Modernization and Analysis of a Neural Image Captioner

Noah Weiner
EECS @ Yale
New Haven, CT
noah.weiner@yale.edu

Kaleb Gezahegn
EECS @ Yale
New Haven, CT
kaleb.gezahegn@yale.edu

Evan Strittmatter
EECS @ Yale
New Haven, CT
evan.strittmatter@yale.edu

***Abstract*—We focus on improving the accuracy of captions for the NIC image to caption model from the paper *Show and Tell: A Neural Image Caption Generator*. We retrain the network from the paper using an open-source TensorFlow adaptation of their model. The bulk of our contribution is modifying the codebase so that (a) the user can optionally train on a subset of the downloaded dataset instead of the entire dataset, (b) the user can train on COCO data but can now also optionally train on the SBU caption dataset, and (c) instead of taking images from the local disk, the user can select to pull images via URL using the COCO API. Part (c) better facilitates experimenting with training on Google Colab, since loading thousands of images into the Colab runtime is difficult. We train the NIC model on COCO 2014, but also on the SBU caption dataset, a large dataset that the original authors did not train on. Our inspiration for this experiment comes from the aforementioned paper by Vinyals et al., which specifically mentions that a major bottleneck for their training process was the size and quality of available datasets. The authors predicted that as the size of available image-caption datasets increased, so would the accuracy of their model. To measure results, we use the BLEU scoring metric to compare ground truth captions to the captions generated by our network for the validation data split. We also used humans (ourselves) to evaluate the accuracy of the captions generated on the validation data. We present some sample test images which we captioned using the COCO- and SBU-trained networks.**

I. Introduction

We focus on the problem of taking a JPG image and attempting to produce a caption for it using a neural network. Image-to-caption models are a growing "'hot topic"' in neural networks. The ability to convert an image, typically composed of hundreds of values for representing pixel data, to a short sequence of words lends itself to many interesting applications both in terms of data storage and translation efficiency.

The problem of image-to-caption generation demands a fusion of computer vision and natural language processing techniques. Not only does the model need to identify objects in the image, but it also needs to identify how those objects relate to and interact with each other which is a much more complicated effort. This is a much more complicated task, so a natural language model needs to be incorporated in order to produce well-ordered sentences.

Image captioning is a relatively well-researched problem since around 2015, but there is room for improvement, which is why we train the model on other datasets and look for any improvements in inference accuracy. Our general approach to do this was to extensively modify an open-source TensorFlow codebase based on NIC so that it could handle training on fewer and varying datasets. After modifying the codebase, we ran training and evaluation using the GPU partition on the Yale HPC cluster, an Intel NUC, and an NVIDIA RTX 3060 GPU.

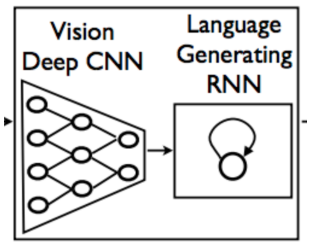


Fig. 1. The high-level architecture of the NIC model.

II. Background/Related Work

Captioning models evolved out of advancements in machine translation and a need for improving the accessibility of images to the visually impaired community with [1]. It has since grown into a significant branch of Machine Learning research with improvements in computational power, robustness of model architecture and growth of datasets allowing for much more varied utility of this technology. It has led to better performance in addressing accessibility issues [3] as well as having broader applications that range from image indexing on our phone's photos app to assisting medical professionals in breaking down X-Rays [2].

A. Caption Generation Models

In particular, we will be expanding upon the model proposed in [1]. The model structure depicted in Figure 1., combines a Convolutional Neural Network (CNN) with a Recurrent Neural Network (RNN). The CNN performs object identification in presented images, while the RNN then constructs a sentence based on the objects identified in the CNN.

