A neural network approach to image description generation

A Master's Project Report submitted to Santa Clara University in Fulfillment of the Requirements for the

Course COEN - 296: Natural Language Processing

Instructor: Ming-Hwa Wang
Department of Computer Science and Engineering

By
Jayant Kashyap
Prakhar Maheshwari
Sparsh Garg

Winter Quarter 2018

Acknowledgement

We would like to extend our gratitude towards Prof. Ming-Hwa Wang, who inspired us to make this project. We would also like to thank Santa Clara University which provided us the opportunity to learn the subject and apply its concept into making this project.

Table of Contents

Abstract

In this project we apply concepts of Natural Language Processing(NLP) and Deep Learning(DL) for the generation of the image description. The project present a generative model based on a deep recurrent architecture that combines recent advances in computer vision and machine translation and that can be used to generate natural sentences describing an image.Here we use a neural and probabilistic framework to generate descriptions from images.The model is to involve attention mechanism which would increase the efficiency of the system. The model is trained and evaluated on Flickr8K and Flickr30K and the results are captured and analysis are done.

Objective

To design a neural network model to create the image description.

What is the problem

A image contains a lots of information regarding the elements in the image and the surroundings of the elements. A description must capture not only the objects contained in an image, but it also express how multiple objects in image are related to each other as well as their attributes and the activities they are involved in. A language model is needed in addition to visual understanding. The neural network model for generation of image description is concerned with the semantic knowledge in the image that has to be expressed in a natural language like English.

Why is this project related to this class

Language model such as recurrent neural network is one of the fundamental Natural Language Processing(NLP) which has application in various fields of science. Traditionally, image generation using recurrent neural network involve various aspects of NLP such as semantic analysis.

Why other approaches are no good

Some pioneering approaches that address the challenge of generating image description have been developed, rely on hard-coded visual concepts and sentence templates, which imposes limits on their variety. The focus of these projects has been on reducing the complex visual scenes into a single sentence, which is an unnecessary restriction.

Why you think your approach is better

In this work we combine Deep CNN for image classification which is state-of-the art for this purpose with RNN for sequence modeling, to create single network that generates descriptions of images. The RNN is trained in the context of single “end-to-end” network. The model is inspired by recent success in sequence generation in machine translation, with difference that instead of starting with a sentence, we provide image processed by convolutional network.

Statement of the problem

Build a multimodal recurrent neural network architecture that takes an input image and generates its description in text. The model was built that inferes the latent alignment between segments of sentences and the region of the image. We will train the model with Flickr8K and Flickr30K datasets which contains set of images and their corresponding sentence description and evaluate its performance on a new dataset.

Hypothesis/Goals

We are proposing through this project, to build a architecture which uses multimodal Recurrent neural network that generates descriptions of the visual data.

Positive Hypothesis

The model will generate the correct natural language description in English of the semantic knowledge given in image.

Negative Hypothesis

The model will generate the natural language description that is not relevant to the semantic knowledge and labels of the image.

Methodology

Data Sets

In this work we will try to experiment our model on multiple data set such as, Flickr8K, Flickr30K datasets and see how our model responds to the different data sets.

All these dataset either provide training sets, validation sets and test sets separately or just have a sets of images ,and description. In both the case we will divide the data sets into three parts as training, validation and test set to train and evaluate the model for accuracy.

