Ankush Govind Chavan, Kuldeepsingh Rajpurohit, Abhishek Kumar Singh, Rishabh Kumar, Mrs. Mansi Bhonsle

G H Raisoni College of Engineering and Management, Pune, Maharashtra, India

cankush625@gmail.com, kr890794@gmail.com, iabhikmr2000@gmail.com,

rishabh.kumar140298@gmail.com, mansi.bhonsle@raisoni.net

**Abstract**

# Inspired by the latest developments in Deep Learning based Machine Translation and Computer Vision based Object Detection have led to high accuracy Image Captioning models. Although these models are very accurate, these tend to rely on the use of expensive computation making it difficult to use these models in real-time applications, where applications can use them. In this paper, we carefully follow some of the heuristic strategies and core ideas of Image Captioning and its common methods and present our simple sequence to a sequence based implementation with a remarkable transformation and efficiency such as using beam search instead of greedy search that allows us to implement these on low-end hardware. The proposed system compares the results calculated using a variety of metrics with high-quality models and analyzes the reasons behind the model trained on the MS-COCO dataset that are lacking due to trade-off between computation speed and quality. In this proposed system,RESTful API endpoint will be created to be used on any device with an internet connection such as a mobile phone, IoT devices, clock, etc, this endpoint used to sent an image to the model running on remote server which in response will generate and sent an caption describing the objects and their relationship with each other in image in a natural language.

**Keywords**: Neural Networks · Assistive Vision · Caption Generator · Deep Learning · Restful API · Optimization

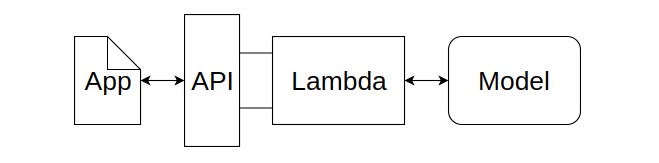
1. **Introduction**

Automatically defining image content and their relationships or actions is an important issue for artificial intelligence that connects computer vision and natural language processing. But this can have a profound effect on helping blind people to better understand their surroundings. These pictures can be used to produce captions that can be read aloud to the visually impaired so that they can better understand what is happening around them. This proposed system provides an API endpoint which uses generative model based on a deep recurrent architecture that incorporates the latest advances in computer vision and machine translation and that can be used to create natural sentences describing a previously captured or camera-captured image. The model is trained to increase the chances of interpreting the sentence using the Maximum Likelihood Estimation (MLE) given the training image. What is most impressive about this is that it is a single end model that can be described as predicting captions, given a picture, instead of requiring sophisticated data preparation or a pipeline of specifically designed models.

Not only must the model be able to solve the computer vision challenges of identifying objects in the image, but it must also be smart enough to capture and express object relationships in the natural language. For this reason, image caption generation is considered a serious problem for long. Its purpose is to mimic a person's ability to understand and process large amounts of visual information in descriptive language, making it an attractive problem in the field of AI.

1. **System Architecture**

In this proposed system, we are creating a RESTful API with a single endpoint that will be used to provide an image to the Image Captioning model running on the server. For creating an API, we will use AWS API Gateway Service and AWS Lambda. AWS Lambda will have a function to send the image received from the API request to the Image Captioning model on the AWS Sagemaker.



**Fig. 1** System Architecture

The starting point of this system will be an application that can run on any platform like a mobile phone, smartwatch, or any IoT devices. This application will send an image through the API request to the AWS Lambda function. The picture that is sent by the application will be a pre-captured image or an image captured by the camera device. The AWS Lambda function will be responsible for transferring this image for further processing to the Image Captioning model where the image is processed, and an appropriate caption describing that image will be generated. This caption will be in a text format. The API response will send this caption back to the application, and this caption will be spoken out loud by the device. Also, the caption will be displayed on the device screen, if that device has a screen.

1. **Datasets**

For solving this problem, there are many open source datasets available like MS COCO (containing 180k images), Flickr 30k (containing 30k images), Flickr 8k (containing 8k images), etc[3]. But for the purpose of this proposed system, we have used MicroSoft's COCO dataset. This dataset contains approximately 180000 images with 5 captions each.

1. **Data Cleaning**

When dealing with text, we often do some basic cleaning, such as inserting lower-case of all the words, removing special tokens, removing words that contain numbers.

