Image Captioning Using a CNN-LSTM System

Bharath Charan	Surya Teja	Sriman Reddy
IMT2018015	IMT2018080	IMT2018525

Unique Group Code: BSS

Problem Statement: Design a CNN-LSTM system that can perform image captioning under following conditions:

The total model size should not exceed 100 MB. Considering that each parameter weight is a float value (4 bytes), your CNN-LSTM model should have approximately 25 million parameters or less.

1. Introduction: Automatically describing the content of images using natural languages is a fundamental and challenging task. It has great potential impact. For example, it could help visually impaired people better understand the content of images on the web. Also, it could provide more accurate and compact information of images/videos in scenarios such as image sharing in social network or video surveillance systems. This project accomplishes this task using deep neural networks. By learning knowledge from image and caption pairs, the method can generate image captions that are usually semantically descriptive and grammatically correct.

In this project, we systematically analyze a deep neural networks-based image caption generation method. With an image as the input, the method can output an English sentence describing the content in the image. We analyze three components of the method: convolutional neural network (CNN), recurrent neural network (RNN) and sentence

generation. Instead of requiring complex data preparation or a pipeline of specifically designed models, a single end-to-end model can be defined to predict a caption, given a photo. In order to evaluate our model, we measure its performance on the Flickr8K dataset using the BLEU standard metric.

2. Algorithms:

2.1 Convolutional Neural Network: Convolutional Neural networks are specialized deep neural networks which processes the data that has input shape like a 2D matrix. CNN works well with images and are easily represented as a 2D matrix. Image classification and identification can be easily done using CNN. It can determine whether an image is a bird, a plane or Superman, etc. Important features of an image can be extracted by scanning the image from left to right and top to bottom and finally the features are combined together to classify images. It can deal with the images that have been translated, rotated, scaled and changes in perspective.

2.2 Long Short-Term Memory: LSTM are type of RNN (recurrent neural network) [2] which is well suited for sequence prediction problems. We can predict what the next words will be based on the previous text. It has shown itself effective from the traditional RNN by overcoming the limitations of RNN. LSTM can carry out relevant information throughout the processing, it discards non-relevant information

3. Model architecture: For defining the structure of our model, we will be using the Keras Model from Functional API. The model is composed of two parts:

- An Encoder which uses pre-trained **ResNet50** network to extract features from the images. The feature volume is passed thru a fully connected layer with the same number of nodes as the word embedding used in the second part below. This allows to combine extracted special features from the images and the sampled captions during training.
- A Decoder primarily composed of a **LSTM** network. The decoder takes on the sample captions during the training phase and generates a caption for a new image during inference, 'decoding' the proposed features extracted from the image. A fully connected layer at the end allows to map the hidden space of the LSTM to the vocab space, producing a prediction for each word sequentially.

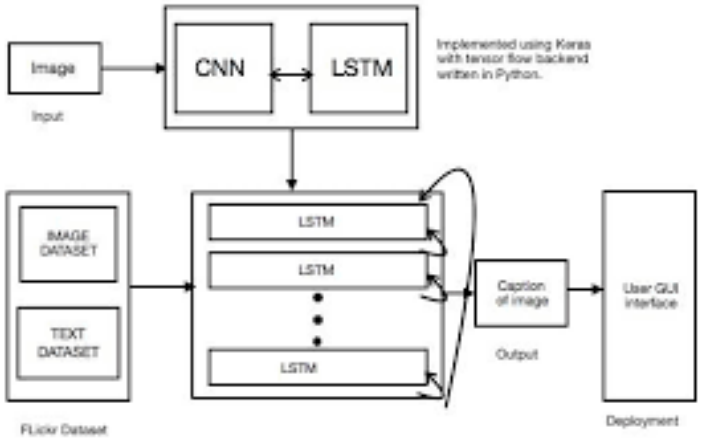


Fig. 1. System Architecture of Image Caption Generator

Words proposed to the Decoder are pre-processed using tokenization and a word embedding step. Using an embedding has the advantage of being independent of the size of the dictionary (contrary to a simpler onehot-encoding approach). Each word is represented by a fixed sized vector in the embedding space. The same embedding dimension is used on the extracted image features which allows to concatenate both (features, caption) as input pair for the training. Using its memory

capabilities, the LSTM learns to map words to the specific features extracted from the images and to form a meaningful caption summarizing the scene. The embedding, jointly trained during learning, contributes to the performance: The learned representations capture some semantic of the language, helping the vision component by defining coherent proximities between words. The model has **1.36 million trainable parameters**. The weights file takes up **5MB space**.

