International Journal of Future Generation Communication and Networking
Vol.14, No. 1, (2021), pp. 4161 – 4172

# Image Annotation Generation using Deep-Learning

**Piyush Navarkar[1] P.G.Chilveri[2] ,TarunNambiar[3], MeghrajNale[4], A. B. Chandgude[5]**

[1,2,3,4,5]*Department of E&TC Engineering, SKNCOE, SPPU, Pune, India*
[1]*piyushnavarkar@gmail.com*
[2]*pgchilveri.skncoe@sinhgad.edu*
[3]*tarunnamb@gmail.com*
[4]*meghrajnale642@gmail.com*
[5]*akshay_chandgude.skncoe@sinhgad.edu*

***Abstract –***

*The paper tends to aim at creating automated captions by learning the contents of the image. Nowadays images are interpreted with human interference and it becomes a nearly unbearable task for enormous commercial records. The image records are given as input to a Convolutional Neural Network (CNN)) encoder for generating "thought vector" which extracts the features and enhances objects out of our image and RNN decoder (as a "Long Short-Term Memory") is used to translate the features given by our image to obtain a sequential, meaningful description of the image. In this paper, we are going to explain the survey about image captioning and our proposed system.*

***Keywords:*** *Image annotation, Deep learning, CNN, LSTM, Python, etc.*

## I. INTRODUCTION

The mechanism by which a computer system automatically assigns metadata to a digital image in the form of captioning or keywords is known as automatic image annotation (also known as automatic image tagging or linguistic indexing). Image retrieval systems use this application of computer vision techniques to arrange and locate images of interest from a database. This approach is a multi-class picture classification with a large number of groups - as large as the vocabulary size. Machine learning techniques typically use image analysis in the form of extracted feature vectors and training annotation terms to attempt to automatically apply annotations to new photos. The first methods learned the relationships between image features and training annotations, and then machine translation techniques were developed to try to convert textual vocabulary into "digital vocabulary," or clustered regions known as blobs. Following these efforts, work on classification methods, relevance models, and so on has been done. The benefits of automated image annotation over content-based image retrieval (CBIR) include the ability for the user to specify queries more naturally. CBIR currently allows users to check for example queries or by image definitions such as color and texture. Certain image features in example images may take precedence over the idea that the user is really interested in. Traditional image retrieval methods, such as those used by libraries, have relied on manually annotated images, which is costly and time-consuming, particularly given the vast and ever-growing image databases. To shape a full image definition approach, we will incorporate techniques from both computer vision and natural language processing in this paper. This will be in charge of creating computer-generated natural descriptions of any

images provided. The idea is to use a deep convolutional neural network (CNN) trained to classify objects in images to replace the encoder (RNN layer) in an encoder-decoder architecture. The SoftMax layer is normally the last layer in a CNN, and it assigns a probability to each object in the picture. We can feed the CNN's rich encoding of the image into the decoder (language generation RNN) designed to generate phrases if we remove the SoftMax layer from CNN. We can then train the entire system directly on images and captions, ensuring that the descriptions it generates are as close as possible to the training descriptions for each image. A dense feature vector can be generated using a convolutional neural network. This dense vector, also known as an embedding, can be fed into other algorithms or networks as a function input. This embedding becomes a dense representation of the image for an image caption model, and it will be used as the LSTM's initial state. An LSTM is a type of recurrent neural network that is frequently used to solve problems with temporal dependencies. In its memory cell state, it is able to capture information about previous states in order to help inform the current prediction. A forget gate, input gate, and output gate are the three key components of an LSTM. Each of these gates is in charge of modifying changes to the memory state of the cell.

