# Automated Image Caption Generation using Deep Learning

Sara Jerin Prithila, Mahmudul Hassan, Tamim Al Ahasan, Mahmudul Hasan,
Abid Hossain, Sania Azhmee Bhuiyan and Annajiat Alim Rasel
Department of Computer Science and Engineering (CSE)
School of Data and Sciences (SDS)
Brac University
66 Mohakhali, Dhaka - 1212, Bangladesh
{sara.jerin.prithila, mahmudul.hassan1, tamim.al.ahasan, mahmudul.hasan5,
abid.hossain}@g.bracu.ac.bd, saniaazhmee98@gmail.com, annajiat@gmail.com

***Abstract*—Image captioning means generating relevant texts from images that describe the image. With the growing advancement of deep learning, automatic image caption generation has become an exciting problem among researchers. Image captioning is used in various fields of computer science including computer vision and NLP. It is helpful in many cases such as visual impairment. Though many research works have already been published on this topic using different deep learning models, we can work with various features of the deep learning models to achieve better accuracy. In this paper, we are focusing on combining several deep-learning models to get the desired accuracy. VGG16 model is used which is a CNN architecture to extract essential features from the image. For the generation of relevant captions, LSTM is adopted rather than RNNs as LSTM generates better captions compared to RNNs and it consumes less time. Finally, the model is trained on Flickr 8k datasets.**

***Index Terms*—Image Captioning, Deep Learning, CNN, LSTM**

## I. Introduction

Image caption generation has become an interesting subject in recent times among researchers in the field of deep learning. This is mostly because it covers two major fields of artificial intelligence: computer vision and natural language processing. There are many practical uses of this application including SEO, image retrieval, etc. The task is to create a system that can produce machine-generated captions by analyzing the image that describes the image properly. Deep learning approaches are usually compatible with this type of problem. Our deep learning model mainly has to do two tasks: detecting objects from the images and generating captions from the extracted features.

Although there are many works already been published, we tried to combine several deep-learning algorithms to solve the problem in a different way. First, we have used a special technology called CNN that will look at pictures closely and find out all the things inside them. Next, we have another technology named LSTM which is great at understanding words and sentences. By embracing the power of CNNs, we enable our model to dissect and comprehend the multitude of details present within an image. The utilization of LSTMs for NLP is a pivotal aspect of our project's architecture. These recurrent neural networks are designed to capture sequential dependencies, making them ideally suited for tasks involving language generation. From selecting and preprocessing the dataset to constructing and training the intricate CNN-LSTM architecture, we tried to read images and explain what these images tell. Libraries such as TensorFlow and Keras have been used as they provide pre-built functions and classes to create, train, and evaluate neural networks. Natural Language Processing concepts such as tokenization, sequence generation, and word embeddings are used. Neural Networks Knowledge of how neural networks function, including the structure of layers, activation functions, loss functions, and optimization algorithms, is crucial for building and training the CNN and LSTM components. Data preprocessing is used to preprocess both image and text data before feeding them into the neural network. Resizing images, normalizing pixel values, and tokenizing text are these kinds of tasks. The ability to generate insightful captions for images finds applications in numerous domains. From assisting the visually impaired in understanding the content of images to enhancing the searchability of visual data in the ever-expansive digital landscape, our image caption generator underscores the profound ways in which AI can reshape human interactions with technology. So, if we break down our workflow, firstly, we choose the right set of pictures. For our model, we have chosen the Flickr 8k dataset. Next, we built a structure mixed up with CNNs and LSTMs for understanding and creating sentences. Lastly, we trained the system using TensorFlow and Keras libraries. We used the Flickr dataset which had around 8000 images and 40,000 captions.

In this paper, we have divided the proceedings into different sections. Section 2 describes related work in image captioning. Section 3 is our proposed methodology whereas sections 4 and 5 describe data preprocessing and model architecture. Finally, section 6 is the result analysis part, and section 7 for the conclusion summarizing the full workings.

