ImageNarrative: From Conventional to Cutting-Edge Captioning

1st Hussain Kanchwala
MS in Robotics, Mech Dept.
Northeastern University
Boston, USA
kanchwala.h@northeastern.edu

2nd Abdulaziz Arif Suria
MS in Computer Science
Northeastern University
Boston, USA
suria.a@northeastern.edu

3rd Prem Sukhadwala
MS in Robotics, EC Dept.
Northeastern University
Boston, USA
sukhadwala.p@northeastern.edu

4th Surabhi Gade
MS in Robotics
Northeastern University
Boston, USA
gade.su@northeastern.edu

***Abstract*—This paper presents a comprehensive exploration of image captioning architectures, focusing on the implementation and comparative analysis of CNN-RNN models, attention-augmented CNN-RNN frameworks, and advanced transformer-based approaches utilizing ViT-GPT2. We have developed the CNN-RNN and its attention-enhanced variant from scratch to understand their underlying mechanisms and performance nuances in image captioning tasks. Additionally, we have adopted the state-of-the-art ViT-GPT2 model from Hugging Face to examine the efficacy of transformer architectures in generating coherent and contextually relevant captions. Our study not only delineates the architectural differences and improvements but also provides insights into the evolving landscape of neural network applications in image captioning.**

Please visit our GitHub Repository for more information and source codes.

I. Introduction

Image captioning deals with the task of generating textual descriptions from visual inputs, this field stands at the intersection of computer vision and natural language processing. The area has witnessed substantial growth, fueled by advances in deep learning architectures that facilitate a more nuanced understanding of visual content. The primary objective of this project is to delve into three core architectures for image captioning: the convolutional neural network-recurrent neural network (CNN-RNN) model, its extension using an attention mechanism based on the pioneering work of Bahdanau et al. (2014) and the state of the art Vit-GPT2 based transformer model.

The CNN-RNN model effectively extracts visual features using CNNs and generates corresponding textual descriptions through RNNs. To address its limitations in dynamic feature utilization, an attention mechanism is integrated, enabling the model to focus on relevant image regions dynamically, much like human vision. This enhancement allows for more contextually accurate captions.

Further advancing our approach, we incorporate the Vit-GPT2 model, which combines Vision Transformers (ViT) with the GPT-2 language model to produce context-aware captions. This state-of-the-art model leverages the strengths of both visual and textual processing, aiming to outperform traditional models in generating coherent and contextually relevant captions.

This exploration not only implements CNN-RNN and it's variant with attention from scratch but also conducts a thorough comparative analysis to elucidate how the introduction of attention enhances the descriptive quality and relevance of generated captions. By doing so, it contributes to ongoing discussions in the field regarding the optimization of neural network architectures for the integration of visual and linguistic information, paving the way for more sophisticated image understanding systems.

We have also evaluated our system based on the rouge score metric specifically rouge-l f1-score to keep track of longest common subsequences of predicted and original captions and get a score on how well each model performs. Apart from that we also compare the best scoring images and the worst scoring images. We also present a small analysis of comparing the most frequent words in the predicted captions and in the actual captions to evaluate all our models based on such metrics.

Through this work, we aim to advance the understanding of how different neural network architectures can be optimized for the task of image captioning, an endeavor that has significant implications for applications ranging from assistive technologies to automated content generation and beyond.

II. Related Work

Paper I : Image Captioning Encoder–Decoder Models Using CNN-RNN Architectures: A Comparative Study Link to the paper (CLICK)
This paper explores different architectures involving CNN and RNN networks, They explore different architectures such as VGG, ResNet and also explore their different combinations with different decoder architectures such as LSTM, GRUs, etc. They also use the ROUGE-L metric similar to us in their evaluation procedures.

Paper II: Neural Machine Translation by Jointly Learning to Align and Translate Link to the paper (CLICK) The seminal paper "Neural Machine Translation by Jointly Learning to Align and Translate" by Bahdanau et al. (2014) introduced a groundbreaking attention mechanism that enables a model to dynamically focus on different parts of the input

sequence for better translation accuracy. This mechanism was a departure from previous models that processed input sequences in their entirety in a fixed order, thus limiting their ability to manage long input sequences effectively. By learning to align and translate simultaneously, the model can generate contextually relevant translations with improved precision. Inspired by this concept, we adapted the attention mechanism to our CNN-RNN based image captioning model, enhancing its ability to focus on pertinent regions of an image for generating more precise and contextually rich captions.

