iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL
FOR RESEARCH
IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 Issue: X Month of publication: October 2021

DOI: https://doi.org/10.22214/ijraset.2021.38652

Image Caption Generator Using Deep Learning

A. V. N. Kameswari[1], B. Prajna[2]
[1]*Information Technology & Computer Applications Department, Andhra University*
[2]*Computer Science & Systems Engineering Department, Andhra University*

Abstract: When humans see an image, their brain can easily tell what the image is about, but a computer cannot do it easily. Computer vision researchers worked on this a lot and they considered it impossible until now! With the advancement in Deep learning techniques, availability of huge datasets and computer power, we can build models that can generate captions for an image. Image Caption Generator is a popular research area of Deep Learning that deals with image understanding and a language description for that image. Generating well-formed sentences requires both syntactic and semantic understanding of the language. Being able to describe the content of an image using accurately formed sentences is a very challenging task, but it could also have a great impact, by helping visually impaired people better understand the content of images. The biggest challenge is most definitely being able to create a description that must capture not only the objects contained in an image, but also express how these objects relate to each other. This paper uses Flickr_8K dataset and Flickr8k_text folder that contains Flickr8k.token which is the main file of our dataset that contains image name and their respective caption separated by newline("\n"). CNN is used for extracting features from the image. We will use the pre-trained model Xception. LSTM will use the information from CNN to help generate a description of the image. In our Flickr8k_text folder, we have Flickr_8k.trainImages.txt file that contains a list of 6000 images names that we will use for training. After CNN-LSTM model is defined we give an image file as parameter through command prompt for testing image caption generator and it generates the caption of an image and its accuracy is observed by calculating bleu score for generated and reference captions.

Keywords: Image Caption Generator, Convolutional Neural Network, Long Short-Term Memory, Bleu score, Flickr_8K

I. INTRODUCTION

A. Problem Definition

Image captioning has various applications such as recommendations in editing applications, usage in virtual assistants, for image indexing, for visually impaired persons, for social media and several other natural language processing applications. Image caption generator is a task that involves computer vision and natural language processing concepts to recognize the context of an image and describe them in a neural language like English. This image caption generator using classic neural network has the following problems

1) Inaccurate results
2) Short internal memory

The main objective of this paper is to build a working model of Image Caption generator by implementing CNN with LSTM. The existing system uses classic Recurrent Neural Network (RNN) which has internal memory (short memory), which was neither popular nor powerful, which may also generate inaccurate results. So, to overcome this problem here we use Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) for efficient and accurate results.

B. Motivation

We can create a product for the blind which will guide them travelling on the roads without the support of anyone else. We can do this by first converting the scene into text and then the text to voice. Both are now famous applications of deep learning. Self-driving cars-Automatic driving is one of the biggest challenges and if we can properly caption the scene around the car, it can give a boost to the self-driving system.

Automatic Captioning can help, make Google Image Search as good as Google Search, as then every image could be first converted into a caption and then search can be performed based on the caption. In web development, it's good practice to provide a description for any image that appears on the page so that an image can be read or heard as opposed to just seen. This makes web content accessible.

CCTV cameras (Closed-Circuit Television Cameras) are everywhere today, but along with viewing the world, if we can also generate relevant captions, then we can raise alarms as soon as there is some malicious activity going on somewhere. This could probably help reduce some crimes and accidents. It can be used to describe video in real time.

II. DATA SET

The Flickr_8K dataset has nearly 8000 images are used for training placed in Flickr_8k.trainImages.txt, 1000 images are used in development placed in Flickr_8k.devImages.txt, 1000 images are used in testing placed in Flickr_8k.testImages.txt and Flickr8k,token.txt file contain 5 captions each for all the images in the dataset. These files are present in Flickr8k_text folder.

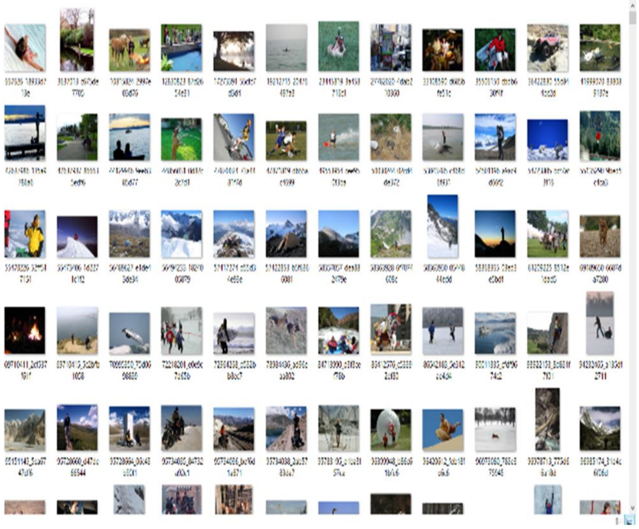
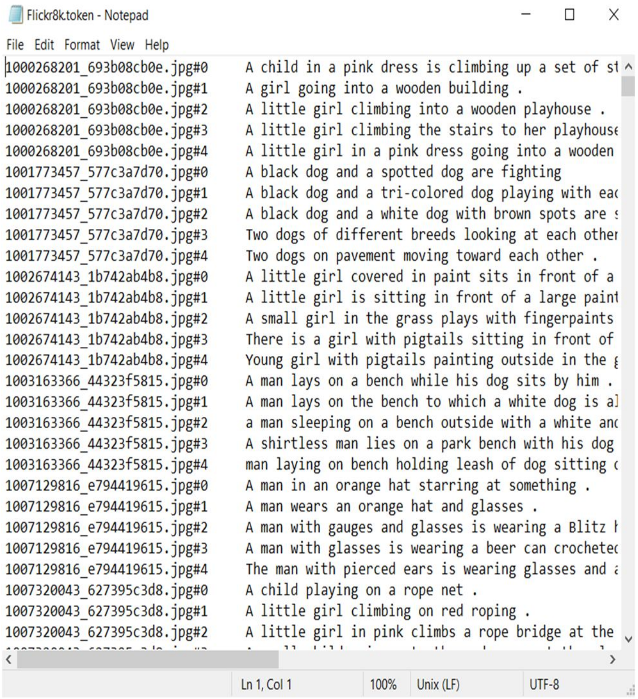


