IMAGE CAPTION GENERATOR USING DEEP LEARNING

Name of 1st Author -Peerzada Salman syeed , Name of 2nd Author –Dr.Mahmood Usman
Designation of 1st Author –Msc Student , Designation of 2nd Author –Assistant professor of Mathematics
Name of Department of 1st Author – Mathematics
Name of organization of 1st Author – Chandigarh University, Gharuan, Mohali City, -Chandigarh Country –India

Department of Mathematics ,Chandigarh University

ABSTRACT

As we are living in the 21st century, image caption is one of the most needed tools these days. This application has a built-in function for producing captions for a specific image. This application is made by the contribution of stable network bodies. The process of developing an image illustration is called a caption. It attains detection of the essentials, with their qualities, and the relationship between the things within the picture. Produces syntactically correct and semantically correct sentences. This paper presents an in-depth reading structure to explain images and create captions using AI vision and machine conveyance. This norm focus to discover the various elements seen in an image, see the interconnections between such objects, and produce legends. We utilized an 8K plus image data set and captions for this project. This content will also explain the programs and format of the various depolarized networks . The production of graphic captions is an integral part of Software Vision and Natural Linguistic motion . Photo subtitles manufacturers can look for properties in photo-sharing as social platforms are used, and further more , their use can be upgraded to moving pictures frames . They will indeed make it work for someone who has to translate images. Not to mention that it has an enormous capacity to help the visually impaired.

Keywords : **Introduction , problem statement, proposed methodology, convolutional neural networks , long short tem memory , evaluation ,result and analysis, conclusion**
DOI Number: 10.14704/NQ.2022.20.12.NQ77261 **NeuroQuantology2022;20(12): 2682-2691**

INTRODUCTION

Making the computer system discover and interpret itself using natural language processing (NLP) in the old Artificial Intelligence problem. This has been regarded as an impossible task by computer vision researchers to date. With the growing advancement in in-depth reading techniques, the availability of large databases, and the ability to integrate, models are often developed to produce image captions. Image captioning is a process that involves image processing and natural language processing concepts to identify the context of an image and interpret it in a natural language such as English or any other language. Image caption generator incorporates the concept of reconfiguring an image and interpreting it in the native language such as English. Our project is to learn the CNN and LSTM model concepts and build an effective Image caption generator model using CNN via LSTM. For this Python project, we will be using a caption generator using CNN (Convolutional Neural Networks) and LSTM (Short-term memory). Image elements will be released on VGG16 which is a CNN model trained in image databases and we will add features with the LSTM model that will be responsible for producing photo captions. Dataset for Python-based Project for generating captions, we will be using an 8K database. There are also other big data sets such as the 30K website and MSCOCO but it can take weeks to train the network so we will be using the smaller Flickr8k website. The advantage of a large database is that we can build better models. In-depth reading has attracted a lot of attention because it is a very good kind of reading that has the potential to be very useful in realworld programs. The ability to access information without labeled or unstructured data is of great benefit to those who want to know about real-world applications.

PROBLEM STATEMENT

The biggest problem in image definition development began with the acquisition of an object using object-based object class libraries and modeling using mathematical language models.

A. Using CNN: It is an in-depth learning algorithm that will capture input images of a 2D matrix, assign value (readable weight and bias) to the various elements in the image, and be smart enough to distinguish one from the other.

B. This model was useful for naming the objects in the image but could not tell us the relationship between them (that is, the separation of the obvious images).

C. In this paper, we present a productive model built on a deep repetitive structure that incorporates the latest advances in computer vision and machine translation and that can successfully produce logical sentences. Using RNN: Networks have loops in them, allowing information to flow. LSTMs are a type of RNN, which can learn to depend on it for a long time.

