EECS442 Final Project Report

Qichen Fu*, Yige Liu*, Zijian Xie*
University of Michigan, Ann Arbor
{fuqichen, yigel, xiezj}@umich.edu

Abstract

We are very interested in how machines can automatically describe the content of images using human language. In order to gain a deeper insight of this computer vision topic, we decided to implement current state-of-the-art image caption generator ***Show, attend and tell: Neural image caption generator with visual attention*** *[12]. Our neural network based image caption generator is implemented in Python powered by Pytorch machine learning library. We have identified five major components in our pipeline:* ***(R1) data prepossessing****;* ***(R2) Convolutional Neural Network (CNN) as an encoder****;* ***(R3) attention mechanism****;* ***(R4) Recurrent Neural Network (RNN) as a decoder****;* ***(R5) Beam Search to find most optimal caption****;* ***(R6) Sentence Generation and evaluation****. BLEU-4 score is picked for evaluating the quality and accuracy of the generated caption. We evenly distributed the five components described above among our group and each member has made equal contributions to push the project forward. We have successfully finished the implementation of the all five components and are able to train our network on Google Colab (which provides free GPU resources). Our implementation of this image caption generator has achieved a very decent accuracy quantified by BLEU-4 score (15.5) which is very close to the result reported in the original paper (18.5). As we finished training the network and obtained a satisfying performance, we continue to visualize the attention mechanism.*

1. Introduction

Training computers to be able to automatically generate descriptive captions for images is currently a very hot topic in Computer Vision and Machine Learning. This task is a combination of image scene understanding, feature extraction, and translation of visual representations into natural languages. This project shows some great promises such as building assistive technologies for visually impaired people and help automating caption tasks on the internet.

There are a series of relevant research papers attempting to accomplish this task in last decades, but they face various problems such as grammar problems, cognitive absurdity and content irrelevance [5].

However, with the unparalleled advancement in Neural Networks, some groups started exploring Convolutional Neural Network and recurrent neural network to accomplish this task and observed very promising results [2]. The most recent and most popular ones include **Show and Tell: A Neural Image Caption Generator** [11] and **Show, attend and tell: Neural image caption generator with visual attention** [12]. While both papers propose to use a combination of a deep Convolutional Neural Network and a Recurrent Neural Network to achieve this task, the second paper is built upon the first one by adding attention mechanism. As shown in Figure 1, this learnable attention layer allows the network to focus on a specific region of the image for each generated word.

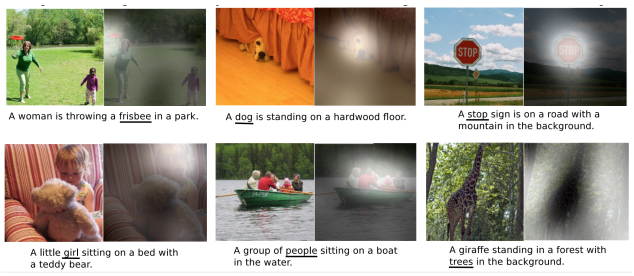


Figure 1. This figure is from Show, attend and tell visualization (adapted from [12])

2. Methodology and Architecture

We have written data preprocessing scripts to process raw input data (both images and captions) into proper format; A pre-trained Convolutional Neural Network architecture as an encoder to extract and encode image features into a higher dimensional vector space; An LSTM-based Recurrent Neural Network as a decoder to convert encoded features to natural language descriptions; Attention mechanism which allows the decoder to see features from a specifically

