University of California, Santa Barbara

Project Report

Image Captioning

Author:
Sai Nikhil Maram

Supervisor:
Prof Yuan Fang Wang

August 12, 2018

1 Objective

Developing a prototype that could generate an appropriate caption for the given image. Understand the Image Captioning problem in Computer vision and Natural language processing. Demonstrate how different components are built, preprocessing and training procedures are employed for the system.

2 Introduction

Detecting the objects present in the image is a fundamental problem in Computer Vision. Objects along with relations between them define the image. Finding the relations between objects present in the image requires detecting the objects and representing the relation between them in a Natural Language, to get an image caption as in Figure 1. Image captioning requires user to understand Computer Vision(CV) and Natural Language processing(NLP).
In this project we look at a generative model based on Convolution Neural Networks(CNN) and Recurrent Neural Network(RNN) to generate the image caption. The model is trained to maximize the likelihood of the target description sentence given the training image. Experiments on MS-COCO image/text datasets show the accuracy of the model and the fluency of the language it learns solely from image descriptions.

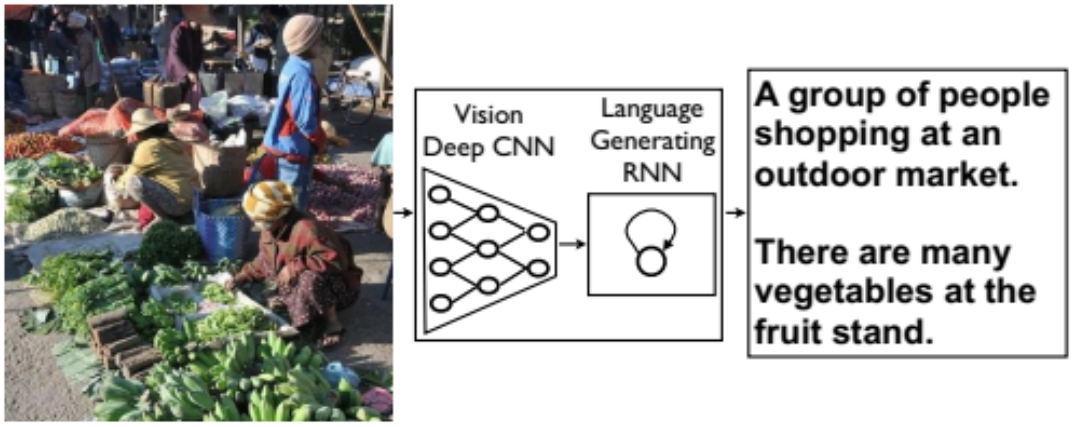


Figure 1: NIC model (vision CNN followed by language generating RNN)

This project based its implementation on the method described in the CVPR2015 paper "Show and Tell: A Neural Image Caption Generator".

3 Model

The current model is inspired from Sequence2Sequence model generally used in Machine Translation(MT) in NLP. RNN's are used in embedding text into vector space by preserving temporal information. In traditional MT, the input and output are texts in different languages. Hence RNN's are used as encoders/decoders in Sequence2Sequence Model. The encoder encodes the input text into common vector space and decoder decodes the vector space to generate the output text in MT. For Image captioning, the input is an image and output is a caption which is in text format. Hence we need a model which can encode the image into common vector space. Therefore RNN is replaced by CNN as an Encoder in our adapted Sequence2Sequence Model as seen in Figure 2.
Our Model is a neural based network consisting of a vision CNN followed by a language generating RNN. The image encoding done by CNN is fed as first input to RNN model(LSTM).

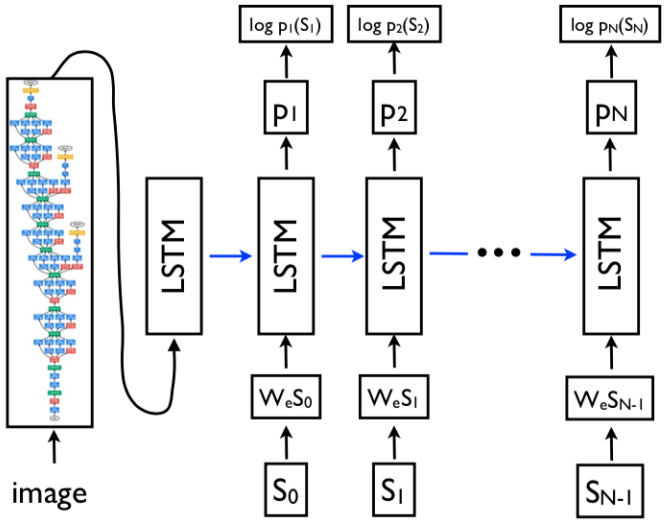


Figure 2: LSTM model combined with a CNN image embedder.

.

