**Automatic Image Captioning**

**A Project Report of Capstone Project**

**Submitted by**

**Group 4**

Chiru Toleti (2201497)

Satyam Kumar (2200718)

K Praveen Kumar (2201598)

Yaswanth Kumar P (2201896)



# ABSTRACT

Automatically describing the content of an image is a fundamental problem in artificial intelligence. Image caption generator is a process which involves natural language processing and computer vision concepts to recognize the context of an image and present it in English. In this current project work, encoder-decoder architecture is used for image captioning. We have used the deep convolutional neural network (CNN) ResNet50 as the encoder and the deep recurrent neural network (RNN) LSTM as the decoder to generate a meaningful caption of the image. To improve the accuracy we have considered soft attention layer. To further improve the accuracies transformer architecture has been implemented. The project is implemented in python on Google Colab notebook. The libraries used for current project are tensor-Flow and Keras.

The model is trained to maximize the likelihood of the target description sentence given the training image. The accuracy of the model and the fluency of the language it learns solely from image descriptions. Our model is trained with Flicker-8k dataset, and tested over the test set. The model has produced the BLEU-1 score (the higher the better) on the Flicker-8k dataset is 0.34; we also show BLEU-1 score improves with Attention layer (0.4545454545) and with transformer (0.4444444444).

# ACKNOWLEDGEMENTS

We are thankful to all the people who joined as part of making this journey of fulfilling this project into working model. We are grateful to IIIT Hyderabad and Talentsprint for giving us a wonderful platform through “Cohort 18” for learning “Artificial Intelligence and Machine Learning” and exploring our AIML skills during development of this project.

We also express our heartfelt gratitude to Prof Vineet, IIIT Hyderabad for rendering all possible help and support during the development, implementation and review of the project.

We are also thankful to our mentor Mr. Nayan Anand, Talentsprint who has been extending continuous support and mentoring with constructive and helpful inputs, all through the project work. We also thank all the staff of Talentsprint for their co-operation and support.

We are also thankful to other teams of Cohort 18 who helped us directly or indirectly in solving problems and making our model more efficient and working.

Table of Contents

[ABSTRACT 1](#_Toc113739881)

[ACKNOWLEDGEMENTS 2](#_Toc113739882)

[1. Introduction 5](#_Toc113739883)

[*1.1.* *Problem Statement* 5](#_Toc113739884)

[*1.2.* *Literature Survey* 5](#_Toc113739885)

[*1.3.* *Image captioning Methodologies* 6](#_Toc113739886)

[1.3.1 Template-based image captioning: 6](#_Toc113739887)

[1.3.2 Retrieval-based image captioning: 6](#_Toc113739888)

[1.3.3 Novel image captioning: 6](#_Toc113739889)

[2. Automatic Image Captioning based on ResNet50 and LSTM with software attention 7](#_Toc113739890)

[2.1 Image Feature Extraction Model: 8](#_Toc113739891)

[2.2 The Language Model: 8](#_Toc113739892)

[2.3 Attention Mechanism: 8](#_Toc113739893)

[2.4 DataSets used: 9](#_Toc113739894)

[2.5 Feature extraction and language model challenges 9](#_Toc113739895)

[3. Model Architecture 11](#_Toc113739896)

[4. Project Workflow 12](#_Toc113739897)

[4.1 Making the vocabulary from all files 12](#_Toc113739898)

[load\_doc( filename ) 12](#_Toc113739899)

[load\_descriptions ( doc 12](#_Toc113739900)

[cleaning\_text( descriptions) 12](#_Toc113739901)

[to\_vocabulary( descriptions ) 12](#_Toc113739902)

[save\_descriptions( descriptions, filename ) 12](#_Toc113739903)

[4.2 Extracting the feature vector from all images: 12](#_Toc113739904)

[4.3 Loading Dataset for training the model 13](#_Toc113739905)

[load\_photos( filename ) 13](#_Toc113739906)

[load\_clean\_descriptions( filename, photos ) 13](#_Toc113739907)

[load\_features(photos) 13](#_Toc113739908)

[4.4 Tokenizing the vocabulary 13](#_Toc113739909)

[4.5 Create Data generator 13](#_Toc113739910)

[4.6 Defining the CNN-RNN model 14](#_Toc113739911)

