Image Caption Generation with Computer Vision, NLP and High Performance Computing

AHASAN HABIB[1]

[1]Department of Computer Science, BRAC University, Dhaka, Bangladesh
Email: ahasan.habib@g.bracu.ac.bd

Abstract

In this report, the construction of and improvement of a model of image captioning have been outlined, built using an image captioning model built around a Convolutional neural network (CNN) and a Long Short-Term Memory network (LSTM). The aim was to develop a more intense and effective image captioning model by presenting a description on how the code has been changed to suit and increase effectiveness. The objective of the project is to design a powerful and effective model that would help produce a natural language description of a specific image automatic. This work is based on publicly available proof-of-concept project that was, thorough, improved by High-Performance Computing (HPC) and parallel computing methods. The main changes were the adoption of mixed precision (mixed float16), the implementation of XLA Just-in-Time (JIT) compilation and the use of a distributed training strategy with tf.distribute.MirroredStrategy. A tf.data.Dataset.from-generator was also optimized on the data pipeline to work with larger batches and have better efficiency. The project contains visually analyzed and outlined data.

Keywords: Image Captioning, Deep Learning, CNN, LSTM, High-Performance Computing.

1 Introduction

Traditional vision systems, which necessitate hand-constructed features has been a thorn in the side of the field of computer vision. This project aims to rationalize by using deep learning, namely CNN-LSTM architecture that is used to solve the visual object recognition and natural language generation problems. The methodology of the project derives the motivation of a prior paper on the ImageNet classification, which had shown the power of deep CNNs in the process of object recognition when trained on massive datasets. The contributions realized in this paper, including the application of ReLU, dropout, and parallel training using several GPUs were the basis of what was adopted in this project.

This project had a goal to evolve the primitive image captioning model into a more effective, scalable and analytically detailed solution. The basic model offered a basic proof-of-concept, and did not include optimizations required to reach high performance and scale with larger datasets. The alterations were aimed at significantly decreasing the training time and the exhaustion of resources which represent the significant constraints of deep learning projects.

2 Methodology

It has been created in a Google Colab setup and the data is stored on Google drive. Preparation of the data, feature extraction, building the model, training and evaluation of the core are one of the keys of the project.

2.1 Dataset and Exploratory Data Analysis (EDA)

The project utilized the Flickr8k Image Dataset, which is a benchmark dataset for image description tasks. It consists of 8,092 images, with each image accompanied by up to five human-written captions. The dataset contains 40,455 entries with 'images' and 'captions' as features. There are 8,091 unique images and 40,201 unique captions.

A comprehensive EDA was performed to understand the characteristics of the dataset. The caption lengths were analyzed, and a histogram showed that most captions are between 10 and 15 words long, with a peak frequency of over 4,000. The mean caption length was found to be 12.78, the median 12.00, and the mode 12. A word cloud was also generated to visualize the most frequent words, with "startseq" and "endseq" being the most common, followed by words like "on," and "dog".

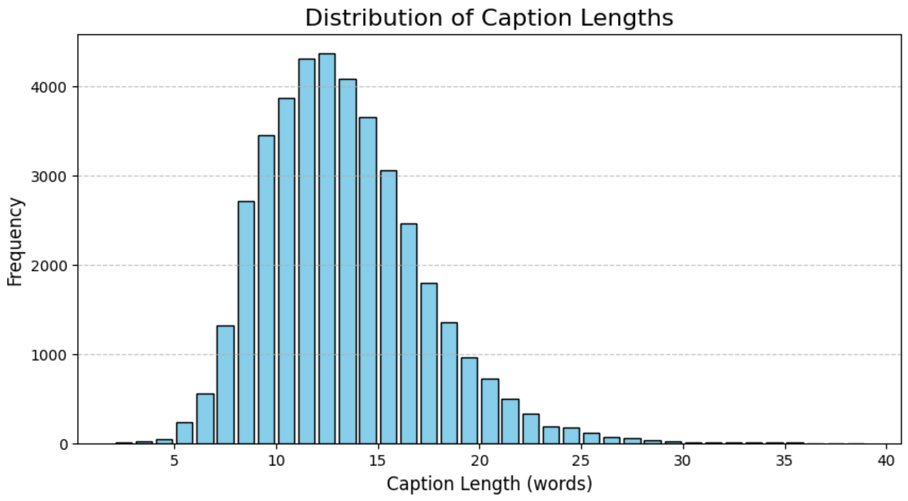


Figure 1: Distribution of Caption Lengths in the Flickr8k Dataset.

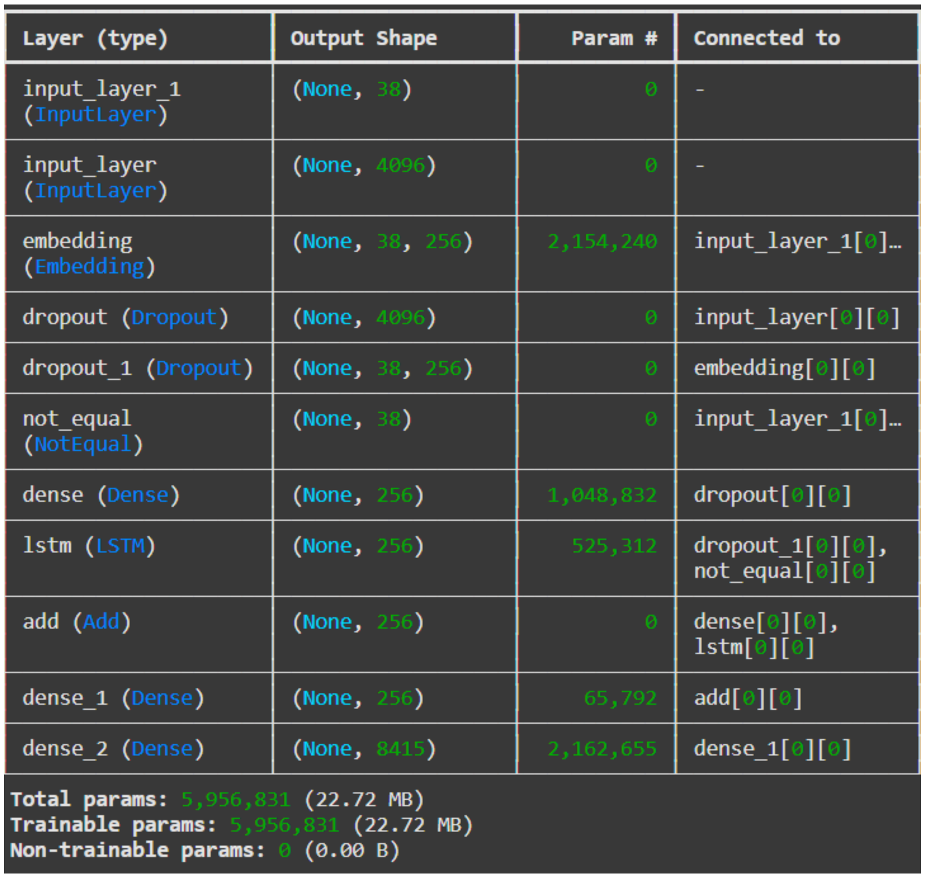
2.2 Data Loading and Preprocessing

The captions were loaded by a captions.txt file, and assigned image IDs. All captions were processed by lowercasing them and adding startseq and endseq tags at the start and the end, respectively. This step-by-step outline assists the model in knowing the start and the conclusion of a sequence. The captions were subsequently tokenized and this formed a vocabulary and words were coded to numbers. The size of the vocabulary was identified as 8415 and the largest length of the caption was.

2.3 Model Architecture

An imagenet weighted VGG16 model was used to extract image features. The VGG16 model was set up with the last output layer removed and the second-to-last layer output taken as the feature vector.The primary captioning model was made up of two input modules to handle the image features and text sequences simultaneously. The inputlayer (image input layer) had an output shape (None,4096), and the inputlayer1 (text input layer) (None,38).

The model architecture was prepared in the following way: The image characteristics were fed to a Dropout layer (rate of 0.4) and a Dense layer consisting of 256 neurons and ReLU activation. An Embedding layer processed the text sequences converting the integer-encoded words to dense vectors. This was then succeeded with a Dropout layer (0.4) and an LSTM layer of size 256 units. An add layer was used to combine the output of the Dense and LSTM layers. This output was again fed through another Dense layer of 256 neurons and a final Dense layer of vocabsize (8415) neurons with a softmax activation in order to predict the next word in the sequence. There were 5,956,831 parameters to the model.



Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 38)	0	-
input_layer (InputLayer)	(None, 4096)	0	-
embedding (Embedding)	(None, 38, 256)	2,154,240	input_layer_1[0]…
dropout (Dropout)	(None, 4096)	0	input_layer[0][0]
dropout_1 (Dropout)	(None, 38, 256)	0	embedding[0][0]
not_equal (NotEqual)	(None, 38)	0	input_layer_1[0]…
dense (Dense)	(None, 256)	1,048,832	dropout[0][0]
lstm (LSTM)	(None, 256)	525,312	dropout_1[0][0], not_equal[0][0]
add (Add)	(None, 256)	0	dense[0][0], lstm[0][0]
dense_1 (Dense)	(None, 256)	65,792	add[0][0]
dense_2 (Dense)	(None, 8415)	2,162,655	dense_1[0][0]

```
Total params: 5,956,831 (22.72 MB)
Trainable params: 5,956,831 (22.72 MB)
Non-trainable params: 0 (0.00 B)
```

Figure 2: The CNN-LSTM Model Architecture for Image Captioning.

3 High-Performance Computing (HPC) Implementation

The most dramatic ones were the incorporation of the concepts of HPC and parallel computing, which is missing the first online project. To allow mixed precision, this was enabled via the global policy of mixedfloat16 that uses a mix of float16 and float32 data types to achieve much faster training on supported hardware such as a GPU. The optimization of the TensorFlow graph to achieve a higher performance was enabled by running the tf.config.optimizer.setjit(True) command to set the XLA Just-in-Time (JIT) compilation. Mirrored Strategy was adapted to make use of parallel computing which makes the model to be trained effectively across a number of the GPUs. In order to exploit this arrangement, the batch size was raised to 256. Another refinement of the data pipeline was with the use of a tf.data.Dataset.from-generator to generate a more efficient input pipeline to process data in parallel and prevent the crash of a session.

4 Results and Analysis

The model was trained for 20 epochs with an estimated 126 steps per epoch.

4.1 Training Performance

The training history of the model was analyzed to evaluate its learning progress over time. The model's loss, a measure of how wrong the model's predictions are, consistently decreased over the training period. This indicates that the model was successfully learning to make more accurate predictions with each epoch. Conversely, the model's accuracy, a measure of how often the model predicts the correct next word in the sequence, steadily increased throughout the training process. This dual improvement in loss and accuracy demonstrates the effectiveness of the training process and the model's ability to learn from the dataset.

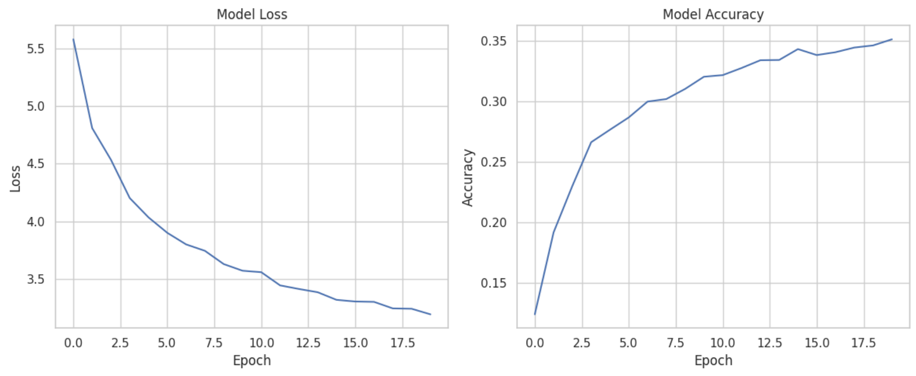


Figure 3: Model Loss and Accuracy over 20 Epochs.

4.2 BLEU Score Evaluation

The model's performance was evaluated on a test set using the BLEU score, a metric for evaluating machine-generated text against a set of human-written references. The evaluation was performed using a corrected, batched greedy decode function. The results were as follows:

1. BLEU-1: 0.5032
2. BLEU-2: 0.3195

3. BLEU-3: 0.1898
4. BLEU-4: 0.1052

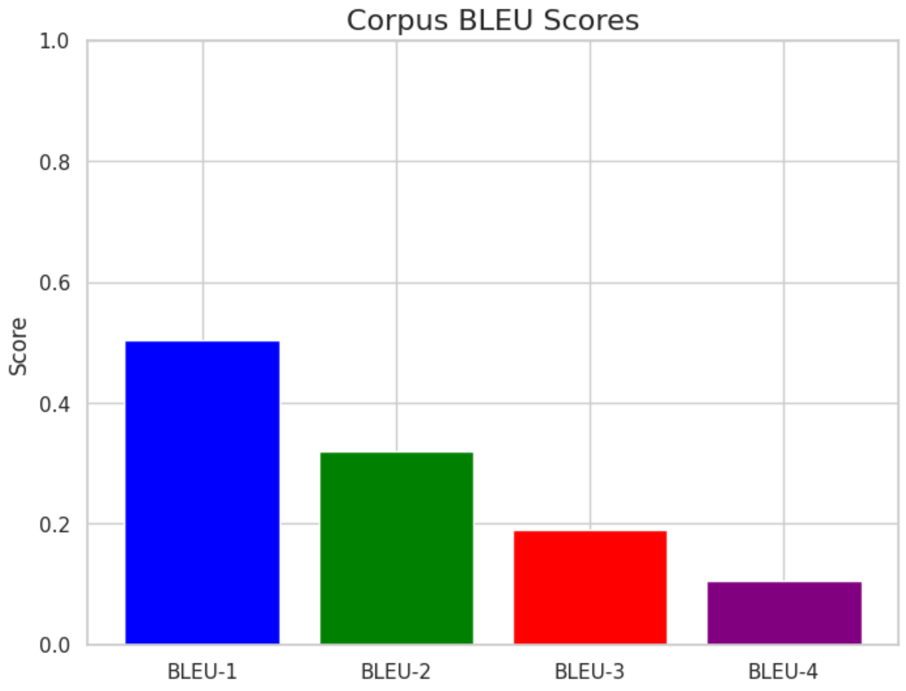


Figure 4: Corpus BLEU Scores for the Trained Model.

4.3 Explainable AI (XAI) and Validation

In order to gain further insight into the behavior of the model, the confusion mat was calculated on a small sample of the test (50 images). The model can see its forecast performance displayed through this matrix of the differences between the forecasted next token and the real next token against the reality captions. The existence of a large number of vocabularies necessitated the focus of the matrix on the 20 most common actual tokens, which offered a viable perspective on what tokens the model something predicted correctly or incorrectly. This may be used as a method of validation, providing a finer account of the strengths and weaknesses of the model than an individual score, such as the BLEU score.

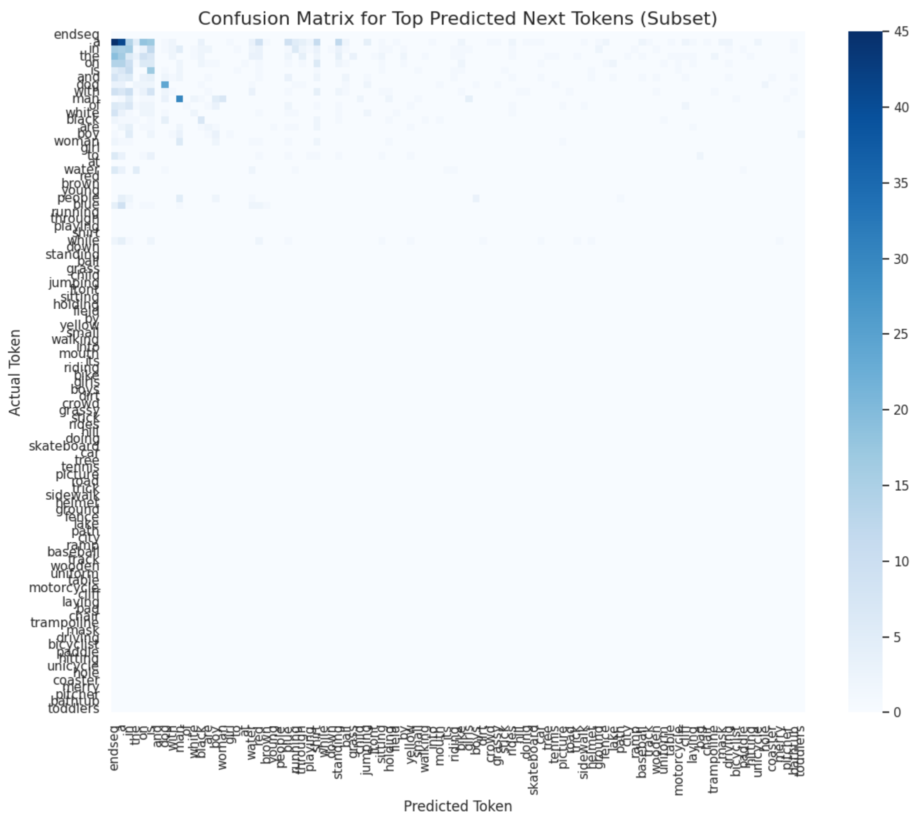
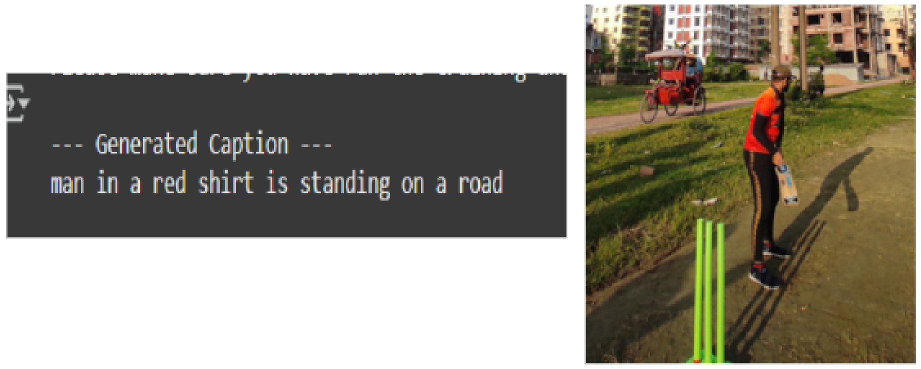


