Project Proposal: Every Image Has A Story

Md. Rifayat Uddin
University of Michigan
Group 29
firstauthor@i1.org

Bayarjavkhlan Batbaatar
University of Michigan
Group 29
bbatbaat@umich.edu

1. Abstract

Every picture tells a story. In computer vision, being able to describe what a picture says has been a very fascinating problem people have tried to solve. Its evident from the number of papers that have been published and the various online resources on learning techniques to be able to make a picture tell a story. This interested both of us, and hence we decided to delve into a project where we could be able to tell a story from a picture. We set out to create a program by using learning techniques we learned in class, to be able to make this happen. This paper highlights our work as well as the various problems we faced in making this program.

2. Introduction

The world is becoming more interested in the various deep learning techniques people are using to solve interesting problems. One such problem in todays day and age is figuring out what a picture describes. This can be useful for various reasons - from trying to figure out things happening in a crowded place to help law enforcers to being able to describe key features in a cancer cell. Since we were interested in this problem, we built a multi-faceted program that takes in images and produces a caption for the image using some added functionalities. This is an extension to the paper called *Show and Tell: A Neural Image Caption Generator* Paper by *Oriol Vinyals, Alexander Toshev, Samy Bengio and Dumitru Erhan* from Google.

As is summarized in the paper in the later sections, our program takes in an image and generates a caption by 2 key steps; using a Convolution Neural Network (CNN) of our choice and a Long and Short Term Memory (LSTM) in a Recurrent Neural Network. We used the attention mechanism to focus on the most relevant parts of the image to generate the best caption and then used a beam search to generate the best caption possible. The caption that we generated focused on being able to tell all the noticeable features within the image, including things in the background, in comprehensive and generally correct English. Finally, to analyze how well the program does, we used Google to find 28 different images and used a big demographic of people from various parts of the world to rate the images.

3. Related Studies

Since the problem that were trying to solve is well-known, there has been plenty of work done in this area before. The paper were following highlights some of them and theres other work available as well. There have been complex systems that compose of primitive recognizers combined with structured formal language, and these recognizers include And-Or graphs or similar logic systems. However, the problem with such a system is that its hand-designed, not very malleable and are very restricted to limited domains.

Using recent advances in recognition of objects, its attributes and locations within an image have also been used. A paper by Ali Farhadi et al [1] called Every Picture Tells a Story: Generating Sentence from Images uses scores to attach a descriptive sentence to an image. There have been other methods too where the detected objects have been matched to phrases containing those objects.

In almost all of these methods, including ours, the constant theme is that the picture is first analyzed using image processing techniques and then rendered into a natural language description with a text generating program.

4. Model

In order to implement the paper, we divided the work into two key parts; the first one is finding a good Convoluted Neural Network to generate the features needed that will be used as input to the Language Generating Recurrent Neural Network(LSTM) which is a type of a Recurrent Neural Network(RNN). This helps create a single network that generates descriptions of images.

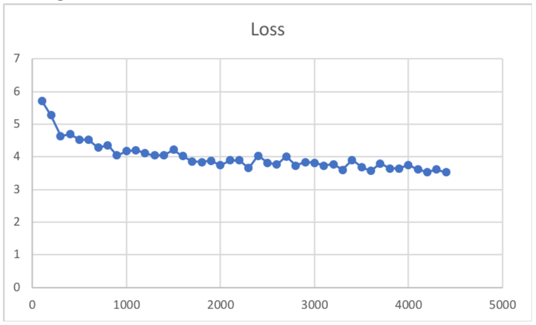
4.1. CNN

Since the paper we used tried many different CNNs including VGG, GoogleNet, ResNet, we decided to go with ResNet. We decided to use a pre-trained model from TorchVision which are interchangeable with another model pretty simply and easy to work with. As TorchVision tested their CNNs, the the best performing model was RestNet-152 with loss of 21.69 in Top-1 and 5.94 in Top-5 (https://pytorch.org/docs/stable/ torchvision/models.html). It does not mean that this model will be the best for this project. Since we could not compare its performance with other models, we just went with it. The CNNs primary job is to encode the image so that we can use it later for decoding via the LSTM, hence we decided to take out the last two layers so that the image is only encoded and not classified. Also, theres a fine tune method that is added to ensure that the calculation of gradients for the parameters can either be enabled or disabled. In this situation, we only fine tune the layers without the first one since it learns very fundamental information like lines, edges, curves that are best left untouched.