B. Model Evaluation Metrics

The performance of a caption generation model is evaluated based on the accuracy of the captions it generates. Intuitively one could do this by reading a generated caption, looking at the image it was based on, and then assigning it a score then and there. There are many proposed ways to do this, with most involving humans rating captions on a scale from 1 to 4 with 4 being a perfect description without error, and 1 being no correlation. Although this is the most accurate way to ensure the quality of generated captions, it runs into major problems with scalability. In order to extract data from a meaningful sample size, raters will have to look through thousands or even tens of thousands of captions, which is both time-consuming and costly. As a result, research has searched for automated methods to rate captions that align with the results of human qualitative metrics, but without the associated manpower cost.

For a while, these autonomous rating metrics relied primarily on the n-gram overlap score. Prevalent metrics include Bleu, CIDEr, ROUGE, and METEOR. These metrics compare generated captions with a human-made ground truth caption, and assign each caption a score. A large portion of the overall score is determined by a series of n-words that overlap between the generated caption, and the ground truth captions, hence the term n-gram.

- A man jogs down the street with a dog
- A man jogs down the street carrying a leash

This is an example of a 6-gram overlap as the words "A man jogs down the street" are shared in both sentences. This is a significant amount of overlap and would result in a high score for this generated caption. In cases like this, the use of n-gram overlap accomplishes exactly its purpose perfectly - ensuring the generated caption matches a caption which is known to match the image. In practice however, there are several flaws with this method. The first being word overlap alone doesn't ensure an accurate caption

- A Brontosaurus takes a big bite of leaves from a tree.
- The little girl takes a big bite of cereal for her breakfast.

This is a 5-gram overlap with "takes a big bite of", so both sentences would receive a high similarity score, however, they correspond to completely different images. Another more significant problem is n-gram scoring metrics struggle to rate novel sentences and as a result, discourage models from generating them. Novel sentence generation is a critical part of caption generation if the model is to accurately describe previously unseen combinations of objects in an image.

- The gym was filled with lots of used equipment.
- People left their weights strewn across the floor after they finished working out.

The only overlap between these two sentences is the word "the", however, both describe the same image. This means despite being a perfect caption for an image, this caption would receive a low score in an n-gram rating metric. To compensate for these shortcomings, evaluation metrics will lump in other factors to the overall score such as word overlap, or length of the caption but the overarching problem remains the same [4]. When compared to Human qualitative metrics, these scoring metrics fall short [5], and even more promising models like SPICE should not be used as a sole metric for evaluation. Instead, they can be used to guide the development of a model.

III. Approach

To train the NIC TensorFlow model on a large amount of data, we used existing open-source code based on the CNN→RNN network structure from the *Show and Tell* paper.

This section details the framework of your project. Be specific, which means you might want to include equations, figures, plots, etc.

A. Model Architecture

The NIC model is inspired by advances in machine translation emerging around 2014, whereby a sentence S in one language was transformed into a translation T in a target language by maximizing $p(T|S)$, the probability that T is correct given S. The key in such translation is to use an encoder-decoder model, where an "'encoder"' recurrent neural network (RNN) transforms S into a vector representation, which is then fed into a "'decoder"' RNN that generates T. In the case where the source is an image, NIC thinks of the encoder as the object detector for the image, which is implemented as a standard convolutional neural network (CNN).

The model maximizes the conditional probability of the correct image caption given the image. As [1] explains, this probability $(S|I)$ can be given by the equation

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

where N is the length of the caption sentence, S_t is this word, and $S_0, ..., S_{t-1}$ are the previous words. During training, (S, I) is a training example pair, and want to optimize the sum of the probabilities over the whole training set, using gradient descent. In other words, we want to adjust the network's weights in a way that makes it more likely to predict each next word of the caption sequence correctly.

