Image Captioning using Transfer Learning

Kushagra Chauhan[1] Yash Mittal[2] Sunil Kumar[3]
[1,2]Department of Information Technology [3]Department of Computer Science & Engineering
[1,2,3]Meerut Institute of Engineering & Technology, Meerut, Uttar Pradesh, India

Abstract — In recent years there is evolution which is to be seen in various fields of science. advancements are made from simple mathematical equations to advanced real-world problem-solving, and computer algorithms. Computer vision is also an area that is introduced and evolved for processing the image and examining various aspects like {image classification and object detection } (image captioning based on deep neural networks). Talking about computer vision uses various artificial intelligence techniques and algorithms for confronting any image and deriving meaningful information or conclusions. With the development of deep learning and the natural language process, we can train and use a certain model to predict 1 or 2 sentences from any image. However, the prediction is based on the objects, certain semantics, and other important factors which can result in some unique observations that cannot be overlooked while generating the caption. The caption which is generated should be grammatically correct and in the dictionary for which we have used natural language processing NLTK and the dictionary of words that are to be used in the generation of sentences. Although the generation of image captions is a complicated and difficult task, we have achieved this using the deep neural networks which are based on CNN-RNN, CNN-CNN, and transfer learning. We have maintained a grammar according to which the sentence is formed of 35-36 words. Humans can describe certain scenes using various patterns that are generated parallel while learning to distinguish things and state the object by combining all the factors. In this research paper, we have focused on this type of learning, but as the images are 2 dimensional, we have to map the space with some most probable words that can occur in the image and learn this mapping by training the model.

Keywords: Computer Vision, NLTK, RNN, CNN, Semantics, Dictionary

I. INTRODUCTION

Image captioning is the subset of computer vision and it is basically an encoding and decoding structural architecture in which the image is converted into vectors and interpreted to make a mapping that is used to encode the image into a sequence which can then be identified by the model. Most of the previous work in this field of recognition of visual images has been focused on labeling the images with the help of a fixed dictionary of spatial and objective categories, also more progress has been made and the resultant goals are achieved in these fields.[1, 2]. However, making grammatically correct captions that are valid for the image is a challenging task, these captions should focus on the objects, surroundings, humans, and animals, the actions, and other miscellaneous events that are happening or about to happen in the image. For this to be achieved we need to combine computer vision and natural language processing with some set of word mappings for the model to map them with certain embeddings of vectors for a particular space.

The development of image captioning systems may help visually impaired people in the future, it is drawing more and more attention and became one of the more salient topics in computer visions [3-4] earlier caption generation methods highly rely on an aggregate image embedded with information using fixed libraries in the image and the method is modeled using statistical language models. Aker and Gaizauskas [5] leverage a dependency model for abstracting multiple files on the web which consists of information based on geotagged images. Li et al. [6] proposed an n-gram method that is heavily influenced by a network scale that collects candidate phrases and merges them to make a new sentence that can describe the image. Yang et al. [7] proposed a language model which is aimed and taught from the English Gigaword corpus for obtaining the approximation in the motion of any entity in an image and computing the expected probability of scenes, nouns, and prepositions and with the help of this estimates the caption this problem of generating the image description is first researched and for the same solution is being provided by well-known computer scientist Andrej Karapathy at Stanford.[8]. The main objective of this paper is to develop a model which is as precise as possible with the help of previous methods and implementation for this problem, we have found that this problem can be solved using various available models and algorithms, but our main focus is on the CNN-RNN layering and using inception v3 model we have paired it with Long Short Term Memory (LSTM in short) which is based on recurrent neural network and used in deep learning in our case we will be processing the sequence of data that is our images as it can work on the data stream as well as data points for input this is the partial caption that we have to generate. we need to greedily search for the word that is most probable to come from a selected space and embeddings in the image the model is trained using transfer learning which focuses more on the intuition of solving the problem rather than the output, it stores all the knowledge in the form of weights while solving problems. That is how we can apply this knowledge to identifying the images furthermore we can identify more complicated things by just using the previous knowledge, for example, the knowledge to identify the cars can be applied to identify the trucks. The image mapping is done by converting the image in the vector that makes each layer a separate state and knowledge is stored using weights for each node in the neural network. This project includes the implementation of Convolution neural networks(CNN) and long short-term memory(LSTM), the features of the image are extracted using Inception V3, a model which is well-trained on the ImageNet dataset, and we feed all the features to LSTM for generation of the image description. Image captioning can be extensively used as an aid for blind people, detecting fraud and automation security. it can provide more accurate information regarding photos and videos shared on the internet and can distinguish age-restricted content from user-friendly content.