4.2. Loading Data and Loss

There are multiple images and caption datasets are available including MS Coco, Flickr30k, and Flicker8k that we could have used. Compared to others, MS Coco is the most user-friendly one. In the given timeline, we only could have worked around MS Coco, since it was the most convenient and didnt need us to get a release or submit any paperwork to use the other sets. Input images were resized to 256 by 256 for uniformity. The JSON files were handled through HDF5 module which helped a lot to deal with big JSON files.

For the loss function, we ended up using cross-entropy loss to measure the loss we had while training the model.



The loss curve for our training is shown above for one epoch

The reason why we only showed one epoch's loss is because we had not saved the loss and google collab was taking too long for our network while we tried to run it the day before the report was due, hence we ended up showing just one epoch.

4.3. Decoder

The decoder we used uses an LSTM to generate the captions. Using the output of the encoder, we first flatten that output which helps get rid of unnecessary reshaping afterwards. We initialize the hidden and cell state of the LSTM using the encoded image with a function to initialize the weights that has two linear layers. Interating through the LSTM layer and the images, we had to perform this manually as we had to execute the Attention mechanism between each decode step. Essentially, the decoder looks at each point in the encoded image to ”pay attention” to, then creates a word and based on that generates the next word. We use beam search to make this process better to not just look at the previous word to generate the next word.

4.4. Attention

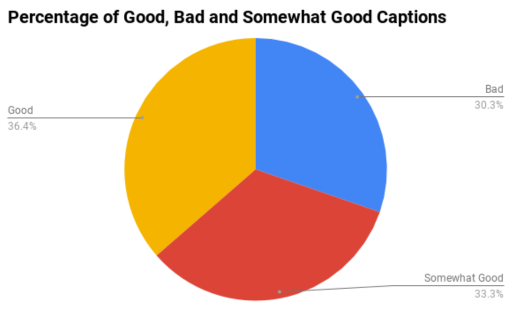
For the attention part, we relied heavily on the internet and various implementations and tutorials used. Eventually, we learned that it’s a simple network, with separate linear layers. These layers transform the encoded image and the output from the decoder through the attention size, which are then added and ReLu is used to activate. Finally, a third layer tranforms the results that we get to 1 demension, where we generate weights.

5. Results

In order to understand how our program faired, we decided to not use the conventional BLEU scores as used in all papers because we wanted to have a more human touch to our analysis. Since these pictures are primarily for humans to observe and use, we figured it would be the ideal route. Hence, we decided to make a Google form with some of the results and used a very diverse group of respondents to understand the success or failure of the program.

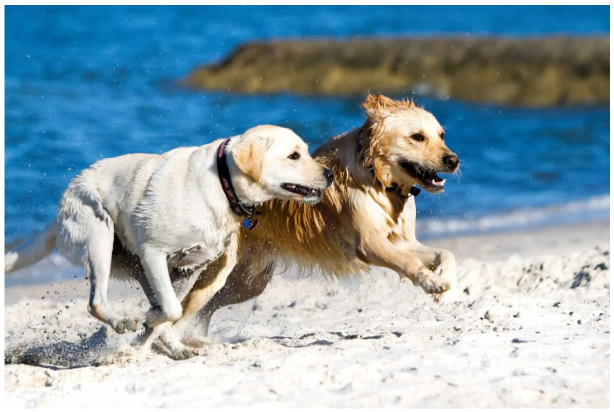
5.1. Evaluation

In the Google form that we distributed to people from various parts of the world (Romania, Bangladesh, Mongolia, and the United States), we asked people to rate the image captions as good, somewhat good or bad based on the accuracy of caption generated. Below are the images with the respective captions under good, somewhat good and bad captions generated.

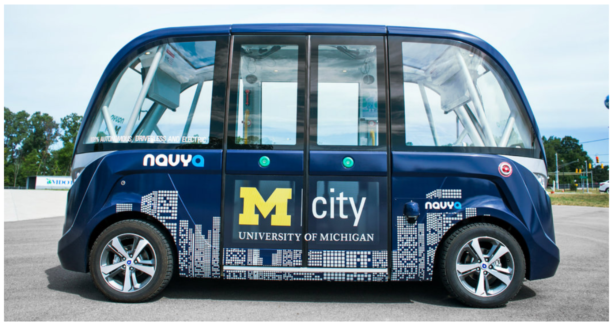


As seen from the image above, the captions generated are somewhat equally distributed among the three groups, with good captions being generated 36.4 percent of the time. In the section below, some examples of images with good captions, somewhat good captions and bad captions are included. We did not count the number of images that were classified as good, bad or somewhat good for this pie-chart. Instead, we counted the total number of observations per image that was recorded. For example, out of the 24 people who rated an image, if 5 people said the image was bad and 15 people said the image was good while the rest said otherwise, we used those numbers to give us a more accurate representation of what people thought of our captions.