Create a glossary of all the unique words that are present in all the 180000\*5 (i.e. 900000) image captions (corpus) in the dataset. As we are building a prediction model, we do not want all the words in our vocabulary but words that may occur frequently or may be common. This helps the model to be more powerful for outliers and makes fewer mistakes [2].

1. **Data Preprocessing**
   1. **Data Preprocessing - Images**

As we have images with different sizes, we will convert every image into a fixed-sized vector that can then be fed as input to the neural network. For this purpose, we will go for transfer learning by making use of the InceptionV3 model that is the Convolutional Neural Network created by Google Research.

For performing image classification in 1000 different classes of an image, this model was trained on the Imagenet dataset. But we not only want to classify the images but we want to get a fixed-length informative vector for each image. This will be done using a process known as automatic feature engineering[4].

For achieving this, we will neglect the last softmax layer from the model and extract a 2048 length vector (bottleneck feature) for every image.

* 1. **Data Preprocessing - Captions**

Prediction of the complete caption does not happen at once. The caption will be predicted word by word. That's why we have to encode every word into a fixed-size vector. For representing every unique word in the vocabulary, we will use an index in the integer form.

1. **Model Overview**

The model is described in three parts:

* 1. **Encoder**

This is a VGG16 model that is pre-trained on the ImageNet dataset. By pre-processing the photos with the VGG model and removing the output layer, we will use the extracted features predicted by this model as input. We have used VGG16 as an encoder because VGG won the image classfication challenge in ILSVRC and so we can harness the state-of-the-art feature extraction capabilities of this model.

* 1. **Sequence Processor**

This is a word embedding layer followed by a Long Short-Term Memory (LSTM) recurrent neural network layer. This layer is for handling the text input.

* 1. **Decoder**

Both the element extractor and sequence processor yield a fixed-length vector. To make a final prediction, these are consolidated and processed by a Dense layer.

1. **Sample Working**

In the first round, we send the image vector and the first word (‘begin’) as inputs to the Sequence Processor and predict the second word, i.e.:

Input = Image\_1\_feature\_vector + ‘begin’;

Output = ‘This’

Then again, we provide the image vector and the concatted first two words as inputs and predict the third word, i.e.:

Input = Image\_1\_feature\_vector + ‘begin This’;

Output = ‘elephant’

And so on…

One image and one caption is not a single point of data but multiple data points depending on the length of the output (description / caption). For all data points, it is not just the image that goes as an entry in the system, but also the captions that are part of it that helps predict the next word in sequence.

1. **Using Data Generators**

In our training data, we have about 150000 images, each with 5 captions. This makes a total of around 750000 pictures and captions. Although we assume that each caption on average is 7 words long, it will result in data points which is equal to 750000 \* 7 i.e. 5250000.

Now even if we assume that one block takes 2 bytes, so, to keep this data matrix, we will need more than 92 GB of memory. This is a huge demand, and even if we can load this large amount of data into RAM, it will make the system run much slower. For this reason, we are using data generators.

We do not need to store all the data in one memory at a time. Even though we have a current set of points in memory, it is enough for our purpose. Data Generators are a traditional Python operation. The function of the generator in Python is used for this purpose. It is like an iterator re-operating from where it left off last time it was called.

1. **Hyper Parameter Tuning**

The model trained for 30 epochs with a starting learning rate of 0.001 step size and 3 images per batch (batch size). However, after 20 times, the learning level was reduced to 0.0001 and the model was trained in 6 images per set.

This usually makes sense because during the most recent stages of training, as the model goes on to meet convergence, we have to lower the level of learning in order to take small steps towards the minima. And increasing the batch size over time helps your gradient updates to be more dynamic and powerful.

1. **Model of Proposed System**

In this system, we have used the sequence to sequence to create an encoder-decoder architecture. The encoder is a pre-trained InceptionV4 Convolutional Neural Network and decoder is a Deep Recurrent Neural Network with long short term memory cells. Encoder InceptionV4 is used to convert raw images ***I*** into a Fixed length embedding ***F*** which represents the convolved features for the images. This embedding is obtained by running a forward pass to the one before the last layer, i.e., the average pool layer of the InceptionV4 model pool. The decoder in our model has two phases, named, training and inference. The decoder is responsible for learning word order given the convolved features associated with the original captions. The hidden state of the decoder **ht** is initialized using these image embedding features ***F*** in timestep ***t=0***.. Thus the basic concept of the encoder-decoder model is shown by the following equations [1].