*Three authors have equal contribution and listed by alphabetic order

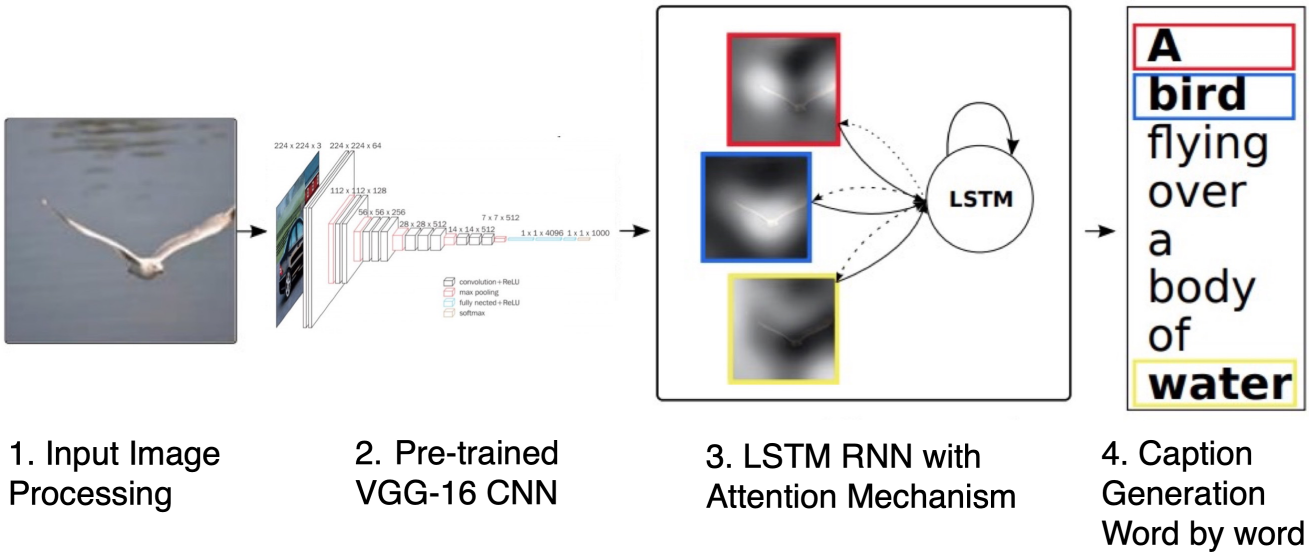


Figure 2. Show, attend and tell Architecture (adapted from [12])

highlighted region of the input image to improve the overall performance; Beam Search to figure out a caption with the highest likelihood. Each individual component of our generator pipeline will be discussed in detail below.

2.1. Data Sources

We have identified the three most commonly used image caption training datasets in Computer Vision research domain - COCO dataset [6], Flickr8k [3] and Flickr30k [8]. These datasets contain 123,000, 31,000 and 8,000 caption annotated images respectively and each image is labeled with 5 different descriptions. Currently, Flickr8k dataset which contains the least number of images is used as our primary data source over the other two due to our limited storage and computational power.

In addition, we also used sanitized Flickr8k data split open-sourced by Andrej Karpathy [4] as a part of our input dataset. This data split has converted original Flickr8k text data to lowercase, discarded non-alphanumerical characters and also split data into train, validation, and test subsets.

2.2. Data Preprocessing

Input data are composed of images and captions, and hence we need to pre-process both the images into proper format for CNN network and the text captions into proper format for RNN network. Since our image caption generator pipeline is leveraging pre-trained state-of-the-art CNN network (PyTorch vgg-16 [7] which will be further discussed in section 2.3), we need to transform images into correct format.

According to PyTorch documentation, pre-trained vgg-16 model expect that input images are normalized to within range [0, 1] and as 3-channel RGB images of shape ($3 \times H \times W$), where H and W are expected to be at least 224. Therefore, data preprocessing involves loading and resizing image data into ($N \times 3 \times 256 \times 256$) dimension and normalizing pixel value to be within range [0, 1] with mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225].

Besides, since PyTorch requires words be converted to numbers (indeces) in order to look up word embeddings, we construct a dictionary of words which is essentially a corpus of frequent words in training captions, including special tokens such as ⟨start⟩, ⟨end⟩, and ⟨pad⟩. Some words only appear less than 5 times across the whole training captions, and in that case it is all represented as ⟨unk⟩ (unknown).

Furthermore, captions are encoded with the dictionary described above and stored in a JSON file which can be fed to RNN model in a later stage.

2.3. Convolutional Neural Network (Encoder)

The encoder needs to extract image features of various sizes and encodes them into vector space which can be fed to RNN in a later stage. VGG-16 and ResNet is commonly recommened as image encoders. We chose to modify the pre-trained VGG-16 model provided by PyTorch library.