5.2. Examples of Good Captions



Caption:A couple of dogs walking on the beach



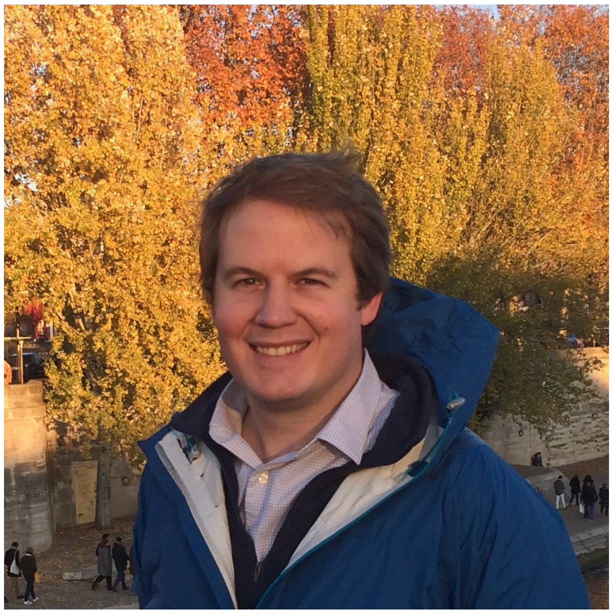
Caption:A blue bus parked in a parking lot

Caption:A young boy is eating a plate of food

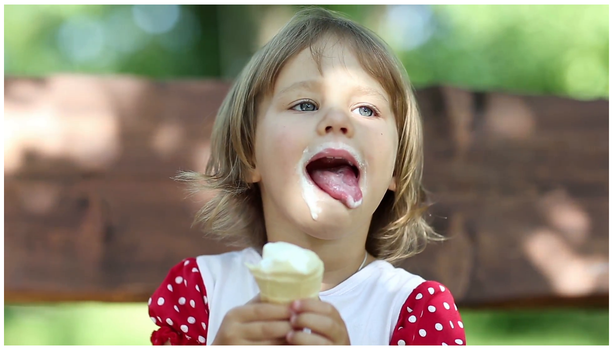
Caption:A public transit bus on a city street

Caption:A small white dog wearing a blue bowtie

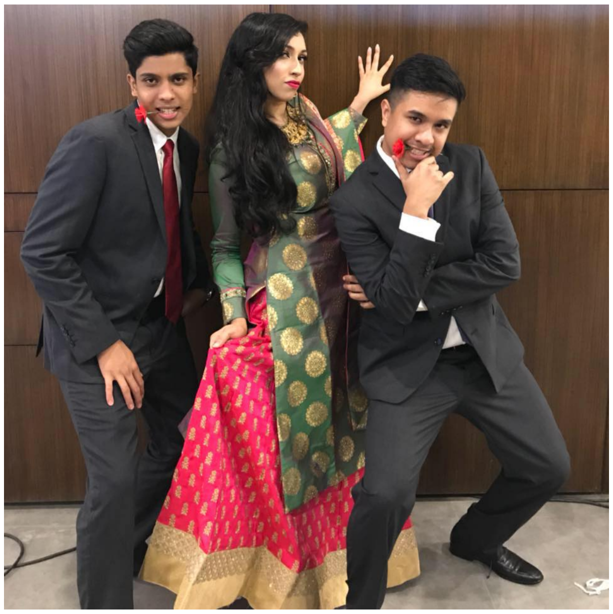
5.3. Examples of Somewhat Good Captions



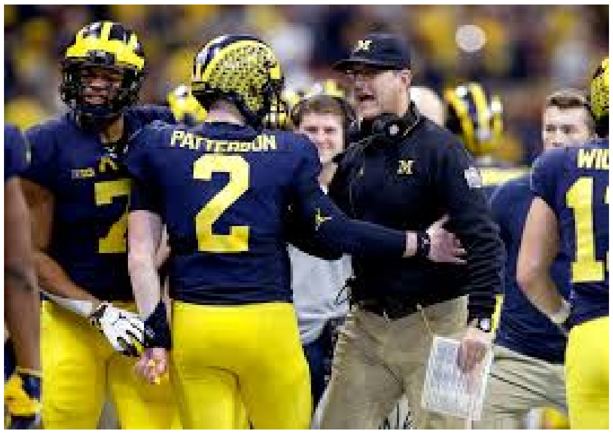
Caption:A smiling man in a blue shirt and tie