F = encoder(I); Xt=0 = F; Ot = decoder(Xt:0 → t)

The RNN training process with LSTM Cell-based decoder works on a probabilistic model where the decoder increases the chances of a word ***p*** in captions given a convolved image features ***F*** and previous words *Xt:0 → t*. To learn the entire sentence of length ***N*** corresponding to the ***F*** features, the decoder uses its repetitive nature to loop over itself over a constant number of timesteps ***N*** with the previous information (features and words sampled in timestep ***t***) stored in its cell memory as a state. The decoder can change *Ct* memory as it unrolls by adding a new state, refreshing or forgetting previous state with LSTM forgetting *ft* , input *it*, and output *ot* memory gates.

ft = σ (Wf . [ht – 1, xt] + bf) (1)

it = σ (Wi . [ht – 1, xt] + bi) (2)

ct = σ (Wc . [ht – 1,xt] + bc) (3)

Ct = ft \* Ct-1 + it  \* ct (4)

ot = σ (Wo . [ht-1, xt] + bo) (5)

ht = ot \* tanh(Ct) (6)

Ot = argmax(softmax(ht)) (7)

σ → sigmoid; Ot → Output word; tanh → hyperbolic tangent; Wo,Wf,Wi → Learnable Weight Vector; bo,bf,bi → Learnable bias Vector;

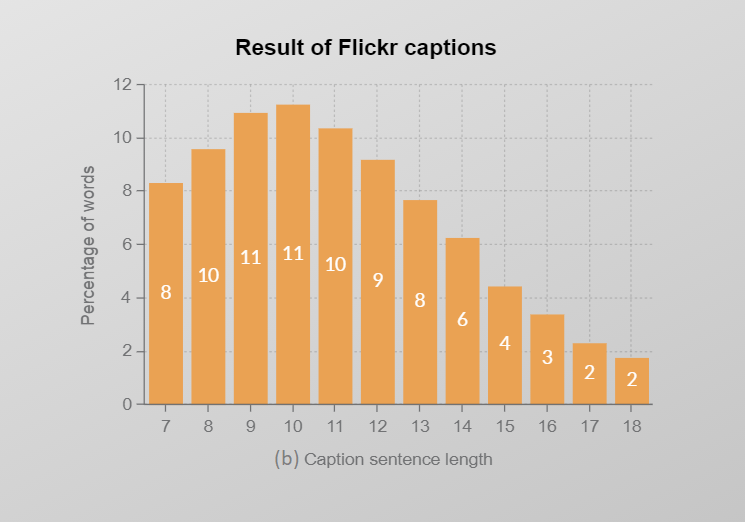
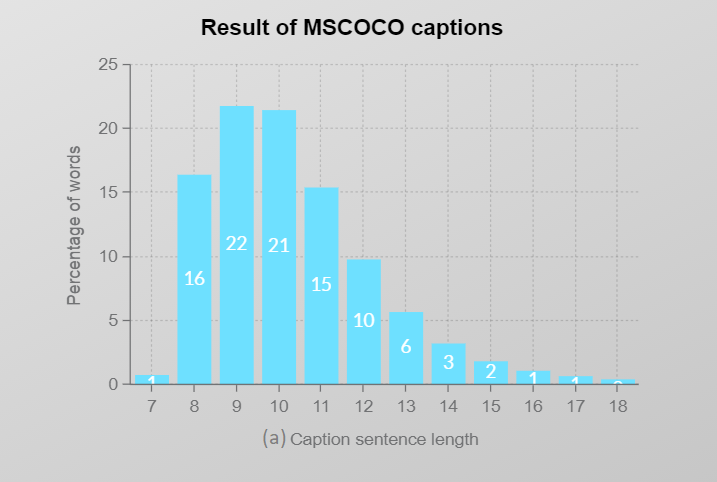
1. **Training**

For offline evaluation, compared to other Caption Bots our implementation will use Batched data, supports CNN finetuning, uses TensorFlow, and runs on a GPU[5]. All of these together will increase the training speed of the Language Model to a greater extent(~100x). Even if the split of 5000 images is not a standardized split, many researchers have been using it for reporting their results.

1. **Result** 
   1. **Datasets**

For evaluating the performance of our methods, we are using the MS COCO[8], Flickr8K[9], and Flickr30K[10] datasets. They are the most popular and efficient datasets for measuring the accuracy of the generated information.