## II. Related Work

In their study, Kulkarni et al. [1] proposed the collection of image attribute tuples, consisting of object, visual attribute,

and spatial connections, as a means to enhance previous techniques such as the template-based approach. Subsequently, they used n-gram-based LMs to produce words for producing the final caption. On the other hand, using these approaches often creates noisy evaluations of the image features and flawed patterns between the pictures. It is possible that further re-ranking methods based on textual qualities will be necessary for more accurately generated captions.

Image captioning is treated as a retrieval task by retrieval-based on image captioning techniques. using the distance measure to find related captioned images, after which the retrieved captions are modified and combined to produce captions [2]. However, in order for these algorithms to work with picture queries, extra steps like modification and generalization processes are typically required. Recently we are seeing that a major proportion of the works starts to use deep learning for generating captioning which is a result of recent developments in deep learning. CNN is frequently used as an encoder to extract data from images. RNN closely resembles CNN. This representation is decoded into a description in natural language using RNN. Before being sent to the decoder, the encoder compressed all input data into a fixed-length single vector. It may outcome in information loss to compress long and complex input sequences using this vector [3]. For this complexity, an attention method has been proposed. They replaced the fixed length vector that was a result of the image encoder and in place of that the newly produced attention vector was used. The Attention mechanism, which is part of the encoder-decoder architecture, allows the decoding process to focus on the significance and characteristics of the given picture at every cycle when its result patterns are being developed [4]. An attentive encoder-decoder model was first presented by Xu et al. [5]. Semantic attention [6] and review networks [7] are just two of the many helpful enhancements that have been suggested for the encoder-decoder structure. A semantic attention framework that fuses semantic understanding with a characteristic description of the image/encoding is utilized. Yang et al.'s [7] proposal for "the reviewer module" is a novel module with the aim of enhancing the encoder-decoder system. In order to calculate the weights assigned to hidden states, this module applies review steps to the encoder's invisible states and provides a vector at each iteration. The approach used by Anderson et al. [8] additionally modified top-down and bottom-up attention mechanisms, allowing for the calculation of attention at the level of objects and other salient picture regions. They suggested a hierarchical deep neural network for Su et al. [9]. The lowest layer and highest layer make up the suggested architecture. The lowest layer gathers perceptual and top-level semantic data from the picture and recognized areas, while the highest layer combines the two of them with an attention system for caption synthesis.

Sequence-to-sequence learning has been achieved through recurrent neural networks. Recognizing speech, translation by machine, and image labeling are a few examples of sequence learning applications that utilize RNN [10]. The nodes of an RNN may create a graph that is directed, make use of internal memory for evaluating streams of inputs, and and swap the present input with an older hidden state. This makes RNN comparatively suitable for sequence to sequence learning tasks, which move in only one way and do not generate cycles. Feed-forward systems never loop but rather move in one direction. Due to the difficulty presented by disappearing and inflating gradients, traditional RNNs are unable to predict words that are involved in long-range connections. As a consequence of this, LSTM, the more refined and advanced variation of RNN, gets used for a lot of research.Compared to the conventional RNN units, an LSTM utilizes memory layer to retain data in the storage for a longer amount of time and choose what data should be kept and what data should be discarded [11].

## III. METHODOLOGY

In this research, we propose an encoder-decoder method to automatically generate captions for given photos. Figure 1 depicts the whole model. The entire system is composed of numerous subsystems, each of which is responsible for a separate set of responsibilities, such as object detection and feature extraction. The system gets an image, analyses it, and then provides the correctly formatted data to the object detector in our proposed method. A pre-trained VGG16 encoder will handle the task of object detection in this model. The feature vector of the image, which describes its contents, is extracted using the encoder. The detector produces a list of things that were found, which is put through some pre-calculated heuristics to create a begining for our model. An extractor of features and a language model combine to create the caption generator. This group of sub-systems is in charge of bringing in an image, utilizing the feature extractor to extract relevant characteristics from it, and then handing the extracted features and the origin over the LM, which through iterative method constructs or title for the image. In this situation, the first beginning point is the one that our heuristics produce utilizing objects that are observed. Each iteration of the language model adds this beginning point. Because we encourage specific information toward the language model in our proposed model, we provide image caption generation method with pre-calculated origins. For instance, if a person was found in an image, we could now construct a starting point called "startseq" and an ending point called "endseq" rather than using the initial marker as a starting point. The subsequent generation of the caption would then be the caption generators' responsibility. As a result, we become less dependent on the caption generator for comprehending visual objects, which frees us to transfer accountability across networks. We are going to depend on the object detection algorithm for obtaining information from the image instead of overloading image caption generation model which can produce a bad outcome.

