Image Captioning Using Neural Networks (CNN and LSTM)

Andrew Candelaresi, Austin Holler, Deepak Kyasaralli Thimmappa,

Hayeong Song, Zhenguo Chen

Abstract

In this paper we present an image captioning model. We use a deep-learning framework known as Keras. Keras implements Tensorflow on the back end. Tensorflow is a framework for doing a series of operations in a chain. The dataset Flickr30K was used to train the model. The general technique is to feed the features of an image to the model, which is capable of generating text of length less than or equal to a predefined caption length.

1. Introduction

Computer vision, machine translation, and object detection are all areas of machine learning that have been growing rapidly in the past decade. As such, there now exist several different frameworks that provide for caption generation implementations. In this project we provide an image captioning implementation using one such framework - Keras [10]. To develop the image captioning model there were three parts to be considered: extracting image features for use in the model, training the model on those features, and finally using the trained model to generate caption text when given an input image's features. How these three parts were eventually accomplished was through the use of two different techniques: a Visual Geometry Group Neural Network (VGG) [11] for the feature extraction and a Recurrent Neural Network (RNN) [12] to train and generate caption text. After this was accomplished, we then evaluated the model using the same scoring standard - a Bilingual evaluation understudy (BLEU) score - as our baseline for comparison [4]. In the end, there was a significant increase, over the baseline, in BLEU score accuracy; with the added advantage that this model can also truly generate image captions - as compared to the baseline which could only ever "borrow" an already known caption.

2. Tools

To build the model we leveraged and learned several different modern machine learning tools/frameworks such as Keras and Google's Tensorflow [13]. 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.

2.1 *Keras*

As described by those who created it, Keras is a "...high-level neural networks API, written in Python...developed with a focus on enabling fast experimentation." It provides a simple wrapper for implementing the building blocks of advanced machine learning algorithms. However, it does not do the heavy lifting itself, rather it leverages Tensorflow, or Theano, for its functionality. The specific library *Keras.layers*, provided for us many of the methods required to manage the image object detection necessary for our model as well as word embeddings, masks, and dropouts.

2.2 *Tensorflow*

Tensorflow, Google's machine learning library, as described by Google is "...an open source software library for numerical computation using data flow graphs." Its main abstraction is a tensor, a node in a graph that defines and stores a series of computationally intensive operations, to eventually be executed in what is called a "session".

To do efficient numerical computing in Python, we typically use libraries like NumPy that do expensive operations such as matrix multiplication outside of Python, using highly efficient code implemented in another language. Unfortunately, there can still a lot of overhead that can come from switching back to Python every operation. This overhead becomes especially noticeable if ran in a distributed manner, where there can be a high cost to transferring data. Tensorflow ends up also doing its heavy lifting outside of Python, but it takes things a step further to avoid this overhead. Instead of running many single expensive operations outside of Python though, Tensorflow describes a graph of interacting operations - not executing the computations until the graph has been built up and the session run.

3. Technique

In our implementation, we use the Flickr30K image dataset to train our model. The model we trained to do our predictions is called a Long Short-Term Memory (LSTM) neural network ([8], [14]). We use a combination of the features extracted from the image along with the sentences that are captions for our data - all provided in the Flickr30k dataset. In prediction of an image the most important parts are then VGG feature extraction and caption prediction using the trained LSTM.

3.1 Feature Extraction with VGG

The first step is extracting the features from images. We did so using a pre-trained VGG neural network model. Using the pre-trained model, the image is read in, resized to 224*224, and then fed into VGG neural 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 [11]. VGG16 is a type of neural network that 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 challenge here is that VGG network was used to do an image classification. So the output from the last layer is the classification of the object in the image; which is not exactly what we want. But we figured out how to get the features from the VGG network. Instead of getting the output from the last layer, we get the output from the fc-2 layer (fully connected layer) which contains the feature data of an image.

We tried both VGG16 and VGG19 version, and both of them have great performance. For our final version, we use VGG16 for our feature extractor.

3.2 Building/Training The Predictive Model

The first step was to preprocess our input captions and build a dictionary. Then we present all the words in our caption as index list - mapping words into an index.

For captioning, 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. For an illustration of the look below, in our images, at Figure 1.

