Image Captioning with RNNs

vikrant@vt.edu

Abstract

Image caption generation is a supervised learning problem generating captions based on an input image. In this project, I build an image captioning system that brings together computer vision with the natural language processing approach. The proposed approach consists of extracting visual features from images by using a pre-trained Convolutional Neural Network (CNN). Then the image features will be used to generate corresponding textual descriptions utilizing Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models. Preprocessing steps are executed for both image and text data, such as tokenization, padding, and feature vector generation. The resultant system is trained to accurately predict informative captions for the input images, showcasing that integrating different deep learning methods for multi-modal tasks demonstrates effective results.

1. Introduction

Generating a caption for an image is a task located at the intersection of computer vision and natural language processing. In image captioning one generates a meaningful and grammatical sentence to describe the content of a given image. This entails recognizing visual features, as well as an understanding of color and language semantics.

A common approach to this task involves combining two models:

- A Convolutional Neural Network (CNN) to extract high-level semantic features from the image.
- A Recurrent Neural Network (RNN) for generating the caption word-by-word based on the extracted image features.

For the RNN to learn how to generate captions, it needs to be trained on annotated datasets where each image is paired with one or more descriptive sentences. During training, captions are encoded into a format that the model can understand. While one-hot encoding is a possible method, it tends to be sparse and lacks semantic richness. Instead, we use GloVe (Global Vectors for Word Representation) embeddings, which provide dense vector representations that capture semantic relationships between words more effectively.

Once training is complete, the model can generate captions by feeding it the image features (output from the CNN). The RNN then predicts the next word in the sentence iteratively until a complete caption is formed. Since images can have multiple valid descriptions, the model learns to generalize based on the diversity of captions in the training set. An example for possible captions for an image is shown in Figure 1.

To evaluate the quality of the generated captions, we use the evaluation matrix like BLEU score (Bilingual Evaluation Understudy). BLEU measures the overlap between the predicted caption and one or more ground truth references using n-gram precision.

Figure 1. Possible captions a.) A girl is stretched out in shallow water b.) A girl wearing a red and multi-colored bikini is laying on her back in shallow water c.) A little girl in a red swimsuit is laying on her back in shallow water d.) A young girl is lying in the sand, while ocean water is surrounding her e.) Girl wearing a bikini lying on her back in a shallow pool of clear blue water

Image captioning has wide-ranging applications, including:

- Enhancing **accessibility** for visually impaired individuals by describing images out loud.
- Improving **image search and indexing.**
- Assisting in **content creation** and summarization.
- Enabling **autonomous systems** to understand their environment in natural language.

1.1. Problem Statement

The problem statement can be summarized as building a model to generate a meaningful, grammatically correct caption with a Recurrent Neural Network (RNN)-based architecture. The model should be able to extract visual information from an image, and then guess words sequentially to generate a coherent description.

2. Approach

To solve this problem of generating meaningful caption for an Image I used an architecture, which integrated both Computer Vision using Convolution Neural Networks (CNNs) for understanding the image and Natural Language Processing using a Recurrent Neural Network (RNN) for generating caption as displayed in Figure 2.

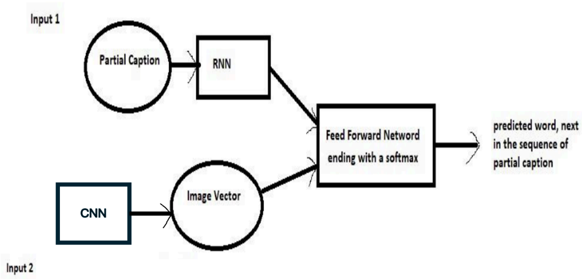

Figure 2. Caption Generation using CNN and RNN

In the sections that follow, I describe the dataset used and the preprocessing steps applied to prepare the data for training and evaluation. I also provide a detailed explanation of the models explored and the different architectural configurations used to assess their performance.

2.1. Dataset

To train and evaluate the image captioning model, we use the Flickr8k dataset. The Flickr8k dataset consists of 8,000 images and each image have 5 unique human-annotated captions that describe the content of the image. However, for my project I used only 7,000 images that were further divided into 6,000 images for model training and remaining 1,000 images for evaluating my model.