Fig. 1 Flickr_8k Data Set



```
1000268201_693b08cb0e.jpg#0	A child in a pink dress is climbing up a set of st
1000268201_693b08cb0e.jpg#1	A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2	A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg#3	A little girl climbing the stairs to her playhouse
1000268201_693b08cb0e.jpg#4	A little girl in a pink dress going into a wooden
1001773457_577c3a7d70.jpg#0	A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg#1	A black dog and a tri-colored dog playing with eac
1001773457_577c3a7d70.jpg#2	A black dog and a white dog with brown spots are s
1001773457_577c3a7d70.jpg#3	Two dogs of different breeds looking at each other
1001773457_577c3a7d70.jpg#4	Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg#0	A little girl covered in paint sits in front of a
1002674143_1b742ab4b8.jpg#1	A little girl is sitting in front of a large paint
1002674143_1b742ab4b8.jpg#2	A small girl in the grass plays with fingerpaints
1002674143_1b742ab4b8.jpg#3	There is a girl with pigtails sitting in front of
1002674143_1b742ab4b8.jpg#4	Young girl with pigtails painting outside in the g
1003163366_44323f5815.jpg#0	A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg#1	A man lays on the bench to which a white dog is al
1003163366_44323f5815.jpg#2	a man sleeping on a bench outside with a white anc
1003163366_44323f5815.jpg#3	A shirtless man lies on a park bench with his dog
1003163366_44323f5815.jpg#4	man laying on bench holding leash of dog sitting c
1007129816_e794419615.jpg#0	A man in an orange hat starring at something .
1007129816_e794419615.jpg#1	A man wears an orange hat and glasses .
1007129816_e794419615.jpg#2	A man with gauges and glasses is wearing a Blitz h
1007129816_e794419615.jpg#3	A man with glasses is wearing a beer can crochetec
1007129816_e794419615.jpg#4	The man with pierced ears is wearing glasses and a
1007320043_627395c3d8.jpg#0	A child playing on a rope net .
1007320043_627395c3d8.jpg#1	A little girl climbing on red roping .
1007320043_627395c3d8.jpg#2	A little girl in pink climbs a rope bridge at the
```

Fig. 2 Flickr8k.token.txt

III. EXISTING SYSTEM

The existing methods for image caption generator can be classified by linear and non-linear models.

1) Linear models can only solve problems that are linearly separable and it only applies to data set with low variance.

2) Non-linear models can be applied to the data set with high variance.

Here non-linear model is used for image captioning. This existing system uses classic Recurrent Neural Network (RNN) which has some disadvantages and may also generate inaccurate results.

A. Recurrent Neural Network

These are the networks which have loops in them. These allow the information to persist.

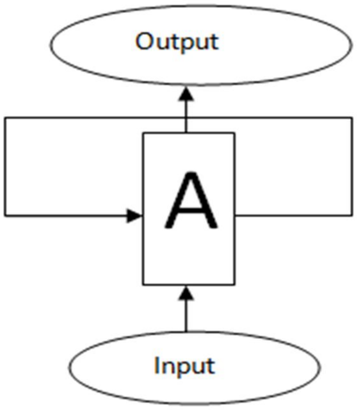


Fig. 3 Recurrent Neural Network

A loop allows information to be passed from one step of the network to the next. These loops make recurrent neural networks seem kind of mysterious. A recurrent neural network can be thought of as multiple copies of the same work, each passing a message to a successor. Consider what happens when we unroll the loop:

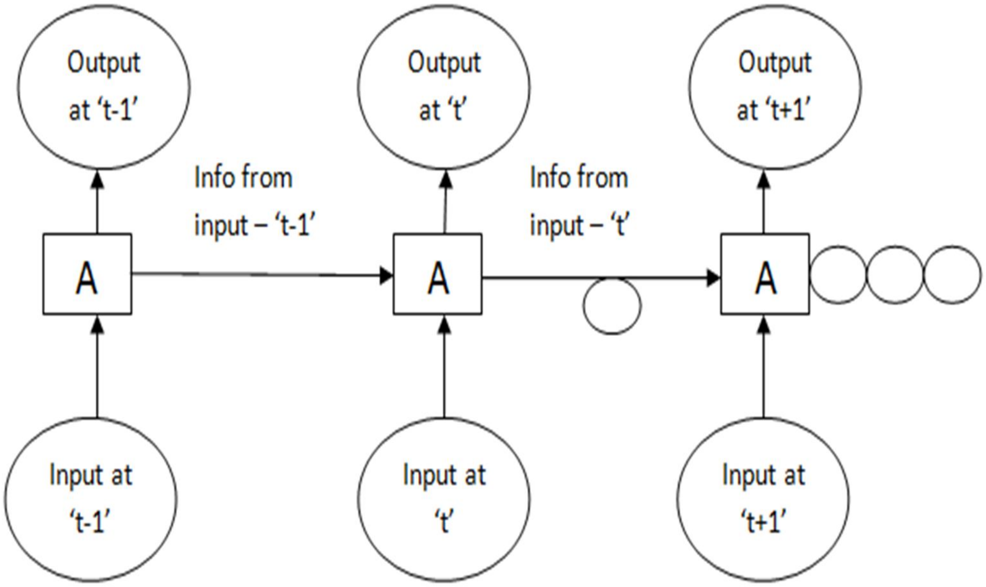


Fig. 4 Unrolled Recurrent Neural Network

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They are the natural architecture of neural network to use for such data.

And they certainly are used! In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modelling, translation, image captioning etc.

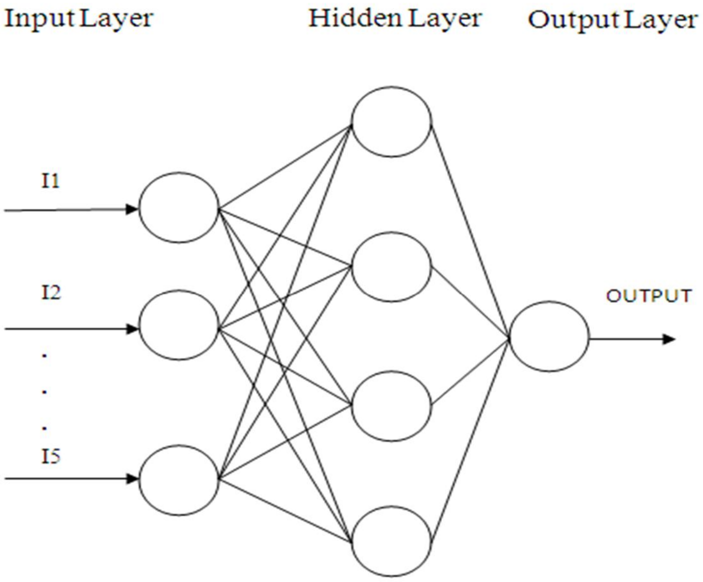


Fig. 5 Neural Network

IV. PROPOSED SYSTEM

To overcome the various problems of classic Recurrent Neural Network, recent major improvement in Recurrent Neural Network gave rise to the popularity of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) which helps to generate accurate results.

