Visual Recognition Part 2

Mini Project

(Image Captioning with CNN-LSTM System)

Team ID: 1

Team Members:

Venkat Suprabath Bitra (IMT2019091)
Vikram Madhavan (IMT2019093)
Prem Sai Reddy Bhumireddy (IMT2019067)

Assignment 3a

Introduction:

1. Convolutional Neural Network

   The convolutional neural network, or CNN for short, is a form of neural network model created for working with two-dimensional picture data, while it may also be utilised with one-dimensional and three-dimensional data. A convolutional neural network is composed of an input layer, hidden layers, and an output layer. Because the activation function and final convolution conceal the inputs and outputs of some intermediary layers in a feed-forward neural network, they are referred to be hidden.

   Convolutional layers are found in the hidden layers of a CNN. This generally entails a layer that does a dot product of the convolution kernel and the layer's input matrix. As the convolution kernel slides along the input matrix for the layer, a feature map is produced, which subsequently contributes to the input of the following layer. Following this, more layers such as pooling layers, totally connected layers, and normalising layers are added.

2. Recurrent Neural Network (RNN)

   A recurrent neural network is a feedforward neural network with internal memory. RNN is repetitive because it executes the same operation for each data input and the current input's outcome is reliant on the prior computation. After it is generated, the output is replicated and returned to the recurrent network. When determining, it considers both the current input and the outcome obtained from the prior input. Unlike feedforward neural networks, RNNs may process input sequences utilising their internal state (memory). Other neural networks' inputs are completely independent of one another. However, with an RNN, all the inputs are linked. Using their internal state, RNNs can handle variable-length input sequences (memory).

3. Long Short-Term Memory Units

   Long Short-Term Memory (LSTM) networks are a type of recurrent neural network that helps people recall information from the past. The vanishing gradient problem of the RNN is handled here. Given unknown time delays, LSTM is ideally adapted to discover, analyse, and forecast time series. The model is trained via back-propagation.

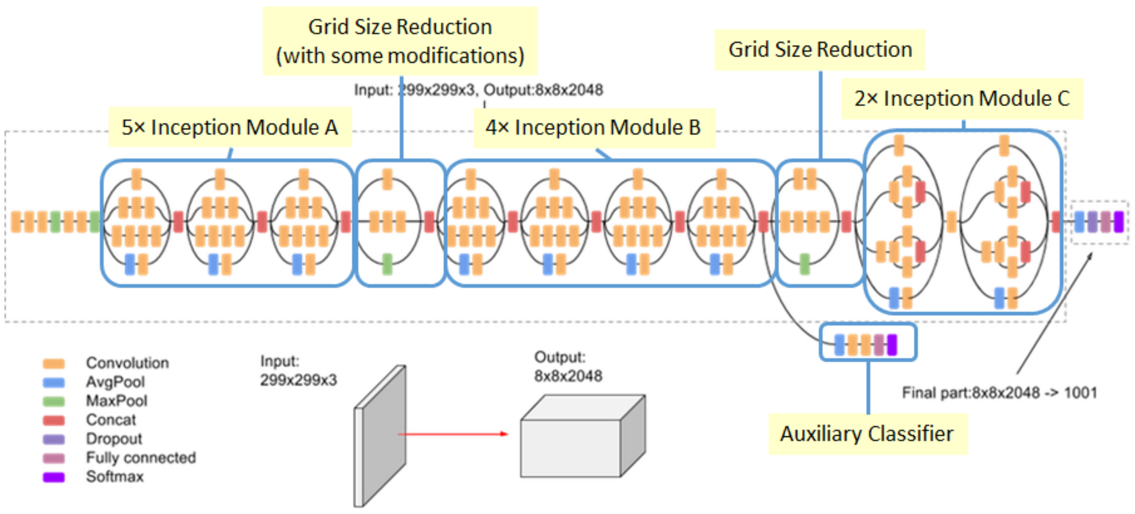
Unlike traditional feedforward neural networks, LSTM incorporates feedback linkages. It is capable of processing not just single data points (such as photos), but also complete data sequences (such as speech or video). A typical LSTM unit consists of a cell, an input gate, an output gate, and a forget gate. The three gates regulate the flow of information into and out of the cell, and the cell stores values for arbitrary time intervals. LSTMs are purpose-built to eliminate the issue of long-term reliance. It is almost their natural behaviour to recall information for extended periods of time.

Image Captioning:

Convolutional Neural Network (CNN) layers for feature extraction on input data are combined with LSTMs to support sequence prediction in the CNN LSTM architecture. On the front end, CNN layers are added, followed by LSTM layers with a Dense layer on the output to create a CNN LSTM. This architecture can be thought of as defining two sub-models: the CNN Model for feature extraction and the LSTM Model for feature interpretation over time steps. We generate a vectorized representation of an image using a deep convolutional neural network, which we then feed into a Long-Short-Term Memory (LSTM) network, which generates captions. Following advances in statistical language modelling and image recognition, image caption generation has emerged as a challenging and important research area. The ability to generate captions from images has a variety of practical applications, ranging from assisting the visually impaired to automating and cost-effectively labelling the millions of images uploaded to the Internet every day. In addition, state-of-the-art models in Natural Language Processing and Computer Vision, two major fields in Artificial Intelligence, are combined in this field. Image captioning can be done in two ways: bottom-up or top-down. Bottom-up approaches, generate items seen in an image and then attempt to combine them into a caption. Top-down approaches, aim to create a semantic representation of an image, which is then decoded into a caption using various architectures, such as recurrent neural networks. The latter approach is based on recent advances in statistical machine translation, and most state-of-the-art models use a top-down approach. The success of the top-down image generation models listed above informs our approach. We use a DCNN to create a semantic representation of an image, which we then decode using an LSTM network.

InceptionNetv3 as Encoder:

In a standard Convolutional Neural Network, we have an input image that is then passed through the network to get an output predicted label.



