Deep Learning Model for Image Caption Generation

Final Project Report

CIS 511: Introduction to Natural Language Processing

Ankita Negi, Elie Rizk, Harsha Machineni

Abstract

In this project, we have tackled the problem of generating descriptive captions from static images. We have built an encoder-decoder neural network model to encode image features and output valid description. We have compared the Inject and Merge architectures with different dropout rates and activation functions. We have found that the Inject model with low dropout rate is more prone to overtraining than the Merge model. We also didn't find the SELU activation function to offer any performance advantage than the RELU one, which suggests that the vanishing gradient problem might not be a real issue in the encoder-decoder RNN architecture. Finally, we agree with the current literature that the merging architecture for image captioning offers slight performance and memory benefits compared to the inject architecture.

Introduction

While the problem of image captioning seemed infeasible for early NLP researchers, newly emerging machine learning models have been able to confidently generate captions from images, expanding the frontier of Artificial Intelligence. In practice, this area of research can help make images more accessible, e.g., auto captioning images for the blind or visually impaired. However, the problem of image captioning goes beyond the simple need for describing static images. In fact, this technology can be thought of as an application to the AI challenge of "grounding symbolic or linguistic information in perceptual data" (Tanti et al., 2017). In other words, how can we represent linguistics within the context of visual information?

Although there are a vast number of approaches to image captioning, the three main ones use techniques from computer vision, information retrieval, or neural models (Tanti et al., 2017). In this project, we have limited ourselves to the latter approach, focusing on building a machine learning model based on the encoder-decoder recurrent neural network architecture to output captions from images. Therefore, we have explored through multiple model architectures, namely the Inject and Merge architectures, and experimented with model hyperparameters, including the dropout rate and activation functions. We have used the *Flickr8k* dataset for our experiments to identify the optimal configuration of our model and used this configuration when training on the larger *Flickr30k* data.

Related Work

As outlined in Chapter 10 of our textbook “Speech and Language Processing,” the encoder-decoder architecture is used for sequence modeling that involves a complex function of the input, i.e., not a simple mapping between input and output (Jurafsky et al., 2021). While the chapter covers its use in machine translation, the same logic applies to caption generation. In fact, a machine learning model that generates captions from images should process all the image input (in the form of color pixels) and the previously generated text to predict the next word to generate. The encoder has to transform the input image into an internal representation, e.g., a fixed length array of features, and the decoder has to use this internal representation to generate the final output, i.e., a series of words that describe the image. For our project the encoder will correspond to the pretrained VGG16 model and the decoder will be a Recurrent Neural Network trained on the *Flickr8k* or *Flickr30k* dataset. We have included below an abstract representation of the encoder-decoder architecture taken from the book.

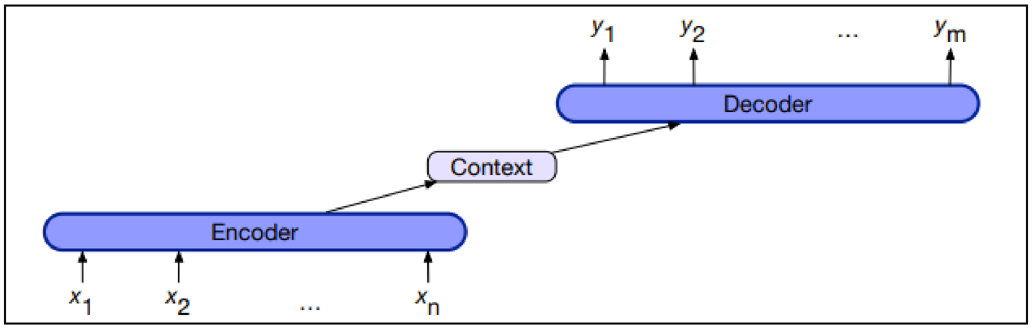
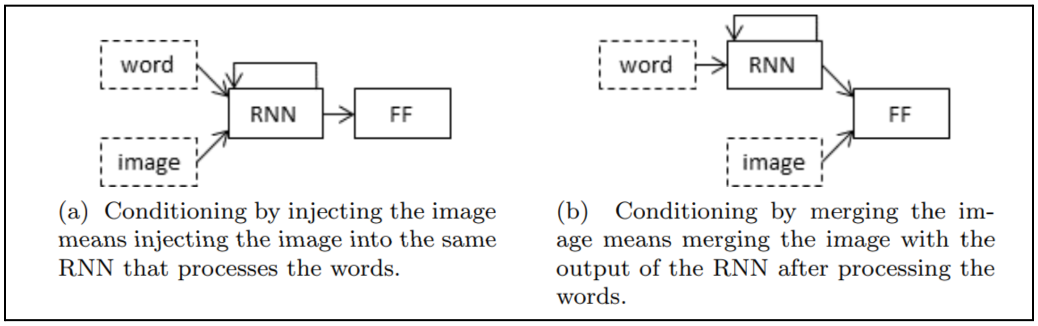