In this task, CNN is used to encode features instead of classify images. As a result, we removed the fully connected layers and the max pool layers at the end of the network. Under this new construction, the input image matrix has dimension $N \times 3 \times 256 \times 256$, and the output has dimension $N \times 14 \times 14 \times 512$. Furthermore, in order to support input images with various sizes, we added an adaptive 2d layer to our CNN architecture.

In our image captioning architecture, we disabled gradient to reduce computational costs. With fine-tuning, we might obtain a better overall performance.

2.4. Soft Attention Mechanism

Following the CNN, we built the soft trainable Attention mechanism introduced in Show, Attend and Tell: Neural Image Caption Generation with Visual Attention[12]. Attention mechanism tells the network which part of the image should be focused on for generating the next word in the description. We calculated the attention area through adding the encoder output and historical state, which is updated in each iteration.

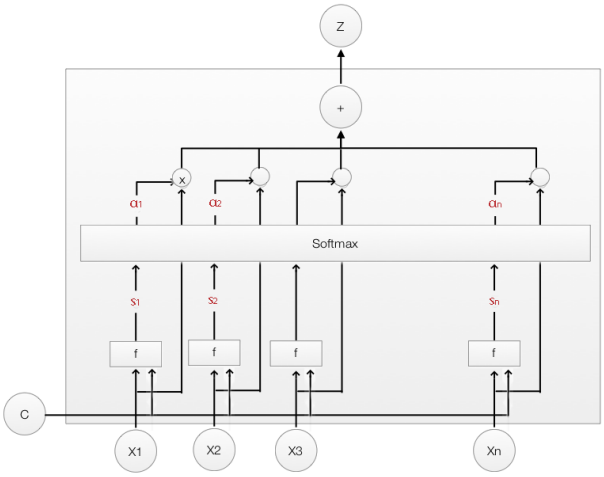


Figure 3. A demonstration of soft attention model [1]

In implementing the attention layer, we performed linear transformation on both the encoder output and historical state output. The linear activated outputs are summed up and the activation function is ReLU. The weight(alpha) and the attention area(applying alpha on encoder output) are returned.

Different from CNN without fine-tuning, the attention mechanism is trainable with back-propagation. The returned attention area is used later in decoder involving RNN.

2.5. Recurrent Neural Network (Decoder)

The decoder needs to generate image captions word by word using a Recurrent Neural Network - LSTMs which is able to sequentially generate words. The input for the decoder is the encoded image feature vectors from CNN and the encoded image captions produced in data preprocessing stage.

The decoder consists of an attention module designed and implemented by ourselves, an LSTM cell module and four fully connected layers provided by PyTorch library for the initialization of the states of LSTMcell and word dictionary.

When receiving the encoded images and captions, we first sort the encoded images and captions by encoded key length of images in descending order. We intend to only process the encoded images which have caption lengths greater than or equal to the number of iteration to increase efficiency and reduce training time.

In each iteration of LSTM network, we first put the historical state of LSTM and the encoded images into the Attention module to get the attention-masked images which have a specific highlighted area. Then we concatenate the embedded captions of all previous words and the attentioned images and feed them to the LSTM to get the next state of LSTM. Then, fully connected layers can predict the probabilities of current word embedding based on the current state and append it to the word embedding prediction matrix.

2.6. Loss Function

The nature of our RNN output is a series likelihood of words' occurances, and in order to quantify the quality of the RNN output, we propose to use Cross Entropy Loss. This is the most popular and effective measurement for the performance of a classification model whose output is a probability value between 0 and 1.

$$E_{\text{entropy}} = -\sum_{n}^{N} t_k^n \ln p_k^n \tag{1}$$

This equation is adapted from Machine Learning Cheatsheet Documentation [9], where N is the number of classes, t is either 0 or 1, and p_k^n is the predicted possibility that observation k is of class n.