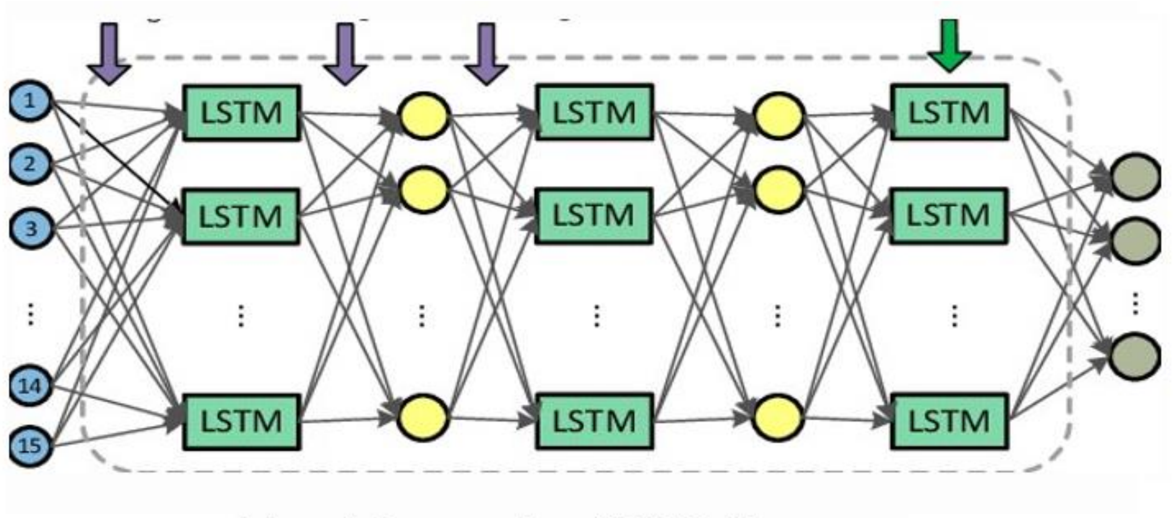
Schematic Representation of Inception Net Architecture

The Inception deep convolutional architecture was introduced as GoogLeNet, named Inception-v1. Later the Inception architecture was refined in various ways, first by the introduction of batch normalization (Inception-v2). Later by additional factorization ideas in the third iteration which is referred to as Inception-v3.

The model takes an input image of size **299 x 299 x 3** and generates an output of dimensions **8 x 8 x 2048**. There are **21768352** trainable parameters and **34432** non-trainable parameters in total.

LSTM as Decoder:

The Attention network is made up of only linear layers and a few activations. Separate linear layers transform both the encoded image, flattened to N, 14 * 14, 1664, and the hidden state (output) from the Decoder to the same dimension, namely the Attention size. A third linear layer transforms the result to a dimension of 1, after which the softmax is used to generate the weights alpha. The LSTMCell is concatenated with this filtered attention-weighted encoding and the embedding of the previous word to generate the new hidden state.

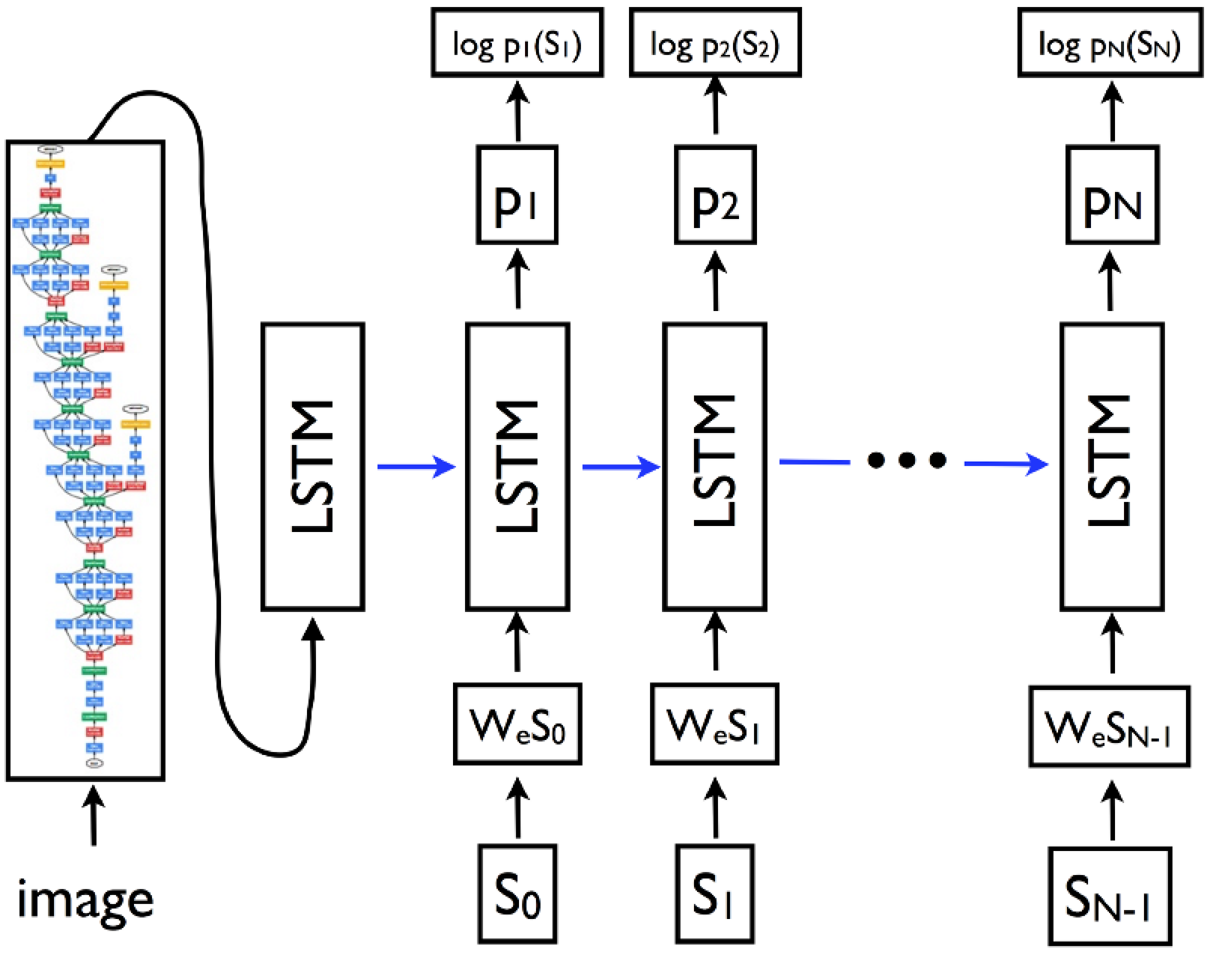


Schematic Representation of LSTM Architecture

The previous hidden state's sigmoid activated linear transform. The attention-weighted encoding is processed with a decoder filter or gate to emphasise the objects in the image. A linear layer transforms this new hidden state into scores for each word in the vocabulary, which are then saved. Instead of using the simple average, the weighted average across all pixels is used, with the weights of the important pixels being higher. At different points in the sequence, the model examines different parts of the encoded image. This weighted representation of the image can be concatenated with the previously generated word at each step to generate the next word. The output of the Decoder is transformed into a score for each word in the vocabulary using a linear layer. The best caption is found using beam search. Rather than greedily choosing the most likely next step as the sequence is constructed, it expands all possible next steps and keeps the k most likely, where k is a parameter that controls the number of beams or parallel searches through the sequence of probabilities. The model takes an input image of size 1644 x 14 x 14 and generates a caption of length L. There are **1484824** trainable parameters and **332000** non-trainable parameters in total.

