Image Caption Generator

Nikhil Shekhar, Sanjana Ramprasad
Department of Computer Science
University Of Buffalo
{nshekhar, srampras}@buffalo.edu

Abstract

Describing the contents of an image in well formed sentences in english is a challenging task that connects computer vision and natural language processing. This can be used to feed images to a deep system and then the captions can be generated. Automatically generating captions can find a variety of usages - Images captured by mobile phones can be captioned and read out loud to help the visually impaired. In this paper, we present a deep recurrent architecture that automatically generates brief explanations of images. We make use of the popular concept of transfer learning on a well known pre-trained model (VGG Net) to extract the image features using a deep convolution neural network. These features are then fed into a natural language processing model - Long Short-Term Memory (LSTM) to generate captions in English. The model is trained on a well known Flickr30k dataset which is of medium size in the current world of deep learning.

1 Introduction

Being able to automatically describe the content of an image using properly formed English sentences is a very challenging task. This task is significantly harder, for example, than the well-studied image classification or object recognition tasks, which have been a main focus in the computer vision community. Indeed, a description must capture not only the objects contained in an image, but it also must express how these objects relate to each other as well as their attributes and the activities they are involved in. Moreover, the above semantic knowledge has to be expressed in a natural language like English, which means that a language model is needed in addition to visual understanding.

In machine translation, an "encoder" RNN reads the source sentence and transforms it into a rich fixed-length vector representation, which in turn in used as the initial hidden state of a "decoder" RNN that generates the target sentence. We use similar concept but the encoder is replaced by a deep convolution neural network (CNN). This is because, over the last few years it has been convincingly shown that CNNs can produce a rich representation of the input image by embedding it to a fixed-length vector, such that this representation can be used for a variety of vision tasks. Hence, it is natural to use a CNN as an image "encoder", by first pre-training it for an image classification task and using the last hidden layer as an input to the RNN decoder that generates sentences.

In this paper, we apply deep learning techniques to the image caption generation task. We first extract image features using a CNN. Specifically, we extract a 4096-Dimensional image feature vector from the fc7 layer of the VGG-16 network pre-trained on ImageNet from Simonyan and Zisserman. This resulting feature vector is then fed into a vanilla RNN or a LSTM. The vanilla RNN and LSTM generate a description of the image in valid English.

2 Related Work

Most work in visual recognition has originally focused on image classification, i.e. assigning labels corresponding to a fixed number of categories to images. Great progress in image classification has been made over the last couple of years, especially with the use of deep learning techniques [2, 3]. The problem of still image description with natural text has gained interest more recently. Leveraging recent advances in recognition of objects, their attributes and locations, allows us to drive natural language generation systems, though these are limited in their expressivity. Farhadi et al. [4] use detections to infer a triplet of scene elements which is converted to text using templates. Similarly, Li et al. [5] start off with detections and piece together a final description using phrases containing detected objects and relationships. A more complex graph of detections beyond triplets is used by Kulkani et al. [6], but with template-based text generation. More powerful language models based on language parsing have been used as well [7, 8, 9, 10, 11]. The above approaches have been able to describe images "in the wild", but they are heavily hand designed and rigid when it comes to text generation.

Generating sentences that describe the content of images has already been explored. Several works attempt to solve this task by finding the image in the training set that is most similar to the test image and then returning the caption associated with the test image [12, 13]. Jia et al., Kuznetsova et al., and Li et al. find multiple similar images, and combine their captions to generate the resulting caption [14]. Kuznetsova et al., and Gupta et al. tried using a fixed sentence template in combination with object detection and feature learning. They tried to identify objects and features contained in the image, and based on the identified objects contained in the image they used their sentence template to create sentences describing the image. Nevertheless, this approach greatly limits the output variety of the model. Karpathy et al.[15] developed a Multimodal Recurrent Neural Network architecture that uses inferred alignments to learn to generate novel descriptions of image regions.

3 Technical Approach

Overview. We implemented a deep recurrent architecture that automatically produces short descriptions of images. Our models uses a CNN, which was pre trained on ImageNet, to obtain images features. We then feed these features into a LSTM network to generate a description of the image in valid English.