First the caption input and position input are merged (using concatenate) and then it go through a word embedding layer. Then the image features, and embedded words are also merged (using concatenate) with the mask input. Together, they all go through the LSTM cell. The output of LSTM cell then goes through Dropout and Batch Normalization

layer to prevent the model from overfitting. Finally, the Activation (softmax) layer is applied and we get the result.

The result itself is a vector with each entry representing the possibility of every word in the dictionary. The word with largest probability would be our current "best word". Along with pre-built dictionary this vector is used to to "interpret" the generated next word - which can be considered a type of ground truth for training in the true caption. The mask plays an important role in all of this, "recording" the previous words used in captions, so that the model knows the words before the current word. And input the model with current position of the sentence so that it will not fall into a loops. To be more specific, loop refers to caption like 'man in white shirt in a garden' may look like 'man in white shirt in white shirt in white shirt in…'. Because, 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. Then the model would generate 'in white shirt' again which will create a loop and generate a caption that does not fully represent the image. Therefore, it is necessary for us to input the current position of the sentence.

feats	Image features
captions	For training, true captions. For prediction, #start# tag and previous prediction.
mask	Record previous words
position	Position of current word in a sentence

Table 1: Explanation for input

3.3 Image Captioning

Like training, we also need to get the features for each image to be predicted. So, the images go through the VGG16 network first, to generate the features. For captioning, we used the same LSTM model. The first word input for this model is the '#start#' tag, and the following input are the prediction result from the previous iteration. We also need to set the mask and input current position of the sentence as we did in training process. When we reach a prediction which produce '.', we have our final prediction!

Figure 3 gives an example of generated caption -- 'a man on a bicycle down a dirt 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 input for the next iteration. Then the word 'a' is marked by the mask input, and the position input move foreword. By doing this until our model predict '.', we stop the model and get the final caption.

3.4 Accuracy Evaluation

BLEU Score: Bilingual evaluation understudy (BLEU) [4] 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. Since it

4. Results

In finalizing our results, we used the Flickr8k dataset to evaluate our model, used in our previous baseline KNN, to test against ([1], [3]). Unlike our baseline though, the results were far from poor. Using the same methodology as our baseline - the BLEU score - over 1000 images, we were able to achieve a peak of approximately 89.95 and a valley of approximately 60.55, effectively doubling our BLEU score accuracy. In the end, this averaged out to about a 74.8 ± 14.7 BLEU score accuracy. For reference some of our top and bottom scoring images see below in Appendix A.

5. Summary - Our Model Works!

Using our model we were able to generate fairly accurate predictions of images our model

had not seen before. In *BLEU* score the average is about 75% which is better than our baseline(implemented using KNN). We feed images into the model and allowed it to create captions. The caption are shorter than 15 words and based on the position input they make logical sense when read.

This research is important because it is useful to generate captions for pictures that a computer may not have a previously captioned. With the training data the system can observe object of new image and generate captions based on it.

Also most of the image caption generation implementations use Tensorflow. We have gone through the effort of implementing image prediction using Keras. Our successful creation of this model will be useful for people who are looking for Keras implementation of image captioning.

As our model can produce human like description of an image, advancement in this area can later support artificial intelligent vision system. This system will be able to generate description of a new object based on what it observed previously and can interact with environment around. Also advancement in this technology can directly benefit those who are impaired visually.

The first challenge we faced was how to build a specific model to fulfill our requirement. We started by trying to feed in our input data, then use a simple LSTM cell and an activation layer to generate caption. Then we ran into another problem. For Image 1, it would generate a prediction like 'a man in a bike in a bike in a bike'. It seemed that our prediction process got trapped in a loop. We tried some different approaches to solve this problem. Finally, we found out our model could not tell the current position in the caption it is generating, which caused this problem. To fix this error we added one more input, the current position, and merged it with caption input. This resolved the issue of looping in caption generation.

Finally, we presented our result as web page, in which user can upload image to the system, then captioning would be done is the backend and return the images with caption.

6. Future Work

Apparently, our model is not perfect, neither is our prediction process. To make our model perform better, beam search would be something worth a try. Currently we are taking a greedy approach. As we take input and iteratively output the word that is most likely to appear the next. And build up the caption eventually. The Beam Search algorithm considers the set of n best sentences and keep track of resulting n of the candidates. And build up sentences so that it can end up with the overall most probable caption. In addition to beam search we could also use l2 regularization and ensembles.

