Image Caption Generator

Abstract:

Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a photograph. 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.

Description:

In our project we have used Flickr8k dataset to train and built an **LSTM (Long Short Term Memory)** model for generating captions.

Data set:

The Flickr8k dataset consist of 8091 images with 5 descriptions (tokens) each. These images are bifurcated as follows:

- Training Set — 6000 images
- Dev Set — 1000 images
- Test Set — 1000 images

I/P & O/P for training the model:

The input and output for training the model is as follows,

Input: Features of image, Padded captions.

Output: Corresponding next word to be predicted.

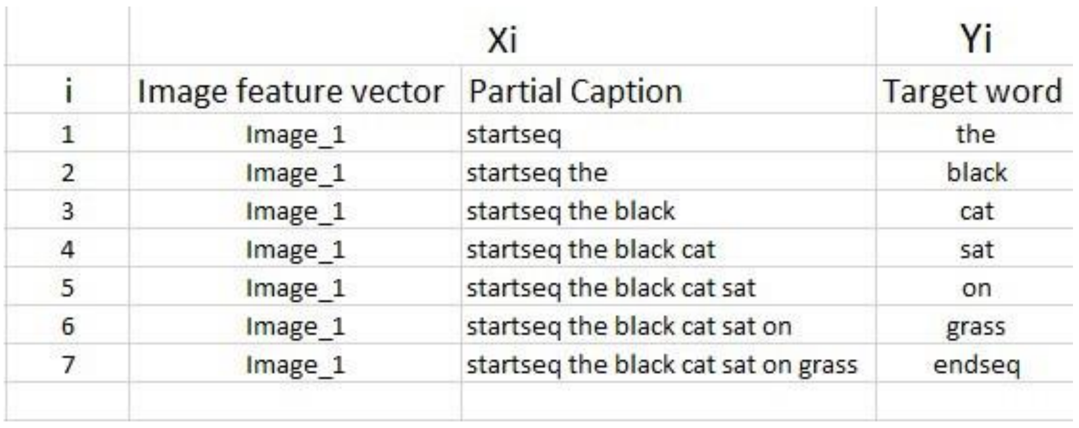
Steps:

1. **Preparing Image Data(Photo Feature Extractor):**

   In this step we have pre-computed the features of the photo, using the pre-trained model **VGG16** and saved them as a pickle file ('extracted_features.pkl'). So that we can then load these features later and feed them into our model as the interpretation of a given photo in the dataset.

So finally, the training data **image input** will be of shape **(None,4096)** from the features extracted from VGG16 model.

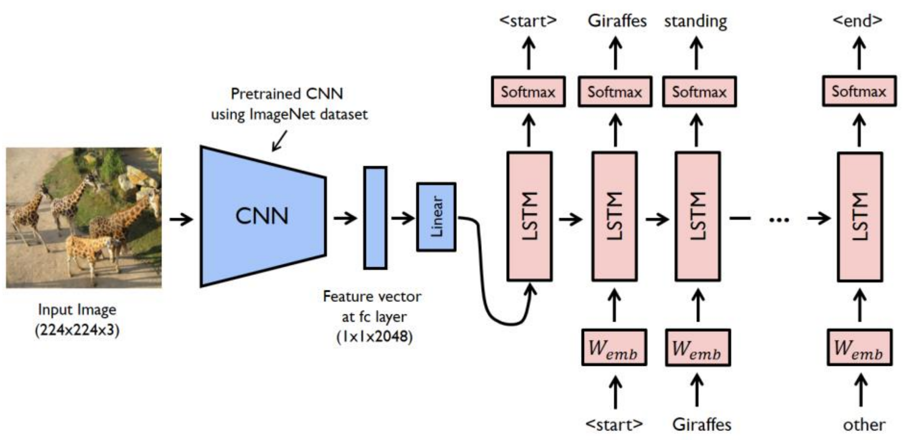
2. **Preparing Text Data**(**Sequence Processor**):
   a. In this step, From the (‘Flickr_token.txt‘) file, that has captions for images, we have cleaned the captions by converting all words to lowercase, removing all punctuation, remove all words that are one character or less in length (e.g. ‘a’) and remove all words with numbers in them and stored it in required form as (descriptions.txt) that will be later used for our training.
   b. We want a vocabulary that is both expressive and as small as possible. A smaller vocabulary will result in a smaller model that will train faster. Therefore, from the cleaned captions in (‘descriptions.txt‘) for training data file we have created a vocabulary of the corpus, with no of unique words = **7579**.
   c. Next we need to fit a tokenizer given caption descriptions, for this we have used ‘keras’ inbuilt Tokenizer functions.
   d. We will now encode the descriptions. Each description will be split into words. At first, the model will be provided one word and the photo as input to generate the next word. Then next, the first two words of the description will be provided to the model as input with the image to generate the next word. This is how the model will be trained.



