Descriptive Image Captioning using Vision and Language Models

Final Project Report

Course: Deep Learning and Multimodal Systems - Fall 2023

Group Members: Awais Naeem, Christian Martin

Instructor: Dr. Abhijit Mishra

School of Information

University of Texas at Austin

Table of Contents

Introduction

Image captions are used to communicate additional information about the image that is not present in the image itself, or the information that is tangentially related to the image. Image captions are useful for a variety of reasons across different domains. On social media websites, user uploaded images can be captioned for accessibility needs. On e-commerce sites, image captions can enhance product listings. In the medical field, generating descriptions of X-ray images can be used to assist radiologists. For other businesses such as travel, real estate and education, captioning images can be used to drive growth in the business. Moreover, captions for any image can be used to trace the origins of any image to any appropriate publisher for intellectual property rights. Thus, a sophisticated image captioning system which can understand the context/relation of different objects in an image and write a descriptive caption about it can prove beneficial for many real-world use cases.

The recent developments in the space of large language models (LLM) have made it possible to perform different operations on the text data including sequence classification, text summarization and even text generation. In a similar context, many large language models have been pretrained and fine-tuned on image-text pairs to generate the captions for any new image [1] [2] [3]. However, the problem with current state-of-the-art image captioning models [1] is that they generate short (non-descriptive) captions. For example, a caption of Figure 1 is generated as "*a woman sitting on the beach with her dog*".

Figure 1: An image of a woman and a dog

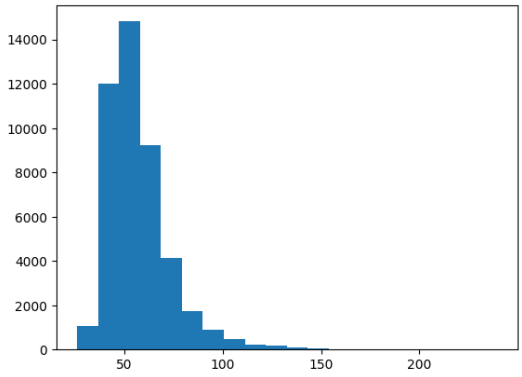
For accessibility needs, a more descriptive caption for the same image (Figure 1) will make more sense such as, "*a smiling woman is enjoying her time on the beach with her dog with the sea waves originating*

by their side. The golden colored dog has a colorful strap on and seems to be playing within the company of the girl."

To generate descriptive captions, we cannot rely on the state-of-the-art methods [1-3] and need to develop a custom image captioning or text generation large language model by fine-tuning the base image captioning models on a dataset containing image-text pairs with descriptive captions.

Data Description

As a reference point, we looked at the nocaps dataset[16], as it's one of the datasets the Salesforce BLIP[18] model was trained on. This dataset was not used for training the models and was used only for researching what was typical length of captions. The histograms of the dataset showed that most captions in the dataset are about 45-70 characters in length, and all being in the range of 30-100 characters. Since these captions tend to be very terse, we set a goal of generating captions 2-4x.



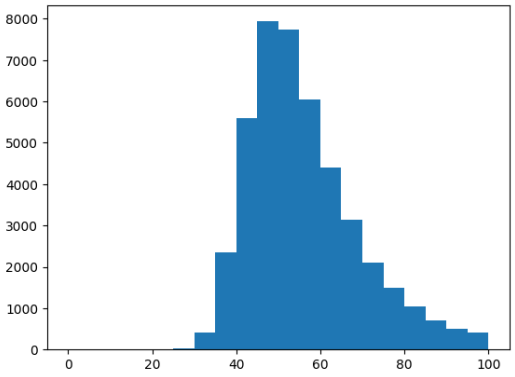


Figure 2 (left) & Figure 3 (right). Length of descriptions in the nocaps dataset, in characters. Figure 3 has an upper bound of 100 characters.