Our aim is to leverage the machine learning algorithms and develop a system that will accurately identify the objects and images and can generate captions describing scenes of the image and compare its results with other models using BLUE-SCORES.

II. RELATED WORK

In this area of study of computer vision, there are various advancements that are made frequently and mostly are done in image recognition as the image plays the role of the building block for every visual media.

Annotating of images: The goal of our work is to attain the descriptions of the contents of any image whether it is a spatial feature or any object references in the image, the work before us is done by Barnard et al. [9] and Scholar et al. [10] had studied and researched the corresponding words for the images for generating a description segment of the features in the images. Many works [11-14] have also researched and studied similar problems for understanding the spatial scene in a holistic manner by which the scene's spatial feature, its type, objects, and other elements of the image are evaluated.

However, although the focus of this work is on accurately labeling objects and regions with a fixed dictionary of word categories, we focus on region accuracy and high-level caption units for the images.

Explanation generation of caption: Many approaches either pose the problem as a search problem, by searching the words from a training dictionary and this is transferred to the test images where the most compatible annotations are picked out, or the training annotations are first split and then concatenated together. as Proposed in. [15-17]. Some approaches generate captions based on fixed dictionary mapping, a set of words based on image content, or generation grammars, These methods limit versatility, probable outcomes, and possible outcomes. Kiros [18] developed and proposed a logarithmic bilinear model which can be used to generate full-sentenced descriptions of images, using efficient algorithms and data binding techniques but this model uses a fixed length of a context for words, whereas our approach uses a recurrent neural network (RNN) model can generate sentence-following We map the word probability distribution as following previous words. At the time of this work's submission, several closely related preprints appeared on Arxiv, some of which have used RNNs to obtain and re-frame image descriptions. Our RNN is quite more primitive than several of these approaches but also sacrifices performance.

Neural networks in the visual and verbal domains: Several approaches have been initiated to develop and represent images and words at a higher level of abstraction. Regarding images, Convolutional Neural Networks (CNNs) [19, 20] have recently emerged and thrived for being a powerful model class that can be used for image classification and object detection [45]. For sentences, our task uses pre-trained word vectors that are formed from word embeddings [21, 22] to obtain sub-word representations. Finally, recurrent neural networks have previously been used for language modeling, but we further condition these models with images.

Our Model: the main aim of our model is to obtain the most probable, precise, and grammatically correct caption for an image by analyzing it using the CNN-RNN layered model, our model is trained using the flicker 8k data set of 8 thousand images and their is 5 feed caption for one particular image. These captions and images are processed using 2 different dictionaries of word-to-index and index-to-word, which are further used to predict the most probable word for any spatial specification or any object in the image.

We first Assign a modal that makes alignment for the sentences using all the spatial fields that are described through multi-correspondence modal embedding.

After this, we will be treating these correlations as training data for a second multiple-modal RNN which can learn to make precise descriptions.

DataSet Used: the data set that is used to train the CNN-RNN model is Flickr 8k data set in which images are needed and results are stored including the knowledge for previous result data-set and reoccur it with several layers in the network for providing decision-making ability similar to the human brain.

We have to perform certain cleaning tasks, whenever we are dealing with the text, we first need to perform some basic rinse and structuring like casing in lower case for every word, or "dog" and "Dog" are regarded as two separate entities thus resulting in a more complex neural network. We also need to remove special tokens like '%', '&', '#' etc to eliminate words like "cooldude69" from the dictionary.

As the word dictionary for the data set of 40,000 captions can result in at most 40,000 different words we need to take the word according to a certain frequency here we have only considered the words whose frequency is at least 10 times.

Generation of sequence: the sequence is generated with the help of 2 certain markups representing the start of the sequence and the end of the sequence namely "start seq" and "end seq" In, in here the "start seq" indicates that this is a start of the sequence and it is a symbol which will be appended in the start of each description and "end seq" indicates that this is the end of the sequence and it is a symbol which will be added at the end of every caption. for training the model we have separated 6 thousand images in another directory whose captions are encapsulated using "start seq" and "end seq".