## IV. DATA PREPROCESSING

### A. Image Feature Extraction

The architectural modifi- cation of the VGG16 model consisted of eliminating the prediction layer and focusing on the

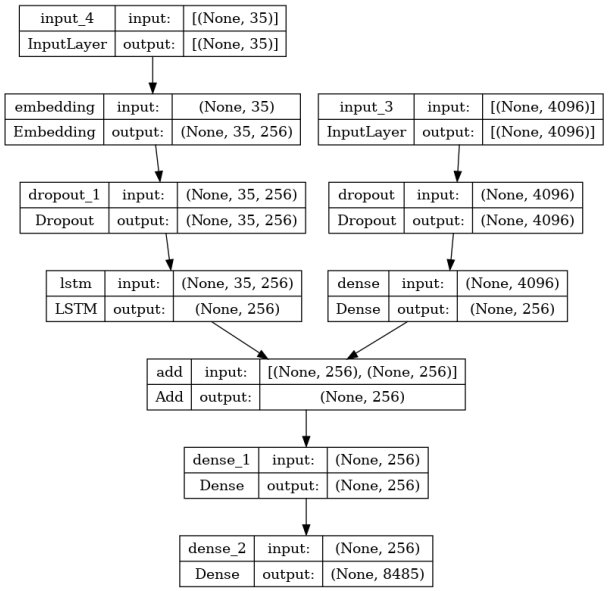


Fig. 1. Block diagram of the model

preceding layer for the extraction of features. By leaving the prediction layer, we dealt with the layer preceding the output layer of VGG16. The input image received by VGG16 has a predetermined dimension of 224x224. After extracting the features, we obtained the image ID by separating the image name from the extension and indexing it to the zero position. Using the "pickle" Python model, we saved the dictionary containing the extracted image features.

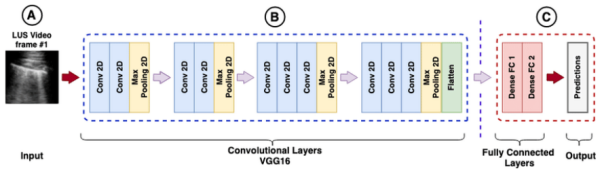


Fig. 2. Feature extraction using VGG16

### B. Text Preprocessing

To load the caption data, we had to create the file path launching with the base directory by reading the whole caption text file. The first line of the caption.txt file was removed since it had labeling that was, image and caption. Now, we require to have a caption that corresponds to each picture ID that we have. In order to achieve this, we used a mapping dictionary in which the picture id was assigned as the key and the corresponding caption as the value. Nevertheless, it was necessary to take into account the storage of several captions for a single image. In the case of a defective caption file, cases, where a line consists of just a single letter, are regarded as invalid and are thus excluded. The JPG file extension was required to be removed from the image ID.

In order to begin preprocessing the captions, we first did some basic text cleaning, such as converting all of the textual

| |
|---|
| Young boy with a black "" ADHD "" t-shirt nodding off |
| "Two people , in bike gear , stand with bicycles near a rocky waterbed ." |
| "On grass , two brown dogs tussle for orange toy ." |

Fig. 3. A snapshot of the caption text

contents to lowercase. Special tokens and digits did not serve a significant impact on text prediction. So using regular expressions, non-alphanumeric characters such as, from Fig 4, were stripped out of the text. These have been altered to a white space. After that, many consecutive whitespaces were eliminated using regular expressions in order to lessen the amount of unnecessary white spaces, with the help of regular expression [^A-Za-z].

| | |
|---|---|
| i) Numeric values | 0 1 2 3 4 5 6 7 8 9 |
| ii) Punctuation marks | ! @ # $ % ^ & * ( ) - _ + = [ ] { } ; : ' " < > / ? |
| iii) Special symbols | ~ , \| `` |

Fig. 4. List of items removed

A "startseq" and a "endseq" were added to the first and last of each and every caption in order to denote the starting and ending of a sequence token. This allows the model to start and stop generating text with the assistance of an indicator.

