Image Captioning Using Deep Learning

Udit Jaitly
Department Of Computer Science
University Of Florida
Gainesville, Florida
uditjaitly@ufl.edu

***Abstract*—Image captioning is a very famous problem in the field of Deep Learning. The problem consists of feeding the image to a computer which in return generates a caption describing the image. However, this task can seem very normal to us, human beings, to come up with a caption after looking at an image. However, it's not that easy for computers to complete this task. Development and research in the domain of deep learning has enabled us to design deep neural networks which can carry out this convoluted task. The paper proposes one of such models which consists of an encoder, a sequence processor and a decoder that can carry out the task of image-captioning and offers decent performance. The performance of the model is evaluated by BLEU score metric.**

***Index Terms*—Convolutional Neural Networks, image captioning, BLEU score, LSTM, Natural Language Processing**

I. Introduction

Image Captioning can be defined as a procedure through which we can achieve a textual depiction of an image. The textual depiction of the image is called the caption. This process was a very difficult challenge in the field of computer science before the boom of the artificial intelligence field. Because of the emergence and development in artificial intelligence, we have very efficient algorithms and modern hardware that can carry out heavy computations which enable us to overcome the challenge of image captioning. However, there are multiple ways to create image-caption generation models but each model has it's own trade-off. There still does not exist an image-captioning model which neither over-fits nor under-fits and has a perfect prediction accuracy. I have proposed a model which by no means perfectly accurate and optimized but has an decent BLEU score which is one of the parameters used to evaluate a captioning model. The proposed model, like most deep learning models consists of an encoder,a decoder and a training phase and testing phase.

II. Motivation

Image captioning has many applications in the current day because it has plenty of relevance to the real world scenarios[2]. Few of the many applications are as follows:

- **Self Driving Cars:** Self Driving Cars have been said to be the future and they are not currently the routine. This is because of the human intelligence that the car agent lacks. Very accurate image captioning systems would definitely give boost to the self-driving agent.
- **CCTV Cameras:** CCTV cameras are a part of ubiquitous computing and are everywhere. These cameras could really help us if they could generate accurate captions based on frames they are capturing. For instance, detect a customer shoplifting at a supermarket and sending an alert.
- **Assisting Visually Challenged People:** Products can implement image captioning which generates captions of the environment around the disabled person and narrates it, making the visually challenged person aware of their surroundings.

III. Description

As discussed above, image captioning is one of the current challenges in field of Artificial Intelligence which has been solved but hasn't by no means, perfected. The basic idea and elements of my proposed image-captioning model is as follows:

A. *Model Definition*

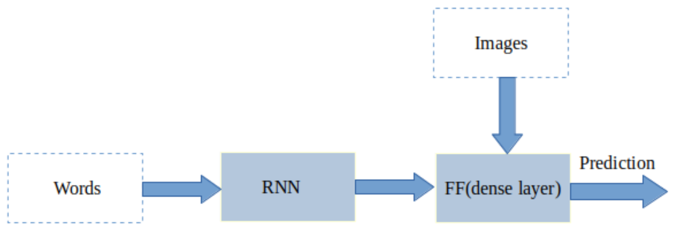


Fig. 1. Structure of the proposed model

1) Encoder: An encoder can be defined as a network which outputs a feature for some specific input. For instance, CNN, which stands for Convolution Neural Network, can simply be interpreted as an encoder. The CNN extracts the features for the given images as input. The output of the CNN acts as the input to the Decoder. For my caption generator, I am using a pre-trained VGG model which consits of 16 layers(Figure 2). This model was trained on the ImageNet dataset. I excluded the VGG output layer from this model because I only want to extract the features of the image and not classify the image.

2) Sequence Processor: It processes the input of text-format and also consists of a LSTM layer, which stands for Long Short-Term Memory. LSTM are designed in such a way that they can remember and retain information for longer periods of time hence they avoid the long-term dependency issue.

3) Decoder: A Decoder can be defined as a network which models the language up-to work to word level. RNN, which stands for Recurrent Neural Network can be interpreted as a decoder. The outputs from the encoder, that is the extracted features and from the sequence generator are fed into this dense layer for processing and prediction.