2.2. Preprocessing

To prepare the dataset for training, we apply the following preprocessing steps:

- As the first preprocessing step, all the annotated captions are converted to lower case, removed all the punctuations and all the short or numeric word were moved.
- Then we build our vocabulary by applying a threshold for more frequent words to include only words that occurred more than 10 times. In addition to his "startSeq" and "endSeq" were added to indicate sentence boundaries.
- Then we create our wordToIndex and IndexToWord matrix that were used for encoding word in numbers were index will start from 1 upto the maximum length of sentence in captions. These mappings are used to convert captions into numerical sequences suitable for training.
- Then we embed the words which were fed to embedding later using pre-trained GloVe embeddings (Global Vectors for Word Representation) with 200 dimensions. These embeddings provide dense vector representations of words that capture rich semantic and syntactic relationships, offering a meaningful initialization for the embedding layer in the decoder model.

2.3. Image Feature Extraction using CNN

I used a pre-trained Convolutional Neural Network (CNN) - InceptionV3 which was trained in image classification using the ImageNet dataset. The last two layers that were used for prediction were removed so that the image features could be extracted from the given image. This returns a 2048-dimensional vector representing the image's content as displayed in Figure 3.

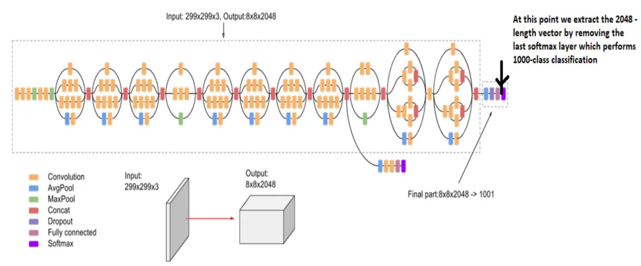

Figure 3. Feature Vector Extraction (Feature Engineering) from InceptionV3

2.4. Caption Generation using RNN

The word embeddings generated during the preprocessing step are fed into the Recurrent Neural Network (RNN) component of the model. This is responsible for processing the sequence of embedded words and learning the temporal dependencies between them — essentially modeling how words are structured in a grammatically and semantically correct sentence.

To explore the effectiveness of different sequence modeling techniques, I experimented with two widely used RNN architectures:

- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

The different configurations of the image captioning model explored in this project are described below:

Configurations 1: This is the baseline model configuration where a single-layer LSTM is used for predictions with 256 neurons. The model was trained using the Adam optimizer, which is effective for achieving faster and more stable convergence. The model was trained for 20 epochs, as shown in *Figure* 4.

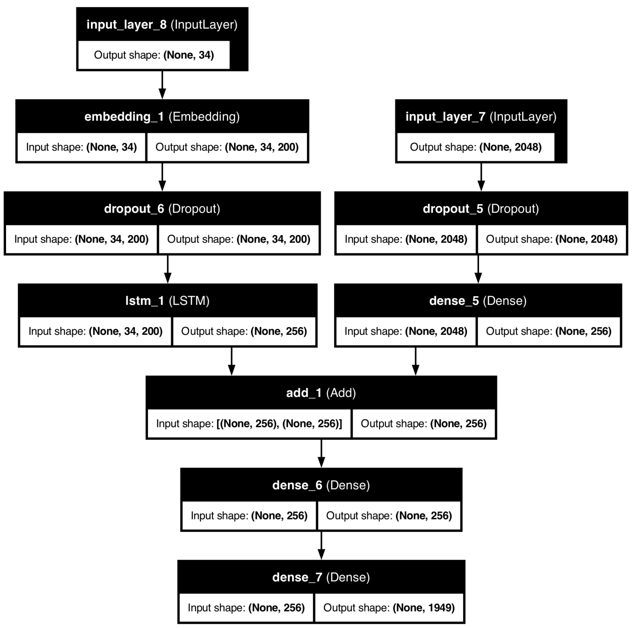

Figure 4. Baseline Model using LSTM

The choice of 20 epochs was based on observation — both the training loss and training accuracy stabilized by this point, indicating that the model had learned meaningful patterns without overfitting. Since the task is to predict the next word from a vocabulary of possible words at each time step, this is treated as a multi-class classification problem. Therefore, categorical cross-entropy was used as the loss function.