Training Methodology:

The data set contains eight thousand images, each of which is accompanied by five different captions that provide detailed descriptions of the key entities and events. Because a pre-trained Encoder was used, the images were pre-processed into a format that the pre-trained Encoder is familiar with. The images were divided into mini-batches, each containing three-channel (RGB) images of the shape (3 x H x W), with H and W set to 256 pixels. Before being normalized, the images were scaled to a range of [0, 1], with mean = [0.485, 0.456, 0.406] and standard deviation = [0.229, 0.224, 0.225]. Because each word is used to create the next, captions are both the Decoder's target and inputs. The first word is generated using the zeroth word, start. The final word in a caption is referred to as end>. The Decoder is programmed to anticipate when it should stop decoding during inference. Because the captions are padded, the lengths of each caption were recorded. The length of the actual caption plus two (for the start> and end> tokens) equals this. Each sequence is only processed up to its length to avoid wasting compute on the pad tokens. The captions were fed to the model as Int tensors of dimension (N, L) where L is the padded length, and the caption lengths as Int tensors of dimension (N,) where N is the batch size. At the start, the N images and captions are sorted in order of decreasing caption lengths. This is done to ensure that only true time steps, not pad tokens, are processed.



Methodology:

Image captioning can be divided into three components. The first component includes cleaning the data which consists of captions from the training dataset. Then Start and End tokens are added to the start and end of each caption. The second component includes data processing. Encoding of captions is done by indexing every word to a number and vice-versa. Images are converted to their corresponding feature vectors. Generator function is used to train and test the model wherein the two images and their captions will be used to train and one image to test the model. The third component includes the Decoder which is used in prediction. For prediction, an image vector is provided along with the start token. The model then predicts the next word using LSTM. Then the model uses the predicted words to predict further. This process continues until the end token is predicted.

Understanding the data:

Flickr8k.token.txt which contains the name of each image along with its 5 captions. A dictionary is created which contains the name of the image (without the .jpg extension) as keys and a list of the 5 captions for the corresponding image as values.

Data Cleaning:

When dealing with text, generally some basic cleaning is performed like lower- casing all the words, removing special tokens, eliminating words which contain numbers. All of this would be done by using Natural Language Processing (NLP). There is a special library used for this purpose called nltk.

Loading the training set:

The text file Flickr_8k.trainImages.txt contains the names of the images that belong to the training set. So, these names are loaded into a list train. Now, the descriptions of these images are loaded in the Python dictionary train_descriptions. However, when they are loaded, two tokens will be added in every caption as follows:

startseq -> This is a start sequence token which will be added at the start of every caption.
endseq -> This is an end sequence token which will be added at the end of every caption.

Data Preprocessing Images:

Images are input (X) to the model. Any input to a model must be given in the form of a vector. There is a need to convert every image into a fixed sized vector which can then be fed as input to the neural network. For this purpose, transfer learning is opted by using the ResNet model (Convolutional Neural Network) created by Google Research. When only the parameters in the last few layers are trained, while the parameters of the earlier layers are only fine-tuned to better fit the purpose. Thus, this training occurs faster. This technique is also referred to as Transfer Learning. This model was trained on Image net dataset to perform image classification on 1000 different classes of images. However, the purpose here is not to classify the image but just get fixed-length informative vector for each image. This process is called automatic feature engineering. All the bottleneck train features are saved in a Python dictionary whose keys are image names and values are corresponding 50 length feature vectors. Similarly, all the test images are then encoded.

Data Preprocessing Captions:

The prediction of the entire caption, given the image does not happen at once. The prediction of the caption word by word. Thus, there is the need to encode each word into a fixed sized vector. Before this, every unique word in the vocabulary will be represented by an integer (index). After this, the maximum length of a caption will be calculated from all the lengths.

Final Data Preparation for Training and Predicting:

Data will be given as input to the deep learning model that will be used to train and predict the captions corresponding to the pictures or images provided. First, the images would be needed to be converted to their corresponding 50 length feature vector as mentioned above. Secondly, the vocabulary for the train captions would be built by adding the two tokens startseq and endseq in both. Assuming the basic cleaning steps using Natural Language Processing via their libraries has already been performed. Then an index would be given to each word in the vocabulary. After that, there is the need to frame it as a supervised learning problem where a set of data points D = {Xi, Yi} is available, where Xi is the feature vector of data point i and Yi is the corresponding target variable. Image vector is the input, and the caption is what needs to be predicted. But the way the caption is predicted is by predicting the following words based on the current word and the sequence. For the first time, the image vector is provided and the first word as input and tries to predict the second word. One image caption is not a single data point but multiple data points depending on the length of the caption. So, it's typically the partial caption that is going to be generated step by step. In every data point, it’s not just the image which goes as input to the system, but also, a partial caption which helps to predict the next word in the sequence. Since sequences are being processed, a Recurrent Neural Network will be employed to read these partial captions. The actual English text of the caption is not going to be passed, rather the sequence of indices where each index represents a unique word is going to be passed. As an index for each word has been created, the words will be replaced with their indices and how the data matrix will look like will be understood. Batch processing would be done in this process, hence there is a need to make sure that each sequence is of equal length. Hence 0s (zero padding) would be needed to be appended at the end of each sequence. For this, the maximum length of a caption is calculated. So those many numbers of zeros will be appended which will lead to every sequence having the maximum length. Every word (or index) will be mapped (embedded) to higher dimensional space through one of the word embedding techniques. During the model building stage, each word/index is mapped to a 50-long vector using a pre-trained GLOVE word embedding model. In this process. There is a huge requirement to manage to load this many data into the RAM, it will make the system very slow. For this reason, data generators are used a lot in Deep Learning. Data Generators are a functionality which is natively implemented in Python. The ImageDataGenerator class provided by the Keras API is an implementation of the generator function in Python. With the help of this, there is no requirement to store the entire dataset in the memory at once. Even if the current batch of points is available in the memory, it is sufficient for the purpose. A generator function in Python is used exactly for this purpose. It’s like an iterator which resumes the functionality from the point it left the last time it was called. After this, the input and the trained vectors will be used to predict the captions of the corresponding images.

