# IMAGE CAPTION GENERATOR USING LSTM

Authors Name: Priyadharshini Boopathy
Department of Computing Sciences
Coimbatore Institute of Technology
Coimbatore-641014, India.
Email id: priyapd2128cbe@gmail.com

Authors Name: Keerthika Jagadeesh
Department of Computing Science
Coimbatore Institute of Technology
Coimbatore-641014, India
Email id: keerthu3103@gmail.com

Authors Name: Sadhvi Anunaya Ravikumar
Department of Computing Sciences
Coimbatore Institute of Technology
Coimbatore-641014, India.
Email id: sadhvianunaya@gmail.com

## ABSTRACT:

Creating an image's description or caption automatically using natural language words is a challenging feat. With the rapid growth of artificial intelligence in recent years, image caption has steadily drawn the attention of many researchers in the field of artificial intelligence and has become a fascinating and difficult task. Initially, it was thought that a computer could not describe an image. With the progress of Deep Learning Techniques and the availability of enormous amounts of data, we can now build models that can generate captions summarizing an image. This study examines a deep neural network-based image caption generating approach in depth. With an image as input, the approach generates an English sentence expressing the image's content. It involves the analysis of three components: The convolutional neural network (CNN), recurrent neural network (RNN), and sentence creation.

Keywords: Caption Generator, Deep Learning, Automated Captions, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM).

## I. INTRODUCTION:

Deep learning is one of the most rapidly advancing and researched fields of study that is making its way into all of our daily lives. It is a subfield of machine learning concerned with algorithms and inspired by the structure and function of the brain making its way into all of our daily lives. It is a subfield of machine learning concerned with algorithms and inspired by the structure and function of the brain. It involves Deep Learning Techniques of Convolutional Neural Networks and a type of Recurrent Neural Network (Long Short-Term Memory) together. In this, an Image caption generator, the basis of our provided or uploaded image file will generate the caption from a trained model trained using algorithms and on a large dataset. Also, it could provide more accurate and compact information about images.

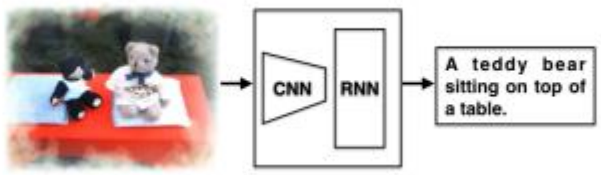


Figure 1: Image caption generation pipeline. The framework consists of a convolutional neural network (CNN) followed by a recurrent neural network (RNN). It generates an English sentence from an input image.[1]

## II. DESCRIPTION OF THE PROBLEM:

Computer vision in image processing has made tremendous advances in recent years, such as image classification and object detection. Image captioning is an issue that can be solved by using developments in image classification and object detection to automatically construct one or more phrases to understand the visual content of an image. In this project, we want to build a system that can generate an English sentence that describes objects, actions, or events.

## III. LITERATURE REVIEW:

Automatically generating complete and natural image descriptions offers a wide range of possible applications, including titles connected to news photographs, descriptions associated with medical images, text-based image retrieval, information access for blind users, and human-robot interaction. These picture captioning applications have significant theoretical and practical research significance. After reading all the articles and being inspired, we achieved a good understanding of the key aspects of image captioning. The structure is presented below: •

- Year 2016; 2017; 2018; 2019;
- Feature extractors: Alex Net; VGG-16 Net; ResNet; Google Net (including all nine Inception models); Dense Net;
- Language models: LSTM; RNN; CNN; cGRU; TPGN;
- Methods: Encoder-decoder; Attention mechanism; Novel objects; Semantics;
- Results on datasets: MS COCO; Flickr30k; Flickr 8k
- Evaluation metrics: BLEU-1; BLEU-2; BLEU-3; BLEU-4; Cider; METEOR.

## IV. PROPOSED METHODOLOGY:

### I. *Task:*

The task of this project is to create a system that takes the image input in the form of a dimensional array and generates an output that should be in the form of a sentence and should describe the image which should be syntactically and grammatically correct.

### II. *Corpus:*

The Flickr 8k image dataset is taken as the corpus. The dataset contains 8000 images and each image has 5 captions. The 5 captions for a single image help in understanding all possible scenarios of that image input. The dataset has a predefined train and test dataset of sizes 6000 images and 1000 images.

