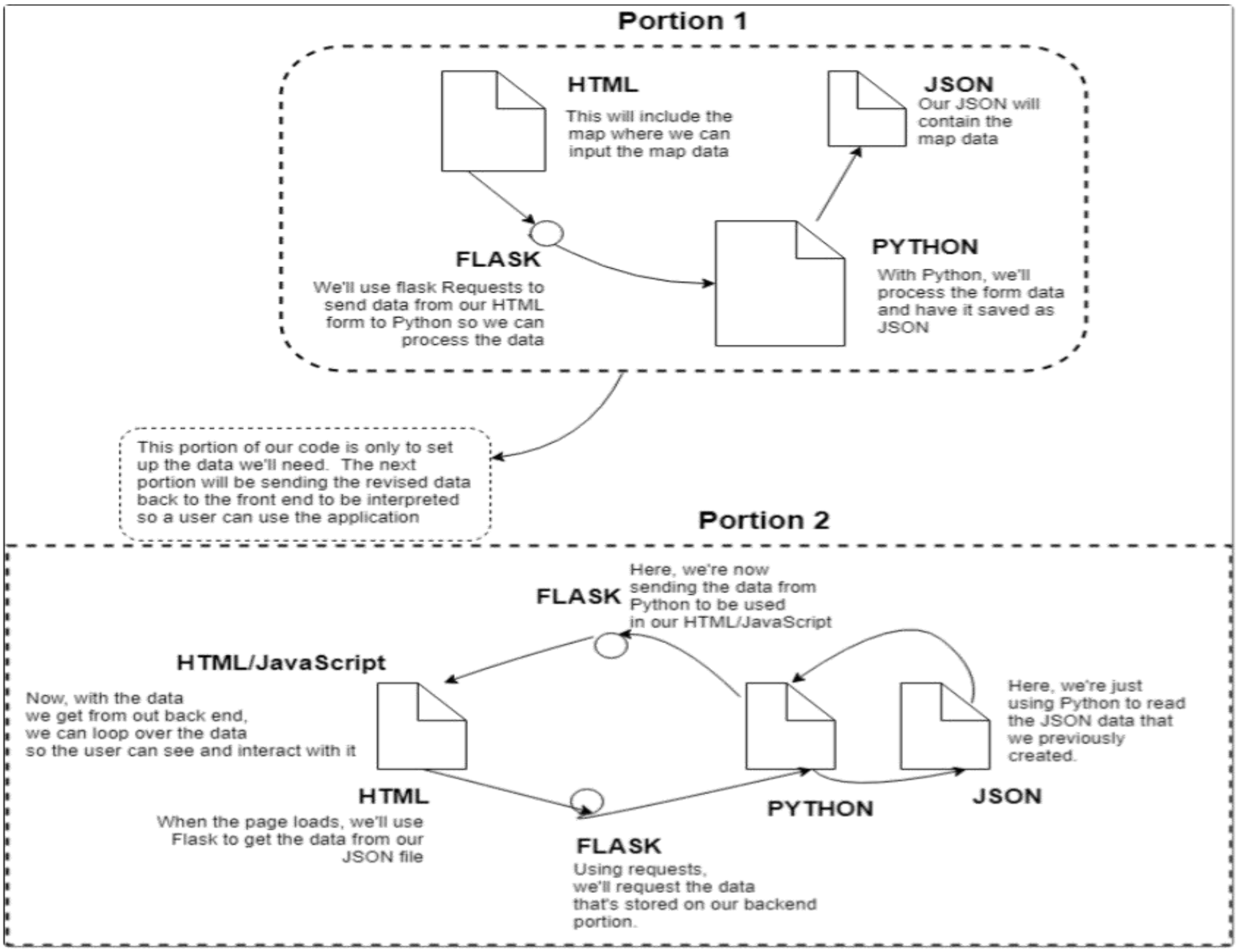

Portion 1
HTML
This will include the map where we can input the map data
JSON
Our JSON will contain the map data
FLASK
We'll use flask Requests to send data from our HTML form to Python so we can process the data
PYTHON
With Python, we'll process the form data and have it saved as JSON
This portion of our code is only to set up the data we'll need. The next portion will be sending the revised data back to the front end to be interpreted so a user can use the application
Portion 2
FLASK
Here, we're now sending the data from Python to be used in our HTML/JavaScript
HTML/JavaScript
Now, with the data we get from out back end, we can loop over the data so the user can see and interact with it
Here, we're just using Python to read the JSON data that we previously created.
HTML
When the page loads, we'll use Flask to get the data from our JSON file
PYTHON
JSON
FLASK
Using requests, we'll request the data that's stored on our backend portion.

Fig 2.2.13.1 Blueprint of Model Deployment using Flask Framework

