International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT21736

Image Caption Generator Using Neural Networks

Sujeet Kumar Shukla[1], Saurabh Dubey[2], Aniket Kumar Pandey[3], Vineet Mishra[4], Mayank Awasthi[5], Vinay Bhardwaj[6]

[1-5]B. Tech. Scholar, Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab, India

[6]Assistant Professor, Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab, India

Article Info
Volume 7, Issue 3
Page Number: 01-07

Publication Issue :
May-June-2021

Article History
Accepted : 01 May 2021
Published : 05 May 2021

ABSTRACT

In this paper, we focus on one of the visual recognition facets of computer vision, i.e. **image captioning**. This model's goal is to come up with captions for an image. Using deep learning techniques, image captioning aims to generate captions for an image automatically. Initially, a Convolutional Neural Network is used to detect the objects in the image (InceptionV3). Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM) with attention mechanism are used to generate a syntactically and semantically correct caption for the image based on the detected objects. In our project, we're working with a traffic sign dataset that has been captioned using the process described above. This model is extremely useful for visually impaired people who need to cross roads safely.

Keywords : Convolutional Neural Network, Recurrent Neural Network, YOLO, Deep Learning

I. INTRODUCTION

In recent years, image captioning, or the generation of a natural language description of an image, has gotten a lot of attention. With research, it has emerged as a significant and challenging area. Image recognition is improving. It's fascinating because it has so many applications, including labeling large image datasets and assisting the visually impaired. It necessitates a level of comprehension far beyond that required for general object detection and image classification, and is thus regarded as a major undertaking. The field is a fusion of **Natural Language Processing** and Computer Vision models from modern **Artificial Intelligence.**

The two most common approaches to image captioning are top-down and bottom-up. Unlike the top-down approach, which begins with an image and then converts it to words, the bottom-up approach begins with words that describe the various aspects that are combined. Both approaches have problems

describing finer details and formulating sentences based on individual aspects.

In the human visual system, **Visual Attention** is a selective mapping process. It's critical for image semantic natural language description. As a result, the importance of the topic is frequently discussed rather than the entire thing. The visual attention approach to image captioning will be discussed. To accomplish this, we used an attention-based model. Because the largest available dataset only has around two million labeled individual data, over fitting the training data is extremely difficult. This information must be put to use for:

- Feature Extraction and Making models for syntactical captions.
- Label for Correlations
- Syntaxial Understanding of Captions

The over fitting problem arises from the model remembering the inputs and using captions that sound similar for images with minor differences in specific details, such as a person on a ramp with a skating and a person on a table with a skating.

II. METHODOLOGY

For the captioning of images with traffic signs, we trained a visual attention-based model. This model can also be used for general captioning, which may necessitate a larger image dataset. The images are first pre-processed by our model using the ***InceptionV3*** pre-trained model. The final output layer is then passed through some convolution layers before being processed to produce the features with the required dimensions. At the same time, the captions of each image are tokenized, and start and end tokens are appended to each sentence.

Finally, the model is trained with each image and the real captions that go with it. The captions for the test images are predicted using the final weights. These captions are created word by word rather than in batches. Every word's prediction necessitates the probability of that word's occurrence based on the preceding words. This method proved to be quite effective, and we propose it based on our findings. The model architecture is shown in the below image.

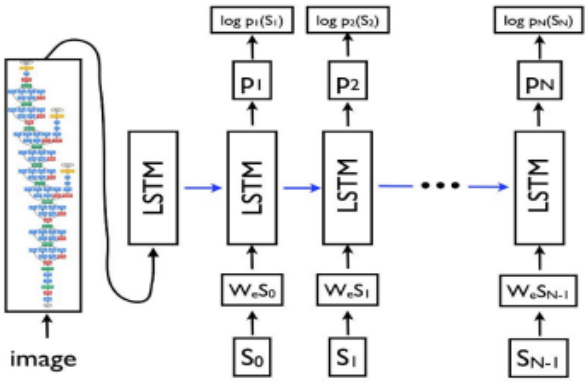

Fig.1- Proposed Architecture for the model

III. CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (ConvNets or CNNs) are a type of **Artificial Neural Network** that has shown to be very good at image recognition and classification. Object detection, self-driving cars, image captioning, and other tasks have all made extensive use of them. Yann Lecun discovered the first convnet in 1990, and the model's architecture was dubbed the LeNet architecture. The diagram below depicts a basic convnet.