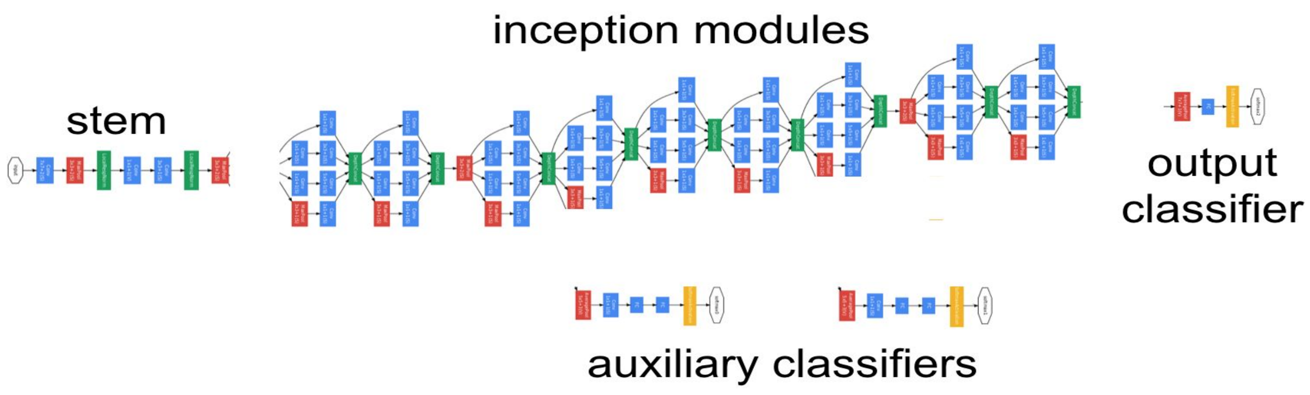
How To Solve the problem

Algorithm Design

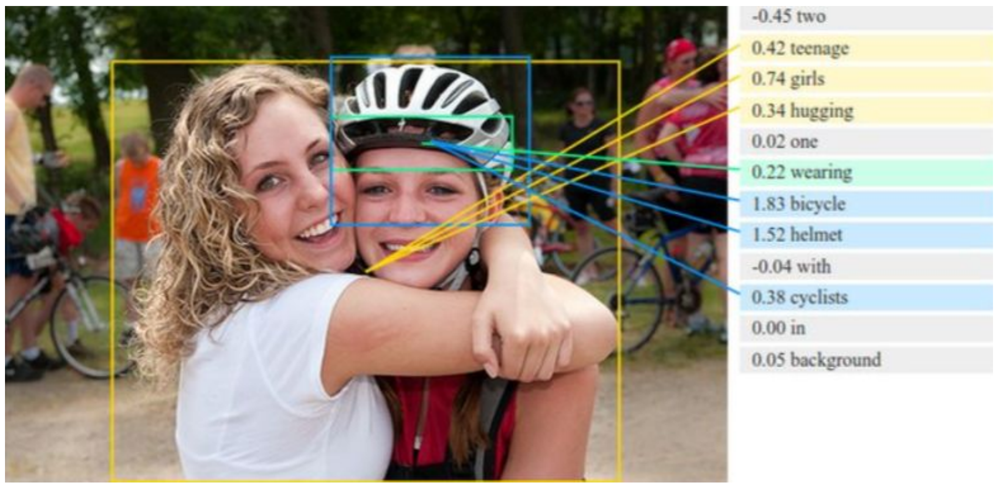
- We would like to create a model that takes an image I as input, and is trained to maximize the likelihood p(S|I) of producing a target sequence of words S = {S1, S2, . . .} where each word St comes from a given dictionary, that describes the image adequately.

- The initial step would be to store all the images in an array of shape [m, n_h, n_w, n_c] where
  m = # training examples
  n_h = height of an image
  n_w = width of an image
  n_c = # of channels (typically = 3 (RGB))

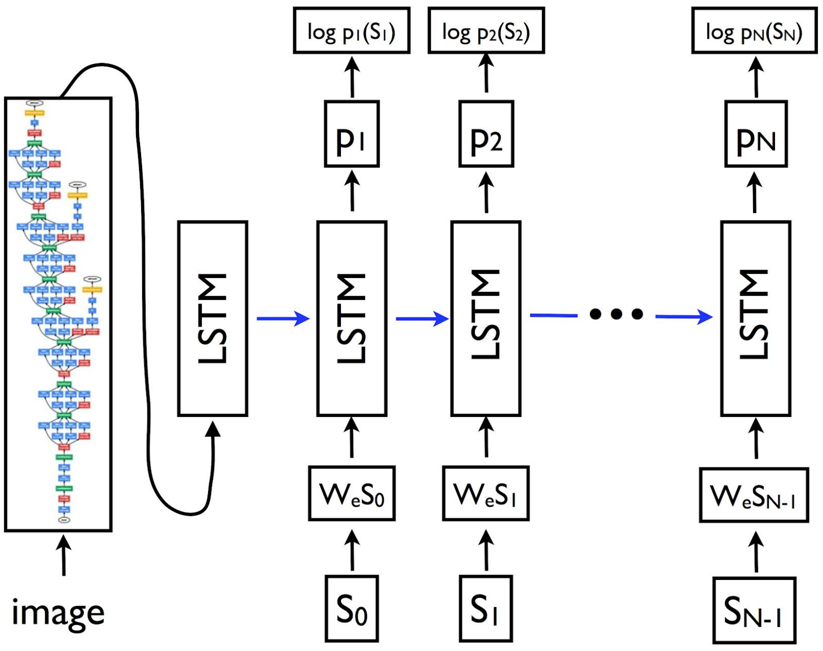
- The next step will be to pass all the images to Inception V3 (GoogleNet) to predict the probabilities of all the elements in an image



- The output of the above image will be the probabilities of the elements in an image like below, which will pass to LSTM model for further processing.



- After getting the output from Inception CNN, then it needs to create meaningful sentences which would be done by implementing LSTM based sentence generator.



- The LSTM model is trained to predict each word of the sentence after it has seen the image as well as all preceding words as defined by p(St|I, S0, . . . , St−1)

- We are expecting the output of above image will be
    a. Two teenage girls are hugging . (low chance)
    b. one of the teenage girls wearing helmet are hugging(med chance)
    c. Cyclists wearing helmet with their bicycle are hugging . (high chance)

Languages used

In this project we plan to use Python3 as it is one of the fastest scripting language and includes a lot of libraries for Deep Learning and Natural Language Processing

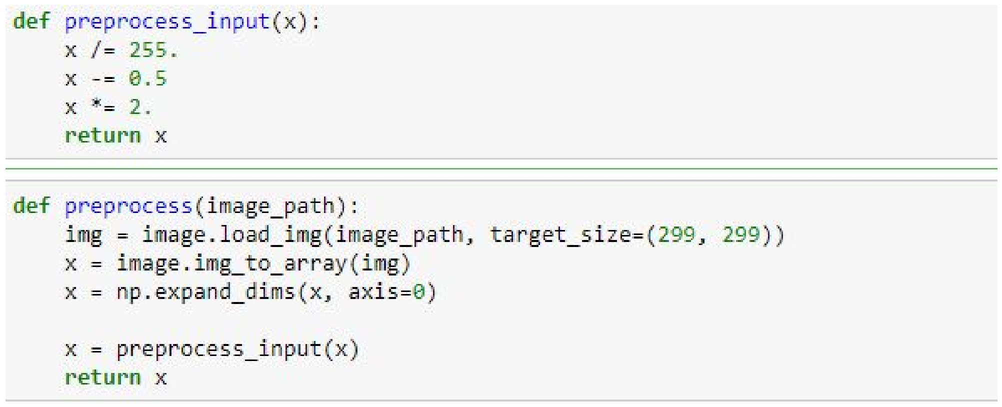
Tools Used

- Pickle
- TensorFlow 1.0 or greater
- NumPy
- Natural Language Toolkit (NLTK):
    1. First install NLTK
    2. Then install the NLTK data package "punkt"
- Glob
- tqdm
- Keras

