International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 03 | March - 2023 Impact Factor: 7.185 ISSN: 2582-3930

From Image Processing to Natural Language Generation: A Study on Image Captioning

Authors:- Ashutosh chouhan [1], Ritik jaiswal [2], Prof. Vandana Kate [3]

[1],[2]Student, Department of Computer Science and Information Technology, Acropolis institute of technology and research, Indore, India

[3]Associate Professor, Department of Computer Science and Information Technology, Acropolis institute of technology and research, Indore, India

Abstract — Image captioning is an essential tool in modern times, and many built-in applications use deep neural network models to generate captions for images. The process of generating image captions entails detecting important objects, describing their characteristics, understanding their connections within the image, and producing sentences that are both grammatically and semantically accurate. In this paper, we propose a deep learning model that uses computer vision and machine translation to detect objects in images, recognize their relationships, and generate captions. Our model employs the Transfer Learning technique and the Flickr8k dataset, and we implemented it in Python3. We also provide details on the functions and structure of various neural networks used in our approach, such as CNN, RNN, and LSTM.

Image caption generators have various applications in the fields of Computer Vision and Natural Language Processing. For instance, they are used in image segmentation, which is utilized by Facebook and Google Photos. These generators can even extend their use to video frames and can automate the process of image interpretation. Additionally, they have significant potential for aiding visually impaired individuals..

Keywords- Image, Caption, CNN, caption, RNN, LSTM, Neural Networks

I. INTRODUCTION

Detecting objects and describing them using natural language processing (NLP) has been a longstanding challenge for artificial intelligence. Until recently, computer vision researchers considered this task impossible. In recent years, the field of image caption generation has undergone significant progress, thanks to the development of advanced deep learning techniques, the availability of large datasets, and powerful computational resources. These models are capable of processing images and generating natural language descriptions that accurately capture the visual context of the image. This requires a combination of both image processing and natural language processing techniques, resulting in captions that are fluent, coherent, and semantically meaningful. While humans can perform this task effortlessly, it requires a strong algorithm and significant computational power for a computer system to do so.

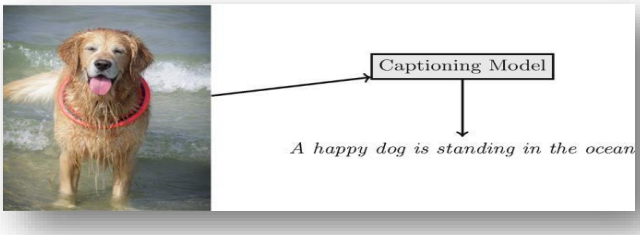


Figure 1: Our model is based on a deep learning neural network that consists of a vision CNN followed by a language generating RNN. It generates complete sentences as an output

To simplify this problem, many attempts have been made to break it down into various simpler problems such as object detection, image classification, and text generation. Computer systems take input images as two-dimensional arrays and map them to captions or descriptive sentences. In the past few years, there has been significant focus on the automated generation of image captions.. However, to encourage rapid progress, benchmark datasets require fast, accurate, and competitive evaluation metrics. The automatic generation of descriptive English sentences that accurately describe the contents of an image has the potential to greatly benefit individuals with visual impairments, providing them with improved comprehension of the visual information available online.

While deep learning methods have demonstrated advanced results on caption generation problems, this task is significantly more challenging than well-studied image classification or visual perception tasks. One of the impressive aspects of these methods is that they frequently define an end- to-end model for generating captions from photographs, which doesn't demand complicated data preparation or a series of specially designed models. Deep learning has attracted a lot of attention because it's particularly good at a kind of learning that has the potential to be very useful for real-world applications. The ability to learn from unlabeled or unstructured data is a significant benefit for those interested in real-world applications.