Configurations 2: To explore the performance of an alternative RNN architecture, I implemented the same encoder-decoder setup using a Gated Recurrent Unit (GRU) with 256 hidden units as displayed in Figure 5. GRUs are known for their computational efficiency and ability to capture temporal dependencies in sequences, often providing performance comparable to LSTMs with fewer parameters.

Unlike the previous configuration that used the Adam optimizer, this model was trained using Stochastic Gradient Descent (SGD as it updates the weights on small batches of data. While it may converge more slowly than adaptive optimizers like Adam, SGD offers greater control through its learning rate and often leads to better generalization. The model was trained for 20 epochs and the same architecture and preprocessing steps as Configuration 1 were retained to enable a fair comparison. The categorical cross-entropy loss function was again used, as the task remains a multi-class prediction problem.

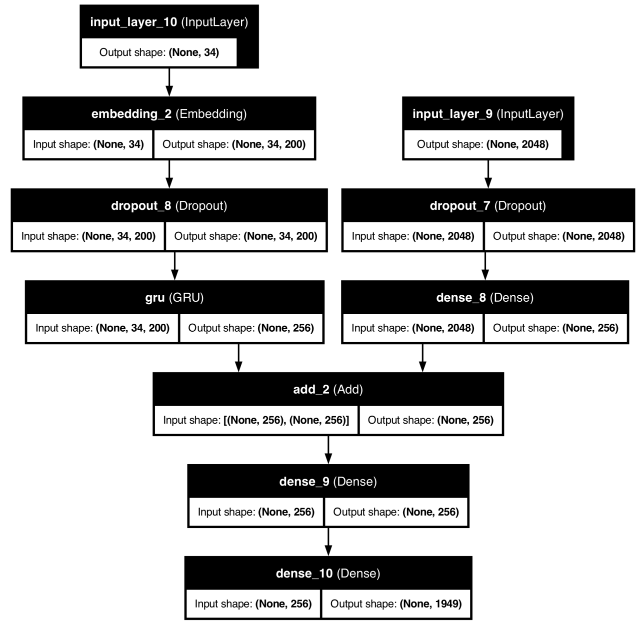

Figure 5. Baseline Model using GRU

Configurations 3: The baseline GRU model illustrated a considerably poorer performance than the baseline LSTM model, as anticipated because GRUs are lighter and faster, but often have difficulty maintaining longer-range dependencies in sequences compared to LSTMs. To help mitigate this limitation, I created a stacked GRU architecture with three GRU layers, each with 256 units, which will increase the model's ability to capture complex temporal dependencies and hold more context over longer sequences. The architecture of this model is shown in Figure 6.

To mitigate the risk of overfitting and improve generalization, dropout with a rate of 0.5 was used between the GRU layers. Dropout helps with regularization of the network by randomly dropping units during training so that the model learns more robust features. This model was also trained with SGD initially, as in Configuration 2, but I immediately recognized that, due to the increased deepness and complexity, the model was not converging well with SGD. I then switched to the RMSprop optimizer, which adapts the learning rate for each parameter and is useful for training deeper recurrent networks. The model was trained for 20 epochs, as with the other configurations, and categorical cross entropy was selected as the loss function, appropriate for the multiclass prediction.

