**Image Captioning, Capstone Project: Final Report**

**Smita Paul**

**Introduction**

Image C**aption generation**is to identify textual description for a given image. As human it is easy for us to describe image. But for computer to make prediction is the way difficult.

Image captioning is a popular research area of Artificial Intelligence (AI) that deals with image understanding and a language description for that image. Image understanding needs to detect and recognize objects. It also needs to understand scene type or location, object properties and their interactions.

We must first understand how important this problem is to real world scenarios. Let’s see few applications where a solution to this problem can be very useful.

In real world scenarios it is very important to understand

* Medical image understanding - Medical imaging is widely used in clinical practice for diagnosis and treatment. Medical images, such as radiology and pathology images, are widely used in hospitals for the diagnosis and treatment of many diseases, such as pneumonia and pneumothorax.

The reading and interpretation of medical images are usually conducted by specialized medical professionals.

Report-writing can be error-prone for unexperienced physicians, and timeconsuming and tedious for experienced physicians

* Aid to the blind — Images with generate captions that can be read out loud to the visually impaired so that they can get a better sense of what is happening around them.
* Tesla/Google Self Drive Cars: All the self drive cars are using image/video processing with neural network to attain their goal.
* CCTV cameras are everywhere today, but along with viewing the world, if we can also generate relevant captions, then we can raise alarms as soon as there is some malicious activity going on somewhere. This could probably help reduce some crime and/or accidents.
* Google Image Search as good as Google Search, as then every image could be first converted into a caption and then search can be performed based on the caption.
* Social Media : Social Media like Facebook can make section of image by finding capions of image

It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. Generating well-formed sentences requires both syntactic and semantic understanding of the language .

Understanding an image largely depends on obtaining image features. The techniques used for this purpose can be broadly divided into two categories: (1) Traditional machine learning based techniques and (2) Deep machine learning based techniques.

This project explores deep learning approach for image captioning.

Data

To train model for image captioning we need huge image data with captions. Below are few open source datasets available for Image captioning

* Flickr 8k : This contains 8k images
* Flickr 30k : This contains 30k images
* MS COCO: COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features. It has 330k images.

For this case study of image captioning is the Flickr8K dataset. It is realistic and relatively small. Dataset has below 2 folders

**Flicker8k\_Dataset**: Contains 8092 photographs in jpg format.

**Flickr8k\_text**: Contains a number of files containing different sources of descriptions for the photographs.

This dataset contains 8092 images each with 5 captions

Concept of Attention Mechanism:

A “classic” image captioning system would encode the image, using a pre-trained Convolutional Neural Network that would produce a hidden state *h*.

Then, it would decode this hidden state by using a Recurrent Neural Network (RNN) and generate recursively each word of the caption. Such a method has been applied by several groups, including [11]:

The problem with this method is that, when the model is trying to generate the next word of the caption, this word is usually describing only a part of the image. Using the whole representation of the image*h*to condition the generation of each word cannot efficiently produce different words for different parts of the image. This is exactly where an attention mechanism is helpful.

With an attention mechanism, the image is first divided into *n*parts, and we compute with a Convolutional Neural Network (CNN) representations of each part *h1,…,hn.*When the RNN is generating a new word, the attention mechanism is focusing on the relevant part of the image, so the decoder only uses specific parts of the image

**Types of Attention Mechanism :**

**Global Attention**

Global attention takes into consideration all encoder hidden states to derive the context vector (c(t)). In order to calculate c(t), we compute a(t) which is a variable length alignment vector. The alignment vector is derived by computing a similarity measure between h(t) and h\_bar(s) where h(t) is the source hidden state while h\_bar(s) is the target hidden state. Similar states in encoder and decoder are actually referring to the same meaning.