3.1 CNN-based Image Feature Extractor

For feature extraction, we use a CNN. CNNs have been widely used and studied for images tasks, and are currently state-of-the-art methods for object recognition and detection [16]. Concretely, for all input images, we extract features from the fc7 layer of the VGG-16 network pre-trained on ImageNet [17], which is very well tuned for object detection. We obtained a 4096-Dimensional image feature vector. We feed these features into the first layer of our LSTM at the first iteration.

3.2 LSTM-based Sentence Generator

Although RNNs have proven successful on tasks such as text generation and speech recognition, it is difficult to train them to learn long-term dynamics. This problem is likely due to the vanishing and exploding gradients problem that can result from propagating the gradients down through the many layers of the recurrent networks. LSTM networks (Figure 1) provide a solution by incorporating memory units that allow the networks to learn when to forget previous hidden states and when to update hidden states when given new information.

At each time-step, we receive an input $x_t \in R^D$ and the previous hidden state $h_{t-1} \in R^H$, the LSTM also maintains an H-dimensional cell state, so we also get the previous cell state $c_{t-1} \in R^H$. The learnable parameters of the LSTM are an input-to-hidden matrix $W_x \in R^{4HxD}$, a hidden-to-hidden matrix $W_h \in R^{4HxH}$, and a bias vector $b \in R^{4H}$.

At each time step, we compute an activation vector $a \in R^{4H}$ as

$$a = W_x x_t + W_h h_{t-1} + b$$

Then the LSTM divides a into 4 vectors $a_i, a_f, a_o, a_g \in \mathrm{R}^H$ where a_i consists of the first H elements of a, a_f is the next H elements of a, etc.. We then compute four gates which control whether to forget the current cell value $f \in \mathrm{R}^H$, if it should read its input $i \in R^H$, and whether to output the new cell value $o \in R^H$, and the block input $g \in R^H$.

$$i = \sigma(a_i)$$
$$f = \sigma(a_f)$$
$$o = \sigma(a_o)$$
$$g = tanh(a_g)$$

where σ is the sigmoid function and *tanh* is the hyperbolic tangent; both are applied element-wise.

Finally, the LSTM computes the next cell state c_t which encodes knowledge at every time step of what inputs have been observed up to this step, and the next hidden state h_t as

$$c_t = f \circ c_{t-1} + i \circ g$$
$$h_t = o \circ tanh(c_t)$$

where $\circ$ represents the Hadamard product. The inclusion of these multiplicative gates permits the regulation of information flow through the computational unit, allowing for more stable gradients and long-term sequence dependencies. Such multiplicative gates make it possible to train the LSTM robustly as these gates deal well with exploding and vanishing gradients. The non-linearities are sigmoid $\sigma()$ and hyperbolic tangent $tanh()$.

Procedure. Our LSTM model takes the image I and a sequence of inputs vectors $(x_1, ..., x_T)$. It then computes a sequence of hidden states $(h_1, ..., h_t)$ and a sequence of outputs $(y_1, ..., y_t)$ by following the recurrence relation for $t = 1$ to T:

$$b_v = W_{hi}[CNN(I)]$$
$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + 1(t = 1) \circ bv)$$
$$yt = Softmax(W_{oh}h_t + b_o)$$

where W_{hi}, W_{hx}, W_{hh}, W_{oh}, x_i, b_h and b_o are learnable parameters and $CNN(I)$ represents the image features extracted via CNN.