[Feature Extractor – 14](#_Toc113739912)

[Sequence Processor – 14](#_Toc113739913)

[Decoder – 14](#_Toc113739914)

[4.7 Generalized Colab Notebook for various models and datasets 14](#_Toc113739915)

[4.8 Training the model 15](#_Toc113739916)

[4.9 Testing the model 15](#_Toc113739917)

[5. Results 16](#_Toc113739918)

[6. Conclusion 17](#_Toc113739919)

[7. References 18](#_Toc113739920)

# Introduction

With the rapid development of digitization, there is a huge repository of images, accompanied with a lot of related texts. Although it is an easy task for a human to describe an image, it becomes very difficult for a machine to perform such a task. Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing.

Image captioning has various applications such as recommendations in editing applications, usage in virtual assistants, image indexing, assisting visually impaired persons, social media, logo identification with the help of AI, Image tagging for Ecommerce, photo-sharing service and online catalogue and several other natural language processing applications.

`

## *Problem Statement*

The objective of automatic image captioning is to generate properly formed English sentences to describe the content of an image automatically, which is of great impact in various domains.

## *Literature Survey*

Automatic image captioning has recently attracted much research interest. One common framework used in this category is the encoder-decoder framework for image captioning. Recent research works have used the deep convolutional neural network (CNN) as the encoder and the deep recurrent neural network (RNN) as the decoder. However, it still remains challenging to identify the proper CNN and RNN models for the image captioning. In this project work, we investigate one single-joint mode, AICRL, for automatic image generation using ResNet50 (a convolutional neural network) and LSTM (long short-term memory) with soft attention mechanism. AICRL consists of

1. An encoder and a decoder. We adopt ResNet50 as the encoder to create an extensive representation of an input image by embedding it into a vector.
2. The LSTM with a soft attention as the decoder which selectively focuses the attention over a certain part of an image to predict the next sentences.

Furthermore, we conduct extensive experiments and empirically determine the structure of the model and fine-tuned the model hyper parameters. Our experimental evaluation indicates that AICRL is effective to generate proper captions for the image.

## *Image captioning Methodologies*

Much research effort has been devoted to automatic image captioning, and it can be categorised into

1. Template based image captioning
2. Retrieval-based image captioning and
3. Novel image caption generation

### Template-based image captioning:

First detects the objects/attributes/actions and then fills the blanks slots in a fixed template

#### Pros:

The template-based approach is aimed at generating captions by using fixed templates with a number of blank slots, in which way different objects, attributes, and actions are detected first and then the blank spaces in the templates are filled. Template-based methods can generate grammatically correct captions.

#### Cons:

However, templates are predefined and length of captions cannot be variable.

### Retrieval-based image captioning:

Approaches first find the visually similar images with their captions from the training dataset, and then the image caption is selected from similar images with captions.

#### Pros:

The retrieval-based approach tries to generate description for an image by selecting the most semantically similar sentences from the sentence pool or directly copying sentences from other visually similar images.

### Novel image captioning:

Generation approaches are to analyse the visual content of the image and then to generate image captions from the visual content using a **language model**.

Compared to the first two categories, novel caption generation can generate new captions for a given image that are semantically more accurate than previous approaches. Most of the works in this category rely on machine learning and deep learning.

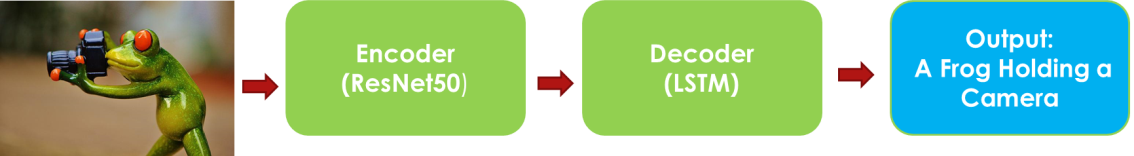
In the current work we have used the approach of Automatic Image Captioning based on ResNet50 and LSTM with software attention, and further improved the model with transformers.

# Automatic Image Captioning based on ResNet50 and LSTM with software attention

The image should be converted to suitable features so that they can be trained into a deep learning model. Feature extraction is a mandatory step to train any image in deep learning model. The features are extracted using Convolutional Neural Network (CNN). The language model used to train and generate the sequence of words is Recurrent Neural Network (RNN).

