Image Caption Generation using CNN and RNN

Gulshankumar Bakle
Oregon State University
Corvallis, Oregon- 97331
bakleg@oregonstate.edu

Yuanhao Zuo
Oregon State University
Corvallis, Oregon- 97331
zuoyua@oregonstate.edu

YuWen Tseng
Oregon State University
Corvallis, Oregon- 97331
tsengyuw@oregonstate.edu

Abstract

The field of artificial intelligence has seen a great spike of advancements in computer vision and natural language generation tasks. The task of image caption generation combines the abilities of these two sub-fields. In order to gain a higher understanding and insight about image captioning we implement previous state-of-the-art caption generators Show and Tell: Neural image captioning [1] and Show, Attend and Tell: Neural Image Caption Generator with visual attention [6]. This image caption generator is implemented using powerful deep learning library Pytorch. This model typically uses an encoder and decoder architecture, which takes an input image and produces a caption describing that image. Pretrained networks like ImageNet, ResNet or Inception networks are used as encoders. The caption decoder takes the image representation and outputs a textual description of it. The decoder is sequence model like Recurrent Neural Network (RNN), LSTM, or Gated recurrent Networks (GRUs). However, we only focus on Resnet as our pre-trained model and LSTM for sequence modeling to create a single model. Finally, we assess the quality of captions generated using various evaluation metrics like Bleu-n and METEOR. We successfully implemented all components of neural image caption generator with and without attention; and achieved a decent Bleu -1 score of 45 and Meteor score of 26 on flickr8k and MSCOCO datasets.

1. Introduction

Recent advancements in deep learning tasks, and availability of large-scale datasets for training networks in these tasks, has resulted into a remarkable surge of producing more and more intelligent systems. Image Captioning is one such application, which has seen improvisations with new state of art models being proposed every now and then. Image captioning can be viewed as process of extracting the spatial features present in images, associating the relationship between various entities in image and generating natural language for describing those relationships between detected features.

For this course project, we implemented an image caption generator using CNN and RNN based on this paper [1]. The applications of image captioning are many-folds ranging from social media to helping visually impaired people. An image caption generation is essentially a translation of image information into textual information. With artificial intelligence technology gaining popularity and increase in production of faster memory processing GPUs, image classification tasks have found its use in daily applications like facial expression detection, object detection and classification etc. Natural Language Processing has also gained great advancements, with the ability of various variants of RNNs like Long Short-term Memory [6], Gated Recurrent units [8] and peephole GRUs to associate linkage between long texts, has led to significant improvements in language to language machine translations. Even if "Speaking with pictures" is easy for a human, it is very challenging for machines. It not only needs to use models to understand the content of pictures but also needs to use natural language to express the relationship between them.

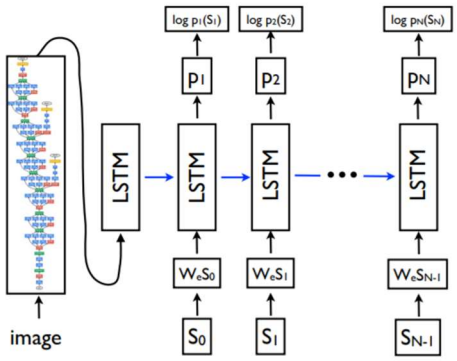

Figure 1. Neural Image Caption generator

Figure 1 describes Neural Image Caption Generator described in the paper [1]. The model explores ability of CNNs to detect the features from images. This is achieved by using pre trained models like ResNets [9], Inception network [10], ImageNet [2] for transfer learning. The fully connected layers at the end of pretrained model encodes all

the features present in images, which are then fed as input to sequence model like LSTMs or GRUs to produce word by word caption. The ability of these sequence models in machine translation tasks [16] can also be used to produce meaningful captions describing the relationships between the objects in images.

The rest of the paper is organized as follows. In section 2, we provide an overview of the related work being done in this field. In section 3, we describe the overall methodology and architecture of the implemented model in detail. In section 4, we discuss various experiments that we performed using different combinations of parameters. In this section we also describe the results in the form of different evaluation metrics, and captions generated on test samples. Finally, in section 5 we conclude our implementation with possible future work.