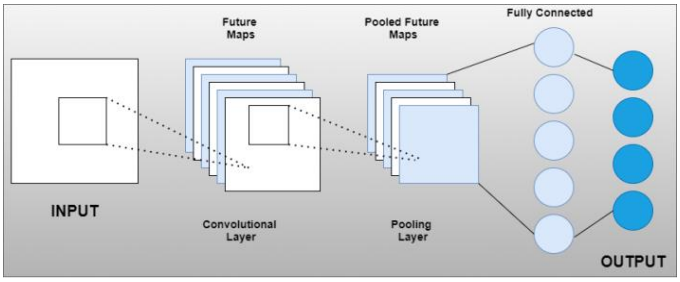

Fig. 2- Working of Convolutional Neural Network

Convolution, Non-Linearity (ReLU), Pooling or Sub Sampling and Classification are the four main operations that can be used to explain the entire architecture of a convnet (Fully Connected Layer).

Because these operations are the fundamental building blocks of every Convolutional Neural Network, understanding how they work is critical to gaining a thorough understanding of ConvNets. Below, we'll go over each of these operations in detail. Every image can be represented as a matrix of pixel values in essence. An image from a standard digital camera will have three channels: red, green, and blue. Imagine three 2d-matrices stacked on top of each other (one for each color), each with pixel range from 0 to 255.

A. Mask R-CNN (Mask Region-based Convolutional Neural Networks)

Mask Region-based CNN in terms of instance segmentation, Convolutional Neural Networks are the newest technology. For reference, there are numerous papers and tutorials with high-quality open-source code. Mask RCNN is a deep neural network designed to solve the problem of instance segmentation in machine learning and computer vision. To put it another way, it can be used to distinguish between different objects in an image or video. It takes an image and returns the output bounding boxes for the objects, as well as their classes and masks.

Mask RCNN is divided into two stages. First, it generates suggestions for regions where an object could be supported by the input image. Second, it predicts the output category, refines the bounding box, and generates a mask for the object at the pixel level, thereby confirming the primary stage proposal. Both stages are linked to ResNet, which serves as a backbone structure.

Let's look at the first stage: a light-weight neural network known as the **Region Proposal Network (RPN)** scans all **Feature Pyramid Network (FPN)** top-bottom pathways (hereinafter referred to as the feature map) and proposes regions that can contain objects. We'd like a way to bind features to their raw image location while scanning the feature map efficiently. The anchors are approaching. Anchors are a collection of boxes with pre-determined locations and scales in relation to images. Individual anchors are assigned ground-truth classes (only object or background binary classified at this time) and bounding boxes based on some Intersection over Union value. Because different scale anchors bind to different levels of the feature map, RPN uses these anchors to figure out where on the feature map an object should go and how big its bounding box should be.

We can agree that convolving, down sampling, and up sampling would keep features in the same relative locations as the objects in the original image and would not jumble them up. Another neural network assigns the proposed regions from the primary stage to many specific areas of a feature map level, scans these areas, and generates object classes, bounding boxes, and masks in the second stage. The procedure resembles RPN in appearance. Stage two used a trick called Region of Interest Align to locate the relevant areas of the feature map without the use of anchors, and there's a branch that generates masks for every object at the pixel level.

B. YOLO (You Only Look Once)

The best available object detection algorithm that works in real-time is “You Only Look Once,” abbreviated as YOLO. Traditional object classification can be accomplished with a simple neural network, but object detection in a specific scenario is a different storey. It necessitates a unique approach, as traditional classification algorithms rely on predefined image dimensions, whereas real-time object detection necessitates the system or model scanning the entire image for object recognition.

YOLO not only solves the problem, but it also provides bounding boxes for the trained objects.
YOLO (You Only Look Once) is an excellent tool for real-time detection. It has a total of 26 layers, with 24 convolutional layers and 2 dense layers. It is based on the Darknet architecture, which was created by the YOLO paper's first author.

The algorithm divides any given image into a *N x N* grid, which is then sent to the neural network, which detects the presence of any objects in any of the grids and generates bounding boxes with probabilities and labels for the corresponding objects. *YOLOv2* and *YOLOv3* were later modified versions of YOLO, with 30 and 106 layers, respectively. They were tweaked to make more precise predictions that could work with even smaller and finer image details.