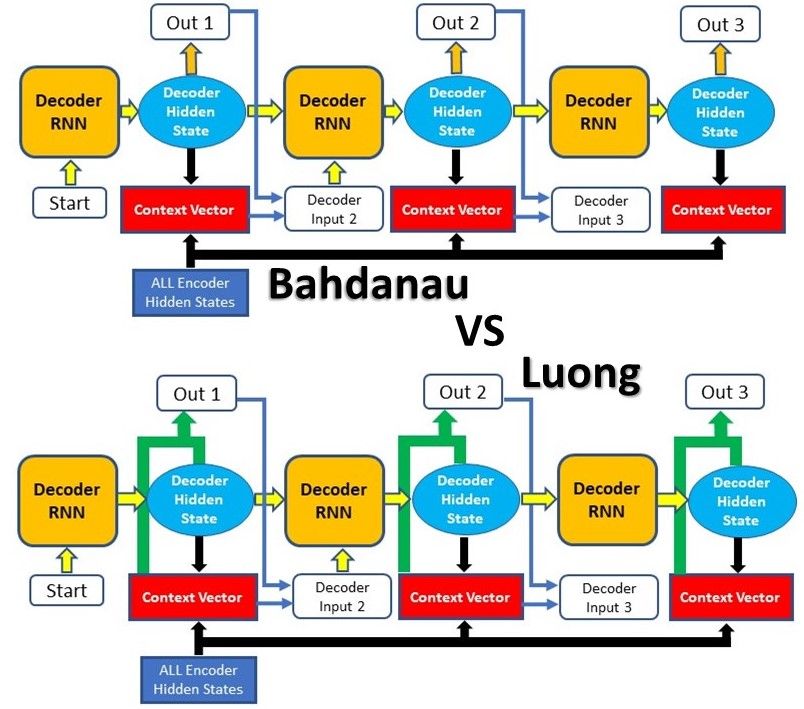
Attention is placed on all source positions.

**Local Attention**

As Global attention focus on all source side words for all target words, it is computationally very expensive and is impractical when translating for long sentences. To overcome this deficiency local attention chooses to focus only on a small subset of the hidden states of the encoder per target word.

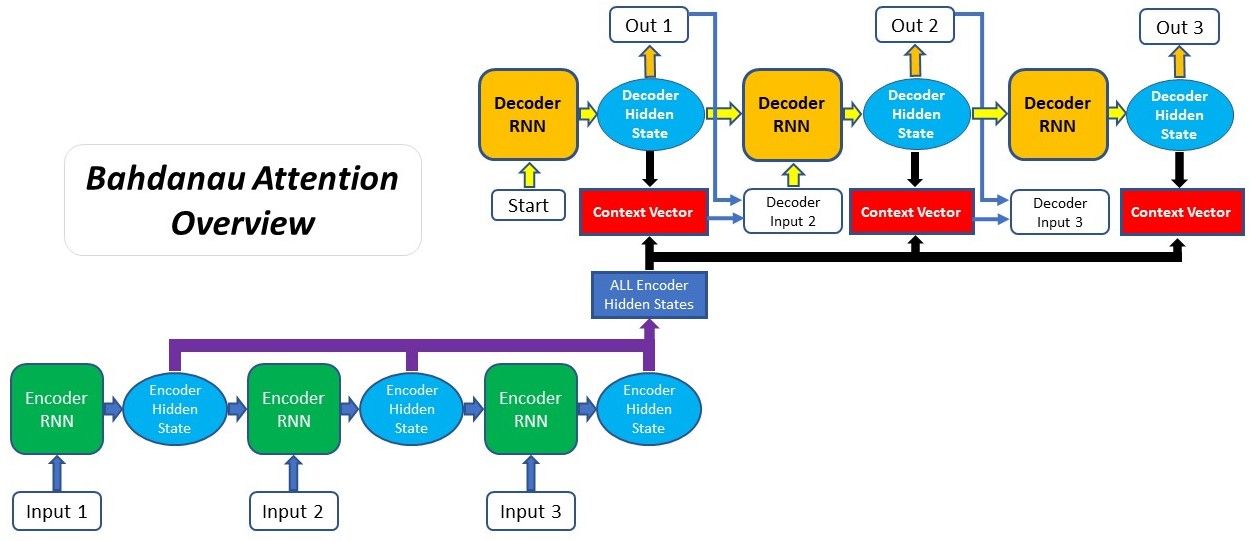
Attention is placed only on a few source positions.