2. Related work

There are several models proposed related to the image caption generator. In [2], P. Sermanet, D. Eigen, et al. describe a well- studied method for large-scale object recognition and image classification projects. Also, they provide a detailed analysis of the current technology of image classification and object detection and make a connection between the computer vision accuracy with human recognition accuracy to get a promoted result. While Li et al. [3] apply several phrases which briefly describe the detected objects and the relationships between objects into their model and get the ultimate image description through combining their feature vectors together. Obviously, this approach has an overwhelming advantage over P. Sermanet' [2] method. It not only exhibits the detected objects themselves but also expresses how these objects relate to each other. Besides, in the work of Kiros et al. [4], a feedforward neural net has been applied to predict the context for a given image in their work. Also, in the work by Mao et al. [5], the authors use a recurrent neural network and then calculate the maximum likelihood for the result to achieve the prediction task. Finally, in [1], Vinyals, Oriol, et al. create a model of combining deep convolutional nets for image classification with LSTM networks, then for a given image, generate image caption word by word corresponds to the maximized probability of the current description. This model is called the neural image caption generator, NIC, that produces descriptions and the relationship between the detected objects in images. In the whole process, NIC first takes an image *I* as input and trains it to maximize the likelihood of each word in sentence, denoted by p(S|I), where S is the probability a particular word. This model when implemented on datasets like flickr8k, flickr30k and MSCOCO has shown a Bleu score of 50-60. This model saw major improvements in results, when it was combined with attention mechanism as proposed by Xu, Lei Ba, et al. [6]. The current state of art being transformers [11], which have shown significant rise in Bleu scores with the ability to inherit long term dependencies in natural language generation tasks like machine translations and image captioning.

3. Methodology and Architecture

There are many datasets which are widely used in various computer vision tasks: Flickr8k [17], Flick30k [20], MS COCO [19] etc. We decided to use flickr8k dataset primarily to produce initial results, but then also implemented the model with MSCOCO. Flickr8k contains around 8000 images, with 5 captions describing each image. The entire dataset is split into 3 sets: training set (6000 images), validation set (1000 insert) and testing set (1000 images). We used Andrej Karpathy's [12] split for training and testing purposes. The split ratio for MSCOCO was 83k, 41k and 41k for training, validation and testing purposes respectively.

The extracted features are processed as input to sequence to sequence model like Long Term Short Memory (LSTM) cells or Gated Recurrent Networks (GRU) to produce word by word caption describing the features in image. The entire pretrained CNN model functions as an encoder and sequence model as decoder. We first experimented the captions generated without using attention mechanism, and then added an additional attention model along with decoder to improve the results. Attention mechanism helps in focusing on specific features of images, to generate more meaningful captions. Each module is briefly explained in following sections.

3.1 Data pre-processing

We pre-processed the raw data into proper format, that was then fed into pre-trained CNN model for extracting the features from the images. Originally the dataset had captions associated with training, validation and testing images in a json file format. Each image had 5 distinct descriptions. For training purpose, we transformed all the images into the same dimensions of (224, 224, 3) RGB format. These images were then normalized to be within [0,1] with mean (0.485, 0.456,0.406) and the standard deviation of (0.229,0.224, 0.225) for faster training. While working with MSCOCO dataset, we used coco API [], which provides various functions for easy navigation within dataset using pycocotools. We used natural language toolkit (nltk) [13] which is a widely used library in natural language generation tasks. We produced our own dictionary comprising of all the words in lower case present in the captions. All kinds of symbols and meaningless entities were ignored. Initially the dictionary was of huge size, as it contained many repeated and helping words, so we added a vocabulary threshold to limit the size. Each word, if it

appeared threshold times in captions, we included into the dictionary. Each word in a caption is termed as a token. Three additional tokens- <start>, <end>, <unk> tokens were also added to the dictionary. Each word in the dictionary is encoded with numbers, which can be reversed back to English words while generating the captions.

