WebEye: Image Captioning tool for The Visually Impaired

Mohamed Nassar
dept. comm. engineering
Cairo University
Cairo, Egypt
MohamedNassar2016@gmail.com

Noha Ahmed
dept. comm. engineering
Cairo University
Cairo, Egypt
NohaAhmad1.na@gmail.com

Ramez Shendy
dept. comm. engineering
Cairo University
Cairo, Egypt
ramezashendy@gmail.com

Omar Nasr
dept. comm. engineering
Cairo University
Cairo, Egypt
omaranasr@cu.edu.eg

***Abstract*—Image captioning is a way of generating a textual body out of the content of an observed photo. Recently the interest in generating robust image captions has increased due to the rapid development of artificial intelligence using deep learning techniques of both computer vision and natural language processing combined. In this paper, a state of art encoder-decoder with visual attention architecture was adopted to develop a fully automatic image captioning system for the visually impaired to be used via a simple browser extension. The integrated solution along with a simple screen reader can help the visually impaired people to not only read textual bodies but also make use of images through their descriptive and detailed captions.**

***Index Terms*—computer vision, deep learning, image captioning, natural language processing**

I. Introduction

The development of the computer vision as well as the natural language processing (NLP) applications in the recent years using deep learning architectures and techniques has done great breakthroughs overtime and made many real life applications achievable with incredible results that sometimes exceed the human performance in some tasks. Naming a few of these applications:

- Image Classification: It's the task of extracting information classes from various band of images.
- Object Detection: The process of detecting different semantic instances within the same image.
- Object Localization: The prediction of an object in an image as well as its coordinates on the image and its boundaries.
- Machine Translation: Refers to the automated software that can do translation of a natural language to another with proper syntactic and semantic meaning.
- Speech Recognition: Software developed methodologies that enables the recognition of a natural language and convert it into text.
- Language Modeling: It's constructing a statistical model over a sequence of words with their probability distribution. One of the main examples of a language model is the prediction problem of next words that come in sequence semantically correct.

Image captioning is an image description generation task. The expected output of an image captioning system is a detailed single sentence describing what is happening in the image scene. The high quality generated text should be both syntactically and semantically correct [4]. The advancements in deep learning especially in computer vision and NLP applications led to the development of a very reliable and coherent image captioning systems capable of handling the complexities and the challenges that hindered such systems before. Example of an automatic image captioning system can be shown in fig.1.

Fig. 1. Automatic image captioning example [3].

Image captioning can hugely help the visually impaired to better understand the content of websites across the internet instead of relying on screen readers which can only provide machine translation and text-to-speech output. These screen readers can also make use of image captions generated by the state of art algorithms combining the power of the latest advancements in computer vision for constructing bottle neck features that can be used by the encode-decoder with visual attention deep neural network architecture to generate these image descriptions using NLP techniques.

In this paper a fully intelligent and automatic software tool powered by an image captioning engine is introduced to help the visually impaired utilize the content of images as well as textual bodies on the internet. The paper is organized as follows. In section II The full solution pipeline is illustrated.

The full deep learning architecture is described in details in section III. More on the software design and solution is depicted in section IV. Image captioning experiments held is exhibited in section V. Finally future work is stated and the paper is concluded in section VI.

II. Solution Pipeline

In this section, the full solution pipeline is discussed. The main idea behind WebEye is to intercept the HTTP response of the requested web-page, this is done via a browser extension. Once having the retrieved document object model (DOM) coming from the HTTP response, the next step is altering the missing alt attributes in the HTML image tag. Populating these missing alt attributes is done by sending the image URLs in the image tag to the local server that handles these URLs, by downloading these images, pre-processing them and feeding them as input to the deep learning encoder-decoder with attention image captioning system which takes these images and outputs a corresponding sentence describing this image. This description is inserted into the alt tag in the image HTTP response. Eventually the end user can utilize the normal screen reader that now not only can read text in any web page but also read captions describing the images. The solution pipeline is shown in fig.2.