**This project we will use local attention i.e Bahdanu Attention**



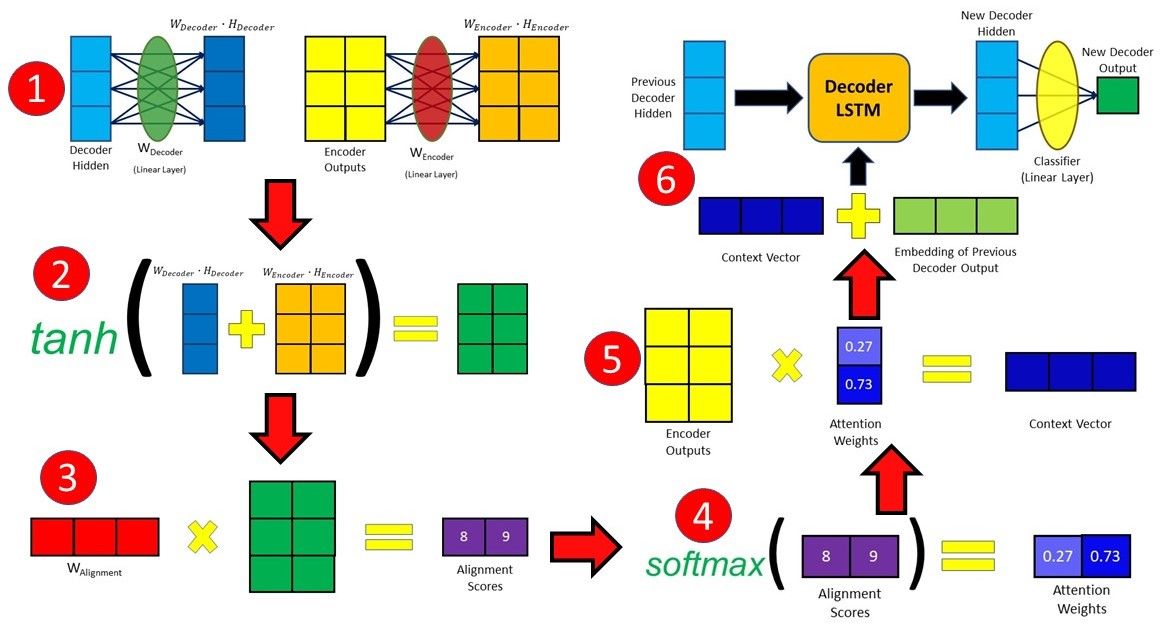
While the underlying principles of Attention are the same in these 2 types, their differences lie mainly in their architectures and computations.

The first type of Attention, commonly referred to as Additive Attention, came from a paper by [Dzmitry Bahdanau](https://arxiv.org/pdf/1409.0473.pdf), which explains the less-descriptive original name. The paper aimed to improve the sequence-to-sequence model in machine translation by aligning the decoder with the relevant input sentences and implementing Attention



**The entire step-by-step process of applying Attention in Bahdanau’s paper is as follows:**

1. Producing the Encoder Hidden States - Encoder produces hidden states of each element in the input sequence
2. Calculating Alignment Scores between the previous decoder hidden state and each of the encoder’s hidden states are calculated (Note: The last encoder hidden state can be used as the first hidden state in the decoder)
3. Softmaxing the Alignment Scores - the alignment scores for each encoder hidden state are combined and represented in a single vector and subsequently softmaxed
4. Calculating the Context Vector - the encoder hidden states and their respective alignment scores are multiplied to form the context vector
5. Decoding the Output - the context vector is concatenated with the previous decoder output and fed into the Decoder RNN for that time step along with the previous decoder hidden state to produce a new output
6. The process (steps 2-5) repeats itself for each time step of the decoder until an token is produced or output is past the specified maximum length.