The primary data we're using comes from the Smithsonian Institute's Open Access dataset [21], which is available from an Amazon S3 bucket [22]. The dataset contains terabytes of data, including metadata, high-res images, and 3D models of the works in their collection (copyright and other law permitting) for all Smithsonian museums and research centers. The subset of this data we're going to be looking at comes from the Cooper Hewitt, one of the museums in the Smithsonian. The Cooper Hewitt was chosen because they've made a concerted effort to improve the descriptions of the works in their collection, including publishing their Guidelines for Image Description [23].

The metadata records were filtered from a few criteria:

1. From the Cooper Hewitt collections
2. Has a description field with a length between 120 and 400 characters (this filters the descriptions to a length about 2-4x of nocaps)
3. Does not have a "/" in the description - this removes descriptions that are a "tombstone," which is a concatenation of multiple things, including the description, artist name and bio, copyright, credit line (e.g. donor/funder), and other information that is not useful for our purposes.
4. Has a JPEG image that's licensed as public domain and has dimensions of at least 1000x1000px. This is meant to prevent copyright issues in the dataset, avoid TIFFs (due to their large size) and

to ensure that the images we're using are reference images (as opposed to thumbnails). If there were multiple JPEG images, we used the first one given.

5. There were a handful of images where the metadata indicated an image should be available, but the file did not exist – these were filtered out by comparing the records with a list of files in the S3 bucket.

Setting these filters allowed us to massively reduce the size of data needed to be downloaded; the Cooper Hewitt has ~770GB of JPEG images in the dataset alone, which would take a long time to download.

Since we needed such a selective part of the S3 bucket, we used a custom downloader using the Python multiprocessing library and the AWS boto3 client library[12][13][14][15]. The downloader took in the list of images to download, created multiple processes with their own boto3 instance, split the list of images across the processes, checked the file was not already downloaded, downloaded any images needed, and then piped the image into the Pillow image library to downsize the image before saving it to disk. These checks allowed us to not re-download images downloaded during testing (and when the downloader needed to be restarted to remove the Pillow file size limit), as well as reduce the data size by only saving the downsized image. All of these resulted in reducing the images to about 2.4GB, a significant reduction from the original ~220GB that was downloaded.

Methodology

To develop a model which can generate long and descriptive captions, we have followed two approaches to fine-tune the large language models. The first approach takes the image and descriptive caption pairs from the processed dataset, and then tries to fine-tune an image captioning large language model [4]. The second approach aims to generate long captions by fine-tuning a text generation large language model using a combination of visual and language models to generate synthetic data for training/prediction [5].

1. Fine-Tuning Image Captioning LLM (Direct Approach)

The first approach we took to creating a model involved fine-tuning the existing Salesforce BLIP model [18]. The Salesforce BLIP model was chosen because it supports image-to-text generation, and we could get the model and processor to perform basic inference.

Despite being able to get it to load, we encountered a lot of issues attempting to fine-tune the BLIP model. The biggest hurdle is that the model takes up a lot of memory during training, and even 16GB of VRAM was not enough to train the model. We attempted to use the load_in_8bit option [20] to reduce the memory footprint, as well as training on the CPU using Intel's IPEX library [19] for AVX2 hardware acceleration, but this was not enough to let the HuggingFace Trainer complete an epoch.

At the suggestion of the professor, we attempted to use PEFT, and from reading the HuggingFace documentation, e.g. [17], the LoRa configuration was used as it was referenced for image-related tasks. The combination of load_in_8bit, PEFT, and LoRa was able to reduce the memory required for training to where it could run.

We made a few attempts at tweaking the training, such as switching from a Word Error Rate metric to the BLEU metric and removing stop words after loading the training dataset, however, despite helping,

these were not able to bring the model to a useful state and it was time-prohibitive to continue working on.

With respect to being time-prohibitive, notably, it took about 4.5 hours for each run of the training code, and evaluating using the test dataset took about 2.5 hours. Additionally, the inference times for the trained model took about 10x longer than the base model before training.