Hyper parameters during training:

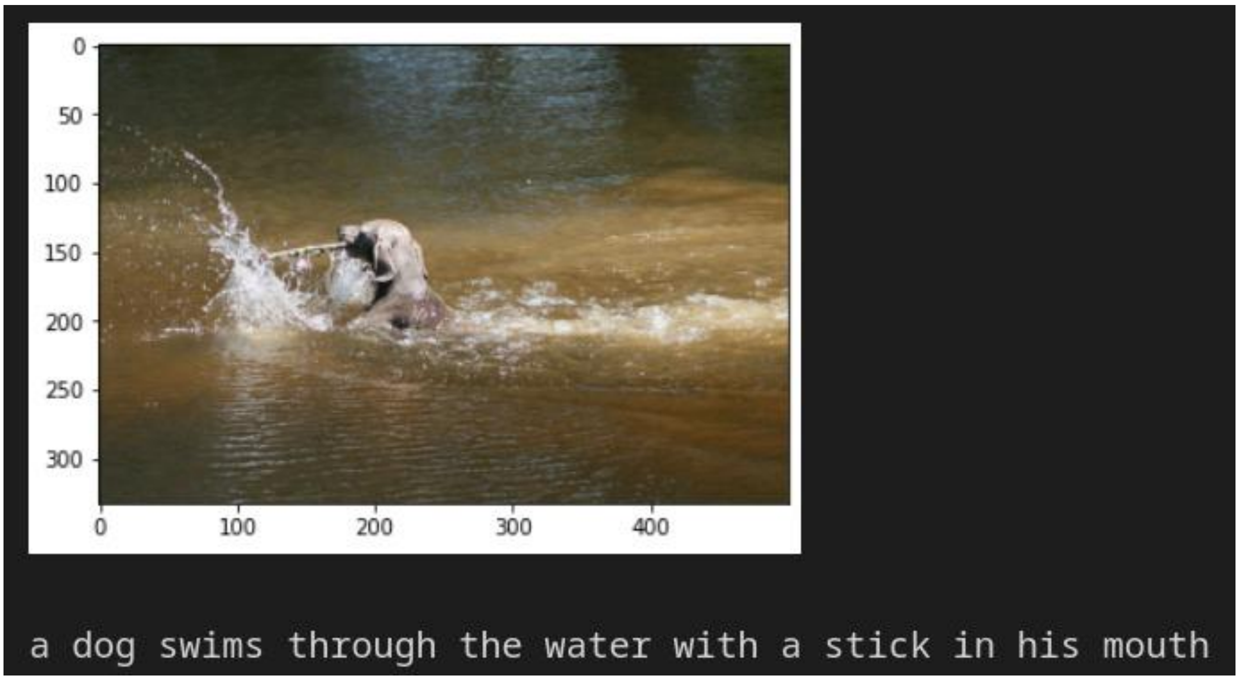
The model was then trained for 20 epochs with the initial learning rate of 0.001 and 3 pictures per batch (batch size). However, after 20 epochs, the learning rate was reduced to 0.0001 and the model was trained on 6 pictures per batch. This generally makes sense because during the later stages of training, since the model is moving towards convergence, we must lower the learning rate so that we take smaller steps towards the minima. Also increasing the batch size over time helps your gradient updates to be more powerful.

Results:

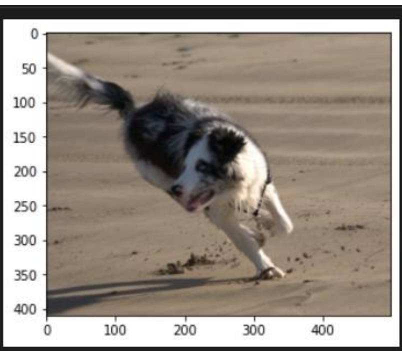
The BLEU scores obtained are:

- BLEU-1: 0.5317
- BLEU-2: 0.3576
- BLEU-3: 0.2348
- BLEU-4: 0.1483

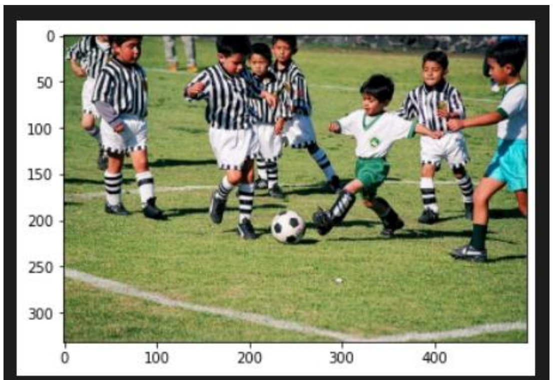
Some examples good image captioning is given below:



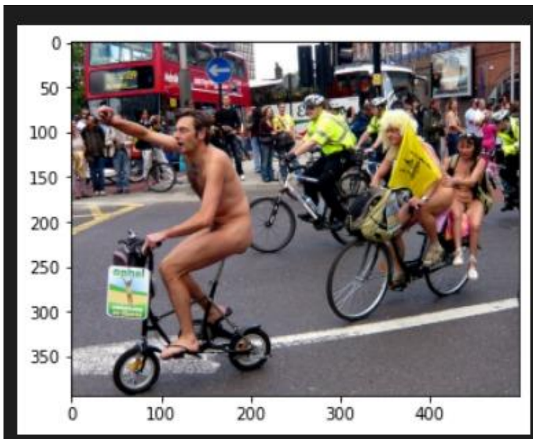
```
a dog swims through the water with a stick in his mouth
```



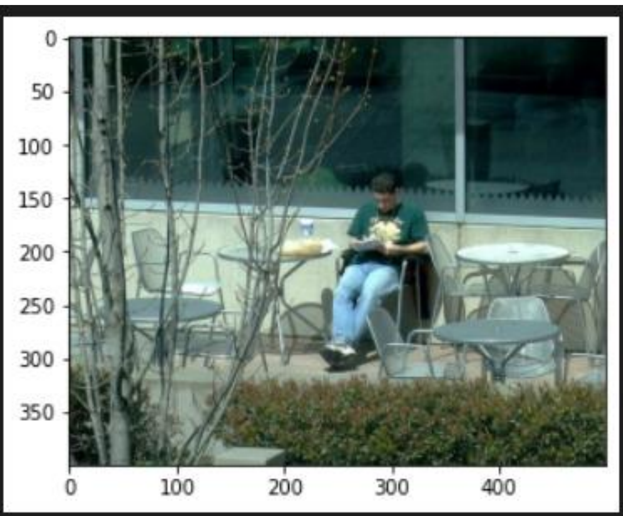
```
a brown and white dog is running on the beach
```



