Image Description Generation using Standard RNN and LSTM

Roshan Kathawate
Indiana University, Bloomington
107 S Indiana Ave, Bloomington, IN 47405
rkathawa@iu.edu

Abstract

We present a comparative study of standard recurrent neural network (RNN) and long short term memory recurrent neural network (LSTM) for image description generation. Our approach leverage per-trained VGGNet model for extracting features from images and RNN learns about inter-modal correspondences between language and visual data. We then describes architectural details about standard RNN and LSTM used for generating image description. We demonstrate that LSTM outperform the RNN in long image description generation on MSCOCO dataset.

1. Introduction

Describing an image is a simple task for a human but for visual recognition models it is a challenging problem. However, remarkable work has been done on labeling images with fixed set of visual categories by [1] and [2] and commendable work on generating dense description for images has been done by [3]. This study is motivated by [3] and tried to implement the same approach.

In this work, we have implemented RNN and LSTM for generating dense descriptions for images using pretrained VGGNet model [4]. In this study my goal is to understand the architecture of *RNNs and LSTMs* and how they are combined with CNN to implement an image captioning system. In achieving this goal, the biggest challenge is to design a model that learns from the data without any fixed templates, categories, or rules and generate natural language descriptions of varying length. Another practical challenge is about efficient implementation of these models without using any specific computer vision libraries like Tensotflow, Theano, Caffe etc.

In this approach, I am considering each description of an image in MSCOCO dataset as weak labels in which contiguous segments of words corresponds to some specific but unknown location in the image. Our approach is to infer these correspondence between image features and their descriptions using an end-to-end model.

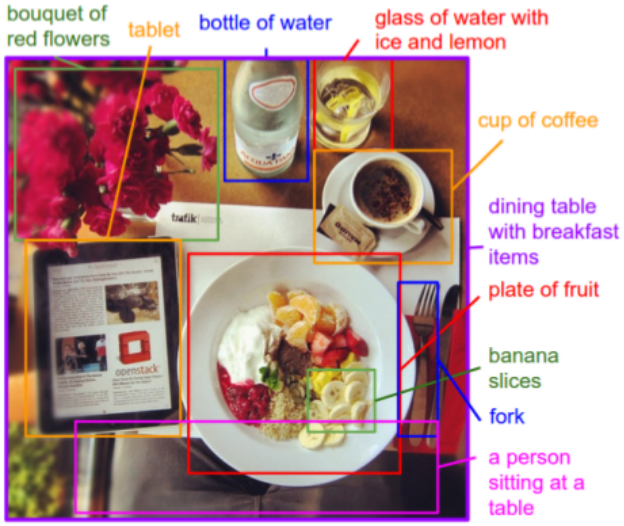

Figure 1: Motivation/Concept Figure: Our model treats language as a rich label space and generates descriptions of image regions.[3] Image features are extracted using pretrained VGGNet.

My contributions are:

- End-to-end model for generating natural language descriptions for an image.
- In network word-to-vect encoder
- Performance comparison between standard RNN and LSTM
- Implementation using Python and Numpy

2. Related Work

Dense image description generation. A commendable work in generating dense descriptions of images is done by Andrej Karpathy et. al. in [3]. In that work inference about the alignment between image regions and their descriptions is learn using a generative model of descriptions . This work is motivated and based on [3] by Andrej Karpathy et. al. Also, work by Barnard et al. [5] and Socher et al. [6] studies the multimodal correspondence between words and images to annotate

segments of images. Several works such as [34, 18, 15, 33] studied the problem of scene understanding. However, these studies are focused on correctly labeling scenes, objects and regions with a fixed set of categories.

studied the problem of holistic
scene understanding in which the scene type, objects and their spatial support in the image is inferred

3. Our Model

Overview. The goal of our model is to generate descriptions of images. At the training time, already extracted image features from VGGNet and their corresponding descriptions are given to our model. In this step our model learns to map segments of words to the corresponding image region using already extracted features. At the test time, only features are given to the model and it infers the descriptions of images using learned parameters in the training step.

