Show and Tell: A Neural Image Caption Generator*

Report† **by:** Akhil Gupta (akhilg3), Gregory Romanchek (romanch2),
Heba Flemban (flemban2), Moitreya Chatterjee (mc12),
Yan Zhang (yanz4), Zihao Yang (zihaoy3)

December 15, 2019

1 Overview

Image captioning has seen a lot of traction in the last few years - it entails automatically describing the contents of an image in natural language. This area links a number of fields within the broad area of artificial intelligence, including: computer vision, natural language processing, machine learning, etc. In this report, we re-implement work of [Vinyals et al., 2015], to build a generative model based on deep recurrent network architecture. This model takes an image as input and generates a sentence that describe the events happening in that image. The performance of the approach has been discussed both qualitatively and quantitatively. Aided by a superior image representation, we are able to match or exceed the performance reported by [Vinyals et al., 2015] on the challenging MS-COCO dataset (2014) [Lin et al., 2014]. We also discuss some extensions to this technique that were explored and highlight some others that could be used to further improve the system in the future. We implemented our system in Python 3 and make our implementation publicly available[1].

2 Introduction

When dealing with image captioning, it's important to express how different objects relate to each other in addition to just capturing the objects. To express this semantic knowledge, a natural language, such as English, is needed - amplifying the role of a language model. Most previous works had focused on gluing together two different algorithms for going from image to description - but the authors propose a single joint model.

The main motivation came from the advancements in machine translation - which was also achieved via a series of separate tasks. As recurrent neural network (RNN) methodology matured and popularized, they changed the paradigm in that domain. Here, the authors follow the same elegant idea but replace the encoder RNN (of machine translation) with a deep CNN. So, a CNN is used as an image encoder by first pre-training it for an image classification task. The representation obtained from the last layer of which is used as an input to the RNN decoder to generate a sentence. From now on, we will refer to this joint model as Neural Image Caption (NIC).

*Based on the research work presented at CVPR 2015 [Vinyals et al., 2015].

†Part of **IE-534** course at the University of Illinois, Urbana-Champaign. Special thanks to **Prof. Justin Sirignano**.

[1]Code available at https://github.com/guptakhil12/show-tell.

This captioning model has various use-cases in the real-world, from augmenting blind individuals by working as a guide in their ear piece, or informing them about their immediate surroundings. This report has been structured in the following order: we discuss related work in Section 3 and cover the joint model NIC in detail (in Section 4). MSCOCO specifics have been detailed in Section 5. Section 6 talks about the training methods, and the experiments have been covered in Section 7. Future directions have been mentioned in Section 8.

3 Related Work

A lot of previous works have dwelled on generating a system which provides an audio output description of an input image. Some of the attempts have involved models which have used visual primitive recognizers that are then merged with certain algorithms such as *And-Or* graphs or other such logical algorithms. These systems had limited domain and have failed to generalize to a large swath of real-world tasks. [Gerber and Nagel, 1996] have also worked in ranking these images by combining them with their textual information in the same vector space.

On the other hand, the model reviewed and reproduced in this report is using a convoluted neural network (CNN), originally intended for the classification of the provided images and then a recurrent neural network is used for the sequence modeling of the data provided by the CNN. Both these models are combined in order to generate a model that creates a decent description of these images [Vinyals et al., 2015].

4 Model

Recent studies have proven that by using sequence models, one can increase the probability of its maximum generated description accuracy, given an input image - this model is used for both training and inference purposes. The training (learning) task on this model may be represented using the following equation:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{(a,S)} \log p(S|a:\theta) \tag{1}$$

Here, we have used the assumption that θ are the parameters to be used for training this model; a represents the feature extracted from the image being provided as an input; and S is the description of the image being provided as an input. Since S has no finite length, we have used the chain rule for modeling process of our probability and limiting the length of a sentence to a maximum number of words. This is captured by the following equation, where N refers to the length of a sentence being generated.

$$logp(S|a) = \sum_{t=0}^{N} \log p(S_t|a, S_0, ..., S_{t-1}) \tag{2}$$

The NIC model essentially features an encoder-decoder architecture that takes as input an image, encodes it, and then decodes it into a sentence.

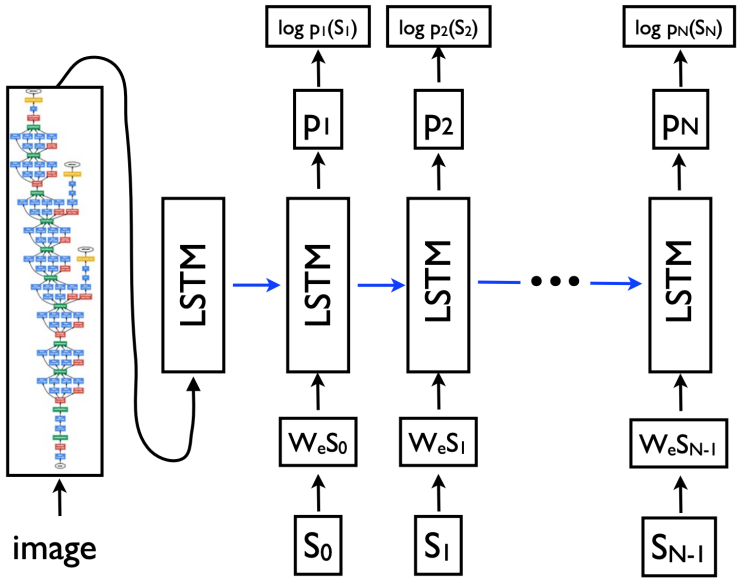


Figure 1: Model Architecture.

4.1 Image Encoder

The image encoding may be represented as follows:

$$x_{img} = f_{img}(a)$$

where $f_{img}(\cdot)$ is any function but in our case is a deep neural network (CNN).

4.2 LSTM-based Sentence Generator

The encoded image is fed as input to a decoder, which is a Recurrent Neural Network (RNN) in our case. This module generates the sentence description for this image.

Here, we select the RNN function based on its ability to compensate the declining values of gradient. Most of the researchers found it difficult to design such a function, so we will be using Long-Short Tem Memory (LSTM) or a Gated Recurrent Unit (GRU) [Chung et al., 2014] to design such a recurrent neural network for our model.

LSTM models basically focus on the memory cell c; the function of this memory cell is to encode all the information which has been provided to the model up until the given stage. This encoding cell can be explained as multi-leveled layers which are controlled by a number of gates. These gates allow information if they are at 1 and blocks information if they are at 0. They also permit partial transmission of the input, if the gate values are somewhere between 0 and 1.

Primarily, three different gates are being used to control these values. First is the forget gate denoted by f; this is to control that all the information is to be forgotten or not. The other two are the input gate i and the output gate o. These gates are represented mathematically as follows:

$$
\begin{aligned}
i_t &= \sigma(W_{ix}x_t + W_{im}m_{t-1}) \\
f_t &= \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \\
o_t &= \sigma(W_{ox}x_t + W_{om}m_{t-1}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \\
m_t &= o_t \odot c_t \\
p_{t+1} &= Softmax(m_t)
\end{aligned}
$$

This LSTM model is trained in such a way that such that it predicts the most likely word at every step given the words that it has already predicted and the image representation . This can be seen in Figure 1.

5 Dataset

For the purpose of building and testing the model, we have used the MSCOCO dataset [Lin et al., 2014]. This dataset contains a number of images and sentences (5 captions) describing these images. Statistical information about the dataset is given in Table 1 as below.

Table 1: Stats of COCO dataset.

Dataset Type	Records
Training	82783
Validation	40504
Testing	40775

When pre-processing the data, we built a vocabulary of words that appeared more than 5 times in all the captions. Then, each caption was tokenized using the vocabulary, with words not in the vocabulary being replaced by an unknown tag. To prepare the images as inputs of our CNN architecture, pretrained ResNet models, we resized the images to 224×224.

6 Training Methods

We applied data augmentation on the resized image inputs, including random horizontal flip, random vertical flip, and z-normalization. For the encoding CNN architecture, we tried ResNet models that were pretrained on ImageNet dataset, and only tuned the last two layers during training. For the decoding RNN architecture, we used a Gated Recurrent Unit (GRU) [Chung et al., 2014] in our implementation, due to its ease of convergence. Results with an LSTM model are presented for comparison, as well. We tried different embedding dimensions for the words. We also tried two different optimization algorithms: stochastic gradient descent (SGD) and adaptive moment estimation (Adam), with learning rate ranging from 0.0001 to 0.01 on a log scale. We additionally, trained an attention model [Kelvin Xu, 2016] to probe improvements over this architecture, details follow in Section 7.2 Furthermore, cross-entropy loss was used as the objective function for training our model.

Table 2: Results on the MS-COCO Validation dataset using GRU.

Method			B1	B2	B3	B4	CIDEr	ROUGE	CE	Hours
Random Baseline			-	-	-	4.6	5.1	-	-	-
[Vinyals et al., 2015]			71.3	54.2	40.7	27.7	85.5	53.0	-	-
[Git, b]			64.6	45.9	31.7	22.0	69.4	47.6	-	-
Embedding	**LR**	**Optim.**								
256	0.01	SGD	63.7	44.72	35.97	30.77	86.77	34.24	2.64	28.6
256	0.01	SGD	66.72	47.23	37.86	32.17	87.08	35.18	2.27	28.61
512	0.01	SGD	63.99	44.88	36.11	30.87	86.17	33.55	2.68	28.61
512	**0.001**	**SGD**	**67.46**	**47.56**	**38.03**	**32.30**	**89.06**	**35.84**	**2.27**	**28.61**
512	0.0001	SGD	58	36.7	30.58	26.89	55.01	31.84	3.11	28.61
512	0.001	Adam	66.44	46.41	36.93	31.31	88.00	35.77	2.41	28.61
512	0.0001	Adam	62.75	43.80	35.32	30.28	84.99	33.68	3.13	28.61

Table 3: Results on the MS-COCO Validation dataset using Architectural Variations.

Architecture			B1	B2	B3	B4	CIDEr	ROUGE	CE	Hours
LSTM										
Embedding	**LR**	**Optim.**								
512	0.0001	Adam	62.4	43.81	35.36	30.33	84.79	33.4	3.59	28.61
CNN : ResNet 50										
Embedding	**LR**	**Optim.**								
512	0.0001	Adam	62.57	43.59	35.19	30.17	83.95	33.71	3.15	28.61

7 Experiments

In order to evaluate the efficacy of the captioning technique [Vinyals et al., 2015], we conduct experiments on the challenging, publicly available Microsoft COCO dataset [Lin et al., 2014]. We used the 2014 version of the dataset for our experiments. We trained our model on the "training" split of the dataset and assess its performance on the "validation" split of the dataset. We evaluated the performance using popular language-translation metrics, such as the BLEU (B1, B2, B3, and B4) [Papineni et al., 2002], CIDEr [Vedantam et al., 2015], ROUGE-L [Lin and Och, 2004], and cross-entropy error (CE). We compared performance with different hyper-parameter choices against the performance of the model trained in the original work [Vinyals et al., 2015, Git, b], and also against a baseline method which predicts the words randomly for a given image. Additionally, we also qualitatively visualize some of the captions generated by the best performing model.

7.1 Results and Discussion

Table 2 shows the performance of the captioning method under different choices of hyper-parameters. The results evince the effectiveness of using 512-dimensional embedding space for the recurrent models over 256 dimensions. Additionally, we also notice that a Stochastic Gradient Descent (SGD) optimizer

initialized with 0.001 learning rate outperforms the models trained with the more recently proposed Adam optimizer [Kingma and Ba, 2014] across all metrics (indicated in bold font). Further, the models superiority of performance over the random baseline is a testament to its generalizability on the challenging MS-COCO dataset. MS-COCO being a substantially large dataset, it takes slightly over a day to train a model for 100 epochs on the state-of-the-art NVIDIA RTX 2080 GPU.

Table 2 shows that when compared to the original performance reported in the paper, our implementation slightly outperforms it. We surmise that this is due to the superior representation of the image obtained by the use of an improved CNN (ResNet-101) [He et al., 2016], over the one used in the original work [Krizhevsky et al., 2012]. Next, we discuss some of the architectural variants that we explored.

7.2 Architectural Variations

LSTM based Model: We also experimented by substituting the GRU unit in the model presented above with a LSTM unit of 5 layers. Table 3 shows the results of this variant of the model. We see that it slightly under performs the GRU based model. Qualitative results for this model are shown in Figure 3.

ResNet-50 CNN: Image encoders are critical to the success of captioning models, however deep neural nets are resource intensive. Hence, we explore the ResNet-50 network architecture [He et al., 2016], as an image-encoder. This network while being better in performance than the AlexNet [Krizhevsky et al., 2012], is much more light weight than the latter. Table 3 shows that this encoder permits us to almost perform at par with the our core model. Qualitative results for this model are shown in Figure 3.

Attention-based Architecture: In order to selectively dwell on the most informative regions of the image that we intend to describe, we resort to an *Attention Model* [Kelvin Xu, 2016].

Generically, we want to generate a caption y is generated from a single raw image a. Each word of the caption is of 1-of-K words:

$$y = (y_1, ..., y_C), y_i \in \mathbb{R}^K$$

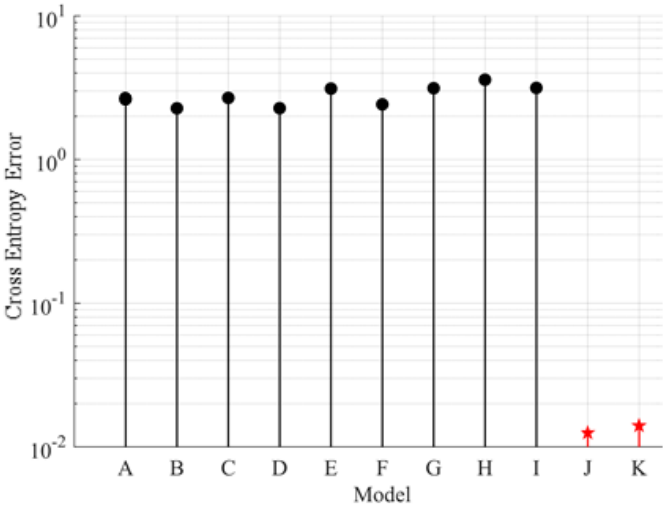
where K is the size of the vocabulary and C is the length of the caption.

Image features are extracted from L different regions of an image using a CNN. Each feature vector is a D-dimensional representation corresponding to a part of the image.

$$a = (a_1, ..., a_L),\ a_i \in \mathbb{R}^D$$

An LSTM generates the caption one word at a time. This word (at time-step t) is conditioned on the image context vector z^t, the previous hidden state of the LSTM, h_{t-1} and the previously generated words. The context vector is computed from the feature vectors a_i as follows:

$$e_{ti} = f_{\text{att}}(a_i, h_{t-1})$$



	CNN	RNN	LR	Optim.	Embed.	Attn.
A	ResNet101	GRU	0.01	SGD	256	-
B	ResNet101	GRU	0.001	SGD	256	-
C	ResNet101	GRU	0.01	SGD	512	-
D	ResNet101	GRU	0.001	SGD	512	-
E	ResNet101	GRU	0.0001	SGD	512	-
F	ResNet101	GRU	0.001	Adam	512	-
G	ResNet101	GRU	0.0001	Adam	512	-
H	ResNet101	LSTM	0.0001	Adam	512	-
I	ResNet50	GRU	0.0001	Adam	512	-
J ★	ResNet101	GRU	0.0001	Adam	512	Yes
K ★	ResNet101	LSTM	0.0001	Adam	512	Yes

Figure 2: Cross Entropy Performance of the listed models on the MSCOCO validation set.

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{L} exp(e_{tk})} \tag{3}$$

$$z^t = \varphi(a_i, \alpha_i) \tag{4}$$

where f_{att} is a multi-layer perceptron, α_{ti} represents the weight on the i^{th} region of the image at step t, ϕ is an aggregating function that outputs a single vector given the set of image feature vectors and their corresponding weights.

We instantiate ϕ as the weighted average of the image feature vectors as follows:

$$z_t = \sum_{i=1}^{L} \alpha_{t,i} a_i \tag{5}$$

This model is differentiable and smooth and is thus backpropable.

The performance of this model vis-á-vis the other models is shown in Figure 2. The plot clearly showcases the dominance of the attention-based method over the others, by around two orders of magnitude.

7.3 Qualitative Results

Figure 3 shows sample captions for the GRU based and LSTM models, with both ResNet-101 and ResNet-50 encoders. The generated captions look realistic and showcases linguistic fluency, although some aspects of the description do not align well with the image. For more qualitative results, we refer the reader to the Appendix.

Model					Caption
Target					a car that seems to be parked behind a parked car.
CNN	RNN	LR	Optim.	Embed.	
ResNet101	GRU	0.001	SGD	512	a car parked on the side of the road.
ResNet101	LSTM	0.0001	Adam	512	a car parked on the side of a street next to a traffic light.
ResNet50	GRU	0.0001	Adam	512	a man riding a bike with a dog carrier attached to the back.