3.2 CNN as an Encoder

A pre-trained CNN acts as an encoder in this model which takes all the images as input and produces a latent space vector encoding all the features present in the images. These encoded features are then fed as input to an RNN. The paper originally shows results using ImageNet [2] as a pretrained network. We used ResNet50 [9] as our pretrained network, present in Pytorch's torchvision library [15]. Since this is not an image classification task, so we altered ResNet50 by removing the last SoftMax, fully connected and max pool layers from the network. By the time we reach the second-to-last fully connected layer, it would have abstracted essential features of the image encoded. The input image has dimensions of (Nx3x224x224) and output has dimensions of (Nx14x14x512). Since the model is already trained, so we disable the gradient descent by setting "require parameters" to False.

3.3 RNNs as Decoder

Recurrent neural networks are widely used in machine translation, sentimental analysis and other sequential data problems. Since RNNs are often prone to vanishing gradient problems, we used Long Short-Term Memory (LSTMs) [6], its variant, as decoder. The ability of LSTM units to remember and forget the states, over the time steps, aids in generation of meaningful sentences describing the image [6]. Each LSTM unit contains memory cell, which encodes essential knowledge about what inputs have been observed till now, at each time step. The necessity of what type of knowledge must be remembered and given away is controlled by using various kinds of gates.

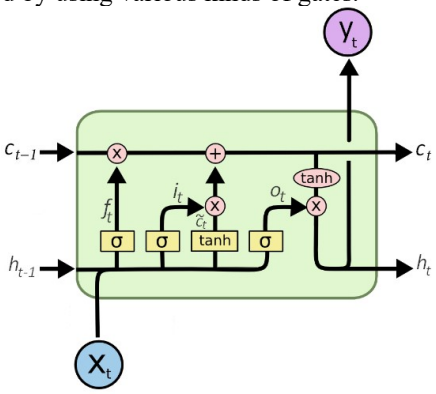

Figure 2. LSTM block with cell state c_{t-1} controlled by gates

The core structure of LSTM as shown in Figure 2 consists of three types of gates- input gate (i), forget gate (f) and output gate(o). The outputs given by these gates are applied multiplicatively, thus limiting the values between 0 and 1. A value of 0 indicates completely forget passed information from previous time step, while a value of 1 indicates remember it. The next step is supplying new information to be stored in cell state. This is achieved by using tanh and sigmoid functions on inputs and hidden states. This new information generated is now coupled with the outputs of forget layer, which is the new hidden cell state passed to next time steps. Finally, the output of LSTM unit will be a filtered version of cell state and input state. Mathematical notations of these gates is shown below:

$$f_t = \sigma(W_f \circ [h_{t-1}, x_t] + b_f) \quad (1)$$
$$i_t = \sigma(W_t \circ [h_{t-1}, x_t] + b_i) \quad (2)$$
$$C`_t = \tanh(Wc \circ [h_{t-1}, x_t] + b_i) \quad (3)$$
$$C_t = f_t * C_{t-1} + i_t * C`_t \quad (4)$$
$$o_t = \sigma(W_o \circ [h_{t-1}, x_t] + b_o) \quad (5)$$
$$h_t = o_t * \tanh(C_t) \quad (6)$$

The output of each LSTM unit is maximum SoftMax probability of a word, given the input as dictionary of words. Mathematically probability of next output word describing an image can be formulated as,

$$\log p(S|I) = \sum_{i=0}^{n} log\ p(Si|I, S0, S1, S2 \ldots Si-1) \quad (7)$$

Here term S_{i-1} denotes the words which are already predicted from previous timesteps, I is the latent vector of features. We maximize equation (7), by picking the word with highest probability from the dictionary.

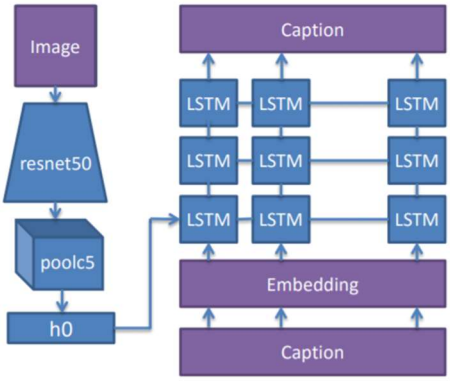

Figure 3. Neural Image Captioning without attention

The LSTM cells also use word embeddings as input which helps in association of different components in image as detected by the pretrained model. The size of word embedding vectors is equal to the dictionary length. Initially the word embeddings are mapped to random values, as they reach towards optimal values along the training process. The feature vector is fed to the decoder

part only once, but word embeddings are fed at all the timesteps. Overall architecture of encoder and decoder is shown in Figure 3.