Paper III: Attention Is All You Need Link to the paper (CLICK) The paper titled "Attention Is All You Need" by Vaswani et al., published in 2017, introduces the Transformer architecture, a novel approach that departs from the previous sequence processing models which relied heavily on recurrent layers. Central to the Transformer is the self-attention mechanism that allows the model to weigh the importance of different words in a sentence, irrespective of their positional distances. This architecture significantly improves performance in machine translation tasks by enabling the model to handle long-range dependencies with ease. Additionally, it enhances training speed owing to its parallelizable structure, eschewing recurrence and convolutions entirely. In our project, we utilized a pre-trained Vit-GPT2 based image captioning model, drawing on the seminal principles introduced in the "Attention Is All You Need" paper.

III. Data Preparation and Analysis

We have used the Flickr8k dataset for the task of image captioning. The dataset can be accessed by following this link from Kaggle : Link to dataset (CLICK)

We have divided the Flickr8K dataset into the 3 following parts:-
Training dataset : 7091 Images
Validation dataset : 500 Images
Testing dataset : 500 Images

Each image consists of 5 captions and we have created a dataframe format to read and display the images. We have used a random state of 42 while splitting the images into their respective splits.

We have also plotted the average length distributions of the captions on the full dataset which can be visualized below

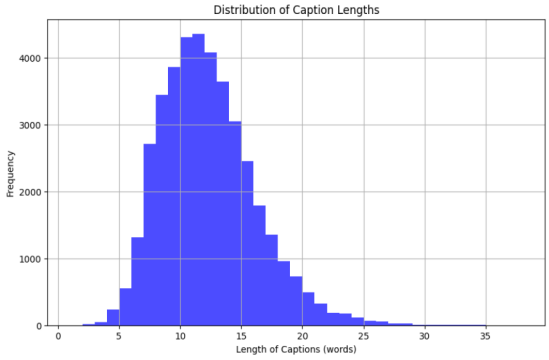

Fig. 1. Caption Length

The maximum length of the caption is 38 words and the average length is somewhere between 10-15 words. We can keep this in our mind when we are training our models. Here is an example of image visualized from the training dataset with one caption picked out of 5 for each image: -

Fig. 2. Original Image

IV. Methods and Model Architectures

A. *CNN-RNN Model Description*

The first model we have developed is a basic encoder - decoder architecture consisting of Google Inception v3 Model as our CNN Encoder and 2 layers of LSTM (Long Short Term Memory) as our decoder architecture. As we have 5 captions for each image, we have adopted some data augmentation where we randomly crop the image by dimensions of 299 x 299 and also normalize it by 0.5 in each dimension as that makes the input image compatible to the input format of the Inception v3 Model. We have also created a Vocabulary building class which takes in the captions from the training dataset and performs the following encoding for each caption
START TOKEN : <s>
END TOKEN : </s >
PAD TOKEN : <PAD >
UNK TOKEN : <UNK >
We are using the spacy tokenizer for tokenizing the text and if the frequency of a word is less than 5 i.e. our threshold frequency then we map the word with an UNK token. Each of these special tokens are indexed from 0 to 3 as special tokens in our vocabulary.

We also have adopted a collate function which pads each caption to it's maximum length for all caption lengths provided in a batch. Our dataloader function helps us prepare our data by performing all the mentioned preprocessing steps. We don't perform the random-cropping on the validation and test dataset when we load it using the dataloader.

*1) **Encoder CNN**:* The Encoder uses Google's Inception v3 Model pretrained on ImageNet dataset as a backbone to extract feature representation from input Images. Here, only the last fully connected layer is modified and we have additionally applied a Dropout layer with p=0.5 and an activation function ReLU. This architecture represents our Encoder CNN which helps us to extract the spatial information from an image and

also prevents overfitting. Here's the updated last layer of the model presented in the image.

```
  (fc): Linear(in_features=2048, out_features=512, bias=True)
)
(relu_activation): ReLU()
(dropout_layer): Dropout(p=0.5, inplace=False)
```

Fig. 3. Encoder Modified Layer

*2) **Decoder RNN**:* We use 2 layers of LSTM with embedding size 512 and hidden size 256. As LSTMs are effective at handling sequences of data. As we are focusing on image captioning, the LSTM layers will help us capture the dependencies between the sequences of words. We have set the maximum caption length to 40 words as our maximum observed caption length is 38 words. The complete decoder architecture consits of an Embedding layer followed by 2 LSTM layers which are then followed by a Dropout and a fully connected Linear layer. The architecture can be quickly visualized by the image below:

```
(decoderRNN): DecoderRNN(
  (embedding): Embedding(2816, 512)
  (lstm): LSTM(512, 256, num_layers=2)
  (linear): Linear(in_features=256, out_features=2816, bias=True)
  (dropout_layer): Dropout(p=0.5, inplace=False)
```

Fig. 4. Decoder Architecture

There is a concept of **Teacher force ratio** where we pass the correct next word to the layer instead of the generated word from the LSTM. This forces the network to understand the correct words present in the caption. We are using a teacher force ratio of 1 i.e. we are always passing the correct words of the caption to the layers of the LSTM network.