The RNN portion of the model is used to implement $p(S_t|S_0, ..., S_{t-1})$. The words are represented using memory cells h_t, which are updated after seeing a new input x_t using the equation

$$h_{t+1} = f(h_t, x_t) \tag{2}$$

where h_{t+1} is this value of the memory, f corresponds to the chain of LSTM (long short-term memory) blocks, h_t is the last value of the memory, and the input x_t corresponds to the image and word. The image is represented by the CNN output, and the word is represented using a word embedding model, where words are represented as vectors in a predefined relational vector space. These vectors can be thought of as weights trained with the rest of the model.

A feed-forward LSTM chain is used for the RNN. LSTM cells are commonly used for sequence generation, where this output depends on the last output. The cells are controlled by multiplicative logic gates, including the output gate, input gate, and forget gate. LSTMs could be seen as flip-flops or a similar kind of electronic memory cell, in that they have their own sort of “enable” and “reset” signals. Each cell has multiple weight matrices, but this single set of weight matrices is shared among all cells. By using LSTMs, sequential control elements can be fine-tuned to better teach the network how to predict the next element of the sequence. A softmax function is used to populate the output of the LSTM chain, which gives the probability distribution p_{t+1} over all the words.

B. Loss Function

From [1], the goal in training is to minimize the loss function

$$L(I,S) = -\sum_{t=1}^{N} \log p_t(S_t) \tag{3}$$

which is really maximizing the probability of the correct caption given the input image.

Once the model is trained, we can run test images by sampling from the probability distributions to get the word in a given index of the caption.

C. Accuracy Calculation

During training, the accuracy is computed by computing the difference between the most probable next caption word (found by taking argmax of the computed probability distribution over all words) and the ground truth next caption word. This computation can be found in the build_rnn() function in model.py.

D. Code Modification and Functionality

The bulk of our project time was spent modifying the code from the DeepRNN/image_captioning repository. This code needed extensive modification in order to run at all as it was built to work with older versions of TensorFlow and Python. We first modified it to be compatible with TensorFlow 2 and Python 3. Once we got the code to run, we made many core modifications for our project experiment, including but not limited to: changing the data preparation functions to optionally pull COCO URL images instead of local filenames so that during training, images could be loaded via the COCO API instead of from disk; changing the data preparation functions to only use the image IDs and corresponding captions for the subset of COCO images stored in a local folder (instead of extracting ALL image IDs and captions from the JSON annotations file); change data preparation functions so that user can specify a certain number of local or URL-fetched images to extract captions for and load during training; adding a script to download images and annotations file for the SBU dataset into a local folder, and adding support for using SBU captions dataset and annotations file instead of COCO during training and evaluation. Adding support for SBU training and eval was especially difficult due to SBU and COCO having very different annotations file structures. Please see our GitHub repo (linked in Conclusion and in supplementary files) for the code and a detailed changelog and README.

The codebase optionally uses a pre-trained VGG16 object detection and classification model for the CNN portion of the network. We decided to skip training the CNN to save training time, so we used this pre-trained .npy file.

The code for training generally functions in the following way: on startup, the dataset is prepared in the following way:

- If “local” is set to True in the config file, the code looks in a local folder for images and selects config.num_train_data of those images, extracts their image IDs from the filenames, then looks in the COCO or SBU JSON annotations file to find ground truth captions for the corresponding images. If “local” is False, the code looks in the COCO JSON annotations file and finds image URLs and corresponding ground truth image captions for the first config.num_train_data annotations. It then loads the images via URL with the COCO API during training.
- The image-caption pairs are filtered by caption length, and any pairs that have captions longer than config.max_caption_length are thrown out.
- A vocabulary is built containing all words found in all ground truth captions. Each word is assigned an index.
- The image captions are converted into lists of word indices (using the Vocabulary).
- A DataSet object is instantiated that contains the image ids, image locations, and indexed captions.
- The TensorFlow model is created via the build_rnn() function in model.py, using tf layers.
- train() is called on the model using the DataSet.

IV. Experimental Results

We ran three training experiments, the process and results of which are outlined below.