The Automatic Image Captioning based on ResNet50 and LSTM with software attention (AICRL) has to generate the proper description for the given images. To do so, the AICRL model is designed with an encoder and decoder architecture based on CNN and RNN. In particular, to extract visual features, we use the ResNet50 network as the encoder to generate a one-dimensional vector representation of the input images.

The choice of language model is determined by its ability to cope with vanishing problems and exploding gradients, which are the most common problems in the design and training of RNN. LSTM networks are successfully used to accomplish the tasks of machine translation and sequence generation. In our design, we adopt LSTM as our language model to generate proper caption based on the input vector from the ResNet50 output.



Further, we utilise the soft attention in the decoder to enable the model to selectively focus the attention over a certain part of an image to predict the next sentence better. We conduct extensive experiments, empirically determine the structure of the model, and fine-tune the model hyper-parameters. The whole model is fully trainable by using a stochastic gradient descent.

## Image Feature Extraction Model:

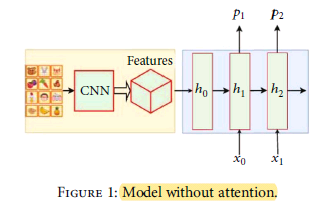
To represent the image, we adopt the convolutional neural network (CNN), ResNet50.

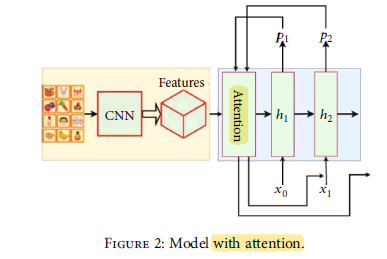
## The Language Model:

In our design, we adopt LSTM as our language model to generate proper caption based on the input vector from the ResNet50 output.

## Attention Mechanism:

Soft attention is implemented by adding an additional input of attention gate into LSTM that helps to concentrate selective attention. The main drawback of the model without attention is that it tries to decode the full image from the last hidden layer of h0 in Figure 1. It is like an analogy with machine translation in the whole process. To do a translation of the whole text is just from the “last word.” So it will lose a lot of useful information from the beginning of the text.





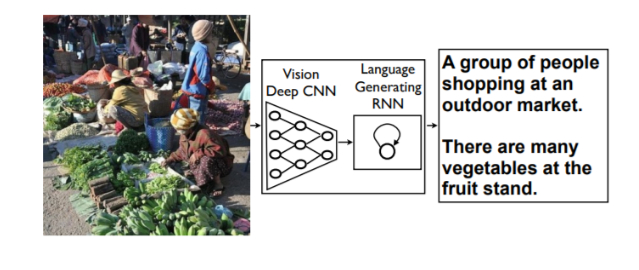
## DataSets used:

Dataset of images with their five descriptions for each image.

|  |  |  |
| --- | --- | --- |
| **Name of the dataset** | **Training Samples** | **Testing Samples** |
| Flickr8K | 7,000 | 1,000 |

## Feature extraction and language model challenges

CNNs can produce a rich representation of the input image by embedding it to a fixed-length vector, such that this representation can be used for a variety of vision tasks. Hence, it is natural to use a CNN as an image “encoder”, by first pre-training it for an image classification task and using the last hidden layer as an input to the RNN decoder that generates sentences. Which will be the neutral way of Image captioning.



The main drawbacks of the work are the quick model overfitting, so they use the heavy and expensive GoogLeNet. ResNet50, which is a very deep network that has 50 layers. The depth of the network is crucial for neural networks, but deeper networks are more difficult to train. The structure of ResNet50 facilitates the training of networks and allows them to be much deeper, which leads to increased performance in different tasks. ResNet50 is much deeper than their “simple” counterparts, but moreover, the number of parameters (weights) of such networks is much smaller.