Image preprocessing: images are just an input for our model so we can easily state that the image is an equation for our function that is our model and in which we are using an advanced CNN-based neural model i.e. inception v3, however, the input for this particular model is given in the form of a byte stream representation also known as a vector so we have to transform each image into a constant sized vector that is capable of feeding the modal.

This is the reason we have opted out of transfer learning making use of the InceptionV3 model which is a Convolution Neural Network developed and managed by Google Research. This model is further trained and taught on over Imagenet dataset for performing Image classification on over 1000 various classes of images and spatial regions, but our purpose for using this technique is just to convert the image into a fixed-length informative vector that can be used

for processing, this is process is also known as Automatic Feature Engineering. That's why we have removed the softmax layer and extracted a 2048-shaped vector having a bottleneck feature for every image as shown in figure 1,2. Fig 3 shows the internal working of the InceptionV3 model which consists of the softmax layer.

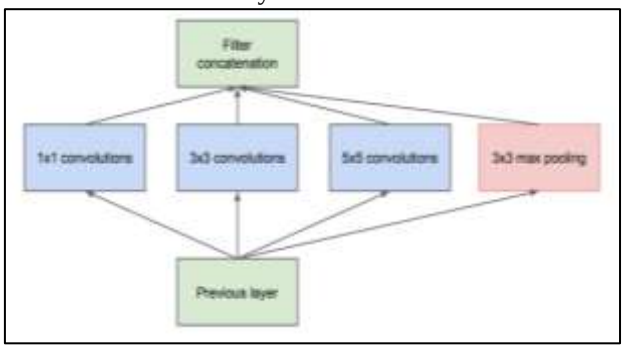

Fig. 1: naive implementation of the inceptionV3 model

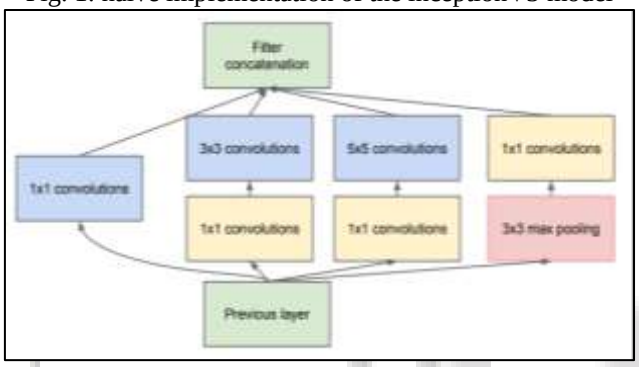

Fig. 2: the improved architecture of inceptionV3 model

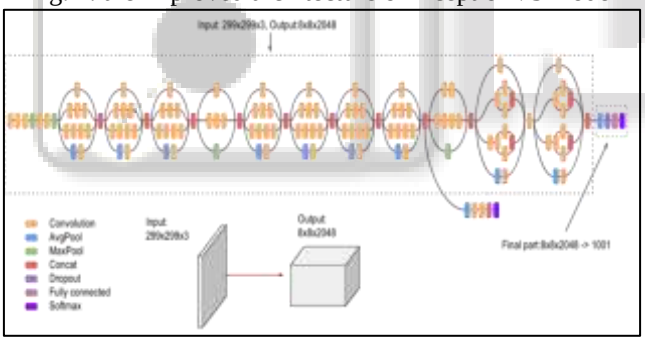

Fig. 3: inception model including softmax layer

Caption Preprocessing: We have to predict the captions for the image, that's why it is necessary during the learning phase and preparation period, captions are used to be the target variable that the model itself has to learn to predict, the process of prediction will be done word by word we have carried a certain threshold of 36 words for one caption.

We will achieve this by encoding one word into a vector, in our grammatically correct dictionary there are 1652 words.

After these steps we will prepare the data using the generator function and split the words for embedding purposes using the image specifications, also we will carry this further by inferring the external image by passing it into the model.

The prediction is done using the model and loading the model with the help of weights that are obtained while training the model. We need not run and train models every single time, we just need to load the model with the weights and use it to get the predictions.

III. PROPOSED MODEL

Prerequisites: our proposed approach is completely based on deep neural networking and Machine learning algorithm technique. Here we have used some mathematical concepts like gradient descent, probability, and using the Keras model layer for providing desirable and most precise output.

