Image Captioning Benchmarking using Encoder/Decoder Deep Learning Models

Arsalan Anwar, Mohammed Zakriah Ibrahim, Tyler Kobil

{ax2134, mi2471, tak7229}@nyu.edu
Github Repository: https://github.com/tkobil/image-captioning-using-encoder-decoder-models

Abstract

In this paper, we analyze the effectiveness of encoder-decoder deep neural network models on the common task of image captioning. A pre-trained ResNeXt Convolutional Neural Network is used as an encoder, while we experiment with both Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) architectures for the decoder. BLEU, METEOR, and ROGUE scores are used to compare both one-layer and three-layer GRU and LSTM models. Some example images along with their generated and target captions are also included.

Introduction

Automatic image captioning, the task of generating natural language descriptions for images, lies at the intersection of computer vision and natural language processing. This project aims to benchmark the effectiveness of encoder-decoder deep learning models for image captioning. Current approaches to image captioning often rely on hand-crafted features, limiting their effectiveness.

This project proposes a generative model based on deep learning, specifically an encoder-decoder architecture, to bridge this gap. We will first train a model from scratch on the Flickr 8k dataset and then compare the model generated captions to the true captions to evaluate the performance. We will also compare the length ratio of the generated captions vs. actual captions in order to evaluate how descriptive our image captioning model is.

Literature Survey

Automatic image captioning, a subfield of Artificial Intelligence (AI), bridges the gap between computer vision and natural language processing (NLP). It aims to generate natural language descriptions for images, enabling applications like image retrieval, accessibility tools for visually impaired users, and content creation automation. This literature survey explores the current state-of-the-art in image captioning, focusing on encoder-decoder deep learning models.

Encoder-Decoder Architectures

Encoder-decoder architectures are well-suited for image captioning due to their ability to handle sequential data. The encoder, typically a Convolutional Neural Network (CNN) like VGG [1] or ResNet [2], extracts high-level features from an image. These features capture visual information such as shapes, objects, and their relationships. The decoder, often a Long Short-Term Memory (LSTM) network [3], processes the encoded image features sequentially and generates a corresponding caption word by word.

Recent advancements incorporate attention mechanisms within the encoder-decoder architecture. Attention allows the model to focus on specific image regions relevant to the caption being generated, leading to more accurate and descriptive captions. Pioneering work by Xu et al [4] introduced a visual attention mechanism that allows the decoder to selectively attend to specific image regions based on the previously generated caption words.

Dataset Description

For training our image captioning model, we utilized the Flickr8k dataset, a popular benchmark dataset widely used in image captioning research [5]. This dataset consists of 8,092 unique images, each paired with up to 5 captions written by different people. This variety of human-written descriptions provides a rich training signal for the model and offers a wider range of vocabulary and sentence structures for the model to learn from.

While the Flickr8k dataset offers numerous advantages, it's important to acknowledge its limitations. Compared to very large-scale image captioning datasets, Flickr8k is relatively smaller. This might limit the generalizability of models trained solely on this dataset to unseen image data. Additionally, there might still be limitations in the diversity of vocabulary and sentence structures used in the captions themselves.

Despite these limitations, the Flickr8k dataset remains a valuable resource for training and evaluating image captioning models, particularly for projects that prioritize the use of human-written captions and require benchmarking against established results. By effectively leveraging the strengths of this dataset, we aim to develop a robust image captioning model capable of generating high-quality and informative descriptions for unseen images.

Methodology

In this section, we outline our approach to designing and training the model, discussing various architectural choices,

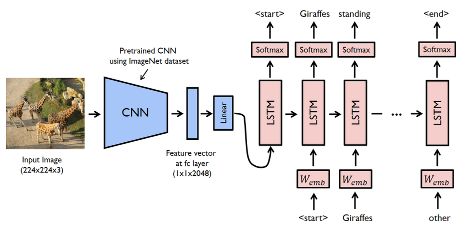


Figure 1: Base Architecture for Image Captioning using Encoder-Decoder Models

their advantages, and potential drawbacks, as well as key lessons learned during the design process.

