IE 498JS SP2020 Final Project

Show and Tell: A Neural Image Caption Generator

Abid Hossain
Hyunjoon Rhee
Jack Yutong Li
Sejeong Yoo

Abstract

The problem of accurately describing an image combines methods and models popular in both computer vision and natural language processing applications. In this analysis, we implemented a well-known Neural Image Captioning model (Vinyals et. al 2015) on the COCO dataset. The model is trained on Blue Waters with different tuning parameters. Our best model achieved a BLEU-1 and BLEU-4 score of 69.9 and 42.6 respectively, and generated captions that are reasonably accurate and intuitive.

Introduction

Recent advances in deep learning, along with the increase in computational power, has dramatically reshaped the landscape of computer vision and natural language processing problems. Image Captioning is a problem that lies within the intersection of computer vision and natural language processing. Given an image, the model needs to generate an accurate, sensible, and grammatically correct caption for the image. This ability to generate coherent descriptions has wide applications, ranging from image search engines to assisting the visually impaired.

Combining both advances in computer vision and machine translation, Vinyals et. al (2015) proposed a generative model based on a deep recurrent architecture (Mikolov et. al, 2010; Sundermeyer, 2012) to generate image descriptions. They implemented a deep convolutional neural network (Donahue et. al, 2014) as an encoder to generate vectorized image features. These features are then served as input into the deep recurrent network, where the model is trained to maximize the likelihood of the target description. In this study, we implement the proposed model (Vinyals et. al, 2015) to the COCO data set to explore its functionality. The model is trained under different tuning parameters on Blue Waters. Our best model achieved a Bleu score of XX and generated sensible textual description for images in the test set.

This report is conducted in the following order. We give an overview of the model architecture, along with the considered tuning parameters in Section 2. Section 3 describes the preprocessing of the COCO data set, including the image and their corresponding captions. The evaluation criterion and results are presented in Section 4, and the summary of this analysis is given in Section 5.

Model

Existing methods tried to stitch together solutions of subproblems: encoding and decoding (Aker and Gaizauskas, 2010; Mitchell et. al, 2012; Elliott and Keller, 2013). However, Vinyals et. al (2015) proposed a single joint model. The inspiration comes from recent advances in machine translation, where given a description in source language S we want to generate a description in language T by maximizing P(T/S).

The goal is to directly maximize the probability of correct description given the image by the following equation:

$$arg\ max_{\theta} \sum_{(Ii,\ Si)\ \in\ D} log\ p(S^{i} \mid I^{i} , \theta) \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots...(1)$$

Here, (I^i, S^i) denotes the image and caption pair respectively. D: training dataset, θ: trainable parameters of the model. We can then apply the chain rule of independent probability distribution to the joint probability over the caption (S) :

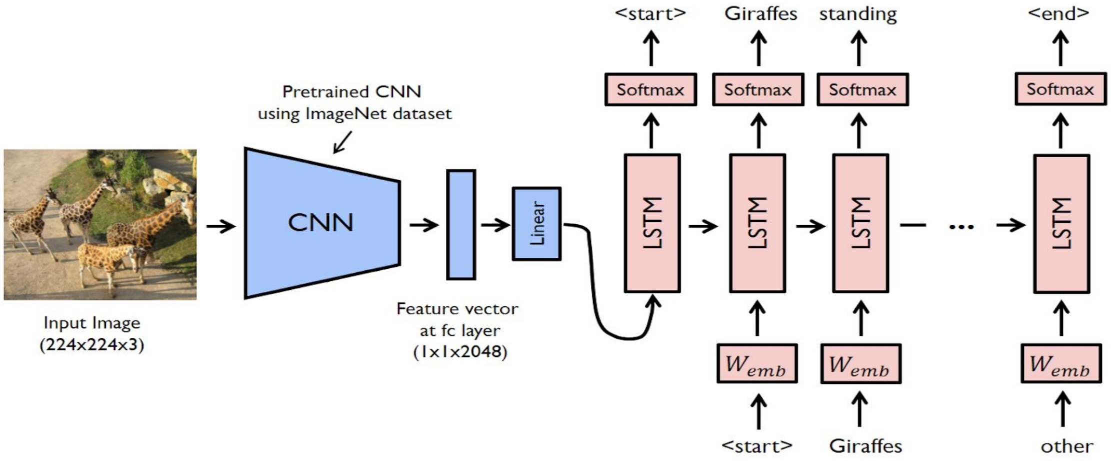
$$log\ p(S|I,\ \theta) = \sum_{t=0}^{N} log\ p\ (S_t|\ I,\ \theta,\ S_0,\ S_1,.......S_{t-1})\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

where S_t is the t-th word in caption S.