II. PROBLEM STATEMENT

The task of image description generation has long been

considered impossible by computer vision researchers. However, with the advancements in deep learning techniques, availability of large datasets, and increased computational power, models have been developed that can generate captions for images. The process involves image processing and natural language processing concepts to recognize the context of an image and describe it in a natural language such as English. While humans can do this task easily, it requires a strong algorithm and a lot of computational power for a computer system to do so. To simplify the problem, researchers have attempted to break it down into various simpler problems such as object detection, image classification, and text generation.

Image Caption Generator: To solve the problem, researchers have made use of Convolutional Neural Networks (CNNs), a deep learning algorithm that can intake a 2D matrix input image, assign importance (learnable weights and biases) to different aspects/objects in the image, and differentiate one from the other. However, while this model was useful in naming objects in an image, it couldn't tell us the relationships between them, which is plain image classification.

To address this limitation, researchers have presented a generative model built on a deep recurrent architecture that combines recent advances in computer vision and machine translation, which can effectively generate meaningful sentences. Recurrent Neural Networks (RNNs) have also been used, which are networks with loops that allow information to persist. Long Short-Term Memory (LSTM) is a particular kind of RNN capable of learning long-term dependencies.

In conclusion, the ability to automatically describe the content of a picture using properly formed English sentences may be a challenging task, but it could have a significant impact on helping visually impaired people better understand the content of images online. Deep learning methods have demonstrated advanced results on caption generation problems, and the ability to learn from unlabeled or unstructured data is a huge benefit for real-world applications

III. PROPOSED METHODOLOGY

Overall, the proposed methodology is focused on building a system that generates a sentence describing an input image in a grammatically correct and syntactically accurate manner. The methodology involves using the Flickr 8K dataset, which consists of 8000 images, and for each image, there are five captions that help in understanding various possible scenarios.

The methodology includes data preprocessing, which is done in two parts - image preprocessing and caption preprocessing. Image preprocessing is done using the Xception application of the Keras API, which is pre-trained on ImageNet, while caption preprocessing is done using the tokeniser class in Keras to vectorize the text corpus.

Figure 2: Glimpse of the Flickr8k Image Dataset

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

Figure 3: Glimpse of Flickr8k Text File

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

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

The proposed model is based on deep learning, with a focus on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNNs are used to process input images, assign importance to various objects in the image, and differentiate one from the other. RNNs are used to generate the sentence by making sense of previous words and keeping them in mind to generate the next words.

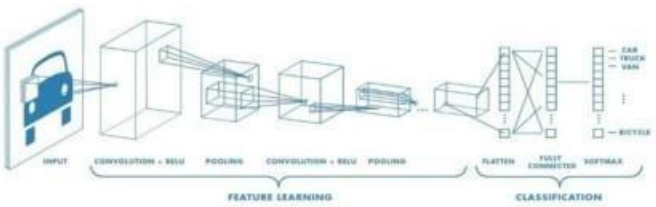


Figure 4: Architecture of Convolutional Neural Networks for object classification.[11]

The proposed model utilizes a CNN + LSTM architecture, where the CNN is used as the encoder and the LSTM is used as the decoder. The encoder maps the input image and transforms it into a fixed-length vector representation, which is used as the initial hidden state of the decoder. The decoder ultimately generates the final sentence as a prediction.

In conclusion, the proposed methodology is a deep learning-based approach that uses CNNs and RNNs to generate descriptive sentences for input images. The method involves data preprocessing, model training, and utilizes transfer learning techniques to improve the accuracy of the generated sentences.