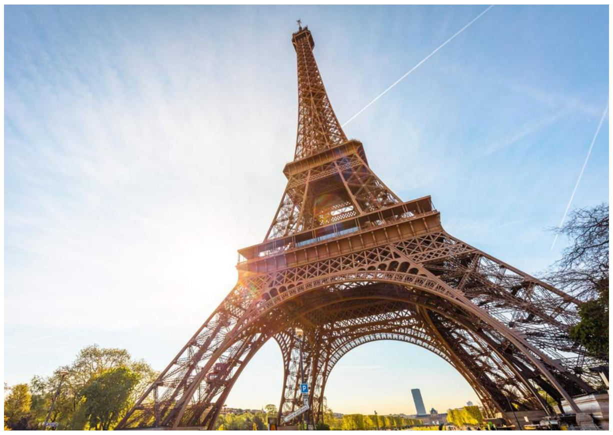
Caption:A young girl is eating a banana



Caption:A couple of people standing next to each other

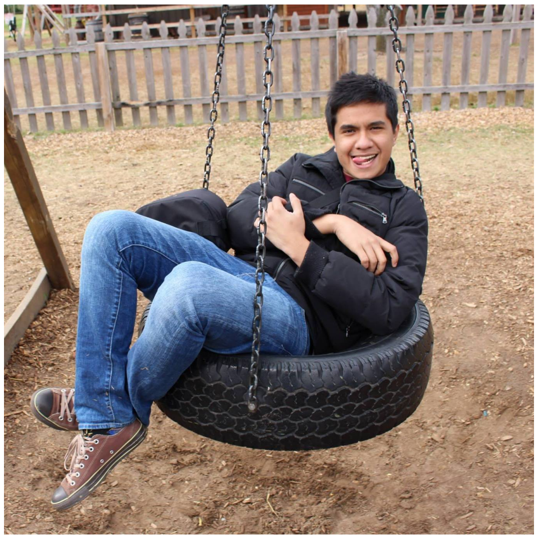


Caption:A group of people standing next to each other

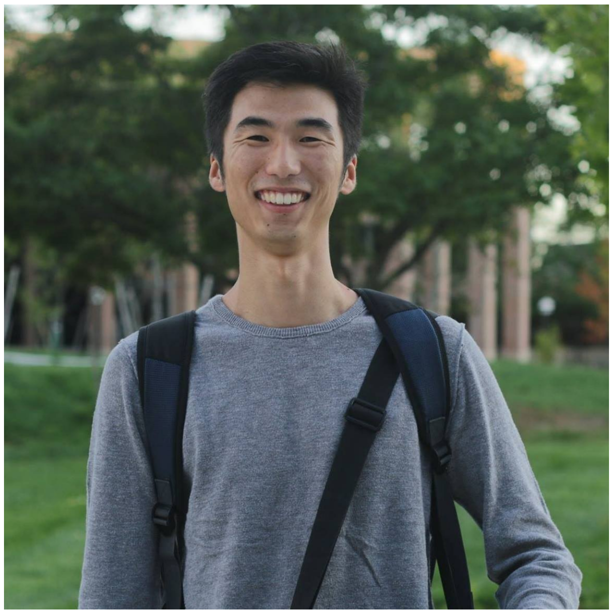


Caption:A tall clock tower with a sky background

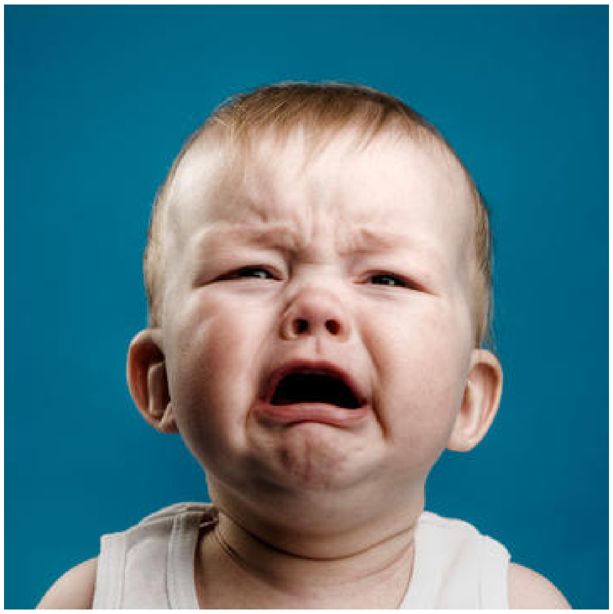
5.4. Examples of Bad Captions



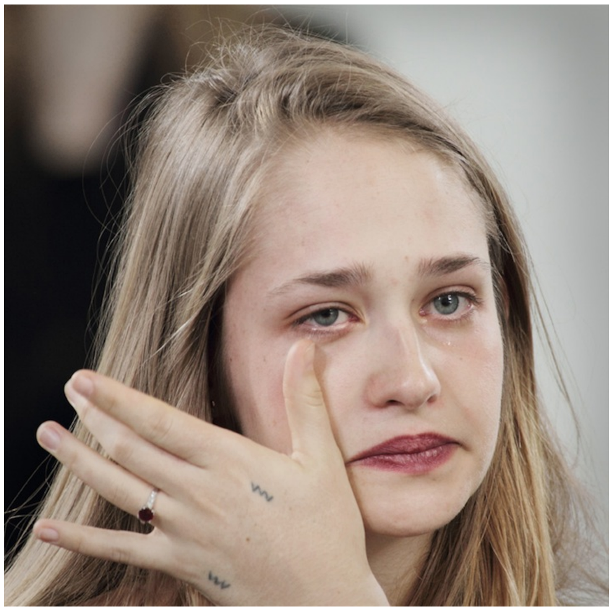
Caption:A man sitting on top of a baseball field



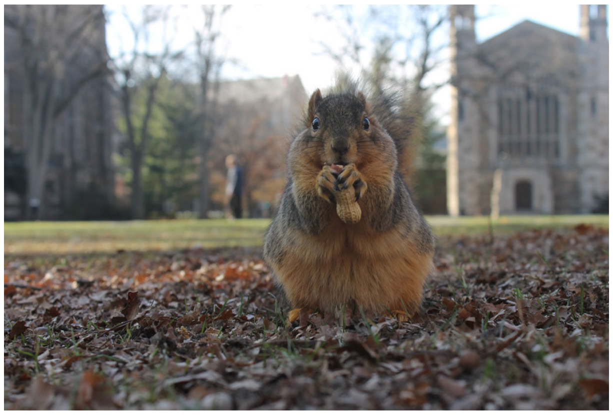
Caption:A woman standing in front of a tree



Caption:A young boy with a toothbrush in his mouth



Caption:A woman holding a cellphone in her hand



Caption:A small brown bear sitting in a field

6. Discussion

There were multiple difficult aspects to our project. One of the primary struggles we had was running our code on google collab and how much time it too to ensure we weren't using cached runs from the past. Hence, we eventually decided to not do the majority of the chunk of our coding in google collab.

Our training time is a lot and we don't utilize the test set or use BLEU scores as is used in several other papers. In our opinion, this may make this paper not up to industry standards. Hence, if we were to work on this project more, we would figure out ways to include that. Our test set was rather small. We would look at making it bigger in the future to include images related to Michigan.

Our captions in general are not able to detect movement. It would also be valuable to see how movement has been incorporated in image caption generating programs. Our key learnings in all of this has been using prebuilt networks are a good way to help save time and energy. We used up a lot of our time trying to come up with the best network from scratch, but realized that it was only half the problem. We also used a lot of documentation to understand how an LSTM works and we made sure to acknowledge these resources in our Acknowledgement section.