A. Training Using 30000 COCO Images on HPC Grace

We trained for about 12 hours using 30000 of the COCO train split images. The training was done on the Yale HPC cluster’s Grace node, using the scavenge_gpu partition. The plots of accuracy over time and loss over time during training are shown in Figure 2. We were only able to train for about four epochs, with a batch size of 32. We achieved an ending accuracy of about 0.25. Some sample test images that we passed through the network are shown in Figure 3. We see that the model makes mediocre captions.

B. Training Using 3000 COCO Images on NUC

We trained to completion using 3000 of the COCO train split images. The training was mostly done on an Intel NUC desktop with no GPU. The plots of accuracy over time and loss over time during training are shown in Figure 4. We trained for 100 epochs, with a batch size of 32. We achieved an ending accuracy of about 0.4.

Some sample test images that we passed through the network are shown in Figure 5. We see that the model makes decent captions!

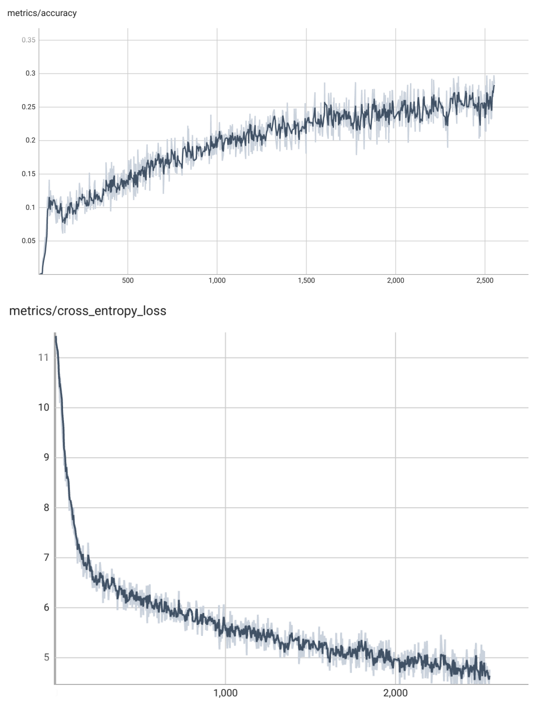


Fig. 2. Training for four epochs using 30000 COCO train images, on HPC Grace.

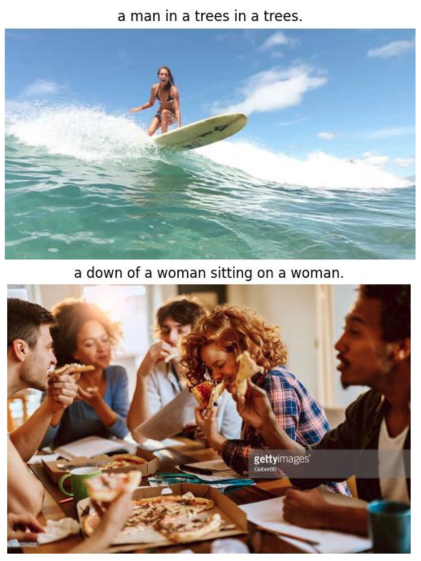

a man in a trees in a trees.
a down of a woman sitting on a woman.
gettyimages

Fig. 3. Generated captions from the four epoch experiment training with 30000 COCO images. These captions are somewhat related to the image, in that they identify present objects, but the sentences are garbled English

