IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024

# Generating Descriptive Text From Images - Image Caption Generator

**Avdhi Pagariya and Riddhi Jain**
Acropolis Institute of Technology and Research, Indore, Madhya Pradesh, India

**Abstract:**. *In the modern era, image captioning has become one of the most widely required tools. Moreover, there are inbuilt applications that generate and provide a caption for a certain image, all these things are done with the help of deep neural network models. The process of generating a description of an image is called image captioning. It requires recognizing the important objects, their attributes, and the relationships among the objects in an image. It generates syntactically and semantically correct sentences. In this paper, we present a deep learning model to describe images and generate captions using computer vision and machine translation. This paper aims to detect different objects found in an image, recognize the relationships between those objects and generate captions. The dataset used is Flickr8k and the programming language used was Python3, and an ML technique called Transfer Learning will be implemented with the help of the caption model, to demonstrate the proposed experiment. This paper will also elaborate on the functions and structure of the various Neural networks involved. Generating image captions is an important aspect of Computer Vision and Natural language processing. Image caption generators can find applications in Image segmentation as used by Facebook and Google Photos, and even more so, its use can be extended to video frames. They will easily automate the job of a person who has to interpret images. Not to mention it has immense scope in helping visually impaired people*

**Keywords:** Image, Caption, CNN, caption, RNN, LSTM,Neural Networks

## I. INTRODUCTION

Making a computer system detect objects and describe them using natural language processing (NLP) in an age-old problem of Artificial Intelligence. This was considered an impossible task by computer vision researchers till now. With the growing advancements in Deep learning techniques, availability of vast datasets, and computational power, models are often built which will generate captions for an image. Image caption generation is a task that involves image processing and natural language processing concepts to recognize the context of an image and describe them in a natural language like English or any other language. While human beings are able to do it easily, it takes a strong algorithm and a lot of computational power for a computer system to do so. Many attempts have been made to simplify this problem and break it down into various simpler problems such as object detection, image classification, and text generation. A computer system takes input images as two-dimensional arrays and mapping is done from images to captions or descriptive sentences.

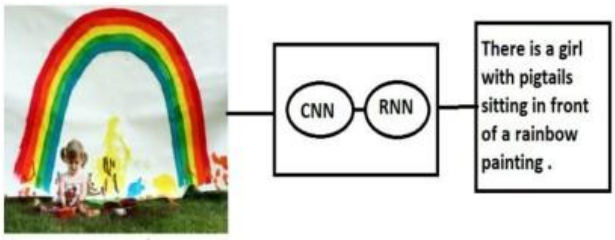


Figure 1: Our model is based on a deep learning neural network that consists of a vision CNN followed by a language generating RNN. It generates complete sentences as an output

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024

In recent years a lot of attention has been drawn towards the task of automatically generating captions for images. However, while new datasets often spur considerable innovation, benchmark datasets also require fast, accurate, and competitive evaluation metrics to encourage rapid progress. Being able to automatically describe the content of a picture using properly formed English sentences may be a very challenging task, but it could have an excellent impact, as an example by helping visually impaired people better understand the content of images online. This task is significantly harder, for instance than the well-studied image classification or visual perception tasks, which are a main focus within the computer vision community Deep learning methods have demonstrated advanced results on caption generation problems. What is most impressive about these methods is that one end-to-end model is often defined to predict a caption, given a photograph, rather than requiring sophisticated data preparation or a pipeline of specifically designed models. Deep learning has attracted a lot of attention because it's particularly good at a kind of learning that has the potential to be very useful for real- world applications.