Implementation

1. Code:

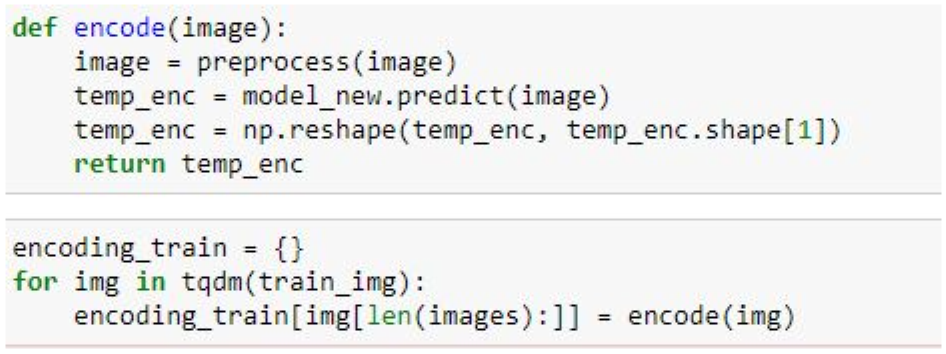
Pre-processing:



```
def preprocess_input(x):
    x /= 255.
    x -= 0.5
    x *= 2.
    return x
```

```
def preprocess(image_path):
    img = image.load_img(image_path, target_size=(299, 299))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    x = preprocess_input(x)
    return x
```

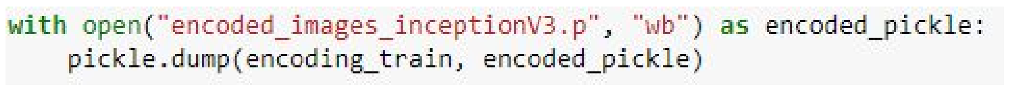
Image Encoding:



```
def encode(image):
    image = preprocess(image)
    temp_enc = model_new.predict(image)
    temp_enc = np.reshape(temp_enc, temp_enc.shape[1])
    return temp_enc
```

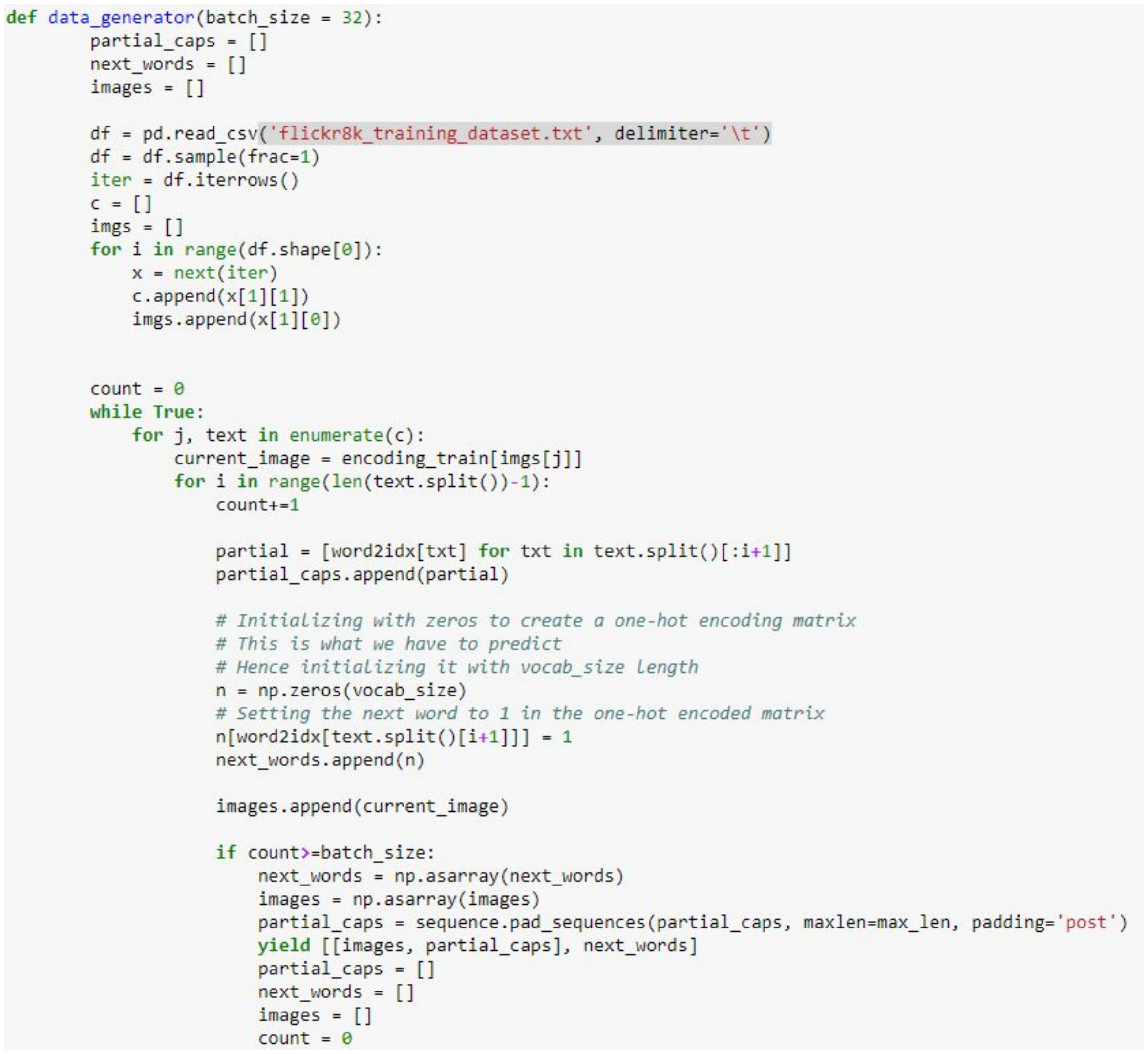
```
encoding_train = {}
for img in tqdm(train_img):
    encoding_train[img[len(images):]] = encode(img)
```