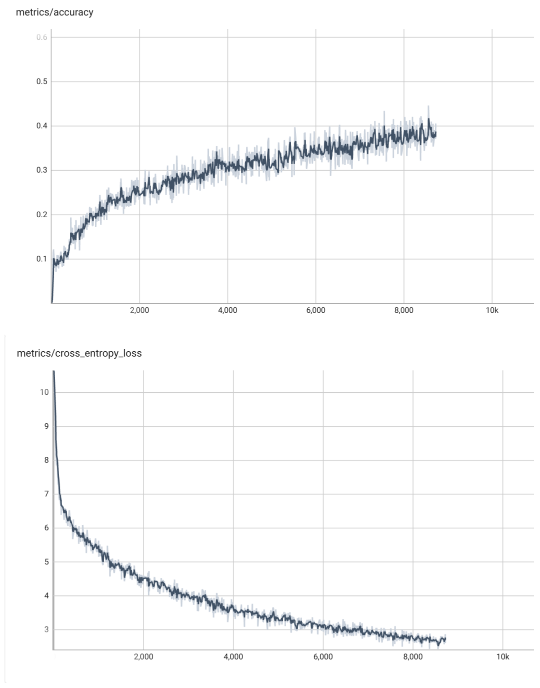


Fig. 4. Training for 100 epochs using 3000 COCO train images, on Intel NUC.

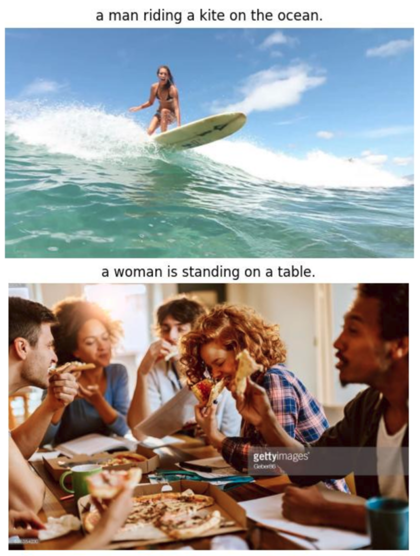
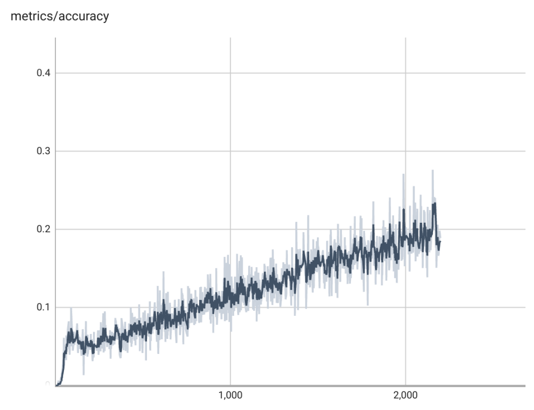

a man riding a kite on the ocean.
a woman is standing on a table.
gettyimages

Fig. 5. Generated captions from the 100 epoch experiment training with 3000 COCO images. These captions are better matches as the form coherent sentences, and relatively accurate object identification



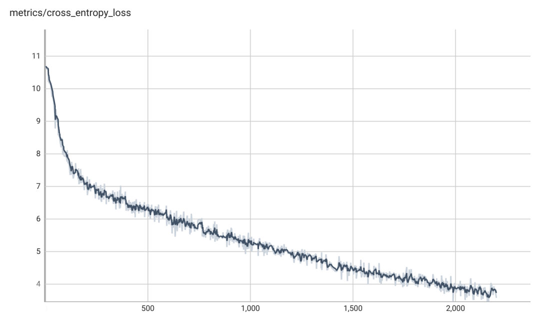


Fig. 6. Training for 100 epochs using first 3000 SBU train images, on Intel NUC.

C. *Training Using 3000 SBU Images on NUC*

We trained to completion using 3000 (the first 3000) of the SBU images. Training was done on an Intel NUC desktop with no GPU. The plots of accuracy over time and loss over time during training are shown in Figure 6. We trained for 100 epochs, with a batch size of 32. We achieved an ending accuracy of about 0.2. To create an evaluation data split, we took 100 images from the end of the dataset. The higher-quality captions indicate that the time spent iterating through multiple epochs is more valuable to the end result than a larger captioned dataset.