Image Dataset: https://github.com/jbrownlee/Datasets/releases/download/Fl ickr8k/Flickr8k_Dataset.zip
Text Dataset: https://github.com/jbrownlee/Datasets/releases/download/Fl ickr8k/Flickr8k_text.zip

### III. *Preprocessing:*

Data Preprocessing can be done in two parts, the images and their corresponding captions are cleaned and pre-processed separately. Preprocessing for images can be done by feeding the input to the

Exception application of the Keras API, which runs on top of TensorFlow. The exception is pre-trained on ImageNet. With the help of Transfer Learning, the images can be trained faster. For this tokenizer class from Keras to clean the description, this will help to vectorize the text corpus and it is stored in a separate dictionary and each text is mapped with a unique index value.

IV. *Model:*

Deep learning carries out the machine learning process using an artificial neural network that is composed of several levels arranged in a hierarchy. The model is based on deep networks where the flow of information starts from the initial level, where the model learns something simple and then the output of which is passed to layer two of the networks, and the input is combined into something that is a bit more complex and passes it on to the third level. This process continues as each level in the network produces something more complex from the input it received from the ascendant level. [2]

**Convolutional Neural Network (CNN):**

Convolutional Neural Network is one of the specialized deep neural networks which can process the data where the input shape should be in a 2D matrix. CNN is difficult in working with images. It takes the images as input, assigns importance to various objects in the image, and differentiates one from the other. The CNN uses Kernel filters, which help in feature learning, which is more or less the same as the human brain identifying objects in time and space. This architecture is a better fit for an image dataset because of the reduction in the number of parameters and the reusability of weights.

**Recurrent Neural Network (RNN):**

The human brain evolved in a way that it should sense the words first and keep those words in mind and then generate the next words, thus forming a sentence. But In Deep Learning the basic Neural Networks don't have this type of ability as humans have. However, advancements in Recurrent Neural networks have a solution to this issue. RNNs are like networks that loop in and also allow the information by the use of their internal states, which will lead to the creation of a feedback loop. [3]

Long Short-Term Memory networks (LSTM) is a special kind of RNN, is capable of learning long-term dependencies which is nothing but remembering the information for a long period. It is practically their default behavior and this kind of behavior can be controlled with the help of gates. RNN can process only single data points, whereas LSTM can process the entire sequence. Not only this, but they can learn which point in the data holds importance and which can be thrown away. So, the information which is relevant is passed to the next layer.

The three main gates are involved in this process, they are input gate, output gate and forget gate. These gates help us to decide whether to forget the current cell value, read a value into the cell or output the cell value. The hidden states play an important role in this process where the hidden states are passed to the next step of the sequence. The hidden state acts as the neural network's memory, as it is storing the data that the neural network has seen before. Thus, it allows the neural network to function like a human brain trying to form sentences. [3,4]

**Architecture:**

CNN + LSTM architecture has been used to take an image as input and output a caption. An “encoder” RNN maps the source sentence (which is of variable length) and transforms it into a fixed-length vector representation, which in turn is used as the initial hidden state of a “decoder” RNN which ultimately generates the final meaningful sentence as a prediction. [4]

However, RNN has to be replaced with a deep CNN - since it can produce a rich representation of the input image by embedding it to a fixed-length vector - 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. [3 - 5]

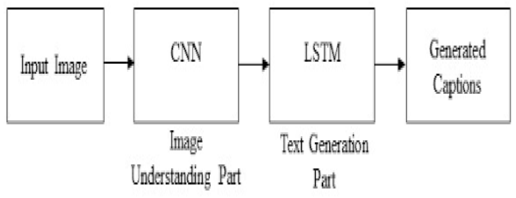


Figure 2: CNN Architecture for image caption generator

## V. Evaluation

There are five main steps that have been executed in the module

### A. *Data Cleaning and Preprocessing*

Google Colaboratory for execution was used which has GPU/TPU processing power. Comparing to local machines GPU completes the work faster and more effectively. First step of the program is will load both the text and the image file into different variables and the text file is stored in the string values. This string is manipulated to create a dictionary that maps each image with 5 descriptions. Next is the data preprocessing step where the punctuation marks are removed, changes the sentence to lowercase, removes the numerical values, and stops the words.
From the description, a vocabulary from the unique words is extracted which is used in the testing process for the generation of captions. Tokenization is another preprocessing technique where you set unique index values for the vocabulary. As the computer does not understand the English words set numbers for each word. The tokens are stored in the form of a character stream file. The preprocessing techniques can be made easier with the use of the Keras.preprocessing module. To indicate LSTM about the start and end of the caption, append <start> and <end> identifiers for the captions. After this, the total number of vocabulary words is calculated and a maximum length of the descriptions is also found for upcoming processes.

