Neural Image Caption Generation

Giri Gaurav Bhatnagar
Dept. of Information Science
PESIT
Bangalore, India
girigauravbhatnagar100@gmail.com

Aniruddha Hore
Dept. of Information Science
PESIT
Bangalore, India
being.Anni@gmail.com

Himanshu Singh
Dept. of Information Science
PESIT
Bangalore, India
kishu0495@gmail.com

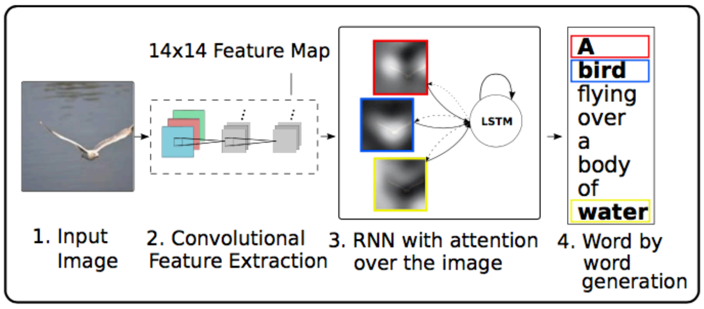
***Abstract*— Inspired by recent work in machine translation and object detection, we introduce an attention based model that automatically learns to describe the content of images. We describe how we can train this model in a deterministic manner using standard back-propagation techniques and stochastically by maximising a variational lower bound. We also show through visualisation how the model is able to automatically learn to fix its gaze on salient objects while generating the cor- responding words in the output sequence. We validate the use of attention with state-of-the- art performance on three benchmark datasets: Flickr8k, Flickr30k and MS COCO.**

Keywords*— RNN, CNN, LSTM, Image Caption,

I. INTRODUCTION

With the developing fields of artificial intelligence and machine learning, image captioning models can be used in several applications and make machines further understand the semantics of human language. Automatically generating captions of an image is a very close to the heart of scene understanding — one of the primary goals of computer vision. Not only must caption generation models be powerful enough to solve the computer vision challenges of determining which objects are in an image, but they must also be capable of capturing and expressing their relationships in a natural language. For this reason, caption generation has long been viewed as a difficult problem. It is a very important challenge for machine learning algorithms, as it amounts to mimicking the remarkable human ability to compress huge amounts of salient visual information into descriptive language.

One of the most curious facets of the human visual system is the presence of attention. Rather than compress an entire image into a static representation, attention allows for salient features to dynamically come to the forefront as needed. This is especially important when there is a lot of clutter in an image. Using representations (such as those from the top layer of a convnet) that distill information in image down to the most salient objects is one effective solution that has been widely adopted in previous work. Unfortunately, this has one potential drawback of losing information which could be useful for richer, more descriptive captions. Using more low-level representation can help preserve this information. However working with these features necessitates a powerful mechanism to steer the model to information important to the task at hand.


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

In this paper, we describe approaches to caption generation that attempt to incorporate a form of attention with two variants: a "hard" attention mechanism and a "soft" attention mechanism. We also show how one advantage of including attention is the ability to visualise what the model "sees". Encouraged by recent advances in caption generation and inspired by recent success in employing attention in machine translation and object recognition, we investigate models that can attend to salient part of an image while generating its caption.

The contributions of this paper are the following:

- We introduce two attention-based image caption generators under a common framework: 1) a "soft" deterministic attention mechanism trainable by standard back-propagation methods and, 2) a "hard" stochastic attention mechanism trainable by maximising an approximate variational lower bound or equivalently by REINFORCE.

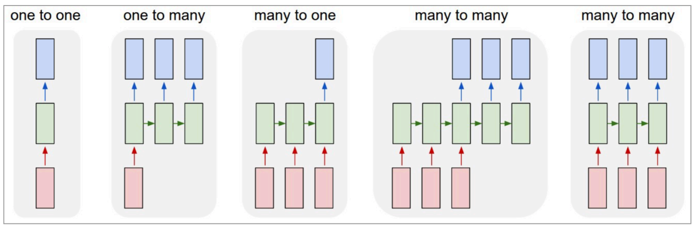
- We show how we can gain insight and interpret the results of this framework by visualising "where" and "what" the attention focused on.