## V. MODEL ARCHITECTURE AND TRAINING

### A. Encoder: Pretrained Feature Extractor

A pre-trained feature extractor that can extract image features from a given image is used in the model's initial stage. A pre-trained VGG16 model was used as the foundation of our network. To avoid classifying the image, the softmax layer of the VGG model is stripped, and information about the full image is instead acquired. The VGG model's hidden layers are made up of two convolutional layers, two max-pooling layers, and a layer to cut the size in half. We did not carry out any transfer learning, and we employed the models just as they were, as provided by frameworks like TensorFlow, in order to offer preliminary results for the experiment. Figure 3 depicts the VGG model's executive summary. And as a result, the network now produces an image feature representation rather than a classification label. The first part of the architecture is now complete.

### B. Object Detection

The following phase of the approach requires the development of a detecting module. We adopted the simplicity approach and chose prepackaged structures that have a track record of success. Our object detector was a CNN model with VGG16 as its foundation. The Flickr 8k dataset was used to train the model for object detection. The detector's job was to find things that belonged to a particular class.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_2 (InputLayer) | [(None, 224, 224, 3)] | 0 |
| block1_conv1 (Conv2D) | (None, 224, 224, 64) | 1792 |
| block1_conv2 (Conv2D) | (None, 224, 224, 64) | 36928 |
| block1_pool (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 112, 112, 128) | 73856 |
| block2_conv2 (Conv2D) | (None, 112, 112, 128) | 147584 |
| block2_pool (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 56, 56, 256) | 295168 |
| block3_conv2 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_conv3 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_pool (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 28, 28, 512) | 1180160 |
| block4_conv2 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| block4_conv3 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| block4_pool (MaxPooling2D) | (None, 14, 14, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| block5_conv2 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| block5_conv3 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| block5_pool (MaxPooling2D) | (None, 7, 7, 512) | 0 |
| flatten (Flatten) | (None, 25088) | 0 |
| fc1 (Dense) | (None, 4096) | 102764544 |
| fc2 (Dense) | (None, 4096) | 16781312 |

Fig. 5. VGG Model Summary

### C. Decoder

The construction of a LM that was going to be in responsible for taking in the refined image features and producing a description using the associated information was necessary as the last phase of the framework. In order to collect data in batches and prevent session crashes, we first constructed a data generator. The length of the sequence vectors is padded to make sure that they are all the same length or the length of the largest sequence. The result of the VGG16 model revealed that the shape of the features is 4096. The input layer, embedding layer, dropout layer, and dense layer make up the decoder. The dropout layer is used to add regularization to the data, prevent overfitting, and drop out a portion of the data from the layers.

A single-dimensional linear layer array makes up the dense layer. We process it by creating a single dense layer from the LSTM dropout layer of the text and the dropout layer of the image. Dividing the number of captions in the data used for training set by the batch size, the total number of iterations per epoch is computed.

$$Stepsperepoch = \frac{totalcaptionsinthetrainingset}{Batchsize}$$

When doing discrete classification tasks where the classes are mutually exclusive, the loss is calculated using sparse softmax cross entropy. Adam is the optimizer that is used to automatically modify the model's learning rate throughout the course of the epochs. In the final dense layer, linear activation is used. The categorical output in this case is 8483, representing one hot encoding that we performed. Because we had previously retrieved the characteristics in this case, we did not utilize CNN to extract the image instead opting to use VGG. If the word had a higher probability, it meant that we would get it. The Flickr 8k dataset, which contains 5 captions for each image in the training dataset, is used to train the model. The training dataset contains 8091 pictures in total. The training caption dataset is shifted by one unit for training the LSTM model to correspond various words with one another for better prediction.