Figure 1: Abstract representation of the encoder-decoder architecture (Jurafsky et al., 2021)

The current literature divides image captioning encoder-decoder architectures into two main types: the Inject model and the Merge model. The Inject architecture corresponds to “injecting” the image into the same Recurrent Neural Network (RNN) that processes the words before feeding the output to a feedforward neural network. Alternatively, the Merge architecture “merges” the output of the RNN with the image features directly into an additional feedforward neural network, i.e., the RNN only processes the words as opposed to the words and the image (Tanti et al., 2018).



(a) Conditioning by injecting the image means injecting the image into the same RNN that processes the words.

(b) Conditioning by merging the image means merging the image with the output of the RNN after processing the words.

Figure 2: The Inject (a) and Merge (b) models for image captioning (Tanti et al., 2018)

In 2017, Tanti et al. have shown that “late merging outperforms injection,” concluding that the role of RNN in image captioning seems to be as encoders rather than generators. But one year later, the same group of researchers revisited their research and were able to show empirically that both models offer relatively equal performance, but the Merge model is more efficient as its RNN requires up to four times less memory in its hidden state (Tanti et al. 2018).

Dataset Description

Our project uses the *Flickr8k* dataset to explore multiple configurations of the machine learning model and the *Flickr30k* dataset to train one final optimal model. The Flickr datasets are widely considered the benchmark collection for image captioning due to their relative small size. The *Flickr8k* consists of a total of 8,092 images while the *Flickr30k* has around 31,000 images. Each image in the dataset is accompanied by five human generated captions which consist of the "gold-standard" output used to train and evaluate the model. The *Flickr8k* dataset we used was already divided into training and testing, but we manually divided the *Flickr30k* dataset on an 80-20 split for training and testing respectively using sklearn's *train_test_split* function with a random state of 25.

Methodology

Our machine learning process consists of three main stages: data processing, model training, and model evaluation.

Firstly, we need to extract the features from the images and the vocabulary from the captions. To do so, we use the pretrained VGG16 model provided by keras which is considered one of the benchmark Convolutional Neural Network models for object recognition from images. But considering that the final output of VGG16 consists of a probability distribution of the objects detected, we have decided to remove the last layer of the model. In effect, this will transform an image into an internal representation that will be useful for our model as it represents the important features of the image as a vector of fixed-length (output of the second to last layer of VGG16). We also process the captions of the dataset by removing all punctuations and lowering all capital letters. Once done, we end up with a pandas dataframe containing the image name, list of five human-generated captions, and image features.

Second, we need to train a recurrent neural network to generate correct words from the image features and sequence of previous words. To do so, we add start and end markers to the caption, create a Tokenizer on the extracted vocabulary, define the architecture of the model, and customize a data generator that yields one batch of input and output for each step of the training. We use teacher forcing in our training phase to speed up parameter tuning of our model (as suggested by Jurafsky et al. 2021). We compiled the model with categorical cross entropy as the loss function and the NAdam optimizer, and we trained each model for five epochs on the whole training data of *Flickr8k*. In addition, we have defined several model architectures with varying sets of hyperparameters to compare different configurations.

Finally, we evaluate the resulting model by generating the BLEU scores of the reference captions with the generated captions. We calculate the BLEU-1, BLEU-2, BLEU-3, and BLEU-4 scores which are based on the common unigrams, bigrams, trigrams, and 4-grams respectively between the gold-standard captions and the generated one.