Figure 5: Loop in RNN[6]

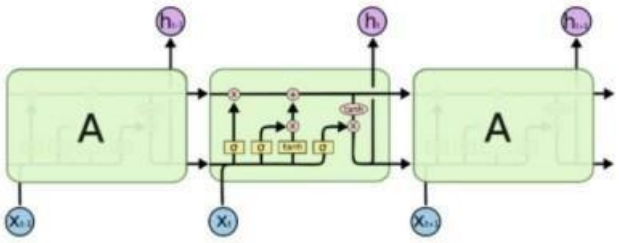
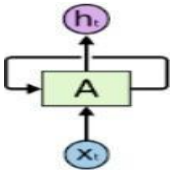


Figure 6: The recurrent module in LSTM contains four interconnected layers.[7]

Architecture



Our proposed architecture for image captioning involves using a combination of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models. The CNN processes the image input, and the LSTM generates the caption output. To handle variable-length source sentences, we utilize an "encoder" RNN that converts them into a fixed-length vector representation. This vector is then used as the initial hidden state of the "decoder" RNN, which generates the final meaningful sentence as the output [7]. Figure 7 illustrates our CNN-LSTM structure [5].

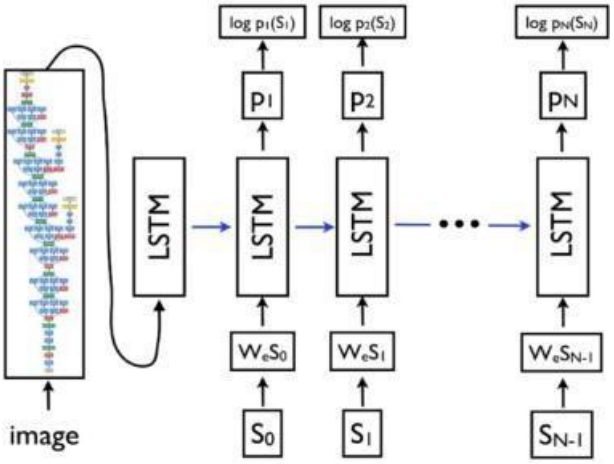


Figure 7: CNN-LSTM structure.[5]

However, we aim to improve the architecture by replacing the RNN with a deep CNN. This is because a deep CNN can produce a richer representation of the input image by embeddingit into a fixed-length vector. To achieve this, we pre-train the CNN for an image classification task and use the last hidden layer as input to the RNN decoder that generates the caption sentences.

IV. EVALUATION.

The program executes in five major steps, which are describedas follows:

A. Data Cleaning and Preprocessing:

1. Google Colaboratory is used for a fast and convenient working experience. This tool provides free GPU/TPU processing power, which is much faster than local machines that can take several hours to process.
2. The program loads both the text file and image file into separate variables, where the text file is stored in a string.
3. The string is manipulated to create a dictionary that maps each image with a list of five descriptions.
4. The main task of data cleaning involves removing punctuation marks, converting the text to lowercase, removing stop words, and removing words that contain numbers.
5. A vocabulary of all unique words from all the descriptions is created, which will be used to generate captions for test images.
6. The data is tokenized using a unique index value, and the tokens are stored in a pickle file. The <start> and <end> identifier are also appended to each caption.
7. The above two preprocessing tasks can be done manually or by using the Keras.preprocessing module.
8. The program calculates the number of words in the vocabulary and the maximum length of the description, which will be used in later phases.