What is Flask?

Flask is an web application framework scripted in Python. It has various modules which help to write applications and make it lot easier for a web developer without stressing about the various details like thread management, etc.

Flask offer us number/range of alternatives for web application development & it also offer us the vital and necessary tools & libraries which permit us to construct a web application better and faster.

Flask is web app framework, while at same time Flask is a datatype of python class.

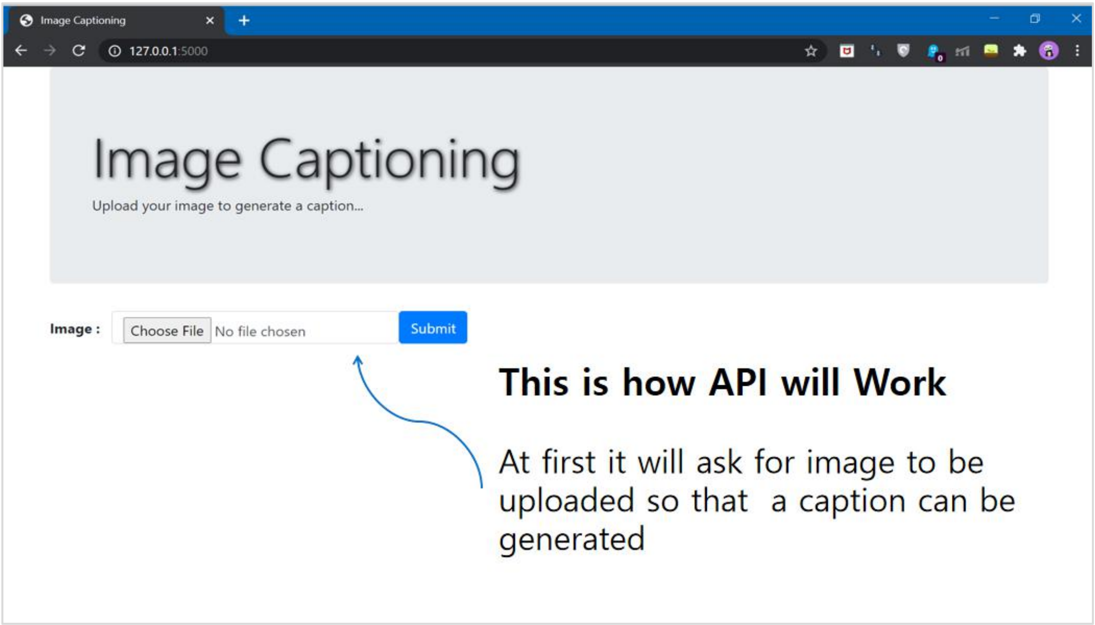
In other words, Flask is the prototype used for instantiating web applications if you want to put it simply.

2.3 CONCLUSION

Since i/p contain 2 parts, a pic feature vector & a partial description, we can't utilize the Appl. Program Interface -[Sequential] present because of Keras. This is the reason, we used API. Program interface -[Functional] that permit us for making Merge Models on the backend & interface. After all this, we begin checking out the data.

