Image Captioning using Deep Learning

By

Parth Jain : 012436632,

Susmit Gaikwad : 012428143

For CS 286

Presented to Dr. Mark Stamp

Table of Contents

1. INTRODUCTION

1.1 What is Image Captioning ?

Image captioning is a process of generating textual description for an image to describe the actions and context of the image. Captioning an image physically, by understanding every image and then writing context for it is a tedious task. Machine learning and deep learning techniques are used for doing this, but the process is a bit complex.

1.2 What are the types of Image Captioning ?

Due to the rapid growth in machine learning and deep learning many approaches have emerged, the two most important one being generative model-based approaches and retrieval based approach. Generative model based approaches use recurrent neural networks (RNN) and long short term memory (LSTM), which have proven to be very effective over the past few years. Whereas, there are instances when retrieval based approaches perform better than generative based approaches. But one of the major issue with retrieval based approach is compute power

1.3 What are the types of Generative model-based approach ?

Caption generation for an image requires both the understanding of computer vision model and also of natural language processing. There are multiple ways of implementing generative model- based approaches. The three important ones with respect to image captioning are :

A. **Model 1: Generate the Whole Sequence -** Generating the entire textual description for the photo given a photograph.

B. Model 2: Generate Word from Word - LSTM generates a prediction of one word given a photograph and one word as input.

C. Model 3: Generate Word from Sequence - Given a photograph and a sequence of words already generated for the photograph as input, predict the next word in the description.

We have implemented generative based model approach and specifically model 1, where we generate the whole sequence for image captioning and also a retrieval based model approach using K nearest neighbors (KNN). The two models are evaluated over Bilingual Evaluation Understudy (BLEU).

2. PROBLEM STATEMENT

In this project we implemented an image captioning model that generates the whole text sequence using the Keras framework [10]. The image captioning model consists of three parts: extracting image features to feed the caption generation model, training the caption generator model on those image features, and finally using the trained model to generate caption text when given an input image features. This was achieved by using an encoder-decoder deep learning architecture: a Visual Geometry Group Neural Network (VGG) [9] for the feature extraction as the encoder and a Recurrent Neural Network model (RNN) [8] with Long Short Term Memory (LSTM) units to train and generate caption text that acts as a decoder. After the model was trained, we then evaluated the model using the Bilingual evaluation understudy (BLEU) score. We used the same score to evaluate a K-Nearest Neighbor (KNN) model for comparison [2]. In the end, there is a significant increase in the BLEU score over KNN. Also, an advantage over KNN is that the LSTM can generate text from scratch, whereasthe KNN only ever "borrows" an existing caption that it knows.

3. TOOLS AND SETUP

To build the model we leveraged and learned several different modern machine learning tools/frameworks such as Keras and Google's Tensorflow [11]. Keras is a high-level neural network API used for quick prototyping and experimentation. It allows for creation of Convolutional Neural Networks as well as Recurrent Neural Networks. The frameworks made it not only possible for us to get a working model together, but also allowed us to feasibly train the model - using the GPU, rather than CPU, of a computer to perform the necessary computations. We used a remote server with two NVIDIA Tesla V100 GPUs with 16 GB RAM each. For the KNN we used a Ball Tree algorithm from the Scikit-Learn library in Python and a GIST[6] library implementation from GitHub.

4. DATASETS

We used two image captioning datasets for training our models: Flickr8k[4] and Flickr30k[5]. Flickr8K has 6000 training images, 1000 validation images and 1000 testing images. Each image has 5 captions describing it. Flickr30k has around ~30,000 images and 158k descriptions, 5 for each image. The Flickr8k dataset is around 1.2 GB while the Flickr30k dataset is around 4.9 GB.