The first word of the caption sentence generated will always be a <start> token denoting the start of the caption. This token is now fed as input to next LSTM timestep along with word embeddings. The LSTM units are trained to predict the words of sentence defined by p (S_i|I, S_0,S_1,…S_{i-1}) greedily. Here we have used greedy approach to generate captions, which produces words of sentence, in word by word fashion. The words are predicted until we receive the <end> token, which denotes the end of sentence. Figure 2 represents the overall architecture without attention mechanism.

3.4 Attention mechanism

After getting significant results using encoder-decoder architecture for image caption generation, we included additional attention module in the architecture as describe in this paper [6]. Attention mechanism has been able to perform significantly better in various machine translation and natural language generation tasks. It differs from normal encoder-decoder architecture by focusing on different spatial features of the image, while generating captions, instead of considering average of entire feature vector. Originally paper [6] shows implementation using both hard and soft attention, but we have limited the implementation of this project to soft attention. Figure 4 represents a basic working of attention mechanism with four hidden states from encoder, having four outputs from decoder and four attention weights.

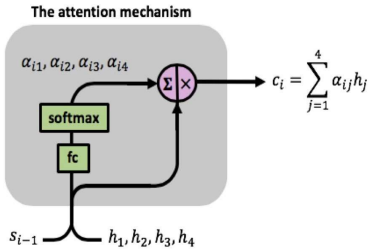

Figure 4. Simple attention module

$$\alpha_{ij}=\exp(e_{ij})/\sum_{i=1}^{4} \exp(si-1, hj) \quad (7)$$

where, $$e_{ij}=f_c(s_{i-1},h_j) \quad (8)$$

The attention block takes input as all the hidden states, along with previous cell states from output generated. The hidden states hi contains information about each feature of image. The attention weights are calculated using a fully connected layer denoted by fc. They are denoted by symbol α. Output from fc layer is limited between [0,1] using SoftMax function. Finally, on which part of image feature, the decoder should focus on every timestep is calculated using context vector c_i, which is multiplicative summation of attention weights and hidden states (feature vector from encoder in this case). Figure 4 represents just a basic working of attention with four hidden states and four attention weights.

Figure 5 represents working architecture of attention block with encoder and decoder. After generation of each word by LSTM cells, previous cell state from decoder is fed as input to attention block. The feature vector from encoder is used as input to calculate attention weights. Attention module generates context vectors, which help in focusing on spatial features of images, based on which decoder predicts the next word. It was observed that captions generated after including attention, were more meaningful and complete.

3.5 Loss Function

The output from our LSTM units is the maximum likelihood of each word of sentence from the dictionary. Thus, we used cross entropy loss as loss function during the training process. This is considered as most effective way for measurement of loss in classification task, holding probability values between 0 and 1.

$$L_{entropy}= -\sum_{n}^{N} t_k^n * \ln p_k^n \quad (9)$$

Equation 9 [14] shows the mathematical formulation of cross entropy loss, where N is number of classes, t takes values either 0 or 1, and p denotes the probability of particular class.

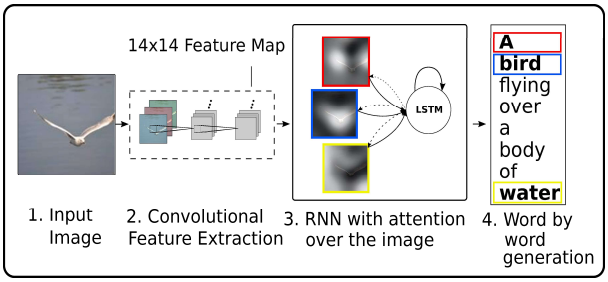

Figure 5. Encoder decoder model with attention

4. Experiments and Results

In this section we propose the combinations and results that were observed. We implemented the described model on flickr8k and MSCOCO datasets, using same pre-trained ResNet50 model. Since both datasets had similar kind of formatting, that is, each image associated with five distinct captions, we implemented a similar pre-processing procedure. A vocabulary list was generated using all the words from the training captions. For data wrangling in MSCOCO dataset, we used COCO API [19] which provides effective functionalities for navigation within the