Also, we can try to replace LSTM with GRU (Gated Recurrent Unit), which is a variation of LSTM. It combines the forget and input gates into a single "update" gate. The resulting model is simpler than standard LSTM model. Or we can also do parameter tuning. With these change the performance will improve.

6. Work Division

- **Baseline**: Andrew, Zhenguo, and Austin
- **Exploring Frameworks and Research**: Everyone
- **Feature Exaction**: Zhenguo
- **Model and Training**: Austin and Zhenguo
- **Prediction**: Deepak, Hayeong, and Zhenguo
- **Evaluation**: Andrew, Austin
- **Webpage**: Zhenguo
- **Writeup**: Everyone

7. Project Repo

And our project is available online at Github - (https://github.com/ZhenguoChen/csci5622Project)

Resources

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

[2] Oliva, Aude, and Antonio Torralba. "Modeling the shape of the scene: A holistic representation of the spatial envelope." *International journal of computer vision* 42.3 (2001): 145-175.

[3] Hodosh, Micah, 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.

[4] 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..

[6] 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

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

[8] 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.

[9] 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

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

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

[12]Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

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

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

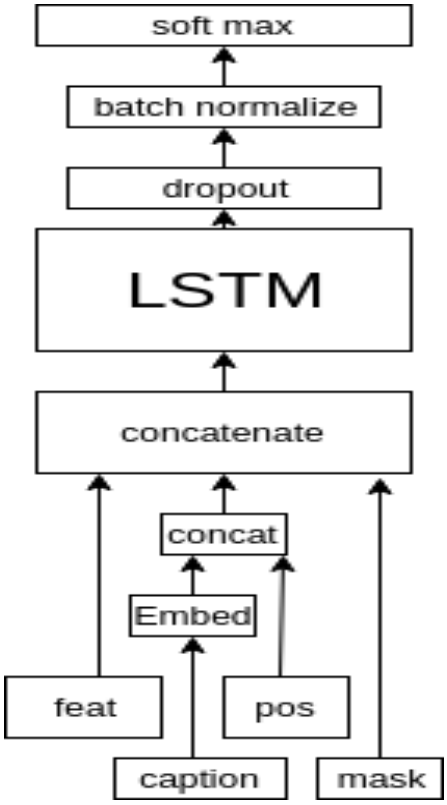

FIgure 1. LSTM model

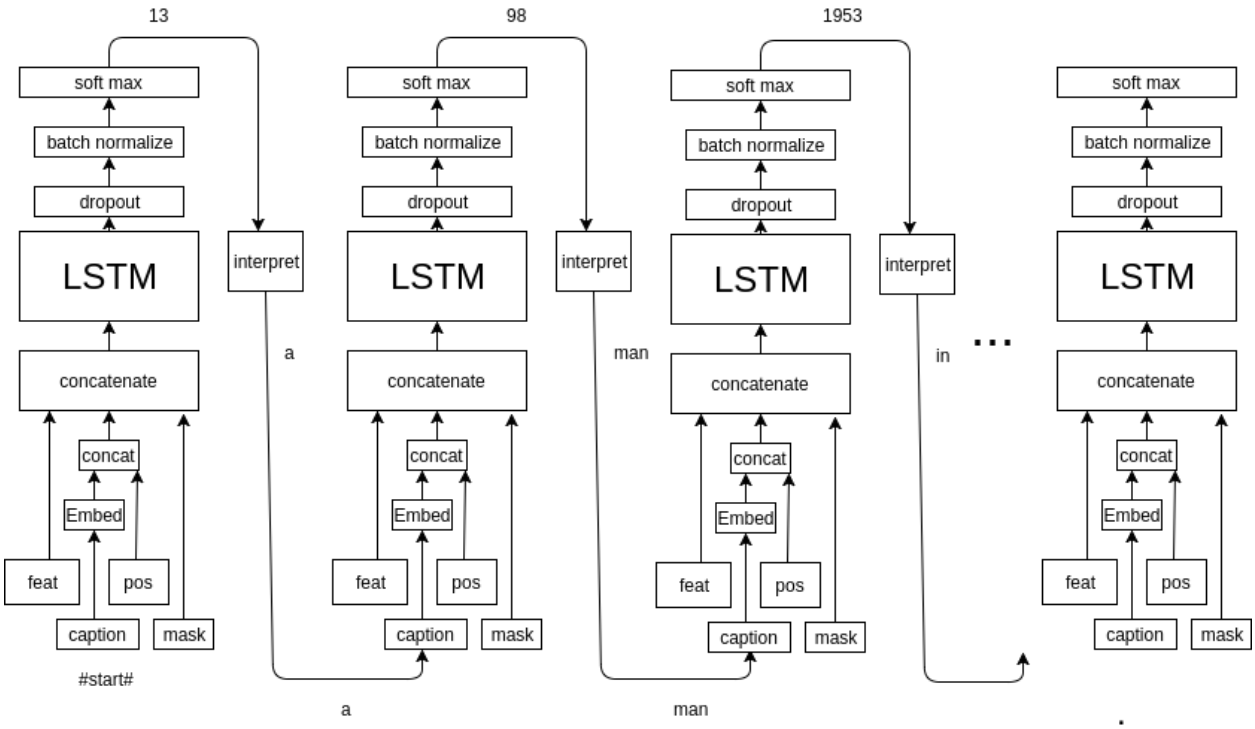

Figure 2. Captioning process to generate ‘a man in a bike down a dirt road’.

```
input_1 (InputLayer)            (None, 1)             0
_________________________________________________________________
embedding_1 (Embedding)         (None, 1, 256)        753408