4 Work Done

The following section describes the work done by us in implementing the complete model. We have used tensorflow and Python. The code we implemented can be found here. This section talks about how the model is built and the next section talks about the preprocessing and training process involved.

4.1 Encoder

A pre-trained vgg16 CNN model on Imagenet is used as an encoder. The Vgg16 model consists of 5 convolution layers, 2 fully connected layers & 1 softmax layer as in Figure 3. To get the pre-trained

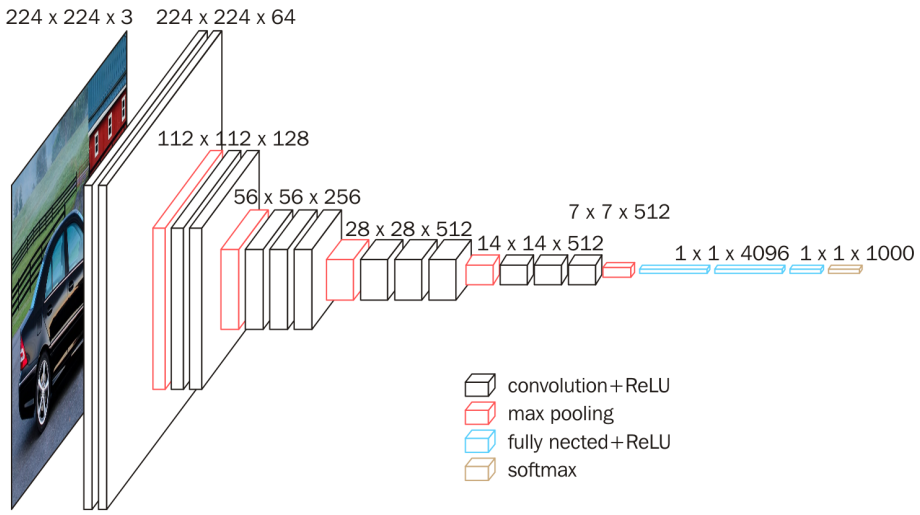


Figure 3: VGG16 model

parameters for the CNN. The model needs to be built in tensorflow and we need to ensure the

namescope of different parameters present in the model we built and pre-trained model file should exactly match for the restore operation in tensorflow to be successful.
The input to CNN is an image of size 224*224*3 (RGB). The output from the second fully connected Layer of size 4096 is taken so we get more information about different objects present in the image and compressed into size of 512 and fed to the decoder.

4.2 Decoder

As previously discussed, RNNs are used to decode the vector into text by preserving temporal information. We have used LSTM as a RNN module to mitigate the problems of vanishing gradient descent. We have a fixed length of LSTM units as we will have a threshold on maximum number of words each caption can have. A lot of pre-processing needs to be done on text as any computation model only deals with vectors. For this purpose following pre-processing steps are done.

1. Add start and stop tokens : As all the captions will not be of same length, we need to know the begin and end of the caption. Hence we use a common start and stop token for each caption.

2. Padding : Since we have constant number of LSTM units in the decoder which each caption has to go through. If the number of words in the caption is less than number of LSTM units, then we pad the rest of the words with $< unk >$.

3. Word to indexes : Since any computation model deals with only vectors. We assign each unique word with an unique index. The index of the word depends on number of unique words that are present in the vocabulary. For this purpose, we first iterate through all the captions that are present in the data set and list out unique words in them and assign an index to each word. After this step, we can represent a caption in form of indexes corresponding to each word.

4. Embedding : If we represent each word by index and representing each index by one hot vector. Then each vector will be of size vocabulary and any computation will be resulting in sparse matrix calculations. So a dimensionality reduction technique is used where each index is now represented as a 512 dimensional vector. This is popularly called word2Vec in NLP. So an embedding matrix(W_e) is created which gives 512 dimension vector to each index. This is the reason that output of CNN is compressed into 512 size vector.

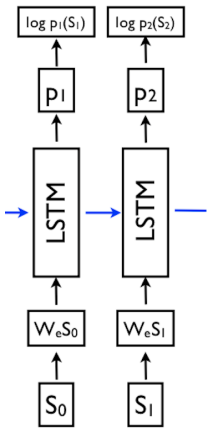


Figure 4: RNN model

After doing above pre-processing steps, we feed the inputs to the RNN model. In Figure 4, S_0 represents index corresponding to a word. W_0S_0 corresponding to embedded vector corresponding to index. This when fed with previous hidden state of LSTM to the current LSTM unit, results in current output and current hidden state. The initial hidden state of the LSTM is zeros. The output which we get will be a vector of dimension 512. This needs to be converted back into index of the word to get the predicted output word.
During training, we have the input word at each LSTM input to predict the output but while evaluation, the output word of the previous LSTM is fed as input to current LSTM.