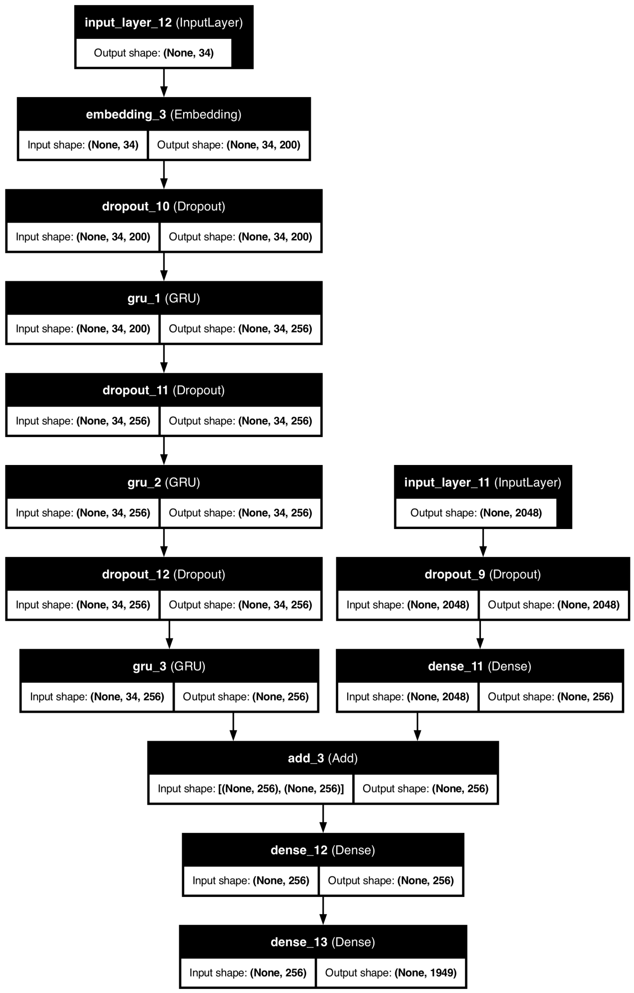

Figure 6. Stacked Model with 3 GRU Layers

Configurations 4: After realizing the significant improvement in performance of my stacked GRU model I extended the same concept to the LSTM architecture by implementing a stacked LSTM model with three LSTM layers, each containing 256 units. The motivation was to leverage LSTM's superior ability to retain long-term dependencies and evaluate whether deeper architectures could further enhance caption quality. The architecture of this model is illustrated in Figure 7. To mitigate the risk of overfitting associated with deep LSTM networks, I added Dropout layers with a rate of 0.5 between each LSTM layer.

For optimization, I continued using the Adam optimizer, known for its efficiency and adaptive learning capabilities. However, to exert finer control over the training process and reduce the chance of overfitting in this deeper model, I manually adjusted the learning rate to 0.001. This adjustment provided more stable and gradual convergence. Despite these modifications, the performance improvement over the baseline LSTM was relatively modest. This suggests that while LSTM networks inherently handle long-term dependencies well, simply stacking more layers does not guarantee significantly better results — especially on smaller datasets like Flickr8k, where deep models may not generalize effectively. This configuration was essential in understanding the trade-offs between model depth and overfitting, and it provided a deeper insight into how LSTM behavior scales with complexity.