At training time (S,I) is an example training pair and we optimize the sum of log probabilities as described in (2) over the whole training set using stochastic gradient descent. The paper then proposes to replace RNN as an encoder by a deep convolutional neural network (CNN). CNNs are natural image encoders because they can capture spatial relationships.
In our implementation, the Encoder is a resnet-152 model (Kaiming et. al, 2016) pretrained on the ILSVRC-2012-CLS image classification dataset. This model is the current state-of-the-art for object recognition and detection. We use recurrent neural network (RNN) to model p (S_t| I, θ, S_0, S_1,.......S_{t-1}), which has a memory. The memory is updated after seeing a new input x_t by a nonlinear function f:

$$h_t = f(x_t,\ h_{t-1})\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.....(3)$$

For f we implement long-short-term-memory (Hochreiter et. al, 1997), which has shown state-of-the-art performance on sequence prediction tasks (Tai et. al, 2015). The following image describes overall flow of our model:



High-level Overview of the Model Architecture proposed in Vinyal et. al 2015

Data Set

We used the MSCOCO (Lin et. al, 2014) dataset which contains 82783 training images and 40401 validation images. Each image comes with 5 human captions. Since the human captions are of different lengths, we padded them to ensure that they are the same length by introducing ‘start’ and ‘end’ to mark the two endpoints of each caption. We also padded ‘unknown’ for less used words in the dictionary.

For data augmentation, we applied random crop (224, 224), horizontal flip, and normalization. We also converted all images to RGB format to have 3 channels which is needed for CNN encoder.

Data Preprocessing

Dataset Statistics

Table 1 shows the statistics of the image datasets used in this project. For each image in the dataset, we have 5 captions.

Dataset	Size	
	Train	Validation
MSCOCO	82,783	40,504

Table 1: Statistics of the datasets used in this project

Image Preprocessing

Since we are using ResNet152 as the encoder, we processed the images to a dimension of (3, 256, 256), the input of ResNet152, Moreover, we normalized the pixel values to be in the range of [0,1], and then standardized the image by the mean and standard deviation of the ImageNet images' RGB channels, that is mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225] for each channel.

PyTorch follows the NCHW convention, where N denotes number of samples in a batch, C denotes the number of channels, H denotes the height, and W denotes the width of an image. Therefore, we followed the convention and preprocessed our input data accordingly.

Caption Preprocessing

Captions are used as input and output of the Decoder, because each word is used to generate the next word. We followed the convention of the language model, and added the <start>, and <end> symbol at the start and end of each caption. Moreover, we limited the maximum length of each caption to 50 and added the <pad> symbol to those captions that are shorter than 50.

Training Method

The training process can be divided into two stages: The CNN part which encodes the image, and the LSTM network which generates caption. The loss is calculated after the caption generated for each training pair (I,S).

Encoding:

We use the pre-trained resnet-152 model which has been trained on ImageNet dataset. We fine tune it with an additional linear layer at the end.

Decoding:

We use the lstm layer in order to train the model. The dropout rate is set to 0.5. We receive an output from the encoder, which we flatten into batch size. The model used Adam as the optimizer for both encoding and decoding processes.

Analysis and Results

In this section, we present the results of our implemented Neural Image Captioning model on the COCO dataset.

