CSC413 - Final Project

Chloe Nguyen
nguy2330

Kimberly Hau
haukimbe

Zoha Rehan
rehanzoh

Abstract

This project aims to recreate the results from the Show, Attend and Tell paper, which describes an model tasked with taking in a vast variety of images representing objects and scenes with multiple attributed captions as input, and generating appropriate captions describing the image. We implemented this model using different hyperparameter values, as well as comparing it to other models using different decoder implementations. The results of each model are evaluated using the accuracy of the generated captions on a test dataset, as well as BLEU score.

1 Introduction

Classifying objects from images is a widely explored area of machine learning. Inspired by the content covered in class regarding classifying multiple items within an image, we decided to extend this by using machine learning models to generate captions describing the image itself.

In this project, we aim to recreate the results from the Show, Attend and Tell paper, which describes an model tasked with taking in a vast variety of images representing objects and scenes with multiple attributed captions as input, and generating appropriate captions describing the image [1]. The model in the paper uses an encoder-decoder architecture with attention to convert raw images into a series of feature vectors, and then generate a sequence of encoded words describing the image. The model is trained on the MSCOCO and Flickr8k datasets, and evaluated using BLEU-4 score.

We recreate this model and perform hyperparameter sensitivity analysis on the size of the embeddings and context vectors generated by the attention mechanism. We then re-implemented this model without attention, as well as replacing the long short-term memory (LSTM) network with a regular recurrent neural network (RNN) and trained it both with and without attention.

The Github repo can be found here: Github

2 Related Works

Generating natural language descriptions from visual input is a problem that has been studied for both video, and imaging. Many deep learning models have implemented the encoder-decoder architecture for this task, as they are useful in translating the input from a visual medium to natural language. Thus, there are several works with similar propositions.

One example is a model proposed by Vinyals et al. (2014) in the paper "Show and Tell: A Neural Image Caption Generator" [2]. Their proposed encoder-decoder framework does not use attention, but rather relies solely on the past memory of the LSTM decoder to guide each time step. As a result, their bilingual evaluation understudy (BLEU) scores were not as impressive as those in the Show, Attend and Tell paper, although it presents a good comparison to the paper, and the importance of attention.

Another proposed encoder-decoder model was by Alikhani et al. (2020) [3]. Unlike other image captioning models, the authors aimed to increase the robustness of the model against inconsistent

Preprint. Under review.

image-text relations. This model fed extracted image features and image classifications into an encoder, whose output was the input for a decoder which created coherence-aware captions. The results were also judged by coherence-related metrics, rather than started evaluation metrics such as BLEU.

3 Method

3.1 Encoder

The encoder receives input as images and returns an encoded representation of each image in the form of a matrix of annotation vectors. The encoder is a convolutional neural network (CNN), and the goal of the encoder is to extract and represent features from the image.

$$a = a_1, ..., a_L a_i, \in \mathbb{R}^D \tag{1}$$

3.2 Decoder

The decoder is a long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997) [4]. The LSTM operates using input, output and forget gates, and makes calculations using the previous output and hidden state. The inputs to the LSTM are an embedding matrix, a context vector, and the previous outputs. The context vectors are generated from an attention model, and represents which parts of the image is relevant, so that the LSTM focuses on different areas on each time step.

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} T \begin{pmatrix} Ey_{t-1} \\ h_{t-1} \\ \hat{z}_t \end{pmatrix} \tag{2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{3}$$

$$h_t = o_t \odot tanh(c_t) \tag{4}$$

c_t and h_t are the memory and hidden state, respectively, while o_t is the output. T represents a linear transformation of the inputs, which are Ey_{t-1}, the embeddings, the previous hidden state, and $\hat{z}_t$, the context vectors. The initial memory and hidden states are initialized based on the mean of the annotation vectors. The gates are calculated through linear transformations of the inputs followed by activations. The memory state and hidden state are calculated using elementwise multiplications of the gates.

For our second model, we implemented the decoder using a regular RNN instead of an LSTM, and ran both versions of the model with and without attention. Attention was removed by setting all of the context vectors equal, such that the encodings have equal weights on each time step.

$$h_t = tanh\left(T \begin{pmatrix} Ey_{t-1} \\ h_{t-1} \end{pmatrix}\right) \tag{5}$$

3.3 Attention Mechanism

Show, Attend, and Tell implements attention using both the stochastic “hard” model and the deterministic “soft” model. In our implementation of the model, we use “soft” attention to generate the context vectors, given the annotation vectors (Bahdanau et al., 2014) [5]. Given the encoder outputs, a linear transformation and applied softmax give the weights $\boldsymbol{\alpha}$. The context vectors are calculated as weighted annotation vectors.

$$f_{att}(a_i, \alpha_i) = \Sigma_i^L a_i \alpha_i \tag{6}$$

Finally, the annotation weights are used in calculating the loss, as part of "doubly stochastic attention". This is a regularization technique to encourage the decoder to pay attention to all parts of the model over the time steps. Loss is calculated as the penalized negative log-likelihood.

Table 1: Validation accuracy and BLEU-4 metrics for changed embedding and attention dimension sizes, trained on MSCOCO.

Embedding Size	Attention Size	Validation Accuracy	BLEU-4
512	512	74.56	0.2104
256	512	74.60	0.1983
512	256	73.03	0.2004

4 Experiments and Results

As indicated by our professor on Piazza, we are free to use open-source code for this project. Thus, we referenced several open-source works when implementing our model in PyTorch, as well as forking some implementations to build upon [6][7][8].

4.1 Data

The experiments were run on either the Flickr8k dataset or Microsoft's COCO dataset. Flickr consists of 8000 images and up to 5 captions for each image. MSCOCO consists of 82,783 images and at least 5 captions for each image and used the Karpathy test split. Hyperparameter sensitivity analysis was performed on the MSCOCO dataset, while experimenting with a second model was performed on Flickr.

To begin, we performed basic cleaning on text captions to handle lower/upper case discrepancies, special characters and numbers. Instead of accounting for every unique word in our train captions, our vocabulary will consist of all unique words that have occurred at least 3 times. This helped to reduce the influence of outliers or one-off words on our algorithms. We then kept track of the start and end of captions by including the tokens '<start>' and '<end>'. Additionally, captions were all padded to have a fixed size of 100.

4.2 Model Details

For our encoder, we used ResNet-101 pretrained on ImageNet, without finetuning. The decoder accepts these annotation vectors along with the encoded captions and caption lengths, and generates embeddings from the encoded captions, as well as context vectors from attention. Attention is implemented as deterministic "soft" attention, where the encoder output is run through linear layers and a softmax to calculate weights. All of these values are then run through an LSTM cell with uniform initialized weights, for several time steps. On each time step, the decoder generates a new word based on the previous generated word and the attention-weighted encoding.

Our training loop uses a batch size of 32, cross entropy loss and doubly stochastic regularization. The decoder is optimized using Adam with a learning rate of $4e^{-4}$, and employs dropout for regularization. The training loop is performed for 2 epochs due to time and GPU constraints with training the model. When performing inference on the decoder output, we implemented Beam Search using a linear layer to ensure that the best possible sequence is formed, rather than the greedy choice. We performed hyperparameter sensitivity training on the sizes of the embedding and attention vectors.

4.3 Training and Validation

For hyperparameter sensitivity analysis, the Show, Attend and Tell model was trained for 2 epochs on the MSCOCO dataset using different values for embedding and attention sizes. For experimenting with different implementations of the decoder, each model was run with and without attention on a training dataset of 400 images from Flickr, followed by validation on 100 images. Each model was trained for 3 epochs using our original hyperparameter values.

4.4 Results

To evaluate our models, we used top 5 accuracy as well as BLEU-4 score.

Table 2: Validation accuracy and BLEU-4 metrics for different decoders, trained on a subset of Flickr8k.

Model	Validation Accuracy	BLEU-4
LSTM with attention	53.10	0.0914
LSTM without attention	52.34	0.0894
RNN with attention	54.32	0.0913
RNN without attention	54.81	0.0934

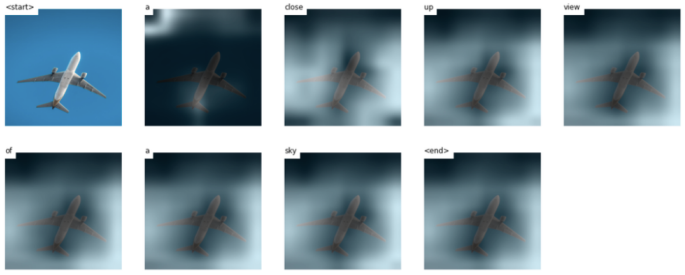

Figure 1: inference run on an image of a plane in the sky.