We need to know about Deep learning concepts such as multilayer perceptrons, convolutional neural networks, recurrent neural networks, transfer learning, etc.

Data Collection: Our model needs some data to be fed and help the model learn about the images and generate the captions using spatial features and object detection. Our approach has used Flickr 8K dataset that contains 8 thousand images, it can be enhanced by training the model on a more advanced dataset like MS COCO which contains at least 180 thousand images for getting a more complex neural network, and using them to precisely find the spatial feature or detecting any object. Each image in the Flickr data set consists of 5 captions giving us a total of 40 thousand captions to train our model.

```
101654506_8eb26cfb60.jpg#0      A brown and white dog is running through the snow .
101654506_8eb26cfb60.jpg#1      A dog is running in the snow
101654506_8eb26cfb60.jpg#2      A dog running through snow .
101654506_8eb26cfb60.jpg#3      a white and brown dog is running through a snow covered field .
101654506_8eb26cfb60.jpg#4      The white and brown dog is running over the surface of the snow
```

Fig. 4: input structure of the image description and image names

We have bifurcated images as The training set is of 6 thousand images and the dev set of thousand images including the test set having a thousand images on which our model is tested for accuracy.

Understanding the data: the dataset is organized with directories of images and captions stored namely in a token.txt file in which every caption is matched with the image file name and has a notation from 0 to 4 that depicts the 5 captions every line in here contains <name_of_image>#pos <token> where pos can vary from 0 to 4, we have to create another hashmap that can be implemented using dictionary named descriptions without having the extension of the image that is (.jpg extensions) as keys and the values are the 5 captions for that particular image.

Representing images: as per the previous work that is done in this field [23,11], it is observed that sentence description makes more recurring references to objects that are to be attributed. We have used and followed the method of Grinshick et al. [24], for detecting the objects in each image by leveraging the concept of a Region Convolutional Neural Network (RCNN), this particular CNN is pre-trained on the Image Net dataset and tested among different 200 classes of the ImageNet Detection Challenge [1].

Data preprocessing for Images: these images are used as input for feeding our model which is true to be converted every image in the form of vectors that can then be used to provide input for our neural network, this application makes use of transfer learning by using InceptionV3 model which is itself a pre-trained Convolutional Neural Network

created and developed by Alphabet that makes an accurate vector and used for object detection in the images, we have removed the last layer of this model for providing us the vector of length 2048 stream points (bottleneck features) for each image. We need to save all the learned knowledge and features to a basic file that can be done using a hashmap and in Python as a Dictionary and save it on the device by saving the file as a Pickle file having <filename>.pkl format, in which the Keys are the name of the images and values are the featured vector of length 2048 stream. As this process can take some time, we need to save this information on the disk.

Similarly, we can encode the rest of the remaining images after that we can save the specifications.

```
# Convert all the images to size 299x299 as expected by the
# inception v3 model
img = image.load_img(image_path, target_size=(299, 299))
# Convert PIL image to numpy array of 3-dimensions
x = image.img_to_array(img)
# Add one more dimension
x = np.expand_dims(x, axis=0)
# preprocess images using preprocess_input() from inception module
x = preprocess_input(x)
# reshape from (1, 2048) to (2048, )
x = np.reshape(x, x.shape[1])
```

Fig 5: converting images into vectors using inception v3

Data preprocessing for Captions: the captions are basically the words that correspond to the image object and spatial features, but we needed to predict the entire caption word by word we have to keep track of the end and the start of the image description using 2 different token "end seq" and "start seq". so we have encoded this text also in a fixed-size vector, these vectors are used while processing the image and predicting the caption, for now, we just make 2 dictionaries that will act as a hashmap namely word_to_idx and idx_to_word, we need to establish the inter correspondences modal relationships, so we have to represent the character sequence and the statements using the same n-dimensional embedding space we have achieved this using a multidimensional numpy array. There is also a way to use the character collections bigrams, or make dependency tree relations as proposed in previous works[25], but it comes with the disadvantage of using the biggest size of the sentence that requires the further Dependency Tree Parser technique which can be trained on the unstructured text description, without having grammar.