2. LLM Text Generation using Visual/Language Models (Indirect Approach)

Rather than taking a straight-forward approach of taking in the images and their corresponding captions to fine-tune an image captioning large language model, this approach takes an indirect approach towards the generation of the long captions.

Apart from just using a text generation LLM [6], this approach uses an object detector [7], a base image captioning model [1] and OpenAI API [8]. The idea is to generate the object list from the images and short captions from the long captions in the dataset. Then an input containing both the object list and short captions (<object list, short captions>) along with the long captions as the output can be used to train a text generation LLM [6].

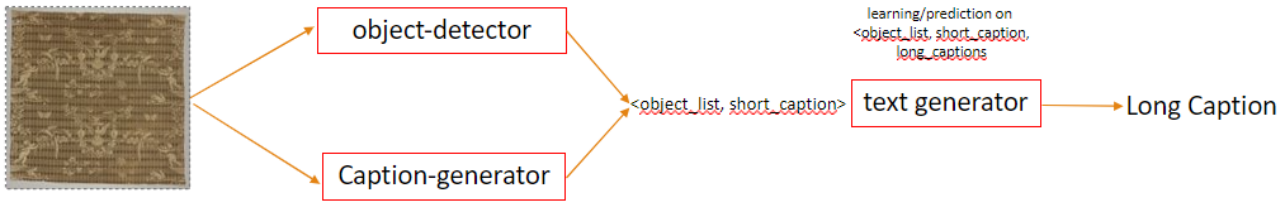


Figure 4: A workflow diagram of LLM Text Generation Approach

During the training, long captions are firstly tokenized. Then the punctuation characters and stop words are removed from the tokenized captions. Finally, all the tokenized words are lemmatized, and we extract a unique set of words which are termed as the object list. The long captions are passed through OpenAI API [8] to generate their summary which are then used as the short captions. Figure 5 depicts a workflow diagram of the entire process. The prompt which was used to get the response from the OpenAI API is given in Appendix A.

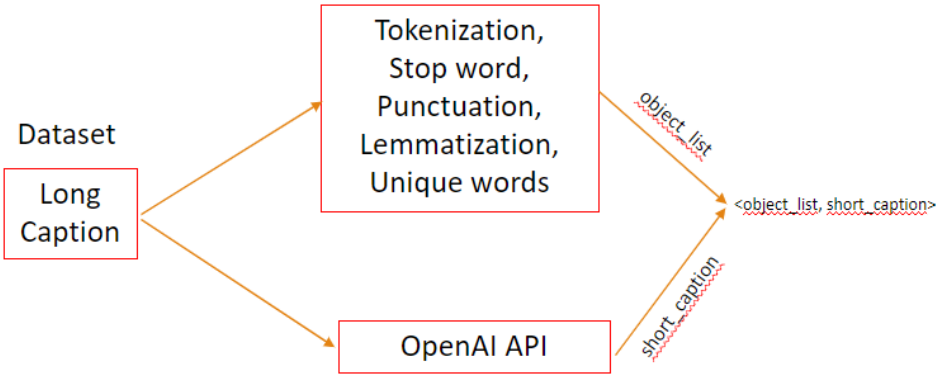


Figure 5: Extracting object list and short captions for training purposes

After getting the object list and short captions, we feed <object list; short caption> as input and long caption as output into the text generation LLM [6] to fine-tune the model. For fine tuning, we used 2000 training samples because OpenAI API was slow to respond to the requests of generating summary of the long captions. Appendix B describes an example of how the prompt of LLM is formed for training purposes for the text generation LLM [6].

For the prediction, an image is fed into the object detector [7] to get the object list. On the other hand, the same image is passed through an image captioning model [1] to generate the short caption for the image. Then a combined input of <object list, short caption> is passed through the fine-tuned text generation LLM [6] to get the long caption as the prediction. Appendix C enlists a prediction example using the fine-tuned LLM as per the prediction workflow shown in Figure 6.