The encoder takes in image as the input as a 4,096 element vector.The output produced by the encoder is a vector consisting of 256 elements. This vector is a condensed representation of the image. The sequence processor takes as input the data in form of text(captions). The length of the word sequences is predefined as 34 words. After going through the embedded layer, it goes to the LSTM layer. The output produced by the decoder also consists of 256 elements. For the regularization used by the encoder and the sequence processor, the dropout rate is set as 50 percent. This is done so as to manage the bias-variance trade off.

Both the inputs are merged and fed into the decoder which consists of 256 neuron layers. A softmax prediction spanning over the output vocabulary is made for the following word in each sequence.

The two phases through which the model undergoes are:

- *Training Phase:* During the training phase, even if a mistake is bad by the decoder, the correct input is fed into the decoder at each step.
- *Testing Phase:* During the testing phase, the input for the decoder at time T+1 is defined as the output for the decoder at the given time T.

B. Data-set

The data-set used to train and validate my image captioning model is the Flickr8 data-set. The data-set consists of images and their corresponding captions. The data set consists of 8092 images which are of JPEG format. The text files consist of five distinct captions for each image which provides lucid depictions of the image. The images and captions do not consist of descriptions of famous people or well-known locations. They were selected so as to describe multiple events and entities.

C. Evaluation

The skill of my model is evaluated using BLEU Scores[1]. BLEU is an algorithm which stands for bilingual evaluation understudy and is used to evaluate the correctness and quality of text which has been translated using a machine. The correct and quality metric is the similarity between a human's output and the corresponding machine output. In other words, if the machine translation is close to a human translation, the corresponding BLEU score would be high. BLEU is one of the most popular and automated metric used in the field of deep learning.

Mathematically, BLEU can be defined as the averaged percentage of all the n-gram matches.

$$\mathbf{P} = \frac{m}{w_t} \tag{1}$$

where P is the unigram precision,

$$p = \begin{cases} 1 & \text{if } c > r \\ e^{\left(1-\frac{r}{c}\right)} & \text{if } c \leq r \end{cases} \tag{2}$$

where p is defined as brevity penalty,

$$\text{BLEU} = p.\, e^{\sum_{n=1}^{N} \left(\frac{1}{N} * \log Pn\right)} \tag{3}$$

m is the total number of words found in the output, wt is the total number of words and r is defined as the effective length of output corpus.

From the publication "Where to put the Image in an Image Caption Generator"[3], following are the approximate BLEU score for skillful and efficient models:

- BLEU-1: 0.401 to 0.578.
- BLEU-2: 0.176 to 0.390.
- BLEU-3: 0.099 to 0.260.
- BLEU-4: 0.059 to 0.170.

D. Workstation Specifications

I trained the model on a system with following configuration:

- Operating System- Ubuntu 19.10
- Processor- Ryzen 3600
- Graphic Processor- NVIDA Geforce RTX 2060 SUPER
- RAM- 16GB DDR4, 3200 MHZ

E. Workstation Environment

The system has Python 3.6 along with basic Machine Learning libraries like Pandas, Numpy, Matplotlib and sci-kit learn. Version 2.2 of Keras is used with Tensorflow 1.1 as the backend.

IV. Process

I followed the following steps in order to create and evaluate my image-captioning model.

1) Data Collection- Train/Test Split
2) Extract Data From Images
3) Extract Data From Captions
4) Generate and Train The Deep Learning Model
5) Evaluate Model
6) Generate Captions For New Images

I will be going over each of the above steps explaining them in detail.

*1) **Data Collection- Train/Test Split** :* As mentioned in III-B, the flickr8k dataset is used which consists of 8,092 images in JPEG format. It is not a very big data-set hence it is easy to train models using this data-set on a workstation with mid-tier configuration. The data-set is split in the following manner:

- *Training data-set*- It consists of 6000 images which is used to train the model.
- *Test data-set and development data-set*- They consist of 1000 images each.

*2) **Extract Data From Images** :* For extracting features from the images, I decided to go with the VGG model, VGG model is a very famous model in the field of deep learning and has been trained using ImageNet data set. The model is available in the Keras library. The model is capable to classify images with more than 98 percent accuracy. However, I have not used this model to classify images. This model is used to compute the features for each image and write them to a file and save it. The saved filed can be used later to be fed into the decoder for interpretting the image and training my model. Using this technique of pre-computing the features, we can optimize our model that will make it train much faster and utilize not a lot of memory.