5. METHODS

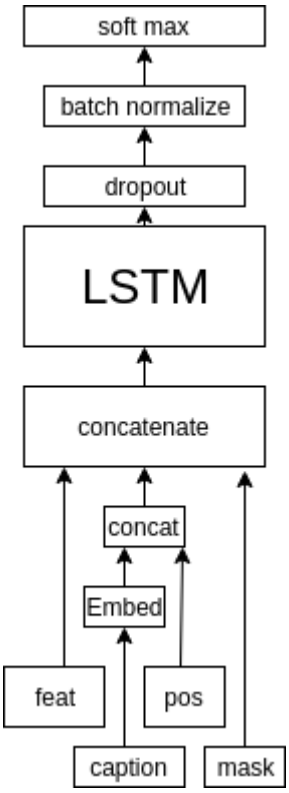
In our implementation, we use the Flickr30K image dataset with 30k images with 5 captions each to train the neural network model. The model we trained is called a Long Short-Term Memory (LSTM) neural network ([1], [8]). LSTMs are particularly good at natural language processing tasks as they are RNN units and hence they are good at learning sequential data like sentences in language. For feature extraction we used a pre-trained VGG16 model. We extracted the image features from the data and save them as NumPy arrays. The features were then fed into the caption generation model and the model was trained on this information. Given a new image, we first a extract the image features, then we feed the features into trained model and generate a caption.

5.1 *Feature Extraction with VGG*

The first step is extracting the features from images. We did so using a pre-trained VGG neural network model. The image is resized to 224*224, and then fed into the VGG network where the features are extracted as a NumPy array. VGG network has two version, one with 16 layers and one with 19 layers. For the purpose of our model, we decided to mainly focus on the 16 layers version - VGG16 []. VGG16 consists of three types of layers: a convolutional layer, pooling layer and fully-connected layer. The last three layers of the VGG16 model are all fully-connected. The problem here is that VGG network was used to do an image classification. So the output of the last layer is the object classification in the image; which is not what we want. To get the features, we remove the last two layers (including the output layer) and get the output from the fc-2 layer (second fully connected layer) which contains the dense feature representation data of the image.

5.2 *Training Caption Generation Model*

First we preprocessed our input captions and built a dictionary. All the words in our caption vocabulary are represented in a indexed list - mapping each word to an index. For caption generation, using Keras, we create a single LSTM cell with 256 neurons. For this cell we have four inputs: image features, captions, a mask, and a current position.



First the caption input and position input are merged (using concatenate) and then it goes through a word embedding layer. The image features and embedded words are then merged (using concatenate) with the mask input. Together they are passed through the LSTM cell. The output of LSTM cell then goes through a Dropout and Batch Normalization layer to prevent the model from overfitting. Finally, the Activation (Softmax) layer is applied and we get the result vector. The optimizer used was Adam and the batch size was set to 128. The result vector

contains the probability of every word in the dictionary for being the next word in the sequence. The word with the largest probability would be the current "best word" at any moment in the sequence.

Along with the pre-built dictionary this vector is used to to "interpret" the next generated word - which can be considered a type of ground truth for training in the true caption. The mask plays acts as a "recording", remembering the previous words used in captions, so that the model knows the words before the current word. And the current position indicates where in the sentence has the model predicted till, so that it will not fall into a loop. To be specific, a loop would refer to a caption like 'man in white shirt in a garden' interpreted as 'man in white shirt in white shirt in white shirt in...'. In this case, after the model generates the second 'in', the word input, feature input and the mask input for the model would all be exactly the same. Hence, it is important to input the current position of the sentence. The training for the encoder-decoder model took around ~7 minutes per epoch (for 20 epochs).

5.3 *Caption Generation*

Similar to training, we extract the features for each image to be caption generated. The images are passed through the VGG16 network to generate the features. For caption generation, we used a saved checkpoint of our model with the best loss. The first word input for inference is the '#start#' tag. The following inputs are the prediction results from each of the consecutive previous iterations. We also set the mask and input current position of the sentence similar to the training process. When caption generation produces a '.', we have our final output in the sequence!