Deep convolutional neural networks have led to a series of breakthroughs for image classification. Recent evidence reveals that network depth is of crucial importance. Many other nontrivial visual recognition tasks have also greatly benefited from the deep models. With the network depth increasing, the accuracy of networks increases rapidly, which is not surprising and then rapidly degrades (saturated). This degradation is not caused by overfitting, and the addition of even more layers leads to a higher learning error. In a sense, this is strange, since a deeper network has a strictly large representational power. It is possible for ResNet50 to get a deeper model trivially, which is not worse than the less deep network. It can be done by adding several identity layers, that is, levels that simply skip the signal further without changes. ResNet50’s deeper levels have to predict the difference between the output of the previous layers and the objective function. They could always drive the weights to 0 and simply skip the signal. Hence, deep residual learning is a good method that makes the network learn to predict deviations from past layers.

Image captioning intersects computer vision and natural language processing (NLP) research. NLP tasks, in general, can be formulated as a sequence to sequence learning. Several neural language models such as neural probabilistic language model, log-bilinear models, skip-gram models, and recurrent neural networks (RNNs) have been proposed for learning sequence to sequence tasks. RNNs have widely been used in various sequence learning tasks. However, traditional RNNs suffer from vanishing and exploding gradient problems and cannot adequately handle long-term temporal dependencies.

LSTM networks are a type of RNN that has special units in addition to standard units. LSTM units use a memory cell that can maintain information in memory for long periods of time. In recent years, LSTM based models have dominantly been used in sequence to sequence learning tasks. Another network, Gated Recurrent Unit (GRU) has a similar structure to LSTM but it does not use separate memory cells and uses fewer gates to control the flow of information.

However, LSTMs ignore the underlying hierarchical structure of a sentence. They also require significant storage due to long-term dependencies through a memory cell. In contrast, CNNs can learn the internal hierarchical structure of the sentences and they are faster in processing than LSTMs.

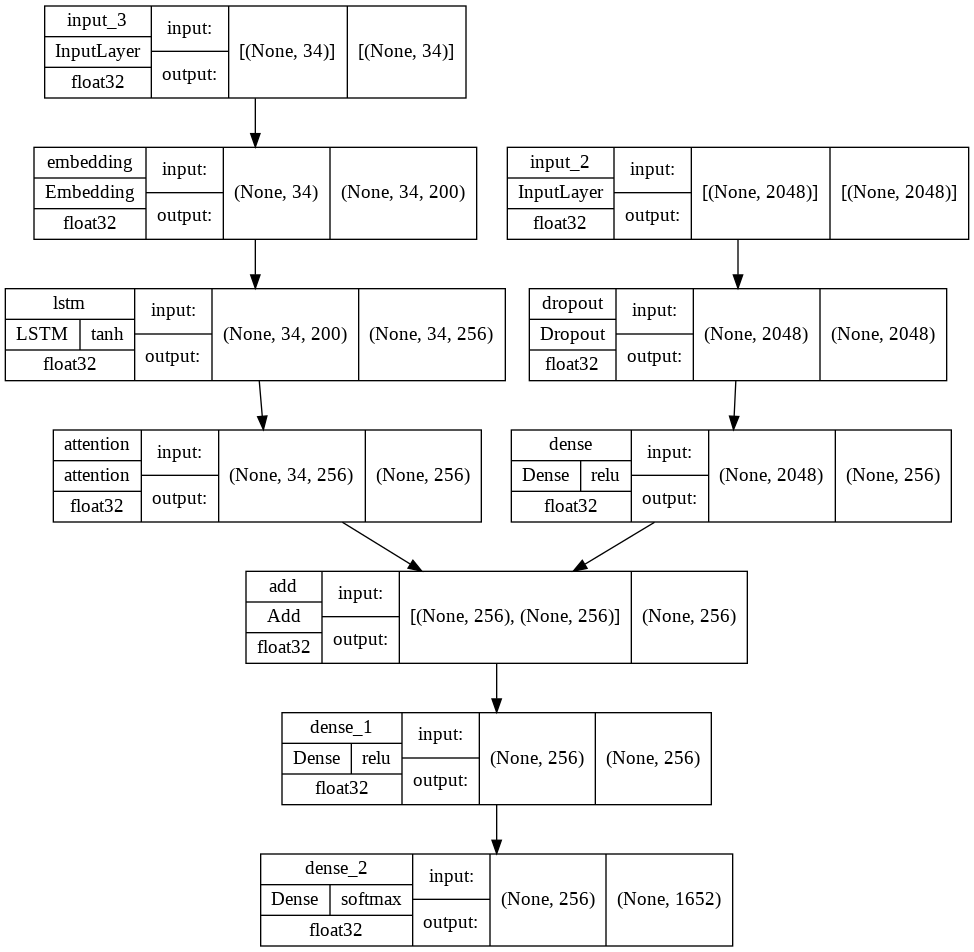
To better isolate the image content, we adopt the soft attention mechanism, which has been widely used to solve the problem of image classification, as there is no need to process all pixels of an image. For example, in the classification problem, the background usually plays an insignificant role. Nevertheless, convolutional neural networks, which are the most popular method for solving such a problem, spend the same amount of computational resources on all parts of the image.