data. Initially we implemented the architecture described in the paper [1], without attention mechanism. The training was effective as substantial drop in training loss was observed, but there were many instances where generation of weird caption was observed. So, we tried improving the performance using attention mechanism as described in this paper [6]. Just observing drop in training loss is not an efficient way to judge the efficacy of a natural language generation model. The paper uses various kinds of evaluation metrics like Bleu-1 (Bilingual Evaluation Understudy) [18], Bleu-2, Bleu-3, Bleu-4, Meteor, Cider etc., Here we evaluated the implemented model based on Bleu-1 and Meteor scores as principal evaluation metrics. These evaluation metrics are explained in detail below:

4.1 Metrics

Bleu scores are the widely used metric in language generation tasks. It uses an n-gram approach for comparing the quality of sentence generated, where n is the number of words that follow one another. For example, Bleu-4 denotes a 4-gram approach, where four words are considered as one context that follow each other. Here we have evaluated the implementation based on Bleu-1, that is a unigram approach. It just compares the individual words between the reference sentence and candidate sentences. METEOR (Metric for Evaluation of Translation with Explicit Ordering) [21] is another evaluation metric which is commonly used in image description tasks. Unlike Bleu-n score, Meteor score follows a unigram approach for evaluating the quality of sentence generated. It does not rely on n-gram matches but uses direct word ordering penalty and captures the fluency in text generated.

Perplexity is another evaluation metric that can be used to evaluate language model. It is a measurement based on how well the model can predict a sentence. Specifically, perplexity captures the degree of uncertainty a model has in predicting some text.

4.2 Results

All the programming was done using Pytorch framework. ResNet50 was used as pre-trained model, with few fully connected layers removed from it. The reason behind choosing ResNet50 being, it was faster to train and fit into GPU memories. Originally paper [1] has been implemented using ImageNet [2] and Inception V3 [10]as pre-trained model, which contains many convolutional layers as compared to ResNet50.

As mentioned in section 3.1, each word of the dictionary is encoded as integer number since training can only performed on numbers and not on words. Initially, the length of dictionary was too large, as it contained many repeated words and helping words, which do not add much relevance in sentences. So, we added a vocabulary threshold variable, that would limit the words with more relevance entering the dictionary. If a word appears threshold times in training captions, it was added in dictionary, else discarded.

On training the encoder-decoder architecture on flickr 8k dataset, for 20 epochs, with 64 as batch size and embedding size being 256, significant drop in training was observed. More better results can be achieved by using flickr30k dataset, and training for longer duration of time. Training for MSCOCO dataset was done only for 3 epochs and for over 6400 timesteps. Since it was a huge dataset, only 3 epochs took 12-14 hours of training. Table 1 describes the configuration of model which predicted fine captions describing images. Table 2 describes the configuration of model with attention mechanism. The captions generated were better as considerable improvement in Bleu and Meteor metrics was observed.

Framework	PyTorch
Batch Size	64
Embedding size	256
Hidden units	512
Number of epochs for flickr8k	20
Number of epochs observed for MSCOCO	3
Loss Function	Cross Entropy Loss
Optimizer	Adam
Vocabulary threshold	4

Table 1. Encoder-Decoder configuration without attention

Framework	PyTorch
Batch Size	64
Embedding size	256
Hidden units	512
Attention dimension	512
Number of epochs for flickr8k	20
Loss Function	Cross Entropy Loss
Optimizer	Adam
Vocabulary threshold	4

Table 2. Encoder-Decoder configuration with attention

Figure 6 and 7 show training and validation losses observed on flickr8k and MSCOCO datasets respectively. Though the training was done for less amount of time compared to original implementation of paper, we observed a fair amount of drop in training and validation losses.