3.1. Image Representation

In this work, I have used extracted features from fc7 layer of the VGG-16 network pretrained on ImageNet. However, to limit the processing time and memory requirement I have used features with reduced dimensionality of 512 from 4096 provided by the fc7 layer.

3.2. Sentences Representation

Working with strings is computationally inefficient so I have encoded all the captions. First, I have created a vocabulary containing 1004 unique words and then represented each word with an integer ID. There are a couple of special tokens that I have added to the vocabulary. I have appended <START> and <END> token to the beginning and end of each caption respectively. Rare words are replaced with a special <UNK> (for “unknown”) token. Also, as I am training with minibatches containing captions of varying lengths, I am padding them a <NULL> token after the <END> token.
As common in deep learning systems, I have represented each word using 1-hot vector. Each word in the vocublary is associated with a vector and these vectors are learned jointly with the rest of the system.

Here, W_w is a word embedding matrix and $\mathbb{I}_t$ is a indicator column vector that has a single one at the index of the t-th word in a word vocabulary.

3.3. Standard RNN

In this work, I have experimented with standard RNN

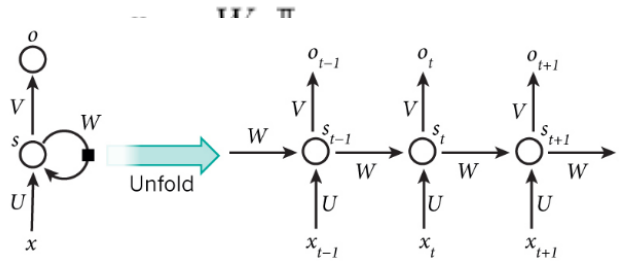

Figure 2: A recurrent neural network and the unfolding in time of the computation involved in its forward computation.

which has no internal cell state.
Predicting word probabilities (Forward Propagation).
*St = tanh(U*Xt+ W*St-1+b)*

*Ot = Softmax(V * St)*

Here, St is a next hidden state, U is a input-hidden weight matrix, W is a hidden-to-hidden weight matrix, Xt is a input vector at t time step, St-1 is a previous hidden state and b is a bais vector. There dimensions for a minibatch of size N, hidden state dimension H and each input vector of D dimensions are:

- Xt = (N, D)
- St = (N, H)
- U = (D, H)
- W = (H, H)
- b = (H,)

Thus, in this model we need to learn 2HC + H*H parameters. The dimensions also shows the bottleneck of our model. Also, each Ot indicates a vector of vocabulary size elements where each element represents probability of that word being the next word in the sentence.
Calculating the loss (Backword propogation). To train our network, we need to measure error that it makes while predicting next word. For this purpose I used cross-entropy loss function L, and our goal is to find the values for parameters U, V, W that minimizes the cost function.

$$L(y, o) = -\frac{1}{N}\sum_{n \in N} y_n \log o_n$$

Here, N is number of training examples, y is ground truth next word and o is a predicted word.
So, to find values for parameters U, V, W that minimizes the loss function on the training data, I used stochastic gradient descent (SGD). Other hyper parameters were:

- number of epochs = 50
- batch size = 25

- learning rate = 0.001
- learning decay rate = 0.95
  However, unlike classification models, image captioning models behave differently at training and test time. At training time, we have ground truth words which we feed as an input at each time step. But, at test time, we sample from the distribution over the vocabulary at each time step , and feed the sampled word as input to the RNN at next time step.

3.4. LSTM

The only difference between standard RNN and LSTM is that it has internal memory cells. These memory cells makes them capable of learning long-term dependencies.