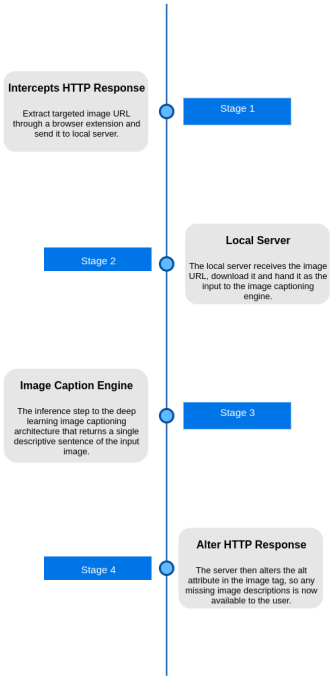

Fig. 2. Full solution pipeline.

III. Deep Learning Model

In this section, we will discuss the core of our product which is the captioning process. Starting from passing an image as input to the network all the way to the predicted output (caption) generation. The implementation we have used is inspired by the Show, Attend and Tell paper [2], but we have used pyTorch implementation unlike the authors of the paper who used theano framework.This section will be organized as follows: First we discuss the Encoder-Decoder architecture used, followed by the attention mechanism then we move to the output selection criteria using the beam search algorithm, and finally we discuss the metrics used for the performance evaluation..

A. Encoder-Decoder Architecture

The encoder is used to encode the input images to a feature vector that has a smaller size than the original input image but contains the sufficient information needed for understanding the image. In our implementation we have used the pretrained ResNet101 Convolutional Neural Network ,trained on the ImageNet classification task, as the encoder. Fig.3. Shows the input and output of the encoder.

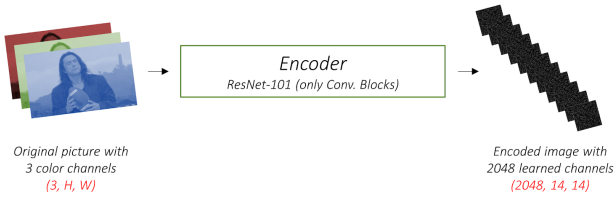

Fig. 3. Encoder input output [6].

The decoder is used to decode the encoded image to a sequence of words (caption). Since this is a sequence generation problem thus we will use a Recurrent Neural Network (RNN), specifically LSTM (Long Short Term Memory). The input to the decoder is the encoded image and the previous word output. Fig.4. Shows the input and output of the decoder without the attention mechanism.

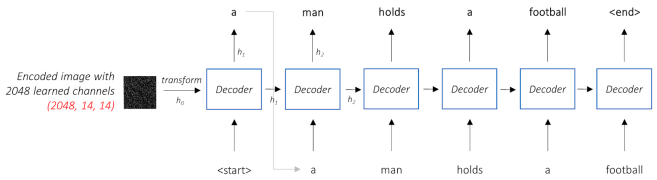

Fig. 4. Decoder architecture [6].

B. Attention Mechanism

In our challenging image captioning problem, an attention mechanism is surely needed. An image may contain many objects thus suffers from strong cluttering and in that case we don't want our model to diverge and get distracted by these irrelevant objects. For that reason we need our model to attend to the parts of the image that are relevant to the next decoded output (word) and give large weight to this part of the image while smaller weights to the other parts of the images. Fig.5. shows a visualization for the attention network usage in our captioning problem.

This can be interpreted as computing the probability that a certain pixel is the pixel of interest when generating the next

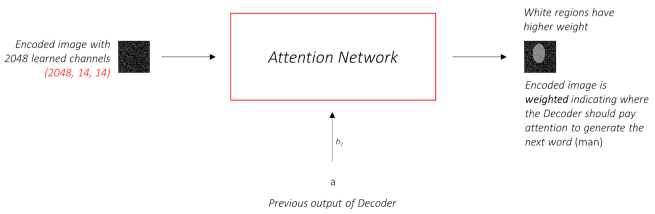

Fig. 5. Attention mechanism [6].

word. Fig.6. Shows the overall architecture with the attention mechanism.