Training. We train our LSTM model to correctly predict the next word (y_t) based on the current word (x_t), and the previous context (h_{t-1}). We do this as follows: we set $h_0 = 0$, x_1 to the *START* vector, and the desired label y_1 as the first word in the sequence. We then set x_2 to the word vector corresponding to the first word generated by the network. Based on this first word vector and the previous context the network then predicts the second word, etc. The word vectors are generated using the *word2vec* embedding model as described by Mikolov et. al. During the last step, x_T represent the last word, and y_T is set to an *END* token**.**

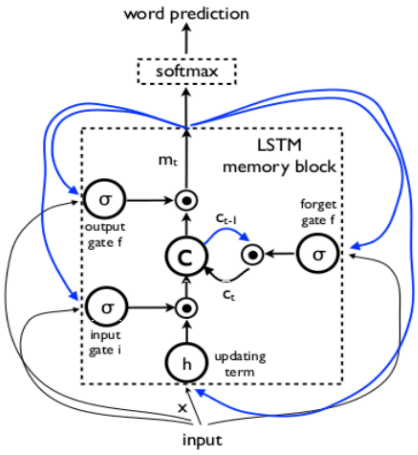
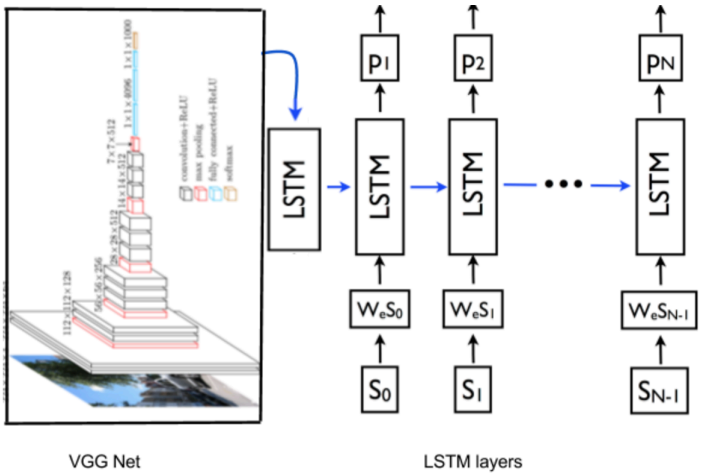


Figure 1: LSTM unit and its gates

Model. The model created and trained looks like below. The output of the extracted features from the CNN is fed into the LSTM which is modeled using tensorflow.



Testing. To predict a sentence, we obtain the image features by setting $h_0 = 0$, set x_1 to the *START* vector, and compute the distribution over the first word y_1. Accordingly, we pick the *argmax* from the distribution, set its embedding vector as x_2, and repeat the procedure until the *END* token is generated.

Cross Entropy Loss. At every time-step, we generate a score for each word in the vocabulary. We then use the ground truth words in combination with the softmax function to compute the losses and gradients. Then the sum of losses is calculated over time and are averaged over the mini-batch. Since we operate over mini-batches and because different generated sentences may have different lengths, we append PERIOD(.) token to the end of each caption so that they all have the same lengths. In addition, our loss function accepts a mask array that informs it on which elements of the scores counts towards the loss in order to prevent the PERIOD(.) tokens to count towards the loss or gradient.

Optimization. We use Adam optimizer, which is a method for efficient stochastic optimization that only requires first-order gradients and computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients, with mini-batches of 128 image - sentence pairs and a learning rate decay of 0.95. Adam's main advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradients, its step-size is approximately bounded by the step-size hyper-parameter, and it automatically performs a form of step-size annealing[18].
The Adam optimizer estimates the first moment - m_t (the mean) and the second moment -v_t (the uncentered variance) of the gradients respectively by:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g^2_t$$

Then the bias-corrected first and second moment estimates are calculated as below:

$$\hat{m}_t = m_t / (1 - \beta^t_1)$$

$$\hat{v}_t = v_t / (1 - \beta^t_2)$$

Then the Adam update rule is applied which is as follows:

$$\theta_{t+1} = \theta_t - \eta / (sqrt (\hat{v}_t) + \epsilon) * \hat{m}_t$$

4 Experiments

4.1 Dataset used to train the model

The dataset used is one of the popular medium sized dataset - Flickr30k, which has 30,000 images with 5 captions describing each image. These 5 sentence descriptions were collected with Amazon Mechanical Turk and is used as one of the standard dataset for deep learning researches related to CNN and captioning. We convert all sentences to lowercase, and discard non-alphanumeric characters. A few examples of datasets with captions available in the original dataset are as below:

Captions from Mechanical Turk:
•A black and brown dog runs along the sand , avoiding an oncoming wave
•The dog runs next to the crashing waves of the ocean .
•A dog is running parallel with the ocean waves .
•A dog running in the surf .
•a dog runs on the shore .