### B. *Extraction of feature vectors:*

A feature vector is a vector that depicts the pixels and objects in the image. It contains characteristics of the object like the image pixel’s intensity value. In our model, the most popular technique used in inception V3 is Transfer Learning. Inception V3 is a convolutional network with 48 layers deep which is trained on imagenet dataset. This dataset contains over 14 million images. Using python this model can be used more easily with Keras.applications.inceptionV3. The classification layer gives a 2048 feature vector which can be used in the LSTM model. Now will load glove vectors. It includes global word co-occurrence to get word vectors. Hence images' weight will be extracted and then names of the images will be mapped with their respective feature array. Based on the processor this process takes time.

## *C. Layering the CNN-RNN model:*

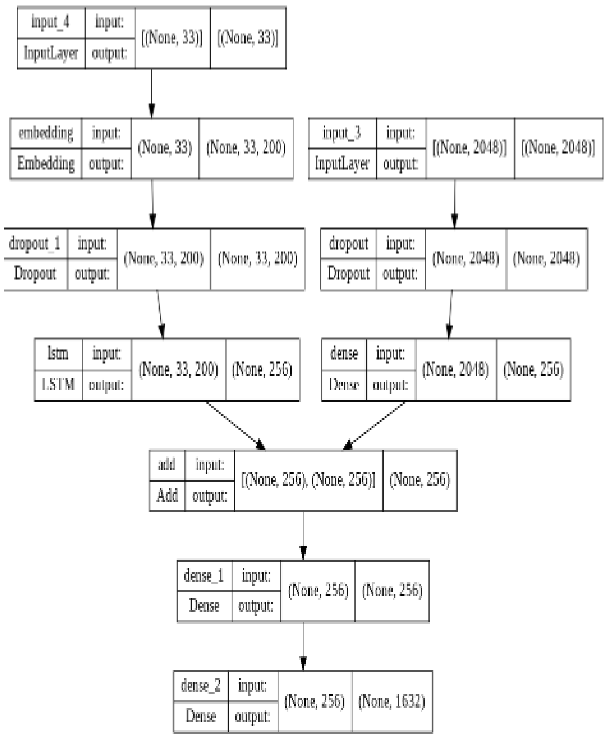
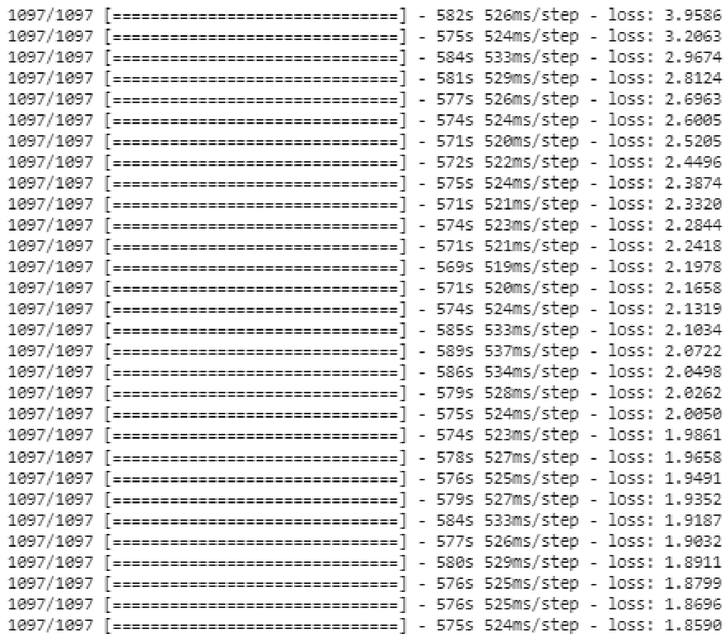