## II. RELATED WORK

[1] Vinyals et al. proposed a method called Neural Image Caption Generator (NIC). The method uses a CNN for image representations and an LSTM for generating image captions. This special CNN uses a novel method for batch normalization and the output of the last hidden layer of CNN is used as an input to the LSTM decoder. This LSTM is capable of keeping track of the objects that already have been described using text. NIC is trained based on maximum likelihood estimation. In generating image captions, image information is included to the initial state of an LSTM. The next words are generated based on the current time step and the previous hidden state. This process continues until it gets the end token of the sentence. From this paper author showed BLEU-1 score improvements on Flickr30k, from 56 to 66, and on SBU, from 19 to 28. Lastly, on the newly released COCO dataset, they achieve a BLEU-4 of 27.7, which is the current state-of-the-art.[2] In this paper Venkatesh N. Murthy et al. proposed a simple and effective model for the image annotations. Their first set of models is based on the Canonical Correlation Analysis (CCA) paradigm, which aids in the modelling of both visual (CNN) and textual (word embedding vectors) features of the data. In the proposed model, the author combined a function (representing an image) from a Convolutional Neural Network (CNN) with word embedding vectors (representing the tags associated with it). The term embedding vector for a tag is extracted using a pre-trained skip-gram architecture, and these CNN features are extracted for images using a pre-trained VGG-16 architecture (word2vec). In word embedding concept they represent the tag (word) by a 300-dimensional real valued feature vector using a Word2Vec tool and they call it as a word embedding vector. In the author's point of view word embedding vectors (W2V) work better than binary vectors (BV) with best performing model CCA-KNN. This suggests that word embedding vectors provide a better representation for words than binary form representations.[3] In this paper Sun Chengjiang et al. proposed a multimodal deep learning framework, which aims to optimally integrate multiple deep neural networks pre-trained with convolutional neural network. The proposed framework explores two-stage learning scheme that consists of learning to fine-tune the parameters of deep neural network and learning to find the optimal combination of diverse modalities. The proposed model is the best performer, beating other approaches which are Lazy learning-based approach (LL), Deep representations and codes-based

approach (DRC) by a statistically significant margin in Hamming Loss and validating the efficacy of learning effective similarity functions on multi-modal data annotation.

[4] Fang et al. introduced generation-based image captioning. It uses visual detectors, a language model, and a multimodal similarity model to train the model on an image captioning dataset. Image captions can contain nouns, verbs, and adjectives. A vocabulary is formed using 1000 most common words from the training captions. The system works with the image sub-regions rather that the full image. Convolutional neural networks (both AlexNet and VGG16Net) are used for extracting features for the sub-regions of an image.[5] In this paper Kyunghyun Cho et al. described system for a variety of tasks: machine translation, image caption generation, video clip description and speech recognition. All these systems are based on set of building blocks: gated recurrent neural networks and convolutional neural networks, along with trained attention mechanisms. In this paper authors focused on a general approach to the alignment problem known as the soft attention mechanism.[6] Chen et al. proposed an attention-based image captioning method. To compute an attention diagram, this approach considers both spatial and channel wise attentions. For the generation of an attention map, current attention-based image captioning methods only consider spatial details. The fact that these spatial attention methods compute weighted pooling only on attentive function maps is a common downside. The CNN with Spatial and Channel-wise Attentions (SCA-CNN) outperforms the others. This is because SCA-CNN takes into account spatial, channel-wise, and multi-layer attentions, while most other attention models only consider one.[7] In this paper Jingqiang Chen et al. proposed a model for news image captioning. The author of this project proposes a news image captioning approach based on the attentional encoder-decoder model and a multi-modal attention system to attend to both image and text at the same time. For image extraction, the author used CNN, and for caption encoding and decoding, he used RNN models. General captions do not provide as much details as news image captions do, such as entity names and incidents. Experiments show that the author's approach outperforms the current state-of-the-art generic image captioning method, which only takes the news image into account.[8] The attention-based model that automatically learns to define the content of images is discussed in this paper by the author. Kelvin Xu et al. used convolutional features and a decoder with backpropagation techniques to demonstrate that the model can learn to produce the corresponding words in output sequence automatically. The authors received some incorrect captions for the first collection of photographs, but he was able to produce proper captions for the images using the attention model. Kelvin Xu et al. demonstrate how learned attention can be used to improve the model generation process' interpretability. [9] In this paper Junhua Mao et al. talks about the Multimodal Recurrent Neural Network model for generating novel image captions. The model consists of two sub networks a deep recurrent neural network for sentences and a deep convolutional network for images. This project can be termed as bilateral as the RNN model helps to retrieve images or sentences and achieves significant performance improvement over the state of art methods.[10] In this paper Junhua Mao et al. focuses on special case of text generation where the goal is to generate a text description that applies to a certain region or one object in an image. Junhua Mao et.al have used the state-of-art deep learning using CNN and RNN. To obtain the text description for a region in an image the authors have encoded a relative location and size of the region using a 5-dimensional vector. The experiment shows that training of listener must correctly decode a generated description which can outclass a model which simply emits captions based on region features.[11] Jin et al. proposed attention-based image captioning method. Based on the semantic relationship between visual and textual content, this method can extract the flow of abstract meaning. It can also gather useful data by presenting a scene-specific context.[12] The author of this paper proposed a method for generating captions that only uses Convolutional Neural