A. Convolutional Neural Network

These are specialized deep neural networks which can process the data that has input shape like a 2D matrix. Images are easily represented as a 2D matrix and CNN is very useful in working with images.
CNN is basically used for image classification and identifying if an image is a bird, a plane or Superman, etc.

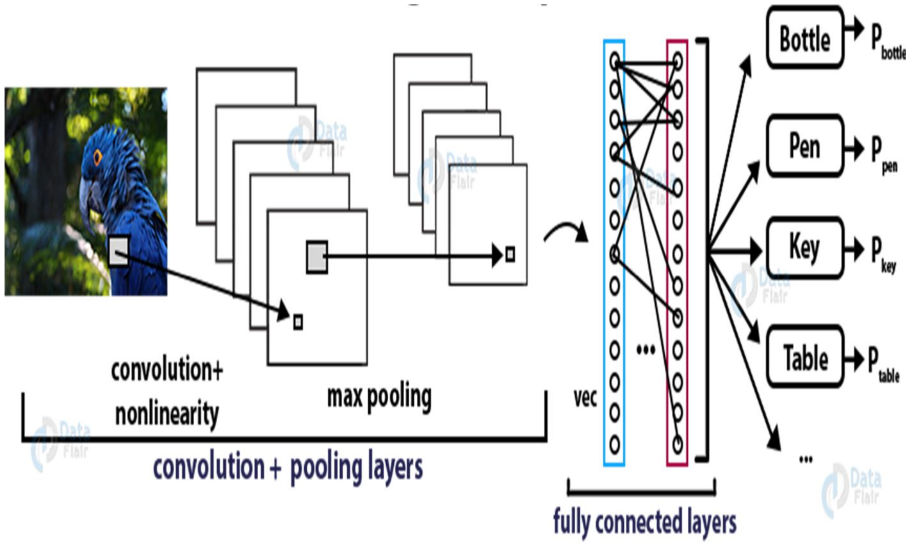


Fig. 6 Working of Deep CNN

It scans images from left to right and top to bottom to pull out important features from the images and combines the feature to classify images. It can handle the images that have been translated, rotated, scaled and changes in perspective.

B. LSTM Networks

These are special kind of RNN, which is well suited for sequence prediction problems. Based on the previous text, we can predict what the next word will be. It has proven itself effective from the traditional RNN by overcoming the limitations of RNN which had short term memory. LSTM is capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

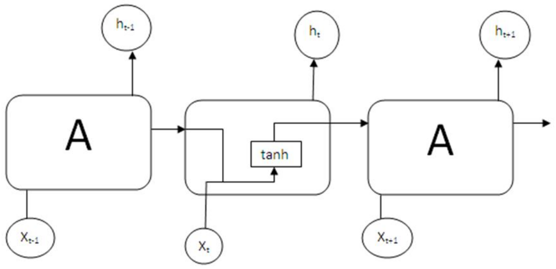


Fig. 7 Simple RNN

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting layers in a very special way.

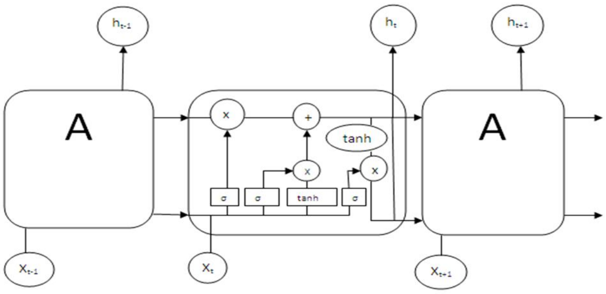


Fig. 8 LSTM and RNN

C. LSTM Architecture

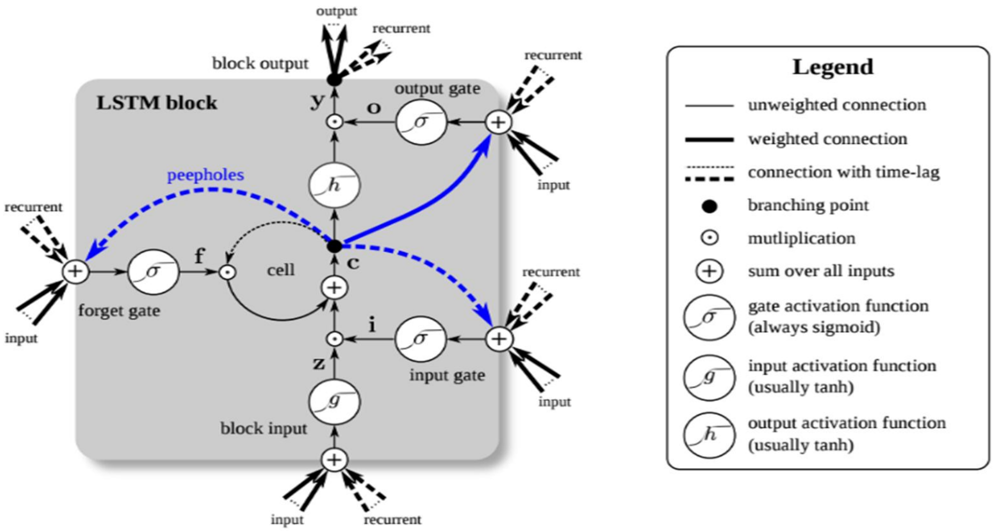


Fig. 9 Block Diagram of LSTM

D. Step-By-Step LSTM Walk Through

The first step in our LSTM is to decide what information we are going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer". It looks at h_{t-1} and x_t and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. Here 1 represents "completely keep this" while a 0 represents "completely get rid of this".

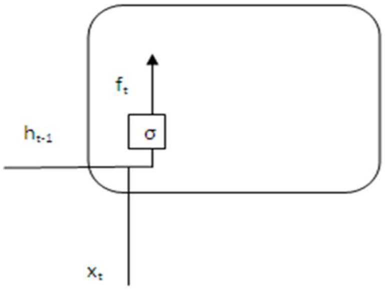


Fig. 10 Forget Gate Layer

The next step is to decide what new information we are going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we will update. Next, a tanh layer creates a vector of new candidate values, $\bar{c}_t$ that could be added to the state. In the next step, we'll combine these two to create an update to the state.