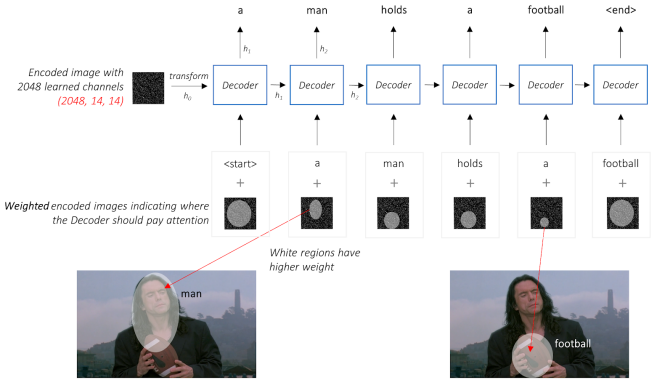

Fig. 6. Overall architecture with the attention mechanism [6].

C. Beam Search Algorithm

The beam search algorithm allows a wider and more accurate results by choosing the best K words at each time step then generating a sequence of output for each chosen word thus maximizing the score for the overall sequence and not a single word. Choosing the beam search algorithm as output selection criteria introduces a great enhancement to the generated sequence (caption) in our problem. Putting in mind that the first word in a sequence is essential and can lead to either a great caption and exact description or misleading caption and irrelevant description, we have decided not to use the greedy search which only considers maximizing the current word and not the entire sequence instead use the beam search algorithm. Fig.7 is an example for a beam search output with a beam width of 3.

D. Performance Evaluation

To be precise, this problem does not have an exact criteria or metric to measure how well the network works. But we have chosen the BLEU (Bilingual Evaluation Understudy) score as an approximate measure of the performance of the network. BLEU score compares a generated sequence to a reference sequence and in case of a perfect match the score is 1 while in case of perfect mismatch the score is 0.

IV. Training Phase

In this section we will discuss the input to the model (datasets) and the pipeline of the training process.

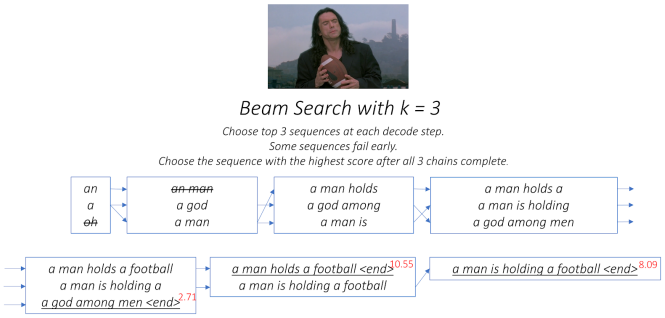

Fig. 7. Beam search output [6].

A. Dataset

Initially the model was trained on the MSCOCO [**?**] dataset which contains around 82,000 images but due to the limitation in our resources, we did not train on COCO dataset. We used the FLICKR30K [**?**] dataset, which is a very popular dataset used in the image captioning problem.

B. Data Generation

First we run `create_input_files.py` to generate these files:

- An HDF5 file containing images for each split in an I, 3, 256, 256 tensor, where I is the number of images in the split. Pixel values are still in the range [0, 255], and are stored as unsigned 8-bit Ints.
- A JSON file for each split with a list of `N_c * I` encoded captions, where `N_c` is the number of captions sampled per image. These captions are in the same order as the images in the HDF5 file. Therefore, the ith caption will correspond to the `N_cth` image.
- A JSON file for each split with a list of `N_c * I` caption lengths. The ith value is the length of the ith caption, which corresponds to the `N_cth` image.
- A JSON file which contains the word map, the word to index dictionary.

HDF5 files are used for the images to be read directly from the disk while training and validation.These files are too large to fit into the RAM all at once. On the other hand all captions and their lengths are loaded into the memory.

C. Loss Function

Cross Entropy loss is used. The raw scores for the predicted from the final layer of the decoder are compared to the ground truth scores. Doubly Stochastic Regularization loss is also used. This loss function encourages the weights of a single pixel p to sum to 1 across all time steps T as shown below.

$$\sum_{t}^{T} \alpha_{p,t} \approx 1 \tag{1}$$

This loss function forces the model to attend to every pixel when generating the entire sequence. Thus it is trying to minimize the difference between 1 and the sum of pixel's weights across all timesteps.