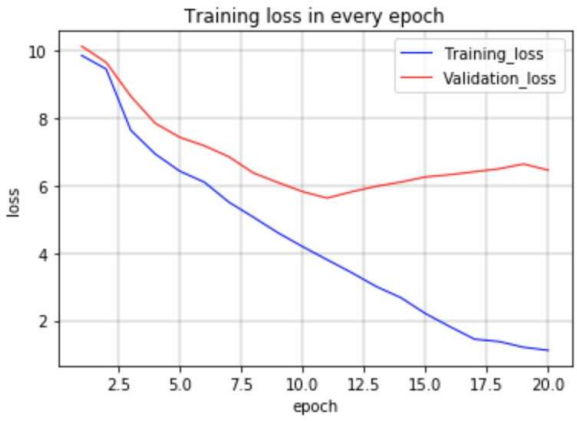

Figure 6. Training and validation loss for flickr8k

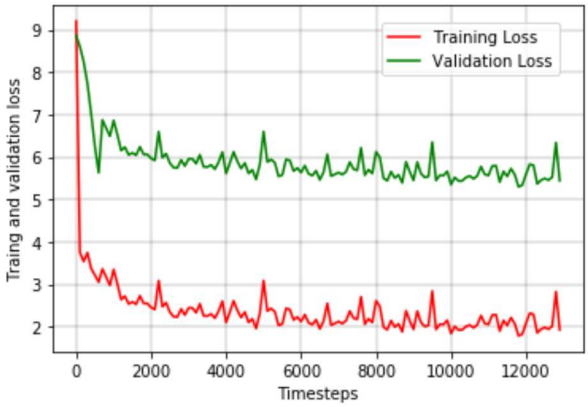

Figure 7. Training and validation loss for MSCOCO

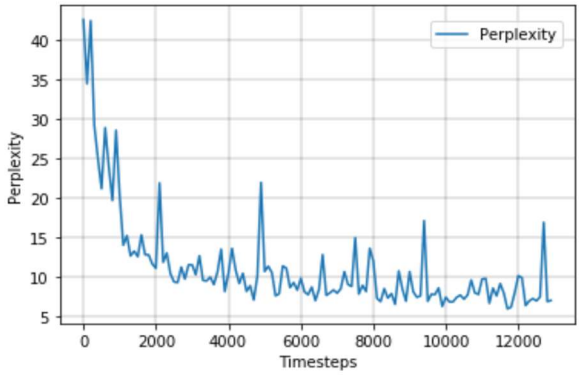

Figure 8. Perplexity for MSCOCO dataset

We observed a notable amount of rise and drop in values of perplexity as it can be observed from Figure 8. At initial timesteps, it was in counts of thousands, but gradually dropped to range 5 to 10 as we reached 12000 epochs. This drop was observed after 12 hours of continuous training of the model. This indicates that our model will now be able to generate meaningful sentences, given an image. The implementation of model has underperformed to some extent, as originally the paper achieved Bleu scores greater than 55 without attention, and with attention it gets more closer to 70. Though our implementation raised the Bleu score after adding attention mechanism, but we got results lesser than what paper has shown. This might be due to fact that originally the model has been trained for longer period, like for 3-4 days, as compared to ours, which was trained for 12-14 hours. Table 3 accounts the summary of result that we achieved.

Another reason can be held accountable, for lower values of Bleu scores, the outputs by our LSTMs are maximum likelihood of each word from dictionary, which is a greedy approach for generating sentences from images. The sentences could have been more meaningful, if a Beam Search approach for inference was used instead of greedy. Beam Search approach iteratively considers a beam size of *k* best sentences with highest probability, up to *t* timesteps and keep only k best resulting sentences out of them. Without using beam search did gave us less scores for Bleu and Meteor.

Model	Bleu-1	Meteor
Neural Image captioning [1] on flickr8k	37-41	24-26
Neural image Captioning on MSCOCO	42-47	20-24
Neural Image Captioning with attention on flickr8k	46-50	20-24

Table 3. Evaluation metrics

Figure 9, figure 10, Figure 11, Figure 12 represents captions predicted on test images from MSCOCO dataset. Tokens <start> and <end> token have been removed. Figure 15 shows an instance of generated caption based on attention. The highlighted part in the image, has higher attention weights, compared to rest of the features.