input_4 | InputLayer | input: | output: | [(None, 33)] | [(None, 33)]
embedding | Embedding | input: | output: | (None, 33) | (None, 33, 200)
input_3 | InputLayer | input: | output: | [(None, 2048)] | [(None, 2048)]
dropout_1 | Dropout | input: | output: | (None, 33, 200) | (None, 33, 200)
dropout | Dropout | input: | output: | (None, 2048) | (None, 2048)
lstm | LSTM | input: | output: | (None, 33, 200) | (None, 256)
dense | Dense | input: | output: | (None, 2048) | (None, 256)
add | Add | input: | output: | [(None, 256), (None, 256)] | (None, 256)
dense_1 | Dense | input: | output: | (None, 256) | (None, 256)
dense_2 | Dense | input: | output: | (None, 256) | (None, 1632)

Figure 3: Layers of CNN and RNN model

To stack the model, Use the Keras Model from Functional API. The structure will consist of 3 parts:

1. **Feature Extractor**: The dropout layer reduces the dimension. Using it the dimensions are reduced from 2048 to 256. One of these will be added to the CNN and the LSTM each. The preprocessed images are preprocessed by the inception V3 model and the extracted features predicted by this model are used as input.
2. **Sequence Processor**: This Embedding layer handles the textual input, followed by the LSTM layer.
3. **Decoder**: By merging the output from the two layers will get a dense layer using it the final prediction can be made. The vocabulary size will be the number of nodes obtained in the final layer.

## *D. Training the model:*



```
1097/1097 [==============================] - 582s 526ms/step - loss: 3.9586
1097/1097 [==============================] - 575s 524ms/step - loss: 3.2063
1097/1097 [==============================] - 584s 533ms/step - loss: 2.9674
1097/1097 [==============================] - 581s 529ms/step - loss: 2.8124
1097/1097 [==============================] - 577s 526ms/step - loss: 2.6963
1097/1097 [==============================] - 574s 524ms/step - loss: 2.6005
1097/1097 [==============================] - 571s 520ms/step - loss: 2.5205
1097/1097 [==============================] - 572s 522ms/step - loss: 2.4496
1097/1097 [==============================] - 575s 524ms/step - loss: 2.3874
1097/1097 [==============================] - 571s 521ms/step - loss: 2.3320
1097/1097 [==============================] - 574s 523ms/step - loss: 2.2844
1097/1097 [==============================] - 571s 521ms/step - loss: 2.2418
1097/1097 [==============================] - 569s 519ms/step - loss: 2.1978
1097/1097 [==============================] - 571s 520ms/step - loss: 2.1658
1097/1097 [==============================] - 574s 524ms/step - loss: 2.1319
1097/1097 [==============================] - 585s 533ms/step - loss: 2.1034
1097/1097 [==============================] - 589s 537ms/step - loss: 2.0722
1097/1097 [==============================] - 586s 534ms/step - loss: 2.0498
1097/1097 [==============================] - 579s 528ms/step - loss: 2.0262
1097/1097 [==============================] - 575s 524ms/step - loss: 2.0050
1097/1097 [==============================] - 574s 523ms/step - loss: 1.9861
1097/1097 [==============================] - 578s 527ms/step - loss: 1.9658
1097/1097 [==============================] - 576s 525ms/step - loss: 1.9491
1097/1097 [==============================] - 579s 527ms/step - loss: 1.9352
1097/1097 [==============================] - 584s 533ms/step - loss: 1.9187
1097/1097 [==============================] - 577s 526ms/step - loss: 1.9032
1097/1097 [==============================] - 580s 529ms/step - loss: 1.8911
1097/1097 [==============================] - 576s 525ms/step - loss: 1.8799
1097/1097 [==============================] - 576s 525ms/step - loss: 1.8696
1097/1097 [==============================] - 575s 524ms/step - loss: 1.8590
```

Figure 4: Epoch after training the model

We have 6000 images for training where every image has 2048 feature vectors. Since the data is big and cannot store large amounts of data, we use a Data generator. This will create batches so that the output can be generated faster.