Additionally, adding more layers to the LSTM network lead to shorter caption generation and overfitting to the training dataset, hence we decided to finalize the above mentioned architecture.

B. CNN-RNN with attention Model Description

The second model that we have developed we have added the attention mechanism to it. The images are transformed as mentioned in the encoder section and the vocabulary building class associates the words with their respective indexes. Additional vocabulary is added for start, end, pad and unknown as defined in the previous section. The data-loader function as in the previous cases prepares our data as mentioned in the previous case.

*1) **Encoder CNN**:* The Encoder uses a ResNet-50 model pre-trained on ImageNet dataset as a backbone to extract rich feature representation from the input images. The ResNet-50 model includes convolution layers that are adept at capturing spatial hierarchies in the images, is modified to exclude the final fully connected layer and the last max pooling layer. This modification allows the network to retain spatial information which is crucial for the attention mechanism in the Decoder.

Input: - The input is resized to 224x244 pixels and and tranformed to result in a image with a distribution with mean 0 and standard deviation of 1 across all channels.

Output:- Feature Maps of shape (batchsize,2048,7,7) reshaped to (batchsize,49,2048), effectiely preparing the set of 2048 dimensional feature vector for 49 spatial locations from original image.

*2) **Attention Module**:* The attention module takes these features and the current hidden state of the decoder's LSTM to compute a context vector that pinpoints areas of image most relevant to the generating the next word in the caption. This is achieved via:

- Linear transformation applied separately to the image features and the hidden state, aligning them to the same dimension.
- Applying a tanh activation to the sum of these transformed features, which is then passed through another linear layer to produce raw attention scores for each feature.
- The final context vector is computed as the weighted sum of the image features, with weights specified by the probability distribution. This context vector represents the specific image parts to be focused on for the next word generation

*3) **Decoder RNN**:* The decoder manages the sequence generation, where each step involves predicting a word based on the context vector provided by the attention modules and the previous hidden state. The sequence generation is achieved via:

- Starting from initial hidden state derived from the encoder output feature vector, the decoder generates one word at a time.
- At each step of the sequence it uses the current hidden state and the encoder's features to compute a new context vector via the attention module.
- The word corresponding to the highest probability is selected as the next word in the caption, and the process repeats until the the maximum sequence length is reached.

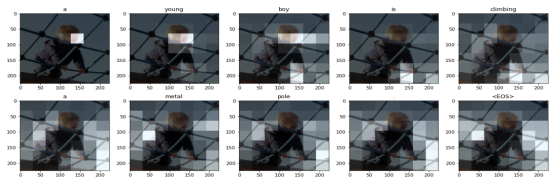

Fig. 5. Feature focus while caption generation

V. ViT-GPT2 Model Description

*1) **Encoder ViT**:* The encoder in this model employs the Vision Transformer (ViT). The ViT model used as the encoder

is pre-trained on the ImageNet dataset, a large and diverse dataset containing over a million images categorized into 1000 classes. The Vision Transformer, or ViT, is a model for image classification that employs a Transformer-like architecture over patches of the image.

Input: - **1. Patch Embedding** :- The encoder processes each input image, resized to 224x224 pixels, by dividing it into fixed-size patches. These patches are then linearly projected into high-dimensional vectors, aligning with the transformer architecture's requirements. **2. Transformer Layers** :- The ViT encoder further refines these vectors through its transformer layers, which utilize multi-head self-attention mechanisms and feed-forward neural networks. This design enables the encoder to discern complex inter-patch relationships and aggregate contextual information across the image. **3. Positional Encoding** :- Moreover, positional encodings are merged with the patch embeddings to instill spatial awareness within the model, offsetting the transformer's intrinsic indifference to sequence order.

Output:- The encoder outputs a sequence of feature vectors, each encapsulating detailed visual information from its corresponding image patch. These vectors, rich in contextual and local image details, are then fed into the GPT-2 decoder for generating coherent and contextually relevant captions, bridging the gap between visual perception and natural language description.