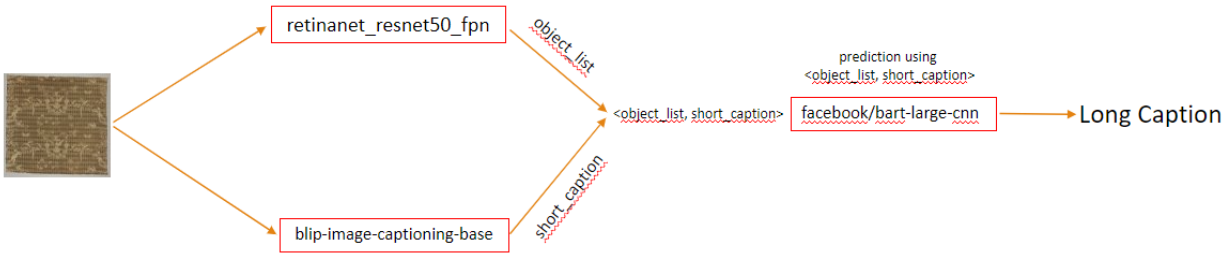


Figure 6: Prediction workflow using fine-tuned Text Generation LLM

For illustration purposes, some examples of the predicted captions along with image and actual captions are listed below:

Actual Caption: Decorated in red and gold, in outline of conventionalized petals. In center of saucer and inside tea bowl is a butterfly.
Predicted Caption: Small cup and saucer on dining table. Vase, wine glass, vase, bowl, spoon, and bottle of wine. Sandwich on top of cup. Cake in cup and on saucer with a spoon and a vase at the bottom of the bottle.

Actual Caption: Horizontal rectangle. Staircase in palace with portico at top of stairs. Some details in graphite only, possibly unfinished.
Predicted Caption: Drawing of a staircase in a building. Traffic light, bench, and potted plant. In the drawing, a person is on a road light, and a building has a staircase with a bench and traffic light above it, with a building with a person on it.

Actual Caption: Wide valance with bands of solid blue and ochre, pink and beige weft ikat. Same fabric used as part of the facing on the back of the valance. valance is bound with plain weave tape.
Predicted Caption: Blue and brown curtain with a white background. Kitchen bench, dining table, sink, refrigerator, freezer, toaster, coffee table, bed, bedside table, sofa, chair, laptop, coffee cup, fork, knife, suitcase, toothbrush, umbrella, vase, frisbee, cat, pizza, cake, books, toys, surfboard, car, boat, train, and elephant. Stop sign with person and cat tied to it.

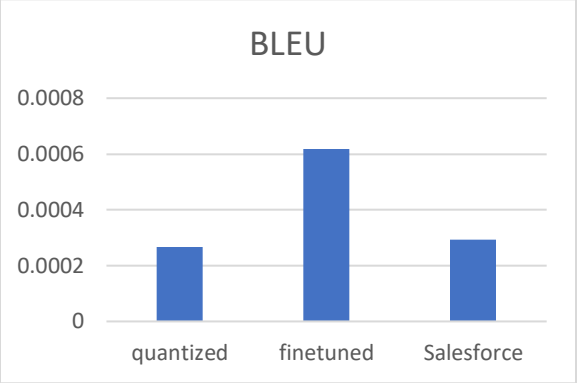
Results

As the models are being fine-tuned to produce the long captions, the evaluation metrics underlying word overlap do not qualify as good measures to evaluate the performance of the models. Thus, we have evaluated the models on the similarity measures between the predicted long caption and the actual long caption.

Fine-Tuning Approach

The fine-tuned models were performing ok. As a note, we also charted the quantized BLIP model in addition to the as-is (not quantized) BLIP model to make sure that the performance didn't drop significantly between the original and quantized model (since our fine-tuning was based on the quantized model); thankfully, it didn't.