Saved encoding to pickle file:



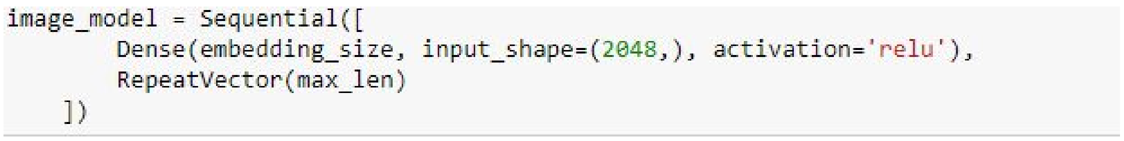
```
with open("encoded_images_inceptionV3.p", "wb") as encoded_pickle:
    pickle.dump(encoding_train, encoded_pickle)
```

Training Batch Generator:



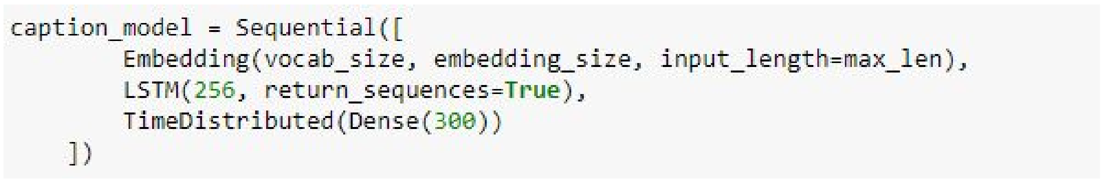
```
def data_generator(batch_size = 32):
        partial_caps = []
        next_words = []
        images = []

        df = pd.read_csv('flickr8k_training_dataset.txt', delimiter='\t')
        df = df.sample(frac=1)
        iter = df.iterrows()
        c = []
        imgs = []
        for i in range(df.shape[0]):
            x = next(iter)
            c.append(x[1][1])
            imgs.append(x[1][0])


        count = 0
        while True:
            for j, text in enumerate(c):
                current_image = encoding_train[imgs[j]]
                for i in range(len(text.split())-1):
                    count+=1

                    partial = [word2idx[txt] for txt in text.split()[:i+1]]
                    partial_caps.append(partial)

                    # Initializing with zeros to create a one-hot encoding matrix
                    # This is what we have to predict
                    # Hence initializing it with vocab_size length
                    n = np.zeros(vocab_size)
                    # Setting the next word to 1 in the one-hot encoded matrix
                    n[word2idx[text.split()[i+1]]] = 1
                    next_words.append(n)

                    images.append(current_image)

                    if count>=batch_size:
                        next_words = np.asarray(next_words)
                        images = np.asarray(images)
                        partial_caps = sequence.pad_sequences(partial_caps, maxlen=max_len, padding='post')
                        yield [[images, partial_caps], next_words]
                        partial_caps = []
                        next_words = []
                        images = []
                        count = 0
```

Image Model:



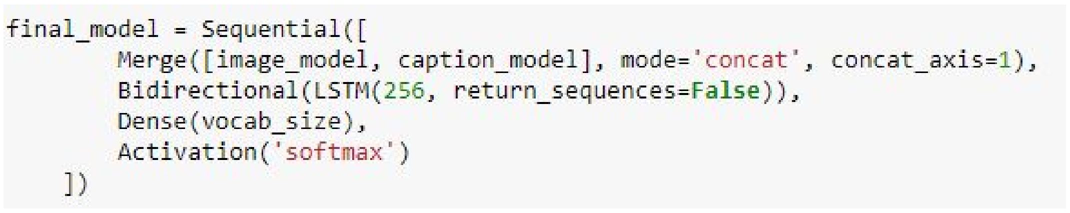
```
image_model = Sequential([
        Dense(embedding_size, input_shape=(2048,), activation='relu'),
        RepeatVector(max_len)
    ])
```

Caption Model:



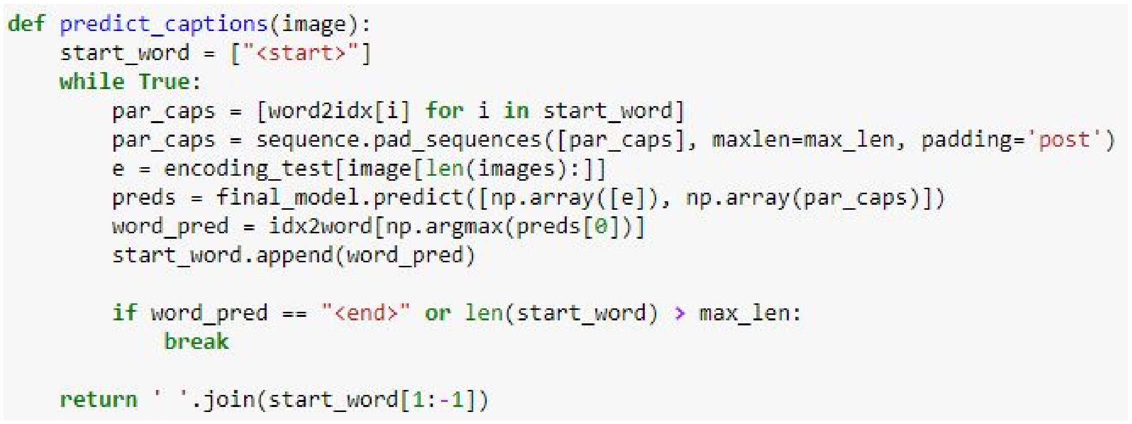
```
caption_model = Sequential([
        Embedding(vocab_size, embedding_size, input_length=max_len),
        LSTM(256, return_sequences=True),
        TimeDistributed(Dense(300))
    ])
```