Captions from Mechanical Turk:

•A woman and a girl walking together on a sidewalk in a city .
•A lady and her daughter are walking down the street .
•A girl and a woman walk through a street and are about to pass a telephone booth
•Two people walking down the sidewalk by a telephone booth wearing jackets in between tall buildings
•A city street with a telephone booth , passing cars , a parked bicycle , and a blond woman with a purse and young girl with a backpack walking away

Captions from Mechanical Turk:
•A man and a young girl in a pink bathing suit float together on an inflatable tube in the water .
•This man and little girl are relaxing in an innertube on the water .
•A man and a girl sit inside a floating tube in the water .
•A man and girl float on the water in an innertube .
•A man and a small girl floating on an innertube **.**

4.2 Captions generated by the trained model

The model trained was saved by using checkpoints while training. These checkpoints (weights) can be used to generate captions. We trained the model for close to 24 hours on a 40 core CPU with 78 iterations in total. The captions generated gives out nice caption on the Flickr30k dataset and even generalizes well on the datasets which are of similar types. A few examples of captions generated are as below:

4.2.1 Captions for images present in the training dataset

Captions from LSTM model:

a black dog is running through the water .

Captions from LSTM model:

First run: a man in swimming trunks is sitting in the water with a red and blue ball in his hand .

Second run: a man in a swimming pool .

4.2.2 Captions for images not present in the training dataset

Captions from LSTM model:
a group of people are gathered around a large mural of a woman lying on the ground .

Captions from LSTM model:

First run: a white dog is jumping over a snow covered pond .

Second run: a dog jumps to catch a ball in the snow .

5 Conclusion:

We have presented a deep learning model that automatically generates image captions. Our described model is based on a CNN that encodes an image into a compact representation, followed by a LSTM that generates corresponding sentences based on the learned image features. Future work can include this text-to-speech technology, so that the generated descriptions are automatically read out loud to visually impaired people. It can also include capturing images from a smartphone and captioning it which can be read out loud in real time by the phone.

Acknowledgments

We would like to thank Professor Sagur Srihari and Junfei Wang for their ongoing support.