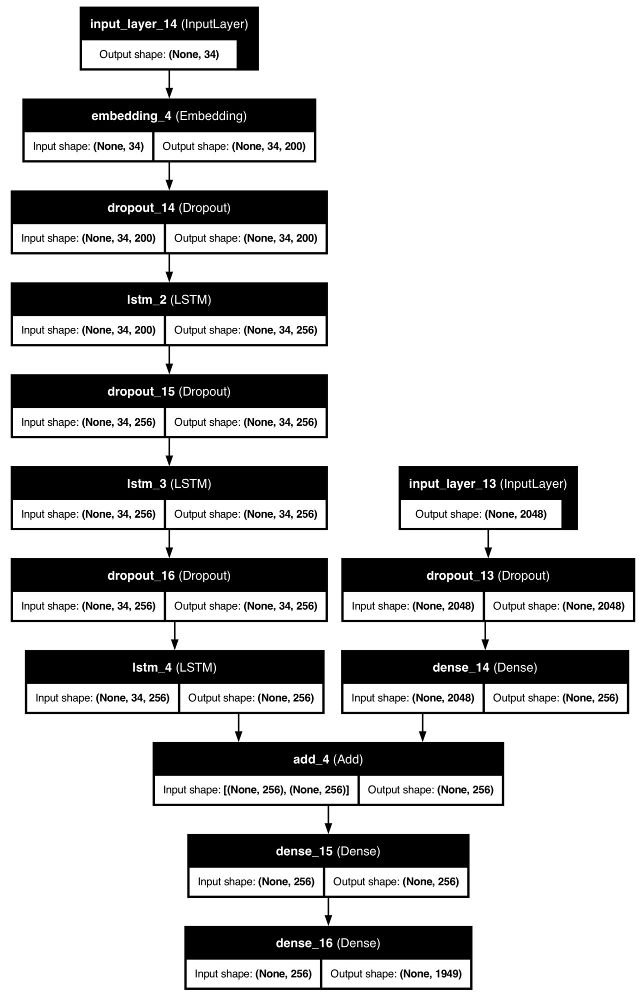

Figure 7. Stacked Model with 3 LSTM Layers

2.5. Output layers

The final stage of the model involves adding the image features extracted by the CNN with the contextual information generated by the RNN using an add() layer. The merged vector is then passed through a fully connected (Dense) output layer, which produces a probability distribution over the vocabulary. Based on this model predicts the most likely next word in the sequence, given the image context and the previously generated words.

2.6. Training

To train the model, I provided the model with the image features as well as the caption sequences. Since the feature vector (image + caption input) can be large, especially with high-dimensional image embeddings and word sequences, it is not feasible to load the entire dataset into memory at the same time.
To address this issue, the training data is divided into mini batches. Each batch consists of:

- The image feature vector extracted by the CNN (InceptionV3)
- The partial caption sequence up to the current word
- And the target word the model should predict next.

2.7. Evaluation

I employed the BLEU score to assess the quality of the generated captions. The BLEU score is a conventional comparison of the generated caption to one or more reference captions, based on n-gram precision (1-gram to 4-grams), for all cases of the models.

3. Experimental Results

After training each model configuration, I evaluated their performance by generating captions for unseen images from the test set.

3.1. Predicted Caption with Images

The following examples from each model show the predicted caption alongside the input image that was presented to the trained model. These examples are useful in determining how well each architecture is modeling the visual input and extracting meaning in language. All models were run on the same set of images for comparison here. All predicted captions indicate the model's understanding of objects, actions, and relationships in the image.

Figure 8. Generated Caption from base LSTM model

Figure 9. *Generated Caption from base LSTM model using Greedy Search*

Figure 10. Generated Caption from base GRU model

Figure 11. Generated Caption from stacked GRU model

Figure 12. Generated Caption from stacked LSTM model

Table 1 showcases the sample captions predictions produced by each model configuration on the test set. These instances show how each model makes sense of visual content and transcribes that understanding into descriptive natural language.

Model with Configuration	Predicted Caption Samples
Base LSTM model with Optimizer as Adam	“brown dog is running on the sand”, “three girls lay on the end of the water”, “man in bikini and shorts is walking along the street”, “man sits on bench with dog in the background”, “the football player in the red helmet is holding football”
Base LSTM with greedy searching with Optimizer as Adam	“brown the the the brown the the the the the the the the the the the the the the the the the the the the the the the the the”, “three three”, “man shirtless” “man man people couple at at sits at at sits sitting at at sits sitting at at sits sitting at at sits sitting at at sits sitting at at sits sitting at at” “the the the the the football the the the football hugging the the the the the the the the football hugging the the the the the an the the the the the in”
Base GRU model with Optimizer as SGD	“”, “”, “”, “”, “”, “dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog”
Stacked GRU with 3 stacked layers with Optimizer as RMSprop with learning rate of 0.01	“two dogs are running through the grass”, “dog is running in the grass”, “two children are playing in the air”, “two children are sitting on sidewalk”, “two soccer players are playing in the grass”
Stacked LSTM with 3 stacked layers with Optimizer as Adam with learning rate as 0.001	“brown dog is digging on the sand”, “two girls are wrestling in pool”, “man in red shirt is standing next to large red bottle”, “two people sit on bench with dog”, “two football players are in the middle of the middle of the field”

Table 1. Sample Predictions from each model

3.2. Evaluation of Model Performance Using BLEU Score

Tables 2 present the BLEU-1 to BLEU-4 scores for each configuration of models and allow a quantitative comparison of their performance. These scores emphasize the extent to which the models output accurate words and structure in generated captions, starting from the individual word correctness (BLEU-1) to the fluency of sentences (BLEU-4).