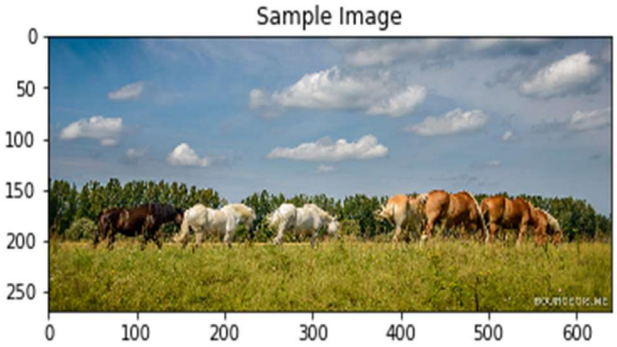

Figure 9. Original: A group of horses of various colors eat in a field.
Generated: A herd of cattle grazing on a lush green hillside..

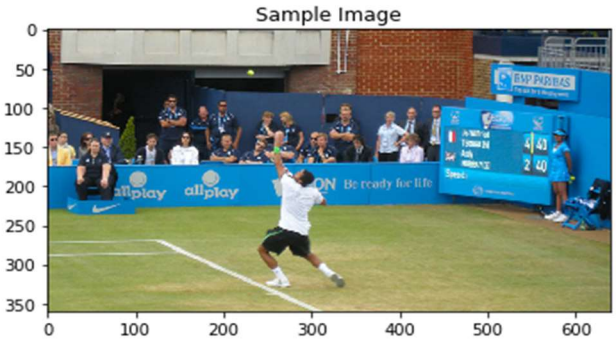

Figure 10. Original: A man hitting the tennis ball on tennis court.
Generated: A tennis player is playing tennis on the court.

Figure 11. Original: A man and woman and some people looking at them
Generated: A group of people standing around a table with a remote.

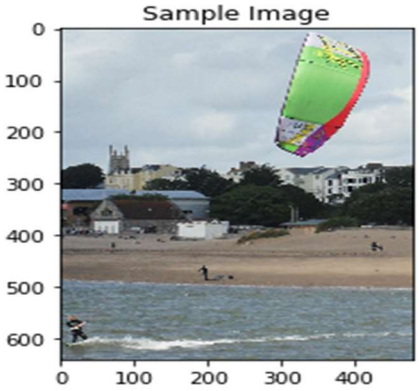

Figure 12. Original: A man is parasailing on the beach in Europe.
Generated: A man on a beach with surfboard…

We also observed some weird captions generated by encoder-decoder model, are shown with Figure 13 and 14.

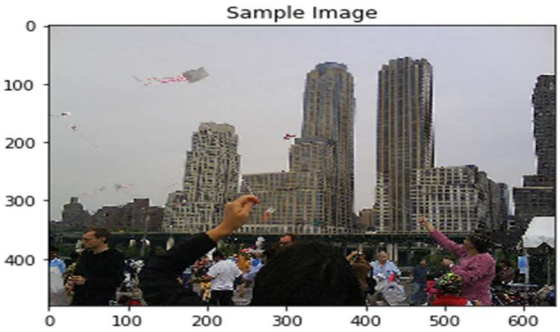

Figure 13. Original: A group of people flying kites
Generated: A group of people standing around a plane..

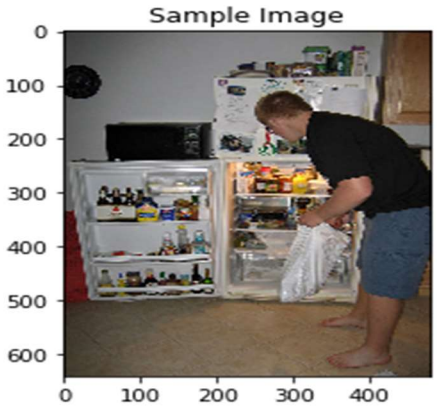

Figure 14. Original: A man bending over to look inside of a fridge.
Generated: A man in suite and tie standing in a room