D. Training setup

We trained the decoder from scratch for 10 epochs at a learning rate of 4e-4, using Adam optimizer, with a batch size of 32 and dropout of 0.5 without fine tuning the encoder. At this point the BLEU score rose up to 20%. Then we fine tuned the encoder and resumed the training from the 7th epoch (Best weights) with a learning rate of 1e-4 at and at this point the BLEU score went as far as 23%.

V. Experiments

In this section we will discuss different approaches and trials that we used to try to achieve higher BLEU score and better captioning output.

A. Optimizers Variation

Flickr8k dataset was used and we followed this training procedure. First we trained only the decoder with a batch size of 64 for 10 epochs, with a learning rate of 4e-4, using without fine tuning the encoder. We then started from the 7th epoch (Best weights) and fine tuned the encoder (convolutional blocks from 2 to 4) with an encoder learning rate of 1e-4 and the decoder learning rate is still 4e-4 and with a batch size of 32. Table.1. Shows the BLEU score for the above training procedure using different optimizers.

TABLE I
Optimizer and corresponding BLEU score.

Optimizer	BLEU score
ADAM	23.21
RMSprop	22.9

Although the results seem quite close in terms of the BLEU score yet the differences are remarkable and the ADAM optimizer got actually much better results. See Appendix for results of this experiment.

B. Captions length variation

When generating the input data files, there is a parameter that specifies the maximum length for a caption to be considered. Initially this threshold was set to 50, but at that point the BLEU score and the results when testing on unseen images were not satisfactory. So we increased the threshold to 100 to allow for more data to be taken into consideration since the FLICKR8K dataset contains only 8000 images unlike the COCO dataset which was initially used and contained 82,000 images. This change in threshold actually improved the accuracy and the BLEU score in a remarkable way, the BLEU score changed from 19 to 23! And also the results when testing on several images were much better. See Appendix for results of this experiment.

C. Datasets Variation

In this experiment, we have trained FLICKR30K for 10 epochs and then fine tuned the encoder for another 5 epochs. We have also tested on the COCO weights that were uploaded in our reference implementation. See Appendix for results of this experiment.

VI. Software role in webEye

In this section we will discuss the software role in our product.

A. Browser Extension

To read all the data (images) from a certain webpage we need a browser extension. The browser extension does the following:

- Intercepts the HTTP response of the webpage.
- Extracts the images URLs (source) and attributes (width,height) and sends it to the local server.
- Finally, updates the HTTP response (the Alt tag and Title tag) with the caption obtained from the server.
- This part is developed using javascript.

B. Local Server

The local server is the link between the browser extension and our AI engine. It is responsible for the following:

- A list of the image URLs are sent to the local server.
- It then downloads the images and passes the image to the image captioning algorithm.
- It then sends a list of text sequences (captions) corresponding to each image.
- This part is developed using python.

C. Enhancements to the software

The first enhancement was crucial as after processing the images in the above manner, we have noticed a great overhead and very slow response due to the fact that we load all the images found in the webpage. Thus we have decided to caption the images on a certain event precisely a mouse hover event, in other words when the user places the mouse on a certain image, only then the image URL is sent to the local server and the caption is retrieved thus saving much time and memory.

The second enhancement is that if the image was already captioned before and the user closes the webpage then reopen it again, the image caption is still saved in a URL list in the extension and thus the image does not proceed to the local server. This also removes the redundancy and makes our code simpler and more optimized.

The third enhancement or feature of our webEye product is that it offers a translation service for the caption to any other language using google API. such a feature is quite important for our product so that it meets the needs of the different users.

VII. Conclusion and Future work

A. Conclusion

In this paper, we introduced a new software tool for the visually impaired to help them better understand the content on the internet. There were two main contributions of our work. The first one is to tackle the problem of the image captioning and reach a point of generating syntactically and semantically correct image descriptions through different experiments and setups. A discussion on the used architectures of the encoder and the decoder and also the attention mechanism has been provided. As well as a full glimpse on the full solution pipeline

that gives an understanding of how the product works as a whole. The second contribution is our software solution that enables the utilization of the proposed image captioning engine. A full discussion on the software role in our product is provided in details with added enhancements.