Networks (CNNs). According to the authors, the model is 3 times faster than NIC (an LSTM-based model) during training time and produces better performance. They also tested the model on the paragraph annotation dataset and found it to have a higher CIDEr score than hierarchical LSTMs. It's worth noting that the NIC model has around 6.7 million parameters, while this proposed model has over 16.0 million.[13] The bi-directional mapping between images and their sentence-based explanations was investigated in this paper. This model can both generate novel captions from an image and recreate visual features from a description of the image. They listed a variety of tasks, including sentence generation, retrieval of sentences, and image retrieval. To create this model, they used open source RNN code and the Caffe system.

## III. METHODOLOGY

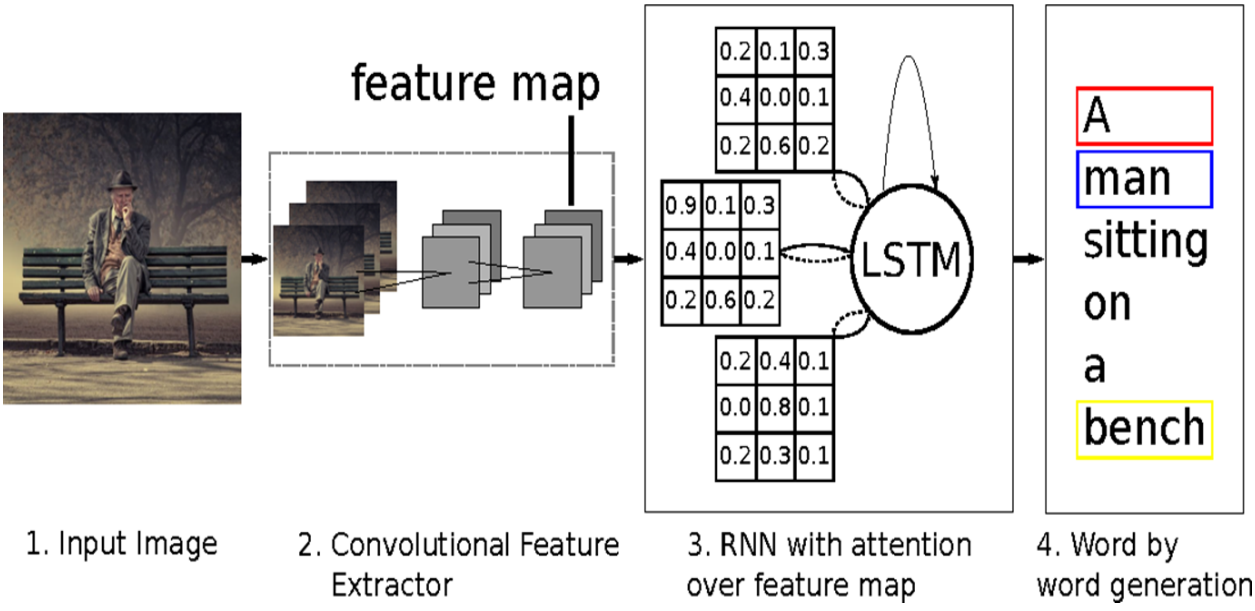


Fig1: Proposed System

The goal of this work is to create a short caption that describes the image's material, scene relationships, and defined objects. The system employs an attention mechanism that focuses on the desired region

(ROI). After feature extraction in the encoder, it is used to assign weights to the identified items, with the ROI receiving more weight. This is then fed into the decoder, which decodes the objects based on the weights assigned and generates the appropriate caption using the language model. The Inception V3 model is used to speed up the training process since it is pre-trained on ImageNet, making it easier to identify the artefacts. After hyper parameter tuning, a few layers from the Inception v3 model can be extracted and used to train the base model. To focus and reassign the weights to the region of interest, the visual attention mechanism is used. This can be tweaked to achieve the desired result. The visual attention system removes all visual words and uses a visual mask to give them more weight. The language model is fed with the extracted visual words and feature similarity is used. The focus layer is constrained by visual terms. The caption is produced by the decoder using LSTM (Long short-term memory) as the language model. Words are sampled and output at time t to produce a caption. The beam search technique is used, in which a predetermined number of best-by-now sentences is computed and extended with new terms. The beam size for this experiment is 10. Greedy scan is also used, with the beam size set to one.