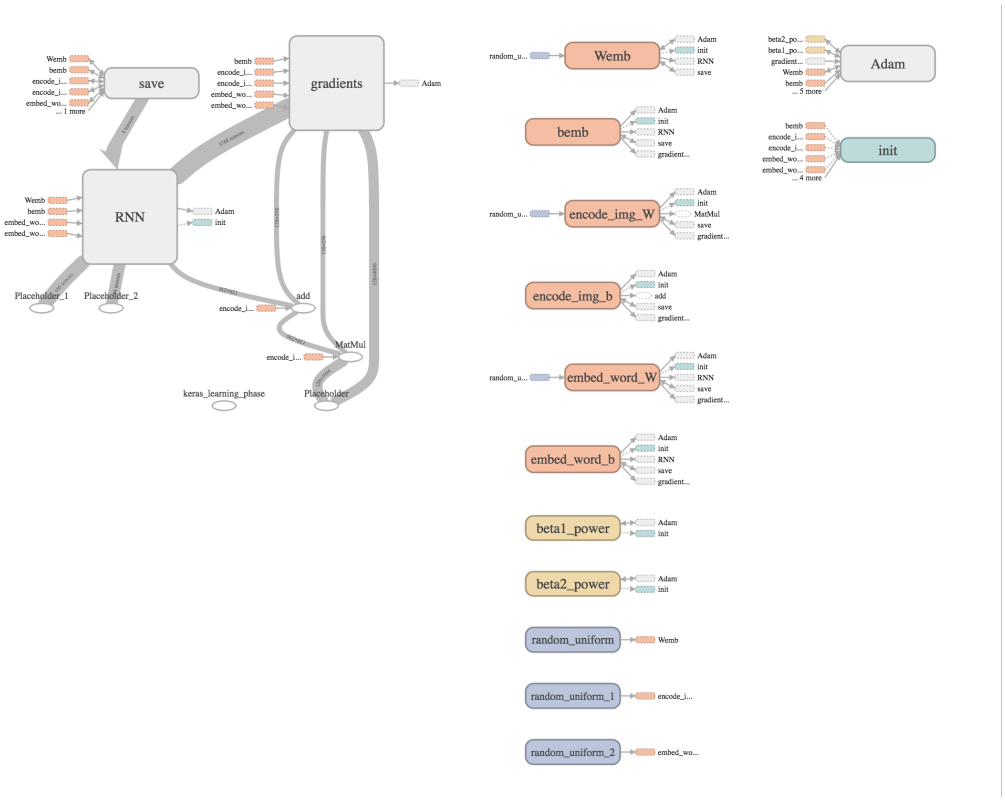
References

[1] Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan. ”Show and Tell: A Neural Image Caption Generator.” 2*015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015). Web.25 May 2016
[2] Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang,Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ”ImageNet Large Scale Visual Recognition Challenge.” International Journal of Computer Vision Int J Comput Vis 115.3 (2015):211-52. Web. 19 Apr. 2016
[3] Everingham, Mark, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. ”The Pascal Visual Object Classes (VOC) Challenge.” International Journal of Computer Vision Int J Comput Vis 88.2 (2009): 303-38. Web. 22 May 2016
[4] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young,C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. In ECCV, 2010.
[5] S. Li, G. Kulkarni, T. L. Berg, A. C. Berg, and Y. Choi. Composing simple image descriptions using web-scale n-grams. In Conference on Computational Natural Language Learning,2011.
[6] Li, Siming and Kulkarni, Girish and Berg, Tamara L. and Berg, Alexander C. and Choi, Yejin. ”Composing simple image descriptions using web-scale n-grams.” Proceedings of the Fifteenth Conference on Computational Natural Language Learning: 220-228 (2011). Web. 27 Apr. 2016
[7] M. Mitchell, X. Han, J. Dodge, A. Mensch, A. Goyal, A. C.Berg, K. Yamaguchi, T. L. Berg, K. Stratos, and H. D. III.Midge: Generating image descriptions from computer vision detections. In EACL, 2012
[8] A. Aker and R. Gaizauskas. Generating image descriptions using dependency relational patterns. In ACL, 2010.
[9] P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi. Collective generation of natural image descriptions. In ACL, 2012.
[10] P. Kuznetsova, V. Ordonez, T. Berg, and Y. Choi. Treetalk: Composition and compression of trees for image descriptions.ACL, 2(10), 2014.
[11] D. Elliott and F. Keller. Image description using visual dependency representations. In EMNLP, 2013.
[12] Farhadi, Ali, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. ”Every Picture Tells a Story: Generating Sentences from Images.” Computer Vision ECCV 2010 Lecture Notes in Computer Science (2010): 15-29. Web. 5 Apr. 2016
[13] Lazaridou, Angeliki, Nghia The Pham, and Marco Baroni. ”Combining Language and Vision with a Multimodal Skip-gram Model.” Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2015). Web. 23 May 2016
[14] Kuznetsova, Polina, Vicente Ordonez, Alexander C. Berg, Tamara Berg, and Yejin Choi. ”Collective generation of natural image descriptions.” Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics 1 (2012): 359:368. Web. 30 Apr. 2016
[15] Karpathy, Andrej, and Li Fei-Fei. ”Deep Visual-semantic Alignments for Generating Image Descriptions.”2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015). Web. 29 May 2016
[16] Krizhevsky, Sutskever, and Hinton. ”Imagenet classification with deep convolutional neural networks.”NIPS (2012). Web. 28 Apr. 2016
[17] Simonyan, Karen and Andrew Zisserman. ”Very deep convolutional networks for large-scale image recognition.”CoRR (2014). Web. 28 May 2016
[18] Kingma, Diederik and Jimmy Ba. ”Adam: A method for stochastic optimization.” CoRR (2015). Web. 19 May 2016