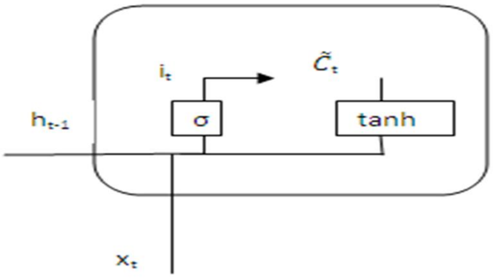


Fig. 11 Input Gate Layer and tanh Layer

It's now time to update the old cell state. C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it. We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * \bar{c}_t$. This is the new candidate values, scaled by how much we decided to update each state value.

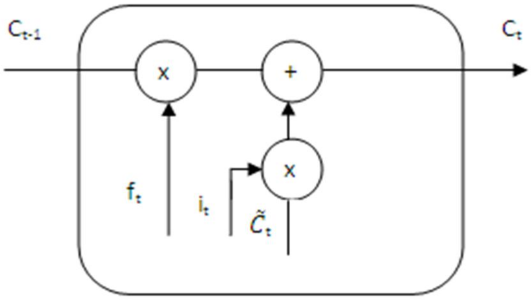


Fig. 12 Updating Each State Value

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

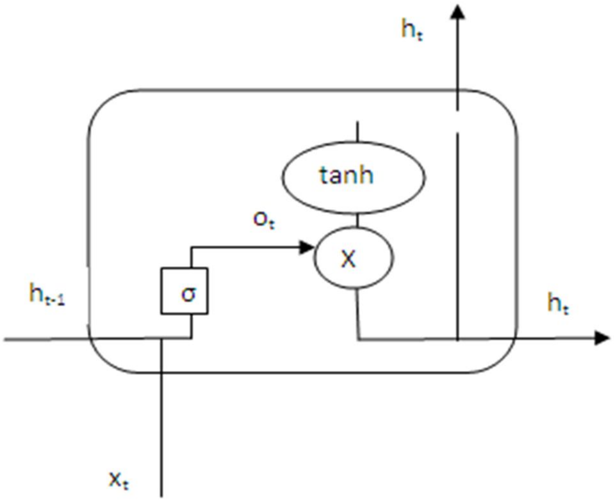


Fig. 13 Getting the Output

So, to make our image caption generator model, we will be merging these architectures. It is also called a CNN-RNN model.

1) CNN is used for extracting features from the image. We will use the pre-trained model Xception.

2) LSTM will use the information from CNN to help generate a description of the image.

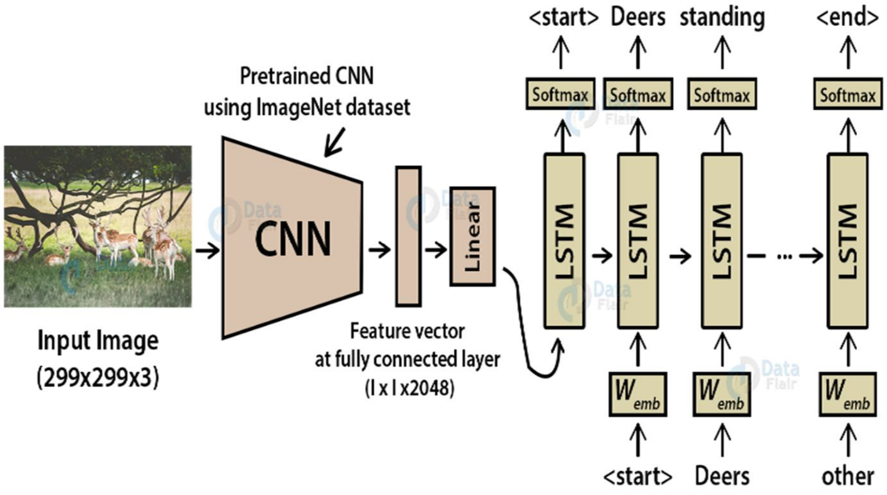


Fig. 14 Model-Image Caption Generator

V. IMPLEMENTATION

It is done using Keras and TensorFlow and many other libraries like NumPy, PIL etc.

We can summarize the construction of deep learning models in Keras as follows:

1) Define Your Model: Create a sequence and add layers.

2) Compile Your Model: Specify loss functions and optimizers.

3) Fit Your Model: Execute the model using data.

4) Make Predictions: Use the model to generate predictions on new data.

A. LSTM Layer

The core idea behind LSTMs is the cell state; it is kind of like a conveyor belt. The LSTM does have the ability to add or remove information to the cell state, carefully regulated by structure called gates. Gates are a way to optionally let information through. It is composed of a cell, input gate, an output gate and a forget gate. The cell is responsible for “remembering” values over arbitrary time intervals; Hence the word memory, in LSTM.

B. LSTM Input Layer

The LSTM input layer is specified by the “input_shape” argument on the first hidden layer of the network. The input to every LSTM layer must be 3D. The three dimensions of this input are:

1) Samples: One sequence is one sample. A batch is compared of one or more samples.
2) Time Steps: One time step is one point of observation in the sample.
3) Features: One feature is one observation at a time step.

This means that the input layer expects a 3D array of data when fitting the model and when making predictions, even if specific dimensions of the array contain a single value. e.g., one sample or one feature.

When defining the input layer of your LSTM network, the network assumes you have one or more samples and requires that you specify the number of time steps and number of features. We can do this by specifying a tuple to the “input_shape” argument.