*2) **Decoder GPT-2**:* In this model, the decoder utilizes GPT-2, a state-of-the-art language model, for generating human-like text. It operates on the Transformer architecture, focusing on predicting the next word in a sequence through stacked decoders and masked self-attention mechanisms via: **Input Processing** :- The process begins with a special token that signifies the start of a text sequence, followed by the sequence of generated tokens thus far. GPT-2 processes the entire sequence, with positional encodings providing token order information. **Output Generation** :- Text generation employs methods like beam search, top-k sampling, or nucleus sampling for a balanced output. The model sequentially emits tokens until an end-of-sequence token appears or a maximum length is reached, producing coherent image captions.

- Starting from initial hidden state derived from the encoder output feature vector, the decoder generates one word at a time.
- At each step of the sequence it uses the current hidden state and the encoder's features to compute a new context vector via the attention module.
- The word corresponding to the highest probability is selected as the next word in the caption, and the process repeats until the the maximum sequence length is reached or the end token is achieved.

VI. Comparative Analysis and Results

In order to establish comparative analysis and result evaluation among our models, we have adopted the following sections of analysis:

- **Loss Results and Assessments:** Assessment of performance of the model based on cross-entropy loss.
- **Rouge Score Results:** Rouge Scores usually help us capture synonyms as well in contrast to BLEU scores. We are using the Rouge-L F-score metric to compare our captions.The ROUGE-L F-Score is a metric for evaluating text like image captions by measuring the longest common sub-sequence between generated text and a reference, balancing precision (relevance) and recall (completeness). It effectively captures the sequential flow and variations in wording, making it ideal for assessing the quality of image captions. For a given image we will derive the best scoring caption against the predicted caption and also plot the rouge scores values across all images.
- **Most frequent words in test captions and predicted captions:** Visualize the 50 most frequent words utilized by our predicted captions and verify how they differ with the most frequent words in the test caption dataset. We will compare the best score caption out of the 5 for a given image.
- **Best Results and Worst Results on Test Dataset:** Visualize the results of the best and worst images captured to derive conclusions and compare any specifics.

A. *CNN-RNN Model Results*

Loss Results and Assessments: We attempted different configurations of the decoder RNN architecture but increasing the layers lead to shorter caption generation and over-fitting of the training data hence, we followed the hyper parameters mentioned in the model description. Our loss function is CrossEntropyLoss and our optimizer is Adam. We ran this model on Kaggle GPU P100 and it took us 7 hours to complete 100 epochs at a learning rate of 0.0003.

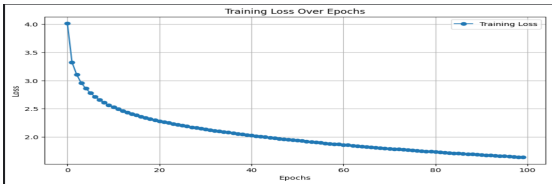

Fig. 6. Training loss

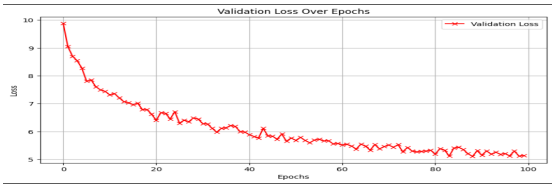

Fig. 7. Validation loss

The validation loss appears to be higher than the training loss throughout the iterations but the training loss appears to be steadily decreasing and validation loss is also decreasing throughout the iterations.

Rouge Scores Results: The Average Rouge score on testing dataset is 0.336 which is visualized on the 500 images as below whereas the average rouge score on validation set is 0.329 :

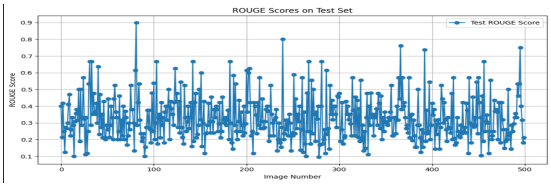

Fig. 8. Rouge-L F-score on Test

The model appears to be performing well on images of dogs as all the high scoring captions are on images of dogs. Additionally, the training dataset does consist of many images of dogs which allows it to capture enough context of these images.