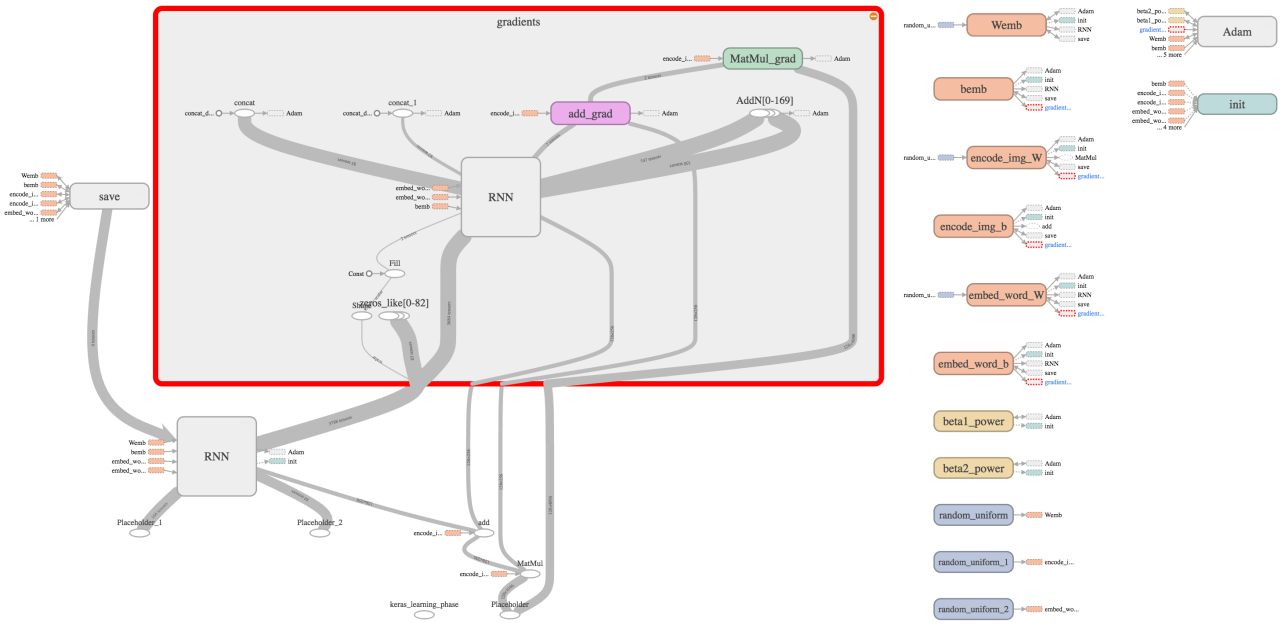
Appendix

TensorFlow graphs

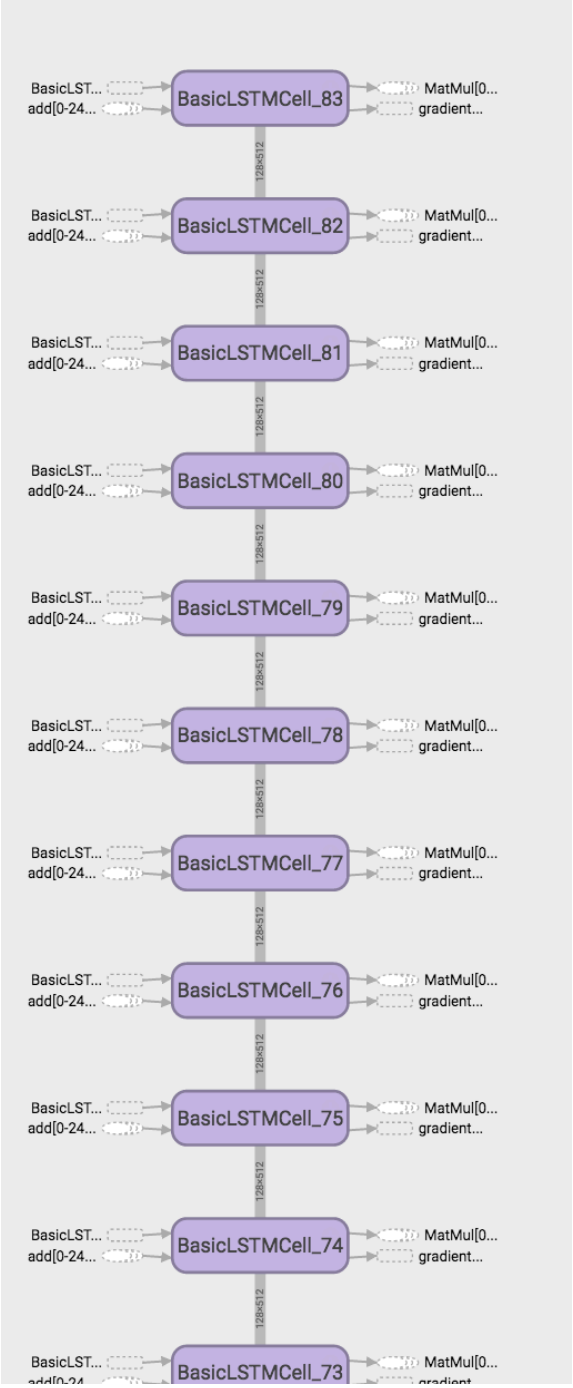
1. Model graph - This shows the complete high level graph of all the modules present in the model. Each of the modules, when unfolded shows up hundreds of sub-graphs of its components. A few of them will be described and shown below.