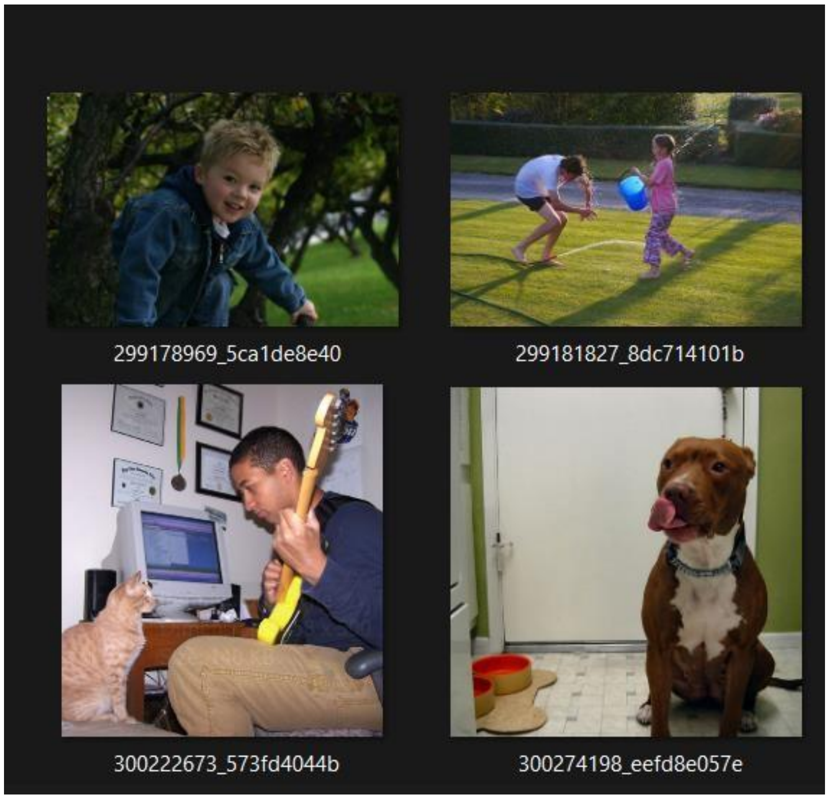
PROPOSED METHODOLOGY

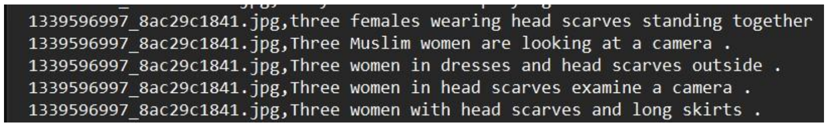
A. Task

The task is to create a system that will capture image input in the form of a dimensional array and produce output that includes descriptive and accurate grammatical sentences.

B. Corpus

We used the Flickr8k database as a corpus. The database contains 8000 images and each image, has 5 captions. 5 captions for a single image help to understand all the different possible situations. The database contains the pre-defined training database captons.txt (8,000images). However, they are individually selected to represent the various scenes. These information sites (Size 1GB) can be downloaded directly online.





```
1339596997_8ac29c1841.jpg,three females wearing head scarves standing together
1339596997_8ac29c1841.jpg,Three Muslim women are looking at a camera .
1339596997_8ac29c1841.jpg,Three women in dresses and head scarves outside .
1339596997_8ac29c1841.jpg,Three women in head scarves examine a camera .
1339596997_8ac29c1841.jpg,Three women with head scarves and long skirts .
```

C. Preliminary consideration

The initial processing of the data is done in two parts, images and related captions are cleaned and pre-processed separately. Pre-image processing is done by uploading input data to the VGG16 Keras API application running on Tensorflow. Keras.applications.vgg16 was previously trained at ImageNet. This has helped us to train images faster with the help of Transfer learning. Definitions are cleared using the tokenizer class in Keras, this will vectorize the chorus text and be stored in a separate dictionary. Then each vocabulary word is mapped with a different index value.