Model	**BLEU-1 weights= (1, 0, 0, 0)**	**BLEU-1 weights= (0.5, 0.5, 0, 0)**	**BLEU-1 weights= (0.3, 0.3, 0.3, 0)**	**BLEU-1 weights= (0.25, 0.25, 0.25, 0.25)**
Base LSTM optimizer= Adam	0.459178	0.268736	0.188599	0.089790
Base GRU optimizer= SGD	0.007531	0.000000	0.000000	0.000000
Stacked LSTM with 3 LSTM stacked optimizer= Adam(learning_rate=0.001)	0.492602	0.305951	0.223007	0.115165
Stacked GRU with 3 GRU stacked RMSprop(learning_rate=0.01)	0.385073	0.209112	0.142176	0.142176

Table 2. Blue Score for each model

3.3. Training Loss and Training Accuracy

The graphs below represent a summary of the training performance for each model architecture, graphed as training loss and accuracy versus the number of epochs. This helps in visualizing the learning of the model over time and allow for comparisons for convergence trends, variability or stability, and potential overfitting or underfitting.

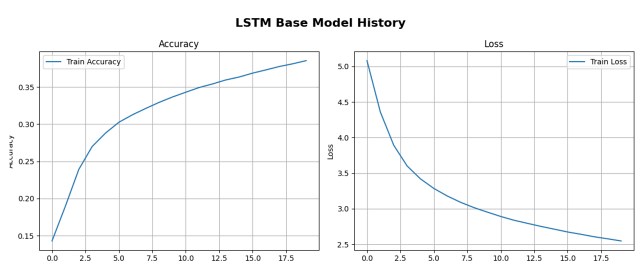

Figure 13. Training Loss and Accuracy for Base LSTM model

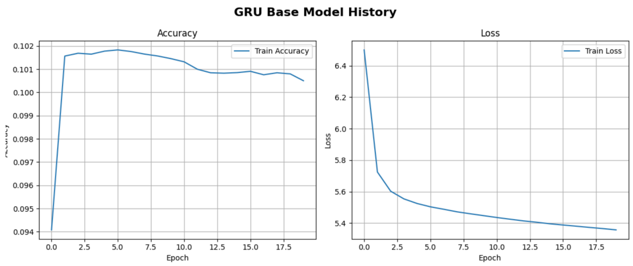

Figure 14. Training Loss and Accuracy for Base GRU model

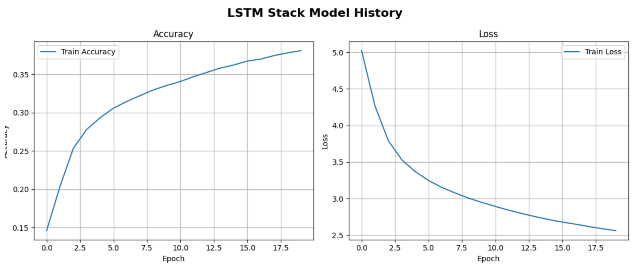

Figure 15. Training Loss and Accuracy for Stacked LSTM model

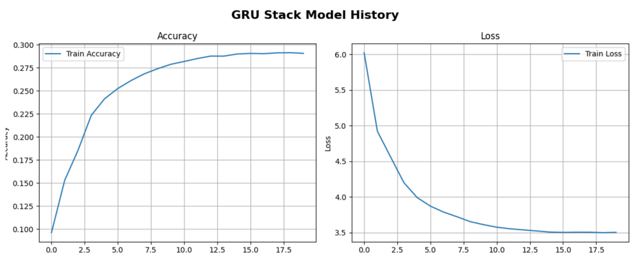

Figure 16. Training Loss and Accuracy for Stacked GRU model

4. Discussion

4.1 Predicted Captions Results

When generating the caption, I considered two decoding approaches:

- Greedy decoding, where the highest probability word is selected at each time step.
- A manual, step-by-step, prediction loop, allowing for more control of the word generation and debugging.