Model Architecture

Our image captioning model employs a standard encoder-decoder architecture as shown in Figure 1. with the following components:

- **Encoder (EncoderCNN):**
  Pre-trained ResNeXt101_32x8d CNN: This pre-trained model extracts high-level features from the input image. As specified in the code, the final fully-connected layer (fc) is replaced with a new linear layer to project the features to a desired embedding size ($embed\_size$). Freezing the weights of all layers except the final linear layer leverages pre-trained features and reduces training time.
- **Decoder (DecoderRNN):**
  Embedding Layer: This layer maps each word in the vocabulary (represented by an index) to a corresponding embedding vector of size $embed\_size$. These embedding vectors capture semantic relationships between words.
  Recurrent Neural Network (RNN): The architecture of the RNN can vary. We experimented with both **Long Short-Term Memory (LSTM) networks** and **Gated Recurrent Units (GRUs)**.
  LSTMs have a more complex gating mechanism compared to GRUs, potentially allowing them to capture longer-term dependencies within the caption sequence. GRUs, on the other hand, are generally simpler and faster to train. The specific RNN type (LSTM or GRU) used is dependent on model selection and hyperparameter tuning as shown in Tables 1 and 2. The RNN takes the encoded image features and the embedded word from the previous time step as input to predict the next word in the caption sequence. The dropout layer (with probability 0.5) helps prevent overfitting.

Loss Function

After experimenting with various loss functions, we utilized the cross-entropy loss function, a common choice for classification tasks like image captioning. It measures the difference between the model's predicted probability distribution for each word in the vocabulary and the actual target word distribution in the ground-truth captions. By minimizing the cross-entropy loss, the model learns to assign higher probabilities to words that are more likely to appear in the caption based on the encoded image features and the previously generated words.

Optimizer

We used the Adam optimizer to update the model weights during training. Adam is an efficient algorithm that adapts the learning rate for each parameter individually, often leading to faster convergence compared to traditional optimizers.

Hyperparameter Selection

Hyperparameters significantly impact model performance and require careful tuning. Here are some key hyperparameters we optimized based on the code:

1. **Embedding size (**$embed\_size$**):** Defined in the EncoderCNN and DecoderRNN code, this hyperparameter determines the dimensionality of the word embedding vectors. We experimented with different embedding sizes to find a balance between model complexity and caption quality. We found that an embed size of 256 gave us the best results.
2. **Hidden Size of the RNN (**$hidden\_size$**):** Defined in the DecoderRNN code, this parameter determines the internal memory capacity of the LSTM/GRU network, impacting its ability to capture long-term dependencies within the caption sequence. We explored different hidden sizes to find a suitable value for capturing relevant information and found that a hidden size of 256 gave us the best results.
3. **Number of RNN Layers (**num_layers**):** Defined in the DecoderRNN code, this parameter controls the depth of the LSTM/GRU network. We experimented with different numbers of layers as shown in Tables 1 and 2 to determine the optimal architecture for our task.
4. **Dropout Rate (**$dropout$**):** Defined in both EncoderCNN and DecoderRNN code, dropout randomly drops a certain percentage of neurons during training to prevent overfitting. We tuned the dropout rate to 50% to optimize the balance between model generalization and training speed.

Training Details

- **Data Preparation and Augmentation**
  The code utilizes the $TrainDataset$ and $TestDataset$ classes to load and pre-process image and caption data from the Flickr8k dataset. Data augmentation is applied using a series of transforms from $torchvision.transforms$:
  1. Image Resizing: We resize the image to a dimension of size 256x256 pixels using $transforms.Resize(256)$
  2. Image Cropping: We take a central crop of size 224x224 pixels from the resized image using $transforms.CenterCrop(224)$.
  3. Image Tensorization: We convert the image data from a PIL image to a PyTorch tensor using $transforms.ToTensor()$.

Table 1: Test Losses and Length Ratios of Different Encoder/Decoder Architecture Combinations

Encoder	Decoder	No. of Layers	No. of trainable Parameters	Loss	Length Ratio
ResNeXt101	LSTM	1	5680192	1606.23269128799	1.148
ResNeXt101	LSTM	3	6732864	1624.51798915863	1.076
ResNeXt101	**GRU**	1	5548608	**1495.68779**	**1.165**
ResNeXt101	GRU	3	6338112	1572.57738709449	1.307

Table 2: Performance Evaluation of Different Encoder/Decoder Architecture Combinations (Fixed ResNext101 Encoder)

Decoder	No. of Layers	BLEU score	METEOR score	Rogue 1 score	Rogue 2 score	Rogue L score
LSTM	1	0.0333	0.3584	0.2486	0.05695	0.22076
LSTM	3	0.0343	0.4096	0.2529	0.05893	0.05892
GRU	1	0.0345	**0.4105**	**0.2535**	**0.06154**	**0.22812**
GRU	3	0.0292	0.3356	0.2370	0.05672	0.21689

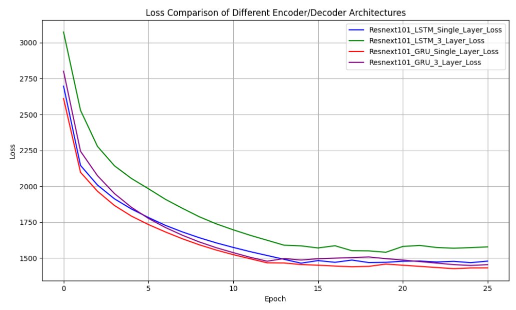


Figure 2: Loss Comparison Graph of Different Encoder/Decoder Architectures

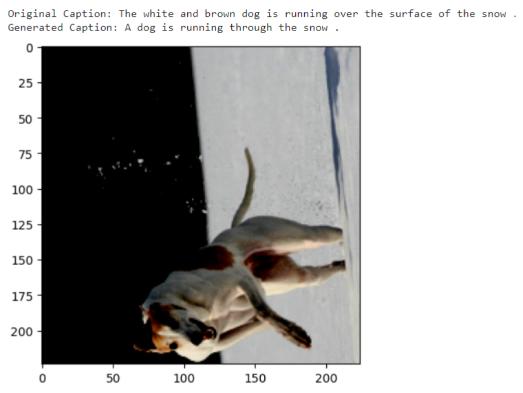
4. Image Normalizaton: We normalize the pixel values of the image tensor using the specified mean and standard deviation values using $transforms.Normalize$(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]).

- **Data Loaders:**
  Data Loader objects are created for both training and testing datasets ($train\_loader$ and $test\_loader$).
  - $batch\_size$: It is set to 50, which controls the number of images processed in each training iteration.
  - $shuffle$: This is set to True to randomize the order of the data samples within the dataset during training. This helps prevent the model from over-fitting to the training data order.
  - pin_memory: It is set to True to pin the data loader to memory (CUDA or MPS) for faster data transfer during training.
  - $collate\_fn$: This custom function is used during data loading. This function pads the captions within each batch to ensure they have the same length, and handles the padding index (pad_idx) defined in the vocabulary.

Results

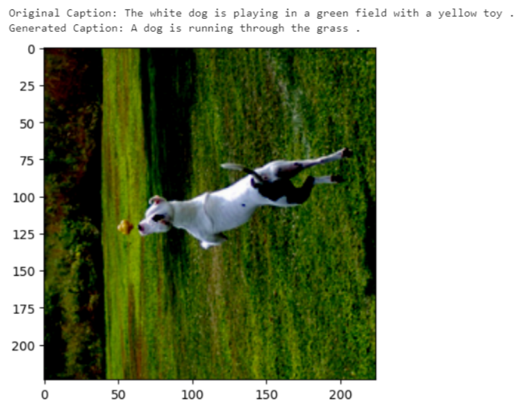
In this section, we present the performance comparison of the different encoder-decoder models trained on the flickr8k dataset and identify the best-performing model based on some key performance metrics such as BLEU, ROUGE, and METEOR.We have trained all our models for 25 epochs. This evaluation is done by comparing the generated image captions with human-written reference captions from the dataset. The results are discussed below:

1. **BLEU (Bi-Lingual Evaluation Understudy) Score:**
   BLEU [6] measures the similarity between a candidate caption and multiple reference captions by considering n-gram precision (n representing the length of the sequence) between them. As shown in the table, even though the model performance is good as seen in Figures 3 and 4, the BLEU scores are less because it calculates the percentage of n-gram in candidate caption that also appear in any of the reference captions but does not take into account the lexical meaning of both the generated and actual captions.
2. **ROUGE (Recall-Oriented Understudy for Gisting Evaluation) Score:**
   ROUGE [7] is another popular metric for image captioning evaluation. Similar to BLEU, ROUGE measures overlap between the candidate caption and reference captions. It also considers different overlap measures like Rouge 1 for unigrams, Rouge 2 for bigrams, and Rouge L for the longest common subsequence (LCS). All of these show good results for Resnext101-GRU single layer model.
3. **METEOR (Metric for Evaluation of Translation with Explicit Ordering) Score:**
   METEOR [8] is a specialized metric primarily intended for assessing machine translation quality, yet its applicability extends to image captioning evaluations as well. It incorporates unigram precision and recall metrics, along with synonym matching, to gauge the semantic similarity between the generated caption and reference captions. Our top-performing model achieved a METEOR score of 0.4105, surpassing the performance of other models in this context.
4. **Length Ratio** The Length Ratio measures the ratio of the length of the target caption to the length of the generated caption from our model. This is a helpful metric for



```
Original Caption: The white and brown dog is running over the surface of the snow .
Generated Caption: A dog is running through the snow .
```

Figure 3: Testing Resnext101-GRU model on random test image showing a dog running on snow



```
Original Caption: The white dog is playing in a green field with a yellow toy .
Generated Caption: A dog is running through the grass .
```

Figure 4: Testing Resnext101-GRU model on random test image showing a dog running on grass but original caption having excessive detail

evaluating how descriptive our model is. As seen in Table 1, the 3 layer GRU-based model had the second largest length ratio, indicating the model's ability to provide detailed captions. Despite not having the largest length ratio, this model outperformed across the other metrics, so it is still deemed our highest performing model.

Conclusion

In this paper, we explore 1 and 3 layer LSTM and GRU based encoder-decoder models for the image captioning tasks. Based on the metrics outlined in the Results section, it is clear that our single layer GRU-based encoder-decoder model performs the best. While the BLEU, METEOR, and Rogue scores are certaintly helpful for evaluating the performance of a model for this type of natural-language based task, they do fail to capture the ability to detect a models ability to reason, and have a bias towards word-for-word prediction. We've taken great care to not overfit our model by utilizing an early stopping technique, and believe that our results in Figure 3 and 4 show that our model showcases the ability to understand what is really being shown in an image. In the future, we would consider exploring other evaluation metrics that might better help measure semantic reasoning, as well as consider fine-tuning the pre-trained vision model on the dataset's images to ensure all features exist in the training data.

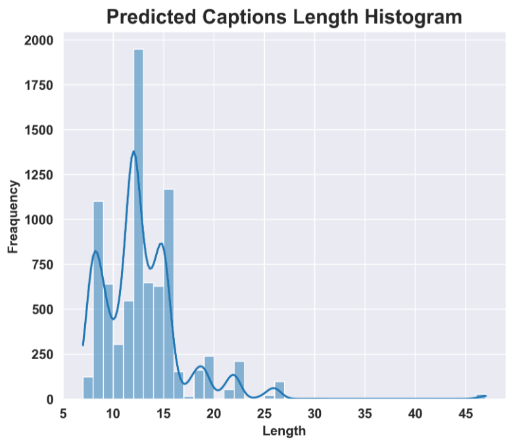


Figure 5: This graph shows the Length of the predicted captions of the single layer GRU-based encoder-decodermodel mode

References

1 Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

2 He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

3 Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

4 Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. and Bengio, Y., 2015, June. Show, attend and tell: Neural image caption generation with visual attention. In International conference on machine learning (pp. 2048-2057). PMLR.

5 Hodosh, M., Young, P. and Hockenmaier, J., 2013. Framing image description as a ranking task: Data, models and evaluation metrics. Journal of Artificial Intelligence Research, 47, pp.853-899.

6 Papineni, K., Roukos, S., Ward, T. and Zhu, W.J., 2002, July. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics (pp. 311-318).

7 Lin, C.Y., 2003. ROUGE: Recall-oriented understudy for gisting evaluation.

8 Alon Lavie and Abhaya Agarwal. 2007. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In Proceedings of the Second Workshop on Statistical Machine Translation, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.