Ref: <https://blog.floydhub.com/attention-mechanism/>

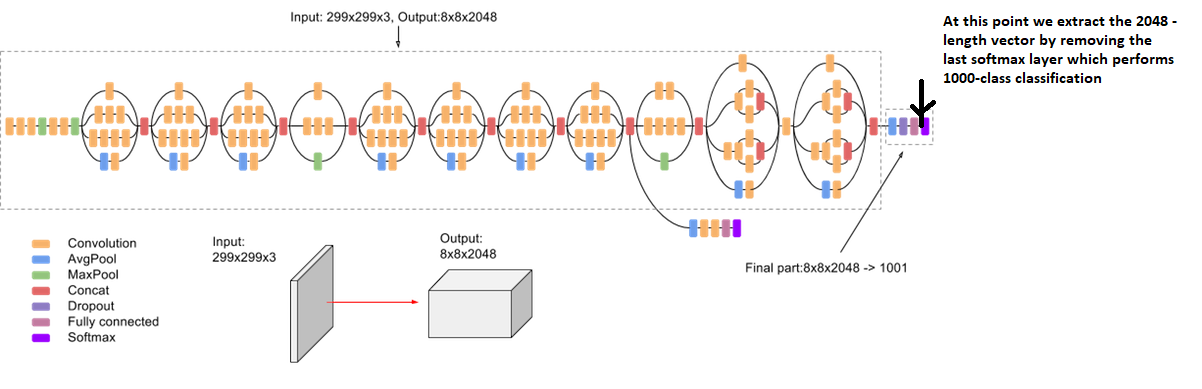
Image Feature Extraction by Inveption V3

We need to convert every image into a fixed sized vector which can then be fed as input to the neural network. For this purpose, we opt for **transfer learning** by using the InceptionV3 model (Convolutional Neural Network) created by Google Research.

This model was trained on Imagenet dataset to perform image classification on 1000 different classes of images. However, our 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.**

Hence, we just remove the last softmax layer from the model and extract a 2048 length vector (**bottleneck features**) for every image as follows:

To train model for image captioning we need huge image data with captions. Below are few open source datasets available



We save all the image features in a Python numpy array and save npy files.

Pre-processing of Text

Need to convert captions of image into understandable format. So we need to perform few pre-processing steps on text.

Below steps will perform:

1. Tokenize the captions (for example, by splitting on spaces). This gives us a vocabulary of all of the unique words in the data (for example, "surfing", "football", and so on).
2. Next, limit the vocabulary size to the top 5,000 words (to save memory). You'll replace all other words with the token "UNK" (unknown). For words not appearing in the vocabulary we will give it <unk> notation
3. You then create word-to-index and index-to-word mappings.
4. Finally, you pad all sequences to be the same length as the longest one.
5. Then we will create vector notations for each word in our vocabulary. Now that we have got the sequences to the words in our captions, the sequences are of different length. So, we need pad the sequences to the maximum length of the captions.

Model

1. In this example, will extract the features from the lower convolutional layer of Inception V3  giving us a vector of shape (224, 224, 3).
2. Then squash that to a shape of (64, 512).
3. This vector is then passed through the CNN Encoder (which consists of a single Fully connected layer).
4. The RNN (here GRU) attends over the image to predict the next word.

Checkpoints

Saving Checkpoint TensorFlow objects provide an easy automatic mechanism for saving and restoring the values of variables they use. All prefixes are grouped together in a single checkpoint file ('./tf\_ckpts/checkpoint') where the CheckpointManager saves its state.

Checkpoints capture the exact value of all parameters (tf.Variable objects) used by a model. Checkpoints do not contain any description of the computation defined by the model and thus are typically only useful when source code that will use the saved parameter values is available.

Training

Now we extract the features stored in the respective .npy files and then pass those features through the encoder.

The encoder output, hidden state(initialized to 0) and the decoder input (which is the start token) is passed to the decoder.