- Finally, we quantitatively validate the usefulness of attention in caption generation with state of the art performance.

II. Recurrent Neural Networks

A. Sequences.

Depending on one's background one might be wondering: *What makes Recurrent Networks so special*? A glaring limitation of Vanilla Neural Networks (and also Convolutional Networks) is that their API is too constrained: they accept a fixed-sized vector as input (e.g. an image) and produce a fixed-sized vector as output (e.g. probabilities of different classes). Not only that: These models perform this mapping using a fixed amount of computational steps (e.g. the number of layers in the model). The core reason that recurrent nets are more exciting is that they allow us to operate over *sequences* of vectors: Sequences in the input, the output, or in the most general case both. A few examples may make this more concrete:



As one might expect, the sequence regime of operation is much more powerful compared to fixed networks that are doomed from the get-go by a fixed number of computational steps, and hence also much more appealing for those of us who aspire to build more intelligent systems. Moreover, as we'll see in a bit, RNNs combine the input vector with their state vector with a fixed (but learned) function to produce a new state vector. This can in programming terms be interpreted as running a fixed program with certain inputs and some internal variables. Viewed this way, RNNs essentially describe programs.

B. Sequential processing in absence of sequences.

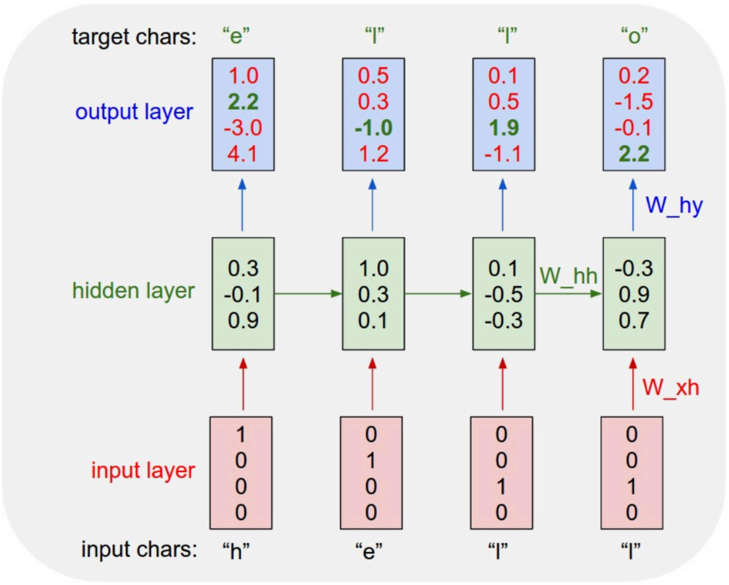
One might be thinking that having sequences as inputs or outputs could be relatively rare, but an important point to realise is that even if your inputs/outputs are fixed vectors, it is still possible to use this powerful formalism to process them in a sequential manner. The takeaway is that even if your data is not in form of sequences, you can still formulate and train powerful models that learn to process it sequentially. You're learning stateful programs that process your fixed-sized data.

III. Character-Level Language Models

So we have an idea about what RNNs are, why they are super exciting, and how they work. We'll now ground this in a fun application: We'll train RNN character-level language models. That is, we'll give the RNN a huge chunk of text and ask it to model the probability distribution of the next character in the sequence given a sequence of previous characters. This will then allow us to generate new text one character at a time.

As a working example, suppose we only had a vocabulary of four possible letters "helo", and wanted to train an RNN on the training sequence "hello". This training sequence is in fact a source of 4 separate training examples: 1. The probability of "e" should be likely given the context of "h", 2. "l" should be likely in the context of "he", 3. "l" should also be likely given the context of "hel", and finally 4. "o" should be likely given the context of "hell".