For the Baseline LSTM model, I first tried greedy decoding. I noticed that the model was generating a lot of repeated outputs, such as predicting the same word multiple times in a row as shown in Table 1 and Figure 9. This was a sign that the model had not generalized sufficiently and was too confident when predicting certain tokens. The repeated predictions were likely due to the model being limited in learning a range of sequence transitions. Because of these issues, I decided not to use greedy decoding in any of the models. Instead, I had a custom prediction method, where I could generate captions word by word and could observe the model outputs in more detail. This allowed for more stable and interpretable results across the configurations. After this I generated captions for each of the four model configurations, I was able to observe some notable differences in the output quality, structure and overall accuracy for context that discussed below:

- Model 1: For Model 1, which used a single-layer LSTM with 256 units, the model was able to produce captions that were generally grammatically correct and coherent. The captions that it predicted were constantly between 5 to 8 words, which aligns well with the average caption length across the training data, meaning it was able to learn the rhythmic nature of sentence structures in the training data. However, the model occasionally had specific semantic opportunities for improvement. For example, in images that featured a single dog without a collar, the model tended to predict, "a dog with a black collar". These errors occur due to training bias or overfitting — the model had learned to correlate frequently identified patterns (dog, collar) regardless of whether the collar was present in that instance. In conclusion, while the baseline LSTM demonstrated a good ability to model sentence structures and predict frequent visual instances (dog with collar), it did sometimes hallucinate details that the image had not visually specified.

- Model 2: The results of Model 2 indicated that having a single-layer Gated Recurrent Unit (GRU) with 256 neurons didn't perform well compared to the LSTM baseline. In many cases, the model failed to generate suitable caption-words at all: it either returned with empty captions or repeated the same word, e.g., "dog dog dog...". Repetition and lack-of-sentences is reflective of the GRU's short capacity for long-term context retention when used in shallow setup. This can also indicate that the model struggled to learn the temporal aspects of generating words in order to learn how to form sentences during training. Overall, the baseline GRU lacked the stability and language signature understanding of the LSTM and could not generalize across different image contexts.

- Model 3: Model 3 represented an important upgrade from the baseline GRU structure. For this model, I stacked three GRU layers of 256 units each and added a dropout rate of 0.5 to reduce overfitting. Another important change was switching the SGD optimizer for RMSprop, which is ideally suited to training deeper recurrent networks based on its adaptive learning. Overall, this architecture led to an observable improvement in performance. In contrast to the GRU baseline, the stacked GRU was able to produce longer, increasingly cohesive, and contextually accurate captions, comparable to the baseline LSTM. That said, while there was generally improved performance, the model still displayed certain obvious inconsistencies. For example, the model had a tendency to begin with "two" regardless of whether or not multiple subjects were depicted in the image, hinting that the model had learned structural conventions well, but was still prone semantic hallucinations or the biases of the dataset. Overall, the fairly strong performance of this stacked GRU architecture warranted continuing exploration of deeper LSTM architectures with the potential to achieve even greater improvement improved accuracy and stability of captioning results.

- Model 4: This model set up was the overall best-performing in terms of fluency and semantic coherence. For Model 4, I stacked three LSTM layers of 256 units and used a dropout of 0.5 between layers for regularization and to lessen overfitting. The model was trained using the Adam optimizer at a learning rate of 0.001 for stable and efficient convergence. The generated predictions from the model were generally correct, fluent, and semantically correct, showing improvement in contextual understanding and diminished repetition compared to prior models. Though improvements were made in the LSTM architecture, the gain in improvement was minimal compared to the baseline LSTM. These observations indicate that more complex and deeper LSTM architectures can improve the model capacity, but the scope in performance gain experienced was not as substantial as that seen moving from the baseline GRU to a stacked GRU, which potentially also reflects the restrictions in dataset size (Flickr8k) in which more complexity in the model does not lead to greater generalization.