Example Results: The top results with high rouge scores are displayed below with the actual and predicted captions:

Actual Caption : A black and white dog is running through grass .
Predicted Caption : a black and white dog is running through the grass .
Rouge Score : 0.842

Actual Caption : A black and white dog is jumping over a hurdle .
Predicted Caption : a black and white dog is jumping over a low beam .
Rouge Score : 0.8

Fig. 9. Results with best scores

The results with lowest rouge scores are displayed below with the actual and predicted captions:

Actual Caption : A man at a casino hugging two colorfully dressed show girls .
Predicted Caption : a group of people gather for a picture .
Rouge Score : 0.111

Actual Caption : Three people are facing the mountains .
Predicted Caption : a man and a woman walk down a hill .
Rouge Score : 0.0

Fig. 10. Results with worst scores

Most frequent word distributions : The top 50 most frequent words in the test dataset and in predicted captions are displayed below. This distribution can give us insights onto what does our model predicts for each image more frequently compared to the actual information which was expected in the images

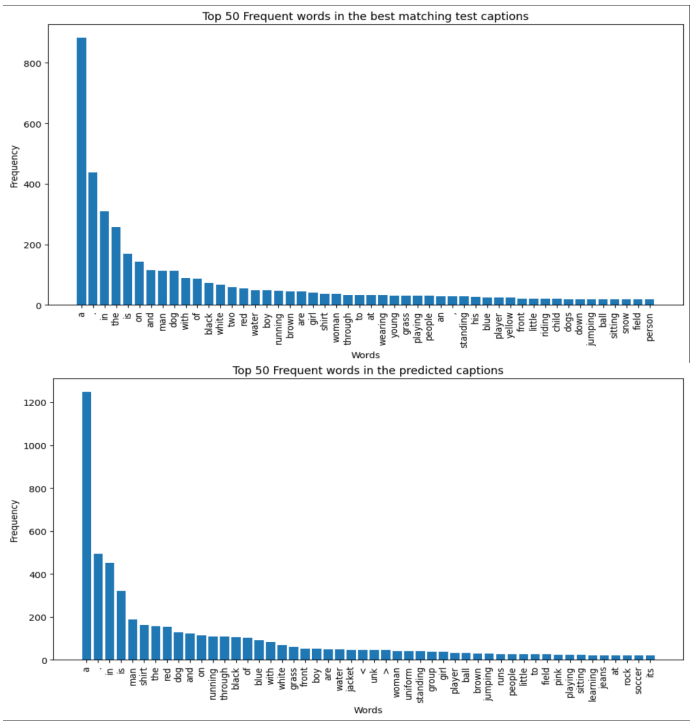

Fig. 11. Top 50 Most Frequent Words : Test vs. Predicted Captions

Assessment : The model is able to identify the objects well enough but the context is not captured essentially well. Moreover, the top 50 most frequent words show that the model has learned a few words extremely well and associates most of the images with those words only. Also, it performs well on images with dogs but does not seem to perform well on images with people. There is some overfitting to the training data and we may need to focus on some attention based mechanism to help the model capture the context or action in the image.

B. CNN-RNN with Attention Model Results

The model is trained for 100 epoches and each word is represented by a 300 size vector and the maximum caption length is set to 40. The loss function is Cross Entropy and the optimizer is Adam with a learning rate of 0.0003.

The model took 8hrs for training on the Nvidia GeForce RTX 3060 GPU.

Loss Results and Assessment:

Training loss and Validation loss:

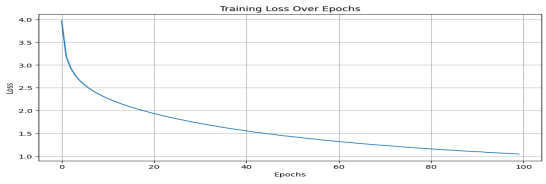

Fig. 12. Training Loss

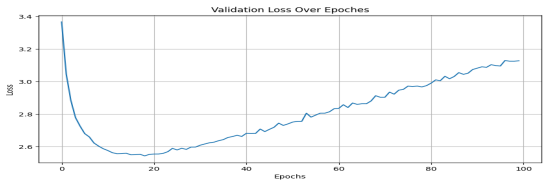

Fig. 13. Validation Loss

The training loss figure demonstrates that the initial training loss of 4.08 at epoch 1 declined to 1.02 by epoch 100, indicating significant learning progress. However, signs of overfitting were observed from epoch 19 onwards, as evidenced by the increasing trend in validation loss. Notably, the model parameters saved at epoch 19 did not effectively learn sentence formation, resulting in elevated loss when tested. In contrast, the model saved at epoch 100 demonstrated advanced sentence construction capabilities and effectively computed attention weights. This allowed for the generation of captions that captured the essence of the image and highlighted key features, a capability not observed in the earlier model iteration during testing.