### D. Caption generation

To construct our model for title production, the stages are then spaced apart. Both the object identification and feature extraction processes get the same image and work in parallel to provide distinct results. Because there are no dependencies between the two processes, this is straightforward. Since the language modelling step depends on the output of the two stages before it in order to function, it must be arranged to ensure it continues operating after they both have ended. Algorithm 1 [12] of Figure 4 depicts the end-to-end system, which is in charge of caption creation.

**Algorithm 1:** Caption generation process

```
1 : caption = startMarker
2 : for i=0 to max length of sequence
3 :     sequence = tokenizer.textsToSequences(caption)
4 :     sequence = padSequences(sequence, maxLength)
5 :     nextword = predictNextWord(image, sequence)
6 :     nextword = indexof(nextword)
7 :     word = indexToWord(nextword, tokenizer)
8 :     if word is None
9 :             break
10:     caption.joinWith(word)
11:     if word is endMarker
12:             break
13: return caption
```

Fig. 6. Caption Generation Algorithm

This section describes how the code described in the system works. The system takes a picture which is then translated into a feature vector and returns a caption describing the contents of the input image. After adding a start tag for the generation process, the algorithm begins iterating over the maximum length of the sequence. Lines 3-4: Send the caption to the tokenizer layer, which converts the first preset number of unique words into integer tokens before padding the sequence. Lines 5-7: The produced starting point and retrieved

features are then delivered in numerous sequential passes to the language modeler. At each pass, the model's output is added to the state of its input. The next sequential model pass is then given this updated state. Until an endpoint is reached, this cycle of state updating and change is repeated. Our index now has a high probability. Language-specific heuristics are numerous and range in complexity; a few of them are as basic as identifying a specific things, this is when object additions are necessary to create the following word. There can be several captions produced, each utilizing a different object collection. Then, by comparing these captions to the original set, it is possible to assess the accuracy of each generation set.

## VI. RESULTS AND ANALYSIS

### A. Dataset

The dataset we used for this experiment is called the Flickr 8k dataset. This dataset as the name suggests has 8091 images that have been carefully manually selected to represent avariety of events and occurrences from six different Flickr groups. Each image has corresponding five distinct captions that describe the prominent features and events that are present in the picture. The captions are in English. There are a total of 40455 captions present in the dataset and all of them are trainable. Total vocabulary size of the captions is 8485.

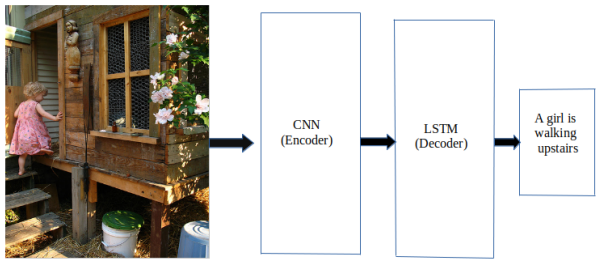


Fig. 7. Image captioning process

### B. Result

Figures 7, 8, and 9 show prediction for an image in the validation set with actual captions and predicted captions using the VGG16 model as an encoder and LSTM network as a decoder with a number of epochs set to 25 for optimum performance using 7281 images from the Flickr 8k dataset as training dataset, which is 90% of the total 8091 images and vocabulary size of 8485 unique words. After repeated efforts, we discovered that decreasing the batch size and increasing the number of epochs yielded superior results. The ADAM adaptive gradient optimizer is a good candidate for our proposed architecture since it provides good accuracy at a faster rate of convergence. We noticed that the loss score, which was 3.8429 at the start of training, started to fall and slowed down at 2. The loss presentation with ADAM is shown in Figure 6. We believe that increasing the number of epochs will allow us to further reduce the loss score. By increasing the number of epochs, we could improve the overall result. However, increasing the epochs takes a long time and places additional burden on our system.