## IV. System Working and Specifications

1) **Pre-Trained Inception v3 -**

Inception v3 may be a widely-used image recognition model that has been shown to achieve greater than 78.1% accuracy on the Image Net dataset. The model itself is formed from symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batch norm is employed extensively throughout the model and applied to activation inputs. Loss is computed via SoftMax.

2) **CNN (Convolution Neural Network)** –

The Convolutional Neural Network, also known as CNN or ConvNet, is a category of specialized neural networks by processing grid-like data topology, as pictured. Convolutional Neural Networks is designed to perform map image data to output. CNN's hidden features have the ability to extract elements from an image.

3) **LSTM (Long Short-Term Memory)**–

LSTM networks are a kind of RNN that uses special units additionally to plain units. LSTM units include a ‘memory cell’ which will maintain information in memory for long periods of time. This architecture

lets them learn longer term dependencies. this is often one among the implementations of LSTM cells, many other architectures exist.

4)**Flickr8k_Dataset-**

It contains 8092 images with different size sand colors. Out of total8092, we have used 6000 pictures to train the model, and 1000 pictures are used for development, and the remaining 1k images are used for testing the model.

5)**Flickr8k_text-**
This text file describes which images are used as Training and Test data. It contains aFlickr8k.token.txtwhich has five captions per image for training the model stored in theform of key-value pair where the keyis the Image id, and the value is the list ofcaptions.

## V. ALGORITHM IMPLEMENTED

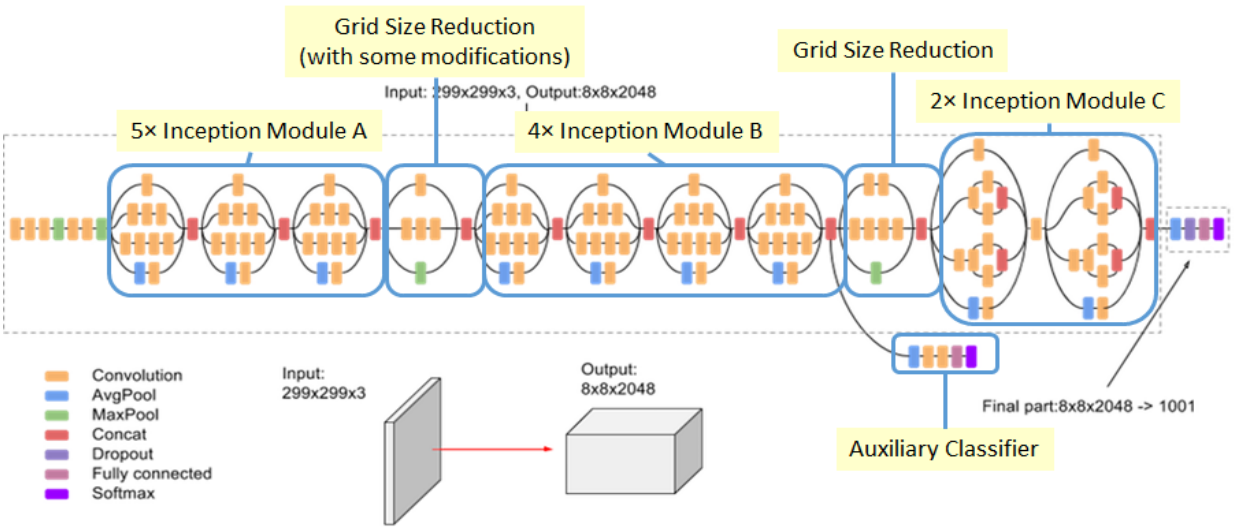


**Fig. 2** Inception v3 Model

- **CNN Encoder**

  Step 1: Dataset containing images along with reference caption is fed into the system

  Step 2: The convolutional neural network is used a encoder which extracts image features f pixel by pixel.

  Step 3: Matrix factorization is performed on the extracted pixels. The matrix is of m x n.

  Step 4: Max pooling is performed on this matrix where maximum value is selected and again fixed into matrix.

  Step 5: Normalization is performed where every negative value is converted to zero.

  Step 6: To convert values to zero rectified linear units are used where each value is filtered and negative value is set to zero.