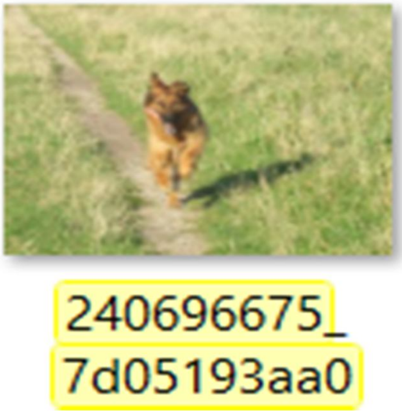
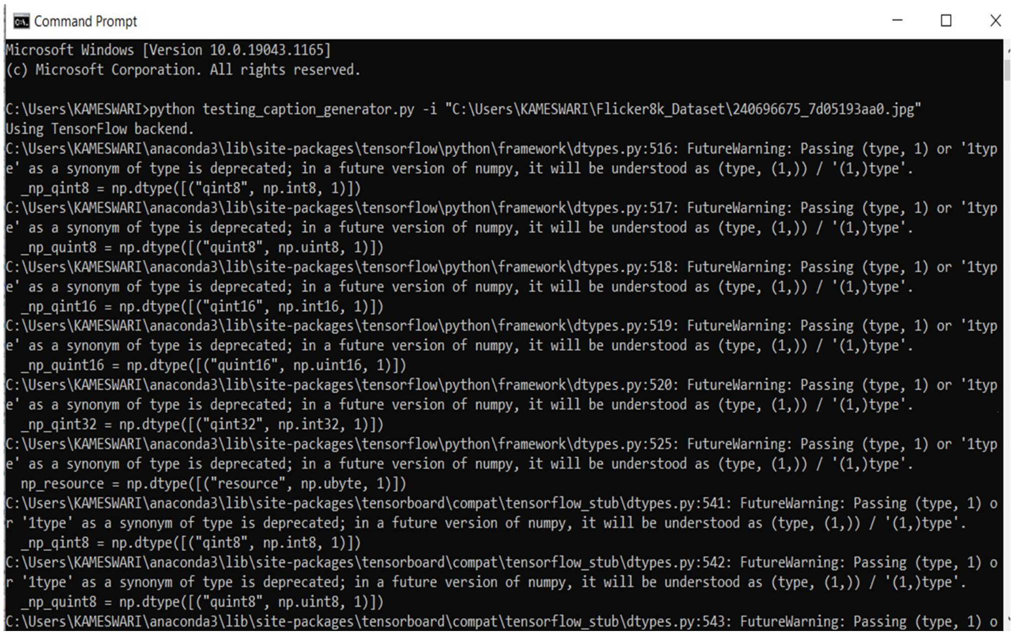
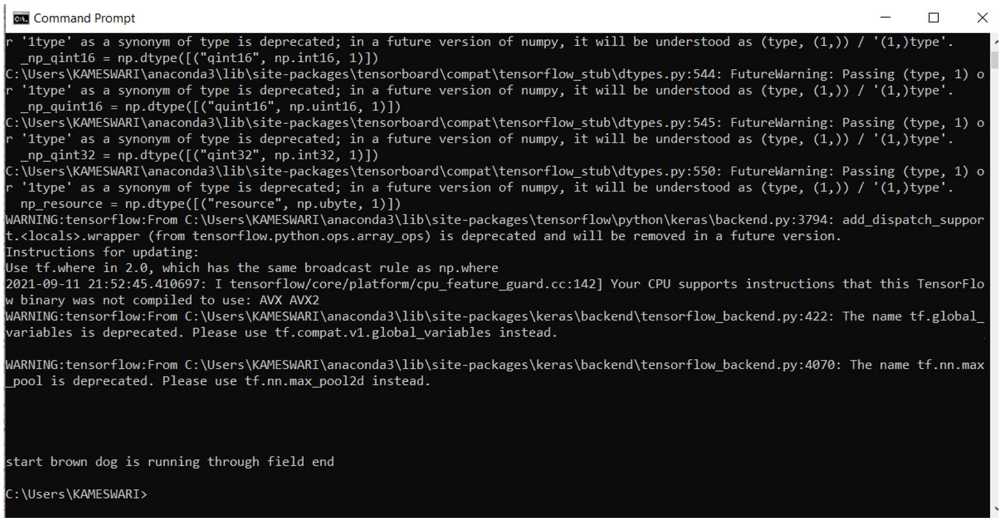


Fig. 15 Testing Image

VI. RESULTS



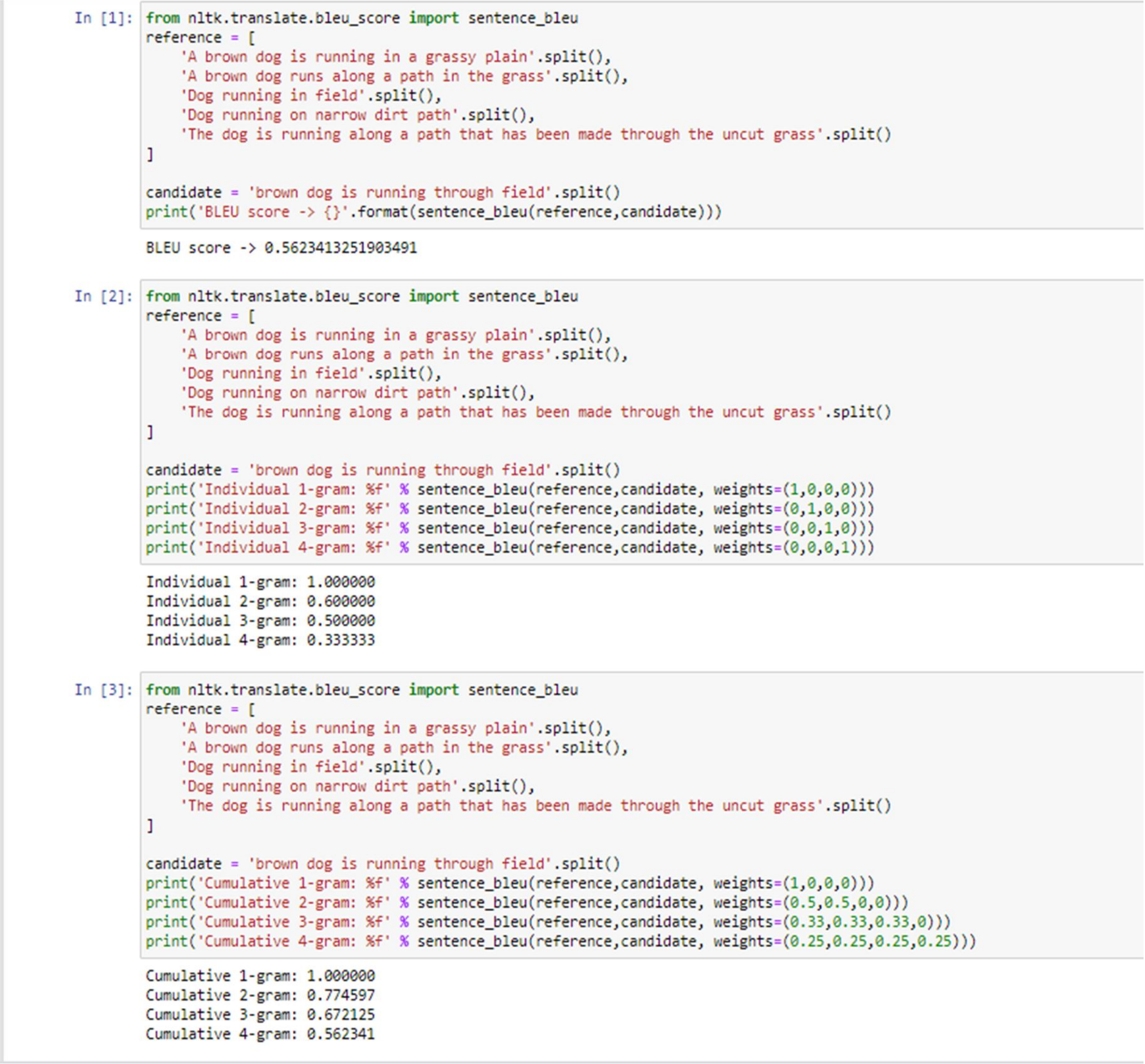
```
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\KAMESWARI>python testing_caption_generator.py -i "C:\Users\KAMESWARI\Flicker8k_Dataset\240696675_7d05193aa0.jpg"
Using TensorFlow backend.
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning: Passing (type, 1) o
r '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning: Passing (type, 1) o
r '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning: Passing (type, 1) o
```