We evaluated our results using the BLEU index, which is a popular statistic for machine translation evaluation and one of the first metrics used to evaluate image descriptions. To avoid overly short sentences, it computes the geometric mean of n-gram precision scores multiplied by the brevity penalty. As the

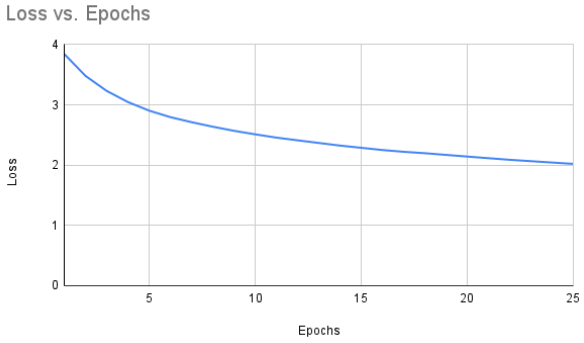


Fig. 8. Loss presentation with ADAM

BLEU score alone does not completely reflect the effectiveness of a model so along with BLEU-1 and BLEU-2 scores we wanted to select individual images and judge the results with the help of human evaluation.

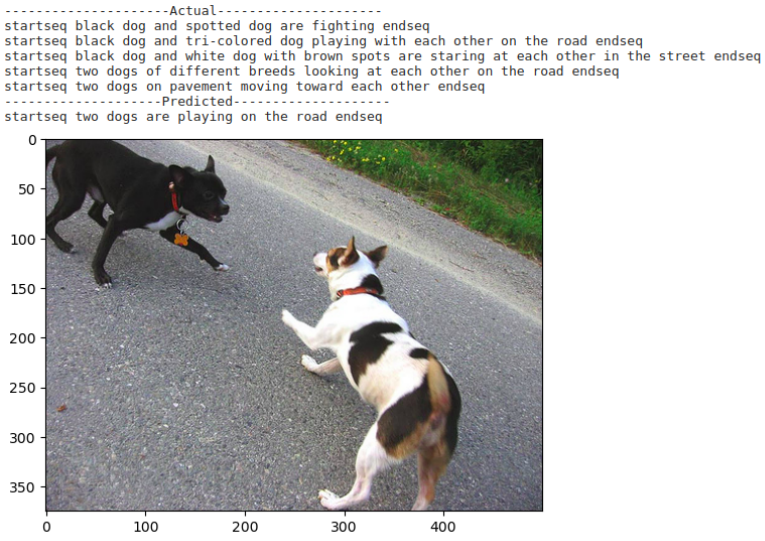


Fig. 9. Actual vs predicted caption of an image

The BLEU-1 score is 0.532248 and the BLEU-2 score is 0.305608 in our training result. We can see in the Fig-7 image of the single dog that our model correctly predicted both the dog's coat and ear color. But it overlooked the crucial detail that it was shaking its head. Also, our algorithm accurately predicted the number of dogs and even the activity that they are playing on the road in the image of two dogs. The color of the kayak in Fig-9 was correctly predicted by our model, but the