  Step 7: The hidden layers take the input values from the visible layers and assign the weights after calculating maximum probability.

- **Bottom-up Attention**

  Step 1: Extracting features from stimuli Sf.

  Step 2: The stimuli features are assigned more weightage in the hidden layers.

  Step 3: Feature similarity is calculated by comparing the visual similar features.

  Step 4: The weights are reassigned after maximum similarity value is calculated.

- **LSTM Language Model**

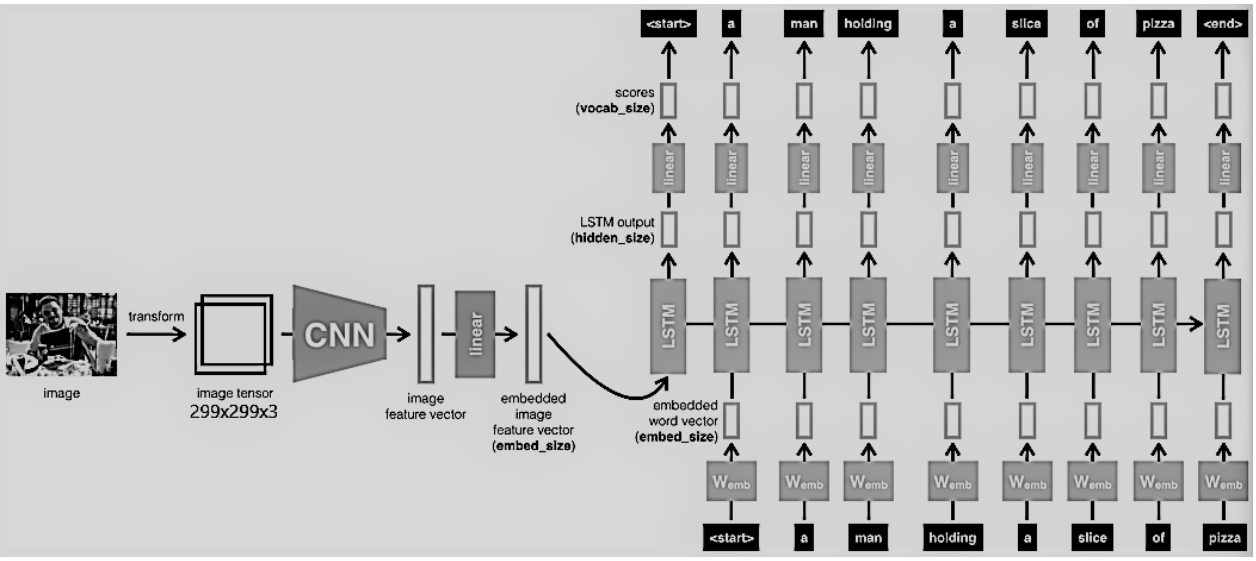


**Fig. 3** LSTM Model

Step 1: Given any sentence P the probability of forming a new sentence y is calculated (y0, y1.yt-1).

Step 2: Tokenize the reference captions from the dataset used.

Step 3: Tokens used are startseq to indicate the start of the sequence words or sentence and endseq to indicate end.

Step 4; Form a vocabulary and map these tokens to one-hot vectors.

Step 5: Remove and replace all the special characters and alphanumeric.

Step 6: All the uncommon words are given unk token.

Step 7: Set input xt = yt-1.

Step 8: Activation function is used to make a soft-max prediction to predict the next word y1.

Step 9: Soft-max calculates the maximum probability of the next word in sequence from the vocabulary.

## VII. RESULTS

```
In [21]: # Call the funtion to encode all the train images
         # This will take a while on CPU - Execute this only once
         start = time()
         encoding_train = {}
         for img in train_img:
             encoding_train[img[len(images):]] = encode(img)
         print("Time taken in seconds =", time()-start)

         WARNING:tensorflow:From C:\ProgramData\Anaconda3\envs\DeepML\lib\site-packages\keras\backend\tensorflow_backend.py:422: The nam
         e tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

         Time taken in seconds = 3989.9589664936066
```