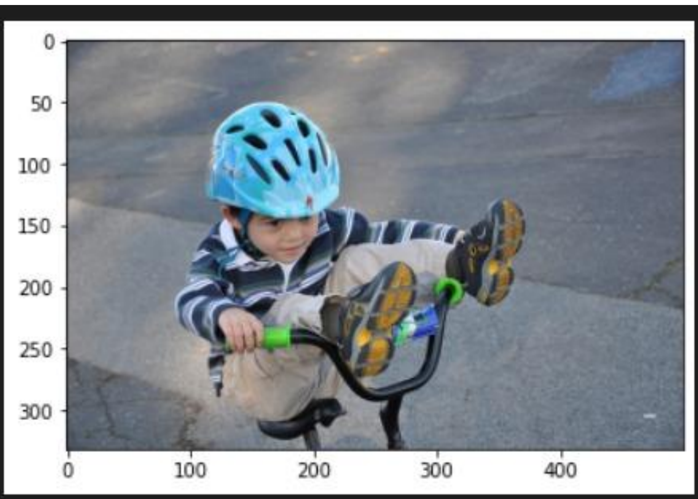
```
boys playing soccer on a field
```



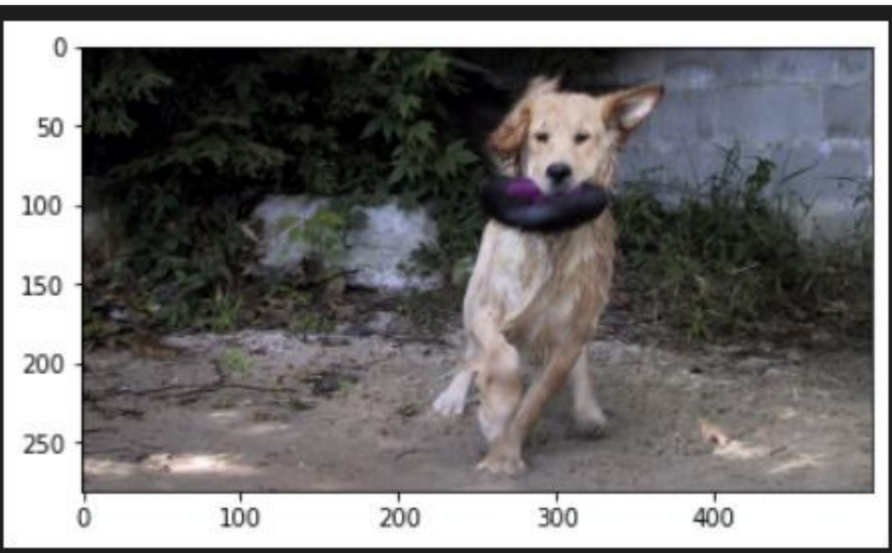
```
a group of people ride bicycles on a busy street
```



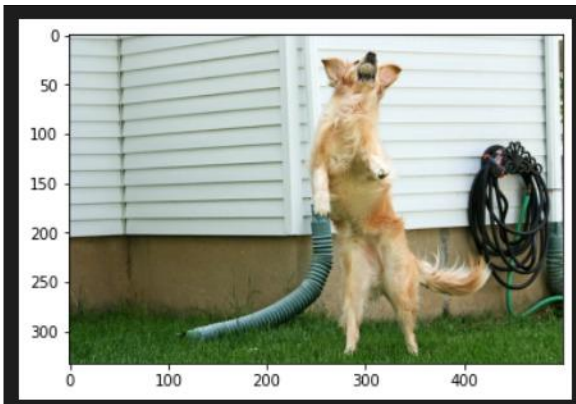
```
there is a man sitting on a park bench reading a newspaper
```



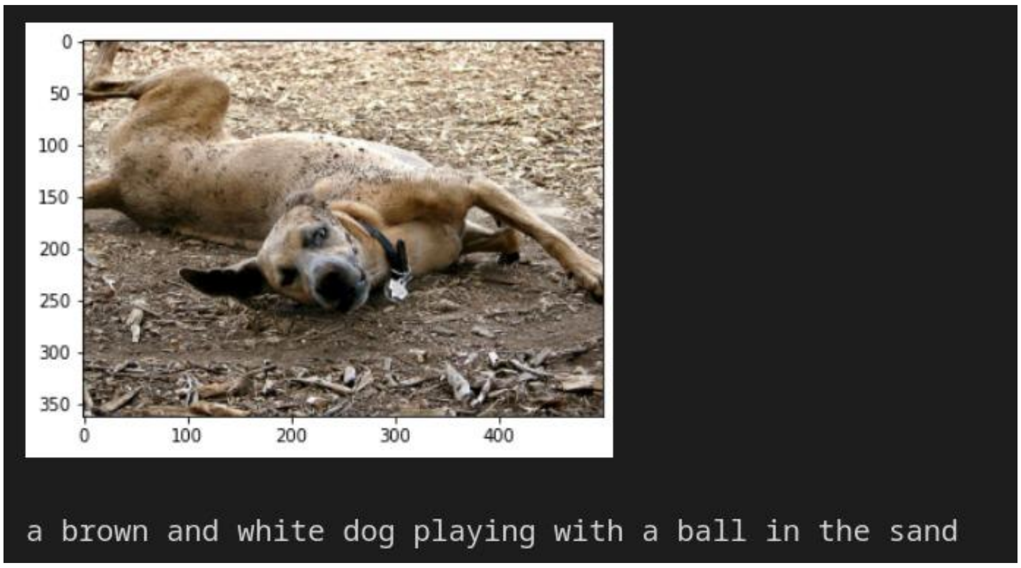
```
a little boy is riding a bicycle in a parking lot
```



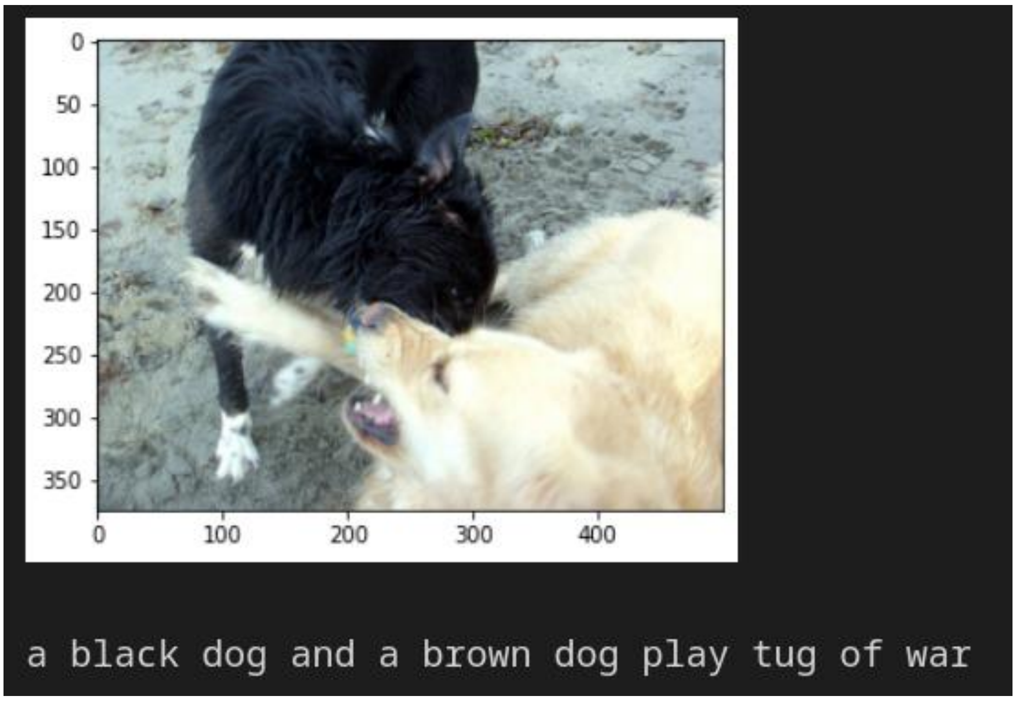
```
a brown and white dog in the grass
```



```
a brown and white dog jumps up to catch a tennis ball in its mouth
```



```
a brown and white dog playing with a ball in the sand
```



```
a black dog and a brown dog play tug of war
```