_________________________________________________________________
reshape_1 (Reshape)             (None, 256)           0
_________________________________________________________________
input_4 (InputLayer)            (None, 82)            0
_________________________________________________________________
concatenate_1 (Concatenate)     (None, 338)           0
_________________________________________________________________
input_2 (InputLayer)            (None, 4096)          0
_________________________________________________________________
dense_1 (Dense)                 (None, 256)           86784
_________________________________________________________________
input_3 (InputLayer)            (None, 2943)          0
_________________________________________________________________
concatenate_2 (Concatenate)     (None, 7295)          0
_________________________________________________________________
reshape_2 (Reshape)             (None, 1, 7295)       0
_________________________________________________________________
lstm_1 (LSTM)                   (None, 512)           15990784
_________________________________________________________________
dropout_1 (Dropout)             (None, 512)           0
_________________________________________________________________
dense_2 (Dense)                 (None, 2943)          1509759
_________________________________________________________________
batch_normalization_1 (BatchNorm (None, 2943)         11772
_________________________________________________________________
activation_1 (Activation)       (None, 2943)          0
=================================================================
Total params: 18,352,507
Trainable params: 18,346,621
Non-trainable params: 5,886
_________________________________________________________________
None
```

Figure 3. Model layers. Four inputs: image features, captions, mask and position. Four layers: word embedding layer, LSTM layer, Dropout layer and batch normalization layer.

Image 1. a man on a bicycle down a dirt road .