Concretely, we will encode each character into a vector using 1-of-k encoding (i.e. all zero except for a single one at the index of the character in the vocabulary), and feed them into the RNN one at a time with the '"step" function. We will then observe a sequence of 4-dimensional output vectors (one dimension per character), which we interpret as the confidence the RNN currently assigns to each character coming next in the sequence. Here's a diagram:



For example, we see that in the first time step when the RNN saw the character "h" it assigned confidence of 1.0 to the next letter being "h", 2.2 to letter "e", -3.0 to "l", and 4.1 to "o". Since in our training data (the string "hello") the next correct character is "e", we would like to increase its confidence (green) and decrease the confidence of all other letters (red). Similarly, we have a desired target character at every one of the 4 time steps that we'd like the network to assign a greater confidence to. Since the RNN consists entirely of differentiable operations we can run the back-propagation algorithm (this is just a recursive application of the chain rule from calculus) to figure out in what direction we should adjust every one of its weights to increase the scores of the correct targets (green bold numbers). We can then perform a *parameter update*, which nudges every weight a tiny amount in this gradient direction. If we were to feed the same inputs to the RNN after the parameter update we would find that the scores of the correct characters (e.g. "e" in the first time step) would be slightly higher (e.g. 2.3 instead of 2.2), and the scores of incorrect characters would be slightly lower. We then repeat this process over and over many times until the network converges and its predictions are eventually consistent with the training data in that correct characters are always predicted next.

Notice also that the first time the character “l” is input, the target is “l”, but the second time the target is “o”. The RNN therefore cannot rely on the input alone and must use its recurrent connection to keep track of the context to achieve this task.

At test time, we feed a character into the RNN and get a distribution over what characters are likely to come next. We sample from this distribution, and feed it right back in to get the next letter. Repeat this process and you’re sampling text! Lets now train an RNN on different datasets and see what happens.

IV. Our Model

A. Overview

The ultimate goal of our model is to generate descriptions of image regions. During training, the input to our model is a set of images and their corresponding sentence descriptions (Figure 2). We first present a model that aligns sentence snippets to the visual regions that they describe through a multimodal embedding. We then treat these correspondences as training data for a second, multi- modal Recurrent Neural Network model that learns to generate the snippets.

B. Implementation

Our alignment model assumes an input dataset of images and their sentence descriptions. Our key insight is that sentences written by people make frequent references to some particular, but unknown location in the image. For exam- ple, in Figure 2, the words “Tabby cat is leaning” refer to the cat, the words “wooden table” refer to the table, etc. We would like to infer these latent correspondences, with the eventual goal of later learning to generate these snippets from image regions. We build on the approach of Karpathy, who learn to ground dependency tree relations to image regions with a ranking objective. Our contribution is in the use of bidirectional recurrent neural network to compute word representations in the sentence, dispensing of the need to compute dependency trees and allowing unbounded interactions of words and their context in the sentence. We also substantially simplify their objective and show that both modifications improve ranking performance.

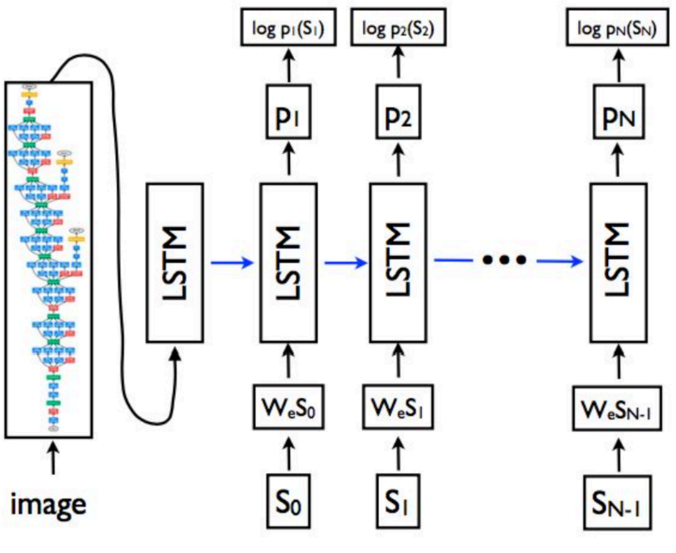
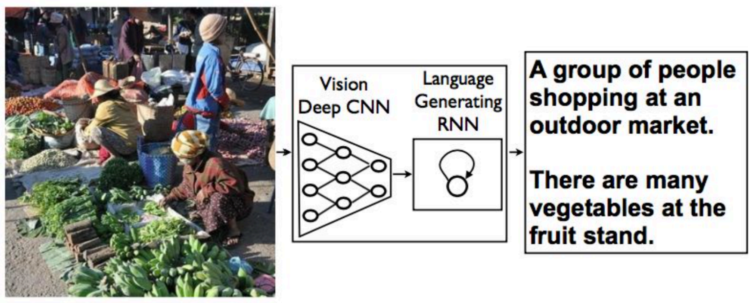


Fig: GoogleNet and LSTM

C. Architecture

We first describe neural networks that map words and image regions into a common, multimodal embedding. Then we introduce our novel objective, which learns the embedding representations so that semantically similar concepts across the two modalities occupy nearby regions of the space.