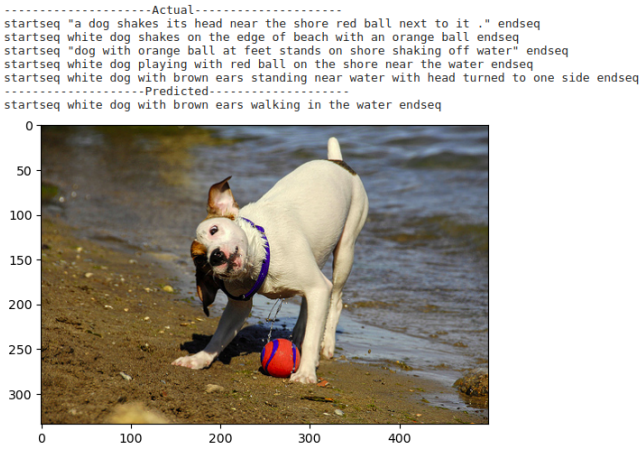


Fig. 10. Actual vs predicted caption of an image

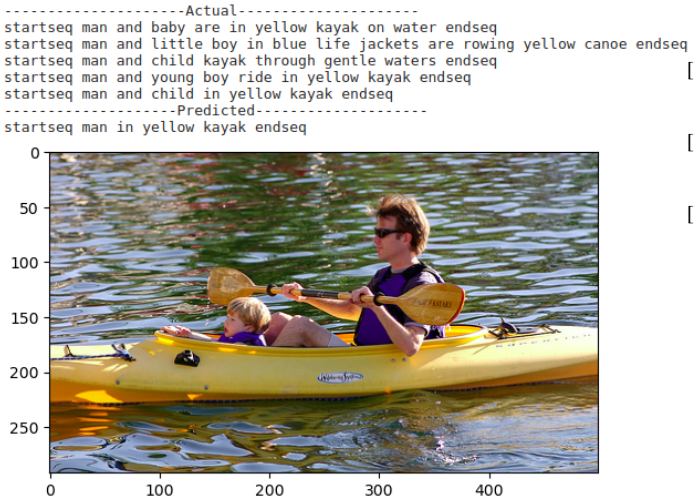


Fig. 11. Actual vs predicted caption of an image

child was not mentioned. It also suggested that the individual is a man. The BLEU-1 score specifies that a substantial portion of the words in the captions were accurately predicted by the model we used. The BLEU-2 score indicates that although the descriptions are understandable the algorithm may have neglected some features of the images.

## VII. Conclusion

For our research, we attempted to use a model that could take an image and provide a caption that properly described the image's key elements. We used CNN and LSTM for our task which produced great results. Despite the fact that our algorithm struggled to identify several elements in the image, the outcome it produced was comprehensible. In the future, we'd like to work with a much bigger dataset that includes a greater number of images from everyday incidents and occurrences. Additionally, we want to include an attention mechanism so that we can get greater results and handle much more complicated data.

## References

[1] G. Kulkarni, V. Premraj, V. Ordonez, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg, "Babytalk: Understanding and generating simple image descriptions," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 12, pp. 2891–2903, 2013.

[2] P. Kuznetsova, V. Ordonez, T.L. Berg and Y. Choi, "Treetalk: Composition and compression of trees for image descriptions," Transactions of the Association for Computational Linguistics, vol. 2, pp. 351-362.,2014.

[3] J. K.Chorowski, D. Bahdanau, D.Serdyuk, K.Cho and Y.Bengio, "Attention-based models for speech Recognition", Advances in neural information processing systems, pp. 577-585, 2015.

[4] D. Bahdanau, K. Cho and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473,2014.

[5] K. Xu et al., "Show, attend and tell: Neural image caption generation with visual attention," In International conference on machine learning ,pp. 2048–2057,2015.

[6] Q. You,H.Jin,Z. Wang,C. Fang and J.Luo , "Image captioning with semantic attention,". In: CVPR, pp. 4651–4659,2016.

[7] L.Yang, K.D .Tang, J. Yang and, L.J. Li , "Dense captioning with joint inference and visual context,". In: CVPR, pp 1978–1987,2017.

[8] P.Anderson et al., "Bottom-up and top-down attention for image captioning and visual question answering," In CVPR, 6077–6086,2018.

[9] Y. Su, Y. Li, N. Xu, and A.A. Liu, "Hierarchical Deep Neural Network for Image Captioning," Neural Processing Letters,pp. 1-11,2019.

[10] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," ACM Computing Surveys (CSUR), vol. 51, no. 6, pp. 1–36, 2019.

[11] Q. Wang and A. B. Chan, "CNN+ CNN: Convolutional decoders for image captioning," in 31st IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018), 2018, pp. 1–9.

[12] I. Azhar, I. Afyouni and A. Elnagar, "Facilitated Deep Learning Models for Image Captioning," 2021 55th Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, USA, 2021, pp. 1-6, doi: 10.1109/CISS50987.2021.9400209.