The model summary is shown below:

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 35)]         0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 2048)]       0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 35, 50)       74700       input_3[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 2048)         0           input_2[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 35, 50)       0           embedding[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          524544      dropout[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 256)          314368      dropout_1[0][0]
__________________________________________________________________________________________________
add (Add)                       (None, 256)          0           dense[0][0]
                                                                 lstm[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          65792       add[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 1494)         383958      dense_1[0][0]
==================================================================================================
Total params: 1,363,362
Trainable params: 1,363,362
Non-trainable params: 0
```

4. <u>Training methodology:</u> The data set contains 8,000 images, each of which is accompanied by 5 different captions that offer detailed explanations of the important people and events. Since a pre-trained Encoder was used, the images were pre-processed into a format that the pre-trained Encoder is familiar with. Three inputs are required by the encoder. Since the vocabulary contains 1652 unique terms, each one will be identified by a unique integral index ranging from 1 to 1652.

Our first step was to find our feature maps for a given image. We imported the **resNet model** present in the keras library. Then we pushed in every image in our training set and testing set, and stored these feature

maps into a pickle file. A pickle file is very useful when we wish to maintain the serialization of an object structure. To match the number of features in our image to the vocabulary size, we write an embedding algorithm that clusters the image features according to the vocabulary size.

For training our model We are using **Adam's optimizer** and l**oss function** as categorical cross-entropy. We are training the model by starting from **5** till **30 epochs** in steps of 5 which will be enough for predicting the output. In case you have more computational power (no. of GPU's) you can train it by decreasing batch size and increasing number of epochs.

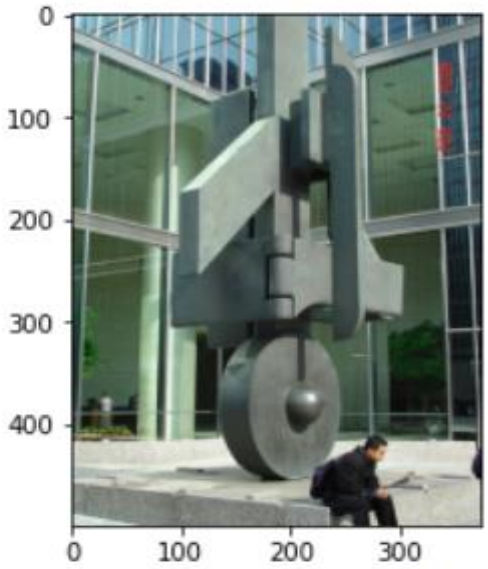
5. Training level experiments:

1. **Models:** We have tried different models like **ResNet50** and **Cgg-13**. The one that best met the computational requirements was chosen.
2. **Epochs:** We have started with **5 epochs** and increased till **30 epochs**. Since it taking more time, we have stopped increasing epochs.

6. Test data evaluation: Initially when tried with 5 and 10 epochs we were able to reasonably caption 2 out of 5 but on increasing the epochs to 30 we obtained better results that reasonably caption 3 out of 5 images provided in the subjective images folder. On the 1000 images in the test dataset, the mean sentence **BLEU score was 0.412**.

The sample images and their predicted captions is shown below:

Sample Image 1:



man be sit on sidewalk near an orange building

Sample Image 2:

group of person ride bicycle down street

Sample Image 3:

three young boy be sit around table of water

Sample Image 4:

woman in white shirt be walk along the sand

Sample Image 5:

dog leap into the air

References:

1. https://drive.google.com/drive/folders/1RQ5qHm0aVFqWDG9VBiSnXlNPl5T15Wf_?usp=sharing

2. https://arxiv.org/abs/1502.03044

3. https://data-flair.training/blogs/python-based-project-image-caption-generator-cnn/

4. https://www.nltk.org/_modules/nltk/translate/bleu_score.html

5. https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c