Final Model:



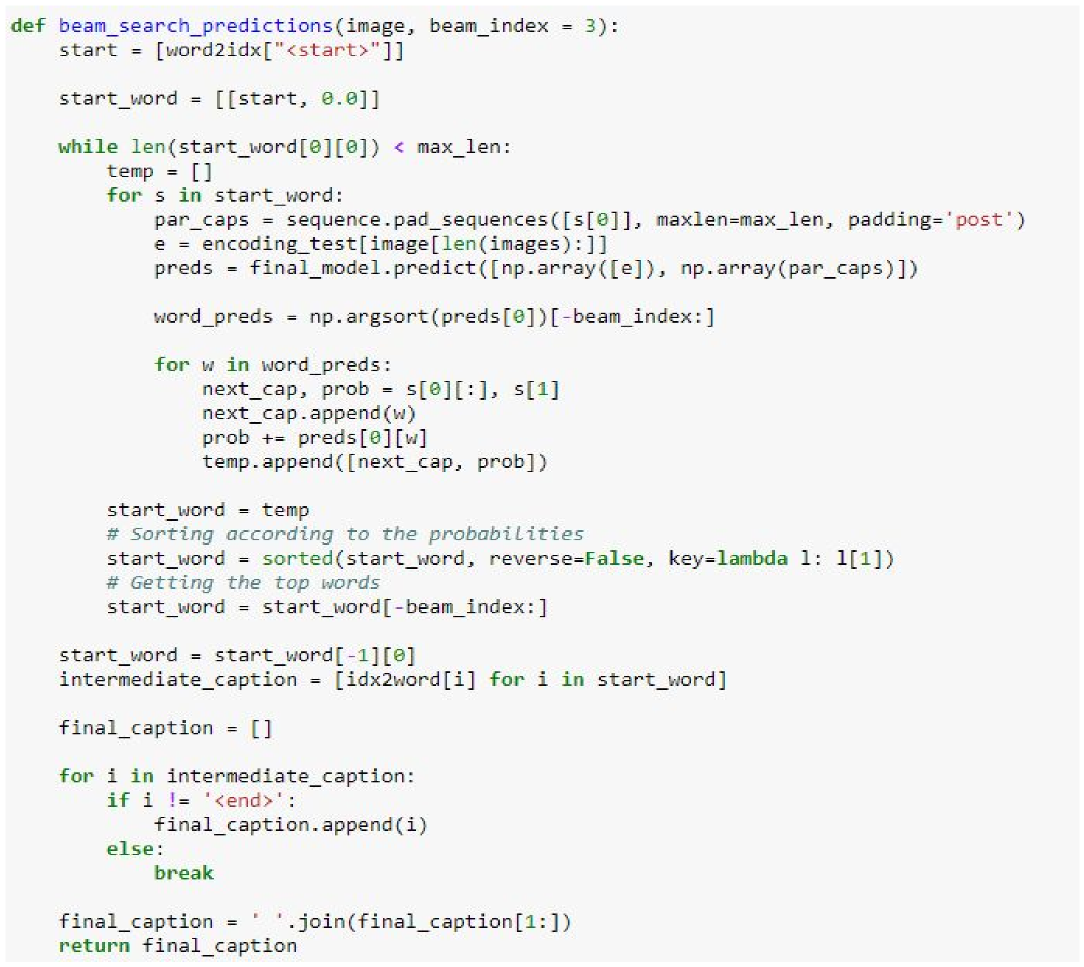
```
final_model = Sequential([
        Merge([image_model, caption_model], mode='concat', concat_axis=1),
        Bidirectional(LSTM(256, return_sequences=False)),
        Dense(vocab_size),
        Activation('softmax')
    ])
```

Sampling Caption Prediction:



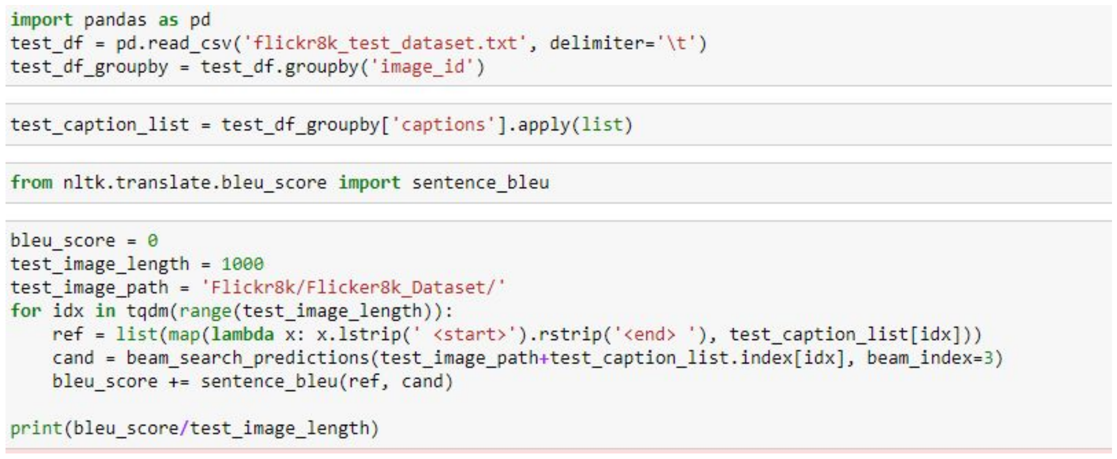
```
def predict_captions(image):
    start_word = ["<start>"]
    while True:
        par_caps = [word2idx[i] for i in start_word]
        par_caps = sequence.pad_sequences([par_caps], maxlen=max_len, padding='post')
        e = encoding_test[image[len(images):]]
        preds = final_model.predict([np.array([e]), np.array(par_caps)])
        word_pred = idx2word[np.argmax(preds[0])]
        start_word.append(word_pred)

        if word_pred == "<end>" or len(start_word) > max_len:
            break

    return ' '.join(start_word[1:-1])
```

