

California State University East Bay

Fall Semester 2020

Deep Learning BAN 676-01

**Image Captioning using**

**Deep Learning Techniques**

by

**Sai Sruthi Bodapati (FS4443)**

**Dheeraj Talasila(YY8386)**

Supervised by

**Professor Surendra Sarnikar**

**Objective:**

The main objective of this project is to generate the captions by just taking the image as an input and producing relative captions.

**Introduction:**

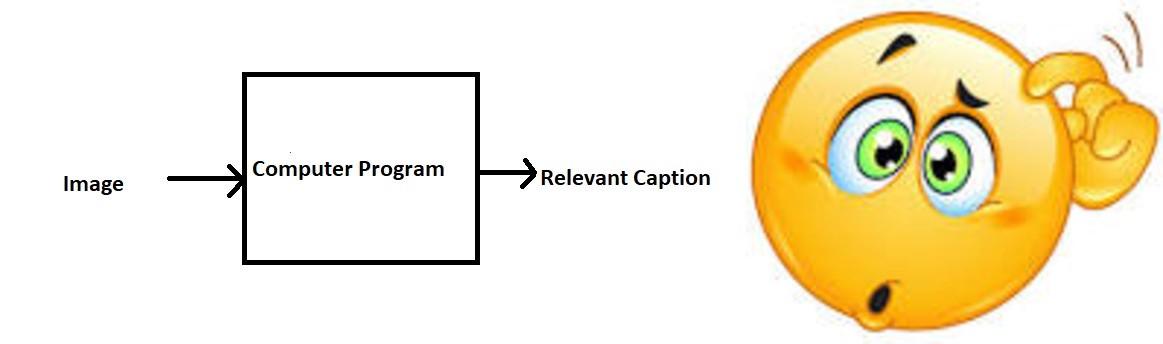


**What caption can you give by looking at this picture?**

* California state university name board
* California state university east bay sign board in the bushes
* White letter name board on red and black background.

All of these captions are certainly related to this picture, and some other captions may also exist. But the point we wanted to make is that as human beings, it's too easy for us to just take a look at an image and explain it in a suitable language.

But can we write a computer program that takes an image as input and produces a relevant caption as output? Yes, it can be done using Deep Learning techniques.



**How important is this problem in real life scenarios?**

Let’s see a few real-world situations where this can be implemented.

For Blind People — We can create a product for the blind which will guide them travelling on the roads without the support of anyone else. We can do this by first converting the scene into text and then the text to voice.

Image Captioning - It can help, make 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.

Self-driving Car - Automatic driving cars are the best example and if we can properly caption the scene around the car, it can give a boost to the self-driving system.

**Concepts Used:**

# Convolution Neural Networks

# Recurrent Neural Networks

# Transfer Learning

# Gradient Descent

# Text Processing

# Overfitting, Probability, Text Processing, Python syntax and data structures

# Bleu (Bilingual Evaluation Understudy) Score

**Tools Used:**

* Python 3.7
* Google Colab
* Python Libraries: numpy, pandas, matplotlib, nltk, itertools**,** keras

**Dataset:**

* There are many open-source datasets available for this problem, like Flickr 8k (containing 8k images), Flickr 30k (containing 30k images), MS COCO (containing 180k images), etc.
* But for the purpose of this project, we have used the Flickr 8k dataset provided by the University of Illinois at Urbana-Champaign. We used the small dataset because of the limited computational power.
* This dataset contains 8000 images each with 5 captions
* These images are bifurcated as follows:
  + Training Set — 6000 images
  + Validation Set — 1000 images
  + Test Set — 1000 images
* The dataset contains images along with some text files related to the images. We could see there are 5 captions for every image (captions numbered as jpg#0 – jpg#4).

A picture containing text

Description automatically generated

**Original Dataset for sample**

**Data Preprocessing:**

* Data preprocessing refers to the various transformations that are applied to clean the data before feeding it to the model
* Each image has 5 captions and we can see that #(0 to 5)number is assigned for each caption.
* We have preprocessed the data by mapping images with their respective captions using lambda expressions. We have done this mapping by creating separate dictionaries for train, validation and test sets.