2.4 FINAL INTERFACE OF PROJECT

This is where the web page asks about the picture to be selected from memory

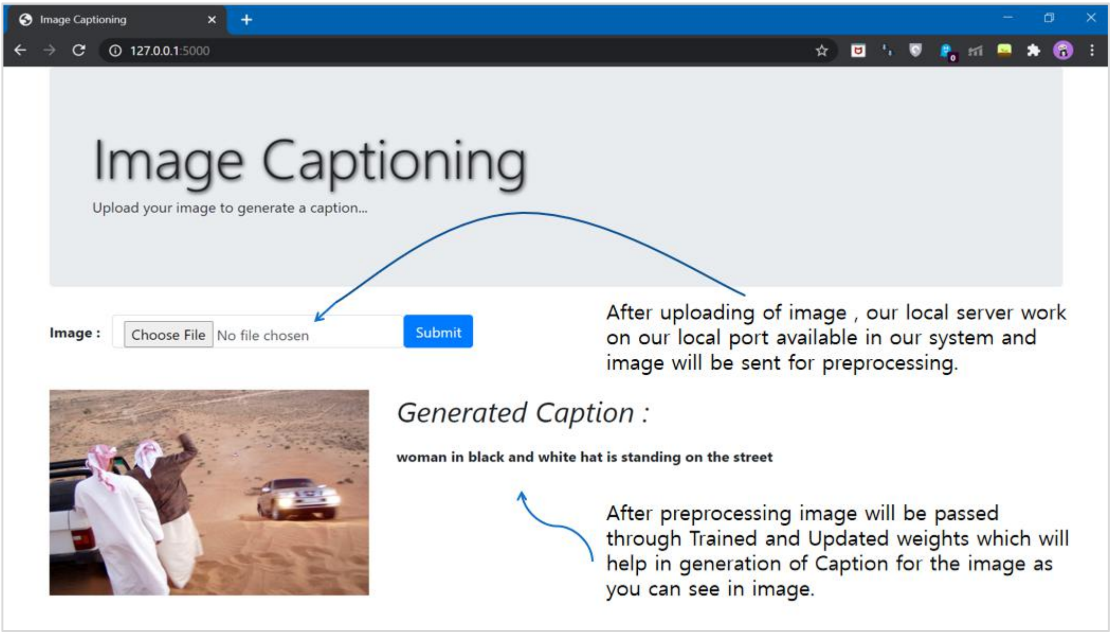


This is how API will Work

At first it will ask for image to be uploaded so that a caption can be generated

Fig 2.4.1 : Frontend Interface

ML-Model updated weights will work at the back-end & This is how caption will be generated for an i/p pic



After uploading of image , our local server work on our local port available in our system and image will be sent for preprocessing.

Generated Caption :

woman in black and white hat is standing on the street

After preprocessing image will be passed through Trained and Updated weights which will help in generation of Caption for the image as you can see in image.

Fig 2.4.2 : o/p interface

CHAPTER-3
CONCEPTS, TECHNOLOGIES & CODE USED

3.1 INTRODUCTION

The ideas, concepts & technologies used to give an understanding of the R & D of the system. It provides insight into the sort of technologies used & how they are matching with deployed concepts.

3.2 PROGRAMMING LANGUAGE & LIBRARIES USED

Python is high-level & integrated programming language. Made by Guido van Rossum & released in 1991 initially, Design & philosophy of python encourages code readability. It has advanced modules & integrated structures for data, blended with dynamic binding & dynamic typing, which turn it into terribly great for Application transformation.

We used many libraries written for machine learning in Python:

- Pandas - This library helps us to open dataset files and folder paths
- Numpy - This is the best library for various maths functions.
- Matplotlib -This library is used to plot Graphs.
- Keras - This is Tensorflow integrated library & helps in implementing NN.
- Nltk - Natural language Toolkit, used to tokenize the strings, etc
- Json - This is used here since we worked with Python dictionary & this is achieved by parsing strings using the json.loads () method.
- Time(all-time function), Regex(nltk support), String(work with sentence)
- Pickle - Useful whenever you need persistence between user sessions

3.3 IMAGE CAPTIONING BOT DECISION-MAKING WORKFLOW

We enforced a deep recurrent design that automatically produces a brief caption of pics. Our model make of CNN, that was pretrained on ImageNet, to get pic features. We tend to feed these features to a LSTM network to generate a caption of pic in simple English.