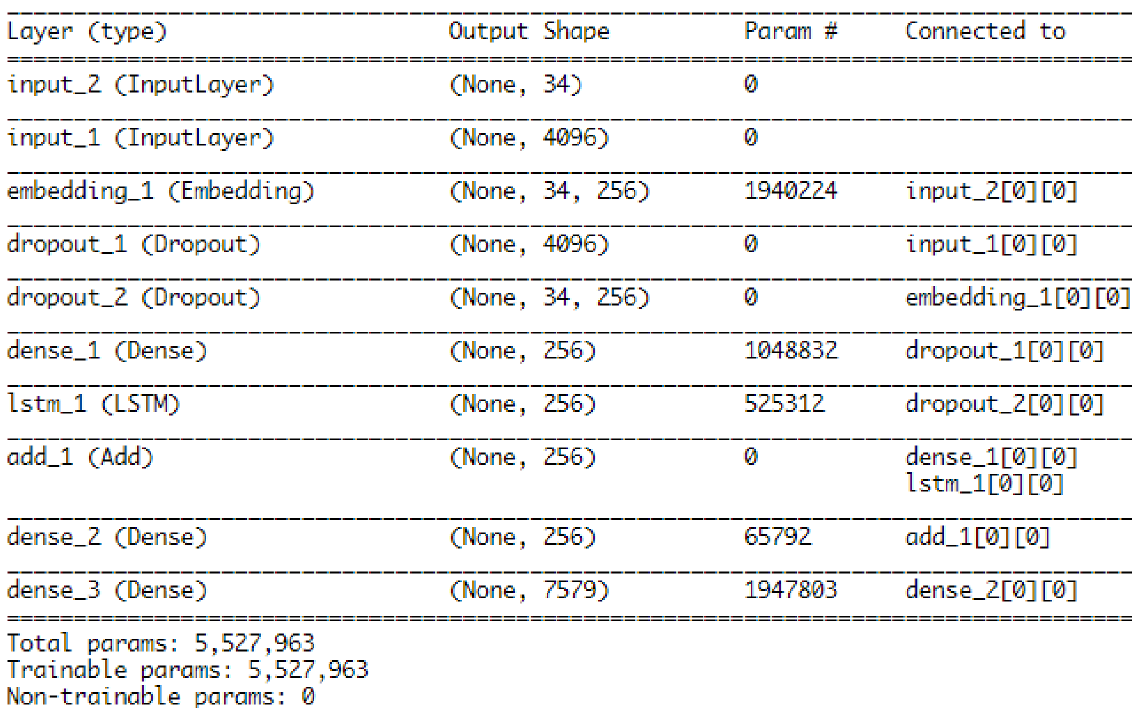
	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	startseq	the
2	Image_1	startseq the	black
3	Image_1	startseq the black	cat
4	Image_1	startseq the black cat	sat
5	Image_1	startseq the black cat sat	on
6	Image_1	startseq the black cat sat on	grass
7	Image_1	startseq the black cat sat on grass	endseq

e. This will be tokenized and the index will be given as input to the model.
f. So finally, the training data **caption input** will be of shape **(None,34)** where 34 is the maximum length of the description from training data.
g. And the size of the output array used to train will be **(None,7579)** where 7579 is the size of the training vocabulary (that is nothing but a vector of size [1,7579] of ones and zeros, 1 –represent the included output word)

3. ***Building Model (LSTM) for caption generation:*** Baseline model for generating captions.



We have defined the model to be as follows,

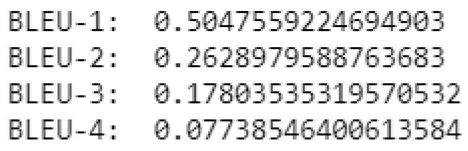


```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_2 (InputLayer)             (None, 34)            0
____________________________________________________________________________________________________
input_1 (InputLayer)             (None, 4096)          0
____________________________________________________________________________________________________
embedding_1 (Embedding)          (None, 34, 256)       1940224     input_2[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 4096)          0           input_1[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 34, 256)       0           embedding_1[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 256)           1048832     dropout_1[0][0]
____________________________________________________________________________________________________
lstm_1 (LSTM)                    (None, 256)           525312      dropout_2[0][0]
____________________________________________________________________________________________________
add_1 (Add)                      (None, 256)           0           dense_1[0][0]
                                                                   lstm_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 256)           65792       add_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 7579)          1947803     dense_2[0][0]
====================================================================================================
Total params: 5,527,963
Trainable params: 5,527,963
Non-trainable params: 0
```

The model is trained and the output is stored as ('model.h5')

4. ***Evaluating the model:*** *We have* evaluated the performance (skill) of the model using BLEU score.