# Model Architecture

**Encoder**

**Input: 224x224x3**

**Output: 2048x1**



# Project Workflow

The Automatic image caption generator model is completely implemented on Google Colab notebook. The total workflow of the cola b is presented below.

## Making the vocabulary from all files

load\_doc( filename ) – For loading the document file and reading the contents inside the file into a string.

load\_descriptions ( doc ) – This function will create a descriptions dictionary that maps images with a list of 5 captions. The descriptions dictionary will look something like the Figure.

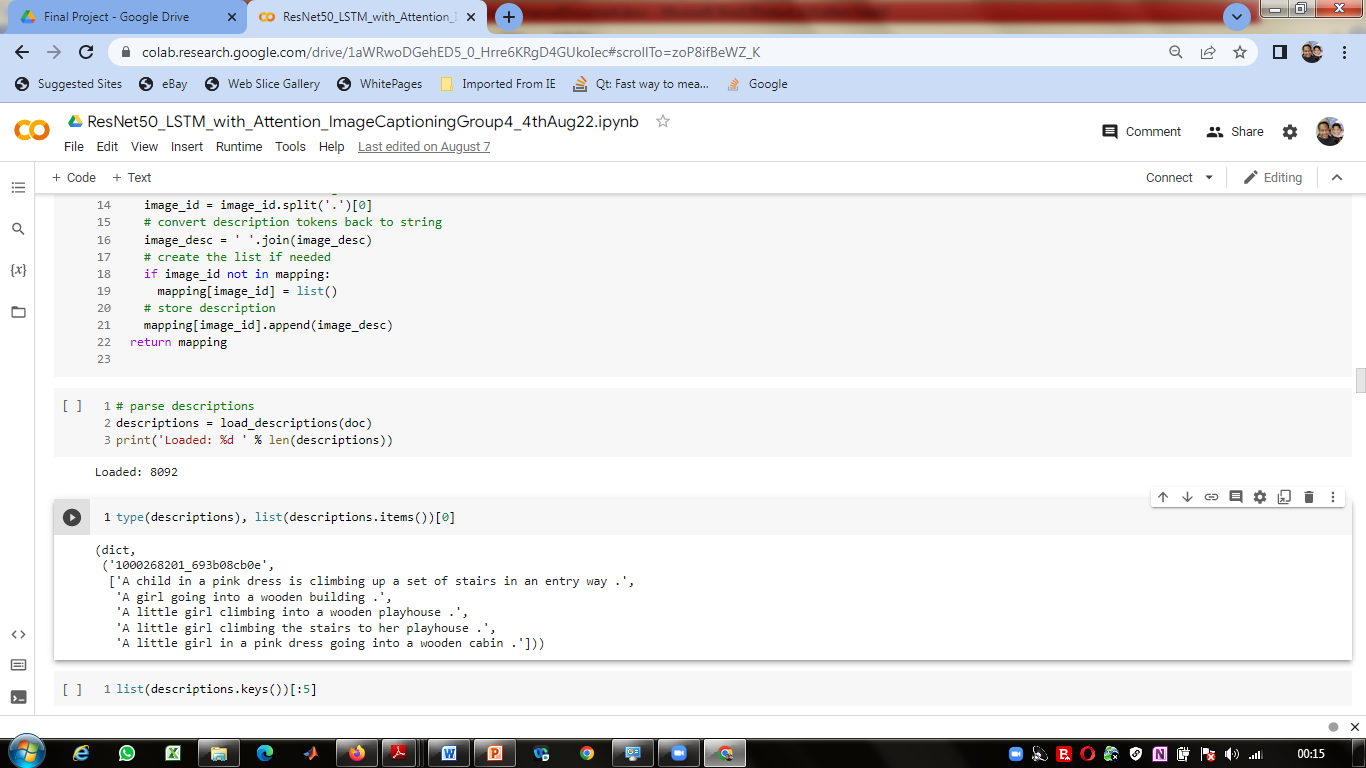


Figure.6.2. Flicker Dataset Python File

cleaning\_text( descriptions) – This function takes all descriptions and performs data cleaning. This is an important step when we work with textual data, according to our goal, we decide what type of cleaning we want to perform on the text. In our case, we will be removing punctuations, converting all text to lowercase and removing words that contain numbers.So, a caption like “A man riding on a three-wheeled wheelchair” will be transformed into “man riding on three wheeled wheelchair”

to\_vocabulary( descriptions ) – This is a simple function that will separate all the unique words and create the vocabulary from all the descriptions.

save\_descriptions( descriptions, filename ) – This function will create a list of all the descriptions that have been preprocessed and store them into a file. We will create a descriptions.txt file to store all the captions.

## **Extracting the feature vector from all images:**

This technique is also called transfer learning, we don‟t have to do everything on our own, we use the pre-trained model that have been already trained on large datasets and extract the features from these models and use them for our tasks. We are using the ResNet50 model which has been trained on imagenet dataset that had 1000 different classes to classify. We can directly import this model from the keras. Make sure you are connected to the internet as the weights get automatically downloaded. Since the ResNet50 model was originally built for imagenet, we will do little changes for integrating with our model. One thing to notice is that the ResNet50 model takes 224\*224\*3 image size as input. We will remove the last classification layer and get the 2048 feature vector. **model = ResNet50( include\_top=False, pooling=‟avg‟ ).**

The function extract\_features() will extract features for all images and we will map image names with their respective feature array. Then we will dump the features dictionary into a “**encoded\_train\_images.pkl**” pickle file. If you are using CPU then this process might take 1-2 hours. You can comment out the code and directly load the features from our pickle file.