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

Figure 8: Text file after performing data cleaning

B. Extraction of feature vectors:

1. A feature vector contains information about an object's important characteristics and is stored as a numerical value in matrix form.
2. Transfer learning is used, which involves using a pre- trained model, in this case, the Xception model, to extractfeatures from it.
3. The Xception model is a 71-layer deep Convolutional Neural Network trained on the Imagenet dataset, which has millions of images and over 1000 different classes to classify from.
4. The program uses keras.applications.xception module to drop the classification layer and obtain the 2048 feature vector.
5. The program downloads the weights for each image and maps the image names with their respective feature array.
6. This process can take a few hours depending on the processor.
7. The model structure consists of three parts: Feature Extractor, Sequence Processor, and Decoder. These parts will reduce the dimensions from 2048 to 256, handle the textual input, and merge the output from both layers,respectively.

```
{'1000268201_693b08cb0e.jpg': array([[0.36452794, 0.12713662, 0.0013574 , ..., 0.221817  , 0.01178991,
        0.24176797]], dtype=float32),
 '1001773457_577c3a7d70.jpg': array([[0.00751702, 0.22909527, 0.        , ..., 0.3349492 , 0.12825981,
        0.01659334]], dtype=float32),
 '1002674143_1b742ab4b8.jpg': array([[9.9985598e-05, 1.3369371e-01, 1.1934584e-02, ..., 0.0000000e+00,
        4.4273719e-02, 3.0947649e-03]], dtype=float32),
 '1003163366_44323f5815.jpg': array([[0.15107858, 0.03335499, 0.37203324, ..., 0.19086674, 0.1891881 ,
        0.07136369]], dtype=float32),
 '1007129816_e794419615.jpg': array([[2.2596777e-04, 4.1892439e-02, 8.0483657e-01, ..., 1.0450631e-02,
        4.4436250e-02, 4.4828591e-01]], dtype=float32),
 '1007320043_627395c3d8.jpg': array([[0.16503273, 0.        , 0.        , ..., 0.        , 0.26633868,
        0.04421796]], dtype=float32),
 '1009434119_febe49276a.jpg': array([[0.        , 0.        , 0.04739637, ..., 0.29316148, 0.29465917,
        0.1394243 ]], dtype=float32),
 '1012212859_01547e3f17.jpg': array([[0.06308731, 0.01702742, 0.24325731, ..., 0.09328046, 0.0102623 ,
        0.        ]], dtype=float32),
 '1015118661_980735411b.jpg': array([[0.        , 0.03463895, 0.11321168, ..., 0.0071128 , 0.        ,
        0.0355743 ]], dtype=float32)
```

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

C. Layering the CNN-RNN model:

To stack our model, we will use the Keras Model from theFunctional API, which will consist of three parts:

Feature Extractor: This component will reduce the dimensions from 2048 to 256 using a Dropout layer. We will add a Dropout layer in both the CNN and the LSTM. Our photo data has already been preprocessed with the Xception model, and we will use the extracted features from this model (without the output layer) as input.

Sequence Processor: The Embedding layer will handle the textual input, followed by the LSTM layer.

Decoder: The output from the feature extractor and sequence processor layers will be merged, and a Dense layer will make the final predictions. Both the feature extractor and sequence processor will output a fixed-length vector, which will be mergedand processed by a Dense layer. The number of nodes in the final layer will match the size of our vocabulary.

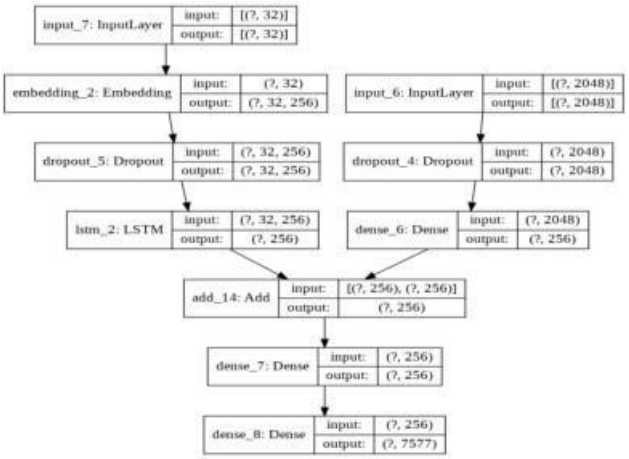


Figure 10: Structure of the Neural Network

D. Training the model:

1. The program will train the model on 6000 images, each having 2048 long feature vectors.
2. A Data Generator is used to create batches of the data since it is not possible to hold all the data in memory at the same time.
3. The program will define the number of epochs, i.e., the iterations of the training dataset.
4. The model.fit_generator() method is used, and this process takes some time, depending on the processor.
5. The maximum length of the descriptions calculated earlier will be used as a parameter value. The program also takes clean and tokenized data as input.
6. The program creates a sequence creator to predict the next word based on the previous word and feature vectors of theimage.
7. The program can use the development dataset to monitor the model's performance and decide when to save the model version to the file.
8. Several models are saved, out of which the final one will be used for testing in the future.