The VGG model is loaded in Keras from the class VGG. I have used VGG16 model from the VGG class. I removed the final layer from VGG16 because I did not not the model to make a predicition of the class of a photo. Classifying the image is not the objective but we are interested in how to image is represented internally. As the VGG16 model requires the image to be a 244 x 244 pixels, I used keras for changing the dimensions for all the images. The output of this VGG16 model gives us a python pickle file which consists of features from all the photos. I will be saving this file for later use.

*3) **Extract Data From Captions** :* Like I extracted features from the images, I repeat the same idea for the text,i.e. the captions and save them to a file for later use. For each image in the dataset, I have 5 captions. The captions have some of the words that do not add to the meaning of the photograph so I intend to remove them. The caption is linked to it's corresponding image by a unique ID. The text file which consists of captions is traversed and with the help of a dictionary, the captions are mapped to the corresponding photo as each photo has multiple captions. Cleaning the caption text is also an important task so as to remove ambiguity raised from words of no significance. I aim to reduce the vocabulary as much as possible because that will result in a more effective model which will be faster to train. The vocabulary is cleaned in the following manner:

- All punctuation are removed
- Upper cases are converted to lowercase.
- Words with length of one character are removed.
- White spaces are removed.
- Words consisting of alpha numeric letters are removed.

The captions after going through the cleaning process are saved in a text file for later use. For instance,

- Description before cleaning-*A little girl in a pink dress going into a wooden cabin*
- Description after cleaning-*little girl in pink dress going into wooden cabin*

*4) **Generate And Train The Deep Learning Model** :* Generation of the final model is broken down into 3 steps:

1) Loading of data
2) Model definition
3) Using training set to fit the model

- **Loading Data**

The modified data obtained from images and captions is loaded into the memory. We will be using data for images in training data set only. When we will be training, we will be using the data set saved for development and keep track of the performance for our model. Depending on that performance, I decide with the models are to be saved to a file. The training dataset and the development dataset have been predefined in the Flickr8k dataset. Using the photo ID, we filter the images and their corresponding descriptions.

Here, I realize the need for the word that indicates the start of the caption as well as a word that indicate the end of the caption. I have used '*startseq*' indicating the start of the sequence and '*endseq*' indicating the end of the caption. These words are concatenated to the beginning and the end of each caption which is loaded in the memory.

A function is defined which loads all the image descriptions and returns a subset of entity for the given photo ID. I used the tokenizer class which is provided by Keras, which can learn the mapping form the caption data which is loaded in the memory. The caption is broken into multiple words. After providing one word and photo from the caption to the model, the next word is generated. Interactively, first of the two words of caption will be fed into the model with the photo in order to generate the third word. The process is followed for all words. This is the training process of the model.

The captions are then encoded one by one in the following way:

Caption: *dog splashes in the water*

The caption "dog splashes in the water" will be broken into six input and output pairs for the training of the model. For the image captioning model, which will generate captions, the words will be joined consecutively and will be provided as the input in an iterative manner in order to crate the caption for the image.

The output for this function will be the encoded word which is coming next in the sequence of caption. Hence, the output will be the probability of distribution which will be spanning over the words which are defined in our vocabulary. In order to get one word from the probability distribution, will will one-hot encode the data and all the words will have the value 0 except the definite word, which will have the value 1.