Beam Search Caption Prediction:



```
def beam_search_predictions(image, beam_index = 3):
    start = [word2idx["<start>"]]

    start_word = [[start, 0.0]]

    while len(start_word[0][0]) < max_len:
        temp = []
        for s in start_word:
            par_caps = sequence.pad_sequences([s[0]], maxlen=max_len, padding='post')
            e = encoding_test[image[len(images):]]
            preds = final_model.predict([np.array([e]), np.array(par_caps)])

            word_preds = np.argsort(preds[0])[-beam_index:]

            for w in word_preds:
                next_cap, prob = s[0][:], s[1]
                next_cap.append(w)
                prob += preds[0][w]
                temp.append([next_cap, prob])

        start_word = temp
        # Sorting according to the probabilities
        start_word = sorted(start_word, reverse=False, key=lambda l: l[1])
        # Getting the top words
        start_word = start_word[-beam_index:]

    start_word = start_word[-1][0]
    intermediate_caption = [idx2word[i] for i in start_word]

    final_caption = []

    for i in intermediate_caption:
        if i != '<end>':
            final_caption.append(i)
        else:
            break

    final_caption = ' '.join(final_caption[1:])
    return final_caption
```

BLEU Score:



```
import pandas as pd
test_df = pd.read_csv('flickr8k_test_dataset.txt', delimiter='\t')
test_df_groupby = test_df.groupby('image_id')
```

```
test_caption_list = test_df_groupby['captions'].apply(list)
```

```
from nltk.translate.bleu_score import sentence_bleu
```

```
bleu_score = 0
test_image_length = 1000
test_image_path = 'Flickr8k/Flicker8k_Dataset/'
for idx in tqdm(range(test_image_length)):
    ref = list(map(lambda x: x.lstrip(' <start>').rstrip('<end> '), test_caption_list[idx]))
    cand = beam_search_predictions(test_image_path+test_caption_list.index[idx], beam_index=3)
    bleu_score += sentence_bleu(ref, cand)

print(bleu_score/test_image_length)
```

2. Code Implementation:

The following are the steps which we followed in the project for achieving the results:

a. The raw data has been obtained from “flickr 8k” dataset provided by University of Illinois at Urbana Champaign and the same has been used throughout the model to generate the description of the images.

b. Image Encoding has been done based on the model “Inception v3”

c. The output of the image classification and encoding has been feeded into the RNN model which is a LSTM network. Long short-term memory (LSTM) units (or blocks) are a building unit for layers of a recurrent neural network (RNN).

d. Model has been trained using the text embedding.

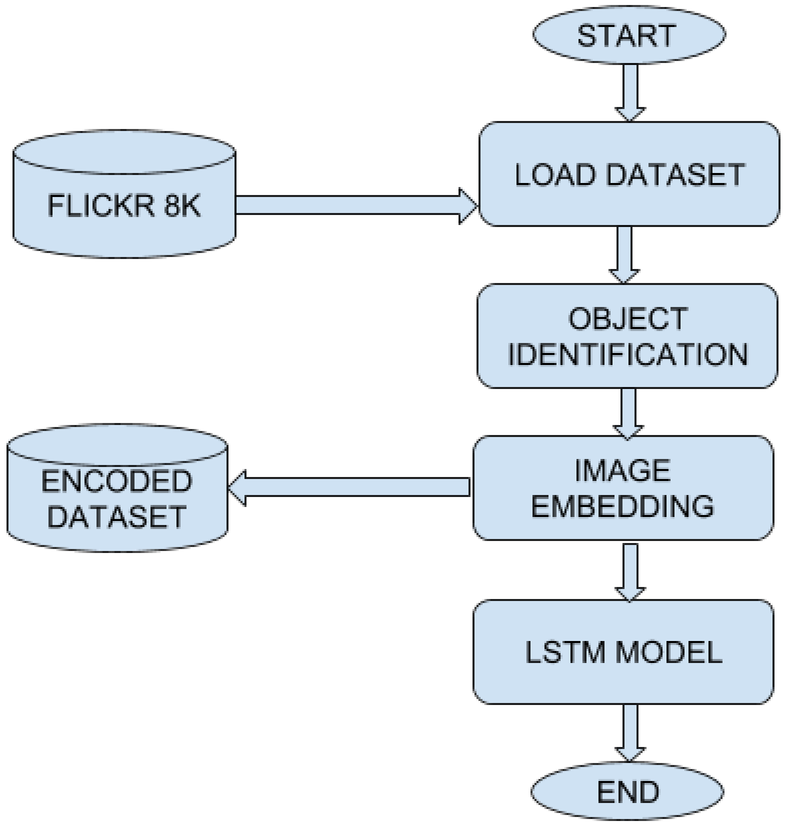
e. After the training of the model, the model has been evaluated and model tuning has been performed.

f. After the model tuning, prediction will be done using the beam search i.e. exploring the graph by expanding the most promising node in a limited set.

g. The image description will be generated in the English Language.