The decoder returns the predictions and the decoder hidden state. The decoder hidden state is then passed back into the model and the predictions are used to calculate the loss. Use teacher forcing to decide the next input to the decoder. Teacher forcing is a method for quickly and efficiently training recurrent neural network models that use the ground truth from a prior time step as input. Teacher forcing is the technique where the target word is passed as the next input to the decoder.

The final step is to calculate the gradients and apply it to the optimizer and backpropagate.

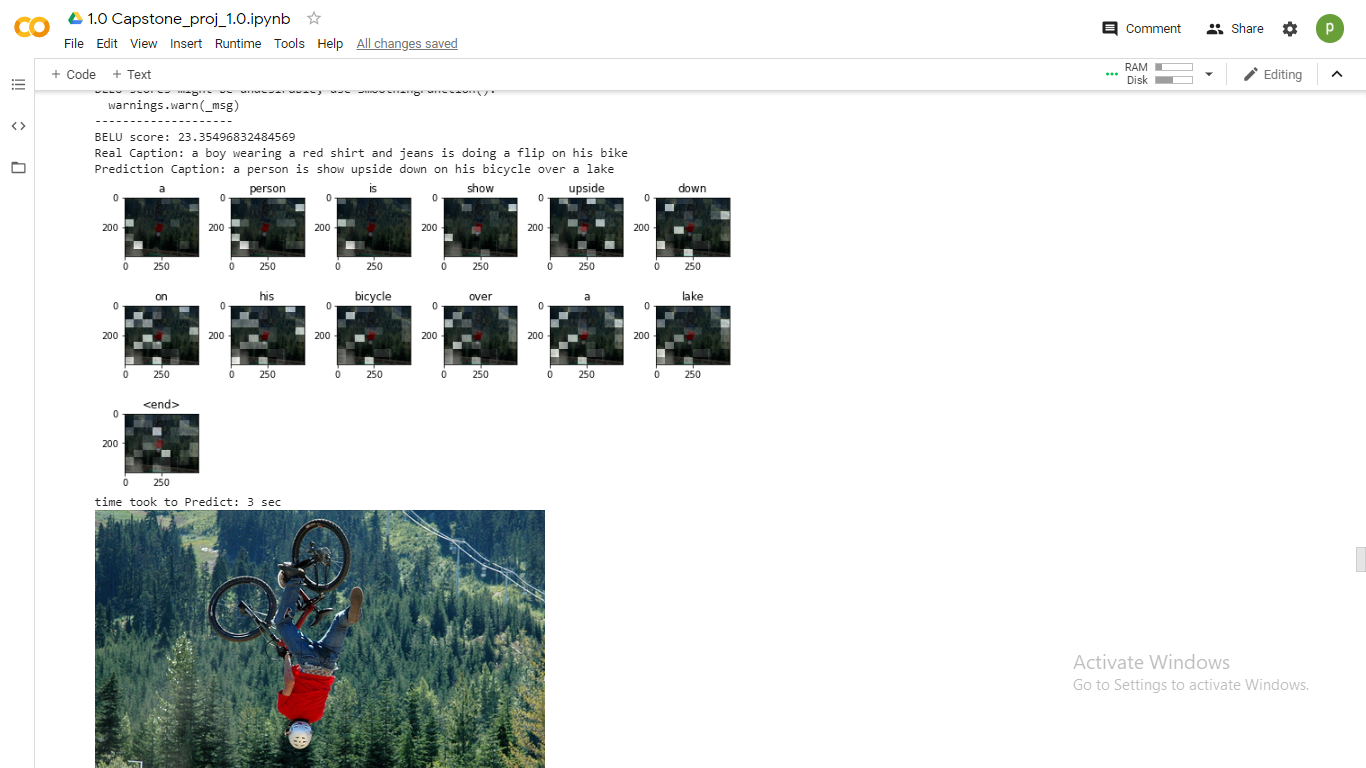
Loss Plot

To Do:

Metric used : BLEU Score

The Bilingual Evaluation Understudy Score, or BLEU for short, is a metric for evaluating a generated sentence to a reference sentence.