D. Model

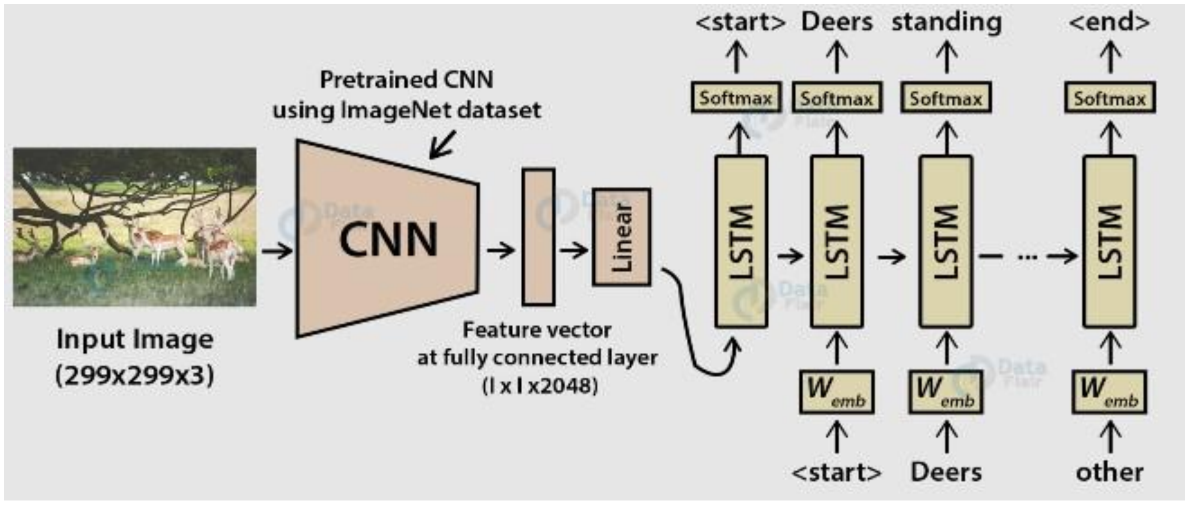
In-depth learning facilitates the process of machine learning using a synthetic neural network on which a few levels are arranged in chronological order. Model-based in deep networks where the flow of information begins at the first level when the model learns something simple then the output is transferred to the second layer of the network, and the input is integrated into the complex object and transferred to the third level. This process continues as each level in the network produces something more complex than the input from which the rising level is found.

CONVOLUTIONAL NEURAL NETWORKS (CNN)

Convolutional Neural Networks are deep-seated nerve networks that can process data input form such as a 2D matrix. Images can be easily represented as a 2D matrix.
CNN is important for working with images. It takes as an input image, gives the significance (weight and bias) of diverse features/items in the image, and separates one from another. CNN uses filters (also known as Kernels) to assist in feature learning (see ambiguous concepts, such as blurring, edge discovery, sharpening, etc.), it is the same with the human brain pointing things at the right time and space. Architecture improves balance in image data sets due to a decrease in the number of parameters involved (from 2048 to 256) and weight reuse.

LONG-SHORT TERM MEMORY (LSTM)

As a form of RNN (continuous neural network), LSTM (short-term memory) can work with sequence prediction problems. It is often used for prediction purposes, as in Google Search our system shows the next word based on the previous text. In all input processing, LSTM is used to generate relevant information and discard non-essential information.
To create a captioning model, we must integrate CNN and LSTM. We can call:
Image Caption Generator Model (CNN-RNN model) = CNN + LSTM.
CNN- Removing features from an image. A pre-trained model called VGG16 is used for this.
LSTM - Generating a description from the extracted image information.