2.7. Beam Search

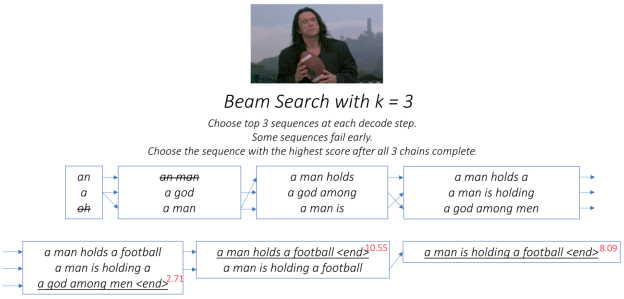


Figure 4. A demonstration of Beam Search in our sentence generation (Image taken from public Github Repository [10])

The Show and Tell paper [11] presents Beam Search as the final step to generate a sentence with the highest likelihood of occurance given the input image. The algorithm is a best-first search algorithm which iteratively considers the set of the k best sentences up to time t as candidates to generate sentences of size t + 1, and keep only the resulting best k of them, because this better approximates the probability of getting the global maximum as mentioned in the paper. We tried beam search size from 1 to 5 and the BLEU score

evaluation tends to be better under beam size 3 and 4. Thus, we chose 3 as the beam search size.

Figure 4 is a good demonstration diagram taken from a github repository [10]. This diagram shows how beam search avoids picking the highest probability word at each step (local maximum) and how it alternatively chose the sequence of words with highest overall probability score (global maximum).

The specific algorithm for the Beam Search is shown below:

Algorithm 1: K-Beam Search(K)

```
k ← K;
KBeamScores ← list(1);
bestKCaption ← list(" < begin > ");
bestKCaptionSequence ← [];
bestKScores ← [];
while True do
    predicted_possibility ← Pred(image, prevWords);
    KBeamScores ← TopK(KBeamScores +
      predicted_possibility);
    nextWords ← TopK(reverse_dic(KBeamScores));
    bestKCaption.append(nextWords);
    endIndex ← list(index(nextWord=" < end > "));
    if len(endIndex) > 0 then
        bestKCaptionSequence.
          append(bestKCaption[endIndex]);
        bestKScores.
          append(KBeamScores[endIndex]);
        k ← k - 1;
    end
    if k = 0 then
        break;
    end
end
bestIndex ← max(bestKScores);
return bestKCaptionSequence[bestIndex]
```

3. Experiments

There are several hyperparamters we adjusted in our experiments. Initially, in the training section, we used learning rate of 4×10^{-4}. However, under this condition, the oscillation of training loss is very significant. As a result, we used lr scheduler which multiplies 0.99 to lr after each epoch. The oscillation is reduced with the scheduler. In addition, without lr scheduler, the best BLUE-4 score we could achieve was around 13.

We also adjusted the batch size in our experiments. The initial batch size we used was 5. Given that flicker 8k consists of thousands of images, batch size of 5 was too small and therefore the loss didn't converge. As a result, we changed to batch size of 50. However, the maximum GPU limit offering by Colab is 12GB, and batch size of 50 made us exceed the GPU memory limit. Finally, we chose to use batch size of 32 – this number is within GPU limit, and is large enough for loss to converge.

4. Evaluation

While the pipeline has been implemented, it is unclear weather it is absolutely correct and can learn as expected. Therefore, the imminent task is to train our model and validate that it will learn properly. After the model is correct, **(R6)** is a another major component of this project and has yet to be finished. Sentence generation and performance evaluation is significantly important because it is the best way to present how well our caption generator can perform.

4.1. Evaluation Metrics

BLEU-1, BLEU-2, BLEU-3, and BLEU-4 (Bilingual Evaluation Understudy) is the most commonly reported metrics in evaluating the quality of text generation in natural language processing tasks. Here we chose 4-gram BLEU score(BLEU-4) as our primary evaluation metric. The next step is to also investigate other metrics (such as METEOR) mentioned in relevant papers [11, 12] and try to improve our model to match the papers' performance.

4.2. Model Performance via BLEU-4 score

Currently, our highest BLEU-4 score achieved within 25 epochs is 15.5 (BLEU-4 is within range 0 to 100). The BLEU-4 score reported in Show, Attend and Tell [12] on flickr8k dataset is 19.5, and hence our model is underperformed compared to the one achieved in state-of-the-art caption generator. As the progress report feedback suggests, we decided not to invest too much effort and time to just improve our BLEU-4 score to match what's reported in the paper, because these results may be tuned repeatedly by graduate students and have undocumented hacks.