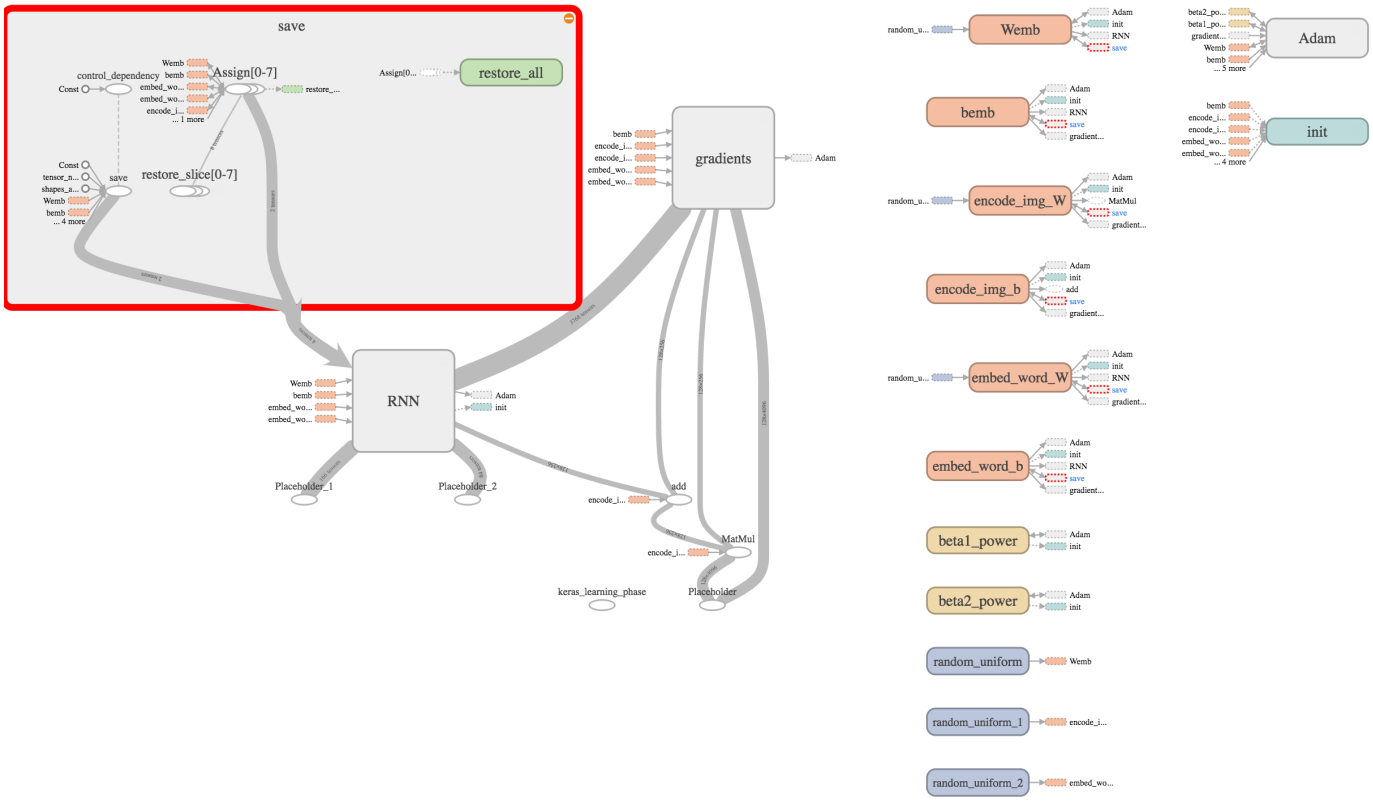
2. Gradients module unfolded - Makes use of Adam optimizer to minimize the cross-entropy loss function