EVALUATION

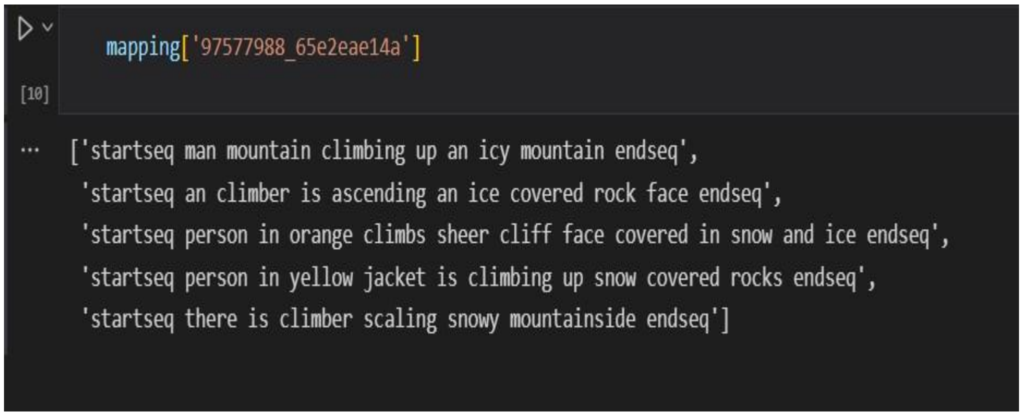
The implementation of the whole process takes place in 5 major steps. The implementation of the five major modules is as follows:

A. Pre-Data Cleaning and Processing:

1. To make it easier and faster, we use Google Collaborator, a free GPU / TPU processing tool, on top of our local equipment, which can take several hours to complete a GPU task that will take a few minutes to complete.
2. Our program starts by uploading both, text files, and various dynamic image files; the test file is stored in the character unit.
3. This alphabet unit is used and used in such a way that it creates a dictionary that puts a map of each image as a list of 5 definitions.
4. The primary function of data cleaning involves removing punctuation, translating

entire text, removing descriptive words, and removing words that contain numeric.

5. In addition, a glossary is created for all the different words from all the descriptions, which will be used in the future to produce captions to test the images.
6. Another aspect of data processing is the process of making our name tokens with a unique reference value. This is because the computer cannot understand common English words, which is why it needs to be represented using numbers. Tokens are then stored in an embossing file that is, in the form of a scattering of characters, but with all the information needed to reconstruct it into a kind of original object.
7. The above two pre-processing functions can be accessed manually or using the Keras .preprocessing module to facilitate coding.
8. We continue to attach the <part> and <end> identifiers to each caption as these will serve as indicators for our LSTM to understand where the captions start and end.
9. We will continue to count the number of words in our vocabulary and find the length of the meaning, which will be used in future paragraphs. Convolutional Neural Networks (CNN)



```
mapping['97577988_65e2eae14a']
[10]
... ['startseq man mountain climbing up an icy mountain endseq',
     'startseq an climber is ascending an ice covered rock face endseq',
     'startseq person in orange climbs sheer cliff face covered in snow and ice endseq',
     'startseq person in yellow jacket is climbing up snow covered rocks endseq',
     'startseq there is climber scaling snowy mountainside endseq']
```