	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	startseq	the
2	Image_1	startseq the	black
3	Image_1	startseq the black	cat
4	Image_1	startseq the black cat	sat
5	Image_1	startseq the black cat sat	on
6	Image_1	startseq the black cat sat on	grass
7	Image_1	startseq the black cat sat on grass	endseq

Fig 6: generation of the caption

Preparation of data using generator function: This step is one of the most important steps that will create the description word by word, we will be framing this method as a supervised instance of a machine learning problem in which we can make a set of the data points D = {Xi, Yi}, in this data point Xi depicts the feature vector of data point at I index in the multidimensional array and Yi represents the target variable. This will reiterate until all the features are predicted the maximum time of iteration is 35 words, however, it is not to be confused that the image + description of character sequence is not considered as a single and separated data point but these are considered various points which are relying on the length of the caption, as we the sequences are processed, we have to deploy a Recurrent Neural Network for reading all of these partial captions, we will be passing the sequence of points of the indices on which every index will represent a unique word from the dictionary of word_to_idx.

We would be processing it using batch processing, so we need to make sure that each sequence is of equal length. That is why we are going to append 0s at the end of every sequence for maintaining the length variant. we would append 34 zeroes for making a caption the data matrix will look as follow:

	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	[9]	10
2	Image_1	[9, 10]	1
3	Image_1	[9, 10, 1]	2
4	Image_1	[9, 10, 1, 2]	8
5	Image_1	[9, 10, 1, 2, 8]	6
6	Image_1	[9, 10, 1, 2, 8, 6]	4
7	Image_1	[9, 10, 1, 2, 8, 6, 4]	3
8	Image_2	[9]	10
9	Image_2	[9, 10]	12
10	Image_2	[9, 10, 12]	2
11	Image_2	[9, 10, 12, 2]	5
12	Image_2	[9, 10, 12, 2, 5]	11
13	Image_2	[9, 10, 12, 2, 5, 11]	6
14	Image_2	[9, 10, 12, 2, 5, 11, 6]	7
15	Image_2	[9, 10, 12, 2, 5, 11, 6, 7]	3

Fig. 7: using indexes and adding padding of zeros

```
def data_generator(train_description,
                   train_encoding,
                   word_to_idx,
                   max_len,
                   batch_size):
    x1, x2, y = [], [], []
    n = 0
    while True:
        for image_name, captions in train_description.items():
            n += 1
            photo = str(train_encoding[image_name]) + ".jpg"
            for caption in captions:
                seq = [word_to_idx[word] *
                       for word in caption.split()
                       if word in word_to_idx]
                for word in range(1,len(seq)):
                    xi = seq[0:word]
                    yi = seq[word]

                    xi = pad_sequences([xi],
                                       maxlen = max_len,
                                       value = 0,
                                       padding = "post")[0]
                    yi = to_categorical([yi], num_classes = vocab_size)[0]

                    x1.append(photo)
                    x2.append(xi)
                    y.append(yi)

                if n==batch_size:
                    yield [[np.array(x1), np.array(x2)], np.array(y)]

                    x1,x2,y = [],[],[]
                    n = 0
```

Fig. 8: use of generator function for generating the caption and keeping the previous knowledge of the caption for any image and its object.

Word Embeddings: We have mapped each word - index to a vector that is 200-byte stream long and for this purpose, we will use a predefined open source, well-trained

GLOVE model by using that we will maintain unique words with at least 1652 in our vocabulary and map it to the embedding matrix that further is to be loaded in the model before initiating the process of gathering features and training. These glove embedding helps us to maintain the vectors that are refined and trained for every word in the word_to_idx and idx_to_word dictionaries.