## Loading Dataset for training the model

In our Flickr\_8k\_test folder, we have Flickr\_8k.trainImages.txt file that contains a list of 6000 image names that we will use for training. For loading the training dataset, we need more functions:

load\_photos( filename ) – This will load the text file in a string and will return the list of image names.

load\_clean\_descriptions( filename, photos ) – This function will create a dictionary that contains captions for each photo from the list of photos. We also append the <start> and <end> identifier for each caption. We need this so that our LSTM model can identify the starting and ending of the caption.

load\_features(photos) – This function will give us the dictionary for image names and their feature vector which we have previously extracted from the ReNet50 model.

## Tokenizing the vocabulary

Computers don‟t understand English words, for computers, we will have to represent them with numbers. So, we will map each word of the vocabulary with a unique index value. Keras library provides us with the tokenizer function that we will use to create tokens from our vocabulary and save them to a “tokenizer.p” pickle file. Our vocabulary contains 1652 words after removing duplications and having threshold value more than 10 . We calculate the maximum length of the descriptions. This is important for deciding the model structure parameters. Max\_length of description is 34.

## Create Data generator

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

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



Word Prediction Generation Step By Step

## **Defining the CNN-RNN model**

To define the structure of the model, we will be using the Keras Model from Functional API. It will consist of three major parts

Feature Extractor – The feature extracted from the image has a size of 2048, with a dense layer, we will reduce the dimensions to 256 nodes.

Sequence Processor – An embedding layer will handle the textual input, followed by the LSTM layer.

Decoder – By merging the output from the above two layers, we will process by the dense layer to make the final prediction. The final layer will contain the number of nodes equal to our vocabulary size. Visual representation of the final model is given in the figure.

## Generalized Colab Notebook for various models and datasets

An effort was made to generalize the Google Colab Notebook, to process the images, extract features from the images, training and testing the model , saving the model and generation of metrics. The flow control for selecting the Models, dataset, optimizer, loss-function, is controlled with various hyper-parameter. The list of hyper-parameters used to control the flow of Colab Note-book is given below.