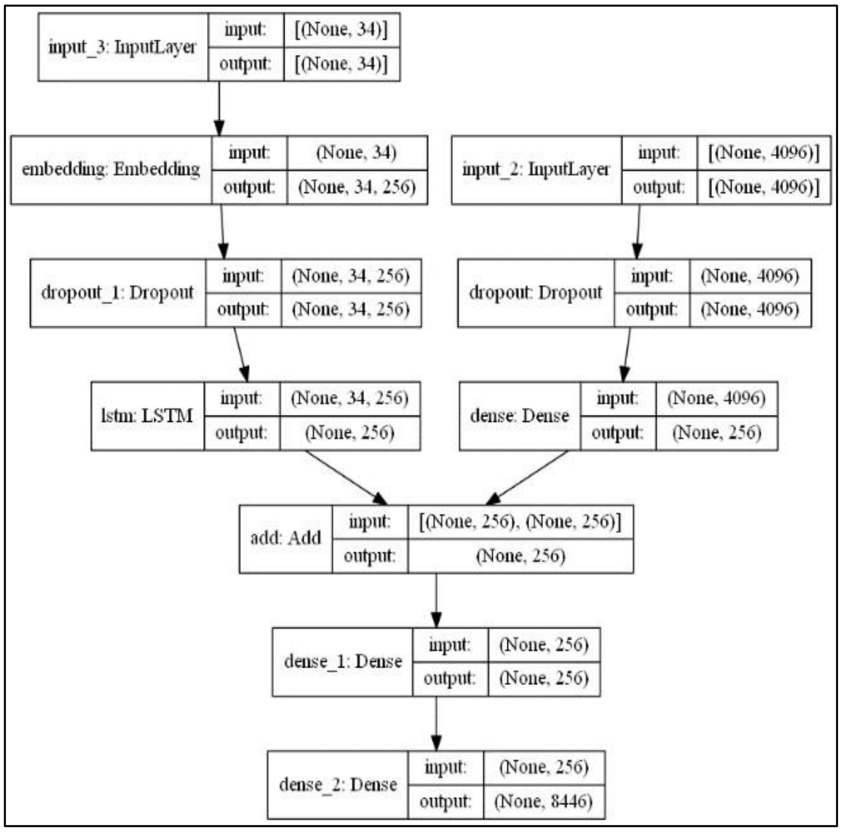
B. Feature vector output:

1. A vector of an element (or a simple element) is a numerical value in the matrix form, which contains information about important features of the object, e.g. strengthening the number of each pixel of an image in our case. These vectors eventually will end up in a pickle file.
2. In our model, we will be using Deep Reading, which is, to uses a pre-trained model (in our case I VGG16 model) to extract features from it.
3. VGG-16 is a convolutional neural network that is 16 layers deep. You can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.
4. Python makes use of this model in our code the most easily with

Keras.applications.vgg16 module. Using it in our code, we will remove the separation layer from it, and thus find the vector of the 2048 element.

5. So, the weights will be downloaded for each image, too then the names of the pictures will be mapped in the correct order with similar features.
6. This process can take a few hours depending on your processor.

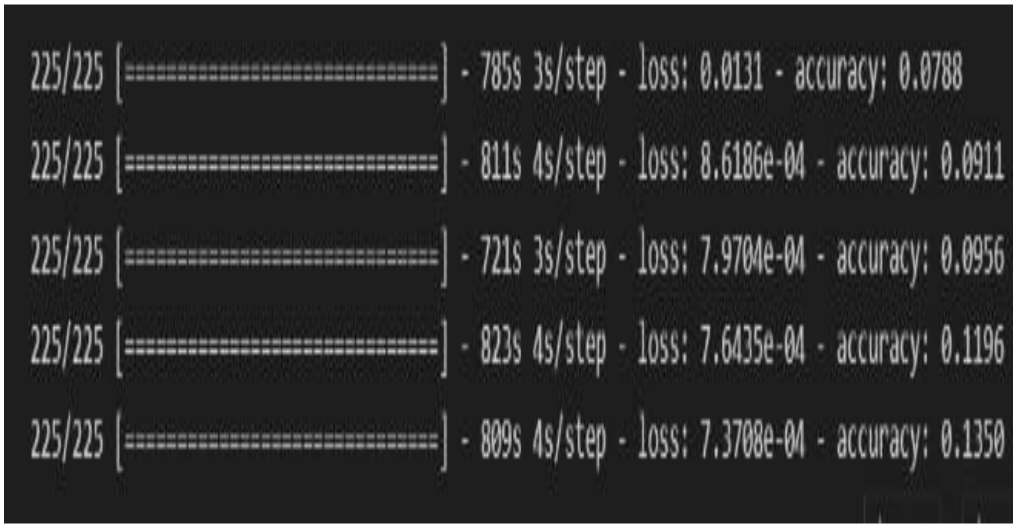
C. Layering the CNN- VGG16 model:



To package the model, we will use the Keras Model from Active API. The building will have 3 parts:

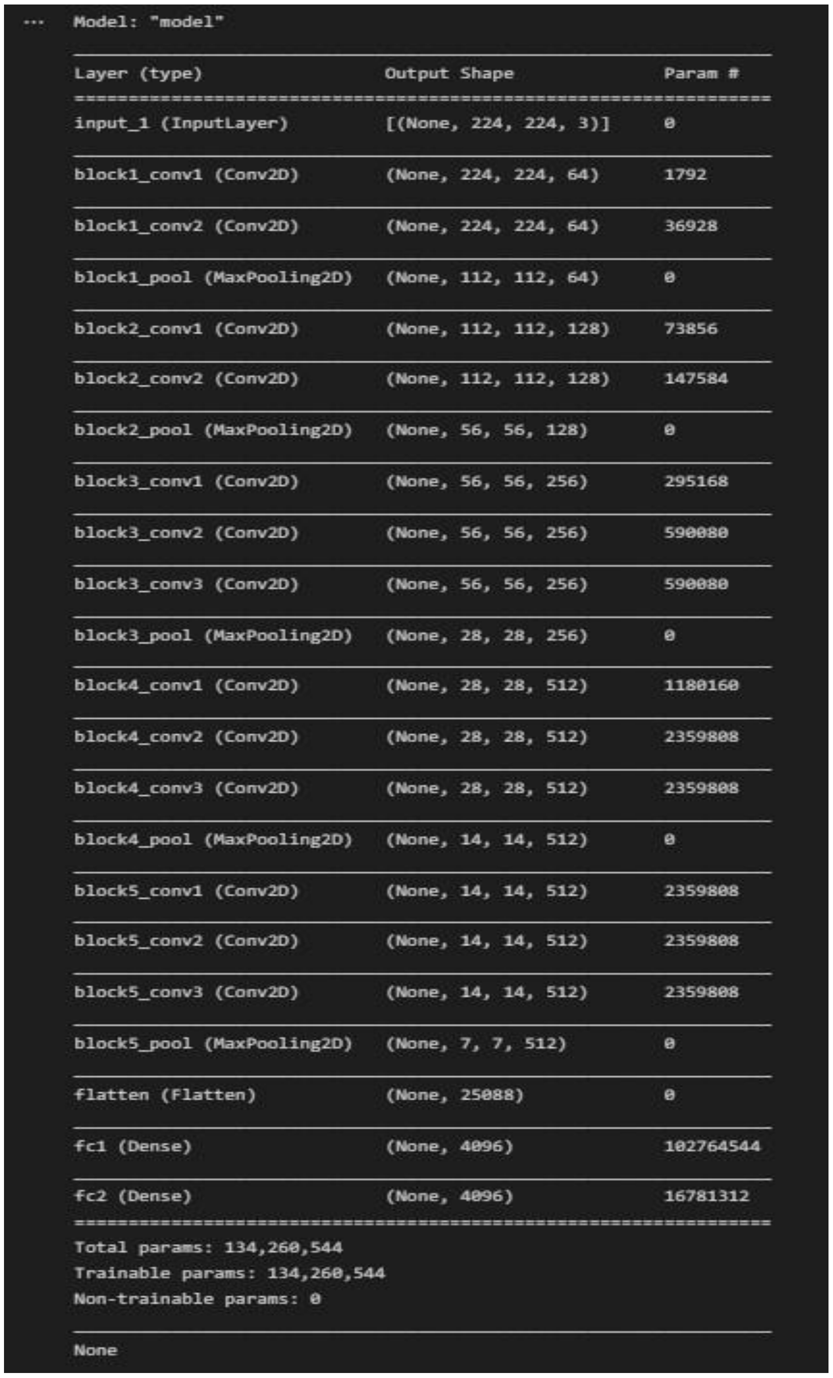
1. Withdrawal Feature: Will be used to reduce the size from 2048 to 256. We will use a Layer to quit. One of these will be added to CNN and each LSTM. We have previously reviewed images with the VGG16 model (excluding the exit layer) and will use the output features predicted by this model as input.
2. Sequence Processor: This embedding layer will manage text input, followed by the LSTM layer.
3. Decoder: We will combine the output from the above two layers, and use a thick layer to make the last predictions. Both feature extractor and sequence the processor emits a vector of fixed length. These are compacted and processed by a dense layer. The number of nodes in the last layer will be equal to the size of our vocabulary.