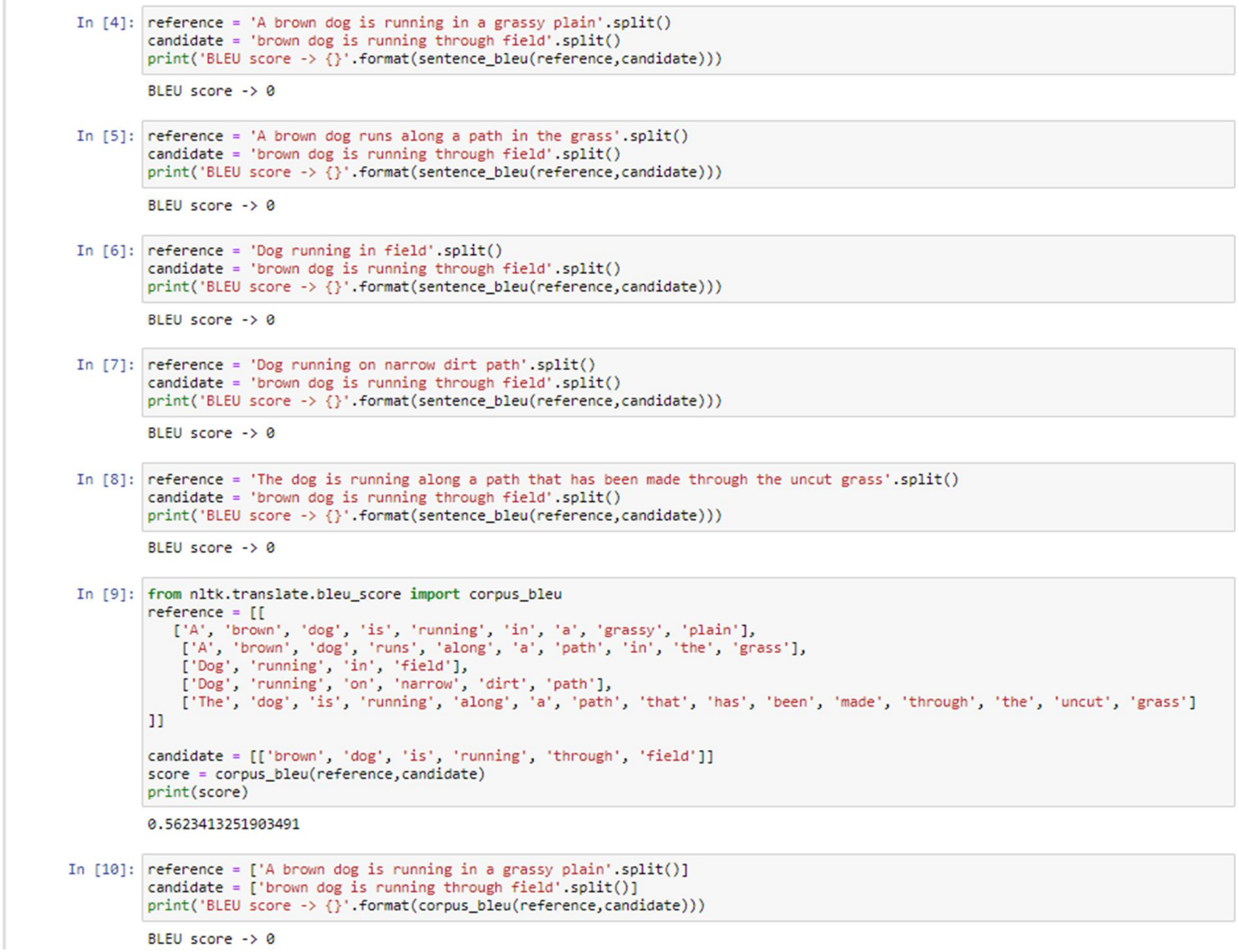
```
r '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning: Passing (type, 1) o
r '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning: Passing (type, 1) o
r '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning: Passing (type, 1) o
r '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
WARNING:tensorflow:From C:\Users\KAMESWARI\anaconda3\lib\site-packages\tensorflow\python\keras\backend.py:3794: add_dispatch_suppor
t.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
2021-09-11 21:52:45.410697: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlo
w binary was not compiled to use: AVX AVX2
WARNING:tensorflow:From C:\Users\KAMESWARI\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_
variables is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From C:\Users\KAMESWARI\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:4070: The name tf.nn.max
_pool is deprecated. Please use tf.nn.max_pool2d instead.



start brown dog is running through field end

C:\Users\KAMESWARI>
```