5 Experiment Setup

5.1 Data Set

We have used MS-COCO dataset which contains images/captions. We first pre-processed the dataset to get an image and its corresponding caption by parsing through the cpation JSON file which contains image ids and captions. Data Description

Dataset	training	valid	test
MSCOCO	82783	40504	40775

Table 1: MSCOCO Dataset

5.2 Training

Following steps are done for training.

1. Initialized weights of CNN component to a pre-trained model (VGG16).
2. Creating a Word Embedding Matrix.
3. Cross Entropy as Loss function
4. Adam Optimizer with initial learning rate of 0.0001 and dropout with keep probability of 0.9
5. All RNN inputs are right padded. Maximum Caption length is 20.
6. Words are Embedded into 512 dimension vector.
7. A Batch size of 32 is used to run for 5 epochs.
8. BLEU Score is used as an evaluation metric.

CNN parameters aren't updated during training. The parameters that are updated during training are Word Embedding matrix, LSTM parameters. Since each epoch runs for 4hours. Due to computational resources constraint we ran the system only for 5 epochs. Hence validation bleu score is not computed between two epochs for early stop to avoid overfitting.

6 Results

6.1 Metrics

BLEU score is used as an evaluation metric. Since the system is run for only 5 epochs. The metric is calculated for validation dataset from the model generated after 5 epochs. Here are the various variations of BLEU score for MS COCO dataset. The BLEU scores compares the word by word between ground truth and predicted text. BLUE-1 checks one to one correspondence where as BLUE-2 checks the context words for a given word in ground-truth and predicted text.

Metric	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Value	62.9%	43.6%	29.0%	19.3%

Table 2: Scores on Dataset

6.2 Output

Here are some test outputs generated by the Image captioning system ...

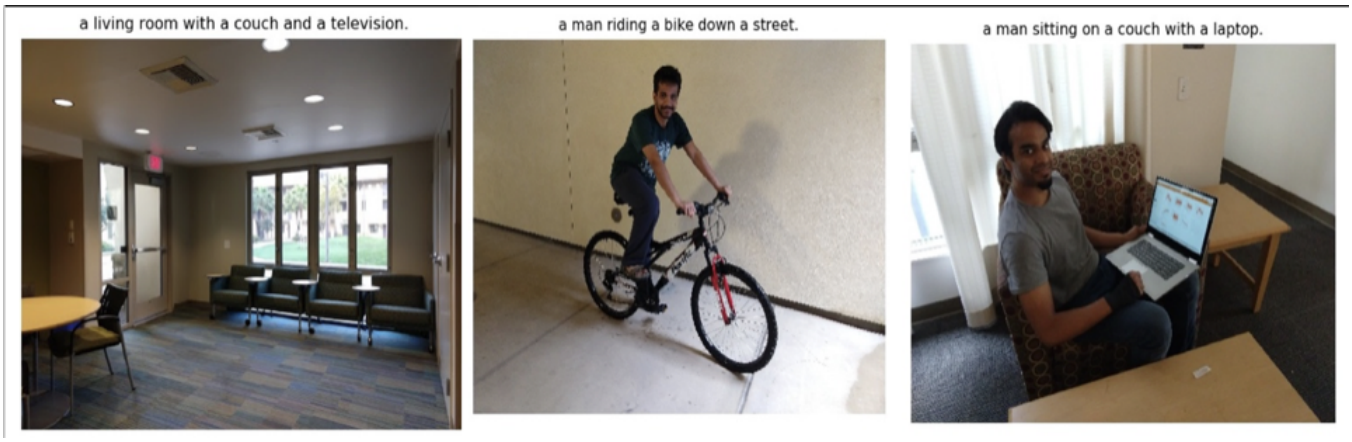
6.2.1 Images from the validation MS-COCO data set

It can be seen that in the left image, the model was able to detect the bus on the street including its color. The middle image it is little confused with pizza as cake. In the right most image, the model was able to predict the surfing on wave with little confusion between man and woman.

6.2.2 Images from the flickr data set

It can be seen in the left image, the model was able to detect skateboarding with a minor error of treating a boy as a man. In the middle image it was able to detect a baby and as the man was not clearly visible so perhaps most of the training set might have woman holding a baby, hence the model detects it as a woman. In the right most swinging of leg is considered to be a bat and instead of woman model detects it as man.

6.2.3 Images we have taken



These are the images taken by us for evaluating our model. The left image is photo of common room in graduate housing. The model was able to detect the chairs as couch as they are arranged together. On careful observation, the windows look like a television displaying some picture. Hence the model detects it as a living room with couch and television, In the middle image the model was able to detect the cycle and the person. In the right image model was able to detect a person,couch and a laptop which accurately describes the image.

6.3 Conclusion

It is clear from these experiments that, the model was able to caption the images in cases where objects are clear and few. With few epoch run, the model was able to get accurate description of images better than a formal rule based approach of image captioning.