**Fig. 4** This is the part of the code which shows the time required to train the dataset for our model and the times varies it depends on the machine we use it has to be executed only once.

```
In [47]: for i in range(epochs):
             generator = data_generator(train_descriptions, train_features, wordtoix, max_length, number_pics_per_bath)
             model.fit_generator(generator, epochs=epochs, steps_per_epoch=steps, verbose=1)
             model.save('./model_weights/model_' + str(i) + '.h5')

         Epoch 1/10
         1000/1000 [==============================] - 934s 934ms/step - loss: 4.2326
         Epoch 2/10
         1000/1000 [==============================] - 927s 927ms/step - loss: 3.4814
         Epoch 3/10
         1000/1000 [==============================] - 897s 897ms/step - loss: 3.2397
         Epoch 4/10
         1000/1000 [==============================] - 885s 885ms/step - loss: 3.0966
         Epoch 5/10
         1000/1000 [==============================] - 907s 907ms/step - loss: 2.9952
         Epoch 6/10
         1000/1000 [==============================] - 940s 940ms/step - loss: 2.9124
         Epoch 7/10
         1000/1000 [==============================] - 930s 930ms/step - loss: 2.8475
         Epoch 8/10
         1000/1000 [==============================] - 1010s 1s/step - loss: 2.7928
         Epoch 9/10
         1000/1000 [==============================] - 978s 978ms/step - loss: 2.7432
         Epoch 10/10
         1000/1000 [==============================] - 1007s 1s/step - loss: 2.7020
         Epoch 1/10
         1000/1000 [==============================] - 982s 982ms/step - loss: 2.6654
         Epoch 2/10
         1000/1000 [==============================] - 978s 978ms/step - loss: 2.6341
         Epoch 3/10
         1000/1000 [==============================] - 1118s 1s/step - loss: 2.6057
         Epoch 4/10
         1000/1000 [==============================] - 1097s 1s/step - loss: 2.5814
         Epoch 5/10
         1000/1000 [==============================] - 1036s 1s/step - loss: 2.5548
```

**Fig. 5** This is the part of the code which shows the time required to train the epochs for our model and the times varies it depends on the machine we use. This process generally takes about 10-11 Hrs. and it is used for training and increasing the accuracy of our model.