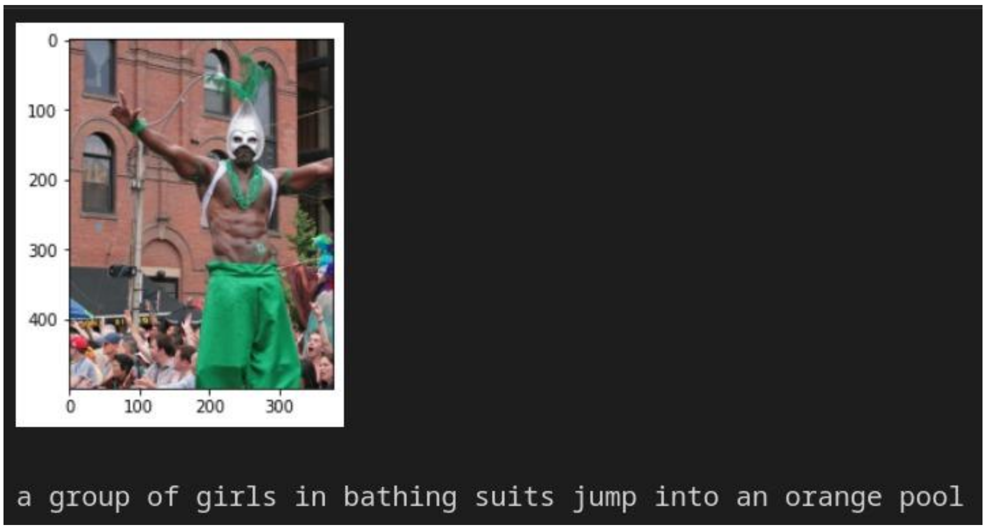
Assignment 2b

The methods have three main issues:

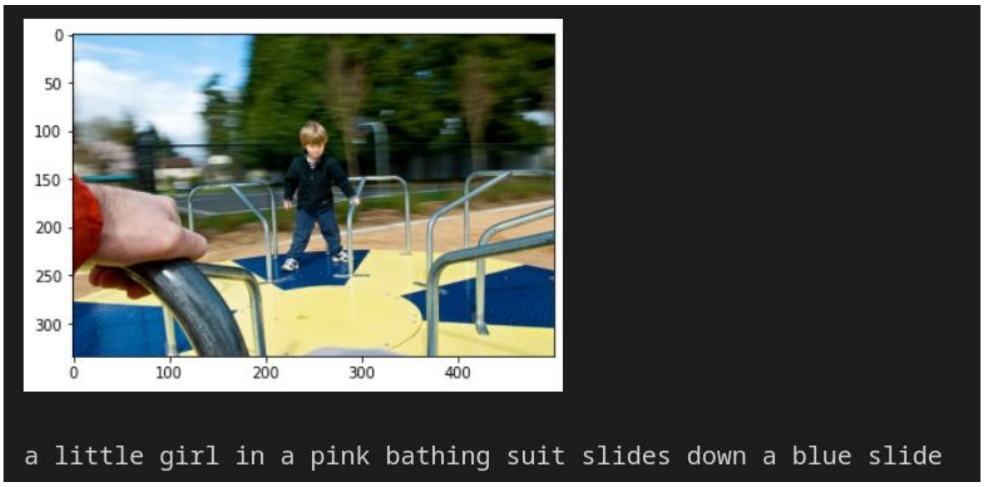
- They are trained using maximum likelihood estimation and back-propagation approaches. In this case, the next word is predicted given the image and all the previously generated ground truth words. Therefore, the generated captions look-like ground-truth captions. This phenomenon is called the exposure bias problem.

- Evaluation metrics at test time are non-differentiable. Ideally sequence models for image captioning should be trained to avoid exposure bias and directly optimize metrics for the test time.
- A CNN and RNN based combined network generate captions. We have also included LSTM to enhance the prediction accuracy of our model. LSTMs provide us with a large range of parameters such as learning rates, and input and output biases. Hence, no need for fine adjustments. The complexity to update each weight is reduced to O(1) with LSTMs, similar to that of Back Propagation Through Time (BPTT), which is an advantage of our proposed method.

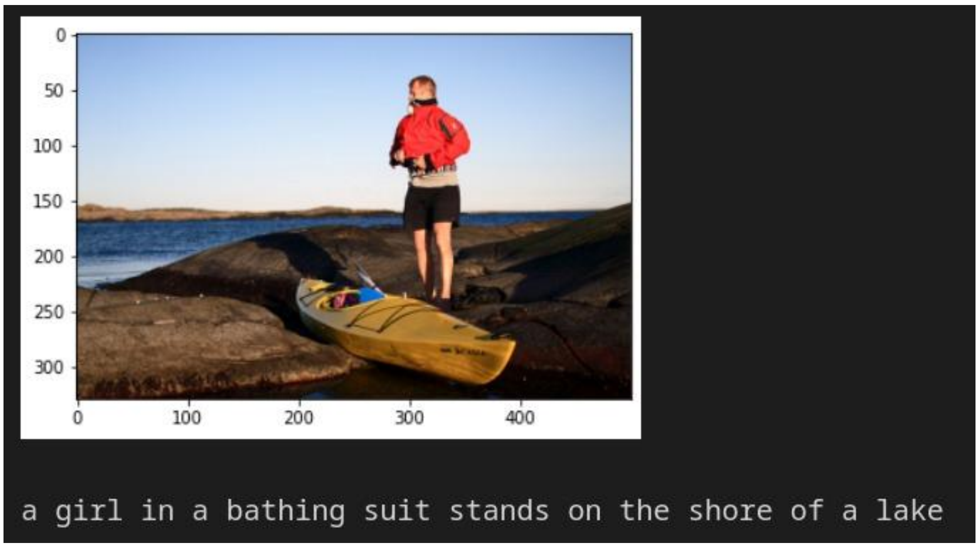
These images were not captioned well due to language bias:



```
a group of girls in bathing suits jump into an orange pool
```