The main focus of our paper was to deliver a fully functional software tool with a very powerful image captioning engine which consists of an encoder-decoder architecture with attention mechanism to generate reliable image descriptions. These descriptions are to be used to fill the missing image captions across different web pages on the internet. The full solution can be used through a web browser extension and only hovering over images to be processed and captioned in the back-end.

B. Future Work

There were so many different experiments, evaluations and various technical selections of architectures and adaptations have been left for the future research because of shortage of time. Some experiments may take days in design and days for a single run. Adaptation selection for different architectures and image captioning techniques using deep learning is also still an open area of research. Although we made sure to use the state of art algorithms but that doesn't mean that they serve our purpose perfectly.

Future work also accommodates intentions for contributions to solve challenges in this specific task and leverage the accuracy by means of changing different mechanisms involved in the image captioning process whether by altering evaluation metrics or using different architectures that may better suit the task.

The software part can be improved on different levels, the stability part that can be achieved by stress testing the product as a whole, deployment of the machine learning model on a remote server or using a cloud provider for higher scalability and nearly unlimited resources, reaching the point of a full online product without the need of any local dependencies.

REFERENCES

[1] Bryan A. Plummer and Liwei Wang and Chris M. Cervantes and Juan C. Caicedo and Julia Hockenmaier and Svetlana Lazebnik. (2015) "Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models" arXiv:1505.04870

[2] Kelvin Xu and Jimmy Ba and Ryan Kiros and Kyunghyun Cho and Aaron Courville and Ruslan Salakhutdinov and Richard Zemel and Yoshua Bengio. (2015) "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention" arXiv:1502.03044.

[3] MAX Image Caption Generator. http://max-image-caption-generator-web-app.codait-prod-41208c73af8fca213512856c7a09db52-0000.us-east.containers.appdomain.cloud/

[4] MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, Hamid Laga. (2018). A Comprehensive Survey of Deep Learning for Image Captioning. arXiv:1810.04020v2

[5] Sgrvinod, Kmario23, Jason718, Ngshya, a-PyTorch-Tutorial-to-Image-Captioning, (2020), GitHub repository, https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning

[6] Tsung-Yi Lin and Michael Maire and Serge Belongie and Lubomir Bourdev and Ross Girshick and James Hays and Pietro Perona and Deva Ramanan and C. Lawrence Zitnick and Piotr Dollár. (2014) "Microsoft COCO: Common Objects in Context" arXiv:1405.0312

APPENDIX

In this section we will show the results of the first experiment which is the optimizers variation, second experiment which is the different caption lengths thresholds, and the third experiment which is the different datasets. we will also view some of the mis-captioned images which motivated us to keep improving the model to reach satisfactory results.

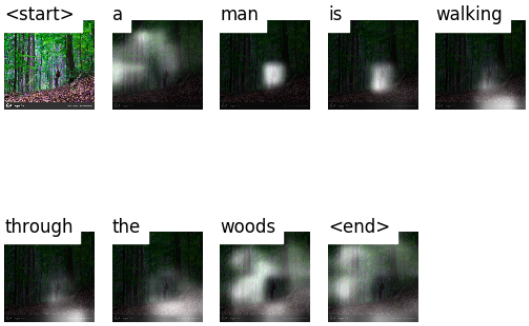

Fig. A.1. Result obtained when training using the ADAM optimizer.

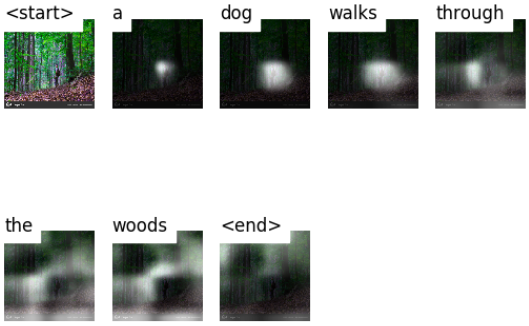

Fig. A.2. Result obtained when training using the RMSprop optimizer.