5. Results

As mentioned in section 2.2, data is encoded to numbers via a dictionary of words. After the data is consumed by the pipeline, the output will also be in the encoded format which needs to be reversed back into English words in order to make sense to human. The other important thing is the output from RNN network is a series of likelihoods of words (likelihoods). Picking highest likelihood word at each decode step in RNN tend to yield a sub-optimal result. Instead, we have implemented Beam Search introduced in Section 2.7 which is a popular solution for finding optimal path for decoding natural language sentences. Some generated captions on test images are presented below.

Apparently, the network generated captions are not all perfect, some of which miss important information in the image and others have misidentified visual features. For

Figure 5. An example of an incorrectly generated caption

example, the top left image in Figure 5 is "A little girl in a white dress is playing a ⟨unk⟩". The ladder in the image is recognized by the CNN network but the RNN failed to generate the word "ladder". The use of ⟨unk⟩ suggests that training images has few number of images with ladders or the word "ladder" is very rare in training captions.

The other flaw we observed is the network sometimes fail to separate objects. For example, in the right-most image on the fourth row has two dogs racing. But our network fails to recognize two dogs, mainly because two dogs are too close and have some parts appearing connected in the image. Therefore, the network thinks there is one black and white dog.

Interestingly, we discovered that images of dogs catching Frisbee often have very accurately generated captions in test images and is likely due to large amount of training images are dogs jumping to catch a Frisbee. Also the pretrained CNN may also has lots of knowledge about what a dog looks like.

5.1. Generated Captions

As shown in Figure 6, Figure 7, Figure 8, and Figure 9, we have successfully generated mostly grammarly correct and human readable captions describing what is happening in the image. Most of the images correctly state what objects appear in the scene, count number of appearance and gives an intuitively correct verb to logically complete the sentence. Colors of the object and spatial relationships between object are also well captured. For example, the bottom right image has a caption "A man in a red shirt is standing on a grassy hill". Objects in the images are identified as "A Man", "Shirt" and "Grassy hill". Then CNN is also able to capture properties of objects such as shirt's color is "red". Lastly, RNN and Attention Mechanism will connect these words logically by conjunctions into a valid sentence.

In addition, we have also included more results we generated at the end of this report - Figure 14 and Figure 15.

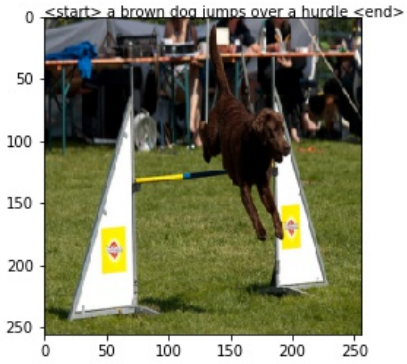


Figure 6. Generated Caption Example1: A brown dog jumps over a hurdle

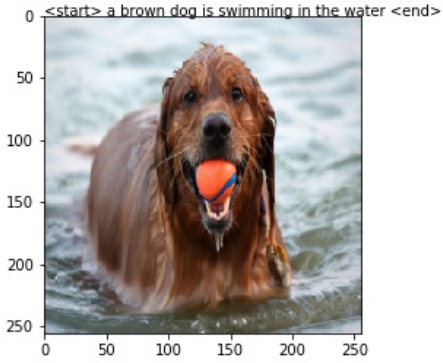


Figure 7. Generated Caption Example2: A brown dog is swimming in the water

Figure 8. Generated Caption Example3: A man and a woman are sitting in a fountain

5.2. Attention Mechanism Visualization

Visualizing attention mechanism allows us to understand which part of the image is being focused when generating a specific word, which is essential for improving the network's scene understanding capability. We implemented

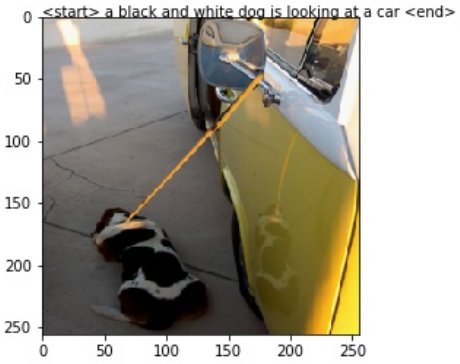
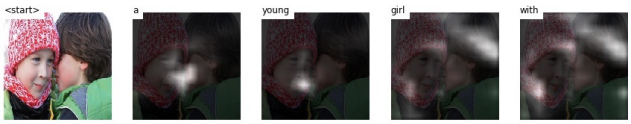


Figure 9. Generated Caption Example4: A black and white dog is looking at a car

our visualization by adding a semi-transparent mask on top of the original image at each RNN node (each word), such that the bright region denotes the focus area and dark region has little influence on the generated word.