4.2 Predicted Captions Results

BLEU (Bilingual Evaluation Understudy) score is a popular evaluation metric for natural language generation tasks, including machine translation and image captioning. It quantifies the similarity between a sentence produced by a model (the hypothesis) and one or more reference sentences produced by humans. I sued this matric to evaluate the model and below were the results:

- The **Stacked LSTM model** achieved the highest scores across all BLEU metrics, indicating the strongest overall caption generation quality.
- The Base GRU model showed very weak performance, with a BLEU-1 score nearing zero and a zero score for BLEU-2, BLEU-3, and BLEU-4. This indicates that the model hardly produced meaningful n-gram sequences and was unable to match up to even simple bigram structures from the reference captions.
    - The low scores can be attributed to both the limited learning capacity of a shallow GRU and the use of the SGD optimizer, which struggled to converge effectively.
- The Stacked GRU model, by contrast, showed major improvements, particularly in higher-order BLEU scores, suggesting that depth and better optimization (RMSprop) helped it retain longer-term context.
- The Base LSTM provided a solid baseline, while the Stacked LSTM slightly improved upon it — indicating that LSTM models remain more stable and expressive across both shallow and deep configurations.

4.3 Training Loss and Accuracy

- Model 1: For the model 1 with base LSTM, the continual uptrend in accuracy and downtrend in loss indicates that the LSTM base model was both well-optimized and stable. The curves both flatten in the end, indicating that training for more than 20 epochs would produce minimal gains unless adjustments were made to learning rates or monitoring validation.
- Model 2: The GRU base model did not reach any significant learning during training, reflecting the very low to zero BLEU scores we obtained. The poor performance is likely a result of:
    - The shallow GRU architecture (single layer and shallow capacity).
    - The SGD optimizer

The accuracy curve and loss both indicate the model likely reached a learning ceiling quite quickly and was not able to generalize well.

- Model 3: The stacked GRU model exhibited considerable advancements compared to the GRU baseline, which presented almost zero BLEU scores and relatively constant training curves. While the same forms of training were observed, there were no signs of overfitting, and the training process was uniform and smooth enough that we can conclude that RMSprop was indeed a significantly better choice of optimizer for GRU over SGD. While the stacked GRU approach did not perform as well as the LSTM models, it is clear that the stacked GRU closes the performance gap enough that it is still a reasonable alternative when configured appropriately.
- Model 4: The training curves for the stacked LSTM are very similar to the base LSTM, but they appear to have slightly more stable and slower you increase, because of the increased depth of the model and lower learning rate. Despite the size of the model, there are no signs of overfitting in the curves, likely due to the use of dropout layers and tuning of the learning rate.

Overall, the minor improvement in accuracy and in loss compared to the base LSTM suggests that stacking LSTMs doesn't inherently improve performance - a finding supported by the BLEU scores.

5. Conclusion

In conclusion during this work I investigated the task of generating image captions using a standard encoder-decoder framework with CNNs used for image feature extraction and RNNs used for generating sequences of captions. I learned how different choices of model depth, optimizer, and regularization impact the quality of the generated captions, a series of experiments were conducted with LSTM and GRU models, in both baseline and stacked configurations. As a result of the experiments, it is clear that the model depth, optimizer choice, and regularization are all important decision points influencing the quality of the generated captions in some way. Results indicate that the best performance overall came from the stacked LSTM model, while the stacked GRU model produced competitive results (with correct configuration). The research shows how model architectures and strategic training can greatly affect performance when building multimodal language models.

References

[1] https://medium.com/data-science/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8
[2] https://www.digitalocean.com/community/tutorials/bleu-score-in-python
[3] https://www.kaggle.com/datasets/adityajn105/flickr8k
[4] https://learn.microsoft.com/en-us/azure/ai-services/translator/custom-translator/concepts/bleu-score