Image Captioning with Recurrent Neural Networks

Adrian Ruvalcaba
Virginia Tech
{adrianr@vt.edu}

Abstract

Image captioning has captured the attention of computer vision researchers over recent years. As our understanding of deep learning architectures improves, so does the need for high-quality training data. While previous experiments largely relied human-labeled datasets such as ImageNet[5] or CIFAR[2], the emergence of recurrent neural networks has allowed the creation of automatic labelling networks that significantly increase the speed at which these datasets can be gathered. In this paper we explore the implementation of an image captioning architecture using recurrent neural networks, and discuss strengths and weaknesses of such systems after BLEU score evaluation.

1. Introduction

As humans, we are easily able to visually interpret our surroundings and understand the context of the scene before us. When we see the world around us, we don't have to stop and think about every object in our view; it comes naturally. For computers, however, this trivial problem becomes a much bigger hurdle. While data scientists had made big strides in image recognition with Convolutional Neural Network feature extraction, they struggled with the bridge between image and text data.

In this project, we look at an implementation that handles this by incorporating the combination of Convolutional and Recurrent Neural Networks. Specifically, we utilize the pretrained InceptionV3 model to generate image vectors, and an LSTM to generate partial captions. The outputs from both of these models is added together and sent through a feed-forward Dense network that predicts the next word.

The architecture for this project is heavily influenced by the implementation of image captioning done by Harshall Lamba [1]. Slight modifications were done throughout in terms of data augmentation and result evaluation, but overall the implementation remained largely unchanged.

2. Approach

This section will describe the implementation details of the image captioning network. We will analyze the network architecture, as well as describe the various word augmentation and embeddings necessary for valid results.

2.1. Architecture

As mentioned in the introduction, the model consisted of two key parts: The Convolutional Neural Network feature encoder and the Recurrent Neural Network partial caption generator. Figure 1 shows the full model architecture created here.

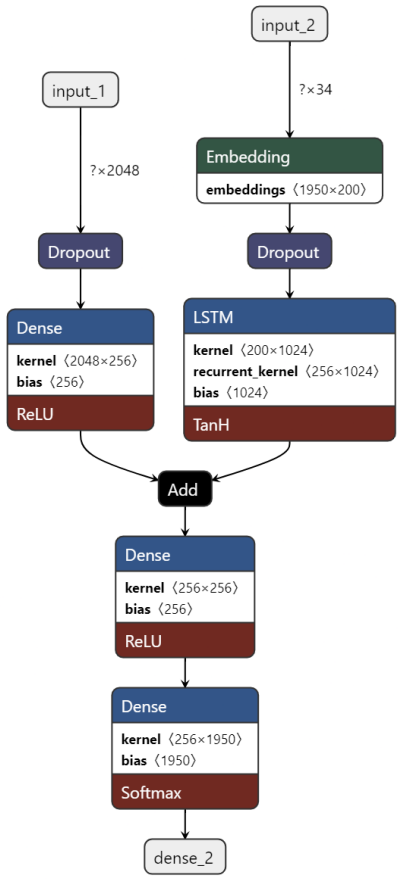

Figure 1. Image Caption Architecture

The InceptionV3[7] network is utilized for image encoding. With weights pretrained on the ImageNet dataset

and the last two layers dropped, we were able to turn images from a (229,229,3) matrix to (2048,) image vectors. The entire dataset was pre-encoded using this method, which allowed for faster training and inference when generating captions. The converted embeddings were saved locally and used through *input_1* in Figure 1.

For the Recurrent Neural Network, and LSTM was used. Partial captions would be sent through the network (*input_2)* and the next word in the sequence would be predicted. As an example, take the following image.

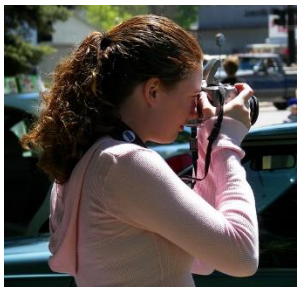

Figure 2. Flickr-8K sample image

During training, we first begin with a start token ***"startseq"***. The network would get the start token and add it to the corresponding image embedding. The feed-forward Dense layers would predict the next word, and the process would restart. This time, however, the partial caption would be something along the lines of ***"startseq woman"***. This would similarly get combined with the image embedding, and the next word would get predicted. Eventually, the network would be satisfied with the predictions and the output token ***"endseq"*** would get appended, ending the process with a result resembling ***"startseq woman taking picture with camera endseq".*** A maximum caption size of 34 was used here, as it was the length of the longest caption in the training set.

The model was compiled with categorical cross entropy as the loss function and the Adam optimizer.

2.2. Dataset

The dataset used for the project was the Flickr 8K[5] dataset from the University of Illinois. This dataset contained 8,000 images, each with an accompanying five captions. The dataset was split into train/test groups, which allowed evaluation on the model performance.

To prepare the dataset, we first create a caption dictionary. For each image, we extract all five captions and split them into individual words. The output would be a new dict with a key of image names and values of word tokens. We then clean up the data by removing punctuation, numbers, and individual letters, as well as converting everything to lowercase. This would not only remove duplicate words (Adrian vs adrian), but it also helps keep the algorithm on track since it does not have to worry about extraneous tokens. This new cleaned dict can be used to generate the full vocabulary available for caption generation.

2.3. Training

Model training was done using a generator. The generator would use input images, positional encodings, and true captions to create encoded input-output pairs for each caption. The purpose is to provide the model with batches of training data, which is much more computationally efficient than attempting to train all at once.