Table 1 shows the detailed comparisons of reference captions on the three datasets above. The MS COCO dataset is a large-scale object detection, segmentation, and captioning dataset. The official version of MS COCO dataset includes over 82000 training images, over 40000 validation images, and over 40000 test images. Since the “Karpathy'' split is the most commonly used split method for reporting results, we use it to split the official MS COCO dataset to obtain around 113,000 training images, 5000 validation images, and 5000 test images. The Flickr8K dataset comes with an officially split of 6000 images for training, 1000 images for validation images, and 1000 images for testing. Without any official split, the Flickr30K dataset has 31,783 images that we will split into 25,000 training images, 2000 validation images, and 3000 images for testing.



**Fig 2.** Percentage of words per caption sentence length

**(a)** shows the statistical results of reference captions on MS COCO dataset; **(b)** shows the statistical results of reference captions on Flickr8K and Flickr30K datasets. The x-axis represents the length of each caption sentence.

* 1. **Evaluation Metrics**

We test the effectiveness of our method with many well-known metrics using image captions, including CIDEr, CIDEr [11], SPICE [12], METEOR [13], ROUGE-L [14], and BLEU [15].

CIDEr and SPICE are both human consensus metrics. CIDEr can be used to measure the similarity between a generated caption and a set of definitions written by humans. In addition, SPICE is a conditional metric, which is used to test how well a structured sentence captures objects, attributes and relationships between them. METEOR calculates sentence level similarities according to the harmonic definition of uni-gram recall and precision. ROUGE-L can be used for gisting evaluation. Its scores are calculated by measuring the number of scattered units, such as n-gram, word order, and the alteration of words between generated captions and human writing sentences. BLEU is a metaphor that is widely used in machine translation operations. It is based solely on the cohesiveness of the n-grams.

1. **Conclusion**

The results above show that the algorithm we had used fits accurately to our proposed system and gives better results than the previous methods used for image caption generation.

1. **References**
2. Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan (2015) Show and tell: A neural image caption generator. CVPR 1, 2
3. K. Xu (2016) Show, attend and tell: Neural image caption generation with visual attention. inProc. Int. Conf. Mach. Learn.
4. Andrej Karpathy, Li Fei Fei (2015) Deep Visual-Semantic Alignments for Generating Image Descriptions. IEEE Transactions on Pattern Analysis and Machine Intelligence (April 2017), vol 39, issue 4:664–676.
5. [Image Captioning with Keras teaching computers to describe pictures](https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8#:~:text=8.,Data Preprocessing — Images&text=However%2C our purpose here is,is called automatic feature engineering.)
6. Ayush Yadav, Rutgers The State University of New Jersey (2017) Camera2Caption: A real-time image caption generator.
7. Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan (2016) Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge PAMI
8. Wei, Y.; Xia, W.; Huang, J.; Ni, B.; Dong, J.; Zhao, Y.; Yan, S. CNN: Single-label to Multi-label. arXiv 2014, arXiv:1406.5726.
9. Lin, T.; Maire, M.; Belongie, S.J.; Bourdev, L.D.; Girshick, R.B.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. arXiv 2014, arXiv:1405.0312.
10. Hodosh, M.; Young, P.; Hockenmaier, J. Framing Image Description As a Ranking Task: Data, Models and Evaluation Metrics. J. Abbr. 2013, 47, 853–899.
11. Young, P.; Lai, A.; Hodosh, M.; Hockenmaier, J. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Trans. Assoc. Comput. Linguist. 2014, 2, 67–78.
12. Vedantam, R.; Zitnick, C.L.; Parikh, D. CIDEr: Consensus-based Image Description Evaluation. arXiv 2014, arXiv:1411.5726.
13. Anderson, P.; Fernando, B.; Johnson, M.; Gould, S. SPICE: Semantic Propositional Image Caption Evaluation. In ECCV; Springer: Cham, Switzerland, 2016; pp. 382–398.
14. 13 Denkowski, M.; Lavie, A. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In Proceedings of the Ninth Workshop on Statistical Machine Translation, Association for Computational Linguistics, Baltimore, MD, USA, 26–27 June 2014; pp. 376–380.
15. Lin, C.Y. ROUGE: A Package for Automatic Evaluation of Summaries. In Proceedings of the ACL-04 Workshop Text Summarization Branches Out, Barcelona, Spain, 25–26 July 2004; pp. 74–81.
16. Papineni, K.; Roukos, S.; Ward, T. BLEU: A method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 311–318. 20. Farhadi, A.; Hejrati, M.; Sadeghi, M.A.