The v2 model performed well on the BLEU and word error rate (WER) metrics, as seen in Figure 6 & Figure 7.



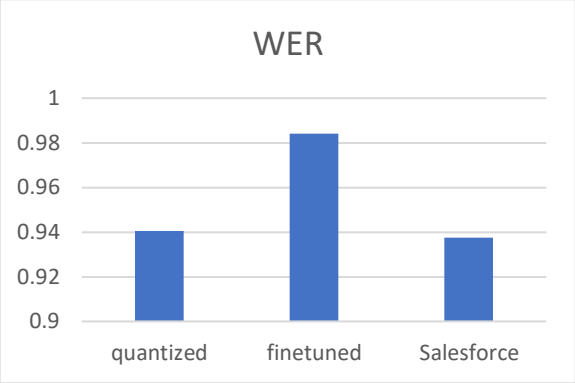


Figure 6 (left). BLEU metric scores. Figure 7 (right). Word Error Rate (WER) metric scores.

The model also had a much better BLEU length ratio, as seen in Figure 8.

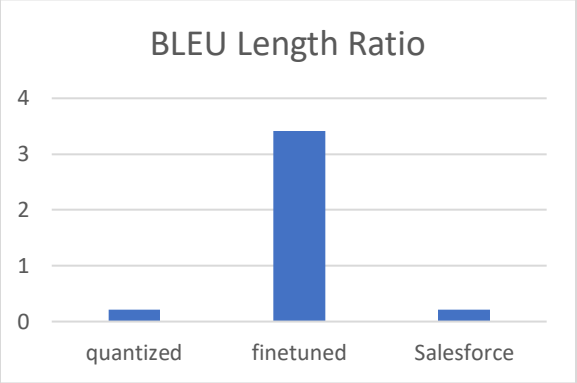
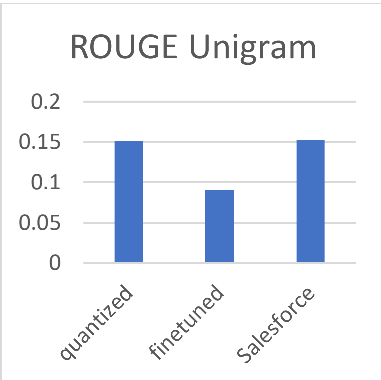
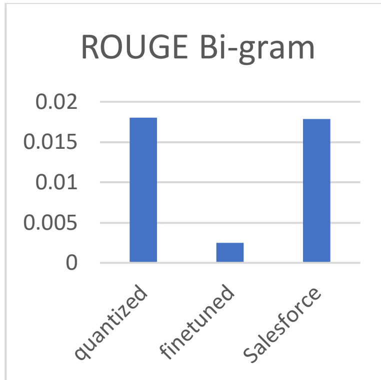


Figure 8. BLEU Length Ratio metrics.

However, when looking deeper, these quickly fell apart. First, the BLEU length ratio is compared to the reference text, meaning that instead of generating captions 2-4x the original BLIP model's length, we were instead generating captions 2-4x our dataset's length. Put differently, instead of generating captions 2-4x as long, we ended up with captions 16x as long.

And while BLEU sounds good on paper, in part due to its brevity penalty, the BLEU and WER metrics also encouraged the model to repeatedly generate common words, such as stop words, due to their frequent appearances. This is reflected in the ROUGE scores, as seen in Figure 9 and Figure 10, which incentivize making sure all words appear, as well as the METEOR score, which balances BLEU and ROUGE, as seen in Figure 11.