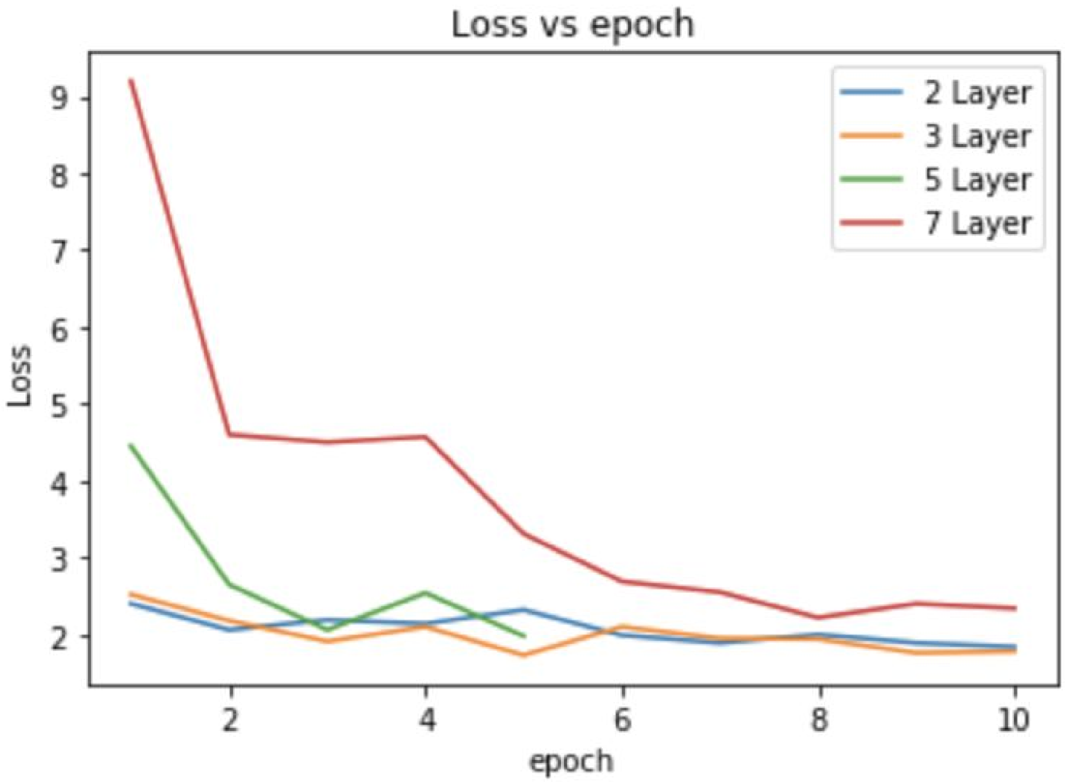
Hyperparameters:

In order to predefine all the hyperparameters before starting the training process. We crop the images during training to size 224. Our batch size is always 128.

For the decoder, we used an embedding and hidden layer size of 512. We investigate the performance under different hidden layers and embedding size for training. One of the hyperparameters that we compared was the number of layers of the model. The layers we experimented with were 2, 3, 5, and 7. We also considered different epochs of either 5 or 10. We compared the difference in the loss and the results through training the model through different numbers of epochs.

Results:

We first present the training loss of the model under different hidden layers.



From the plot of loss vs. epoch, it’s clear that 3-lstm layer has the least training loss. We also notice that the loss converges quickly for the 3 and 5 layers but took more epochs for the model with 7 layers.

Model Architecture	7-layer, 10-epoch, hidden size = 512	2-layer, 10-epoch, hidden size = 512	3-layer, 10-epoch, hidden size = 512	5-layer, 5-epoch, hidden size = 256	Theoretical
Bleu-1	0.645	0.699	0.696	0.668	0.642
Blue-2	0.444	0.504	0.498	0.468	0.447
Bleu-3	0.395	0.428	0.423	0.407	0.385
Bleu-4	0.418	0.431	0.426	0.423	0.401

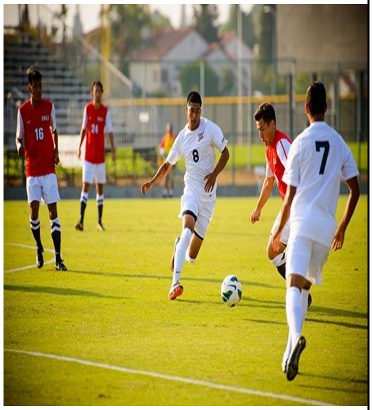
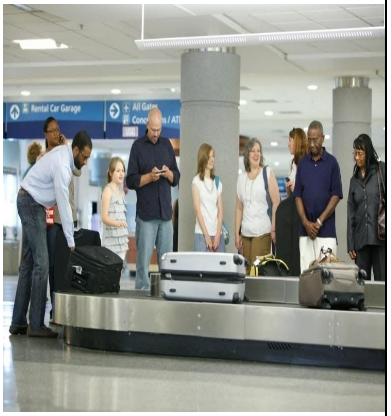
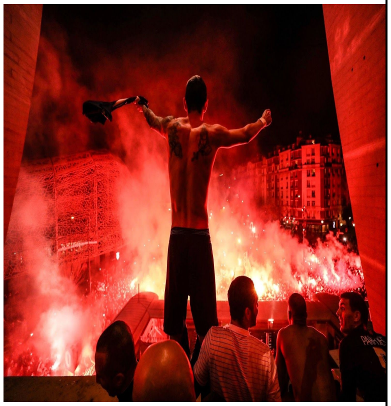
Table 2: Bleu scores on validation dataset according to different model

We observe that the best results are achieved under the 2-layer model, with the 3-layer model very close behind. The BLEU score gets progressively lower as the number of hidden layers increases, suggesting a sign of overfitting. All models out-performed the theoretical BLEU score, which is computed by comparing one human caption to other's from the list of 5 captions per image. Note that the theoretical BLEU score is the same for all model architecture.