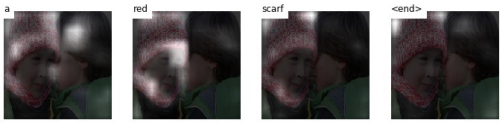


Figure 10. Attention Visualization: A young girl with a red scarf

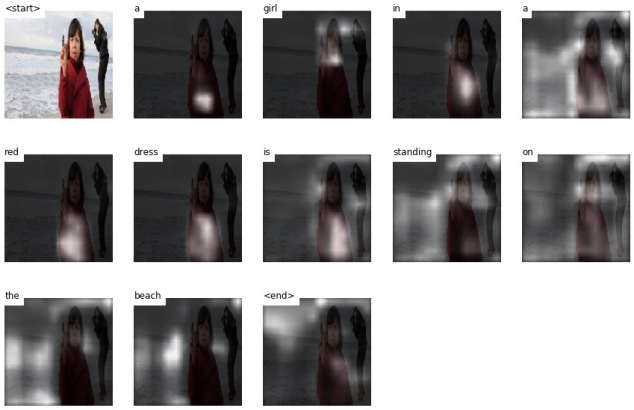


Figure 11. Attention Visualization: A girl in a red dress is standing on the beach

As we can see in Figure 10, Figure 11 and Figure 13, the network has different focuses when generating different words.

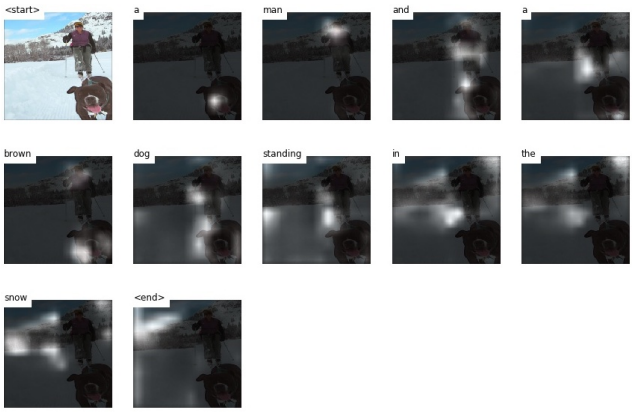


Figure 12. Attention Visualization: A man and a brown dog standing in the snow

Figure 13. Attention Visualization: A surfer rides a wave

6. Conclusion and Future Work

Automatically image captioning is far from mature and there are a lot of ongoing research projects aiming for more accurate image feature extraction and semantically better sentence generation. We successfully completed what we mentioned in the project proposal, but used a smaller dataset (Flickr8k) due to limited computational power. There can be potential improvements if given more time. First of all, we directly used pre-trained CNN network as part of our pipeline without fine-tuning, so the network does not adapt to this specific training dataset. Thus, by experimenting with different CNN pre-trained networks and enabling fine-tuning, we expect to achieve a slightly higher BLEU-4 score. Another potential improvement is by training on a combination of Flickr8k, Flickr30k, and MSCOCO. In general, the more diverse training dataset the network has seen, the more accurate the output will be. We all agree this project ignites our interest in application of Machine Learning knowledge in Computer Vision and expects to explore more in the future.