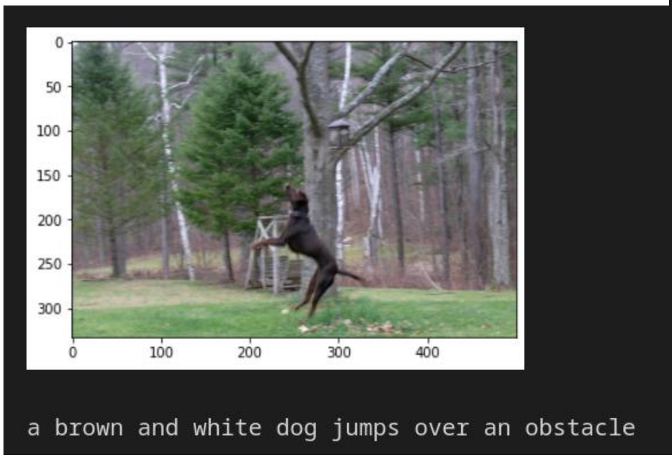


```
a little girl in a pink bathing suit slides down a blue slide
```

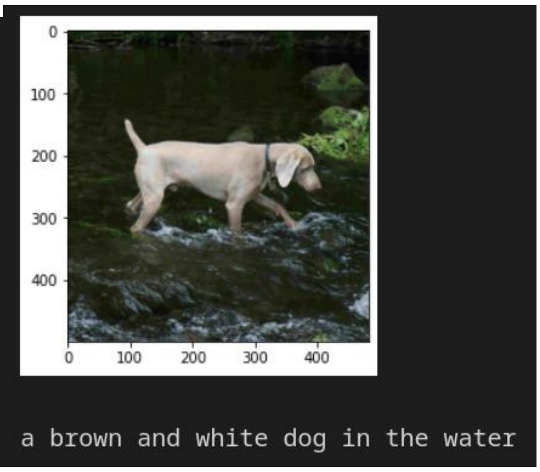


```
a girl in a bathing suit stands on the shore of a lake
```

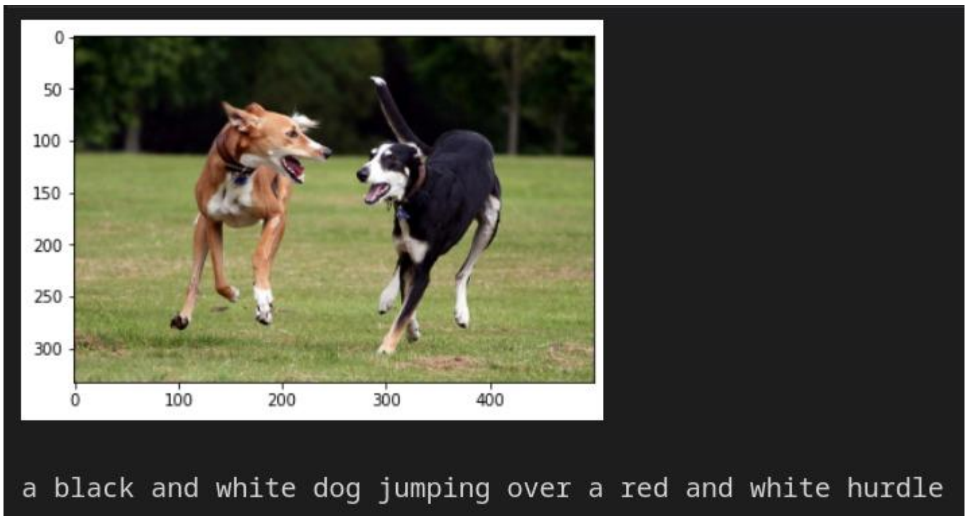
In the above three images, we observe that even though there are no girls it thinks that there are girls in bathing suits. Here, we can clearly observe language bias as the model associates girls with bathing suits.



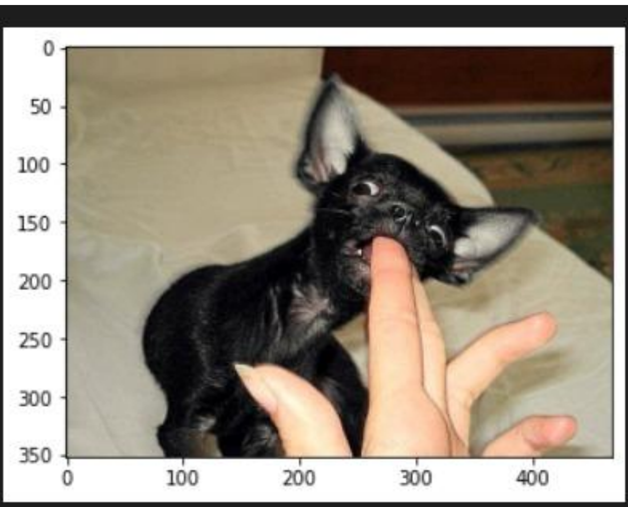
```
a brown and white dog jumps over an obstacle
```



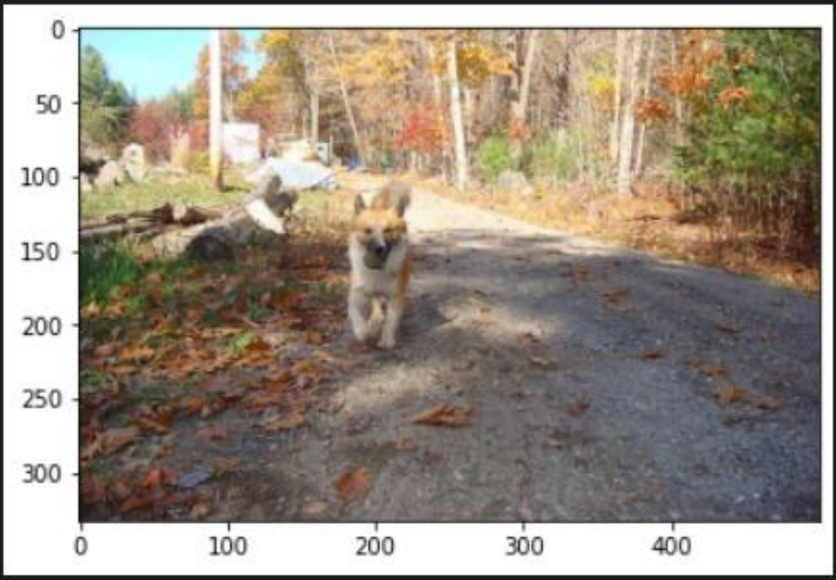
```
a brown and white dog in the water
```



```
a black and white dog jumping over a red and white hurdle
```

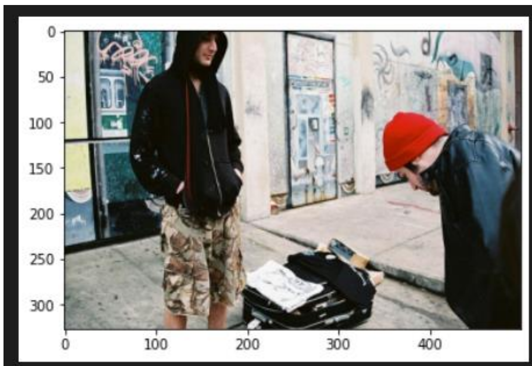


```
a black dog and a brown dog bending on a bed
```