Some sample test images that we passed through the network are shown in Figure 7. We see that the model makes nonsensical captions. We believe there might have been a problem with training. We noticed that many of the 3000 images were pruned out of the dataset due to their captions being longer than the max caption length parameter set in the config file, so only 679 image-caption pairs were used in the training. It seems like something else might have also gone wrong since the BLEU evaluation scores for the SBU-trained network didn't make much sense either.

D. *BLEU Evaluation Scores*

We used BLEU as a guiding metric to compare the results of our model to other trained models. Table I shows the BLEU-1, BLEU-2, BLEU-3, and BLEU-4 scores computed during the evaluation for each experiment. We see that something probably went wrong in the SBU-trained network evaluation process.

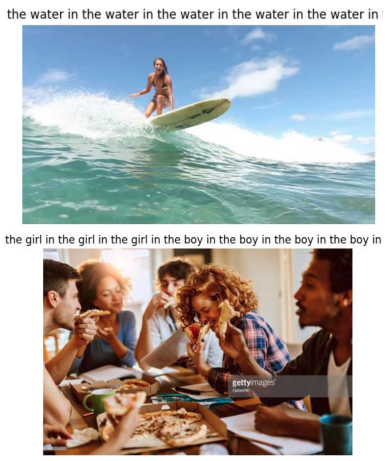

the water in the water in the water in the water in the water in
the girl in the girl in the girl in the boy in the boy in the boy in the boy in
gettyimages

Fig. 7. Generated captions from the 100 epoch experiment training with 3000 SBU Images. These captions are nonsensical.

TABLE I
BLEU SCORE RESULTS

Experiment	BLEU-1	BLEU-2	BLEU-3	BLEU-4
1	0.184	0.076	0.024	0.012
2	0.221	0.098	0.047	0.019
3	1.000	0.001	0.000	0.000

Unfortunately, we lacked the computing resources and time to produce results matching the BLEU scores achieved in [1]. What's interesting to note is that training on a much smaller amount of data for many more epochs yielded much better results than training on a much larger amount of data for very few epochs.

Based on our human evaluations of the image captions produced by the networks during the evaluation run, and on our human evaluations of the captions generated for the test images, we also select experiment 2 to yield the best results and rank experiment 3 as having the worst results.

V. CONCLUSION

We've learned that training a complex, large model with many parameters is difficult and takes an extensive amount of time and computing resources. We've also learned that tweaking the amount of data, number of training epochs, batch size, and learning rate can affect training and that less data but more epochs could be the way to go.

In the future, we'd like to try training the model on more datasets, like the newer Conceptual Captions dataset from Google AI. We'd also like to try tweaking the model structure more, possibly by changing the number of LSTMs in the chain, which according to [1] was set to 512 as the default. It would also be interesting to swap out the pre-trained VGG16 for some other pre-trained CNN like ResNet50 to see if that has an effect on test accuracy.

Note: we have zipped in a supplementary text file, which contains a link to the GitHub repo. We hacked through and modified the code extensively (see commit history), and the README contains a detailed changelog and instructions for running training, eval, and test with either COCO or SBU.

REFERENCES

[1] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator." CVPR , page 3156-3164. IEEE Computer Society, (2015).

[2] D.R. Beddiar, M. Oussalah, and T. Seppänen. "Automatic captioning for medical imaging (MIC): a rapid review of literature." Artificial Intelligence Review (2022): 1-58.

[3] W. Shaomei, J. Wieland, O. Farivar, J. Schiller "Automatic alt-text: Computer-generated image descriptions for blind users on a social network service." Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing. 2017.

[4] K. Papineni, Kishore, S. Roukos, T. Ward, W. Zhu "Bleu: a method for automatic evaluation of machine translation." Proceedings of the 40th annual meeting of the Association for Computational Linguistics. 2002.

[5] P. Anderson, B. Fernando, M. Johnson, S. Gould "Spice: Semantic propositional image caption evaluation." European conference on computer vision. Springer, Cham, 2016.