Experimental Discussion

In total, we trained six machine learning models with the following architecture and hyperparameters.

Model Name	**Architecture**	**Dropout Rate**	**Activation Function**
Merge Model 1	Merge	0.5	RELU
Merge Model 2	Merge	0.2	RELU
Merge Model 3	Merge	0.0	RELU
Merge Model 4	Merge	0.5	SELU
Inject Model 1	Inject	0.5	RELU
Inject Model 2	Inject	0.2	RELU

For the Merge and Inject architectures, the models are made up of several layers stacked as shown in Figure (3) and (4) respectively. The *input_1* layer corresponds to the image features while the *input_2* layer is reserved for the previous words of the caption. As shown, the previously generated words of the caption pass through an embedding layer before going through the LSTM. We also have two dropout layers to avoid overtraining the model (that way the model can't rely on a few specific input weights). The *dense_1* layers have the SELU activation function for Merge Model 4 and RELU for the other models, while the *dense_2* layer has the softmax function to output a probability distribution. Also, we join two layers by simply adding both layers.

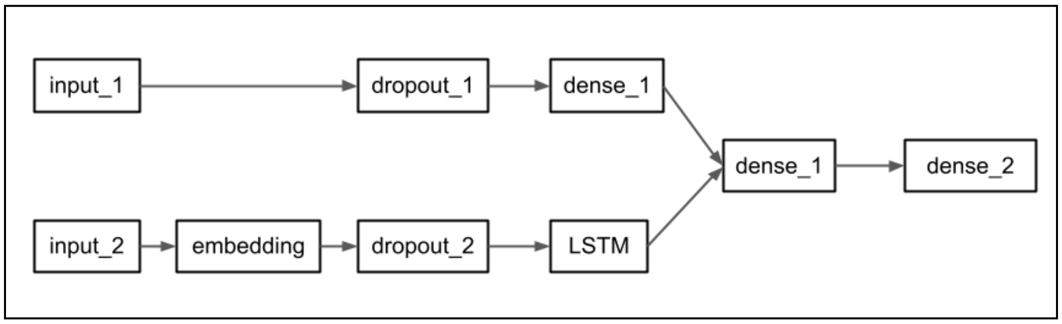


Figure 3: Merge architecture of our model

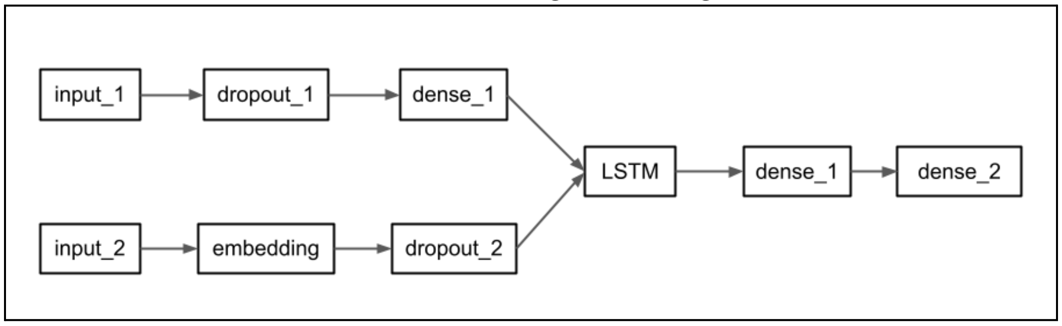


Figure 4: Inject architecture of our model

We obtain the following results shown in Figure (5) after evaluating the previous models on five epochs and averaging the results.

Model Name	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Inject Model 1	0.515955	0.279709	0.189231	0.084811
Inject Model 2	0.338539	0.188347	0.131229	0.055442
Merge Model 1	0.558628	0.299682	0.201715	0.093111
Merge Model 2	0.516851	0.277632	0.189583	0.086936
Merge Model 3	0.457284	0.244938	0.17	0.077445
Merge Model 4	0.491308	0.250439	0.170217	0.07715