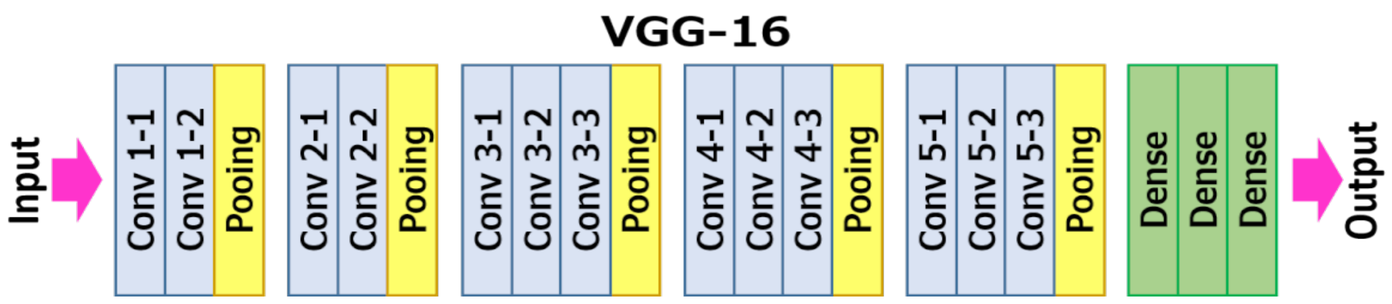
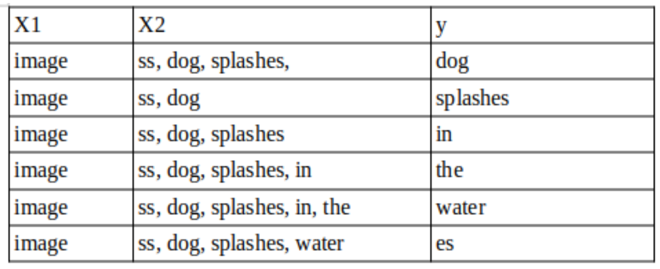


Fig. 2. Summary of VGG-16 model, used for feature extraction from images



X1	X2	y
image	ss, dog, splashes,	dog
image	ss, dog	splashes
image	ss, dog, splashes	in
image	ss, dog, splashes, in	the
image	ss, dog, splashes, in, the	water
image	ss, dog, splashes, water	es

Defining the model As discussed in section III-A, we propose a model which consists of a Sequence Processor, an Encoder and a Decoder.

Training the model I used my graphic processor for training the model, however, the memory of my graphic processor 8 gigabytes which is not enough to fit the data while the training process. I will be altering the training process due to this reason. One of the ways through which I can still use my GPU without exceeding the memory limit is by progressively loading the photos and the corresponding captions as they are required by the model. Keras has inbuilt support of loading the data-sets progressively. It is a generator type function which means that it returns chunks of the sample which the model uses to train. Multiple epochs are run, for our model, 20 of them, for the training process. One epoch means that it utilizes the whole data-set to train. As we will be training the model progressively, each epoch consists of multiple chunks or batches of training set. When each batch ends, the weight of the model is updated. Few of the other ways through which the data generator process could improve are:

- Interatively process more photos per batch
- For each epoch, introduce images in a random order

Our model is run for 20 epochs with the objective to minimize the loss on the training data-set. The model with the minimum loss value for training data-set is chosen when moving forward to generate the image-captioning model.

5) ***Evaluating The Model:*** After we have finished fitting the test data-set to our model, we evaluate the skill of the predictions made by the model on the testing data-set. We carried out evaluations of the model by creating captions for images present in the test data-set. The predictions made are then evaluated using a cost function. In order to carry this out, we need to be able to generate the description for each image in the test data-set.

As mentioned in the training section of the model, each caption starts with '*startseq*'. So we pass '*startseq*' to the model calling it recursively with the words it generate as it's input. We repeat this process until the end of the caption '*endseq*' is reached. This processes is carried to generate predcitions for captions in both the train data-set and the test-dataset. [twocolumn]article [showframe]geometrylipsumgraphicx [1-2] [3-10] A dedicated evaluate function is defined which is responsible to perform the evaluation of our model. The labelled captions and the captions that the model predicted are aggregated and both of them are evaluated using the BLEU score metric. As discussed earlier in III-C, BLEU score is dependent on the closeness of the predicted caption to the labelled caption.

For our model, as each image has 5 distinct captions, the predicted caption is evaluated against each reference caption for the image. Following this, BLEU score for 1 to 4 n-grams is calculated. The closer the BLEU score is to 1, more skillful the model is. The closer the BLEU score is to 0, more inaccurate the model is. Even during the evaluation for the model, Tokenizer function is used in order to encode the predicted words as the input for the model.

6) ***Generate Captions For New Images :*** After the evaluation phase, we head onto the final stage, which is the final model that generates captions for new images. The trained model file which we obtained after the training phase is all we need in order to predict captions for new images. However, we would be needing the tokenizer so as to encode the predicted words to be input in the model when we are generating the caption. As tokenizing process will be taking place everytime we try to predict a caption for a new image, we can reduce this work by saving the generated tokenizer as a python pickle file. Having a saved tokenizer file can be used when predicting captions for new images without the need to load the caption dataset.