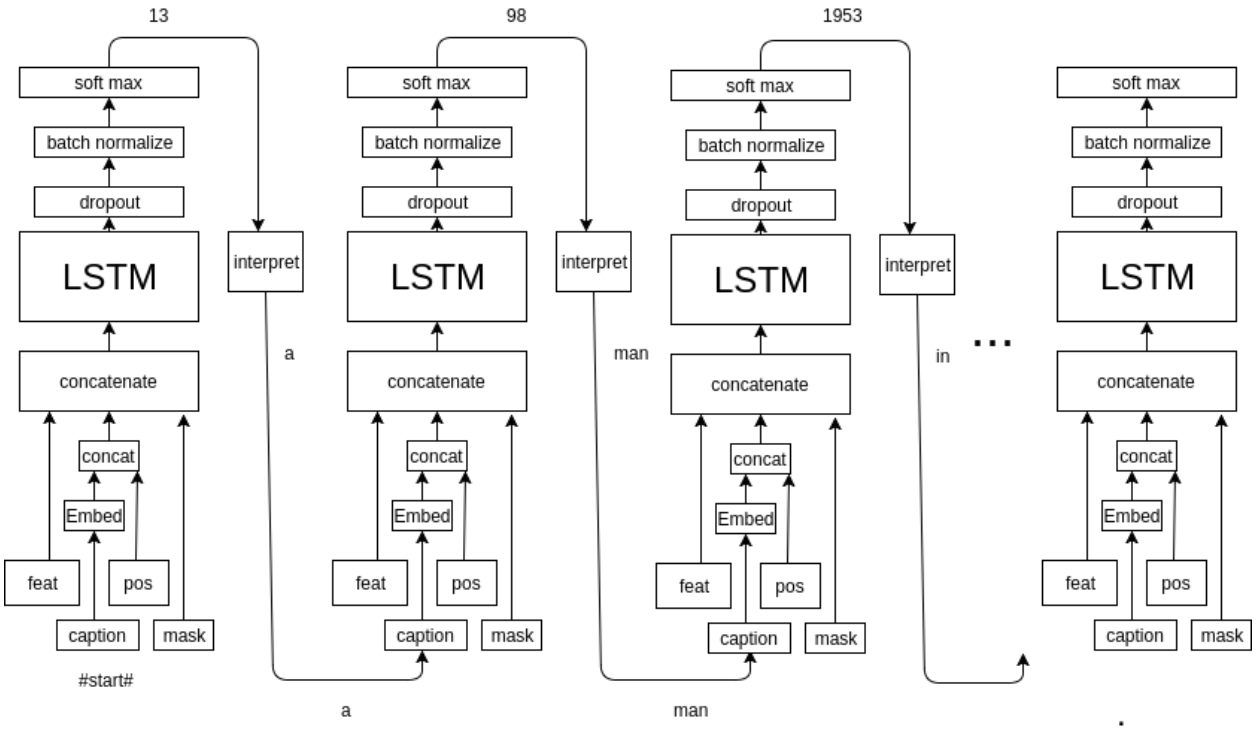


Figure 2 gives an example of generated caption -- 'a man is riding a bike in the road.', for Image 1. So, the first word is '#start#', the position is 0, and mask input is empty. The model would generate 'a' which is used as the input for the next iteration. In the second iteration, the word 'a' is marked by the mask input, and the position input moves forward. By doing this until the model predicts till '.', we stop the model and get the final output caption.

5.4 *Accuracy Evaluation*

BLEU Score: Bilingual evaluation understudy (BLEU) [3] is an algorithm that was originally designed to evaluate the quality of machine-translated natural language text; doing so using n-grams to compare and assign sentences an appropriate score. It has frequently been shown to be an accurate model as compared to human judgement and is considered a standard benchmarking tool.

5.5 *K- Nearest Neighbors*

For the KNN we used the smaller Flicker8k datatset. We used GIST to extract the features from the images into was arrays with length 4096. Then we fed these features into a KNN model using ball tree in sklearn as a distance metric to train. To get the prediction for an image M, we use GIST to extract features M' from M. Then we use the BallTree algorithm to find the K-nearest neighbors of M' to get all the candidate captions. The final step is deciding which caption to use. This decision is made by a consensus formula. We use the BLEU score in NLTK library to measure the similarity between two captions, then we choose the caption that maximizes the similarity formula.