The ability to find out from unlabeled or unstructured data is a huge benefit for those curious about real-world applications. [1 - 4]

## II. PROBLEM STATEMENT

The main problem in the development of image description started with object detection using static object class libraries in the image and modeled using statistical language models. [5]

**Image Caption Generator**

- Making use of CNN: It's a Deep Learning algorithm that will intake in a 2D matrix input image, assign importance (learnable weights and biases) to different aspects/objects in the image, and be intelligent enough to be able to differentiate one from the other.
- This model was advantageous in naming the objects in an image but it could not tell us the relationship among them(that's plain image classification).
- In this paper, we present a generative model built on a deep recurrent architecture that unites recent advances in computer vision and machine translation and that can effectively generate meaningful sentences.
- Making use of an RNN: They are networks with loops in them, allowing information to persist. LSTMs are a particular kind of RNN, capable of learning long-term dependencies. [5]

## III. PROPOSED METHODOLOGY

**Task**

The task is to build a system that will take an image input in the form of a dimensional array and generate an output consisting of a sentence that describes the image and is syntactically and grammatically correct.

**Corpus**

We have used the Flickr 8K dataset as the corpus. The dataset consists of 8000 images and for every image, there are 5 captions. The 5 captions for a single image helps in understanding all the various possible scenarios. The dataset has a predefined training dataset Flickr_8k.trainImages.txt (6,000 images), development dataset Flickr_8k.devImages.txt (1,000 images), and test dataset Flickr_8k.testImages.txt (1,000 images).

The Images are opted from six varied Flickr groups and do not contain any well-known personality or places. However, they are manually selected to show a variety of scenes.[1] These datasets(Size 1GB) can be directly downloaded from the given links

**Image Dataset**: Datasets/releases/download/Fl ickr8k/Flickr8k_Dataset.zip

**Text Dataset:** /Datasets/releases/download/Fl ickr8k/Flickr8k_text.zip

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024


**Figure 2: Glimpse of the Flickr8k Image Dataset**

```
1000268201_693b08cb0e.jpg#0A child in a pink dress is climbing up
a set of stairs in an entry way .
1000268201_693b08cb0e.jpg#1A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2A little girl climbing into a wooden
playhouse .
1000268201_693b08cb0e.jpg#3A little girl climbing the stairs to
her playhouse .
1000268201_693b08cb0e.jpg#4A little girl in a pink dress going
into a wooden cabin .
```

**Figure 3: Glimpse of Flickr8k Text File**

**Preprocessing**

Data preprocessing is done in two parts, the images and the corresponding captions are cleaned and pre-processed separately. Image preprocessing is done by feeding the input data to the Xception application of the Keras API running on top of TensorFlow. Xception is pre-trained on ImageNet. This helped us train the images faster with the help of Transfer learning. The descriptions are cleaned using the tokeniser class in Keras, this will vectorize the text corpus and is stored in a separate dictionary. Then each word of the vocabulary is mapped with a unique index value.

**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 network and 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. [1]

**Convolutional Neural Networks (CNN)**

Convolutional Neural networks are specialized deep neural networks that can process the data that has input shape like a 2D matrix. Images can be easily represented as a 2D matrix CNN is crucial in working with images. It takes as input an image, assigns importance (weights and biases) to various aspects/objects in the image, and differentiates one from theother. The CNN makes use of filters(also known as Kernels) which help in feature learning(detect abstract concepts, like Blurring, Edge Detection, Sharpening, etc), much the same as a human brain identifying objects in time and space. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved(from 2048 to 256) and the reusability of weights.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024

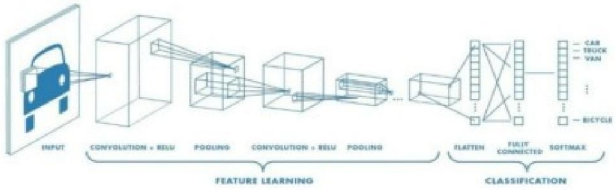


**Figure 4: Architecture of Convolutional Neural Networks forobject classification.[11]**

**Recurrent Neural Networks (RNN)**

The human brain is evolved in such a way so as to make sense of previous words, and keeping these in mind generate the next words, thus forming a perfect sentence. Basic Neural networks don't have the ability to do this. However, advancements in Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist for a while, by making use of their internal states, thus creating a feedback loop. [6]

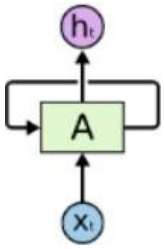


**Figure 5: Loop in RNN[6]**

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. Remembering information for long periods is practically their default behaviour, and this behaviour is controlled with the help of "gates".