V. Evaluation

For evaluation, as discussed in section III-C, we use BLEU Scores. The BLEU scores obtained by our proposed image-captioning model are:

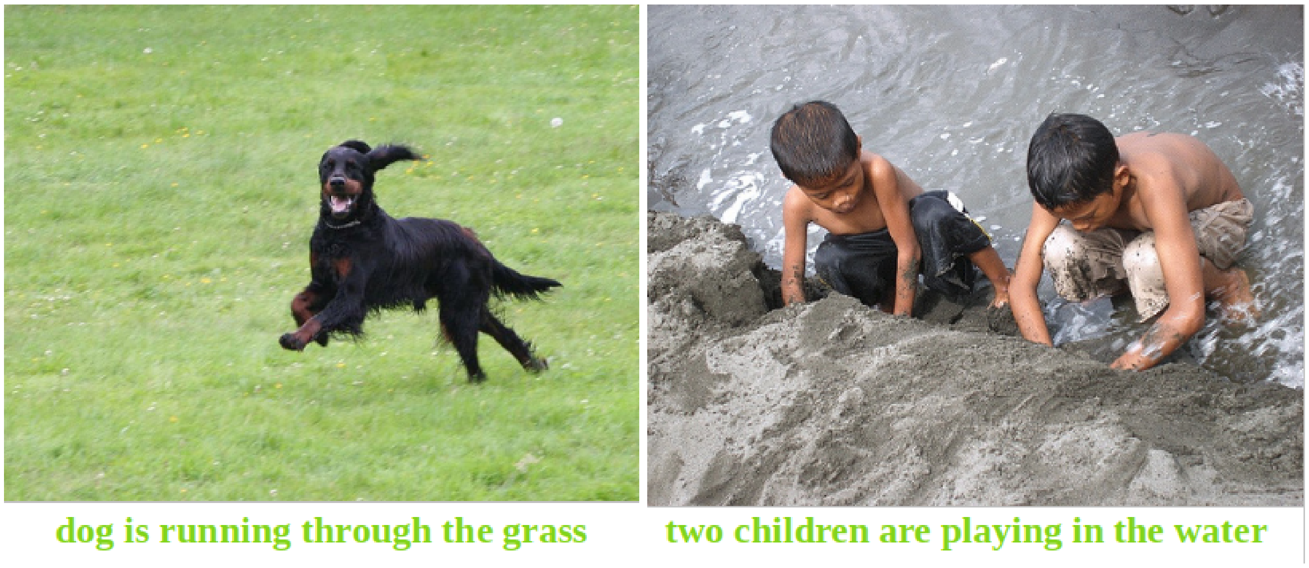
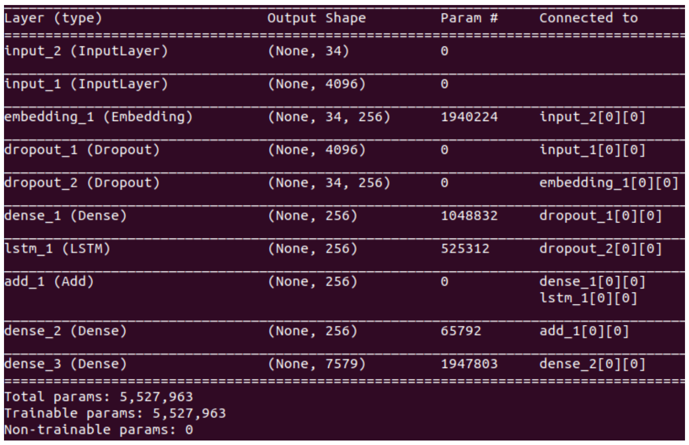
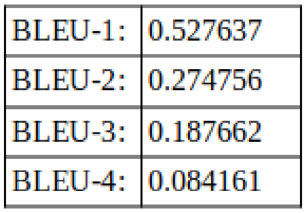