Vision Deep CNN
Language Generating RNN
A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.

D. Representing images

Following prior work, we observe that sentence descriptions make frequent references to objects and their at- tributes. Thus, we follow the method of Girshick et al. to detect objects in every image with a Region Convolutional Neural Network (RCNN). The CNN is pre-trained on ImageNet [6] and fine-tuned on the 200 classes of the Ima- geNet Detection Challenge. Following Karpathy et al. [24], we use the top 19 detected locations in addition to the whole image and compute the representations based on the pixels Ib inside each bounding box as follows:

$$v = W_m[CNN_{\theta_c}(I_b)] + b_m, \qquad (1)$$

where CNN(Ib) transforms the pixels inside bounding box Ib into 4096-dimensional activations of the fully connected layer immediately before the classifier. The CNN parameters θ contain approximately 60 million parameters. The matrix Wm has dimensions h × 4096, where h is the size of the multimodal embedding space (h ranges from 1000-1600 in our experiments). Every image is thus represented as a set of h-dimensional vectors {vi | i = 1…20}.

E. Representing sentences

To establish the inter-modal relationships, we would like to represent the words in the sentence in the same h-dimensional embedding space that the image regions occupy. The simplest approach might be to project every individual word directly into this embedding. However, this approach does not consider any ordering and word context information in the sentence. An extension to this idea is to use word bigrams, or dependency tree relations as previously proposed [24]. However, this still imposes an arbitrary maximum size of the context window and requires the use of Dependency Tree Parsers that might be trained on unrelated text corpora.

To address these concerns, we propose to use a Bidirectional Recurrent Neural Network (BRNN) [46] to compute the word representations. The BRNN takes a sequence ofN words (encoded in a 1-of-k representation) and trans- forms each one into an h-dimensional vector. However, the representation of each word is enriched by a variably-sized context around that word. Using the index t = 1 . . . N to denote the position of a word in a sentence, the precise form of the BRNN is as follows:

$$x_t = W_w \mathbb{I}_t \tag{2}$$
$$e_t = f(W_e x_t + b_e) \tag{3}$$
$$h_t^f = f(e_t + W_f h_{t-1}^f + b_f) \tag{4}$$
$$h_t^b = f(e_t + W_b h_{t+1}^b + b_b) \tag{5}$$
$$s_t = f(W_d(h_t^f + h_t^b) + b_d). \tag{6}$$

Here, It is an indicator column vector that has a single one at the index of the t-th word in a word vocabulary. The weights Ww specify a word embedding matrix that we initialise with 300-dimensional word2vec weights and keep fixed due to overfitting concerns. However, in practice we find little change in final performance when these vectors are trained, even from random initialisation. Note that the BRNN consists of two independent streams of processing, one moving left to right (hft) and the other right to left (hbt) (see Figure 3 for diagram). The final h-dimensional representation st for the t-th word is a function of both the word at that location and also its surrounding context in the sentence. Technically, every st is a function of all words in the entire sentence, but our empirical finding is that the final word representations (st) align most strongly to the visual concept of the word at that location (It).

We learn the parameters We , Wf , Wb , Wd and the respective biases be, bf , bb, bd. A typical size of the hidden representation in our experiments ranges between 300-600 dimensions. We set the activation function f to the rectified linear unit (ReLU), which computes f : x → max(0, x).