7. Acknowledgement

As mentioned in the README of our published github repository, our project's methodology and architecture de-

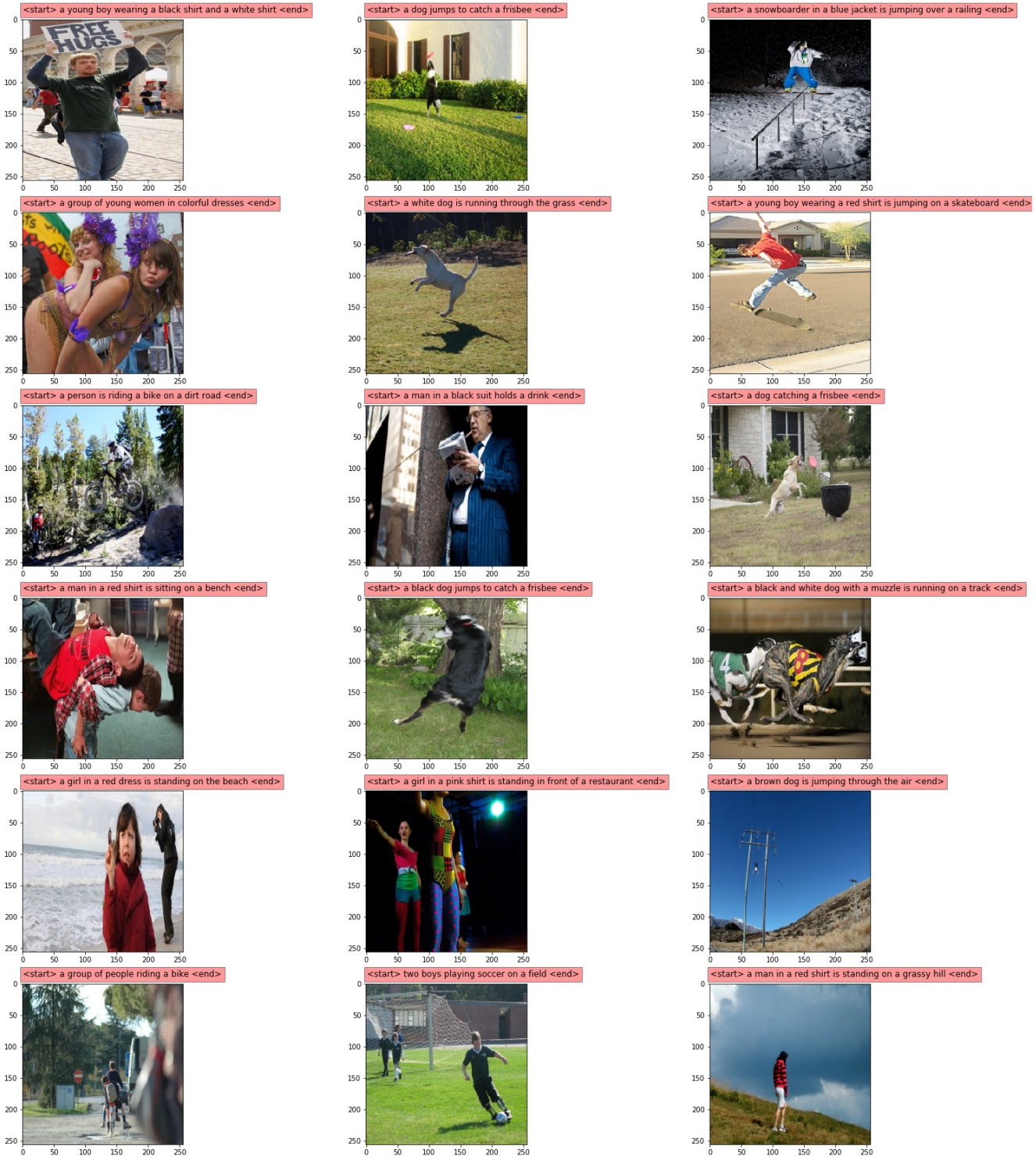


Figure 14. Image and caption pairs generated by our caption generator

sign are adapted from the Show, Attend and Tell Paper [12]. We built the whole project from scratch and hence had encountered many implementation obstacles along the way, many of them are related to PyTorch syntax and proper usage. We were able to overcome these issues with helps from PyTorch developer community, Stack Overflow and by referring to some relevant public Github repositories.