VII. EVALUATION

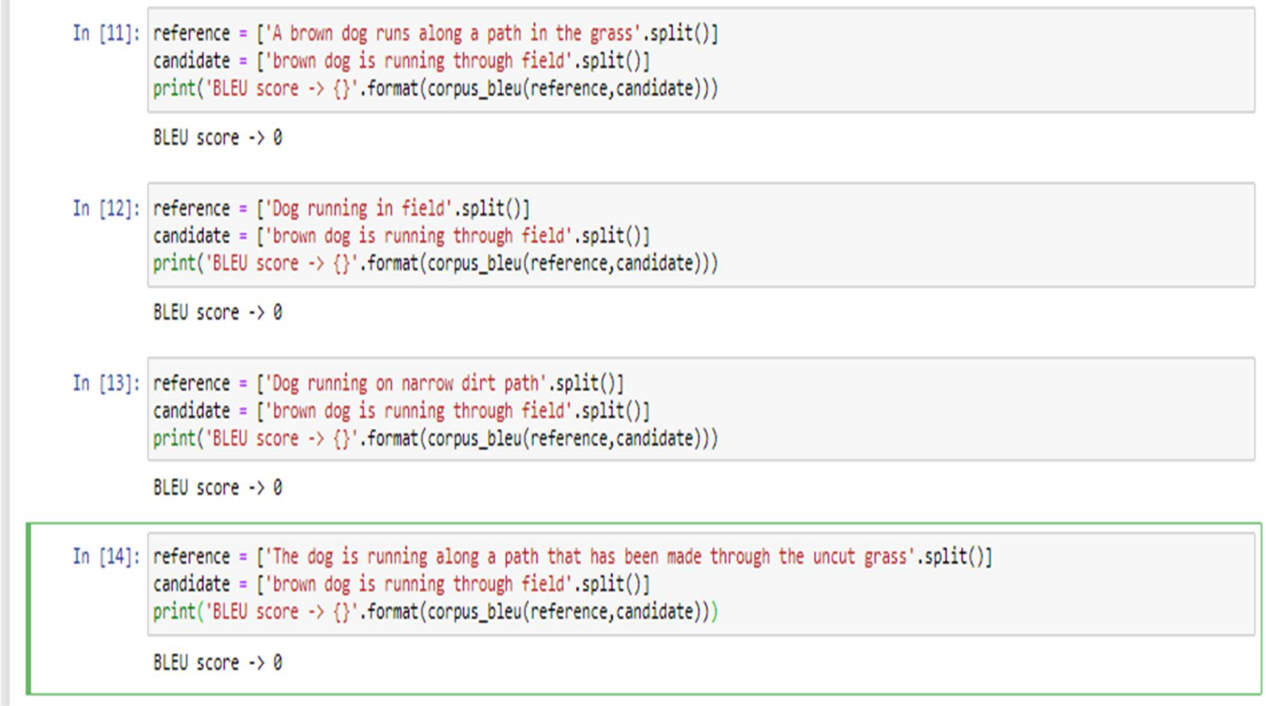
Evaluation is done using bleu_score. It can be done by using either sentence_bleu function or corpus_bleu function.



```
In [1]: from nltk.translate.bleu_score import sentence_bleu
        reference = [
            'A brown dog is running in a grassy plain'.split(),
            'A brown dog runs along a path in the grass'.split(),
            'Dog running in field'.split(),
            'Dog running on narrow dirt path'.split(),
            'The dog is running along a path that has been made through the uncut grass'.split()
        ]

        candidate = 'brown dog is running through field'.split()
        print('BLEU score -> {}'.format(sentence_bleu(reference,candidate)))

        BLEU score -> 0.5623413251903491

In [2]: from nltk.translate.bleu_score import sentence_bleu
        reference = [
            'A brown dog is running in a grassy plain'.split(),
            'A brown dog runs along a path in the grass'.split(),
            'Dog running in field'.split(),
            'Dog running on narrow dirt path'.split(),
            'The dog is running along a path that has been made through the uncut grass'.split()
        ]

        candidate = 'brown dog is running through field'.split()
        print('Individual 1-gram: %f' % sentence_bleu(reference,candidate, weights=(1,0,0,0)))
        print('Individual 2-gram: %f' % sentence_bleu(reference,candidate, weights=(0,1,0,0)))
        print('Individual 3-gram: %f' % sentence_bleu(reference,candidate, weights=(0,0,1,0)))
        print('Individual 4-gram: %f' % sentence_bleu(reference,candidate, weights=(0,0,0,1)))

        Individual 1-gram: 1.000000
        Individual 2-gram: 0.600000
        Individual 3-gram: 0.500000
        Individual 4-gram: 0.333333

In [3]: from nltk.translate.bleu_score import sentence_bleu
        reference = [
            'A brown dog is running in a grassy plain'.split(),
            'A brown dog runs along a path in the grass'.split(),
            'Dog running in field'.split(),
            'Dog running on narrow dirt path'.split(),
            'The dog is running along a path that has been made through the uncut grass'.split()
        ]

        candidate = 'brown dog is running through field'.split()
        print('Cumulative 1-gram: %f' % sentence_bleu(reference,candidate, weights=(1,0,0,0)))
        print('Cumulative 2-gram: %f' % sentence_bleu(reference,candidate, weights=(0.5,0.5,0,0)))
        print('Cumulative 3-gram: %f' % sentence_bleu(reference,candidate, weights=(0.33,0.33,0.33,0)))
        print('Cumulative 4-gram: %f' % sentence_bleu(reference,candidate, weights=(0.25,0.25,0.25,0.25)))

        Cumulative 1-gram: 1.000000
        Cumulative 2-gram: 0.774597
        Cumulative 3-gram: 0.672125
        Cumulative 4-gram: 0.562341
```



```
In [4]: reference = 'A brown dog is running in a grassy plain'.split()
        candidate = 'brown dog is running through field'.split()
        print('BLEU score -> {}'.format(sentence_bleu(reference,candidate)))

        BLEU score -> 0

In [5]: reference = 'A brown dog runs along a path in the grass'.split()
        candidate = 'brown dog is running through field'.split()
        print('BLEU score -> {}'.format(sentence_bleu(reference,candidate)))

        BLEU score -> 0

In [6]: reference = 'Dog running in field'.split()
        candidate = 'brown dog is running through field'.split()
        print('BLEU score -> {}'.format(sentence_bleu(reference,candidate)))

        BLEU score -> 0

In [7]: reference = 'Dog running on narrow dirt path'.split()
        candidate = 'brown dog is running through field'.split()
        print('BLEU score -> {}'.format(sentence_bleu(reference,candidate)))

        BLEU score -> 0

In [8]: reference = 'The dog is running along a path that has been made through the uncut grass'.split()
        candidate = 'brown dog is running through field'.split()
        print('BLEU score -> {}'.format(sentence_bleu(reference,candidate)))

        BLEU score -> 0

In [9]: from nltk.translate.bleu_score import corpus_bleu
        reference = [[
           ['A', 'brown', 'dog', 'is', 'running', 'in', 'a', 'grassy', 'plain'],
            ['A', 'brown', 'dog', 'runs', 'along', 'a', 'path', 'in', 'the', 'grass'],
            ['Dog', 'running', 'in', 'field'],
            ['Dog', 'running', 'on', 'narrow', 'dirt', 'path'],
            ['The', 'dog', 'is', 'running', 'along', 'a', 'path', 'that', 'has', 'been', 'made', 'through', 'the', 'uncut', 'grass']
        ]]

        candidate = [['brown', 'dog', 'is', 'running', 'through', 'field']]
        score = corpus_bleu(reference,candidate)
        print(score)

        0.5623413251903491

In [10]: reference = ['A brown dog is running in a grassy plain'.split()]
         candidate = ['brown dog is running through field'.split()]
         print('BLEU score -> {}'.format(corpus_bleu(reference,candidate)))

         BLEU score -> 0
```



```
In [11]: reference = ['A brown dog runs along a path in the grass'.split()]
         candidate = ['brown dog is running through field'.split()]
         print('BLEU score -> {}'.format(corpus_bleu(reference,candidate)))

         BLEU score -> 0

In [12]: reference = ['Dog running in field'.split()]
         candidate = ['brown dog is running through field'.split()]
         print('BLEU score -> {}'.format(corpus_bleu(reference,candidate)))

         BLEU score -> 0

In [13]: reference = ['Dog running on narrow dirt path'.split()]
         candidate = ['brown dog is running through field'.split()]
         print('BLEU score -> {}'.format(corpus_bleu(reference,candidate)))

         BLEU score -> 0

In [14]: reference = ['The dog is running along a path that has been made through the uncut grass'.split()]
         candidate = ['brown dog is running through field'.split()]
         print('BLEU score -> {}'.format(corpus_bleu(reference,candidate)))

         BLEU score -> 0
```