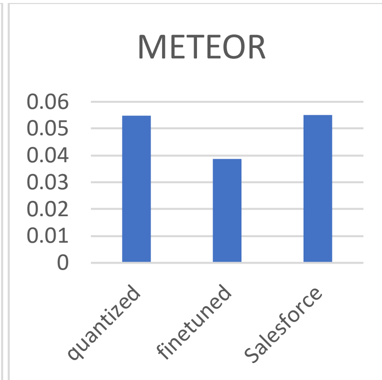


Figure 9 (left). ROUGE Unigram scores. Figure 10 (center). ROUGE Bi-gram scores. Figure 11 (right). METEOR scores.

Likewise, this is also visible in the captions generated. For example, here is a caption generated by the v1 model for its corresponding image:

a and a - in a - large, large, large, large, large, large, large, large, large, large, large, large, large, large, large, large, large, large, multi, multi, natural, la, la, la, la, la, la, la, la, the, la, the, the, la, la, the, the, the, the, the, the, the, the, the, the, the,

And here is a caption generated by the v2 model, with stopwords removed during training:

a - - -, blue, blue, blue, and, blue, blue, blue, blue,,, ',,, ' ,'' ' ,

To test if the model might be over-fitting to the metrics, we performed an experiment where we trained the model using the ROUGE metric on a very small amount of the dataset, using 1000 records instead of the whole ~20k.

The results were not as blatantly bad, but still performed poorly; for example, the model scored a zero on ROUGE bi-grams and on the BLEU score, and the ROUGE unigram score, as seen in Figure 13, is the lowest of all the models.

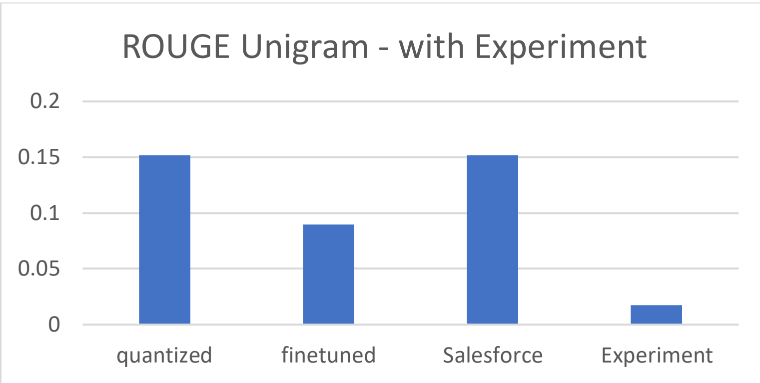


Figure 13. ROUGE Unigram scores, including over-fitting experiment results.

And this is visible in the captions themselves; here's an example of a generated caption:

a, book bottom bottom bottom bottom bottom bottom bottom bottom bottom wall wall wall wall wall wall wall wall wall wall wall wall wall wall wall personrarararararararararararararararararararararararararararararar ararararara

If you de-duplicate words and removing the errata at the end, the caption amounts to *"a book bottom wall person"* – which is neither more desirable than the original model's *"a book with drawings of various objects"* nor does the image contain a person.

While this is as far as this approach was able to go, there is room for more exploration on fine-tuning a model. For example, evaluating the Microsoft GIT model or other optimization techniques. However, with the long training and inference times, this work is expected to be slow and computationally expensive.

LLM Text Generation Approach

The test data consisted of around 100 samples for long captions and images each. For the testing purposes, the object list for each image was generated using the object detector model [7] using PyTorch vision libraries. Short captions were generated for each image using image captioning LLM [1] via HuggingFace model checkpoints. A small sample size was selected because of the limited processing capacity in context of RAM on Google Colab Pro. The collected object list and short captions were then passed through the fine-tuned text generation model [6] to predict the long captions for each image in the test data.

Firstly, the word overlap based metrics were calculated between the predicted and the actual captions i.e., Rouge 1 (1-gram), Rouge 2 (2-gram) and Rouge longest common sequence (Rouge L) [9]. From Table 1, it is evident that the model performed well enough for 1-gram (Rouge 1) overlaps as well as for the longest common sequence (Rouge L) considering that it was fine-tuned for only 2000 training samples which were synthetically generated using text processing and summarization. The overall deficient

performance on the 2-gram overlap can be explained by the fact that mode was fine-tuned to generate the long captions and did not care about the order of the words.