In Table 1, we provide a summary of our results in performing hyperparameter sensitivity analysis on the size of the embeddings and context vectors, run on the full MSCOCO dataset. From the results, halving the embedding vector size has very little effect on accuracy or BLEU-4 score of the model. Halving the size of the context vectors caused a decrease in both accuracy and BLEU-4 score, since each time step would look at a wider area.

Figures 1 and 2 show the result of inference of the first model on an image of an airplane in the sky, and an image of cats on a counter. Respectively, they are examples of a non-busy vs. a busy image, and the model generates a more accurate caption for the non-busy model. We can also see that attention has a better focus on objects in the first Figure, compared to the second.

In Table 2, we summarize our results from training the 4 variations of the model. They have similar validation accuracies as well as BLEU-4 scores, indicating that the presence of the LSTM and attention are not largely important. However, given that training was only for 3 epochs, it is possible that they will converge to different scores when trained for more epochs.

4.5 Limitations

Limitations of our experiment include paring down our training dataset to only 400 images and our validation set to 100 images for the Flickr dataset, as well as running the training for only 3 epochs at most. This was due to GPU limitations in Google Colab, as well as each batch of 32 images taking several minutes to run, resulting in between 1 and 3 hours per epoch. As a result, our presented results have a poor accuracy and BLEU-4 score. We expect that rerunning our experiments using the full dataset and more epochs would result in respectable values for accuracy and BLEU-4.

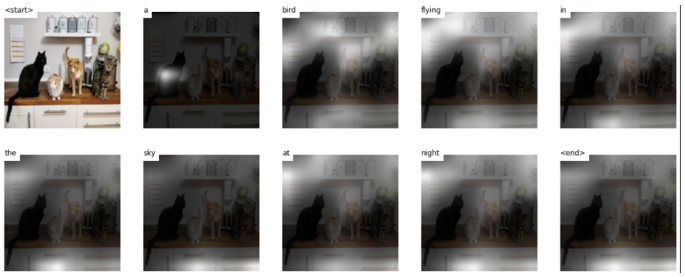

Figure 2: inference run on an image of cats on a counter.

5 Conclusion

We recreated the results of the paper Show, Attend and Tell: Neural Image Caption Generation with Visual Attention by Xu et al. (2015), on the Flickr8k dataset [1]. We performed hyperparameter sensitivity training on the size of the embedding dimension in the decoder, the size of the context vector through the attention dimension, and on the size used in beam search when performing inference. For our second model, we replaced the decoder LSTM with a regular RNN, and ran both versions of the decoder with and without attention. Due to the time required for training, our experiments ran for at most 3 epochs on a smaller dataset.

Contributions

Chloe Nguyen - Data loading, cleaning and prepping. Models: RNN w/ attention, LSTM w/ attention.

Zoha Rehan - Models: LSTM w/ attention, LSTM w/o attention.

Kimberly Hau - Models: RNN w/ attention, RNN w/o attention.

References

[1] Xu, Kelvin and Ba, Jimmy and Kiros, Ryan and Cho, Kyunghyun and Courville, Aaron and Salakhutdinov, Ruslan and Zemel, Richard and Bengio, Yoshua. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. arXiv:1502.03044, February 2015.

[2] Vinyals, Oriol and Toshev, Alexander and Bengio, Samy and Erhan, Dumitru. Show and Tell: A Neural Image Caption Generator. arXiv:1411.4555, November 2014.

[3] Alikhani, Malihe and Sharma, Piyush and Li, Shengjie and Soricut, Radu and Stone, Matthew. Clue: Cross-modal Coherence Modeling for Caption Generation. arXiv:2005.00908, May 2020.

[4] Hochreiter, S. and Schmidhuber, J. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.

[5] Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, September 2014.

[6] Vinodababu, Sagar (2020). A PyTorch Tutorial to Image Captioning [source code]. https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning?fbclid=IwAR3kwMkjvABsMnetaBoGIgrYMToHADIXqM8S4OdX29h9FwuRv0piSaW3y-8

[7] Gala, Jay (2021). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention [source code]. https://github.com/jaygala24/pytorch-implementations/blob/master/Attention

[8] Arora K., Raj A., Goel A., Susan S. (2022). A Hybrid Model for Combining Neural Image Caption and k-Nearest Neighbor Approach for Image Captioning [source code]. https://github.com/rizal-rovins/hybrid-image-captioning-model