V. Image Caption Generation

A. Model Details

In this section, we describe the two variants of our attention-based model by first describing their common

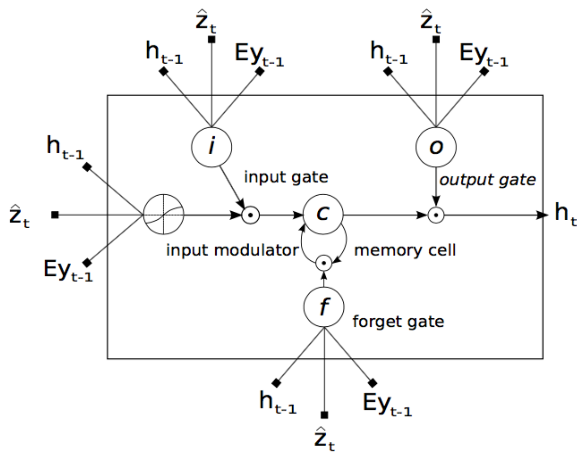


Figure 4. A LSTM cell, lines with bolded squares imply projections with a learnt weight vector. Each cell learns how to weigh its input components (input gate), while learning how to modulate that contribution to the memory (input modulator). It also learns weights which erase the memory cell (forget gate), and weights which control how this memory should be emitted (output gate).

framework. The main difference is the definition of the φ function which we describe in detail in Section 4. We denote vectors with bolded font and matrices with capital letters. In our description below, we suppress bias terms for readability.

B. Encoder: Convolutional Details

Our model takes a single raw image and generates a caption 'y' encoded as a sequence of 1-of-K encoded words.

$$y = \{y_1,\dots,y_C\},\ y_i \in R^{K_m}$$

where K is the size of the vocabulary and C is the length of the caption.

We use a convolutional neural network in order to extract a set of feature vectors which we refer to as annotation vectors.

$$a = \{a_1,\dots,a_L\},\ a_i \in R^D$$

The extractor produces L vectors, each of which is a D-dimensional representation corresponding to a part of the image.

In order to obtain a correspondence between the feature vectors and portions of the 2-D image, we extract features from a lower convolutional layer unlike previous work which instead used a fully connected layer. This allows the decoder to selectively focus on certain parts of an image by selecting a subset of all the feature vectors.

C. Decoder: Long Short-Term Memory

We use a long short-term memory (LSTM) network that produces a caption by generating one word at every time step conditioned on a context vector, the previous hidden state and the previously generated words.Using $T_{s,t} : R^s \rightarrow R^t$ to denote a simple affine transformation with parameters that are learned,

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} \mathbf{E}\mathbf{y}_{t-1} \\ \mathbf{h}_{t-1} \\ \hat{\mathbf{z}}_t \end{pmatrix} \tag{1}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \tag{2}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \tag{3}$$

Here, i_t, f_t, c_t, o_t, h_t are the input, forget, memory, out- put and hidden state of the LSTM, respectively. The vector $z\hat{} \in R^D$ is the context vector, capturing the visual information associated with a particular input location, as ex- plained below. $E \in R^{m\times K}$ is an embedding matrix. Let m and n denote the embedding and LSTM dimensionality respectively and σ and ⊙ be the logistic sigmoid activation and element-wise multiplication respectively.

$$e_{ti} = f_{\text{att}}(\mathbf{a}_i, \mathbf{h}_{t-1}) \tag{4}$$
$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})}. \tag{5}$$

$$\hat{\mathbf{z}}_t = \phi\left(\{\mathbf{a}_i\}, \{\alpha_i\}\right), \tag{6}$$

VI. CONCLUSIONS

We propose an attention based approach that gives state of the art performance on three benchmark datasets us- ing the BLEU and METEOR metric. We also show how the learned attention can be exploited to give more interpretability into the models generation process, and demonstrate that the learned alignments correspond very well to human intuition. We hope that the results of this paper will encourage future work in using visual attention. We also expect that the modularity of the encoder-decoder approach combined with attention to have useful applications in other domains.

We introduced a model that generates natural language descriptions of image regions based on weak labels in form of a dataset of images and sentences, and with very few hard-coded assumptions. Our approach features a novel ranking model that aligned parts of visual and language modalities through a common, multimodal embedding. We showed that this model provides state of the art performance on image-sentence ranking experiments. Second, we described a Multimodal Recurrent Neural Network architecture that generates descriptions of visual data. We evaluated its performance on both full-frame and region-level experiments and showed that in both cases the Multimodal RNN outperforms retrieval baselines.

VII. REFERENCES

[1] http://cs.stanford.edu/people/karpathy/deepimagesent/

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

[3] http://karpathy.github.io/2015/05/21/rnn-effectiveness/