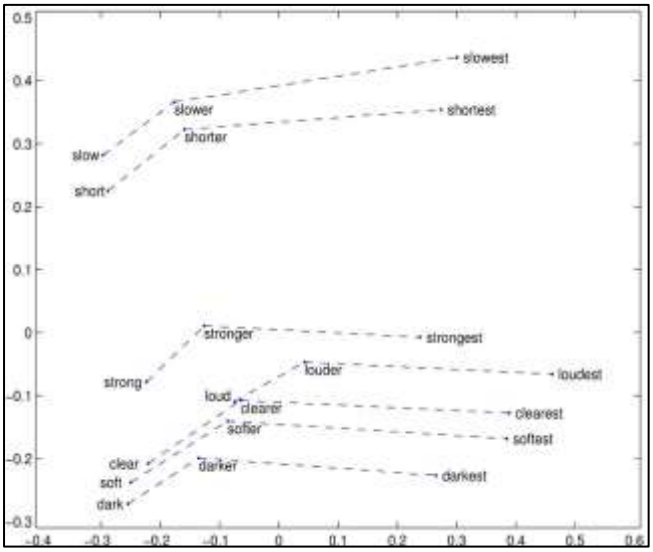

Fig. 9: adding vectors for the description in captions

The architecture of the model: the input system is included with two parts, one is the image that is transformed into a vector and another is the incomplete description, for this purpose, the API provides the ability of sequential feeding cannot be used which is provided by default with the help of Keras library. This is the reason we will be using the hybrid functional API which makes us available to create model correspondence and Merge both Models.

In the below diagram, we can see the structure of the network and better understand the 2 input streams of inputs. These layers are then connected to and from an RNN-CNN model that will further help in the processing of description from the given image.

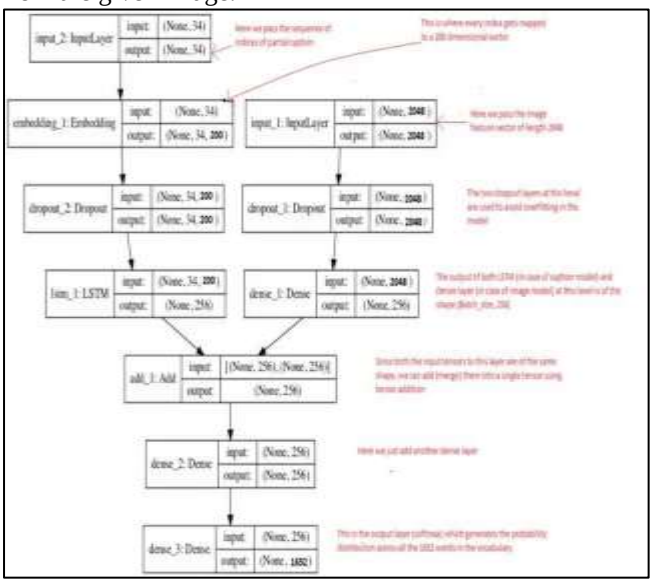

Fig. 10: whole model layer system of feeding with the data and predicting the output

```
model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	(None, 34)	0	
input_3 (InputLayer)	(None, 2048)	0	
embedding_2 (Embedding)	(None, 34, 200)	330400	input_4[0][0]
dropout_3 (Dropout)	(None, 2048)	0	input_3[0][0]
dropout_4 (Dropout)	(None, 34, 200)	0	embedding_2[0][0]
dense_2 (Dense)	(None, 256)	524544	dropout_3[0][0]
lstm_2 (LSTM)	(None, 256)	467968	dropout_4[0][0]
add_2 (Add)	(None, 256)	0	dense_2[0][0] lstm_2[0][0]
dense_3 (Dense)	(None, 256)	65792	add_2[0][0]
dense_4 (Dense)	(None, 1652)	424564	dense_3[0][0]

Total params: 1,813,268
Trainable params: 1,813,268
Non-trainable params: 0

Fig. 11: whole model summary generated using Keras library.

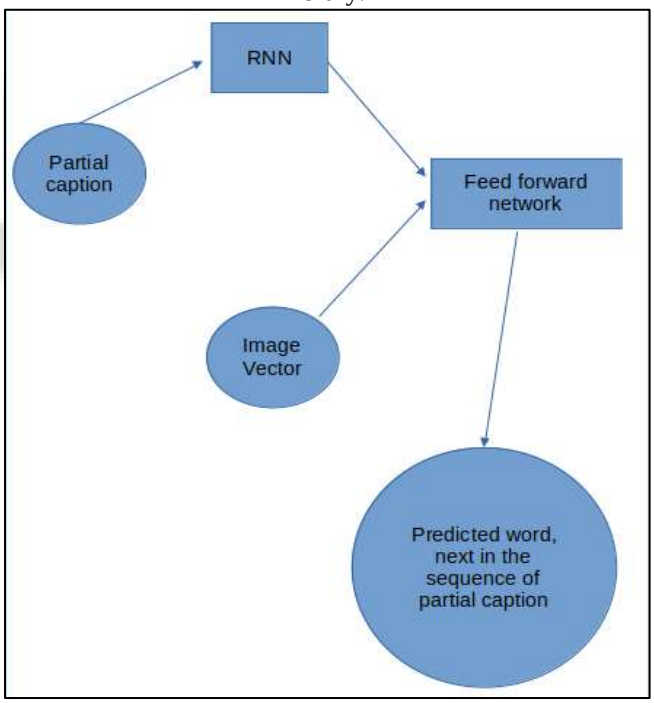

Fig. 12: overall working of the model in predicting the output.

IV. RESULTS AND DISCUSSION

Now to the result part we have collected the data that can be used in the context of our model and we have refined all the data including the captions and formed a multidimensional numpy array having paddings and the tokens, we have generated the function that will help us to find a more probable word for the spatial feature or the vectors in the sequences, now we just need to iteratively find the caption one by one, we have built the model having 2 different input

streams and made multi-model corresponding using LSTM and CNN deep neural network for generating the output.

In this step, we will be testing our model by referencing the up-to-the-minute images and let us see how we can create and process a new description for a test image and find the accuracy.

Fig. 13: test image for generating the captions

The model creates a unique twelve-unit-in-length vector that is formed using the statistical probabilistic distribution of all the valid character sequences in the vocabulary, that is why we have to pick up and select the words greedily, and those words having the most probability are taken using the feature vector and partial descriptions. This model is taught well, as we have trained and precompiled the model with various epochs making the loss as less as possible. using the generator function and statistically analyzing the probability of the caption.

vocabulary generated for this image contains words as follows:

vocab = { cat, road, end seq, grass, is, sat, black, on, start seq, the, run, white, shadow, of, walking, gazing }

Iteration 1: Input - vector + "start seq" [partial caption]

most probable word: the

Iteration 2: Input - vector + "start seq the" [partial caption]

most probable word: black

Iteration 3: Input - vector + "start seq the black"[partial caption]

most probable word: cat

…