7. Conclusion

Overall, in conclusion, our program gave good captions only one third of the time. This shows massive room for improvement. It would be interesting to try different networks and run them to be able to see how the loss and results change over time. We were unable to incorporate these due to time constraints but these are key things to look at in the future.

8. Acknowledgement

We would like to thank several people for the help in this project. Firstly, the Professor and GSI's of the class for helping us learn key concepts and giving us feedback through the proposal and progress report submissions. While we made considerable changes from each step to the next, the steps were crucial in keeping us on our toes. Secondly, we would like to acknowledge that the dataset was taken from MSCOCO website. We also want to acknowledge papers we used for reference including the primary paper by Oriol Vinyals, Alexander Toshev, Samy Bengio and Dumitru Erhan. We referenced a paper written by the same authors called "Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge" and tutorials we found useful such as the medium tutorial (https://medium.freecodecamp.org/building-an-image-caption-generator-with-deep-learning-in-tensorflow-a142722e9b1f) and Image Captioning tutorial from Analytics Vidya.

9. References

1. Datasets from MSCOCO 2014
2. Test Images from Google
3. Show and Tell: A Neural Image Caption Generator Paper by Oriol Vinyals, Alexander Toshev, Samy Bengio and Dumitru Erhan
4. Medium tutorial: https://medium.freecodecamp.org/building-an-image-caption-generator-with-deep-learning-in-tensorflow-a142722e9b1f
5. Tutorial from Analytics Vidya: https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/