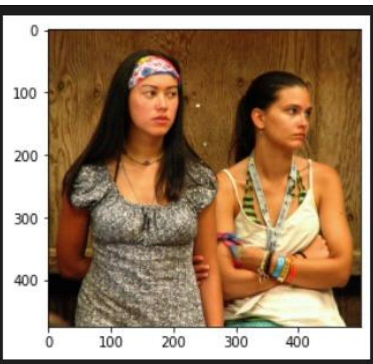


```
a brown and white dog jumps over a pile of fire
```

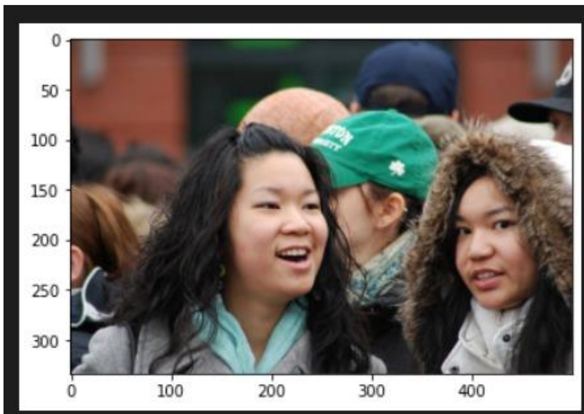
We observe language bias in the above 4 images as we see that the images always associate one dog or two dogs with black/brown and white color dogs.



```
a man in a white shirt and a woman in a white shirt walk down the street
```



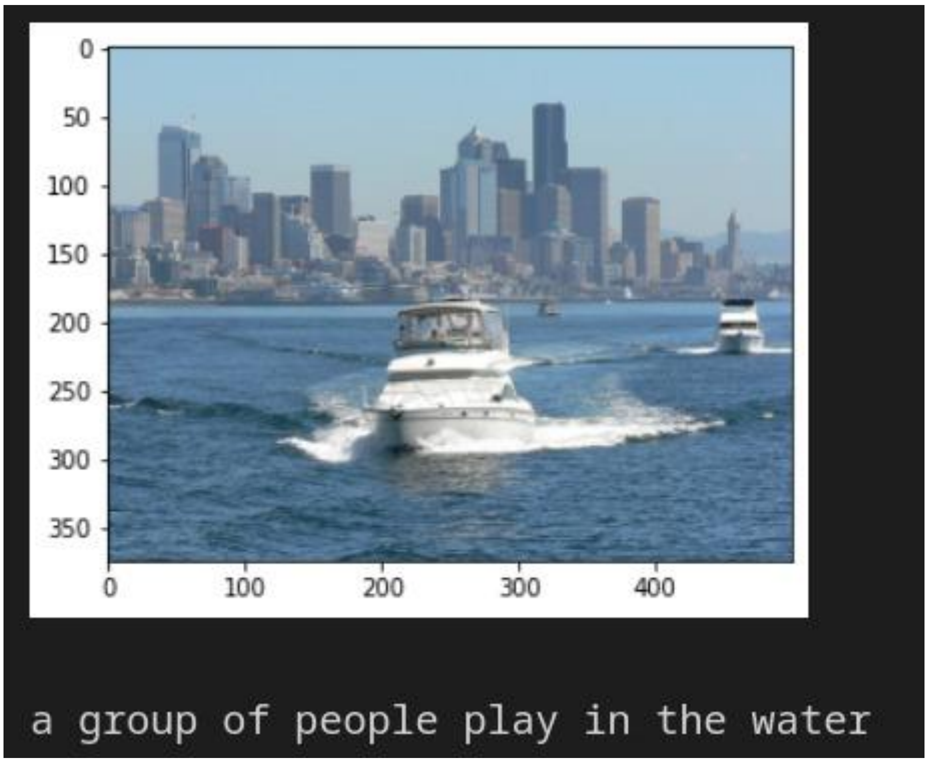
```
a man in a white shirt and a woman in a white shirt smiling for the camera
```



```
a woman in a white shirt and a man in a white shirt smiling for the camera
```

The above set of images have language bias in the form whenever it thinks there is a man and a woman it thinks they are both wearing white shirts.

Below are couple of images which are totally wrong.



```
a group of people play in the water
```

The results above have been obtained by sampling the images and sorting them in increasing order of bleu_1 score which gave images in increasing order of improved captions. Even among the good captions we see that the language bias persists.

The improvements or observations are as follows:

- with the fast development of deep neural networks, employing more powerful network structures as language models and/or visual models will undoubtedly improve the performance of image description generation.
- images consist of objects distributed in space, while image captions are sequences of words, investigation on presence and order of visual concepts in image captions are important for image captioning. Furthermore, since this problem fits well with the attention mechanism and attention mechanism is suggested to run the range of AI related tasks, how to utilize attention mechanism to generate image captions effectively will continue to be an important research topic.
- due to the lack of paired image-sentence training set, research on utilizing unsupervised data, either from images alone or text alone, to improve image captioning will be promising.

- current approaches mainly focus on generating captions that are general about image contents. However, to describe images at a human level and to be applicable in real-life environments, image description should be well grounded by the elements of the images. Therefore, image captioning grounded by image regions will be one of the future research directions.
- most of the previous methods are designed to image captioning for generic cases, while task- specific image captioning is needed in certain cases. Research on solving image captioning problems in various special cases will also be interesting. As a future scope, using a larger dataset would be a better option. Changing the model architecture, e.g., include an attention module could be a better solution for image captioning process. Doing more hyper parameter tuning (learning rate, batch size, number of layers, number of units, dropout rate, batch normalization etc.) and using the cross validation set to understand overfitting can be beneficial. By using Beam Search with greater ‘k’ instead of Greedy Search during Inference and by using BLEU Score to evaluate and measure the performance of the model could give more accurate results.

References:

1. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8718290
2. https://www.cse.iitk.ac.in/users/cs671/2015/_submissions/vinsam/project/proposal.pdf
3. https://www.mecs-press.org/ijigsp/ijigsp-v11-n6/IJIGSP-V11-N6-1.pdf
4. https://openaccess.thecvf.com/content_ICCV_2017/papers/Gu_An_Empirical_Study_ICCV_2017_paper.pdf
5. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00571-w
6. https://mdpi-res.com/d_attachment/applsci/applsci-10-05841/article_deploy/applsci-10-05841.pdf?version=1598248441
7. https://link.springer.com/article/10.1007/s12559-018-9581-x
8. https://arxiv.org/pdf/1810.04020.pdf
9. https://www.irjet.net/archives/V8/i8/IRJET-V8I824.pdf
10. http://cse.anits.edu.in/projects/projects1920B11.pdf