**Creating Vocabulary:**

* Here, the captions are the ones we want to predict. So, during the training period, captions will be the target variables (Y) that the model is learning to predict.
* The prediction of the complete caption can not happen at once. It is done by predicting word by word.
* Thus, we need to encode each word into a fixed size vector.
  + Word to Index
  + Index to Word
* We will represent every unique word in the vocabulary by an integer (index).
* These two Python dictionaries can be used as follows:

words\_to\_indices[‘abc’] -> returns index of the word ‘abc’

indices\_to\_words[k] -> returns the word whose index is ‘k’

* The importance of creating these dictionaries is that machines can only understand numbers, so the input (captions split into words) is converted to indices and are fed to the model. Also, the machine gives a number as an output, but as humans we want the captions to be represented by words and not numbers, so we convert these outputted numbers (indices) back to their words and print the captions (in words).

**Extracting the feature vectors from all images:**

* Any input to a model must be given in the form of a vector..
* We have used transfer learning to extract the feature vectors from the images.
* We have used **Restnet-50.**
* RestNet-50 is a convolutional neural network that is 50 layers deep. We can load this pretrained version of the network trained on more than a million images from the ImageNet database. The ‘pretrained network’ can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.
* After extracting the feature vectors, we have saved them in .pkl files so that we can use them again and again.

# Model:

* The caption generator model we developed, relies on two main components, a CNN and an RNN.
* Convolutional Neural Network: These are designed to map image data to an output variable. They have proven so effective that they are the go-to method for any type of prediction problem involving image data as an input.
* Recurrent Neural Network: RNN and LSTMs have received the most success when working with sequences of words and paragraphs, generally called natural language processing. This includes both sequences of text and sequences of spoken language represented as a time series.
* We have used merge architecture to combine these two components.
* Since the input consists of two parts, an image vector and a partial caption, we have used the Functional API which allows us to create Merge Models.
* So, by merging the two, we can get a model that can identify patterns from images, and then use that information to help generate a description of those images.
* As mentioned before we used a pre-trained Convolutional Neural Network (CNN) to extract feature vectors from input images.

Diagram

Description automatically generated

**Model Architecture**

**Model-1:**

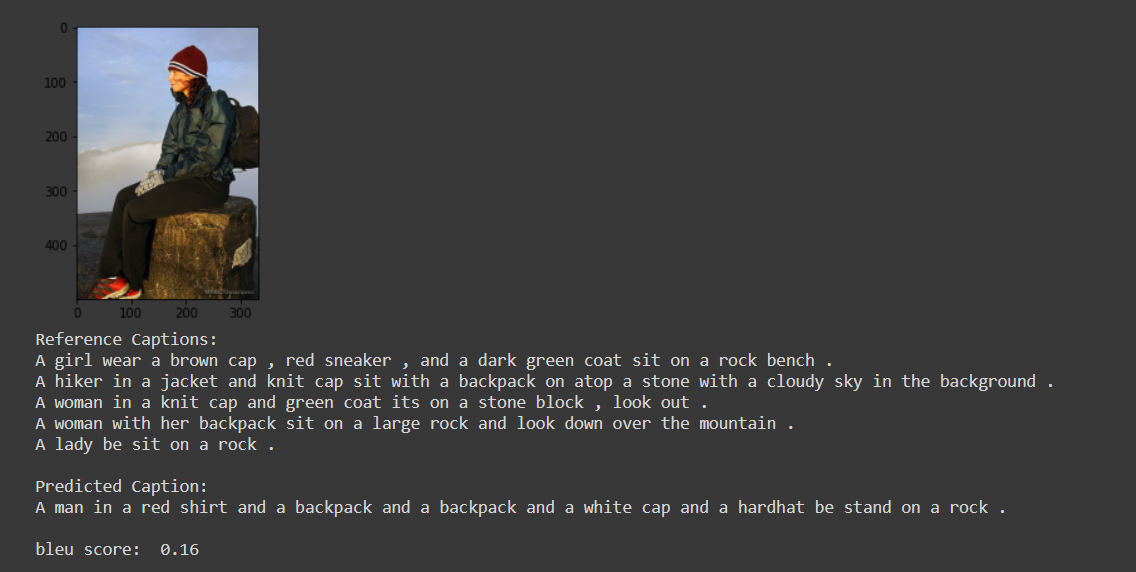
The below plot helps to visualize the structure of the network and better understand the two streams of input:

Diagram

Description automatically generated

* The LSTM (Long Short Term Memory) layer is nothing but a specialized Recurrent Neural Network to process the sequence input (partial captions in our case).
* We have provided feature vector data as input 1 for the model and partial captions as input 2.
* Input 1 is passed through a simple dense layer with a dropout layer(used as a solution for overfitting).
* Input 2 is passed through an Embedding layer with fixed length of 40 and then through LSTM.
* We merge both the branches and pass the outputs from these branches to dense layers with relu and softmax activations respectively.
* The output generated is a word (class) from a group of words (classes). It is similar to the multi-class classification problem.So, we have selected the activation function for the output layer as softmax.
* Finally, we compile the model using the Adam optimizer and categorical\_crossentropy as our loss function (multi-class classification).
* Summary, we have:
  + Input\_1 -> Partial Caption
  + Input\_2 -> Image feature vector
  + Output -> An appropriate word, next in the sequence of partial caption provided in the input\_1 (or in probability terms we say conditioned on image vector and the partial caption)
* The model was then trained for 50 epochs with 5 images per batch (batch size) using data generator function.

**Results:**





**Model-2:**

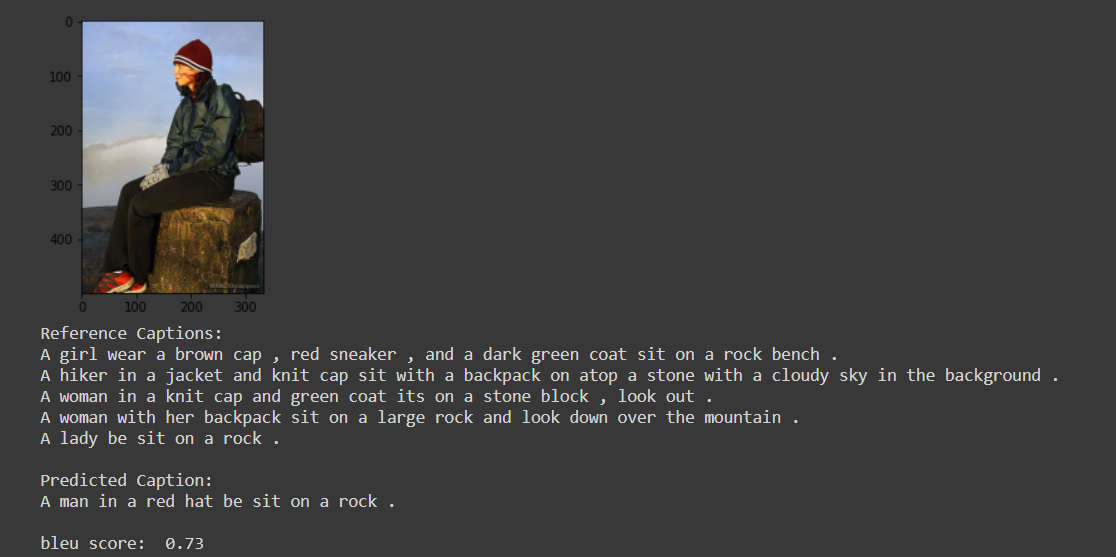
Diagram

Description automatically generated

* We have used the same architecture for the second model as well. This time we have used multiple LSTM layers (with return sequences =True).
* We trained this model with 50 epochs with 5 images per batch using the data generator.

* Model 2 has predicted better results compared to Model 1.

**Results:**



****

**Evaluation:**

We have used, BLEU score (**bilingual evaluation understudy)** as a metric to evaluate the captions predicted by the model. Scores are calculated for individual translated segments—generally sentences—by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole [corpus](https://en.wikipedia.org/wiki/Text_corpus) to reach an estimate of the translation's overall quality. Intelligibility or grammatical correctness are not taken into account. BLEU's output is always a number between 0 and 1. This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts.

We have achieved an average BLEU score of 0.43 and 0.71 with model 1 and model 2 respectively.

# Conclusion and Improvements:

## 

* We have developed an image caption generator by building a CNN-RNN model. We used a small dataset consisting of 8000 images. For better results, we need to train on datasets larger than 100,000 images which can produce better accuracy models.
* Using CNN and RNN we were able to effectively generate the captions for the testing images.
* Of course, this is just a first-cut solution and a lot of modifications can be made to improve this solution like:
  + Using a larger dataset.
  + Changing the model architecture, e.g., include an attention module.
  + Doing more hyper parameter tuning (learning rate, batch size, number of layers, number of units, dropout rate, batch normalization etc.).
  + Use the cross validation set to understand overfitting.