* We use the **BLEU** measure to evaluate the result of the the test set generated captions. The **BLEU** is simply taking the fraction of n-grams in the predicted sentence that appears in the ground-truth.
* BLEU is a well-acknowledged metric to measure the similarly of one hypothesis sentence to multiple reference sentences. Given a single hypothesis sentence and multiple reference sentences, it returns value between 0 and 1. The metric close to 1 means that the two are very similar.

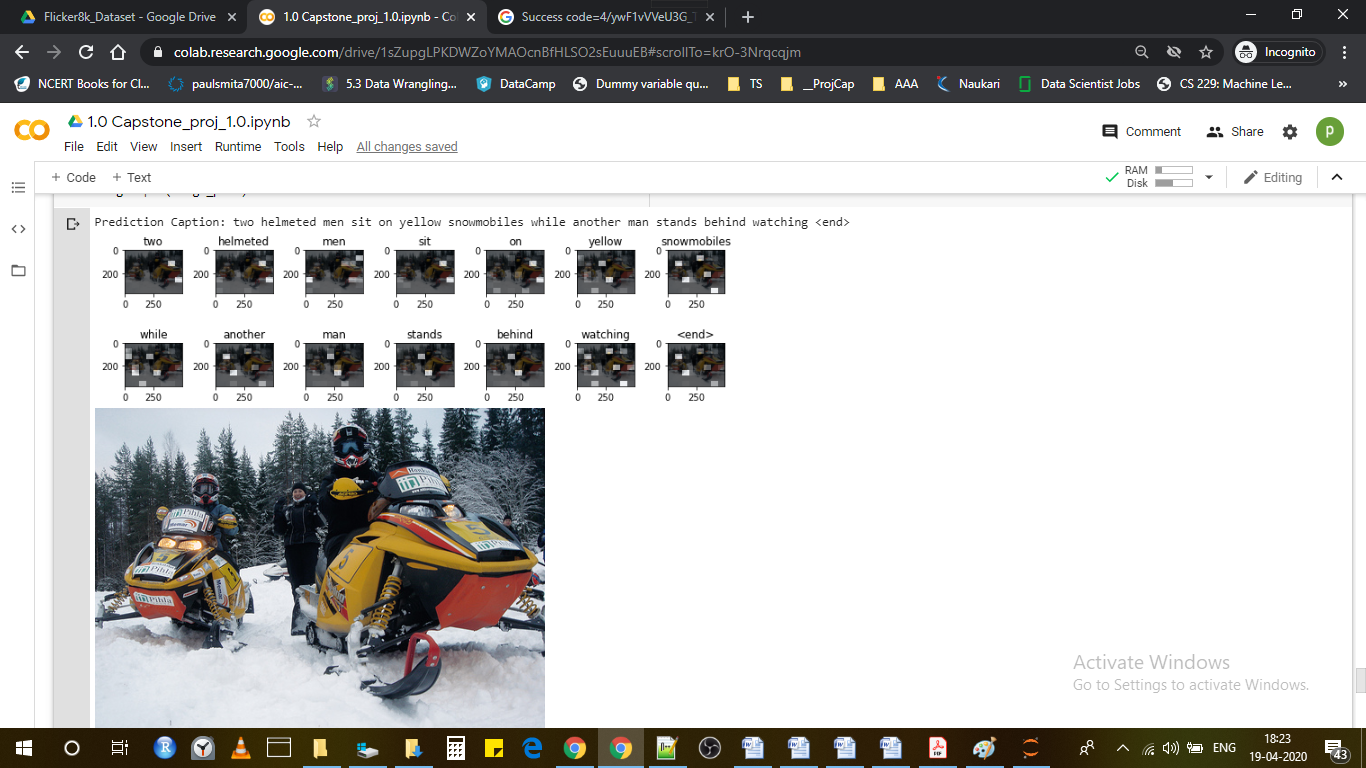


BELU score: 23.35496832484569

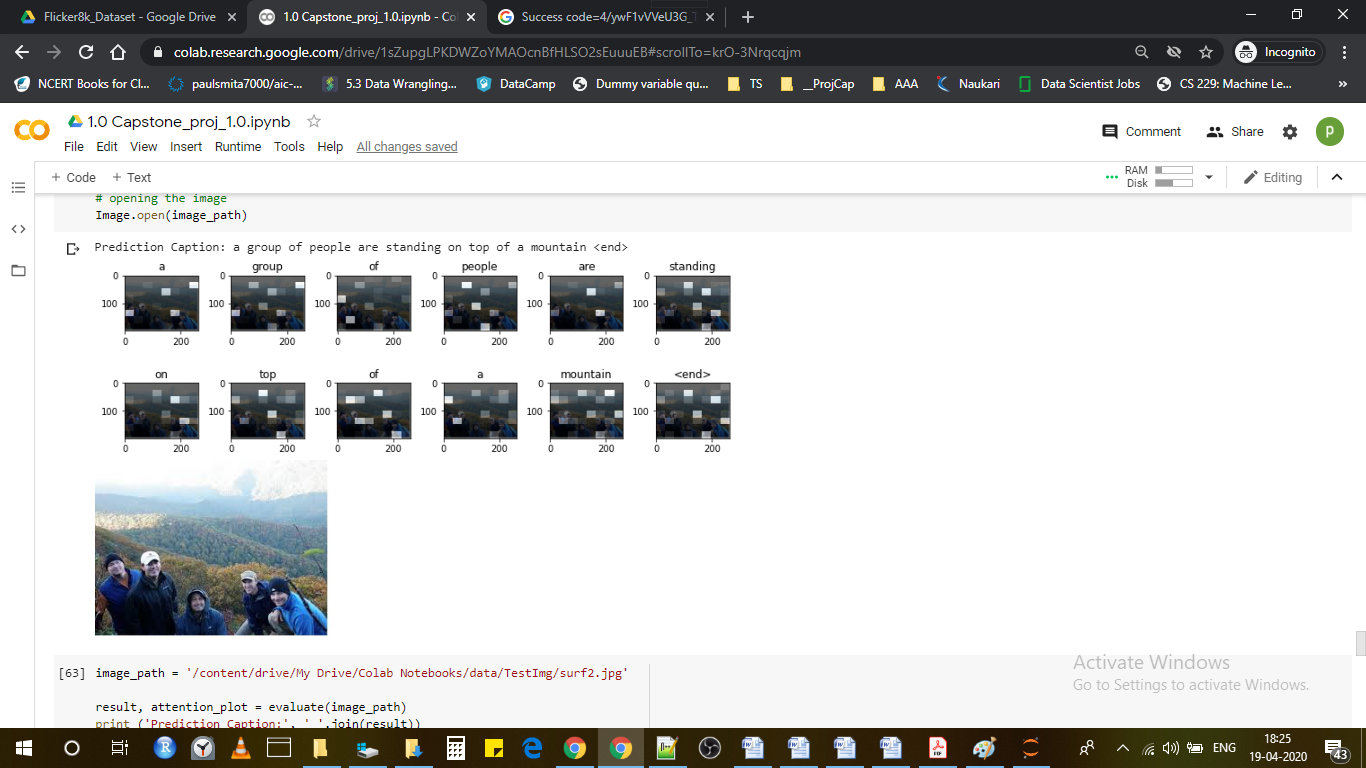
Real Caption: a boy wearing a red shirt and jeans is doing a flip on his bike

Prediction Caption: a person is show upside down on his bicycle over a lake

Unseen Image



Prediction Caption: two helmeted men sit on yellow snowmobiles while another man stands behind watching <end>



Prediction Caption: a group of people are standing on top of a mountain <end>

Reference

Image captioning with visual attention: <https://www.tensorflow.org/tutorials/text/image_captioning>

Attention Mechanism: <https://blog.floydhub.com/attention-mechanism/>

BLEU Score: <https://machinelearningmastery.com/calculate-bleu-score-for-text-python/>