dog is running through the grass
two children are playing in the water

Fig. 3. Predicted captions for images generated by proposed model



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
input_2 (InputLayer)            (None, 34)           0
__________________________________________________________________________________
input_1 (InputLayer)            (None, 4096)         0
__________________________________________________________________________________
embedding_1 (Embedding)         (None, 34, 256)      1940224     input_2[0][0]
__________________________________________________________________________________
dropout_1 (Dropout)             (None, 4096)         0           input_1[0][0]
__________________________________________________________________________________
dropout_2 (Dropout)             (None, 34, 256)      0           embedding_1[0][0]
__________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          1048832     dropout_1[0][0]
__________________________________________________________________________________
lstm_1 (LSTM)                   (None, 256)          525312      dropout_2[0][0]
__________________________________________________________________________________
add_1 (Add)                     (None, 256)          0           dense_1[0][0]
                                                                 lstm_1[0][0]
__________________________________________________________________________________
dense_2 (Dense)                 (None, 256)          65792       add_1[0][0]
__________________________________________________________________________________
dense_3 (Dense)                 (None, 7579)         1947803     dense_2[0][0]
==================================================================================
Total params: 5,527,963
Trainable params: 5,527,963
Non-trainable params: 0
```

Fig. 4. Summary of the image-caption generator model



BLEU-1:	0.527637
BLEU-2:	0.274756
BLEU-3:	0.187662
BLEU-4:	0.084161

The obtained scores belong to the the range of skillful prediction models as discussed in the section III-C

Few of the examples are shown in IV-4(Figure 3) and IV-5(Figure 5). The predictions made for both the images in Figure 3 *"dog is running through the grass"* and *"two children are playing in the water"* are very accurate and matches the human translation. The left image in Figure 5 "man in red shirt is riding the bike on the street" is also captioned correctly except for the part that the model does not detect the color of the biker's shirt correctly(Orange and not red). The right image in Figure 3 *"two dogs are playing with red ball"* is not an accurate caption. However, model was able to detect the presence of the dog and a ball, it incorrectly states the presence of *two dogs* and *red ball*. I believe this is due to the shadow of the dog which is being cast on the beach as well as the absence of colors in the image due to poor lighting conditions.

The model delivers a decent performance for how small the data-set is and how light the weight of the model is, enabling it to be trained on any modern day workstation. I believe that the skill level of the model can be improved in the following ways:

- **Using alternate model for extract features from images:** Instead of using the VGG-16 model, other popular models that offer high performance like Inception can be used for feature extraction.
- **Tuning the model differently:** Playing around and tweaking the configurations of the proposed model can help us achieve better performance.
- **Condensing the vocabulary:** Most of the words in the vocabulary were the ones which only occurred only once in the whole data-set. Condensing the vocabulary approximate to half of it's current size would help us see a gain in performance.

VI. Related Work