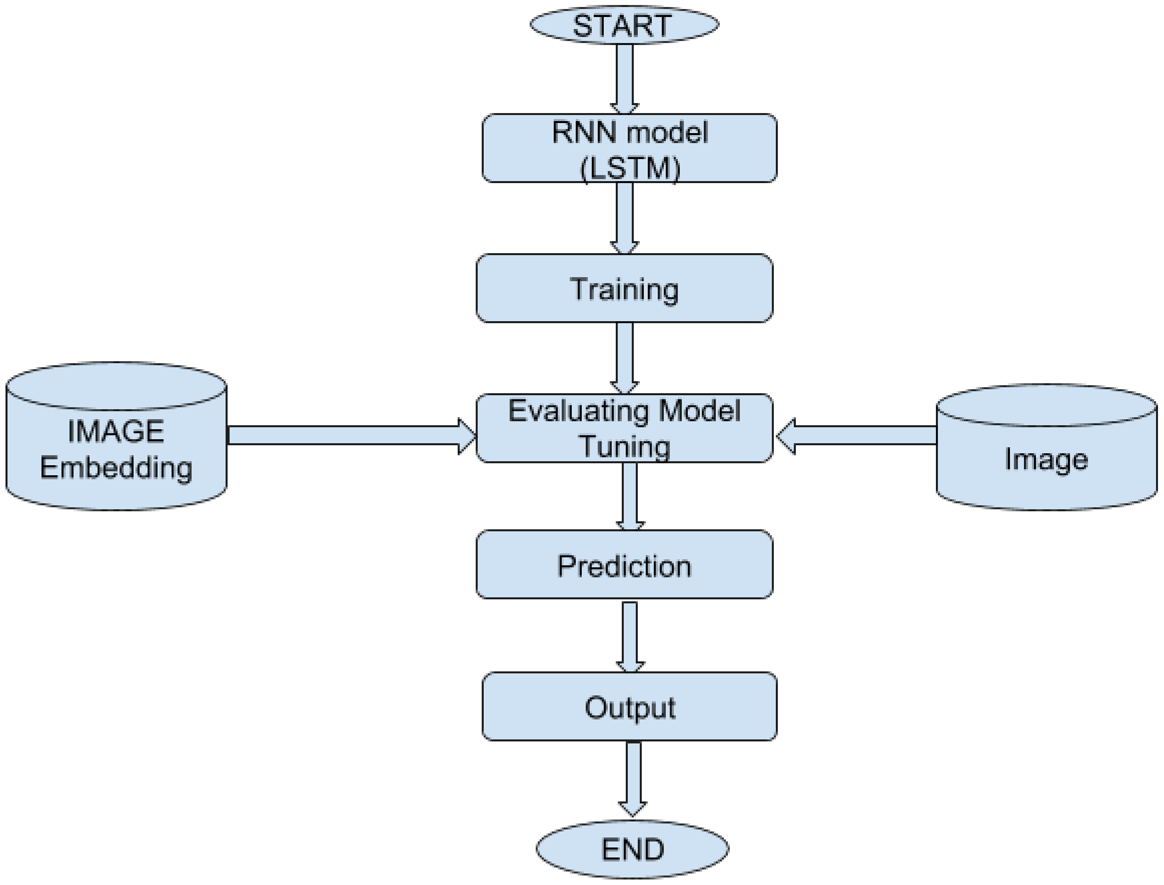
Flow Charts

Loading Data and Classification



Loading Data and providing classification of the Images to LSTM model

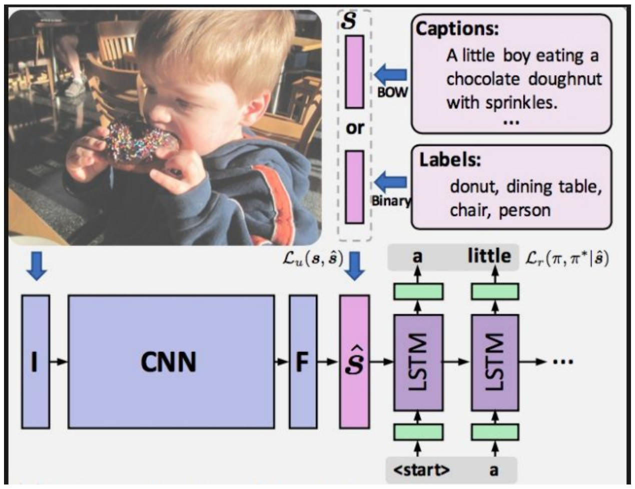
Model training and generating Output



Generating description by feeding the Model with Image Classification

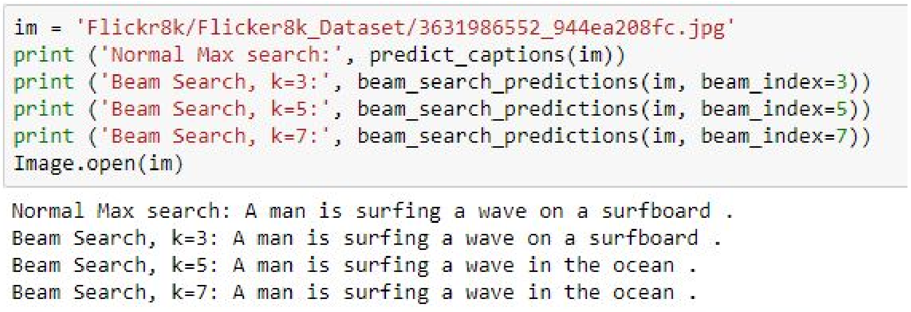
Data Analysis and Discussion

Output Generation



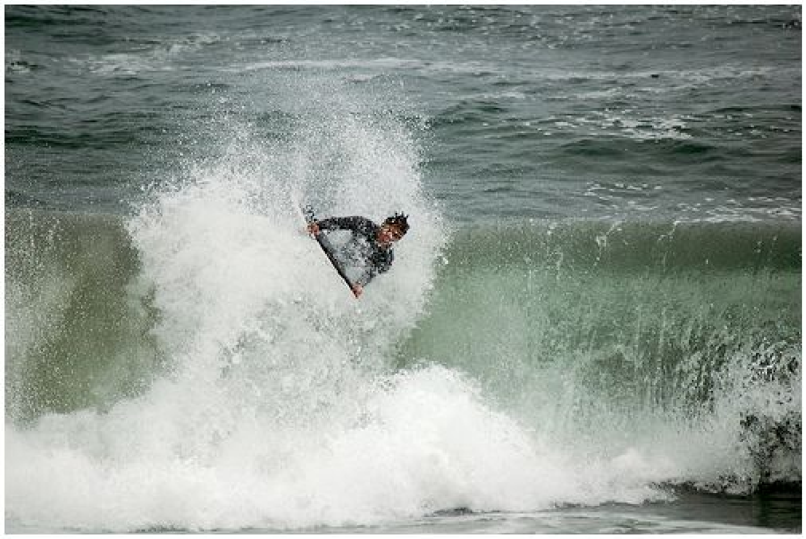
S
Captions:
A little boy eating a chocolate doughnut with sprinkles.
...
BOW
or
Labels:
donut, dining table, chair, person
Binary
$\mathcal{L}_u(s, \hat{s})$
a
little
$\mathcal{L}_r(\pi, \pi^*|\hat{s})$
I
CNN
F
$\hat{S}$
LSTM
LSTM
...
<start>
a

Output generated would be like:

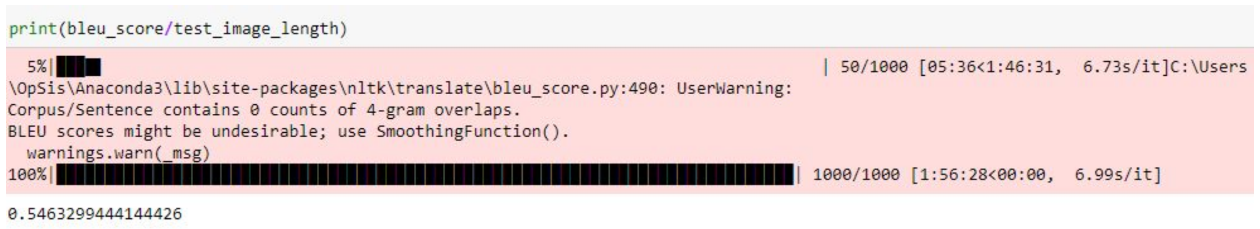


```
im = 'Flickr8k/Flicker8k_Dataset/3631986552_944ea208fc.jpg'
print ('Normal Max search:', predict_captions(im))
print ('Beam Search, k=3:', beam_search_predictions(im, beam_index=3))
print ('Beam Search, k=5:', beam_search_predictions(im, beam_index=5))
print ('Beam Search, k=7:', beam_search_predictions(im, beam_index=7))
Image.open(im)
```

```
Normal Max search: A man is surfing a wave on a surfboard .
Beam Search, k=3: A man is surfing a wave on a surfboard .
Beam Search, k=5: A man is surfing a wave in the ocean .
Beam Search, k=7: A man is surfing a wave in the ocean .
```



Output Analysis



```
print(bleu_score/test_image_length)

  5%|███                                                    | 50/1000 [05:36<1:46:31,  6.73s/it]C:\Users
\OpSis\Anaconda3\lib\site-packages\nltk\translate\bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 4-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
100%|██████████████████████████████████████████████████████████| 1000/1000 [1:56:28<00:00,  6.99s/it]

0.5463299444144426
```

a. The output generated is a Natural Language textual sequence.
b. With our model we scored a BLEU 4-Gram Score of 0.54 for Beam Search (beam index = 3) and same for normal sampling search