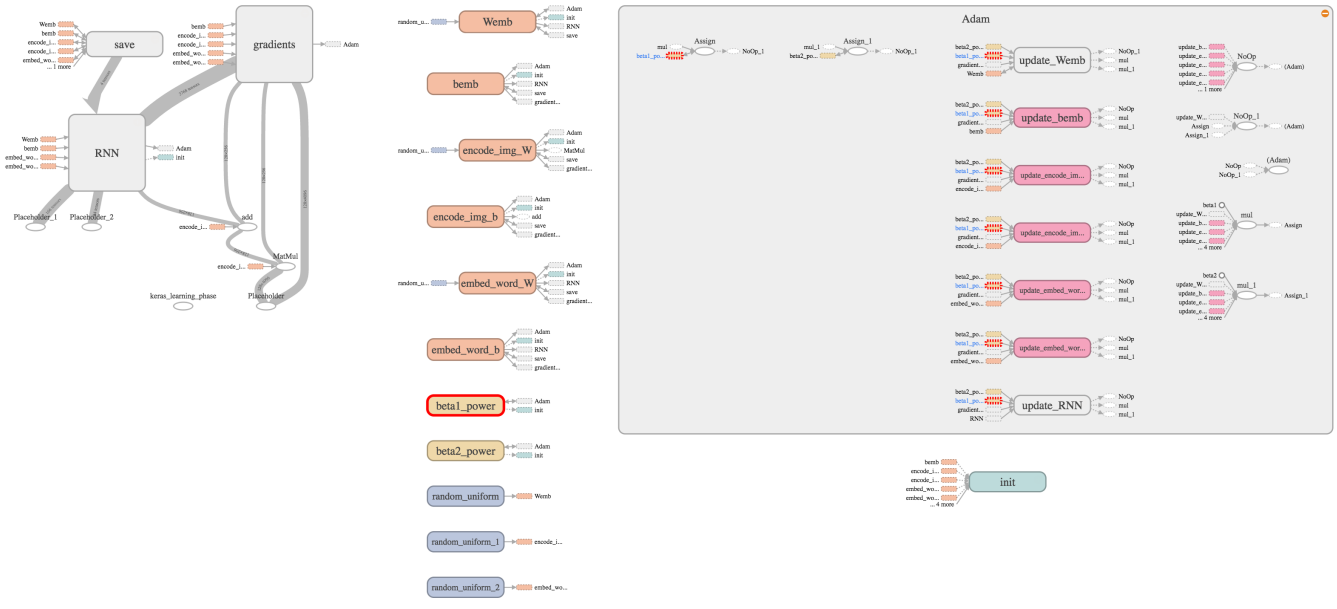
3. RNN module unfolded - The model has 84 LSTM cells which is equivalent to the max length of any individual caption present in the training data. The number of LSTM cells are chosen dynamically in accordance to the training data and the model is created.



4. Save module unfolded - This is used to save and restore the model checkpoints



5. Adam optimizer - Shows the nodes in the graph where it is used in the model generated



6. Unfolded view of a few other modules from the Graph