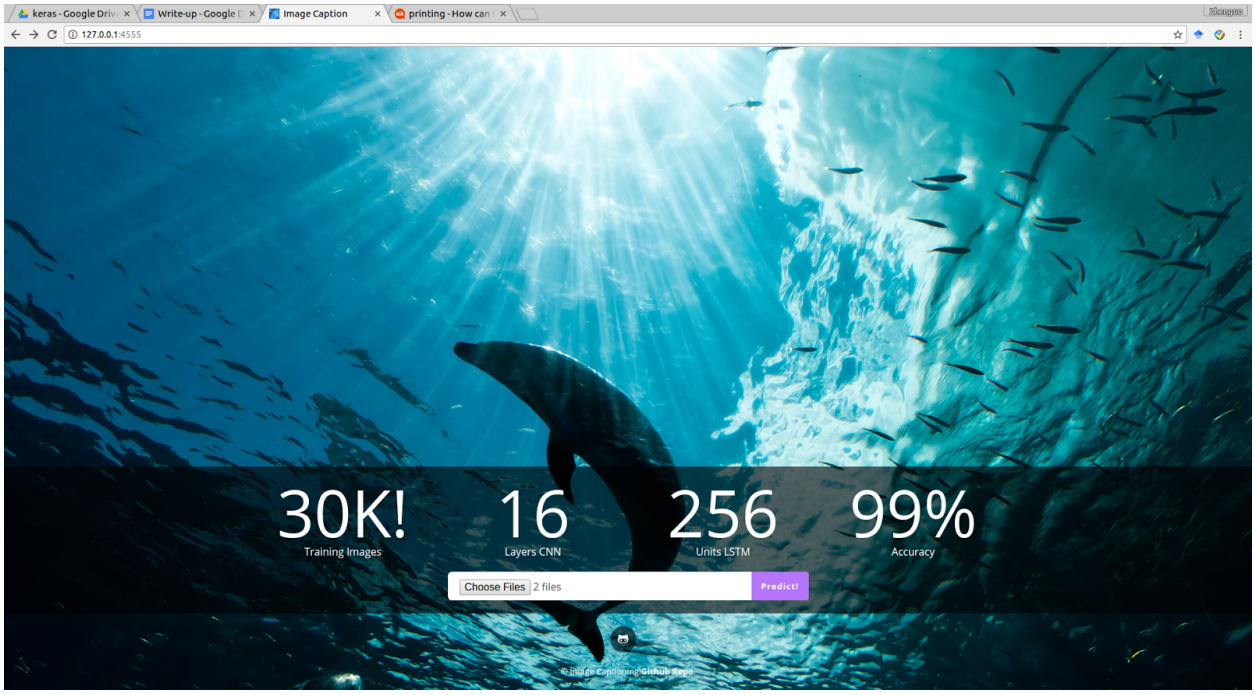

Image 2. Home page

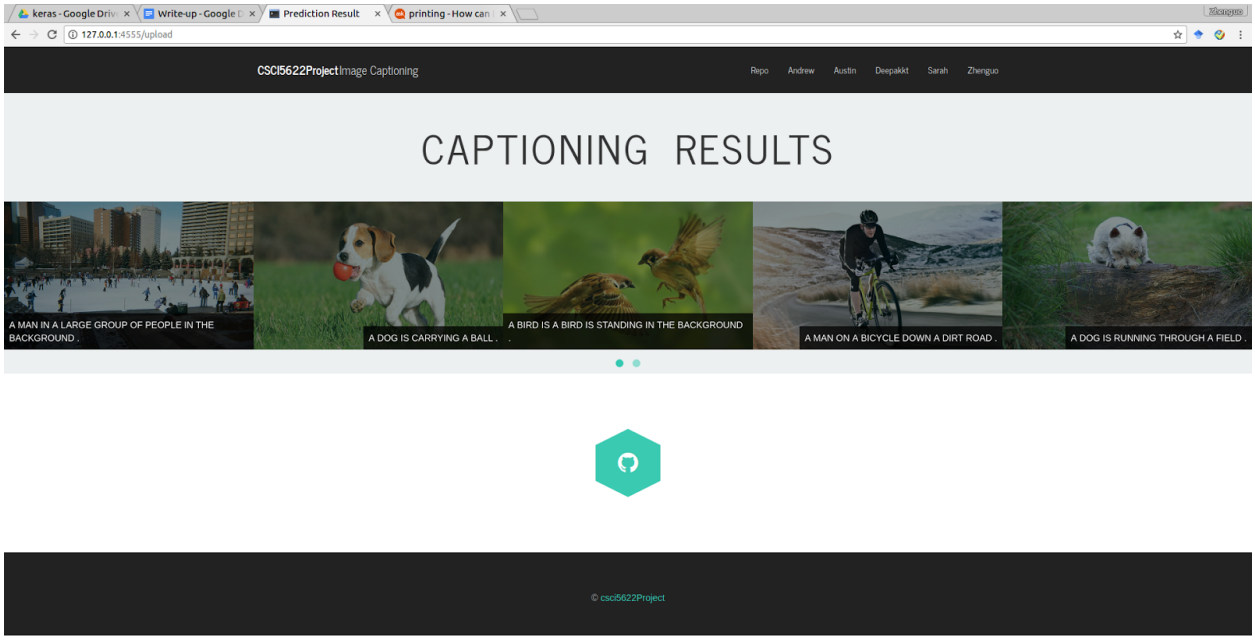

Image 3. After upload images, the backend would caption the images and generate a new web page.

Appendix A.

Top Scoring Images:

Predicted: A black dog in the snow.
BLEU Score: 89.95

Predicted: A man in the snow.
BLEU Score: 88.38

Predicted: A dog is in the water.
BLEU Score: 87.68

Bottom Scoring Images:

Predicted: A person in a blue shirt and black pants.
BLUE Score: 65.04

Predicted: A man in a black jacket and a blue jeans.
BLEU Score: 64.96

Predicted: A man in a gray shirt and gray shirt is standing in the distance.
BLEU Score: 64.92