VIII. CONCLUSION

The main motto of this paper is to predict captions of the image by using convolutional neural network and Long Short-Term Memory Algorithm. In this, we have predicted the captions for the images of Flickr_8k dataset.

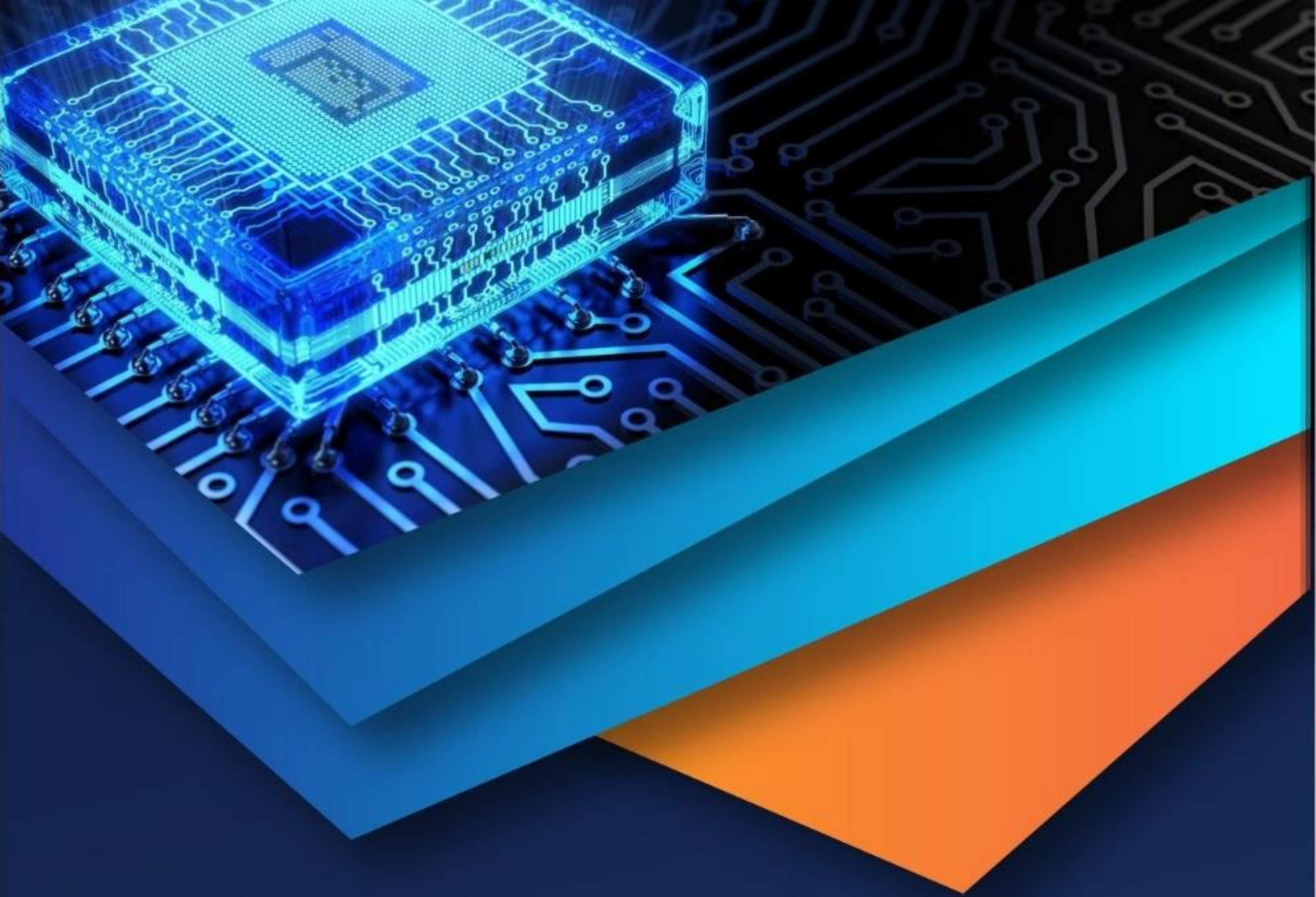
This helps the user especially visually impaired people to better understand the content of images.

IX. FUTURE ENHANCEMENT

In this the image caption generation is done using Flickr_8k dataset. But this can also be done using Flickr_30k dataset which has nearly 30,000 images. This has text file which consists of above 1,58,000 captions. This provides much more accuracy than Flickr_8k dataset as it has much more data.

REFERENCES

[1] https://data-flair.training/blogs/deep-learning-project-ideas/
[2] https://data-flair.training/blogs/python-based-project-image-caption-generator-cnn/
[3] https://www.analyticsvidhya.com/blog/2020/11/create-your-own-image-caption-generator-using-keras/
[4] https://blog.clairvoyantsoft.com/image-caption-generator-535b8e9a66achttps://blog.clairvoyantsoft.com/image-caption-generator-535b8e9a66ac
[5] https://machinelearningmastery.com/calculate-bleu-score-for-text-python/



10.22214/IJRASET

45.98

IMPACT FACTOR:
7.129

IMPACT FACTOR:
7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY