CNN-LSTM based framework for Automated Image Captioning

Mohith Damarapati
New York University
md4289@nyu.edu

Abstract

Describing an image is an easy and obvious task for humans. Computers find it challenging to do the same as they require to understand the content of an image and to describe that understanding in a natural language. The problem of making machines to automatically caption an image is referred as "Image Captioning". I present a CNN-LSTM based framework to solve this problem. This model achieves a BLEU-4 score of 20.98 points on MSCOCO dataset which trails human base line by 0.72 BLEU-4 points. Experiments show that the captions generated are mostly sensible and human understandable.

1. Introduction

Humans can give rich descriptions of a visual scene. However, machines don't have that ability as they lack commonsense and intelligence. Thus, "Image Captioning" is considered as a challenging research problem connecting the fields computer vision and natural language processing. Describing an image is important because it is one of the crucial components of an artificial intelligent machine and it has practical benefits like aiding visually impaired.

The central idea to solve this problem is to extract features of the input image and use these features to generate a human understandable text. CNNs provide a rich representation of the features which can be used for various vision tasks like classification, localization and detection [1]. These features contain important information about objects present in the image. We also call the process of extraction of rich image features as learning disentangled representations [2]. These disentangled representations are given as input to sequence models like RNN.

In this project, I used ResNet-152 [3], which is pre-trained on the ImageNet [8] data to capture image features. Features in the last layer of ResNet-101 after removing the fully connected layer are given as inputs to an LSTM [4], which is a modified version of RNN. LSTMs have an ability to capture long term dependencies which is crucial while generating natural language. Figure 1 depicts framework of our model.

2. Related Work

Image captioning problem is solved by two approaches – Bottom up and Top Down. In bottom up approaches like [5], objects are first detected and then are combined to generate a caption. In top down approaches like [6], image features are extracted and then decoded by sequence models like RNNs. Most of the state-of-art methods follow top down approach.

The architecture described in [7] is based on a combination of CNNs over image regions and bidirectional RNNs over captions. Here CNN is pre-trained on ImageNet [8] and finetuned on the 200 classes of the ImageNet detection challenge. Every image is brought to a multimodal embedding space of size h and are represented as a set of h-dimensional vectors. Words in the captions are also represented in the same h-dimensional space. Then, Bidirectional RNNs are used to compute word representations. These BRNNs take a sequence of words and transform them into an h-dimensional vector. Their model obtained a BLEU4 score of 23 on MSCOCO dataset.

The model in [6] uses CNNs for obtaining image representations and LSTMs to generate sentence. LSTM model is trained to predict each word in a sentence based on image representations and preceding words. Words are sampled based on Beam Search. Instead of considering best word as the next word, we compute 'b' best words at every level and consider the ones that maximizes the probability. Here, 'b' is called beam width. Their model obtained a BLEU4 score of 27.2 on MSCOCO dataset. My model directly extends this architecture.

The work described in [9] is an extension to [6] with a powerful mechanism called attention. Attention is implemented by adding an additional gate to LSTM architecture. It lets the algorithm to attend to relevant parts of an image while predicting words. This paper describes two types of attention named Hard and Soft Attention. Soft attention and hard attention models obtained BLEU4 scores of 24.3 and 25 respectively on MSCOCO dataset.

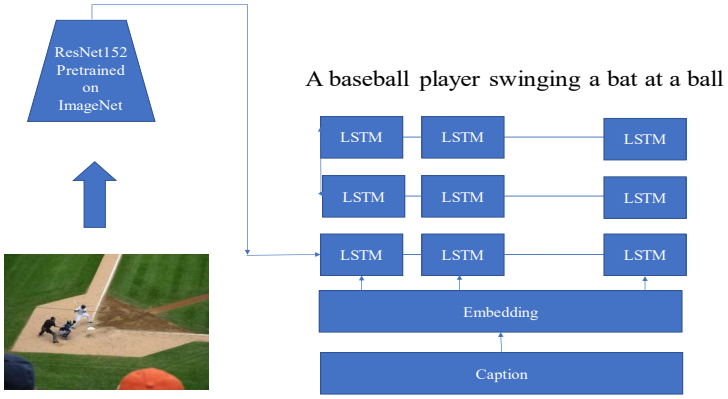


Figure 1: My CNN-LSTM architecture modelled similar to NIC architecture [6]

3. Approach

I adopted a top-down approach with an encoder-decoder framework. ResNet-152 which is pretrained on ImageNet is used as encoder and LSTM is used as decoder.

3.1. Deep Residual Network as Encoder

Deep ConvNets improved performance of image classification models. Features can be enriched as we go deeper. But very deep models can lead to problems like exploding/vanishing gradients and overfitting. To avoid these problems, 'deep residual learning' framework [3] is followed. Figure 2 shows the basic building block of a residual network. These networks have shortcut connections which skip one or more layers.

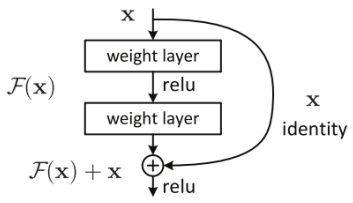


Figure 2: A basic building block of Residual Networks [3]

I used ResNet-152, a 152-layer Residual Network which has best performance on ImageNet (top-1 error: 21.43 and top-5 error: 5.71) as the encoder. It is pre-trained on ImageNet and the last soft-max layer is dropped.

3.2. LSTM as Decoder

LSTM is a variant of an RNN. They are good at maintaining long term dependencies and hence they perform well as sequence models. A brief description of a LSTM cell and related equations are explained below.

LSTMs are first introduced by [4] in 1991. Gates are the major components of an LSTM which are Input gate - $i(t)$, forget gate – $f\ (t)$, output gate – $o\ (t)$, $g\ (t)$ is input modulation gate and $c\ (t)$ is memory cell. At each time step, an LSTM cell takes $x(t)$, $h(t-1)$ and $c(t-1)$ as inputs and outputs $h(t)$ and $c(t)$. Working of an LSTM cell is described using sequence of equations below.

$$
\begin{aligned}
i\ (t) &= \sigma\ (W_{xi}\ x\ (t) + W_{hi}\ h\ (t-1) + b_i \\
f\ (t) &= \sigma\ (W_{xf}\ x\ (t) + W_{hf}\ h\ (t-1) + b_f \\
o\ (t) &= \sigma\ (W_{xo}\ x\ (t) + W_{ho}\ h\ (t-1) + b_o \\
g\ (t) &= \phi\ (W_{xc}\ x\ (t) + W_{hc}\ h\ (t-1) + b_c
\end{aligned}
$$

$$
\begin{aligned}
c\ (t) &= f\ (t) \bullet c\ (t-1) + i\ (t) \bullet g(t) \\
h\ (t) &= o\ (t) \bullet \phi\ (c\ (t))
\end{aligned}
$$

Here $\sigma\ (x) = \frac{1}{1+e^{-x}}$ (sigmoid activation) and $\phi\ (x) = \tanh x$ (tanh activation)

Memory gate enables LSTM to learn complex long-term temporal dependencies. Additional depth can be added by stacking these on top of each other.

3.3. Word Vectors and Minimum Word Frequency

First step before training is to create a vocabulary set V. Vocabulary set is created in such a way that the words with word frequency less than a minimum threshold are removed. Threshold has to be selected carefully as it can affect the performance of our model.

After creating vocabulary set, I mapped words with integer values. We call the mapped words as embeddings. These word embeddings are given as inputs to LSTM.

We will also have four additional words in our dataset - <s>, <e>, <u>, <pad>. <s> is the start symbol of a caption and <e> is the end symbol which when generated finishes the caption generation. If our model encounters a new word, then it is encoded as <u>. <pad> words are added after <e> to balance the size of captions.

4. Experiments

I coded up the entire project from scratch except the model architecture which I took it from [11], similar to NIC.

4.1. Dataset description

I used Microsoft Common Object in Context (MSCOCO) dataset. It has a total of 123375 images which is split into a train set of 82783 images and a test set of 40592 images. Each image has 5 reference captions on an average. So, there are around 600000 annotations stored in Train and Test Json files. Lengths of the captions also vary between 7 to 57. Figure 1 below shows the distribution of caption lengths with respect to their frequency in train set.

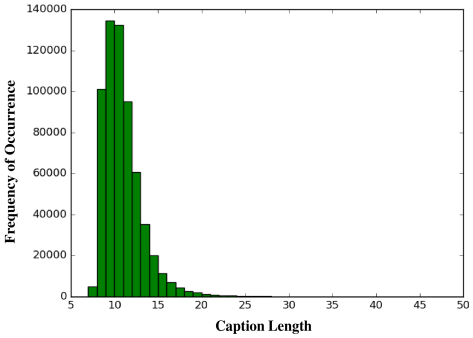


Figure 3: Visualization of Caption Lengths in MSCOCO Train Set

One more insight into dataset is to check Top-5-word occurrences:

i. a – 684577
ii. . – 310919
iii. on – 150675
iv. of – 142760
v. the – 137981

Figure 4 shows a sample image in COCO dataset with its reference captions.

REFERENCE CAPTIONS
1. Candles, pumpkins and skull decorations on table
2. A table topped with Halloween decorations and food
3. A table decorated with skulls, apples and oranges

Figure 4: Sample image with three reference captions

Word frequencies play a role while building vocabulary. I dropped words whose frequency is less than a threshold, to improve performance of the model.

4.2. Data Extraction and Vocabulary Builder

MSCOCO is a huge dataset. PyCocotools API [16] built by Microsoft have provided methods to handle dataset with ease. A COCO object is created, and all the annotations are extracted from the train Json file. These annotations are converted into string and are tokenized using NLTK [10]. I also kept track of frequency of occurrence for each word. This is important because I removed words whose frequency is less than a threshold value T. I tried different values of T and T = 4 seems to work the best (detailed in Section 5). Vocabulary set is built after removing the words whose frequency is less than 4. Start <s>, End <e>, Unknown <u> and Padding <pad> are also added into vocabulary set.

4.3. Training

ResNet-152 model, pre-trained on ImageNet is available in the torchvision package of PyTorch [12]. PyTorch's torch.nn and torchn.nn.utils.rnn provided amazing methods to build encoder and decoder classes. The last fully connected layer of ResNet-152 is removed and batch-norm layer with momentum 0.01 is added.

Decoder LSTM model contains 512 hidden states and I investigated its performance for one to four LSTM layers. Having more than two layers gave me

similar results, but performance is much improved when compared to one layer (detailed in Section 5). I experimented with both greedy search and beam search (beam width = 3) techniques to predict words in the decoder phase. Batch-size is set to 64 and learning rate to 0.001. I used cross entropy loss as loss criterion and Adam as optimizer.

4.4. Evaluation Metric

The best way to evaluate the quality of a generated caption is to manually rate them. But it is tedious for humans to evaluate such a huge volume of data. Bilingual Evaluation Understudy (BLEU) metric [13] is one of the simplest techniques to evaluate the quality of machine generated captions and is widely used. BLEU was originally designed to rate the quality of machine translations. I evaluated my model using four types of BLEU – BLEU1, BLEU2, BLEU3 and BLEU4, although BLEU4 is used in many papers. In BLEU1, we consider individual 1-gram scores alone, in BLEU2, we consider both individual 1-gram and 2-gram scores by calculating weighted geometric mean. Likewise, BLEU4 takes weighted geometric mean of 1-gram, 2-gram, 3-gram and 4-gram scores. Weights for each are 0.25. So, we consider 0.25 of each of 1-gram, 2-gram, 3-gram and 4-gram scores while computing BLEU4 score.

5. Results

5.1. Quantitative Analysis

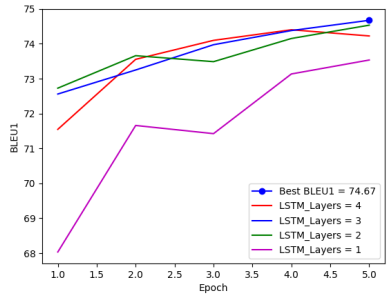
As described in section 4, I used BLEU score to evaluate my models. Variations in the models are based on minimum word frequency and number of LSTM layers.

Minimum Word Frequency Threshold: I investigated the performance of my model with three different thresholds – 2, 4 and 8. I chose them to experiment my model for two extremities. Threshold of 4 gave better results compared to 2 and 8. So, I fixed my threshold value to 4 for my future experiments.

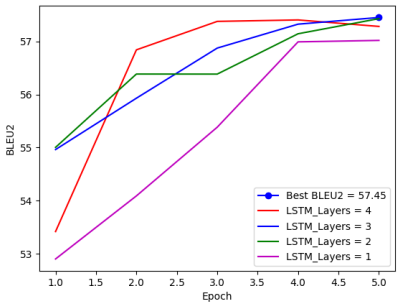
LSTM Layers: I experimented with 1, 2, 3 and 4 LSTM layers. Having more than two layers have given almost similar results and better when compared to one layer. Figure 5 (a) to (d) show the variations of BLEU score Vs Epoch.
Table 1 shows BLEU scores of models for different LSTM layers.

Model	BLEU1	BLEU2	BLEU3	BLEU4
LSTM-1 Layer	73.44	57.01	35.78	19.99
LSTM-2 Layer	74.53	57.42	36.26	20.77
LSTM-3 Layer	74.67	57.45	36.56	**20.98**
LSTM-4 Layer	74.22	57.28	36.18	20.61

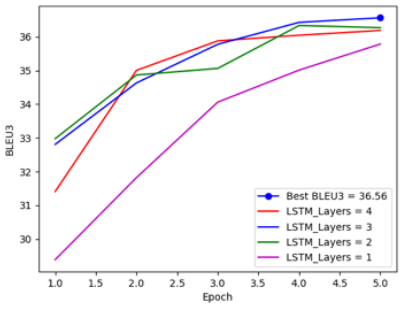
Table 1: BLEU1-4 scores of my models
(Minimum word frequency threshold set to 4)



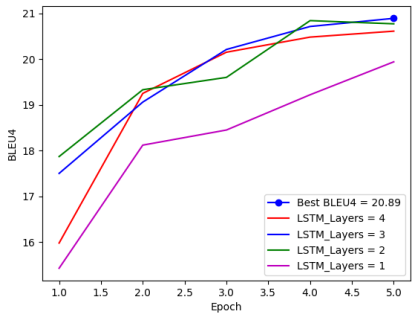
(a)



(b)



(c)



(d)

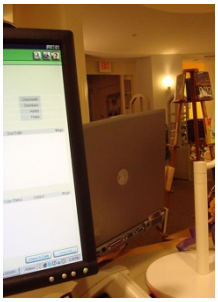
Figure 5 (a) BLEU1 (b) BLEU2 (c) BLEU3 (d) BLEU4 Vs Epoch

5.2. Qualitative Analysis

Below are the captions generated by my best model i.e. minimum word frequency threshold is 4 and number of LSTM layers is 3. Figure 6 shows images whose generated captions are successful and Figure 7 shows images whose generated captions failed. BLEU4 scores are also mentioned.

(a)

A man flying through the air while riding a snowboard.
BLEU4: 100



(b)

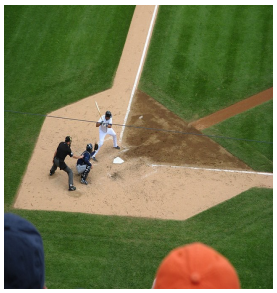
A laptop computer sitting on top of a wooden desk.
BLEU4: 45.99

A teddy bear is sitting on a couch.
BLEU4: 27.48

A clock tower with a clock on its side.
BLEU4: 28.16

(e)

A group of children are sitting around a table.
BLEU4: 47.71



(f)

A baseball player swinging a bat at a ball.
BLEU4: 71.86

Figure 6: Successfully captioned images with their BLEU4 scores

(a)

A dog and a dog are looking at a dog
BLEU4: 6.6e-153

(b)

A group of horses walking across a dirt road
BLEU4: 8.1e-153

(c)

A table with a laptop and a laptop on it.
BLEU4: 8.16e-153

(d)

A motor cycle parked on the grass with a man standing next to it. BLEU4: 3.16e-76

Figure 7: Failed images with their BLEU4 scores

5.3. Comparison with Baseline Models

As described in section 2, we have considered the below listed models as baselines. These include, NIC [6], NIC with attention [9], Deep Visual-Semantic Alignment [7] and human baseline.

Model	BLEU4
Random Forest	4.6
Nearest Neighbor	9.9
Human	21.7
NIC	27.7
Deep Visual-Semantic	23
NIC with Hard Attention	25.0
NIC with Soft Attention	24.3
My best model	**20.98**

Table 1: Comparison with baseline models

6. Problems with current captioning models

While working on this project, I spent a reasonable amount of time in understanding the problems of current captioning models and reading literature that are trying to address those.

6.1. Lack of Distinctiveness:

Current models lack to describe the unique aspects of the input image. A human can describe an image in a distinctive way. With a distinctive caption of an image, someone can easily identify what the image is referring to among various other similar images. The reason for it is described in [14]. According to them, lack of distinctiveness is due to Maximum Likelihood Estimation (MLE) as this method does not take the differences among captions of different images into account. They proposed a new learning method called Contrastive Learning which during learning takes mismatched pairs in addition to true image-caption pairs.

6.2. Cannot perform arithmetic on objects

Current models cannot count objects in the input image i.e. they cannot generalize numerical quantities not seen in training. For instance, if we train our model with images containing two cats playing, and if we give an image with four cats playing – then our model cannot caption it as four. The reason for it is described in [15] as learning behavior is more like memorization than abstraction. They tried to solve the problem with the help of new modules called NALU (Neural Arithmetic and Logical Unit) which can be used with standard neural networks to generalize numerical computation.

6.3. Cannot capture deep meanings

Humans can infer deep meaning in an image, which current models fail to. Unless explicitly trained machines cannot infer emotions present in a visual scene like humor, love, and other interpretations that human commonsense can easily capture. For instance, if an image contains a man petting a cat, then humans can infer love shown by the man towards the cat, but current captioning models would just describe it something like a man moving his hand over a cat.

7. Conclusion and Future Work

Image captioning is an interesting research problem that connects two fields of AI – Computer Vision and Natural Language Processing. As a part of this project, I explored different approaches to solve the problem. Specifically, I investigated CNN-LSTM based architecture to automatically generate captions. I experimented models based on minimum word frequency threshold and number of LSTM layers. My best model, where I set four as minimum frequency threshold and three LSTM layers obtained a BLEU score of 20.98 on MSCOCO dataset. My model trails human baseline by just 0.72 points. Current SOTA methods have crossed human baseline by a good margin.

However, we are lagging much behind in our quest to build a “truly intelligent” image captioning model. As mentioned in section 6, current models cannot capture deep meanings inside images, cannot generalize numerical quantities that are not seen while training and also cannot identify unique aspects of a visual scene which human can. So, although our models crossed human baseline, we have a lot more to do.

While working on the project, I defined these problems and studied literature related to them. I found few of them to be interesting and would like to work on them in future. Contrastive Learning is a generic learning technique that takes both positive and negative pairs as input which I would like to explore. Also, NALUs are interesting as they could let the current models generalize well for numerical quantities which is a basic trait that humans and animals possess.

In addition to these, one idea always fascinates me. Connecting symbolic AI and representative learning techniques could possibly inject intelligence into our captioning systems. To my knowledge, there are no significant methods in the literature which describes this in the domain of Image Captioning. I would be very much interested to take this approach in future.

References

[1] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and LeCun, Y., 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

[2] Siddharth, N., Paige, B., Desmaison, A., van de Meent, J.W., Wood, F., Goodman, N.D., Kohli, P. and Torr, P.H., 2016. Learning Disentangled Representations in Deep Generative Models.

[3] 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* (pp. 770-778).

[4] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

[5] Farhadi, A., Hejrati, M., Sadeghi, M.A., Young, P., Rashtchian, C., Hockenmaier, J. and Forsyth, D., 2010, September. Every picture tells a story: Generating sentences from images. In *European conference on computer vision* (pp. 15-29). Springer, Berlin, Heidelberg.

[6] Vinyals, O., Toshev, A., Bengio, S. and Erhan, D., 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3156-3164).

[7] Karpathy, A. and Fei-Fei, L., 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3128-3137).

[8] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A.C., 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), pp.211-252.

[9] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. and Bengio, Y., 2015, June. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048-2057).

[10] Bird, S. and Loper, E., 2004, July. NLTK: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions* (p. 31). Association for Computational Linguistics.

[11] https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning

[12] https://pytorch.org/tutorials/

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

[14] Dai, B. and Lin, D., 2017. Contrastive learning for image captioning. In *Advances in Neural Information Processing Systems* (pp. 898-907).

[15] Trask, A., Hill, F., Reed, S.E., Rae, J., Dyer, C. and Blunsom, P., 2018. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems* (pp. 8045-8054).

[16] https://github.com/cocodataset/cocoapi/tree/master/PythonAPI/pycocotools