```
227/227 [==============================] - 789s 3s/step - loss: 5.2349
227/227 [==============================] - 747s 3s/step - loss: 4.0266
227/227 [==============================] - 747s 3s/step - loss: 3.5950
227/227 [==============================] - 748s 3s/step - loss: 3.3310
227/227 [==============================] - 753s 3s/step - loss: 3.1360
227/227 [==============================] - 771s 3s/step - loss: 2.9874
227/227 [==============================] - 753s 3s/step - loss: 2.8711
227/227 [==============================] - 743s 3s/step - loss: 2.7797
227/227 [==============================] - 753s 3s/step - loss: 2.7058
227/227 [==============================] - 757s 3s/step - loss: 2.6342
227/227 [==============================] - 751s 3s/step - loss: 2.5721
227/227 [==============================] - 755s 3s/step - loss: 2.5172
227/227 [==============================] - 743s 3s/step - loss: 2.4646
227/227 [==============================] - 746s 3s/step - loss: 2.4135
227/227 [==============================] - 748s 3s/step - loss: 2.3687
227/227 [==============================] - 737s 3s/step - loss: 2.3338
227/227 [==============================] - 742s 3s/step - loss: 2.2974
227/227 [==============================] - 748s 3s/step - loss: 2.2642
227/227 [==============================] - 748s 3s/step - loss: 2.2341
227/227 [==============================] - 739s 3s/step - loss: 2.2068
```

Fig. 14: Reduction of the loss in the model prediction

We iterate through until we reach the most probable token that is "end seq". Also, the previous words are found by using the previous knowledge gained by the generated function and the knowledge yield during the captions word by word these whole words are then embedded in a numpy array sequence.

Another stoppage condition is when we reached the greater threshold of the number of words generated by the model.

V. CONCLUSION AND FUTURE SCOPE

we are checking the accuracy of our model using the BLEU score which is basically a metric or unit for measuring the translated text generated using (Bilingual Evaluation Understudy) this score is generally between 0 to 1 which matches the similarities between the machine-generated text and the set of high-quality reference translations.