Figure 5: Confusion Matrix for Top Predicted Next Tokens.

5 Live Demonstration

The trained model was offered a new image and shown in its application to the real world. Features of the image are extracted by using VGG16 model and caption then constructed word by word was produced by LSTM, depending on features and vocabulary acquired. The produced caption was: man in a red shirt is standing on a road. The live demonstration revealed that the model perceived the prominent components in the image such as the individual, his or her attires, and the setting. This is evidence of the fact that this model will produce a plausible semantically relevant caption, which confirms its usefulness within a real world setting.



```
--- Generated Caption ---
man in a red shirt is standing on a road
```

Figure 6: A live demonstration showing the model's generated caption for a new image.

6 Discussion and Conclusion

The changes were effective in turning around a basic demonstration and making it more efficient as well as analytic. The implementation of HPC functions together with a sophisticated data pipeline led to a more scaled model. Complete visualizations and detailed data analysis give a further insight into both data and the decision of the model, so the final project will be more complete.

Performance of the project (as assessed using the BLEU scores) represents a favorable achievement, the goals of the BLEU-1 and BLEU-2 scores are particularly high, as they suggest a great number of intersections between the individual scores and the reference captions.

6.1 Limitations

Although there is the improvement, the model has limitations. The large labeled dataset that is crucial to the performance is questionable in relation to the need to promote the practice to areas where the labeled data is limited. The use of center-cropping and fixed image sizes could equally result in the loss of critical image content and compromise generalization in the real world. In addition, the HPC optimizations reduced the duration of training, albeit, the mechanism is computationally intensive, and consumes heavy hardware resources. Another additional error encountered with the model during loading is an Unknown layer: NotEqual that simply represents a problem that may be experienced in the saving or loading of the model architecture.

7 Future Work

According to the limitations of the project, it is possible to focus on the areas in the future work and improve the areas related to the work of the model more. This involves engaging in other model architectures, either transformer-based models, which have demonstrated an encouraging success in both computer vision and natural language processing. More powerful image preprocessing pipelines that support image sizes even with changes without absorbing vital information should be applied also. By optimising the training process further, it would become more usable by users with less access to computing power. Lastly, the model loading error must be addressed by formulating a solution that will allow the trained model to be deployed with robustness.

References

[1] Jambhalae, J., Sangale, S., Avhad, A., Vairagade, P., Kotwal, J. (2022). *Image caption generator using convolutional neural networks and long short-term memory*. International Research Journal of Modernization in Engineering Technology and Science, 4(5), 4664-4669. `https://www.irjmets.com`

[2] Parate, D., Choudhary, M. (2022). *Image Caption Generator using deep learning with Flickr Dataset*. International Journal for Research Trends and Innovation, 7(8). `https://www.google.com/search?q=https://www.ijrti.org/viewpaperforthis%3Fval%3DIJRTI2208183`

[3] Sawant, V., Mahadik, S., Sisodiya, D. S. (2023). *Review paper image caption generator using an artificial intelligence*. International Journal of Advanced Research in Science, Communication and Technology, 3(1), 199-202. `https://www.ijarsct.co.in`

[4] Ramkeesoon, R. (2019, December 24). *Benchmarking data: Parallel distributed training of deep learning models in PyTorch and TensorFlow*. Analytics Vidhya. `https://medium.com/analytics-vidhya/benchmarking-data-parallel-distributed-training-of-deep-learning-models-in-pytorch-and-`