While RNNs process single data points, LSTMs can processentire sequences. Not only that, they can learn which point in the data holds importance, and which can be thrown away. Hence, the only relevant information is passed on to the next layer.

The 3 main gates involved are: input gate, output gate and forget gate. These gates 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 since the previous 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. [6, 7]

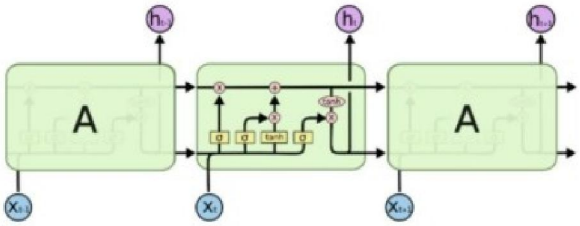


**Figure 6: The recurrent module in LSTM contains fourinterconnected layers.[7]**

**Architecture**

We utilize a CNN + LSTM 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. [7]

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024

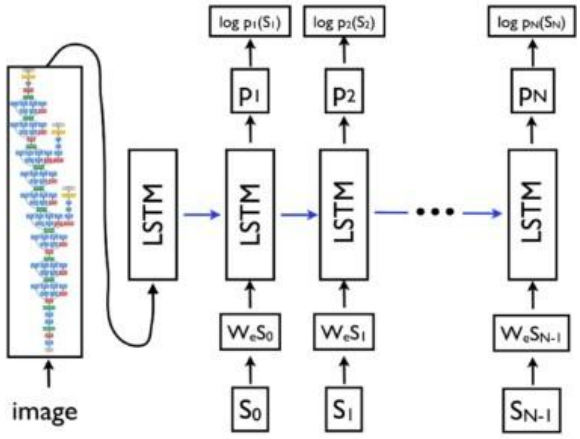


**Figure 7: CNN-LSTM structure.[5]**

However, we are going to replace this RNN 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. [5 - 7]

## IV. EVALUATION

Execution of the entire program takes place in 5 major steps. The implementation of the five major modules is as follows:

**Data Cleaning and Preprocessing:**

1. For a comfortable and fast work experience, we use Google Colaboratory: a tool which provides free GPU/TPU processing power, over our local machines, which can take several hours to do the task that a GPU will take few minutes to do.
2. Our program starts with loading both, the text file, and the image file into separate variables; the test file is stored in a string.
3. This string is used and manipulated in such a way so asto create a dictionary that maps each image with a list of 5 descriptions
4. The main task of data cleaning involves removing punctuation marks, converting the whole text tolowercase, removing stop words and removing words that contain numbers.
5. Further, a vocabulary of all unique words from all the descriptions is created, which in the future will be used to generate captions to test images.
6. Another aspect of Preprocessing the data involves tokenizing our vocabulary with a unique index value. This is because a computer won't understand regular English words, hence they need to be represented using numbers. The tokens are then stored in a *pickle* file. i.e. in the form of character stream, but with all the information necessary to reconstruct it into the original object type.
7. The above two preprocessing tasks can be achieved manually or by using the *Keras.preprocessing*module for ease of writing the code.
8. We proceed to append the <start> and <end> identifier for each caption since these will act as indicators for our LSTM to understand where a caption is starting andwhere it's ending.
9. We will proceed with calculating the number of words in our vocabulary and finding the maximum length of the description, which will be used in later phases.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024

```
1000268201_693b08cb0e.jpg  child in pink dress is climbing up set
of stairs in an entry way
1000268201_693b08cb0e.jpg  girl going into wooden building
1000268201_693b08cb0e.jpg  little girl climbing into wooden
playhouse
1000268201_693b08cb0e.jpg  little girl climbing the stairs to her
playhouse
1000268201_693b08cb0e.jpg  little girl in pink dress going into
wooden cabin
```