```
from nltk.translate.bleu_score import corpus_bleu
# validate with test data
actual, predicted = list(), list()

for key in tqdm(test):
    # get actual caption
    captions = mapping[key]
    # predict the caption for image
    y_pred = predict_caption(model, features[key], tokenizer, max_length)
    # split into words
    actual_captions = [caption.split() for caption in captions]
    y_pred = y_pred.split()
    # append to the list
    actual.append(actual_captions)
    predicted.append(y_pred)

# calculate BLEU score
print("BLEU-1: %f" % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
print("BLEU-2: %f" % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
```

```
  0%|          | 0/810 [00:00<?, ?it/s]

BLEU-1: 0.543608
BLEU-2: 0.316664
```

Fig. 14: calculating the BLEU scores for getting the accuracy of the generated text.

Output descriptions:

REFERENCES

[1] O Russakovsky, J Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "Imagenet large scale visual recognition challenge", pp. 211-252, April 2015.
[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) Challenge". International Journal of Computer Vision, vol. 88, no. 2, pp. 303-338, June 2010.
[3] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, Lei Zhang, "Bottom-Up and Top-Down Attention for Image Captioning," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6077-6086, June 2018.
[4] Q. You, Z. Zhang, and J. Luo, "End-to-End Convolutional Semantic Embeddings," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5735–5744, June 2018.
[5] A. Aker and R. Gaizauskas, "Generating Image Descriptions using Dependency Relational Patterns," Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, vol. 49, no. 9, pp. 1250–1258, July 2010
[6] S. Li, G. Kulkarni, T. L. Berg, and Y. Choi, "Composing Simple Image Descriptions using Web-scale N-grams," in Proceeding of Fifteenth Conference on Computational Natural Language Learning, pp. 220–228, June 2011.
[7] Y. Yang, C. L. Teo, H. Daume, and Y. Aloimonos, "Corpusguided Sentence Generation of Natural Images," Proceeding of the Conference on Empirical Methods in Natural Language Processing, pp. 444–454, July 2011.
[8] Andrej Karpathy Li Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3128-3137, 2015.
[9] K. Barnard, P. Duygulu, D. Forsyth, N. De Freitas, D. M. Blei, and M. I. Jordan. "Matching words and pictures". Journal of Machine Learning Research (JMLR), pp. 1107–1135, Feb 2003.
[10] R. Socher and L. Fei-Fei, "Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora", 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, pp. 966-973, August 2010.
[11] L.-J. Li, R. Socher, and L. Fei-Fei, "Towards total scene understanding: Classification, annotation, and segmentation in an automatic framework," 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2036–2043, August 2009.
[12] S. Gould, R. Fulton, and D. Koller, "Decomposing a scene into geometric and semantically consistent regions," 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 2009, pp. 1-8, May 2009.
[13] L. -J. Li and Li Fei-Fei, "What, where and who? Classifying events by scene and object recognition," 2007 IEEE 11th International Conference on Computer Vision, Rio de Janeiro, Brazil, 2007, pp. 1-8, December 2007.
[14] X. Chen and C. L. Zitnick, "Mind's eye: A recurrent visual representation for image caption generation," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 2422-2431, June 2014.
[15] P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi, "Collective generation of natural image description", Association for Computational Linguistics, pp. 359-368, July 2012.
[16] S. Li, G. Kulkarni, T. L. Berg, A. C. Berg, and Y. Choi. Composing, "Simple image descriptions using web-scale n-grams", Conference on Natural Language Learning, pp. 220–228, 2011.
[17] P. Kuznetsova, V. Ordonez, T. L. Berg, U. C. Hill, and Y. Choi. Treetalk: Composition and compression of trees for image descriptions", Transactions of the Association for Computational Linguistics, pp. 351–362, 2014.
[18] R. Kiros, R. S. Zemel, and R. Salakhutdinov, "Multimodal neural language models", Proceedings of the 31st International Conference on Machine Learning, pp. 595-603, 2014.
[19] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.
[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", In Non-Invasive Prenatal Screening, 2012.
[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge", International Journal of Computer Vision (IJCV), 2014.
[22] R. Jeffrey Pennington and C. Manning, "Glove: Global vectors for word representation", Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543, 2014.
[23] G. Kulkarni et al., "Baby talk: Understanding and generating simple image descriptions", Proceedings of the 2014 Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, CO, USA, pp. 1601-1608, 2011.
[24] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, pp. 580-587, 2014.
[25] A. Karpathy, A. Joulin, and L. Fei-Fei. "Deep fragment embeddings for bidirectional image sentence mapping", pp. 1406-5679, December 2014.