Dataset	BLEU1 Our code | paper	BLEU2 Our code | paper	BLEU3 Our code | paper	BLEU4 Our code | paper
COCO	69.9 | NA	50.4 | NA	42.8 | NA	43.1 | 27.7

Table 3: Best model score compared to paper (2-layer, 10-epoch, hidden size = 512)

We present some interesting captions that were generated under different tuning parameters.

Correct caption	Somewhat related caption	Completely unrelated caption
		
2 layers, 5 epoch: "<<start>> a group of young children playing soccer on a field . <<end>>" **5 layers, 5 epoch:** "<<start>> a group of people playing soccer on a field . <<end>> " **3 Layers, 10 epoch:** "<<start>> a group of young boys playing soccer on a field . <<end>>"	**2 layers, 5 epoch:** "<<start>> a group of people standing around a table with a cake . <<end>>" **5 layers, 5 epoch:** "<<start>> a group of people standing around a table . <<end>> " **3 Layers, 10 epoch:** "<<start>> a group of people standing around a	**2 layers, 5 epoch:** "<<start>> a man is cooking in a building with a crowd of people . <<end>>" **5 layers, 5 epoch:** "<<start>> a man in a kitchen preparing food in a kitchen . <<end>>" **3 Layers, 10 epoch:** "<<start>> a group of people standing in front of a fire hydrant . <<end>>

7 layers, 10 epoch: “<<start>> a group of people playing soccer in a field .<<end>>”	luggage carousel . <<end>>” **7 layers, 10 epoch:** “<<start>> a man is sitting at a table with laptop . <<end>>”	**7 layers, 10 epoch:** “<<start>> a man standing in a kitchen with a refrigerator . <<end>>”

Table 4: Some example of captions

Computation Time

Our model was trained and tested on Bluewaters. Specifically, we used the xk-nodes of Bluewaters.

More information on xk node: 8 Bulldozer core/node.

CPU characteristics: 2 CPU/core (16 CPU/node), 156.8 GF peak CPU performance, 51.2 GB/s CPU memory bandwidth.

GPU characteristics: 2688 CUDA cores, 1.31 TF peak GPU performance, 250 GB/s peak GPU memory bandwidth.

1. For 3-lstm layer and 10-epoch, 512 hidden and vocabulary size the training took ~12.6 hour on 4 xk nodes with 16 processes per node. So, total training time is 4x16x12.6 hrs = 806.4 hours.
2. For 7-lstm layer and 10-epochs, 512 hidden and vocabulary size, the training took ~20 hours on 4 xk nodes with 15 processes per node. Total training time = 4*16*20 = 1280 horus.

Conclusion

In this study, we successfully implemented the Neural Image Captioning model on the COCO dataset under different tuning parameters, using Blue Waters as the computing resource. Our results show that the 2-layer LSTM model performs the best with the highest BLEU-1 and BLEU-4 score of 69.9 and 42.6 respectively. The BLEU score gets progressively worse as the number of hidden layers increases, suggesting that the model might be overfitting, especially for the 7-layer model. Furthermore, we noticed that even though 5 epochs were sufficient to generate valid captions for models with less than 5 layers, it was insufficient for the 7-layer model. To address this issue, we ran 10 epochs for the 7-layer model to generate captions, suggesting that more epochs are needed to train more complex models. Second, the captions generated are grammatically correct, but differs to an extent across different tuning parameters. Additionally, the content of the captions varies drastically, both literally and semantically, for images with very ambiguous meaning. In all, we have implemented and show that the proposed model is highly effective at generating readable captions on the COCO dataset.

References

1. Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
2. J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In ICML, 2014.
3. Mikolov, Tomáš, et al. "Recurrent neural network based language model." *Eleventh annual conference of the international speech communication association*. 2010.
4. Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. "LSTM neural networks for language modeling." *Thirteenth annual conference of the international speech communication association*. 2012.
5. A. Aker and R. Gaizauskas. Generating image descriptions using dependency relational patterns. In ACL, 2010.
6. D. Elliott and F. Keller. Image description using visual dependency representations. In EMNLP, 2013.
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. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
9. Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. "Improved semantic representations from tree-structured long short-term memory networks." *arXiv preprint arXiv:1503.00075* (2015).
10. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
11. Palangi, Hamid, et al. "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.4 (2016): 694-707.
12. Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." *European conference on computer vision*. Springer, Cham, 2014.
13. https://medium.com/explorations-in-language-and-learning/metrics-for-nlg-evaluation-c89b6a781054
14. https://github.com/kelvinxu/arctic-captions
15. http://cs.stanford.edu/people/karpathy/deepimagesent/caption_datasets.zip