Bridging Pixels and Words: Enhancing Image Captioning with Attention Mechanisms

Pratik Doshi
University of California, Santa Cruz
prdoshi@ucsc.edu

Ayush Ranjan
University of California, Santa Cruz
aranjan1@ucsc.edu

Abstract

In this project, we explore Vision Language Models. These models can be used for a variety of tasks, including generating image captions, question answering based on an image, image retrieval, and generating images given their description. This project focuses on the image captioning task. We use a simplistic LSTM model to observe baseline performance and then apply dynamic attention-based deep learning techniques to improve performance over this baseline. The codebase for this project is: https://github.com/Pratik-Doshi-99/image-captioning-transformers

1 Introduction

The image captioning task is fundamentally a token sequence generation problem. We want to train a deep neural network to extract relevant image features and learn the association between those features and the captions from the training dataset. We then use this model on the validation set to evaluate whether the model can use learned associations to generate meaningful captions for these unseen images.

This project explores some of the foundational work in image captioning. This work predates the use of modern transformers for vision tasks. With this in mind, we try to understand our model's performance on a relative scale by using a baseline that uses the pre-trained Resnet50 network to extract features from the image and then an LSTM network to generate the sequence from those features. This is an appropriate baseline because the primary model explored here uses an attention mechanism to enhance a plain vanilla LSTM network.

As discussed further in this report, we successfully trained the attention-enhanced LSTM decoder and achieved a significant and stable improvement over the baseline with as low as 10 epochs of training. For this project, we decided to generate our own train/validation split. Later, we successfully reproduced similar results on the widely used Karpathy splits. Furthermore, we evaluate our models on the BLEU metric, giving a weight of 0.6 and 0.4 to BLEU-1 and BLEU-2 respectively.

2 Related Work

The foundational research for this project is from [5]. This research introduces the attention mechanism that is used as the primary model in this project. The soft attention mechanism is adopted from this paper. Work done by [2] uses Gated Recurrent Units (GRU) for sequence generation. This project demonstrates a piece-wise breakdown of the entire model. Generating the image feature map/feature vector once for the entire dataset, and subsequently using this as an input to the decoder part of the network saves valuable training and inference time.

The authors of [5] published the code used to train their models. Their code [1] uses scikit-learn and predates the more efficient Pytorch. Subsequent work by [4] was done to shipping legacy code to the latest PyTorch version. However, this codebase does not support the Resnet50 encoder nor the

Preprint. Under review.

Flickr8k dataset used in this project. The code used to train our models was referenced from these sources but modified for network architecture (encoder architecture, embedding dimensions, and hidden dimensions), vocabulary, dataset, and train/validation split.

3 Experiments

Here is a visual description of the baseline model used.

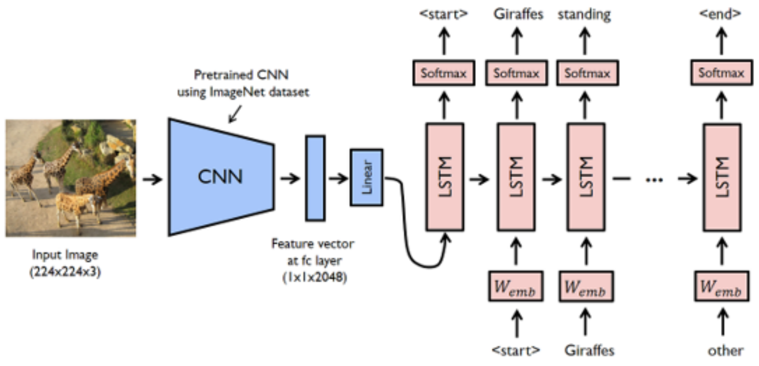

Figure 1: Baseline Architeture

Here is a visual description of the primary model used. The encoder and LSTM components are the same, the difference is in the attention layer.

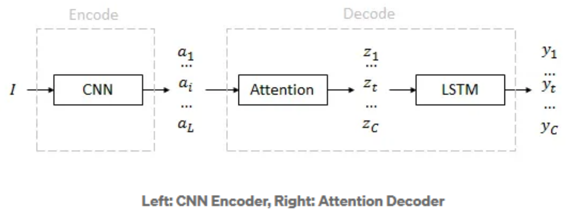
Encode
Decode
I
CNN
a_1 ... a_i ... a_L
Attention
z_1 ... z_t ... z_C
LSTM
y_1 ... y_t ... y_C
Left: CNN Encoder, Right: Attention Decoder

Figure 2: Primary Model Architecture: Ref- [3]

Mathematical details of the model are omitted here. They can be found under Soft Attention in [5]. The original paper used VGGnet as the encoder; this project uses Resnet50 for both the baseline and the primary model. In the baseline, there is a linear layer at the end that produces a feature vector in 256 dimensions. In the primary model, this layer is absent. The raw feature map from Resnet50 is 49x2048 and is directly used. The embedding and hidden dimensions in the baseline and primary model are maintained at 256 to ensure fair performance comparison between the two.

In the case of both models (baseline and primary), we are only training the decoder component i.e. keeping the encoder frozen. Due to this, the part of the network trained is quite small and should require fewer training epochs. The model implementation from [4] uses a step-based schedule to lower the learning rate as the training progresses. In our experiments, we tried both with and without such a schedule.

Another key aspect of training LSTMs is whether to use the previous predicted token or the previous ground truth token to predict the next token. We used the previous ground truth token to train both the baseline and primary models. This is called teacher forcing.

We observed an irregular validation loss while training the primary model. To verify the correctness of the training process, we ran the implementation from [4], with a small modification to integrate the Flickr8k dataset. We observed the same irregularity here also. The exact cause of the irregularity remains unclear at this point.

4 Discussion

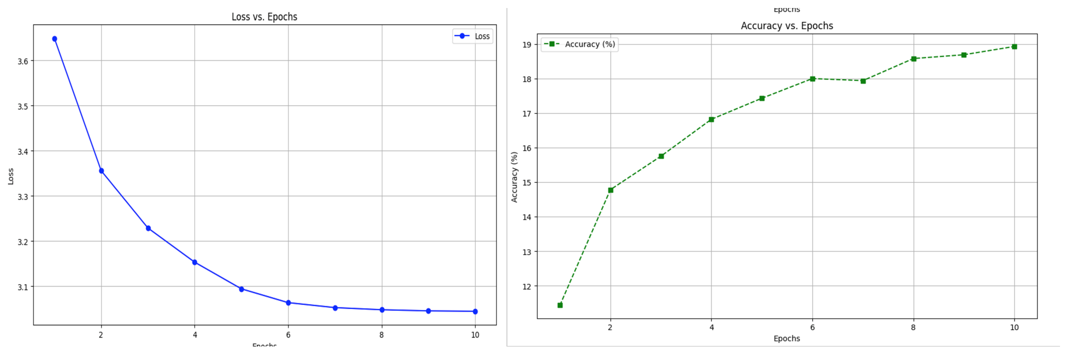

Figure 3: Baseline Loss and Accuracy

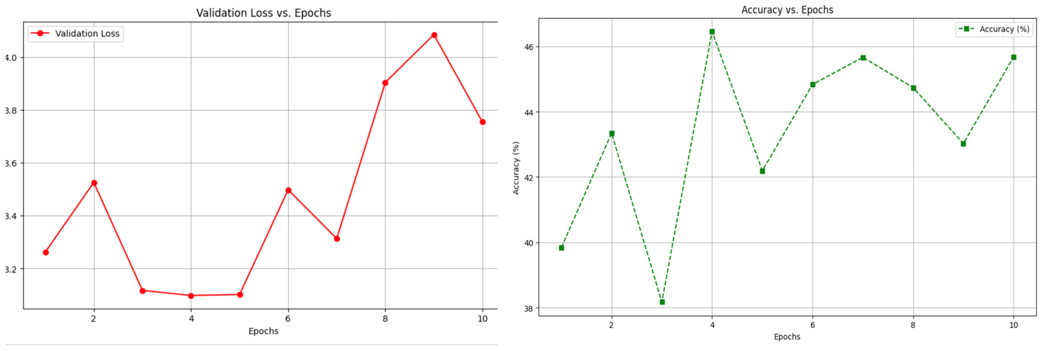

Figure 4: Primary Model Loss and Accuracy

4.1 Tackling the Erratic Validation Loss Problem

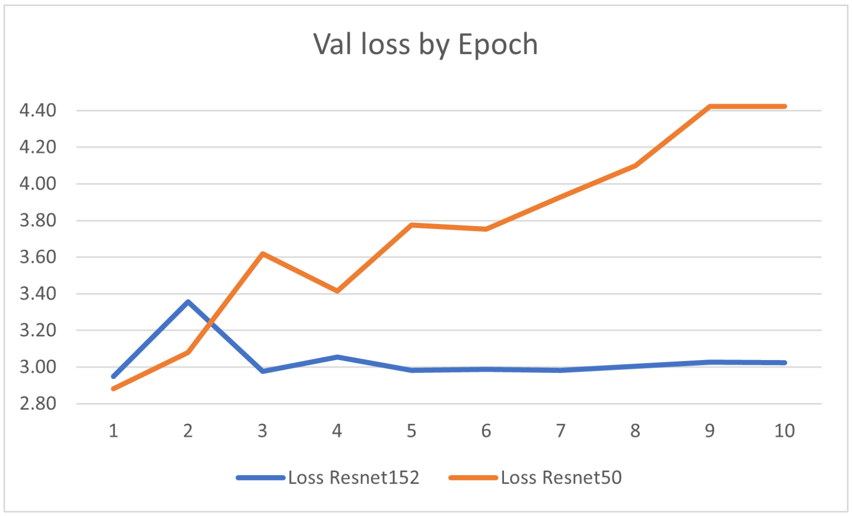

Figure 5: Resnet50 (this project) vs Resnet152 (Open Source Implementation)

The loss and accuracy (BLEU) of the primary model is erratic; it opens the scope for further investigation. After a thorough sanity check of the code, we conducted another experiment to assess the results of [4]. After small modifications to accommodate the Flickr8k dataset, we train that model and observe the results. We observed results similar to our primary model. As training progresses, there is no improvement in validation loss. It increases after the first epoch and then comes down to where it started. We tried to implement a learning rate schedule as a second remedy to the erratic validation loss problem. However, even after an aggressive decay of the learning rate, the problem persists.

5 Future Scope

1. Although the primary model saw an erratic validation loss, it promises significant performance improvement on the BLEU metric. It is still to be seen whether similar BLEU improvements and erratic validation loss patterns are observed in other larger datasets like MSCOCO or Flicker30k.
2. At present the model predicts the token with the maximum probability. An alternative approach is to use beam search to maximize overall probability over generating the entire sequence instead of selecting most likely token at each time step of the sequence.

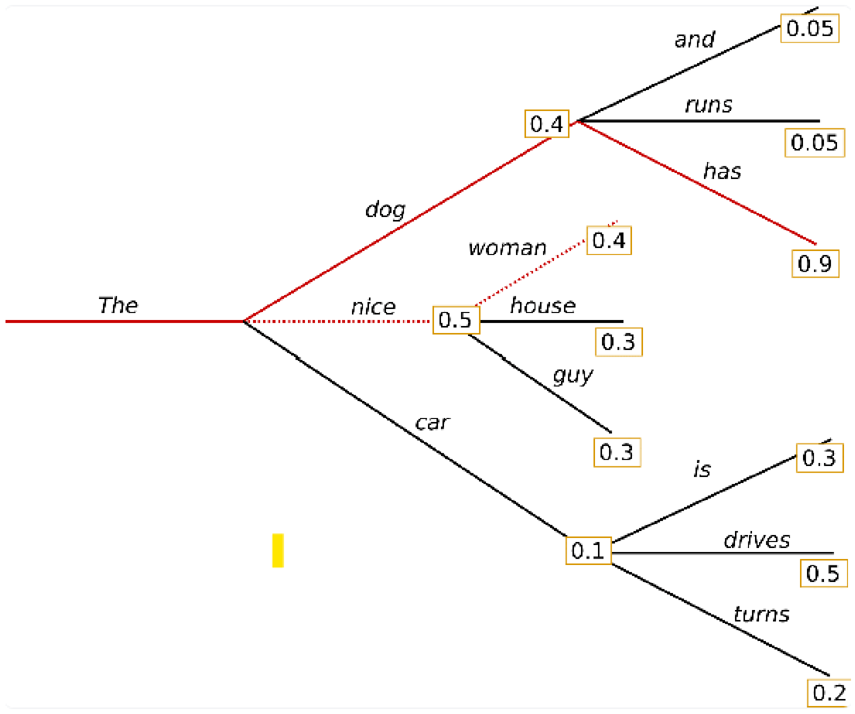

Figure 6: Visual Demo of Maximizing Sequence Probabilities

3. With the advent of the modern transformer, it would be interesting to see how better can a transformer-based image captioning model perform over this foundational attention mechanism. Such a model would typically view the image feature vector as a prefix and predicted tokens as the context window for the next token. Both decoder-only transformers and encoder-decoder transformers would be relevant to this task.

6 Conclusion

The erratic validation loss problem emerges as a stubborn obstacle to a convincing image captioning model based on this attention mechanism. However, this model does show a promising improvement on the BLEU metric. A deeper dive is needed to get to the root of the problem. Notwithstanding, the attention mechanism has motivated a lot of research in model architectures that allow the model to focus or pay attention to certain parts of the image. Retrospectively, this attention mechanism can be viewed as an important milestone for the AI community and an important step in the direction of the eventual Self-Attention construct.

References

[1] Show-attend-and-tell: Original implementation. Original implementation. https://github.com/kelvinxu/arctic-captions.

[2] H. Jain. image-captioning-gru. Uses a Resnet + GRU Combination for image captioning. Uses tensorflow for training and inference. https://github.com/HJ899/image-captioning-gru.

[3] S.-H. Tsang. Review — show, attend and tell: Neural image caption generation. Medium Article: https://sh-tsang.medium.com/review-show-attend-and-tell-neural-image-caption-generation-f56f9b75bf89.

[4] A. C. Wong. Show-attend-and-tell: Implementation in pytorch. Implements the original paper in Pytorch. Uses the MSCOCO dataset, Karpathy splits for train/validation and a slightly different BLEU metric. https://github.com/AaronCCWong/Show-Attend-and-Tell.

[5] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

A Appendix

Here are a few example images and the primary model's predictions.

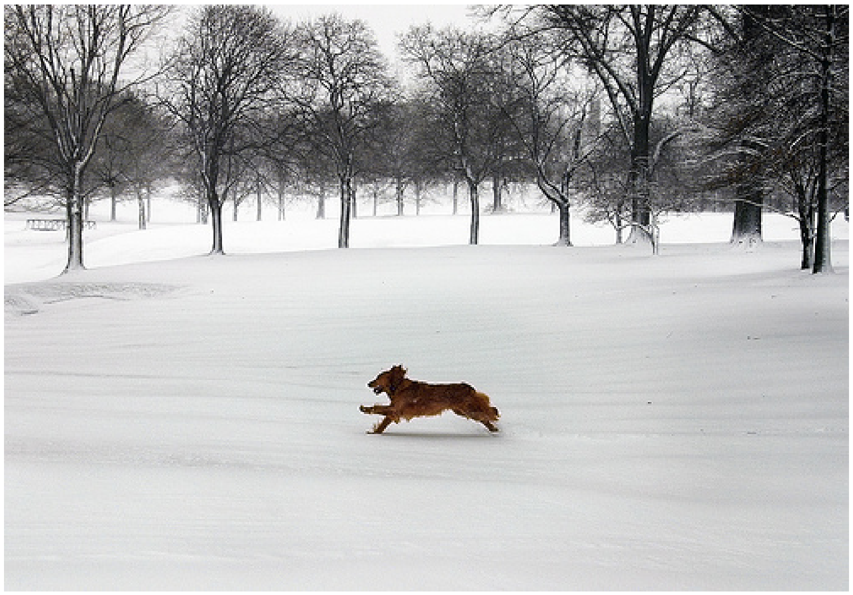

Figure 7: Actual Caption: a dog is running through the snow; Predicted Option: a dog is running in the grass

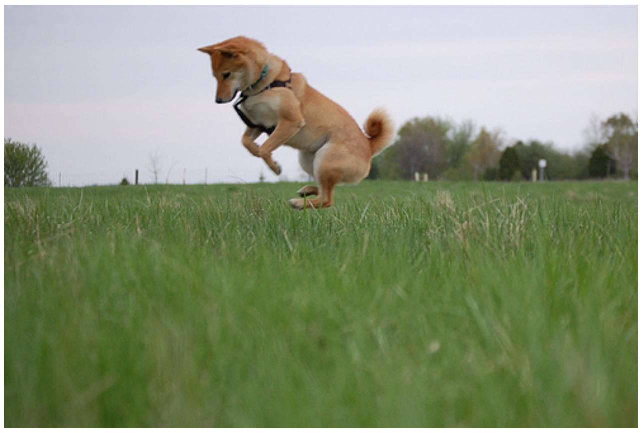

Figure 8: Actual Caption: a brown dog jumps high on a field of grass; Predicted Option: a brown dog is in in the grass

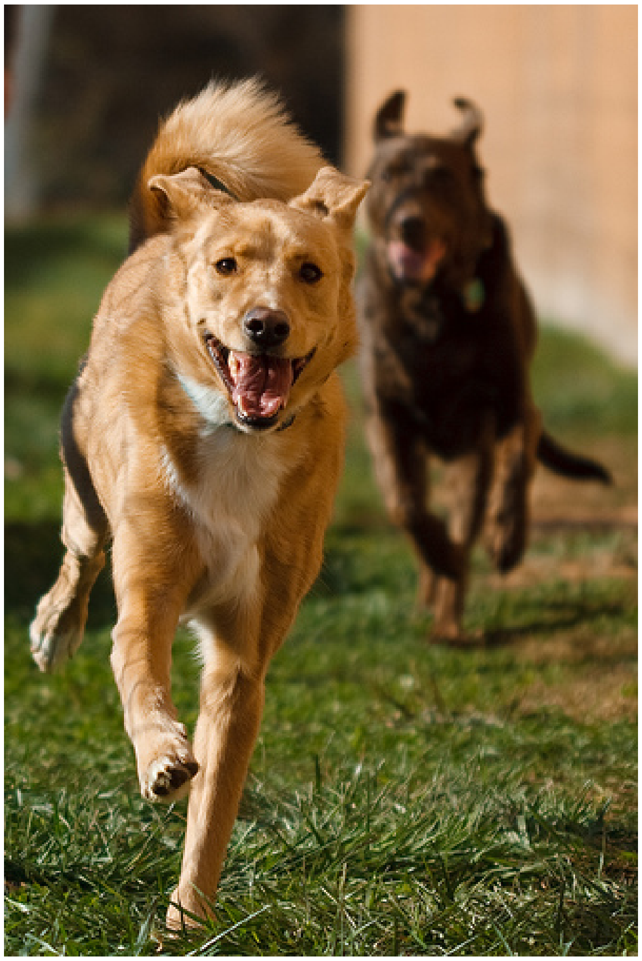

Figure 9: Actual Caption: two dogs run over the grass; Predicted Option: a dogs are in a grass