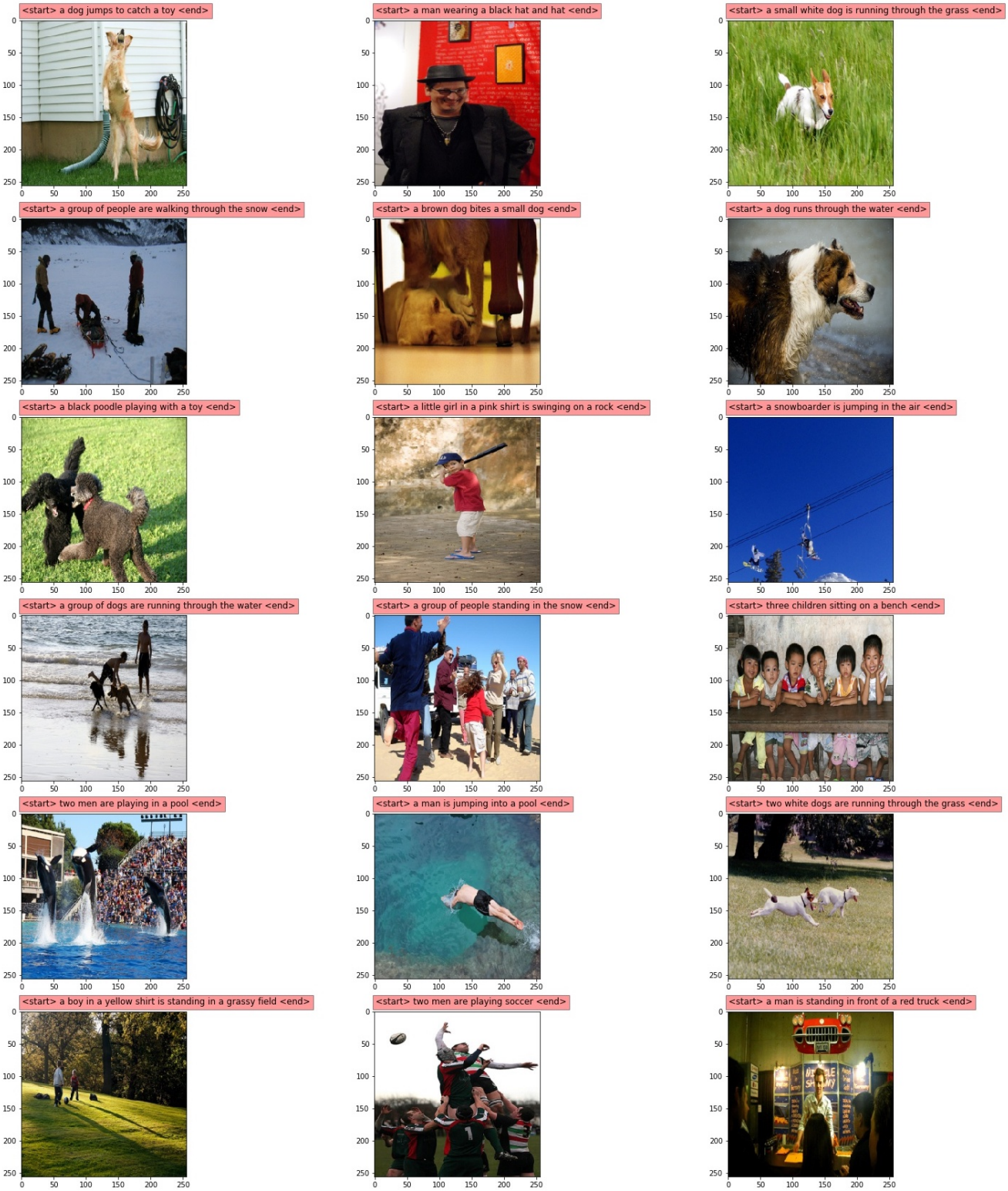


Figure 15. Image and caption pairs generated by our caption generator

References

[1] Jonathan Hui Blog. https://jhui.github.io/2017/03/15/Soft-and-hard-attention.

[2] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[3] Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *J. Artif. Int. Res.*, 47(1):853–899, May 2013.

[4] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676, Apr. 2017.

[5] Polina Kuznetsova, Vicente Ordonez, Alexander C. Berg, Tamara L. Berg, and Yejin Choi. Collective generation of natural image descriptions. pages 359–368, 2012.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[7] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. pages 730–734, Nov 2015.

[8] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *Int. J. Comput. Vision*, 123(1):74–93, May 2017.

[9] PyTorch. `https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html`.

[10] Sagar Vinodababu. `https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning`.

[11] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

[12] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.