6. RESULTS

To evaluate our results, we used the Flickr8k dataset to test our model, used to train the KNN. Unlike the KNN however, the results were much better. Using the same evaluation technique as the KNN - the BLEU score - for over 1000 images, we were able to achieve a peak of 69.95 and a valley of approximately ~35, effectively doubling the BLEU score accuracy. In the end, this averaged out to about a 64.1 ± 5.7 BLEU score accuracy. For reference some of our top and bottom scoring images see below in Appendix A. Some of the KNN predictions are in Appendix B as well.

7. CONCLUSION

The model generated fairly accurate predictions of images the model had not seen before. In BLEU score the average is about 69% which is better than the baseline KNN. Images are fed into the model and it creates captions shorter than 15 words and based on the position input, they usually tend to make logical sense when read.

More work in this field can directly benefit those who are impaired visually. We studied existing models to build this architecture of an encoder-decoder model [1][7] as we were unaware of how a neural network handled this task. The main issue we faced was for an image, eg: Image 1, was that it generated a caption like ‘a man in a bike in a bike in a bike’. The prediction process got trapped in a loop. Finally, we realised the model could not tell the current position in the caption being generated, which caused this problem. To fix the error we added one more input, the current position, and merged it with caption input. This resolved the issue of the looping in caption generation.

8. FUTURE WORK

The model could do certainly better. A beam search is an algorithm worth trying. Currently the model takes greedy approach. The output for each input is the word that is most likely to appear next in the sequence at that point in the sequence. The Beam Search algorithm considers the set of n best sentences, keeps track of resulting n candidates and creates sentences so it ends up with the overall most probable caption. We could also try to replace LSTM with GRU (Gated Recurrent Unit), which is a variation ofLSTM. It combines the forget and input gates into a single “update” gate. The resulting model is simpler than standard LSTM model.

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] Devlin, Jacob, et al. "Exploring nearest neighbor approaches for image captioning." arXiv preprint arXiv:1505.04467 (2015).

[3] Papineni, Kishore, et al. "BLEU: a method for automatic evaluation of machine translation." Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002.

[4] M. Hodosh, P. Young and J. Hockenmaier(2013) "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics", Journal of Artificial Intelligence Research, Volume 47, pages 853-899

[5] Plummer, Bryan A., et al. "Flickr30k entities:Collecting region-to-phrase correspondences for richer image-to-sentence models." Proceedings of the IEEE International Conference on Computer Vision . 2015.

[6] Itti, Laurent. "Gist/Context of a Scene."Gist/Context of a Scene Home Page. University of Southern California, ILab, 2000. Web. 14Apr. 2017.

[7] Puri, R., & Ricciardelli, D. (2017, March 28). Caption this, with TensorFlow. Retrieved April 01,2017,from https://www.oreilly.com/learning/caption-this-with-tensorflow

[8] Hochreiter, Sepp,and Jürgen Schmidhuber. "Long short-term memory."Neural computation 9.8 (1997): 1735-1780.

[9] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXivpreprint arXiv:1409.1556 (2014)

[10] Keras: Deep Learning library for Theano and TensorFlow. (n.d.). Retrieved April 01, 2017, from https://keras.io/

[11] TensorFlow. N.p., n.d. Web. 01 Apr.2017

APPENDIX A –

Neural Network Results

a man is riding a bike in the road .
BLEU: 58.6

a man is holding a beach .
BLEU: 46.5

a dog is running through a field .
BLEU: 56.4

a young girl is wearing a red and a large bush .
BLEU: 45.1

a man is on a wave .
BLEU: 61.4

a man in a hat is sitting on a pile of grass .
BLEU: 41.3

APPENDIX B

KNN Results

Predicted; Male performing water sports acrobatics while being pulled by a boat .
Actual: the boy laying face down on a skateboard is being pushed along the ground by another boy .

Predicted; This man with three dogs takes his route along the tree-lined street .
Actual: A lady walking her dog through an obstacle course , while other people are in the background .

Predicted: Two brown dogs run along a gravel road while another dog watches
Actual: A middle Eastern woman wearing green is sitting on a stone step holding a box of cigarettes .