D. Training the model



```
225/225 [==============================] - 785s 3s/step - loss: 0.0131 - accuracy: 0.0788
225/225 [==============================] - 811s 4s/step - loss: 8.6186e-04 - accuracy: 0.0911
225/225 [==============================] - 721s 3s/step - loss: 7.9704e-04 - accuracy: 0.0956
225/225 [==============================] - 823s 4s/step - loss: 7.6435e-04 - accuracy: 0.1196
225/225 [==============================] - 809s 4s/step - loss: 7.3708e-04 - accuracy: 0.1350
```

1. We will be training our model in 8000+ pictures each with 2048 long feature vectors.
2. Since it is not possible to capture all of this data in memory at the same time, we will use Data Generator. This will help us to create data collections and will improve speed.
3. In line with this, we will be defining their number epochs (i.e. duplication of training database) model it must be completed during its training. This number should choose in such a way that our model is no other not adequately included or overused .model.fit_generator () method will be used. And this whole process will take time depending on the processor.
4. The maximum length of previously defined definitions will be used as a parameter value here. It will also take as enter clean data and tokens.
5. We will also create a sequence creator, who will play the role of predicting the next word based on the original name and vectors of the image feature.
6. While training our model, we can use improvement data (Provided with other files), to monitor model performance, and determine when to save the version to the file.
7. We will continue to maintain several models, from which they came the latter will be used for future testing.



```
...  Model: "model"
     _________________________________________________________________
     Layer (type)                Output Shape              Param #
     =================================================================
     input_1 (InputLayer)        [(None, 224, 224, 3)]     0
     _________________________________________________________________
     block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
     _________________________________________________________________
     block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
     _________________________________________________________________
     block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
     _________________________________________________________________
     block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
     _________________________________________________________________
     block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
     _________________________________________________________________
     block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
     _________________________________________________________________
     block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
     _________________________________________________________________
     block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
     _________________________________________________________________
     block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
     _________________________________________________________________
     block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
     _________________________________________________________________
     block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
     _________________________________________________________________
     block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
     _________________________________________________________________
     block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
     _________________________________________________________________
     block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
     _________________________________________________________________
     block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
     _________________________________________________________________
     block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
     _________________________________________________________________
     block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
     _________________________________________________________________
     block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
     _________________________________________________________________
     flatten (Flatten)           (None, 25088)             0
     _________________________________________________________________
     fc1 (Dense)                 (None, 4096)              102764544
     _________________________________________________________________
     fc2 (Dense)                 (None, 4096)              16781312
     =================================================================
     Total params: 134,260,544
     Trainable params: 134,260,544
     Non-trainable params: 0
     _________________________________________________________________
     None
```

D. Testing the model

1. A separate Python notebook can be created, or the same can be used to perform tests. Either way, we will do it download the professional model we have saved step by step, and generate predictions.
2. The sequence generator will work on this section, except for the tokenizer file we created.
3. The first step is to remove the particular feature the image below the view will be made.
4. How about one of the photos from the remaining 2000 test images is transmitted