		Precision	Recall	F-Measure
Rouge 1	Low	0.12	0.12	0.13
	Mid	0.009	0.011	0.010
	High	0.15	0.19	0.16
Rouge 2	Low	0.009	0.011	0.010
	Mid	0.012	0.016	0.013
	High	0.016	0.021	0.017
Rouge L	Low	0.098	0.132	0.109
	Mid	0.108	0.144	0.118
	High	0.117	0.154	0.126

Table 1: Rouge scores of Predicted Long captions vs actual long captions

Since the word overlap based metrics only capture the ordering of the words in the captions, we tried to find the sentence level embeddings for the captions to evaluate the contextual similarity between the predicted and the actual captions. Firstly, we extracted the sentence level embeddings for both the actual and predicted captions using transformer-based BERT model [10]. Then a cosine similarity score was determined for each test/predicted caption and averaged to get an overall similarity score of **0.7098**. Secondly, we extracted the word embeddings of each caption using the word2vec model trained on google news data [11] and then took the mean for word vectors to get the sentence level embeddings for each predicted/actual long caption. The average cosine similarity for word2vec based sentence vectors was found to be around **0.6026**. These similarity scores (especially for the BERT model) depict that the generated captions are contextually alike the actual ones.

Moreover, the caption lengths generated by the model were compared against those generated by the state-of-the-art and target captions. On average, actual captions had a word count of **35.84** and state-of-the-art [1] generated captions with an average word count of **10** whereas the word count of the fine-tuned text generation LLM [6] was found to be **51.38**. This showed that the model learnt to produce captions significantly longer than the state-of-the-art and the captions it was trained on.

Conclusion

LLM text generation using visual/language models can generate long captions, but random words appear which are not representative of the image itself. The primary reason for this behavior is the fine-tuning of the model on a smaller number of training samples i.e., 2000. Moreover, the object list is synthetically generated using text pre-processing techniques which is limited in extracting all the objects present in the image.

Further improvements can be made by fine-tuning the model on a complete set of training samples (18000) using a resource intensive GPU. Some methods of fine-tuning may also be computationally problematic, as seen by the inference times increasing 10x while using PEFT+LoRA, which will require additional time and care to pursue.

References

[1] “BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation”, 2022, Li, Junnan and Li, Dongxu and Xiong, Caiming and Hoi, Steven, Doi: 10.48550/ARXIV.2201.12086

[2] https://ankur3107.github.io/blogs/the-illustrated-image-captioning-using-transformers/, Ankur Kumar, doi:10.57967/hf/0222

[3] “FuseCap: Leveraging Large Language Models for Enriched Fused Image Captions”, 2023, Noam Rotstein, David Bensaid, Shaked Brody, Roy Ganz, Ron Kimmel, Doi: 10.48550/ARXIV.2305.17718

[4] “Fine Tuning an Image Captioning Model”, HuggingFace, https://huggingface.co/docs/transformers/main/tasks/image_captioning

[5] “Fine-Tuning the BART Large Model for Text Summarization”, 2021, Francois St-Amant, Toward Data Science, https://towardsdatascience.com/fine-tuning-the-bart-large-model-for-text-summarization-3c69e4c04582

[6] “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”, 2019, Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer, doi: 10.48550/ARXIV.1901.13461

[7] “Focal Loss for Dense Object Detection (Retina net Resnet50 FPN)”, 2018, Tsung Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, Piotr Doll, doi: 10.48550/ARXIV.1708.02002

[8] “ChatGPT: Language Model.”, 2021, OpenAI, GPT-3.5. Retrieved from https://openai.com/chatGPT.