This is feasible because of extraction of features using CNN. After extraction of features, we use embedding matrix LSTM & glove vector.

[2] Self-critical Sequence Training for pic Captioning

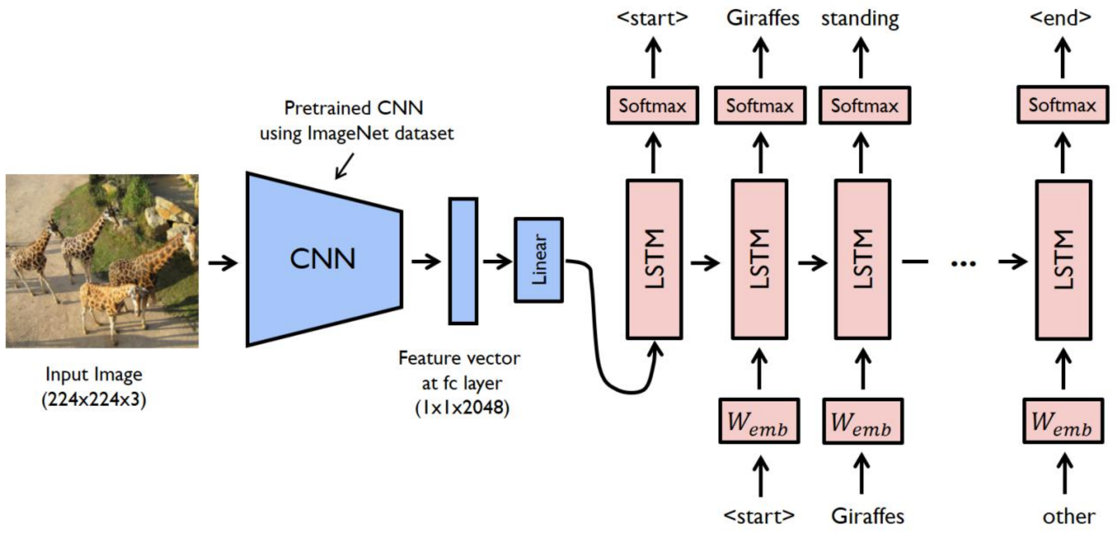


Fig 3.1 : Full Workflow Diagram for pic captioning Bot

Then we will use Google's Pre Trained model i.e. Resnet-50 (picnet).

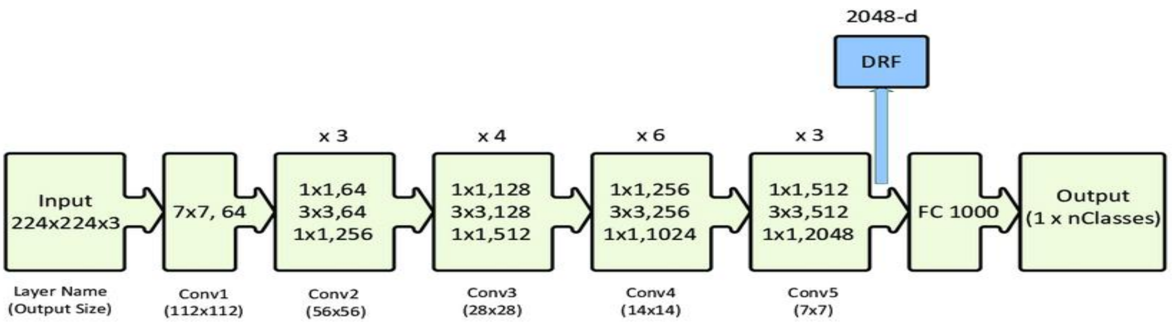


Fig 3.2 : Resnet 50 model explained layer by layer

3.4 HARDWARE & SOFTWARE REQUIREMENTS

3.4.1 HARDWARE REQUIREMENTS

i. Processor >=2.7GHz

ii. RAM >= 8GB

iii. Hard Disk >=20GB free space

iv. Graphics Card Nvidia 4GB dedicated

3.4.2 SOFTWARE REQUIREMENTS

i. Operating system - Microsoft Windows (8 or higher)

ii. Python 3.7.3 (64 bit)

iii. Jupyter Notebook

iv. Google Colab

3.5 CONCLUSION

Hence, most of the prominent technologies used & concepts deployed are in-depth explained here. Their background is provided.