Guiding Long-Short Term Memory for Image Caption Generation [5] proposes an extension to LSTM model, calling it gLSTM. Semantic information is extracted from the image as an additional input to every LSTM block. On top of this, beam search with different length normalization is implemented so that it is not biased towards shorter sentences. The semantic information here means how correlated the image and it's caption is. Based on the experiments, addition of semantic information results in significant improvement in the efficiency

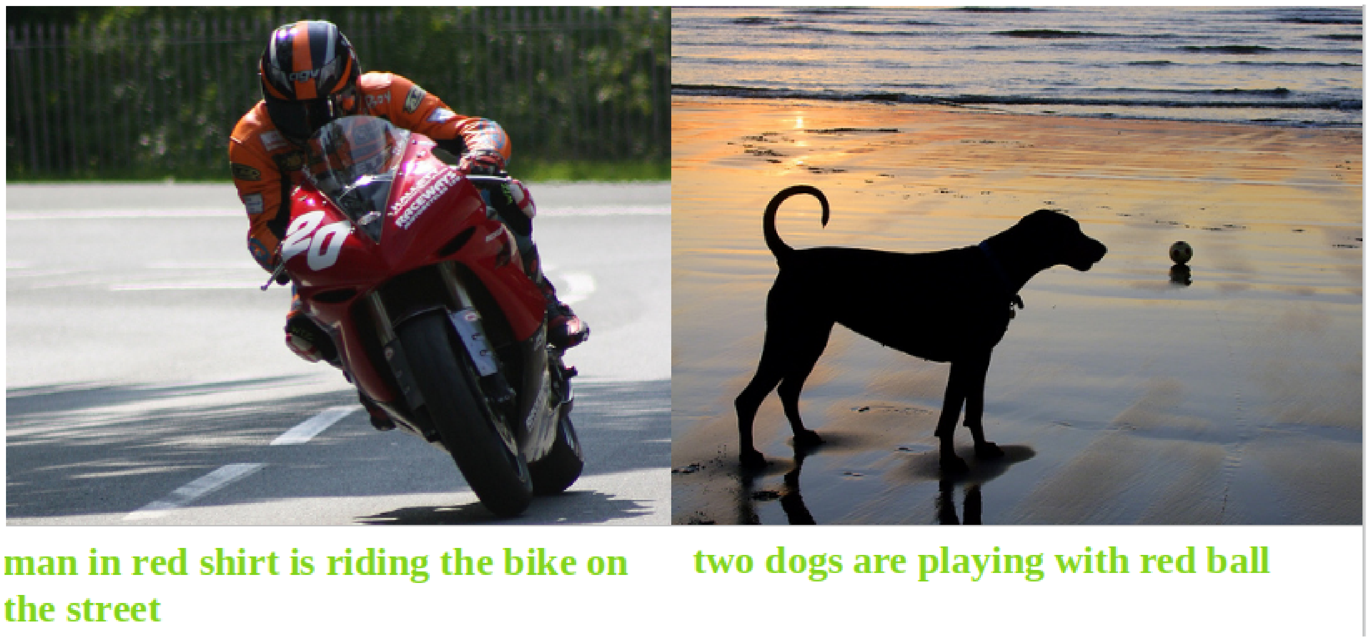

man in red shirt is riding the bike on the street
two dogs are playing with red ball

Fig. 5. Predicted captions for images generated by proposed model

and performance of the proposed model. Beam search, which is a decoder method for RNN's is also another factor of this improvement in performance. The scores BLEU-1: 0.647, BLEU-2: 0.459, BLEU-3: 0.318, BLEU-4: 0.216 for their model trained on Flickr8k dataset are very impressive and on-par with the best image captioning models out today.

Deep Captioning with Multi-modal Recurrent Neural Networks(m-nn)[4] is very similar to our proposed model and consists of a deep RNN for captions, and a deep CNN for images. The m-RNN model is powerful to work with sophisticated language data-set and complex images. Apart from the task of caption generator, the model is also able to retrieve sentences given the query and can retrieve images given the query. This exhibits the bidirectional nature of the prediction model. The obtained BLEU-1: 0.565, BLEU-2: 0.386, BLEU-3: 0.256, BLEU-4: 0.170 for their model trained of Flockr8k belong to the range score of skillful image captioning models.

VII. Summary and Conclusions

It's very normal for us, human beings to look and at a photo and come up with a caption based on the context of the image. All we need is a glance at the image and we can easily come up with a caption. However, it's not so easy for computer programs, especially before the introduction of Deep Learning. I was able to achieve carry out this task through this project. Through this project, I learned:

- How to extract data from images and text for a deep learning model.
- Designing and training a caption generation model.
- Evaluating a trained caption generator and using it to describe new images

Learning the fundamentals of image-captioning models, learning about Encoders, Sequence Processors, Decoders and mapping output of one component to another component's input, extracting relevant data from images and text, fitting the training data into the model via loading it progressively, evaluating the performance of the model and testing it to generate captions for new images was challenging but an overall rewarding experience. However, there are more skillful models out there, but the proposed model is very light and can easily be trained on most of present day's workstations. This project helped me understand neural nets in depth and also the literature research done throughout this process made me realize the potential of development in the field of Deep Learning.

References

[1] (2020). Retrieved 28 April 2020, from http://ssli.ee.washington.edu/ mhwang/pub/loan/bleu.pdf

[2] Image Captioning with Keras—"Teaching Computers to describe pictures". (2020). Retrieved 28 April 2020, from https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8

[3]TANTI, MARC, ALBERT GATT, and KENNETH P. CAMILLERI. "Where to Put the Image in an Image Caption Generator." Natural Language Engineering 24.3 (2018): 467–489. Crossref. Web..

[4] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodalrecurrent neural networks (m-rnn). InInternational

Conference on Learning Representations (ICLR)

[5] Xu Jia, Efstratios Gavves, Basura Fernando, and Tinne Tuytelaars. 2015. Guiding the long-short term memory model for image caption generation. In Proceedings of the IEEE International Conference on Computer Vision. 2407–2415.