3. Experimental Results

The results after training showed impressive performance given the complexity of the problem. We first analyze the results using BLEU[4] score. We calculate four different values: BLEU-1, BLEU-2, BLEU-3, and BLEU-4. Smoothing method 2 is applied to all calculations. For each BLEU configuration, we iterate through each of the images and compare the predicted caption to the true value of each of the five captions per image. The results over the entire test set are averaged and shown in Table 1

BLEU-1	BLEU-2	BLEU-3	BLEU-4
0.1318	0.1076	0.0982	0.0934

Table 1. Average BLEU Score over Test Set

It can be seen that as more and more words are considered, the lower the BLEU score. Further analysis showed that a good proportion of the predicted captions had a score of 0.0.

Visualizing the results can be useful for comparing the captions. Figures 3 and 4 shows a random image with the appropriate caption.

Figure 3. Random Image with Captions

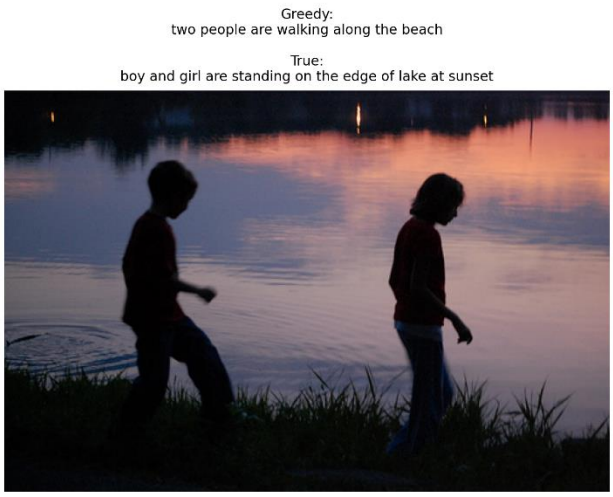

Figure 4. Random Image with Captions

With the captions, it is clear to see that while the network is gaining some knowledge on the context of the image, there is still fine-tuning. For example, in Figure 4, the caption accurately describes two people walking alongside a body of water. While the network believes the scene is from a beach, the true caption says it is instead a lake. The true caption also goes into more detail, mentioning the time of day in the picture. Ultimately, the results from the model are incorrect. It was not able to predict exactly what the caption was able to, but the difference can be understood. A majority of the key context in the image was still able to be successfully extracted.

It is important to note that the method of prediction is as recommended in the implementation, which is a Greedy search. This is a simple approach that generates the captions based solely on highest probability for the next word, but more advanced models would utilize different predictors. A common one is Beam search, which was briefly experimented with. The results of the two predictions are as follows:

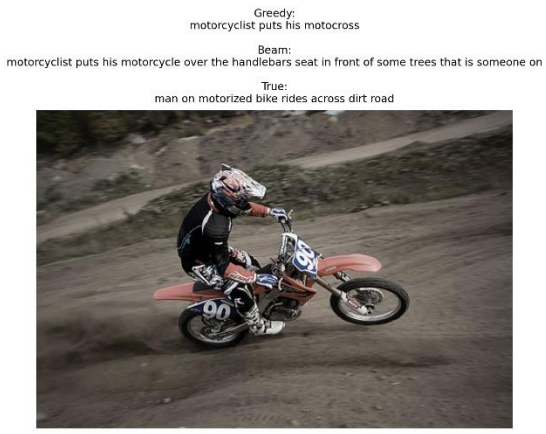

Figure 5. Random Image with Beam and Greedy Captions

As can be seen, the Beam prediction performed much worse. It seems to have a propensity for generating longer text, which in turn causes a more sporadic result.

4. Conclusion

Image captioning is a complex and difficult task that requires a firm grasp of several Deep Learning concepts. By utilizing positional embeddings and image/word vectors, we are able to generate captions with a surprising level of accuracy. While more complex architectures are available, this implementation was a good introduction into the domain of multimodal algorithms.

Further experimentation would explore the influence of different hyperparameters, mainly different method of predicting the output words. As always, more images and captions to train with are often useful for higher results, so different datasets can also be explored. Regardless, the implementation shown here demonstrates the ability to create these networks and understanding of the underlying concepts.

References

[1] Dhruvil Patel. "Image Captioning with Keras: Teaching Computers to Describe Pictures." Accessed on: April 16, 2023. Available at: https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8

[2] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images." Accessed on: April 16, 2023. Available at: https://www.cs.toronto.edu/~kriz/cifar.html

[3] DigitalOcean Community. "BLEU Score in Python." Accessed on: April 16, 2023. Available at: https://www.digitalocean.com/community/tutorials/bleu-score-in-python

[4] Jason Brownlee. "How to Calculate BLEU Score for Text in Python." Accessed on: April 16, 2023. Available at: https://machinelearningmastery.com/calculate-bleu-score-for-text-python/

[5] Micah Hodosh, Peter Young, and Julia Hockenmaier. "Framing image description as a ranking task: Data, models and evaluation metrics." Journal of Artificial Intelligence Research 47 (2013): 853-899.

[6] Alex Krizhevsky and Geoffrey Hinton. "Learning to Represent Images using Deep Convolutional Neural Networks." Accessed on: April 16, 2023. Available at: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[7] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images." Accessed on: April 16, 2023. Available at: https://www.cs.toronto.edu/~kriz/cifar.html