We used the **COCO** dataset to test a **YOLOv3** pretrained model, which was able to detect 80 different labeled classes. Later, we used our custom dataset of 790 traffic sign images to train the algorithm. Google Open Image Dataset v5 was used to create these images. We were able to train the model quickly because the data was pre-annotated. In a real-time image, our model was able to detect traffic signs.

The optimizer for this model was set to Adam, the batch size was set to 8, and the model was trained for 50 epochs with 88 steps per epoch.

IV. RECURRENT NEURAL NETWORKS

A **recurrent neural network** (RNN) has the ability to process arbitrary length sequences and is thus used to generate text sequences. We used a Shakespeare dataset for this model. A model trained to predict the next character in a sequence of characters from this data ("Shakespear") is given a sequence of characters ("e") from this data ("Shakespear"). By repeatedly calling the model, longer text sequences can be generated. The model demonstrates how a character-based RNN can be used to generate text.
It is trained on small batches of text and can produce a longer sequence of text with a logical structure. The Tensorflow library is used in conjunction with other libraries like numpy and os. After that, the dataset is downloaded and read. After that, the data is vectorized, which means the strings are mapped to a numerical representation before training. The model receives a sequence of characters as input, and we train it to predict the output. The text was split into manageable sequences using tf.data, and the data was shuffled and packed into batches.

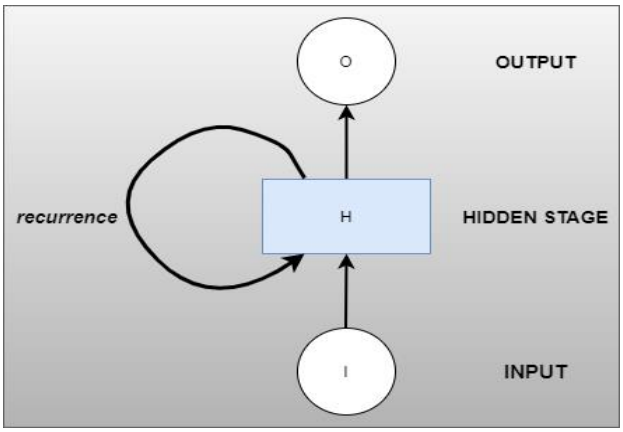

Fig 3- Working of Recurrent Neural Networks

GRU is used to construct the model. Sparse categorical cross entropy and Adam were used as the loss function and optimizer, respectively. After that, the model is trained over a set number of epochs. The model is then used to generate the text from the Shakespeare data after it has been trained.

Hyperparameters are variables that determine the structure of a network and how it is trained (predefined). Sequence length, batch size, buffer size, embedding dimension, RNN units, optimizer, epochs, and number of char to generate are the hyperparameters in this model.

The process of tuning the above parameters in order to optimize the model is known as hyperparameter tuning. The resource exhausted error was displayed when the sequence length, batch size, buffer size, embedding dimension, RNN units, and number of char were increased to a greater extent. And changing the optimizer to RMSprop reduces the model's accuracy, so we use the Adam optimizer because it provides the best accuracy for this model. The greater the number of epochs, the greater is the accuracy.

V. RELATED WORK FOR MODEL BUILDING

Caption generation is a type of artificial intelligence problem in which a textual description for a photograph is required. Try to give a caption for the image below.

Caption generation is a type of artificial intelligence problem in which a textual description for a photograph is required. Try captioning the image below.

“A dog on grass with some pink flowers,” a human might write as a caption. Other descriptions may exist, but they all mean the same thing. It's so simple for us, as humans, to look at a picture and describe it in a language that makes sense. But how does a computer programme come up with an appropriate caption for an image?

At this point, deep learning enters the picture, with two models: CNN and RNN being used to caption the images. The objects in the image are detected using CNN, and the captions are generated using RNN. Both of these models' working models were previously shown. In order to train our model, we used the MS-COCO dataset. The dataset contains approximately 82,000 images, each with at least five different caption annotations.