**Figure 8: Text file after performing data cleaning**

**Extraction of feature vectors:**

1. A feature vector(or simply feature) is a numerical value in the matrix form, containing information about an object's important characteristics, eg. intensity value of each pixel of the image in our case. These vectors, we'll ultimately store in a pickle file.
2. In our model we'll be using *Transfer Learning*, which simply means, using a pretrained model( in our case the *Xception* model) to extract features from it.
3. The Xception model is a Convolutional Neural Network that is 71 layers deep. It is trained on the famous Imagenet dataset which has millions of images and over 1000 different classes to classify from.
4. Python makes using this model in our code extremely easy with *keras.applications.xception*module.To use it in our code, we'll drop the classification layer from it, and hence obtain the 2048 feature vector.
5. Hence weights will be downloaded for each image, andthen image names will be mapped with their respective feature array.
6. This process can take a few hours depending on your processor.

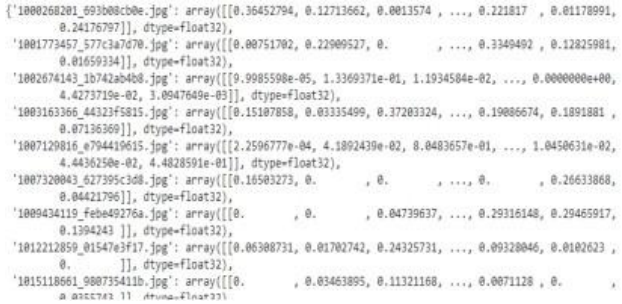


Figure 9: Glimpse of extracted features with corresponding image names, that we'll store in a picklefile

**Layering the CNN-RNN model:**

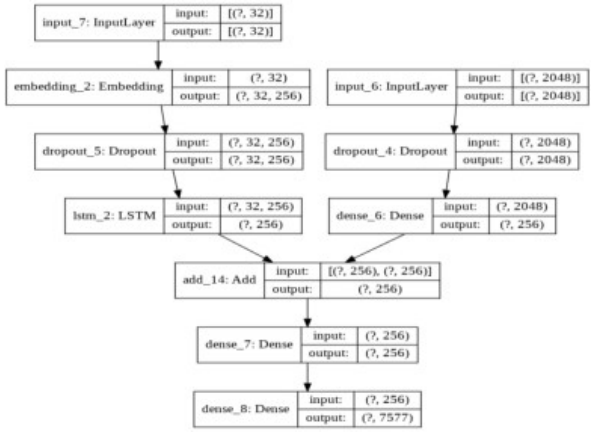


**Figure 10: Structure of the Neural Network**

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024

To stack the model, we'll use the Keras Model from Functional API. The structure will consist of 3 parts:

- **Feature Extractor:** It will be used to reduce the dimensions from 2048 to 256. We'll make use of a *Dropout Layer.* One of these will be added in the CNN and the LSTM each. We have pre-processed the photoswith the Xception model (without the output layer) andwill use the extracted features predicted by this model as input.
- **Sequence Processor:** This *Embedding layer* willhandle the textual input, followed by the LSTM layer.
- **Decoder:** We will merge the output from above two layers, and use a *Dense layer* to make the final predictions. Both the feature extractor and sequenceprocessor output a fixed-length vector. These are merged together and processed by a Dense layer. The number of nodes in the final layer will be the same as the size of our vocabulary. [6,7].