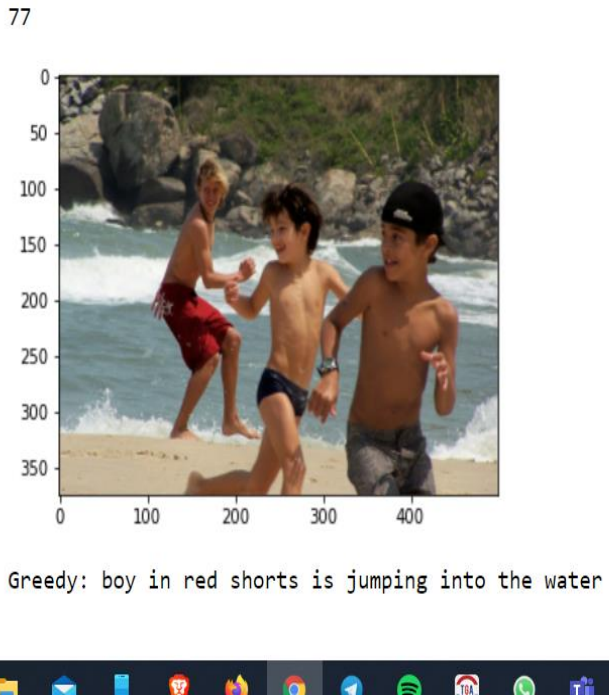


**Fig. 6** Caption generated for the image is showing a boy in red shorts jumping into water which is correct.

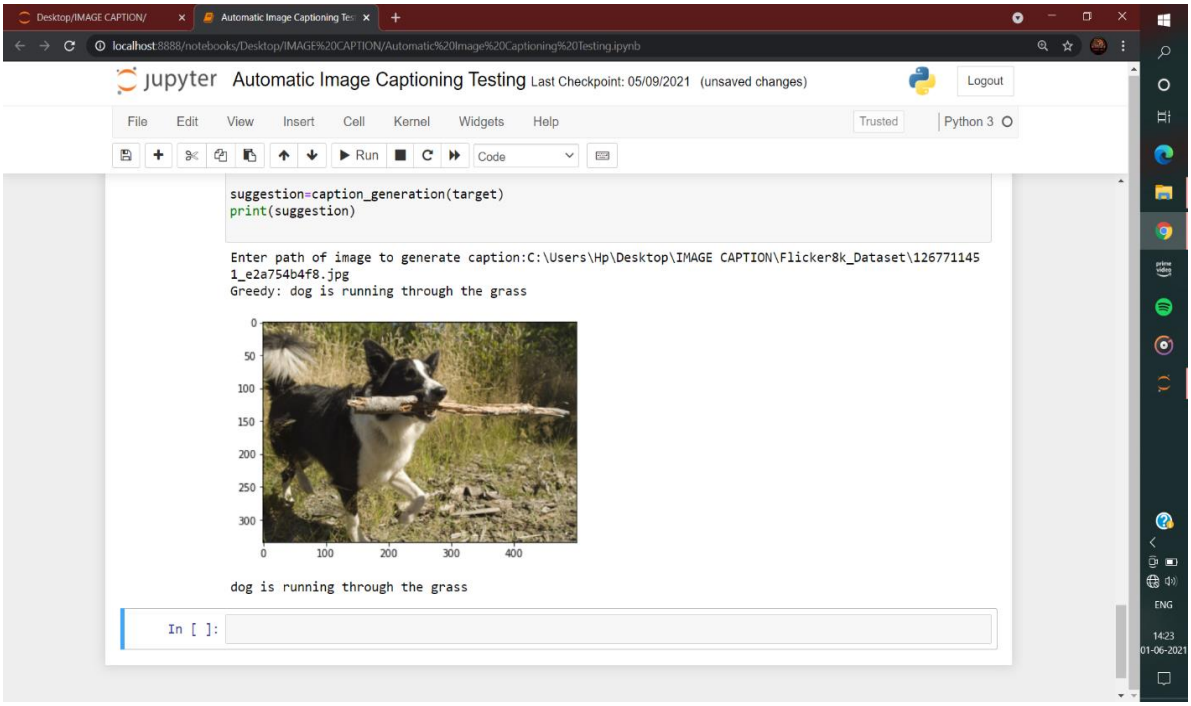

**Fig. 7** Caption generated for the image is showing the dog running through the grass but not able to point out the stick in his mouth.

## VIII. Conclusion:

Here the implemented system automatically generate annotations for given image. The programming language used is a python and algorithm used is CNN. The purpose of the model is to maximize the likelihood of the sentence given the image. So, using deep learning techniques it is possible to generate automatic image annotations. This model is end-to-end system neural capable of viewing given the image and generating a reasonable description that is syntactically as well as semantically accurate. This caption generation is on the basis of tokens considering the reference captions in the dataset assigned as train captions. Future scope of this work would be a modification can be applied in real life scenarios to tackle the real-world problems. The model has a convolutional neural network encoder and a LSTM decoder that will help in generation of sentences.

## IX. REFRENCES

[1] Vinyals, Oriol, et al. Show and tell: A neural image caption generator. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.

[2] Venkatesh N. Murthy, Subhransu Maji, R Manmatha, “Automatic Image Annotation using Deep learning representations”, ICMR '15 Proceedings of the 5th ACM on International Conference on Multimedia Retrieva.

[3] Oriol Vinyals, Alexander Toshev, SamyBengio, Dumitru Erhan, “Show and Tell: A Neural Image Caption Generator”, published 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

[4] Fang, Hao, et al. From captions to visual concepts and back. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.

[5] K. Cho, A. Courville, and Y. Bengio, describing multimedia content using attention-based encoder-decoder networks, IEEE Transactions on Multimedia, vol.17, no. 11, pp. 18751886,2015

[6] Long Chen, Hanwang Zhang, Jun Xiao, LiqiangNie, Jian Shao, and Tat-Seng Chua. 2017. SCA-CNN: Spatial and Channel-wise Attention in Convolutional Networks for Image Captioning. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 6298–6306.

[7] Jingqiang Chen, Hai Zhuge News Image Captioning Based On Text Summarization Using Image As Query.2 019

[8] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and YoshuaBengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning.

[9] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). In International Conference on Learning Representations (ICLR)

[10] Junhua Mao, Jonathan Huang, Alexander Toshev, OanaCamburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition. 11–20.

[11] JunqiJin, Kun Fu Runpeng Cui, Fei Sha Changshui Zhang Aligning where to see and what to tell: image caption with region-based attention and scene factorization 2015.

[12] Qingzhong Wang, Antoni B. Chan CNN+CNN: Convolutional Decoders for Image Captioning 2018.

[13] Xinlei Chen C. Lawrence Zitnick Mind's Eye: A Recurrent Visual Representation for Image Caption Generation 2015