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

Figure 11: Model under Training

E. Testing the model:

1. To perform testing, we can either create a separate Python notebook or use the same one as before. In either case, we will load the trained model that was saved in the previous step and generate predictions.
2. At this stage, the sequence generator and the tokenizer file we created will come into play.
3. We will perform the primary step of feature extraction for the particular image being observed.
4. We will manually pass the path of one of the remaining 2000 test images to the function.Alternatively, we can iterate through the test data set and store the prediction for each image in a dictionary or a list.
5. The actual process of generating images involves using the start and end sequences, and recursively calling the model to generate meaningful sentences.

```
Dataset:  6000
Descriptions: train= 6000
Photos: train= 6000
Vocabulary Size: 7577
Description Length:  32
Model: "functional_5"
```

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	[(None, 32)]	0	
input_6 (InputLayer)	[(None, 2048)]	0	
embedding_2 (Embedding)	(None, 32, 256)	1939712	input_7[0][0]
dropout_4 (Dropout)	(None, 2048)	0	input_6[0][0]
dropout_5 (Dropout)	(None, 32, 256)	0	embedding_2[0][0]
dense_6 (Dense)	(None, 256)	524544	dropout_4[0][0]
lstm_2 (LSTM)	(None, 256)	525312	dropout_5[0][0]
add_14 (Add)	(None, 256)	0	dense_6[0][0] lstm_2[0][0]
dense_7 (Dense)	(None, 256)	65792	add_14[0][0]
dense_8 (Dense)	(None, 7577)	1947289	dense_7[0][0]

```
Total params: 5,002,649
Trainable params: 5,002,649
Non-trainable params: 0
```

Figure 12: Model details

V. RESULT / ANALYSIS

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

1. Path of Image 1:
Flicker8k_Dataset/111537222_07e56d5a30.jpg

Output:

```
start man in red shirt is climbing up sheer cliff end
<matplotlib.image.AxesImage at 0x7f01db8ad0b8>
```

Figure 13: Caption generated using deep neural network for input Image 1

2. Path of Image 2:
Flicker8k_Dataset/256085101_2c2617c5d0.jpg

Output:

```
start dog is running through the grass end
<matplotlib.image.AxesImage at 0x7f905ce334a8>
```

Figure 14: Caption generated using deep neural network for input Image 2

3. Path of Image 3:
Flicker8k_Dataset/3344233740_c010378da7.jpg

Output:

```
start man in black shirt and black shoes stands in front of traffic end
<matplotlib.image.AxesImage at 0x7f9062861e48>
```

Figure 15: Caption generated using deep neural network for input Image 3

Table: Comparison between original andpredicted values

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

VI. CONCLUSION

- Based on the results obtained we can see that the deep learning methodology used here bore successful results.

- The CNN and the LSTM worked together in proper synchronization, they were able to find the relation between objects in images.

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

VII. REFERENCES

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

2. S. S. Gudla and M. V. R. Murthy, "A Comprehensive Survey of Deep Learning Techniques for Image Caption Generation," International Journal of Advanced Research in Computer Science, vol. 9, no. 4, pp. 1-7, 2018.

3. A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3128-3137, 2015.

4. H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt,
C. L. Zitnick, and G. Zweig, "From Captions to Visual Concepts and Back," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1473-1482, 2015.

5. Q. Wu, C. Shen, L. Liu, and A. van den Hengel, "Image Captioning with an Intermediate Attributes Layer," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2626-2634, 2016.

6. K. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," Proceedings of the International Conference on Machine Learning, pp. 1597-1607, 2020.

7. S. Ebrahimi, N. Mansouri, and H. R. Rabiee, "Deep Learning-Based Image Captioning: A Comprehensive Review," Journal of Computer Science and Technology, vol. 36, no. 5, pp. 891-909, 2021.