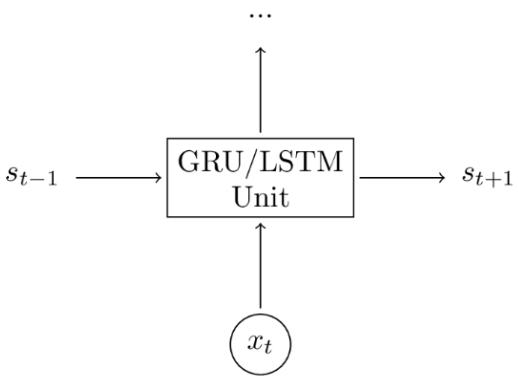

Figure 2: A long short term memory recurrent neural network

In case of standard RNN, due to chain of matrix multiplication in back propagation gradients vanishes or explodes thus making network less effective in learning long sequences. However, in LSTM these problems are taken care of by gating mechanism.

$$
\begin{aligned}
i &= \sigma(x_t U^i + s_{t-1} W^i) \\
f &= \sigma(x_t U^f + s_{t-1} W^f) \\
o &= \sigma(x_t U^o + s_{t-1} W^o) \\
g &= \ tanh(x_t U^g + s_{t-1} W^g) \\
c_t &= c_{t-1} \circ f + g \circ i \\
s_t &= \tanh(c_t) \circ o
\end{aligned}
$$

Here, at each time step model receives input Xt and previous hidden state St-1. But most importantly LSTM maintains h-dimensional cell state, so our model also receives previous cell state. Learnable parameters are input-to-hidden matrix of dimension 4H X D, a hidden-to-hidden matrix of 4H X H dimensions, and a bias vector of 4H dimensions.

LSTM are same as RNN but here we have input (i), forget (f), and output gates(0). They are called gates because they use sigmoid functions which squashes the values of input vectors between 0 and 1 thus control another multiplying vector and defines how much of that other vector you want to "let through".The input gate defines how much of the newly computed state for the current input you want to let through. The forget gate controls the previous state. Finally, The output gate defines how much of the internal state you want to expose to the higher layers and the next time step. All the gates have the same dimensions, the size of our hidden state. Gate g is a hidden state that is computed according to the current input and the previous hidden state. And Ct is a internal memory of the unit. It consists of previous memory Ct-1 multiplied by the forget gate, and the newly computed hidden state g, multiplied by the input gate i. What I observed that if we fix the input gate all 1's, the forget gate to all 0's and the output gate to all one's you almost get standard RNN.

4. **Data**

In this work, I have used the 2014 release of the Microsoft COCO dataset which is the standard testbed for image captioning. The dataset consists of 80,000 training images and 40,000 validation images. Each image is annotated with 5 captions written by workers on Amazon Mechanical Trunk.

5. **Results**

I have compared training losses for both networks.

As seen in the training loss, my LSTM implementation has not performed up to the mark and after tuning for hyper parameters and several attempts of debugging, RNN is still outperforming the LSTM. One reason could be that I could not train it on entire dataset due to memory and time constraints.

6. **Result Examples**

<START> a wooden sign with several <UNK> on it pointing <END>

<START> a small white bird walking across a lush green field <END>

<START> a white bus driving down a street next to traffic lights <END>

<START> a slice of pepperoni and <UNK> pizza with fries <END>

7. **Conclusion**

In this work, I have studied nuances of RNN and LSTM. Observed how gradients are vanishes and explodes in RNN and how they are overcomes by LSTM. However, implement ion wise I have to explore LSTM so that it can outperformed the RNN.

8. **References**

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge, 2014.
[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. International Journal of Computer Vision, 88(2):303338, June 2010.
[3] Andrej Karapahty, Li Fei-Fei, Deep Visual-Semantic Alignments for Generating Image Descriptions
[4] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large scale visual recognition challenge 2015
[5]K. Barnard, P. Duygulu, D. Forsyth, N. De Freitas, D. M.Blei, and M. I. Jordan. Matching words and pictures. JMLR,2003.
[6]R. Socher and L. Fei-Fei. Connecting modalities: Semisupervised segmentation and annotation of images using unaligned text corpora. In CVPR, 2010.

Code and other Refereces

1. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

2. Skeleton code for captioning_solver.py and captionImage.py are refered from http://cs231n.stanford.edu/
3. http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/