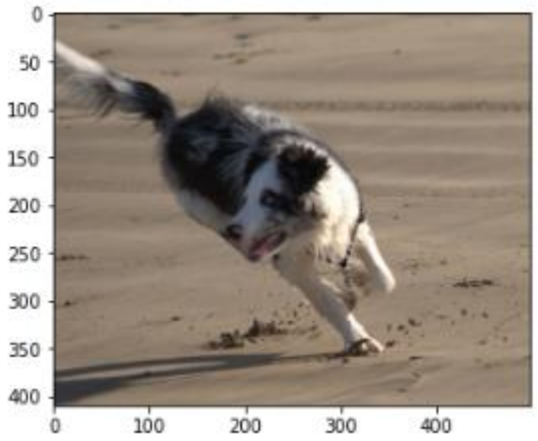
After batching we will be defining the number of epochs to train the data in such a way that it should not underfit or overfit the model. The model.fit() method will be used. And this whole process will take some time depending on the processor. The parameter value is obtained from the maximum length of the description. It will also take as input the clean and tokenized data and the tokenized data is also taken as the input. Next, the sequence creator will be created which is used in predicting the next word from the previous work and from the image's feature vectors.
After this while training our model, we can save the model likewise we will be saving several models from which we will decide which model to be used for final testing.

E. *Testing the model:*

Load the training models that have been already saved. Now the sequence generator will give start and end identifiers. Using Greedy search generates different captions by giving different input values from the list of test images. The caption is generated for each image and checks the accuracy of the image with the help of the BLEU(Bilingual Evaluation Understudy) score to know the efficiency of the model.

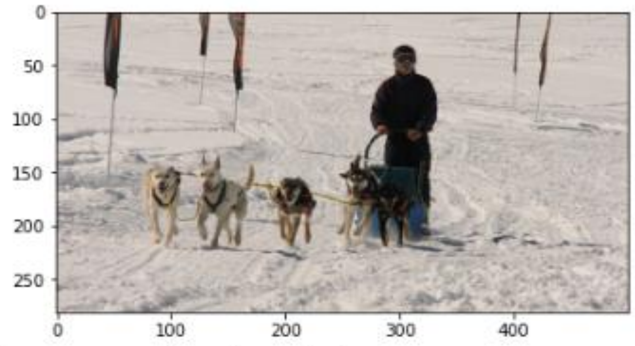
**OUTPUT:**

**1. Result One**



```
dog is running on the beach
```

**2.Result after training even more**



```
dogs are running through the snow
```

## VI. CONCLUSION:

Based on the results, we can see that the deep learning methodology has helped us to get a successful result. The CNN and LSTM have worked together in proper synchronization and thus help us to find the relation between the objects in the images. To compare the accuracy of the predicted caption, we can compare them with target captions in our Flickr 8k test dataset, using BLEU (Bilingual Evaluation Understudy) score [5, 6].

BLEU scores are used for evaluating the translated text. Over the years, many neural network technologies have been used to create an image caption generator similar to the one that we have proposed here. BLEU score can also be used to compare the different models and to see which model gives the maximum accuracy. This paper helped us get to know about more developments in the field of Deep Learning. In Fact, this paper tries to cover the basic methods which have been used to create an image caption generator but there are several topics in this paper that are open to further research and development.

## REFERENCE:

1. https://www.math.ucla.edu/~minchen/doc/ImgCapGen.pdf
2. HaoranWang , Yue Zhang, and Xiaosheng Yu, “An Overview of Image Caption Generation Methods”, (CIN-2020)
3. Priyanka Kalena, Nishi Malde, Aromal Nair, Saurabh Parkar, and Grishma Sharma, “Visual Image Caption Generator Using Deep Learning”, (ICAST-2019)
4. Pranay Mathur, Aman Gill, Aayush Yadav, Anurag Mishra, and Nand Kumar Bansode,“Camera2Caption: A Real-Time Image Caption Generator”, International Conference on Computational Intelligence in Data Science(ICCIDS) - 2017
5. Oriol Vinyals, Alexander Toshev, SamyBengio, and Dumitru Erhan, “Show and Tell: A Neural Image Caption Generator”,(CVPR 1, 2- 2015)
6. K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, et al., "Show, attend and tell: Neural image caption generation with visual attention", Proceedings of the International Conference on Machine Learning (ICML), 2015.