Neural Image Captioning

Justin Nguyen
Khoury College of Computer Sciences
Northeastern University
Boston, MA 02115
nguyen.jus@northeastern.edu

Abstract

We reimplemented the neural image captioning model with soft attention from Xu et al. [4] and evaluated it on the Flickr8k dataset [1]. Our model reaches a final individual BLEU-4 score of 16.4 and a cumulative BLEU-4 score of 22.1. Xu et al. report a BLEU-4 score of 19.5 without specifying whether it is individual or cumulative. Our experiments suggest that our model was successful in its task of captioning images and reached its peak performance given the current dataset without overfitting the data. More data may increase the model's expressivity and scores.

1 Introduction

Neural image captioning is the task of using neural architectures to map an image to a sentence from

$$\mathbb{R}^{C \times W \times H} \rightarrow \mathbb{R}^{K}$$

where C is the number of channels of the image (3 for color, 1 for grayscale), W is the width of the image, H is the height of the image, and K is the length of the generated sentence. In the context of this paper, we are essentially learning a mapping between a 2-dimensional color image of dimensions $W \times H$ to a K-lengthed sentence which best describes the image.

This problem statement naturally leads to the machine learning paradigm of encoder-decoder networks, in which the encoder learns to generate a latent representation from the input and the decoder learns to generate an output from that latent representation. On the encoder end, we process an image and generate a set of features which describe it. On the decoder end, we take those features and describe them via natural language. Thus, the problem breaks down into two steps, both of which have deep learning solutions published recently: perform feature extraction on an image, then perform text generation on those features.

Feature extraction can be done using deep convolutional networks. In short, we pass sets of filters over an image to generate new representations of the image. Then we pass consecutive layers of filters over those representations to generate new representations. These representations are a lower-dimension latent representation of the original input image. The filters' weights are learned via gradient descent.

Text generation can be done using RNNs, LSTMs, or transformers. In this paper, the authors used an LSTM. Briefly, LSTMs are deep neural networks which propagate information through the layers selectively based on gates. These are particularly useful because they can store time-based information in their cell states. This means that it can store some context of a sentence at a given position, and then use that context to generate next words. The hidden and cell state weights of the LSTM are learned via gradient descent.

The authors also used attention mechanism in their model. In essence, attention mechanism takes in a set of inputs, such as features or a caption, and output weights for each sub-input which correspond to how much the model should "focus" on each sub-input.

Preprint. Under review.

To assess correctness, we also must consider formulating this as a supervised learning task. In other words, the gradient needs some measure of correctness, a loss, in order to move in the correct way. The loss function is dependent on how the problem is formulated. Xu et al. proposed the captioning task essentially as a time-series classification task. Each position of the caption represents a single timestep. At each timestep, we generate a vector with size equal to the total number of words in our vocabulary. The index of the maximum of this vector corresponds to the index of the most likely word in the vocabulary. Note that this means that the model will never be able to predict words out of its vocabulary. As this is a classification task, we use cross entropy loss between the predicted output classes and the target classes at each timestep. Furthermore, the authors proposed an additional loss term which corresponds to the attention portion of the model. The proposed loss function is

$$L_d = -\log(P(\mathbf{y} \mid \mathbf{x})) + \lambda \sum_{i}^{L}(1 - \sum_{t}^{C} \alpha_{ti})^2$$

Finally, we need some way to produce captions without consulting reference captions. Normally, the decoder predicts a new caption by passing a reference caption through the LSTM. However, test images may not have reference captions. This can be done by consecutively passing in an incomplete caption of previous words and having the network predict the best word for the next index. Instead of applying a greedy selection approach, the authors used beam search (but did not specify in their paper). In short, beam search will produce the top k words at a given timestep, and then base its next predictions on those k words. Then, it picks the top k set of words and uses those k sets as previous captions.

2 Method

We reimplemented the architecture proposed by Xu et al. as best as we could. Some architecture decisions were not apparent in the publication, so we consulted the authors' code [3] or other open-source references. Finally, other architecture decisions were made by us where we saw made sense. Consequently, the results will be slightly different than published.

2.1 Dataset and Data Pre-processing

We use the Flickr8k dataset [1], which contains 8091 images captioned with 5 expert captions each for all our experiments. We used a random 80:20 split for train and validation datasets.

All input images were resized to 224×224 and normalized to

$$\mu = (0.485, 0.456, 0.406)$$
$$\sigma = (0.229, 0.224, 0.225)$$

over the $C \times H \times W$ dimensions respectively. This is consistent with the VGG architecture and the ImageNet distribution.

We built a vocabulary containing every word that appears in the set of captions. This vocabulary maps each word to an index and contained a total of 8092 words. For consistency, we set the max vocabulary size to 10000, which means that there are indices which map to nothing and will yield the "unknown" token. We preprocessed the vocabulary by setting all words to lowercase, but did not conduct any spelling checks or pruning of rare words.

For all captions, each word was mapped to the position in the vocabulary. Essentially, we passed around an index array for computation. We added a start tag at the beginning of each caption and an end tag at the end of each caption. We zero-padded these captions to a max caption length of 20, which was arbitrarily chosen. Any captions beyond this length were ignored. This parameter exists in the original paper, but its value was not apparent.

2.2 Architecture

2.2.1 Encoder

The authors did not specify which VGG model they used for the encoder. It is implied that they used one of the VGG-19 models since their final feature extractor block contained four convolutional

layers. We use a pretrained VGG-19 with batch normalization model for the encoder, but we also considered that there exists a VGG-19 without batch normalization.

As with the authors, we removed the final three layers of the model which are used for averaging and classification. We considered keeping the third-to-last maxpool layer as it empirically increased performance on smaller tasks but, ultimately, we removed it for consistency.

The encoder takes in an input of size $3 \times 224 \times 224$ and outputs a representation of size $512 \times 14 \times 14$. As with the authors, we flattened the pixel dimensions to get a representation of size 512×196.

2.2.2 Decoder

We use a decoder structure identical to that proposed by the author, but perhaps with different number of neurons. The authors did not specify the number of neurons for each layer. For simplicity, we set all layers to a static 512 neurons, which corresponds to the number of neurons in the feature extractor's final layers.

The decoder takes in the output of the encoder of size 196×512 and a caption of size 20. The caption is passed through an embedding layer.

As per the authors, we initialized hidden and cell states via a forward pass through the h_0 and c_0 layers. The authors suggested that they used a multilayer perceptron for these initializations, but we used a single layer instead. The input of these was simply the mean of the encoder features.

For each position in the caption, we computed attentions for the features and the hidden states using two separate layers. These attentions were combined, passed through a ReLU nonlinearity, passed through another combined attention layer, and softmaxed to get probabilities, alpha, for each feature, which we use as weighting factor. The product between alpha and the feature attention yields an alpha-weighted-context at each feature dimension. We also computed a gating scalar term, beta, which is the sigmoid-normalized output of the hidden state through another layer. The alpha-weighted-context was scaled by this beta.

The LSTM takes the caption embedding and alpha-weighted-context as input and recomputes the hidden and cell states.

Finally, we receive a caption output of size 20×10000 by passing the hidden state through an output layer. We also employ dropout with probability $p = 0.5$ on the hidden state. We considered removing dropout, but decided to keep it to be consistent with the authors. This output corresponds to scores of words at each position in the caption.

2.3 Training

We trained our model over 26 epochs on a random 80% split of the data using the Adam optimizer with initial parameters $lr = 10^{-3}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. These correspond to the default Adam parameters in PyTorch. Notably, the authors used RMSProp instead for their Flickr8k experiments. Instead of early stopping as suggested by the authors, we decayed the learning rate by a factor of 0.1 if the validation loss stopped decreasing over a period of 2 epochs. This decay continued until the learning rate reached 10^{-8}.

This training took approximately 10 hours on a GTX1080 GPU, but convergence may have been reached after around 3 hours.

3 Results

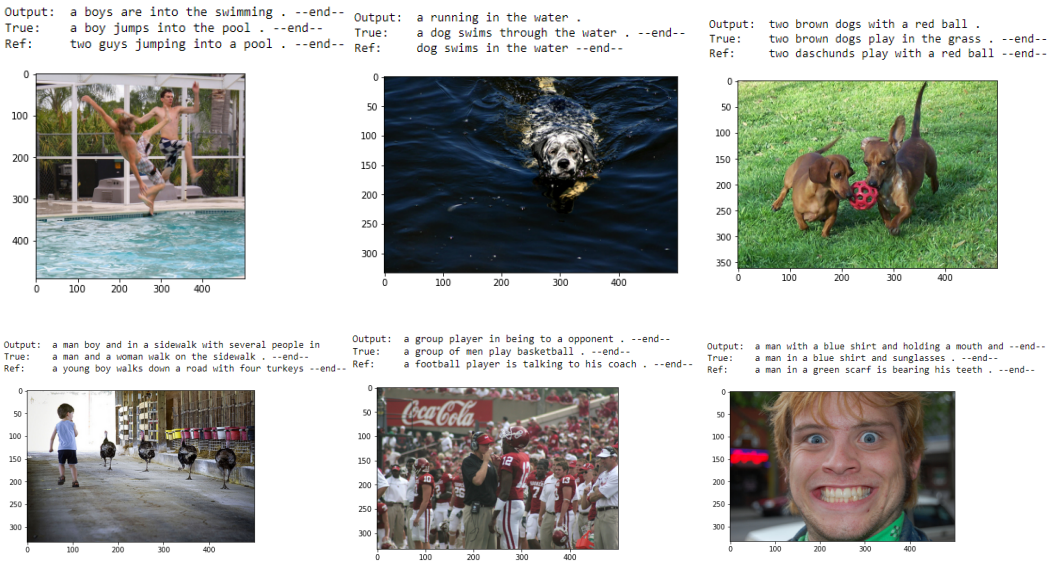
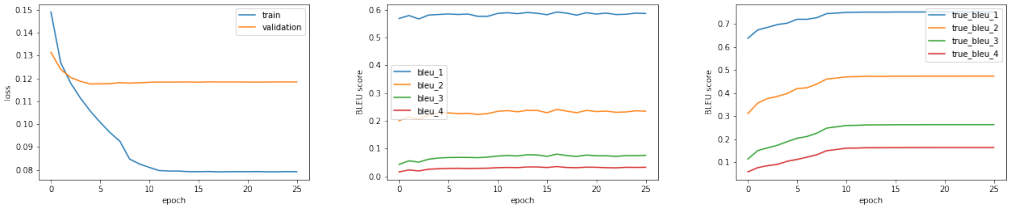


Figure 1: Example captions produced by the model. The top row shows cases which we consider were successful captions while the bottom row shows cases which we consider were failures. "Output" refers to the output of the model given the reference caption. "True" refers to the model prediction without a reference. "Ref" refers to one of the five reference captions. We present more example captions in the appendix.



(a) Train and Validation Loss (b) Output BLEU Scores (c) Prediction BLEU Scores

Figure 2: These plots correspond to metrics computed at the completion of each epoch and thus are functions of epochs. (a) Train and validation losses of the model at each epoch. Notably, loss was computed by division of number of captions processed rather than the typical number of batches processed as our batches were varying in length. (b) Individual BLEU scores for 1-gram to 4-grams on the output of the validation set given the reference caption at each epoch. This corresponds to the caption used to compute loss. (c) Individual BLEU scores for 1-gram to 4-grams on the output of the validation set without the reference caption at each epoch. These were computed by beam search with a beam size of 3.

Learning Rate	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
Epoch	0	8	11	14	17	20

Table 1: Summarization of the epochs at which the learning rate was decayed. The learning rate was decayed by a factor of 0.1 after 2 epochs if the validation loss did not reduce by more than 10^{-4}. With a patience of 2, the learning rate will decay after a minimum of 3 epochs. We see that the learning rate decays exactly every three epochs after the 8th epoch, which suggests that learning stopped after that epoch despite the learning rate still being relatively high. Considering figure 2c, we see that true BLEU scores also peaked after 8 epochs. This is consistent with Xu et al.'s decision to early stop their model training based on BLEU score.

Model	Type	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Our	Output	58.7	23.5	7.5	3.2
	True	75.2	47.4	26.4	16.4
	Cumulative	72.8	55.7	34.8	22.1
Xu et al.	?	67	44.8	29.9	19.5

Table 2: BLEU scores for 1- to 4-grams of the model upon training completion. We report individual BLEU scores for the output with reference, the true prediction without reference, and the cumulative BLEU scores for true predictions. Xu et al. did not specify which BLEU scores they reported, but it is likely that their scores are for true prediction. We are unsure whether it is individual or cumulative. All BLEU scores are reported without brevity penalty.

4 Discussion

The goal of this model is to learn to generate representative captions for given input images. To that realm, we believe we have achieved success and report results similar to those from Xu et al. However, the model is not without issues. Since this is a supervised learning task, the model is subject to distributions in the data's images and captions. Empirically, we noticed that every expert caption describes the scene with an actionable item performed by the subject. To this end, the model will try to caption every image with the subject-verb structure. Furthermore, without fine tuning, the model will struggle somewhat with out-of-domain images. See Appendix figure 5 for some examples of these.

We are unable to conclude whether our model outperforms the original authors' model as they did not specify which metric they evaluated on. In either case, our model outperforms theirs in BLEU-1 and BLEU-2 scores. This means that our model identifies individual words or concepts better than theirs. If they evaluated on individual BLEU-score, our model performed worse on BLEU-3 and BLEU-4. This would suggest that our model does not connect sequences of words together as fluently or accurately as their model. If they evaluated on cumulative BLEU-score, our model performed better all around.

While training smaller models, we noticed that more data means greater expressiveness of the model. However, this does not necessarily imply overfitting by our model – we did not notice any deprecation in BLEU scores on the validation dataset. Since the BLEU scores plateaued, we conclude that our model has reached its limit of expressiveness given the current data. More datapoints means a larger vocabulary as well as new caption styles to learn from. Furthermore, we also believe that our model's final weights could be used to warm-start for models training on the Flickr30k and MSCOCO datasets. Unfortunately, we ran out of time to pursue this avenue of research.

5 Conclusion

We reimplemented the LSTM with soft attention model for image captioning by Xu et al. and achieved similar success on the Flickr8k dataset. In the future, we would like to explore larger datasets as well as perform qualitative evaluation on the model. The latter would show us exactly what parts of the image the model focuses on when it generates its captions, which would increase our understanding of the black boxes of neural networks.

6 Acknowledgement

This project was completed for CS6120 under the instruction of Professor Uzair Ahmad.

I would like to thank Sagar Vinodababu [2] for providing an open source implementation of a similar model, which I referenced when I got stuck with my implementation.

References

[1] Flickr8k Dataset. Retrieved October 2021, from https://www.kaggle.com/adityajn105/flickr8k/

[2] Vinodababu, Sagar. (2020). A PyTorch Tutorial to Image Captioning. from https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning

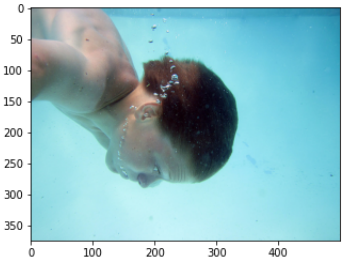
[3] Xu, K. (2015) Arctic Captions. from https://github.com/kelvinxu/arctic-captions

[4] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y. (2016). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of Machine Learning Research, 37, 2048-2057.

A Appendix

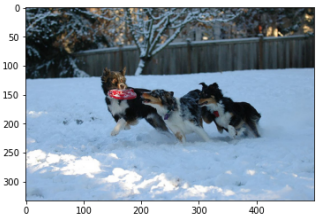
Output: a man is down the mountain of a mountain . --end--
True: a man is climbing up a mountain . --end--
Ref: the man repels down the side of a cliff . --end--

Output: a boy boy is jumping in a pool pool . --end--
True: a young boy splashes in the water . --end--
Ref: a young boy is underwater in a swimming pool . --end--



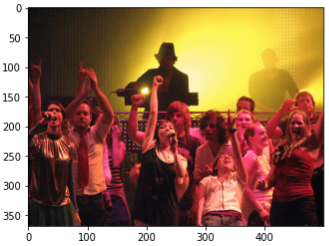
Output: a black is in a field . a stick in its mouth .
True: a black and white dog is running through a field . --end--
Ref: a dog running through a field with a ball in its mouth --end--

Output: a dogs are one with a toy leash in in in front . --end--
True: three dogs playing in the snow . --end--
Ref: three dogs , one holding a red frisbee , standing in snow . --end--



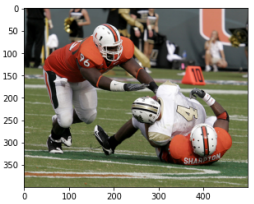
Output: two men are cameras poles . standing on the beach .
True: two men are standing on the beach with a fishing pole . --end--
Ref: two men with fishing rods are standing on a shore --end--

Output: a group of people sing in a of a men . --end--
True: a group of people gather together . --end--
Ref: a group of performers dancing in front of two djs . --end--



Output: a group of people children standing down the street . --end--
True: a group of people in red and white uniforms are walking down the street . --end--
Ref: a group of santas are walking in the city . --end--

Output: three football are a the field in the ball . the race game .
True: two football players are tackling a football player . --end--
Ref: two players take down the player with the ball in a football game --end--



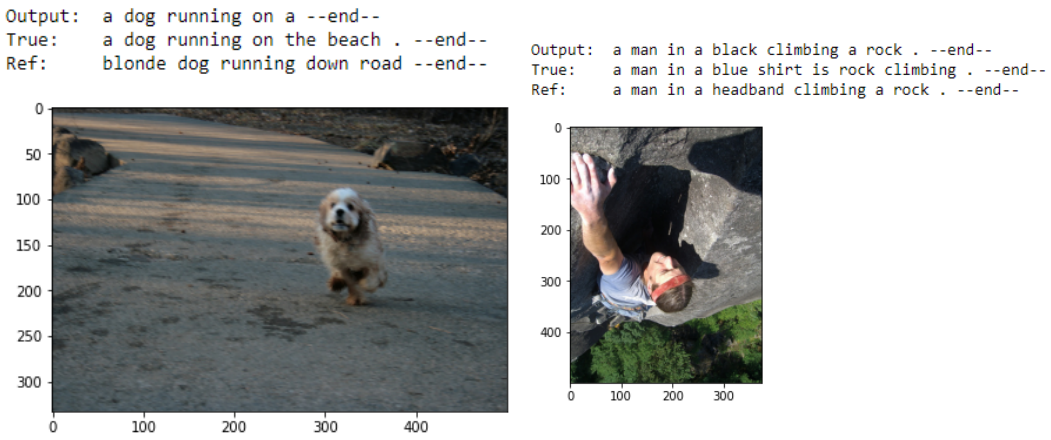


Figure 3: Some captions we considered to be a success.

Figure 4: Other captions we considered to be a failure or partial failure.

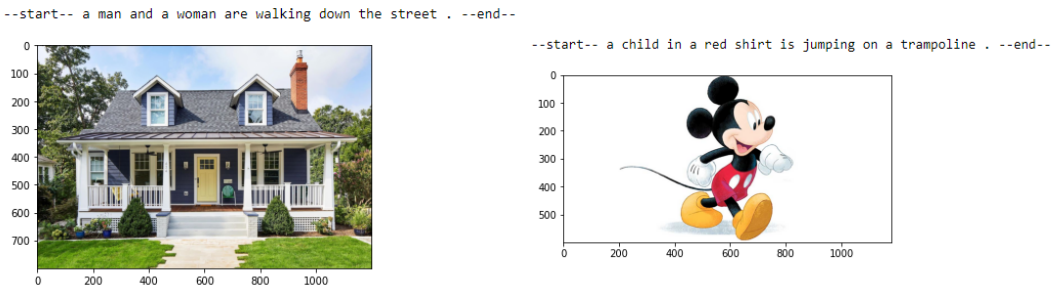


Figure 5: Out of domain samples.