CHAPTER-4
SYSTEM TESTING

4.1 INTRODUCTION

System testing suggests testing the entire system. All components/modules are integrated to check if the system is properly working or not. System testing occurs after integration testing. This plays vital role in giving a high-quality product and services.

ML application development needs in depth & extensive testing. To get elaborated info regarding many errors, some tests were done like test environment - done on Jupyter notebook running on Windows OS with white background for gesture recognition.

System Testing is a technique implemented to assess the entire compliance of the system also against certain specific requisites. In such testing, the diverse services of the machine and processes are examined.

System Testing is managed by a crew which isn't associated to improvement crew, to evaluate the unbiased calibre of the machine. It includes functional & non functional validation also.

Usually, software modules are simplified elements of bigger computerized systems. The software applications are interfaced with different software or hardware systems. This testing can include a range of checks whose main motive is to check and practice the full system based on computer.

24.2 TESTING REPORTS

4.2.1 UNIT TESTING is a sort of software program checking, where units of the software program undergo testing. Here motive is to check that every part of software program module plays as we want. Unit Testing is performed at the time of coding phase by the developers of that appl. Unit Tests separate a few components of code & check their correctness. A unit can be a single function, character module or methodology, process, or component.

TYPE OF TESTING	WILL TEST BE PERFORMED ?	COMMENTS /EXPLANATION	SOFTWARE COMPONENTS
Requirement	Yes	This testing is required because we need to verify Requirements Whether our selected Complete attributes can solve the current problem or not.	Complete
Unit	Yes	This testing allows us to whether individual's attributes affect the Application performance or not	Individual Attributes
Backend	Yes	Model working fine having some orthodox	Complete
Frontend	Yes	Integration of Web Api on Local Server - Cloud Ready	Complete

Table 4.2.1 Unit Testing

4.2.2 INTEGRATION TESTING is software testing where many single component or unit are tested as a group.

Component integration testing: It is a form of testing, which is done to find lackings/errors in the interfaces & interaction btw the unified components.

System integration testing: It is a form of testing in which the packages & system unification, interfaces to outside organizations (e.g. Electronic Data Interchange) are tested.

S. No.	**i/p**	**Expected o/p**	**Status**
1	Code Running in CMD as Administrator	Installed	Success
2	Code Launch	Launched	Success
3	CNN (Working of all filters)	Validates	Success
4	RNN (Word Embedding)	Validates	Success
5	Transfer Learning (Resnet 50)	Validates	Success
6	Data Cleaning & Data Preprocessing	Validates	Success
7	Testing the Dataset	Validates	Success
8	Training & Validation	Needs third party platform I.e. Google Colab	Success
9	Interface Web - API	Local Server	Success

Table 4.2.2 Integration Testing

4.2.3 PERFORMANCE TESTING is test of stability of network, computation, software-program or device under a work-load & determining its speed & responsiveness.

Performance testing arise in the production environment in limited cases. It can involve quantitative tests done in a lab & typical parameters include data transfer rate, network bandwidth, processing speed, & throughput, workload efficiency, & reliability.

Test Case Id	**Test Name**	**o/p**
1	Initialization Time	25-50 sec. avg.
2	Processing Time	30-41 sec. avg.
3	Captioning the pic	10-20 sec. avg.
4	Initialization of Transfer Learning Model	200-500 sec. avg.
5	Training & Validation	10-30 min. avg.
6	Web Page Data Loading & Retrieving	3-5 sec avg.