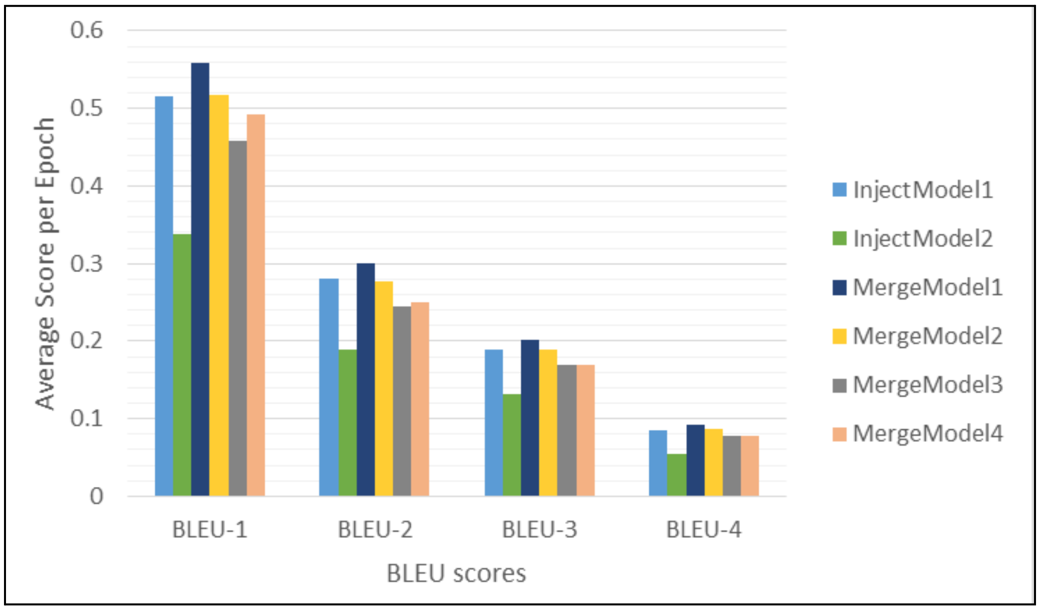


Figure 5: BLEU scores per model

After an 80-20 split on the Flickr30k dataset, we train the Merge Model 1 on 25,426 images and evaluate it on 6,357 images. We obtain the following BLEU scores: BLEU-1: 0.48, BLEU-2: 0.24, BLEU-3: 0.1, BLEU-4: 0.07.

Conclusion

From the results we obtained above, we were able to show how the Merge model with 0.5 dropout rate and the RELU activation function slightly outperformed the Inject model with the same hyperparameters. Also, we can conclude that the Merge model can sustain dropout rate variation (of 0.5, 0.2, or 0–no dropout layers). However, the Inject model is very susceptible to variations in the dropout rate: a decrease in the dropout rate from 0.5 to 0.2 leads to a 35% drop in the BLEU-1 score of the model. This indicates that the Inject model is more prone to overtraining than the Merge model, which is why a dropout rate of 0.5 is necessary for good performance in the Inject architecture. This makes sense since the LSTM layer of the Inject model gets fed both the image and text input (as opposed to only the text input in the Merge architecture), which in turn makes it more prone to overtraining on a specific subset of features from the image. Hence, it explains the need for a higher dropout rate. Lastly, we can see that the SELU activation function slightly hindered the performance of Merge Model 4. Although research has shown that using SELU solves the vanishing gradient problem, it doesn't seem to be particularly beneficial for image captioning.

Looking forward, we believe this project could be greatly improved by training the model on the Microsoft COCO dataset which includes around 328,000 images along with their captions. This might allow a better comparison of model architectures and hyperparameters. We also suggest using different kernel initializers, layering multiple recurrent networks, and exploring potentially different techniques or model architectures in the scope of generating captions from images.

References

Brownlee, Jason. "How to Develop a Deep Learning Photo Caption Generator from Scratch." *MachineLearningMastery.com*, 22 Dec. 2020, https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/.

Jurafsky, Dan, and James H. Martin. *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Prentice Hall, 2021, https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.

Tanti, Marc, et al. *What Is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?* arXiv, 2017, https://doi.org10.48550/ARXIV.1708.02043.

Tanti, Marc, et al. 'Where to Put the Image in an Image Caption Generator'. *Natural Language Engineering*, vol. 24, no. 3, Cambridge University Press (CUP), Apr. 2018, pp. 467–489, https://doi.org10.1017/s1351324918000098.