* HYP\_DEBUG = True
* HYP\_DATASET = "Download"      # Drive Or Download  or SKIP
* HYP\_SOURCE\_DATA\_SET = "F8K" # "F30K" / "COCO123K"
* HYP\_ENCODER\_MODEL = "RESNET50" #"RESNET50" VGG16, INCEPTIONV3 , etc
* HYP\_WORD\_EMBEDDING = "GLOVE" # word2vec, etc
* HYP\_DECODER\_LOSS = 'categorical\_crossentropy'
* HYP\_DECODER\_OPTIMIZER = 'adam'
* HYP\_NO\_OF\_PICS\_PER\_BATCH = 100
* HYP\_EPOCH = 45
* HYP\_DECODER\_LEARNING\_RATE = 0.0001
* HYP\_ENABLE\_ATTENTION = False # False
* HYP\_ENABLE\_OPTIMIZER = False
* HYP\_ENABLE\_TRANSFORMER = True
* HYP\_GENERATE\_BLEU\_SCORE = True
* HYP\_GENERATE\_METEOR\_SCORE = True

## Training the model

To train the model, we will be using the 6000 training images by generating the input and output sequences in batches and fitting them to the model using lstm\_model.fit\_generator() method. We also save the model to our models folder. This will take some time depending on your system capability. Final Model Structure is shown in chapter 3

## Testing the model

The model has been trained, now, we will make a separate file testing\_caption\_generator.py which will load the model and generate predictions. The predictions contain the max length of index values so we will use the same tokenizer.p pickle file to get the words from their index values.

# Results

Automatic image captioning model was developed with Encoder – decoder architecture. A base line model was developed initially with ResNet50 as encoder and LSTM as decoder. This model was first trained on Fliker-8K dataset. As an improvement the model was trained with “categorical\_crossentropy” loss function and “adam” optimizer. To improve the accuracy further an Attention Layer was implemented with ResNet50 as encoder and LSTM as decoder. The BLEU score computed was showing improvements with attention layer.

Towards improving the accuracy of the model further, we have modified the Image Captioning model with transformers as Decoder and ResNet50 as Encoder. The following tables show the metrics of the model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model Architecture** | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | Meteor score |
| **ResNet50+LSTM+Optimizer** | 0.247679399 | 0.00E-01 | 0.00E-01 | 0.00E-01 | 0.06329113924 |
| **ResNet50+LSTM+Attention+Optimizer** | 0.4545454545 | 0.4545454545 | 2.23E-308 | 2.23E-308 | 0.2652310924 |
| **ResNet50+Transformer** | 0.44444444 | 2.23E-308 | 2.23E-308 | 2.23E-308 | 0.1388888889 |

The model was saved and it was successfully deployed using FastAPI framework.

# Conclusion

In this project report, we have reviewed deep learning-based image captioning methods. We have given taxonomy of image captioning techniques, shown generic block diagram of the major groups and highlighted their pros and cons. A brief summary of experimental results is also given. Although deep learning-based image captioning methods have achieved a remarkable progress in recent years, a robust image captioning method that is able to generate high quality captions for all images is yet to be achieved. With the advent of novel deep learning network architectures, automatic image captioning will remain an active research area for some time.

We have used Flickr\_8k dataset which includes nearly 8000 images, and the corresponding captions are also stored in the text file.

The model is trained to maximize the likelihood of the target description sentence given the training image. The accuracy of the model and the fluency of the language it learns solely from image descriptions. Our model is trained with Flicker-8k dataset, and tested over the test set. The model has produced the BLEU-1 score (the higher the better) on the Flicker-8k dataset is 0.34; we also show BLEU-1 score improves with Attention layer (0.4545454545) and with transformer (0.4444444444).

# References

1. Image Captioning Based on Deep Neural Networks
2. Automatic Image Captioning Based on ResNet50 and LSTM with Soft Attention

Yan Chu ,1 Xiao Yue ,2 Lei Yu,1 Mikhailov Sergei,1 and Zhengkui Wang3

1. A Neural Image Caption Generator

Oriol Vinyals Google , Alexander Toshev Google, Samy Bengio Google, Dumitru Erhan Google

1. You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhad

University of Washington, Allen Institute for AI†, Facebook AI Research

1. <https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/>
2. <https://rupamgoyal12.medium.com/image-caption-generator-using-resnet50-and-lstm-model-a5b11f60cd23>
3. <https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/>