Testing Loss: The average testing loss on the model trained for 100 epoch is 3.09.

Rouge Sores Results:

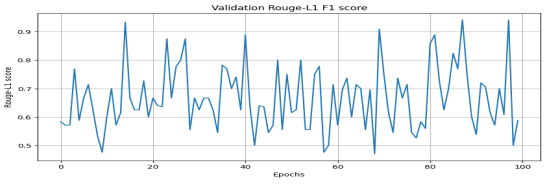

Fig. 14. Validation Rouge-L1 F-score

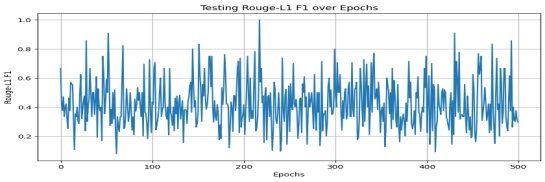

Fig. 15. Testing Rouge-L1 F-score

The validation process involved recording the maximum ROUGE score for each epoch across the entire validation dataset, as depicted in the graph below. In the test phase, the model achieved an average ROUGE score of 0.41, assessed over 500 images. Notably, the top 10 ROUGE scores consistently exceeded the 0.8 threshold across various iterations, indicating the model's robust ability to extract pertinent image features and accurately associate them with well-formed English sentences. This performance demonstrates the effectiveness of the model in capturing and translating visual content into coherent textual descriptions.

Example Results:

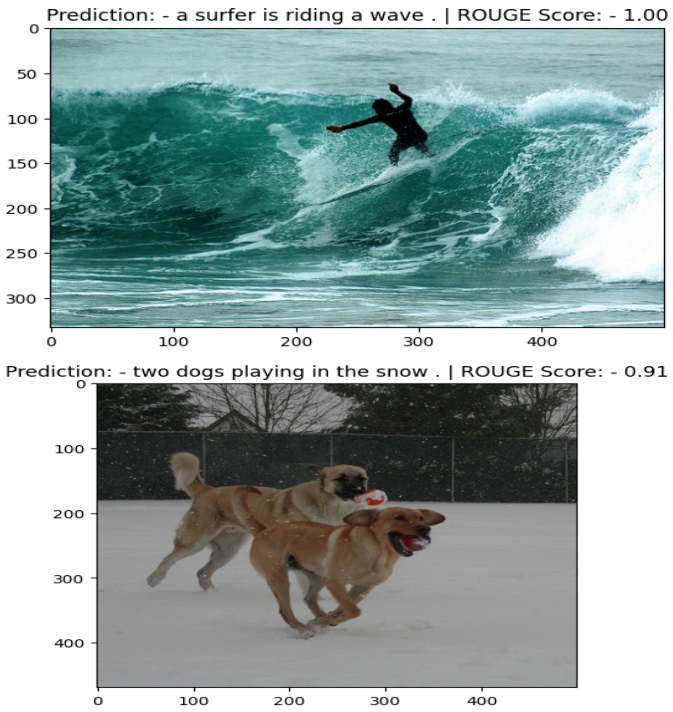

Fig. 16. Results with best scores

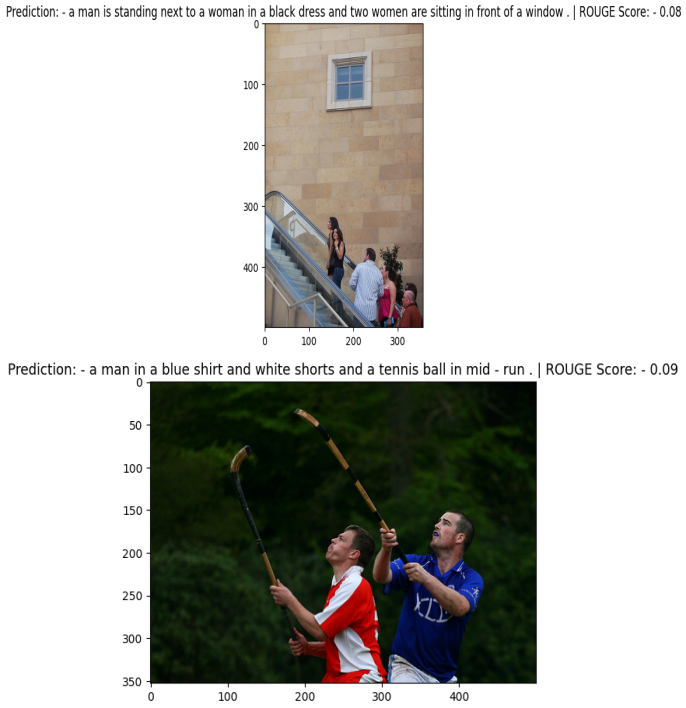

Fig. 17. Results with worst scores

Most Frequent Words The top 50 most frequent words in the test dataset and predicted captions are displayed below:

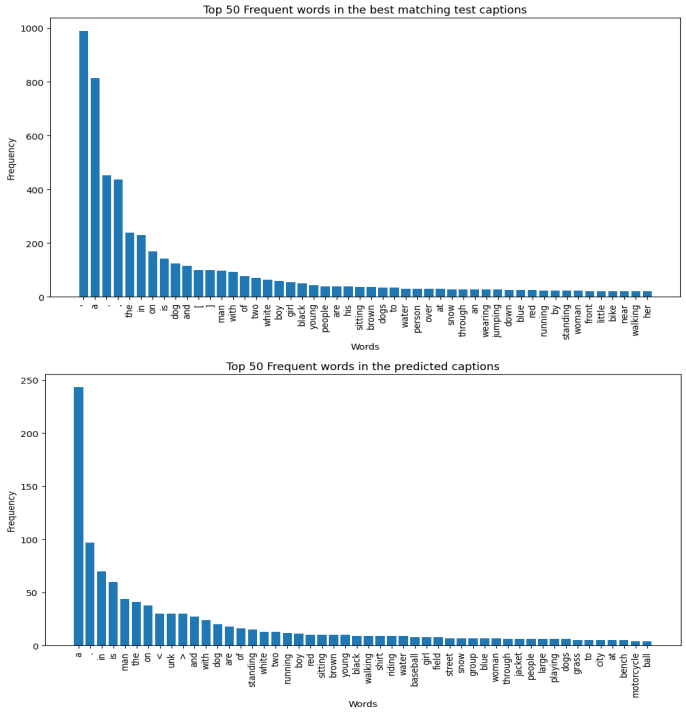

Fig. 18. Top 50 Most Frequent Words: Test vs Prediction

C. ViT-GPT2 Model Results

As a result of lack of computational resource this model is only trained for 3 epochs which took around 8 hours on Kaggle using the P100 GPU. The loss function is cross entropy and the optimizer is Adam with a learning rate of 0.00005.

Loss Results and Assessment:

Epoch	Training Loss	Validation Loss	Rougel Precision	Rougel Recall	Rougel Fmeasure
1	2.896700	2.785194	0.065217	0.300000	0.107143
2	2.657400	2.728507	0.071429	0.300000	0.115385
3	2.495900	2.726482	0.065217	0.300000	0.107143

Fig. 19. Testing and Validation Loss

From the above figure it can be observed that the losses reduce and the rouge score improved with the number of epochs. However, the small number of epochs here cannot give us a full picture. But we decided to test this model on a test dataset.

Figure 20 shows the Rouge-L1 score on the test dataset. The average Rouge score for testing is 0.08 which is really low and can be attributed to the low number of epochs.

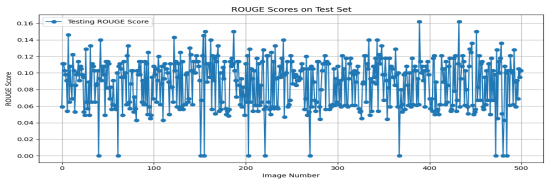

Fig. 20. Testing Rouge-L1 F-score

Figures 21 and 22 shows the top 50 most frequent words in test dataset and in the predicted captions respectively.

Fig. 21. Top-50 most frequent words for test captions

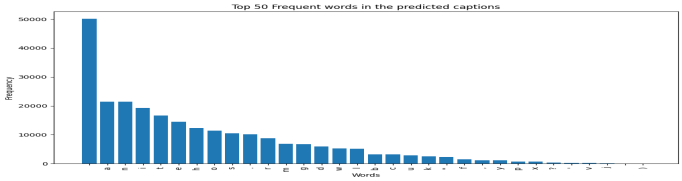

Fig. 22. Top-50 most frequent words from predicted captions

Following the results and analysis of this model, it was clear to us that the results could be a lot better with better training or model. Hence, we decided to try a different model Link to the model (CLICK)that has been fine tuned for the flickr8k dataset.

Results of Second Model:

The second model perform relatively better than the previous one with an average Rouge score of 0.46 over the testing data. Figure 23 shows the plot for rouge score over test images. Meanwhile, figures 24 and 25 shows the top 50 most frequent words in test dataset and in the predicted captions respectively for the second model.

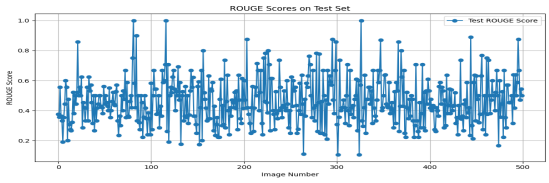

Fig. 23. Testing Rouge-L1 F-score

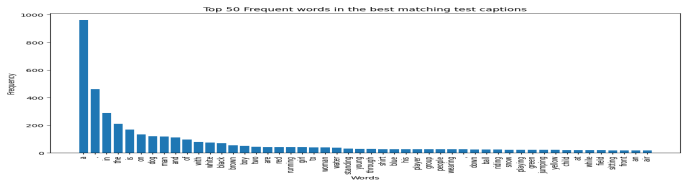

Fig. 24. Top-50 most frequent words for test captions

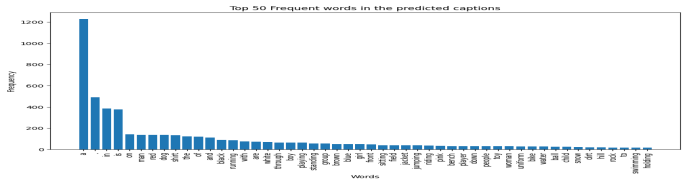

Fig. 25. Top-50 most frequent words from predicted captions

Example Results:

Following images show the top-3 example images according to the rouge scores.

Fig. 26. Top-1 result

Fig. 27. Top-2 result

Fig. 28. Top-3 result

VII. Discussion and Summary

In the evaluation of the three image captioning models—CNN-RNN, CNN-RNN with attention, and ViT-GPT2—conducted on the Flicker8k dataset, several intriguing results emerged. The CNN-RNN model exhibited a training loss of 1.67 and a test loss of 4.82, indicating a moderate gap that might suggest a slight overfitting. On the other hand, the CNN-RNN with attention demonstrated improved performance with a training loss of 1.02 and a test loss of 3.09. Most notably, the pre-trained ViT-GPT2 model achieved the best results with test loss of 2.78, suggesting superior learning and generalization capabilities likely attributed to its advanced architecture. Regarding the average ROUGE-L F1 scores across the test data, the CNN-RNN, CNN-RNN with attention, and ViT-GPT2 scored 0.336, 0.41, and 0.46 respectively. These scores illustrate a clear trend: the integration of attention mechanisms and the sophisticated transformer model architecture substantially boost the linguistic relevance and coherence of the generated captions, with ViT-GPT2 leading in capturing complex dependencies and contextual nuances in the dataset.

Furthermore, analysis of the top and worst-performing images with respect to their ROUGE scores highlighted the strengths of the attention mechanism. The models equipped with attention, particularly the CNN-RNN with attention and the ViT-GPT2, showcased their capability to identify specific features in the images while generating words for the captions.

Acknowledgment

We extend our deepest gratitude to Prof. Bruce Maxwell for his invaluable guidance and expertise throughout the duration of this project. His generosity in sharing his profound knowledge and his meticulous attention to detail have been instrumental in the success of our work. Without his unwavering support and dedication, this project would not have achieved its goals. Thank you, Prof. Maxwell, for your mentorship and belief in our capabilities.

References

[1] D. Bahdanau et al., Neural Machine Translation by Jointly Learning to Align and Translate, https://doi.org/10.48550/arXiv.1409.0473.
[2] Suresh, K.R., Jarapala, A. & Sudeep, P.V. Image Captioning Encoder–Decoder Models Using CNN-RNN Architectures: A Comparative Study. Circuits Syst Signal Process 41, 5719–5742 (2022), https://doi.org/10.1007/s00034-022-02050-2
[3] Image captioning Tutorial Pytorch : Link to Video (CLICK)
[4] Dataset Creation approach for Flickr8k : Link to Video (CLICK)
[5] Attention Is All You Need, https://doi.org/10.48550/arXiv.1706.03762.
[6] A Frustratingly Simple Approach for End-to-End Image Captioning, https://doi.org/10.48550/arXiv.2201.12723.