Table 4.2.3 Performance Testing

4.3 CONCLUSION

Hence, a sufficient amount of observational testing is done. Software testing is examined. Concepts functioning is well noticed here.

CHAPTER-5

FUTURE SCOPE, SWOT ANALYSIS & CONCLUSION

5.1 INTRODUCTION

Here, how the concepts used & technologies used in this system can lead to further possibilities is touched upon. An analysis from the future's point of view is provided.

Through SWOT Analysis, present as well as future concerns are touched upon.

5.2 FUTURE SCOPE

- Using a dataset which is larger.

- Hyperparameter tuning is also a good option (no. of layers, dropout rate, learning rate, group size, no. of units, group normalization, etc.).

- To overcome Overfitting, use the cross-validation set

- To measure model QOS. If needed, Use BLEU Score

- API'fy this model using **FLASK** and deploy it in **AWS or Google Cloud.**

5.3 SWOT ANALYSIS

5.3.1 STRENGTHS

The strength of the method is in its end-to-end learning framework.
Also Model can generate sentences with correct preposition & conjunctions

5.3.2 WEAKNESSES

Training in Data requires High Computation which is challenging & can be a very expensive affair.

5.3.3 OPPORTUNITIES

This project opens various opportunities like Automated Vehicles, Robotic emotion Development.

5.3.4 THREATS

The limitations in achieving accuracy might pose a major threat but it can be covered by using new & good pretrained models

5.4 CONCLUSION

The system's working as explained shows a genuine possibility of plugging the lapses in the present problem at apparatus at various levels. The systems ideation & implementation are such that the model is quite reliable & solving many problems.

REFERENCES

[1] A Comprehensive Survey of Deep Learning for pic Captioning by Md. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, & Hamid Laga. 2018. ACM Comput. Surv. 51, 6, Article 118 (February 2019), 36 pages. https://doi.org/10.1145/3295748

[2] Self-critical Sequence Training for pic Captioning by Steven J. Rennie, Etienne Marcheret1 , Youssef Mroueh, Jerret Ross & Vaibhava Goel1 Watson Multimodal Algorithms & Engines Group IBM T.J. Watson Research Center, NY, USA {sjrennie, mroueh, rossja}@us.ibm.com, {etiennemarcheret, vaibhavagoel}@gmail.com https://openaccess.thecvf.com/content_cvpr_2017/papers/Rennie_Self-Critical_Sequence_Training_CVPR_2017_paper.pdf