Compare Output against Hypothesis

Our hypothesis based on generating the image description in plain English text and is accurate. We have promising results that have good accuracy.

Summary and Conclusion

We have presented an end-to-end neural network system that can view an image and generate a reasonable description in plain English. Project is based on a convolution neural network that encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence. The model is trained to maximize the likelihood of the sentence given the image. Experiments on several datasets show the robustness of proposed model in terms of qualitative results and quantitative evaluations,It is clear from these experiments that, as the size of the available datasets for image description increases, so will the performance of approaches.

Recommendation for further studies

We can enhance and improve the description generation by the model by taking into consideration how one can use data, both from images alone and text alone which is loosely labelled or not classified. It will be interesting to see how one can use the unsupervised data to create the description of the image.

Bibliography

1. Karpathy, A. and Fei-Fei, L., 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3128-3137).

2. Vinyals, O., Toshev, A., Bengio, S. and Erhan, D., 2015, June. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on* (pp. 3156-3164). IEEE.

3. You, Q., Jin, H., Wang, Z., Fang, C. and Luo, J., 2016. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4651-4659).

4. Pan, J.Y., Yang, H.J., Duygulu, P. and Faloutsos, C., 2004, June. Automatic image captioning. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on* (Vol. 3, pp. 1987-1990). IEEE.

5. Meng C, Wang Y, Zhang S. Image-Question-Linguistic Co-Attention for Visual Question Answering

6. Wang, C., Yang, H., Bartz, C. and Meinel, C., 2016, October. Image captioning with deep bidirectional LSTMs. In *Proceedings of the 2016 ACM on Multimedia Conference* (pp. 988-997). ACM.