Fig. A.3. Result obtained when training using the ADAM optimizer.

Fig. A.4. Result obtained when training using the RMSprop optimizer.

Fig. A.5. Result obtained when training using the ADAM optimizer.

Fig. A.6. Result obtained when training using the RMSprop optimizer.

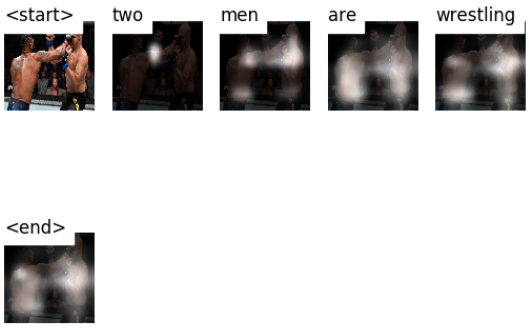

Fig. A.7. Result obtained when training using the ADAM optimizer.

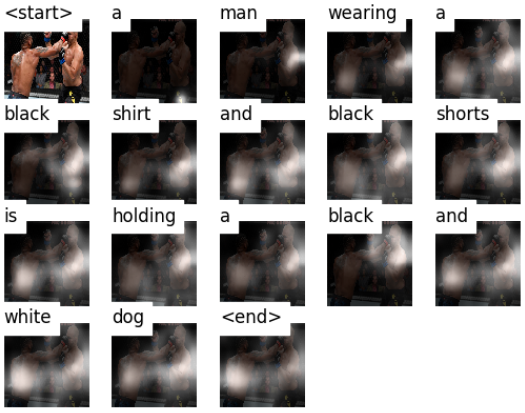

Fig. A.8. Result obtained when training using the RMSprop optimizer.

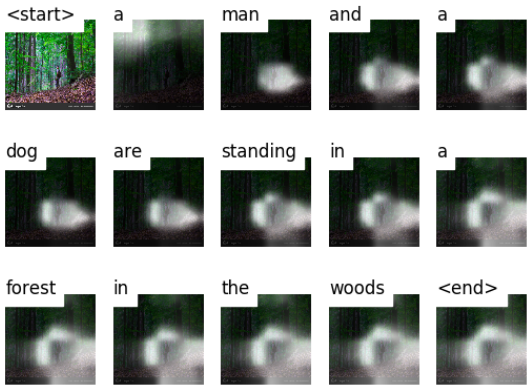

Fig. A.9. Result obtained when training the model with data caption length of 50.

Fig. A.10. Result obtained when training the model with data caption length of 50.

Fig. A.11. Result obtained when training the model with data caption length of 50.

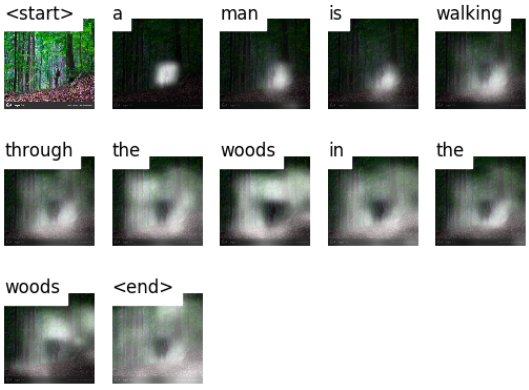

Fig. A.12. Coco Dataset: a man is walking through the woods in the woods.

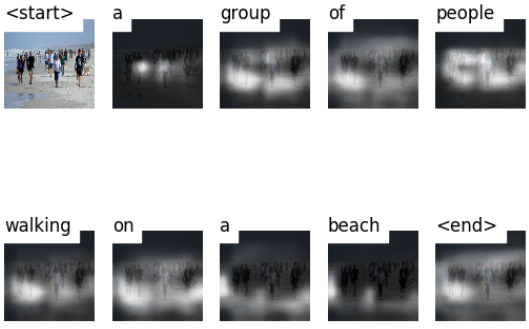

Fig. A.13. Coco Dataset:A group of people walking on a beach.