Figure 3: Captioning results for a random image in the validation set.

7.4 Code Description

We implemented the core modules ourselves, while drawing inspiration from some publicly available repositories on github [Git, a, Git, b]. Further, the attention module was entirely written by ourselves. For details regarding the execution instruction and code documentation, we refer the reader to our publicly available code repository.

8 Future Directions

In terms of the generation of output sequences, one of the most straight-forward way is conventional greedy algorithm (argmax), which selects each token with the highest probability. Such methods often lead to sub-optimal output sequences. Beam Search, on the other hand, is a more accurate and efficient way to break this sub-optimality. It adopts breadth-first search to build its search tree, but only keeps top N (beam width) nodes at each level in memory, i.e. only a limited number of best partial solutions are kept as candidates. The subsequent level will then be expanded only from these top candidates. Larger beam widths would lead to better performance because the multiple candidate sequences increase the probability of matching a target sequence. As a trade-off, this increased performance comes with decrease in decoding speed. This method presents an interesting direction for future work.

Furthermore, we also intend to evaluate the attention-based model using the standard language translation metrics, i.e. BLEU, CIDEr, and ROUGE.

References

[Git, a] Image captioning. `https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning`. Accessed: 2019-12-15.

[Git, b] Show and tell: Neural image caption generator. `https://github.com/kaurrachneet6/ShowAndTell-neural-captioning`. Accessed: 2019-12-15.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[Gerber and Nagel, 1996] Gerber, R. and Nagel, H.-H. (1996). Knowledge representation for the generation of quantified natural language descriptions of vehicle traffic in image sequences. In *IEEE*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Kelvin Xu, 2016] Kelvin Xu, Jimmy Lei Ba, R. K. K. C. A. C. R. S. R. S. Z. Y. B. (2016). Show, attend and tell: Neural image caption generation with visual attention. *arXiv:1502.03044v3 [cs.LG]*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Lin and Och, 2004] Lin, C.-Y. and Och, F. J. (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 605. Association for Computational Linguistics.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*.

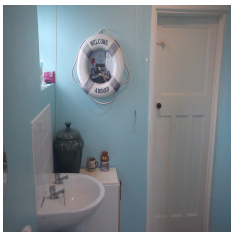
[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

[Vedantam et al., 2015] Vedantam, R., Lawrence Zitnick, C., and Parikh, D. (2015). Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164.

Appendices

Appendix A Additional Qualitative Results



Model					Caption
Target					a room with blue walls and a white sink and door .
CNN	**RNN**	**LR**	**Optim.**	**Embed.**	
ResNet101	GRU	0.001	SGD	512	a bathroom with a sink , mirror and a toilet.
ResNet101	LSTM	0.0001	Adam	512	a bathroom with a toilet and a sink.
ResNet50	GRU	0.0001	Adam	512	a bathroom with a toilet , sink and mirror.

Model					Caption
Target					a bicycle replica with a clock as the front wheel.
CNN	**RNN**	**LR**	**Optim.**	**Embed.**	
ResNet101	GRU	0.001	SGD	512	a black and white photo of a clock and a clock.
ResNet101	LSTM	0.0001	Adam	512	a clock with a poem attached to a wall with.
ResNet50	GRU	0.0001	Adam	512	a clock with a plant in front of it.

Model					Caption
Target					a black honda motorcycle parked in front of a garage.
CNN	**RNN**	**LR**	**Optim.**	**Embed.**	
ResNet101	GRU	0.001	SGD	512	a motorcycle parked in a parking lot near a fence.
ResNet101	LSTM	0.0001	Adam	512	a motorcycle parked on top of a lush green field.
ResNet50	GRU	0.0001	Adam	512	a motorcycle is parked next to a car.

Model					Caption
Target					a large passenger airplane flying through the air.
CNN	**RNN**	**LR**	**Optim.**	**Embed.**	
ResNet101	GRU	0.001	SGD	512	a large jetliner flying through a cloudy sky.
ResNet101	LSTM	0.0001	Adam	512	a small white and red plane flying in the air.
ResNet50	GRU	0.0001	Adam	512	a large air plane flying in the air.

Figure 4: Captioning results for 4 additional random images in the validation set.