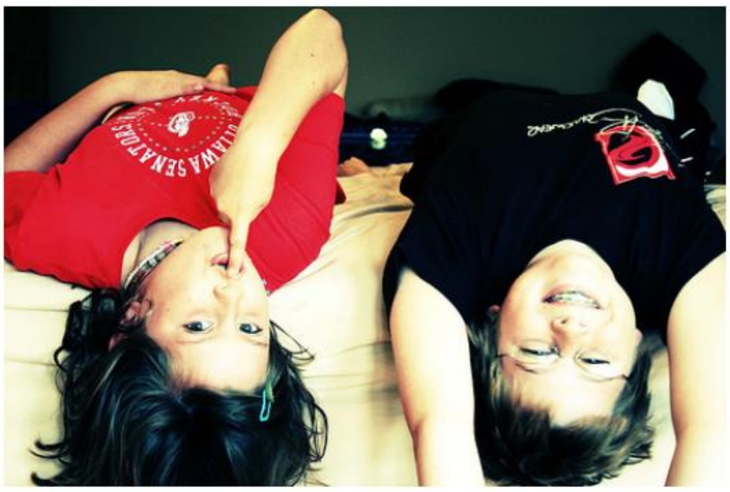
   Results:



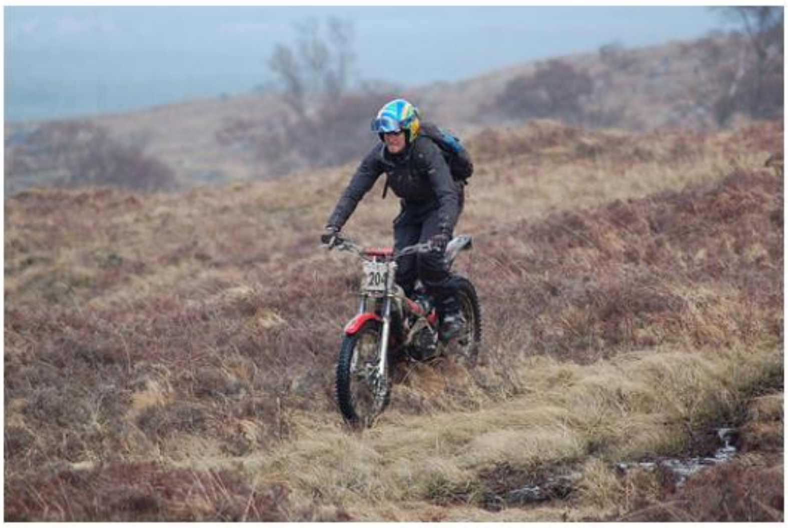
```
BLEU-1:  0.5047559224694903
BLEU-2:  0.2628979588763683
BLEU-3:  0.17803535319570532
BLEU-4:  0.07738546400613584
```

5. ***Generating Caption:*** Few of our models prediction is shown below.

```
startseq two children are playing on the grass endseq
```



```
startseq man in red shirt is riding bike on dirt bike endseq
```



Improvement:

Word Embedding Models:

word2vec Embedding.

Problems Faced:

- Training keras model from data that don’t fit in memory.
  - → Therefore we tried to train the model from batches of .npy input files.
  - → We tried progressive loading approach, by Training on batch using generators, which will provide data to the model given directory.

Future Exploration:

Network Size:

Next we can see how the variations in the network structure will impact model skill, and choose parameters that can train the model better.

- Size of the sequence encoder model →

  We can reduce the number of memory units in the LSTM layer from 256 to 128 and see whether that will impact the model.
- Size of the language model.

  We can add another LSTM layer of the same size and look at the impact of doubling the capacity of the language model.

Configuring the Feature Extraction Model:

The use of the pre-trained **VGG16** model provides some additional points of configuration.

→We can try using a global average pooling layer after the VGG model.