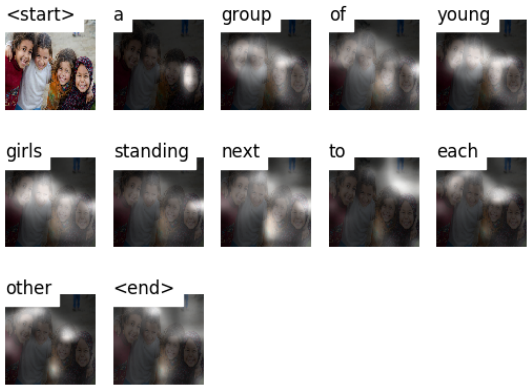

Fig. A.14. Coco Dataset: A group of young girls standing next to each other

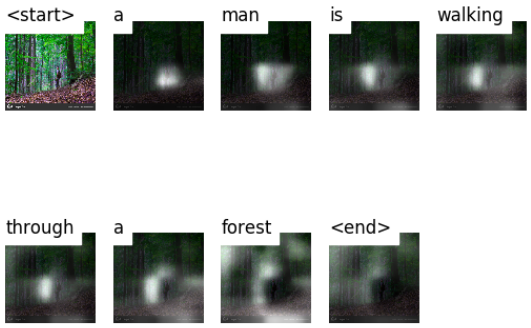

Fig. A.15. Flickr30k Dataset: a man is walking through a forest.

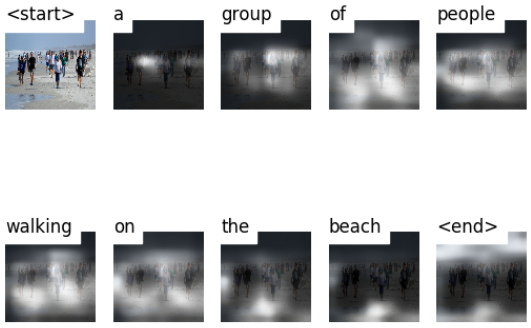

Fig. A.16. Flickr30k Dataset: a group of people walking on the beach.

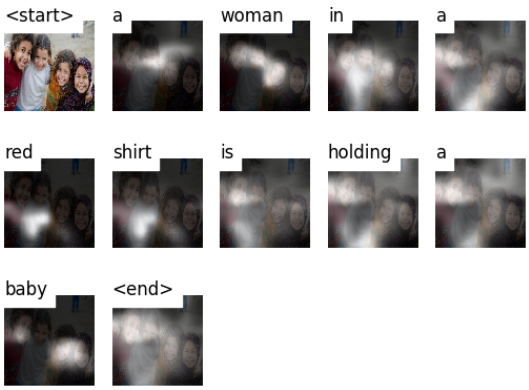

Fig. A.17. Coco Dataset: Flickr30k Dataset: a woman in a red shirt is holding a baby.

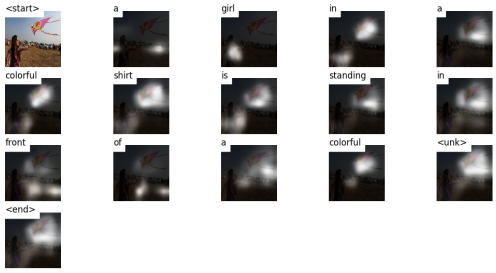

Fig. A.18. In this example the model failed to recognize the kite.

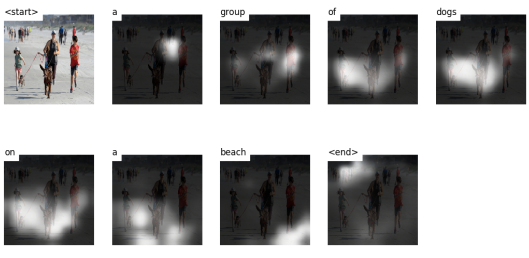

Fig. A.19. In this example the model failed to recognize the humans. It only recognized the dogs.

Fig. A.20. In this example, the model misclassified the pizza as a peach.