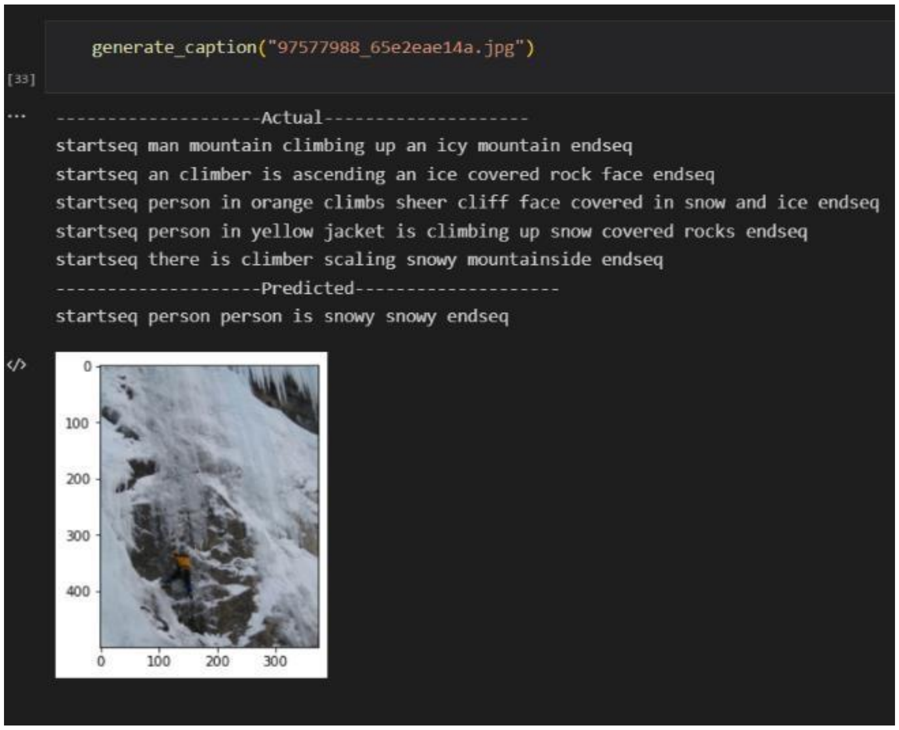
to the user in person. You can repeat with the test data set, and save this file prediction for each image in a dictionary or list.

5. Real performance after image production includes using the first sequence and the final sequence and calling the model over and over again to produce logical sentences.

RESULT / ANALYSIS

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

generate_caption("97577988_65e2eae14a.jpg")



```
generate_caption("97577988_65e2eae14a.jpg")
[33]
--------------------Actual--------------------
startseq man mountain climbing up an icy mountain endseq
startseq an climber is ascending an ice covered rock face endseq
startseq person in orange climbs sheer cliff face covered in snow and ice endseq
startseq person in yellow jacket is climbing up snow covered rocks endseq
startseq there is climber scaling snowy mountainside endseq
--------------------Predicted--------------------
startseq person person is snowy snowy endseq
```

CONCLUSION

Based on the results obtained we can see that depth the learning method used here has been very successful. CNN and LSTM have worked well together in synchronization, they were able to find a connection between objects in pictures. To compare the accuracy of predicted captions, we can compare targeted captions in our Flickr8k trial Database, using BLEU (Bilingual Evaluation Understudy) as the result. BLEU scores are used for text translation to examine translated text by comparing one or more reference versions. Over the years several more neural networks technology used to create captions for mixed image generators, such as those proposed here. E.g., the VGG16 model, or the GRU model instead of the STM model. In addition, the BLEU Score can be used to draw comparisons between these examples, see which provides high accuracy. This paper introduces various developments in the field of machine learning and AI, and how big this arena is. A few of the topics within this paper are open to further research and development, while the paper itself attempts to cover the essential elements needed to create captions for the image generator.

REFERENCES

1. HaoranWang, Yue Zhang, and Xiaosheng Yu, “An Overview of Image Caption Generation Methods”, (CIN-2020)
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 Image Captioning",(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)
7. 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.