[9] “ROUGE: A Package for Automatic Evaluation of Summaries”, 2004, Lin, Chin-Yew, Text Summarization Branches Out, Association for Computational Linguistics, pages 74-81

[10] “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, 2019, Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, doi: 10.48550/ARXIV.1801.04805

[11] “Efficient Estimation of Word Representations in Vector Space”, 2013, Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Doi: 10.48550/ARXIV.1301.3781

[12] “File transfer configuration” https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3.html

[13] “Multiprocessing vs Threading Python” https://stackoverflow.com/questions/3044580/multiprocessing-vs-threading-python

[14] “Downloading multiple S3 objects in parallel in Python” https://stackoverflow.com/questions/48091874/downloading-multiple-s3-objects-in-parallel-in-python

[15] “How do I download a JPG file from Amazon S3 into memory using Python?” https://stackoverflow.com/questions/68847323/how-do-i-download-a-jpg-file-from-amazon-s3-into-memory-using-python

[16] “nocaps: novel object captioning at scale”, 2019, Agrawal et al, 2019 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, doi: 10.1109/iccv.2019.00904

[17] “Image classification using LoRA” https://huggingface.co/docs/peft/task_guides/image_classification_lora

[18] “BLIP” https://huggingface.co/docs/transformers/main/model_doc/blip

[19] “Efficient Training on CPU” https://huggingface.co/docs/transformers/perf_train_cpu

[20] “Quantize 🤗 Transformers models – bitsandbytes Integration” https://huggingface.co/docs/transformers/main_classes/quantization#bitsandbytes-integration

[21] “Smithsonian Open Access” https://www.si.edu/openaccess

[22] “Smithsonian Open Access – Registry of Open Data on AWS” https://registry.opendata.aws/smithsonian-open-access/

[23] “Cooper Hewitt Guidelines for Image Description” https://www.cooperhewitt.org/cooper-hewitt-guidelines-for-image-description/

Appendices

Appendix A: ChatGPT Prompt

messages= [
{"role": "system", "content": "You are a text summarization assistant, skilled in summarizing long image captions to short captions"},
{"role": "user", "content": "Summarize the following caption to only 10 words: " + long caption
}]

Appendix B: Text Generation LLM Prompt

Long Caption: *Landscape with two horsemen enter the composition from the left foreground, one with a falcon standing on his left hand. Falcons hunt birds in the background. Framing line at the bottom.*
Object List: *['enter', 'bird', 'bottom', 'composition', 'framing', 'falcon', 'hand', 'one', 'background', 'foreground', 'left', 'line', 'standing', 'hunt', 'two', 'landscape', 'horseman']*
Short Caption: *Two horsemen, falcon, hunting birds, framed line at bottom.*
LLM Prompt: *objects: enter, bird, bottom, composition, framing, falcon, hand, one, background, foreground, left, line, standing, hunt, two, landscape, horseman; caption: Two horsemen, falcon, hunting birds, framed line at bottom.*

Appendix C: Predicting Long caption using Text Generation LLM

Object List: *['cat', 'laptop', 'chair', 'train', 'sink', 'suitcase', 'bed', 'bench', 'couch', 'person', 'boat', 'tv', 'potted plant', 'dining table']*
Short Caption: *textile fragment with floral design*
LLM Prompt: *objects: cat, laptop, chair, train, sink, suitcase, bed, bench, couch, person, boat, tv, potted plant, dining table; caption: textile fragment with floral design*
Predicted Long Caption: Textile fragment with a potted plant, couch, chair, sofa, bed, laptop, chair and suitcase. In the sink are people on boats, in the living room a couch, on the dining table a chair, on a bench, a couch and chair, and on the sofa are a cat, a bed, and a suitcase.

Code

The source code for fine-tuning and prediction results both the approaches can be found in the following open-source GitHub repositories:

https://github.com/ct-martin/dlmm-final-project

https://github.com/embedded-robotics/deep-learning-multimodal-systems-final-project