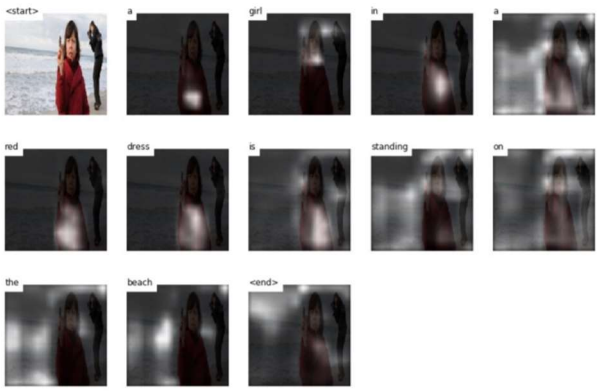

Figure 15. An example of caption generated using attention

5. Conclusion and Future Work:

Thus, we were successful in effective implementation of image caption generator with and without attention. There is always scope for producing more better results, if given

more time. Training models for longer epoch lengths, can reduce loss to much more extent. Using Beam search in caption generation can produce more meaningful sentences which can increase the Bleu and METEOR scores. We used already pre-trained network without tuning its parameters. Additionally, we could add more batch normalization layers and drop out layers to make our encoder even more susceptible to detect features. Changing the sequence model to GRUs could also show some varied results. Moreover, the current state of art – transformers [11] can also be implemented which incorporate multiple attention modules instead of one. Seeing so many possibilities in this sub field of deep learning ignites our interest to try these implementations and expect to explore more in coming future.

References

[1] Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[2] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115.3 (2015): 211-252.

[3] Li, Siming, et al. "Composing simple image descriptions using web-scale n-grams." Proceedings of the Fifteenth Conference on Computational Natural Language Learning. Association for Computational Linguistics, 2011.

[4] Kiros, Ryan, Ruslan Salakhutdinov, and Rich Zemel. "Multimodal neural language models." International conference on machine learning. 2014.

[5] Mao, Junhua, et al. "Explain images with multimodal recurrent neural networks." arXiv preprint arXiv:1410.1090 (2014).

[6] Xu, Kelvin, Jimmy, Ryan, Cho, Courville, … Ba. (2016, April 19). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Retrieved from https://arxiv.org/abs/1502.03044

[7] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735 Xu, Kelvin, Jimmy, Ryan, Cho, Courville, … Ba. (2016, April 19). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Retrieved from https://arxiv.org/abs/1502.03044

[8] Chung, Junyoung, Caglar, Cho, KyungHyun, Bengio, & Yoshua. (2014, December 11). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. Retrieved from https://arxiv.org/abs/1412.3555

[9] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/cvpr.2016.90

[10] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/cvpr.2016.308

[11] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, … Illia. (2017, December 6). Attention Is All You Need. Retrieved from https://arxiv.org/abs/1706.03762

[12] Karpathy. (2016, September 23). karpathy/neuraltalk2. Retrieved from https://github.com/karpathy/neuraltalk2

[13] Natural Language Toolkit¶. (n.d.). Retrieved from http://www.nltk.org/

[14] Loss Functions¶. (n.d.). Retrieved from https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

[15] torchvision¶. (n.d.). Retrieved from https://pytorch.org/docs/stable/torchvision/index.html

[16] Luong, T., Pham, H., & Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. doi: 10.18653/v1/d15-1166

[17] Hussain, S. (2019, March 24). Flickr8K. Retrieved from https://www.kaggle.com/shadabhussain/flickr8k

[18] Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2001). Bleu. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL 02*. doi: 10.3115/1073083.1073135

[19] Common Objects in Context. (n.d.). Retrieved from http://cocodataset.org/#home

[20] Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., & Lazebnik, S. (2015). Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models. *2015 IEEE International Conference on Computer Vision (ICCV)*. doi: 10.1109/iccv.2015.303

[21] METEOR: An Automatic Metric for MT Evaluation with ... (n.d.). Retrieved from https://www.cs.cmu.edu/~alavie/METEOR/pdf/Banerjee-Lavie-2005-METEOR.pdf