**Training the model:**

```
None
6000/6000 [==============================] - 613s 102ms/step - loss: 4.5162
6000/6000 [==============================] - 577s 96ms/step - loss: 3.6705
6000/6000 [==============================] - 591s 98ms/step - loss: 3.3801
6000/6000 [==============================] - 582s 97ms/step - loss: 3.2075
6000/6000 [==============================] - 571s 95ms/step - loss: 3.0898
6000/6000 [==============================] - 600s 100ms/step - loss: 2.9996
6000/6000 [==============================] - 617s 103ms/step - loss: 2.9267
6000/6000 [==============================] - 629s 105ms/step - loss: 2.8732
6000/6000 [==============================] - 636s 106ms/step - loss: 2.8224
6000/6000 [==============================] - 638s 106ms/step - loss: 2.7851
```

**Figure 11: Model under Training**

1. We'll be training our model on 6000 images each having 2048 long feature vectors.
2. Since it is not possible to hold all this data in the memory at the same time, we'll make use of a *Data Generator.* This will help us create *batches* of the data, and will improve the speed.
3. Along with this we'll be defining the number of epochs(i.e. iterations of the training dataset) the model has to complete during its training. This number has to be selected in such a way that our model is neither *underfitted* nor *overfitted.*
4. *model.fit_generator()* method will be used. And this whole process will take some time depending on the processor.
5. The maximum length of descriptions calculated earlier will be used as parameter value here. It will also take as input the clean and tokenized data.
6. We'll also create a *sequence creator,* which will play the role of predicting the next word based on the previous word and feature vectors of the image.
7. While training our model, we can use the *development dataset*(It's provided with the rest of the files), to monitor the performance of the model, to decide when to save the model version to the fiel
8. We will proceed to save several models, out of which the final one will be used for testing in future.

```
Dataset:  6000
Descriptions: train= 6000
Photos: train= 6000
Vocabulary Size: 7577
Description Length:  32
Model: "functional_5"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_7 (InputLayer)            [(None, 32)]         0
__________________________________________________________________________________________________
input_6 (InputLayer)            [(None, 2048)]       0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 32, 256)      1939712     input_7[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 2048)         0           input_6[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 32, 256)      0           embedding_2[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 256)          524544      dropout_4[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   (None, 256)          525312      dropout_5[0][0]
__________________________________________________________________________________________________
add_14 (Add)                    (None, 256)          0           dense_6[0][0]
                                                                 lstm_2[0][0]
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 256)          65792       add_14[0][0]
__________________________________________________________________________________________________
dense_8 (Dense)                 (None, 7577)         1947289     dense_7[0][0]
==================================================================================================
Total params: 5,002,649
Trainable params: 5,002,649
Non-trainable params: 0
```

**Figure 12: Model details**

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024

**Testing the model:**

A separate Python notebook can be created, or the same can be used to perform testing. Either way, we'll load the trained model that we had saved in the previous step and generate predictions.

The *sequence generator* will come into play at thisstage, besides the tokenizer file we created.

The primary step of feature extraction for the particularimage under observation will be performed.

The path of one of the images from the remaining 2000test images is passed to the function manually. You canalso iterate through the test data set, and store the prediction for each image in a dictionary or a list.

The actual functioning behind image generation involves using the start sequence and the end sequence,and to call the model recursively to generate meaningful sentences.

## V. RESULT / ANALYSIS

For simplicity, only three images have been subjected totesting, and the results can be seen in the following images:

Path of Img 1:

Flicker8k_Dataset/111537222_07e56d5a30.jpg

**Output:**

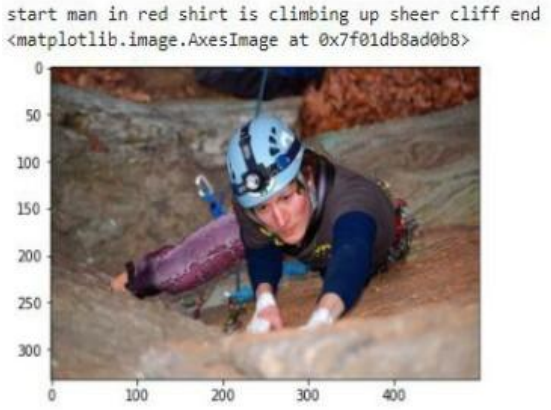


**Figure 13: Caption generated using deep neural networkfor input Image 1**

**Path of Img 2:**

Flicker8k_Dataset/256085101_2c2617c5d0.jpg

Output:

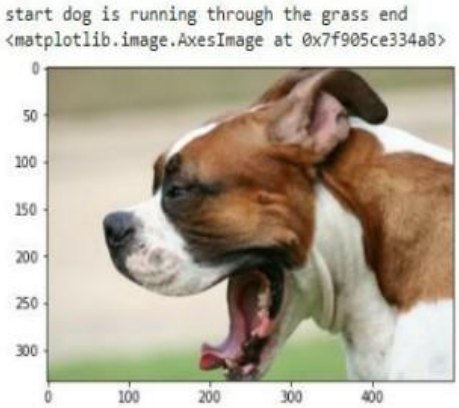


**Figure 14: Caption generated using deep neural networkfor input Image 2**

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024

**Path of Img 3:**
Flicker8k_Dataset/3344233740_c010378da7.jpg
Output:


**Figure 15: Caption generated using deep neural networkfor input Image 3**

| Image | The original description | Predicted description |
|---|---|---|
| 111537222_07e56d5a30.jpg | climber wearing blue helmet and headlamp is attached to rope on the rock face | man in red shirt is climbing up sheer cliff |
| 256085101_2c2617c5d0.jpg | dog with its mouth opened | dog is running through the grass |
| 3344233740_c010378da7.jpg | man is standing on sidewalk in the background with blurry image of another man in the foreground | man in black shirt and black shoes stands in front of traffic |

Table 1: Comparison between original and predictedvalues

## VI. CONCLUSION

Based on the results obtained we can see that the deep learning methodology used here bore successful results.
The CNN and the LSTM worked together in proper synchronization, they were able to find the relation betweenobjects in images.
To compare the accuracy of the predicted caption, we can compare them with target captions in our Flickr8k test dataset, using BLEU(Bilingual Evaluation Understudy) score [5, 8]. BLEU scores are used in text translation for evaluating translated text against one or more reference translations. Over the years several other neural network technologies have been used to create hybrid image caption
generators, similar to the one proposed here. eg VGG16 model instead of the Xception model, or the GRU model instead of the STM model. Furthermore, BLEU Score can be used to draw comparisons between these models, to see which one provides maximum accuracy. This paper introduced us to various new developments in the field of machine learning and AI, and how vast this field is. In Fact several topics within this paper are open to further research and development, while this paper itself tries to cover the basic essentials needed to create an image caption generator.

## REFERENCES

**[1].** HaoranWang , Yue Zhang, and Xiaosheng Yu, "An Overview of Image Caption Generation Methods", (CIN-2020)

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, July 2024

**[2].** B.Krishnakumar, K.Kousalya, S.Gokul, R.Karthikeyan, and D.Kaviyarasu, "IMAGE CAPTION GENERATOR USING DEEP LEARNING", (international Journal of Advanced Science and Technology-2020 )

**[3].** MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga, "A Comprehensive Survey of Deep Learning for ImageCaptioning" ,(ACM-2019)

**[4].** Rehab Alahmadi, Chung Hyuk Park, and James Hahn, "Sequence-to- sequence image caption generator", (ICMV-2018)

**[5].** Oriol Vinyals, Alexander Toshev, SamyBengio, and Dumitru Erhan, "Show and Tell: A Neural Image Caption Generator",(CVPR 1, 2- 2015)

**[6].** Priyanka Kalena, Nishi Malde, Aromal Nair, Saurabh Parkar, and Grishma Sharma, "Visual Image Caption Generator Using Deep Learning", (ICAST-2019)