Transfer learning is a machine learning technique in which a model created for one task is used as the basis for a model on a different task. To preprocess and classify the images, we use InceptionV3 (which is pretrained on Imagenet). The output is cached to disc after preprocessing, and the captions are tokenized, with the vocabulary size limited to 5000 words to save memory, and the remaining words replaced with UNK tokens. The features are extracted from InceptionV3's lower convolutional layer, giving us a shape vector (8, 8, 2048).

The **CNN Encoder** (which is made up of a single fully connected layer) is then used to process this vector. The **RNN** (in this case GRU) then looks at the image and predicts the next word. The model is then trained by extracting the features stored in the.npy files and then passing those features through the encoder. The decoder receives the encoder output, hidden state (initialized to 0), and decoder input (which is the start token). The decoder returns the predictions as well as the hidden state of the decoder. The hidden state of the decoder is then passed back into the model, and the loss is calculated using the predictions. To determine the next input to the decoder, use teacher forcing.

The target word is passed as the next input to the decoder in a technique known as teacher forcing. The model must then be back-propagated after the gradients have been calculated and applied to the optimizer.

VI. EXPERIMENTAL RESULTS

The below image shows how Mask R-CNN works on an image and objects are separately detected in form of mask of pixels.

Fig. 4- Working Demo of Mask R-CNN

Below is the output snap of model generator with epochs shown in figure

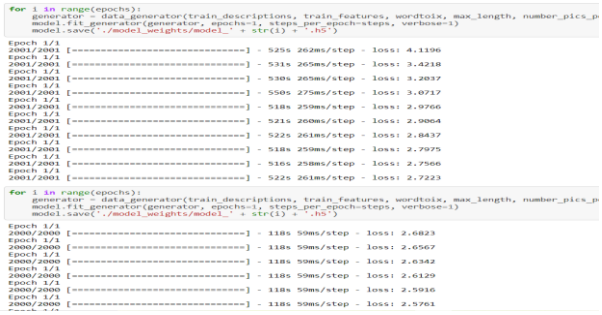

Fig. 5. The result of Image Captioning model using greedysearch function is shown below

VII. CONCLUSION

We created a model to caption the images in our project. Our model was also expanded by converting our captions to speech. We hope to continue working on voice synthesis in the future. We conducted research to fully comprehend our models and then executed each one separately. We learned how deep learning techniques work and how to build models using them. The dataset contained 82000 images with 5 captions for each image. For better accuracy our model can be trained for bigger dataset. While running the model and working with our datasets, we encountered numerous difficulties. However, as time went on, we learned how to correct our errors and create a more efficient model.

VIII. REFERENCES

[1]. R. Gerber and H. Nagel. Knowledge representation for the generation o f quantified natural language descriptions of vehicle traffic in image sequences. In ICIP. IEEE, 1996.

[2]. S. Hochreiter and J. Schmidhuber. Long short - term memory. Neural Computation, 9(8), 1997.

[3]. O. Vinyals, A. Toshev, S. Bengio and D. Erhan," Show and tell: A neural image caption generator," In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3156- 3164, 2014.

[4]. L. Yang, K.D .Tang, J. Yang and, L.J. L, "Dense captioning with joint inference and visual context". In: CVPR, pp 1978–1987, 2017.

[5]. J. Chen, W. Dong and M. Li, "Image Caption Generator using Deep Neural Networks", March 2018.

[6]. J. H. Tan, C. S .Chan and J. H. Chuah, "Image Captioning with Sparse Recurrent Neural Network, arXiv: 1908.10797,2019.

[7]. International Journal of Recent Advances in Multi disciplinary Topics, VOL. 2, NO. 4, APRIL2021

[8]. Andrej K, Li F-F Deep visual-semantic alignment for generating image descriptions.

https://cs.stanford.edu/people/karpathy/cvpr2015.pdf

[9]. Wang W, Hu H (2019) Image captioning using region-based attention joint with time-varying attention. Neural Process Lett 1–13

Cite this article as :

Sujeet Kumar Shukla, Saurabh Dubey, Aniket Kumar Pandey, Vineet Mishra, Mayank Awasthi, Vinay Bhardwaj, "Image Caption Generator Using Neural Networks", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7 Issue 3, pp. 01-07, May-June 2021.
Available at
doi : https://doi.org/10.32628/CSEIT21736
Journal URL : https://ijsrcseit.com/CSEIT21736