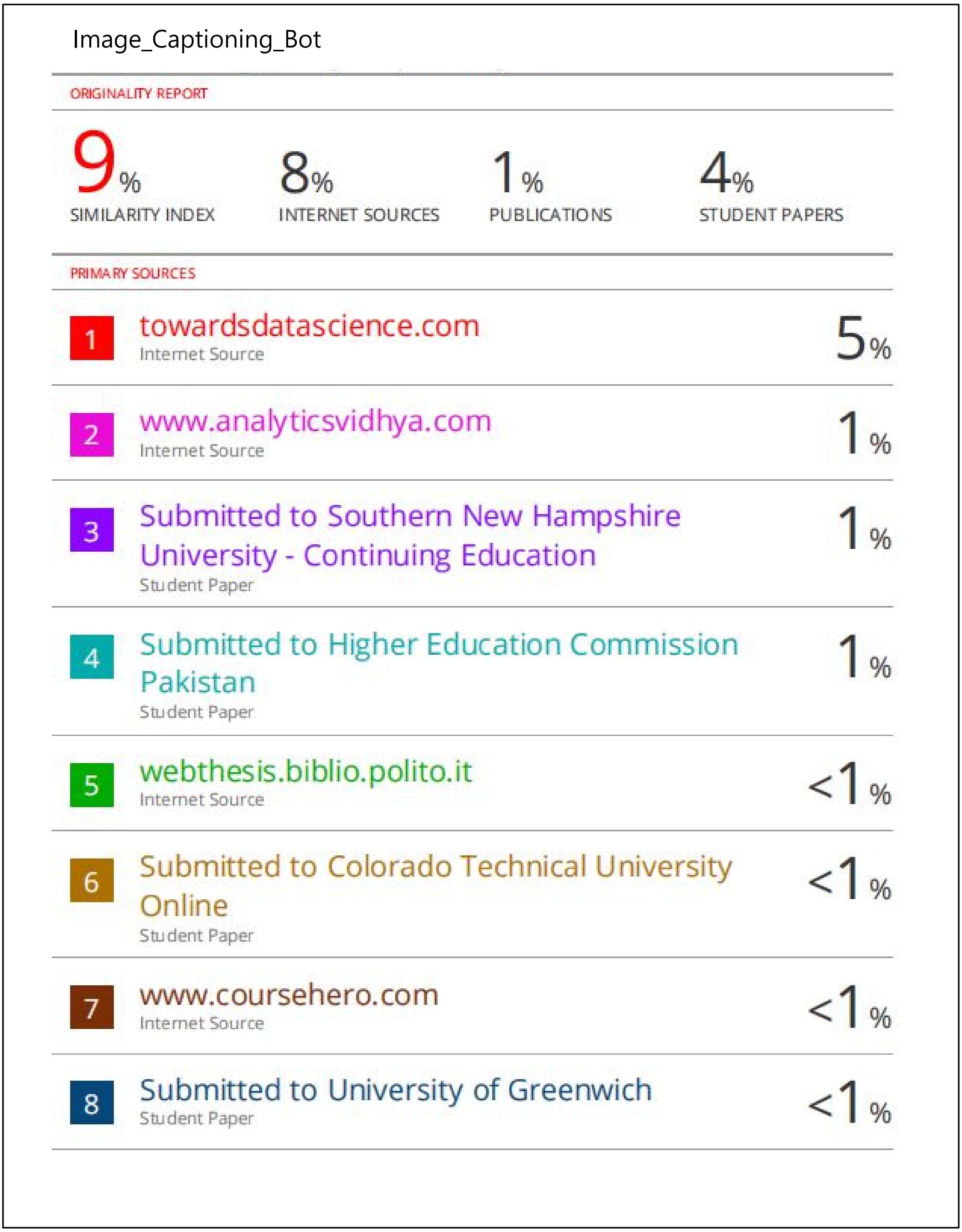
[3] Deep Reinforcement Learning-based pic Captioning with Embedding Reward Zhou Ren1 Xiaoyu Wang1 Ning Zhang1 Xutao Lv1 Li-Jia Li2 * 1Snap Inc. 2Google Inc. {zhou.ren, xiaoyu.wang, ning.zhang, xutao.lv}@snap.com lijiali@cs.stanford.edu https://openaccess.thecvf.com/content_cvpr_2017/papers/Ren_Deep_Reinforcement_Learning-Based_CVPR_2017_paper.pdf

[4] Guided Open Vocabulary pic Captioning with Constrained Beam Search Peter Anderson1 , Basura Fernando1 , Mark Johnson2 , Stephen Gould1 1The Australian National University, Canberra, Australia firstname.lastname@anu.edu.au 2Macquarie University, Sydney, Australia mark.johnson@mq.edu.au
https://arxiv.org/pdf/1612.00576.pdf

[5] Medium.com - [Article] pic-captioning-with-keras by Hrshall Lamba 2019

[6] https://www.analyticsvidhya.com/blog/2018/04/solving-an-pic-captioning-task-using-deep-learning/

PLAGIARISM TEST



Image_Captioning_Bot

ORIGINALITY REPORT

9%	8%	1%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	towardsdatascience.com Internet Source	5%
2	www.analyticsvidhya.com Internet Source	1%
3	Submitted to Southern New Hampshire University - Continuing Education Student Paper	1%
4	Submitted to Higher Education Commission Pakistan Student Paper	1%
5	webthesis.biblio.polito.it Internet Source	<1%
6	Submitted to Colorado Technical University Online Student Paper	<1%
7	www.coursehero.com Internet Source	<1%
8	Submitted to University of Greenwich Student Paper	<1%