OBSS AI Image Captioning Challenge

Participant Report

---

Name: Kubilay Karaçar
Email: kubilay.karacar@gmail.com
Phone Number: 546 520 3232
Kaggle Username: kubilaiswf

---

## 1. Tech Stack

- Programming Language: Python 3.11 (as per Google Colab environment, you can verify with !python --version in your notebook)

- Core Libraries:
  - PyTorch (e.g., 2.3.0+cu121 - verify with torch.__version__)
  - HuggingFace Transformers (e.g., 4.41.2 - verify with transformers.__version__)
  - Pillow (PIL) (for image loading and processing)
  - Pandas (for CSV file manipulation)
  - Tqdm (for progress bars)

- Models and Techniques:
  - Model: Salesforce BLIP-2 OPT 2.7B (Pre-trained image captioning model)
  - Processor: Blip2Processor from HuggingFace Transformers
  - Quantization: 8-bit quantization using BitsAndBytesConfig from the transformers library.
  - Precision: Inference performed using torch.float16.

- Development Environment:
  - Google Colab (Free Tier, utilizing NVIDIA T4 GPU with 15GB VRAM)

---

## 2. Summary

For this image captioning challenge, my approach centered on leveraging a powerful pre-trained multimodal model, specifically the Salesforce BLIP-2 OPT 2.7B. Given the constraints and the desire for a robust baseline, I opted to use this model in a zero-shot inference mode, meaning I didn't perform any additional fine-tuning on the provided training dataset. To ensure the solution could run efficiently within the Google Colab Free Tier environment (NVIDIA T4 GPU), I implemented 8-bit

quantization and utilized float16 precision during inference. This helped manage memory usage and processing time. For each image in the test set, I generated captions by providing the model with a consistent prompt, "a photo of", and then processed the outputs to create the final submission file.

---

## 3. Approach

My core strategy for this competition was to utilize the capabilities of a state-of-the-art, pre-trained vision-language model. I selected the Salesforce BLIP-2 OPT 2.7Bmodel due to its strong performance on general image captioning tasks and its availability through the HuggingFace Transformers library.

Model Selection and Setup:

- Environment and Optimization: A key consideration was adhering to the competition's execution constraints, particularly the Google Colab Free Tier (NVIDIA T4 GPU with 15GB VRAM). To make the 2.7B parameter model manageable in this environment, I employed two main optimization techniques:
    - 8-bit Quantization: I used the BitsAndBytesConfig from the transformers library to load the model in 8-bit precision. This significantly reduces the model's memory footprint without a drastic loss in performance for inference tasks.
    - Float16 Precision: During the inference stage, calculations were performed using torch.float16. This further aids in reducing VRAM usage and can speed up computations on compatible hardware like the T4 GPU.
    - The model was loaded with device_map="auto", allowing Transformers to optimally distribute the model layers across the available GPU and CPU memory.

Data Processing and Caption Generation Pipeline:

1. Image Loading: For each image ID in the test.csv, the corresponding .jpgimage was loaded from the obss_data/test/test/ directory using the Pillow (PIL) library. Each image was converted to the RGB format to ensure consistency.
2. Inference: The preprocessed image and tokenized prompt were packaged into a tensor and moved to the CUDA device (.to("cuda", torch.float16)). The model.generate() method was then called. To control the length of the generated captions and prevent overly verbose or run-on descriptions, I set max_new_tokens=50. This parameter limits the number of tokens the model can generate after processing the initial prompt.

3. Post-processing: The output from model.generate() consists of token IDs. These IDs were converted back into human-readable text using processor.batch_decode().
   The skip_special_tokens=True argument was used to ensure that special tokens (like end-of-sequence tokens) were not included in the final caption. The resulting caption often included a newline character (\n) at the end, which was kept as per the observed format in the submission.csv example.

4. Submission File Generation: The generated caption for each image_id was stored, and finally, a Pandas DataFrame was created and saved as submission.csv in the required format (image_id,caption).

---

## 4. Sample Outputs

Below are some sample outputs from my pipeline, demonstrating a range of prediction qualities. The captions are presented as generated by the model.

Strong Predictions:

1. Image: 100003.jpg:
   
   
   Prediction: a photo of a red laptop on display at a store
   Comment: This is a good, specific prediction. The model correctly identifies the color (red), the object (laptop), its status (on display), and the likely context (at a store).

2. Image: 100007.jpg:
   
   Prediction: a photo of a small airplane parked on the tarmac
   Comment: This is a clear and accurate caption. It correctly identifies the type of aircraft (small airplane), its state (parked), and its location (on the tarmac), which matches the visual well.

3. Image: 100019.jpg:

Prediction: a photo of a samsung notebook computer
Comment: This is a concise and accurate identification, correctly naming the brand (Samsung) and the type of device (notebook computer), which are the key elements in the image.

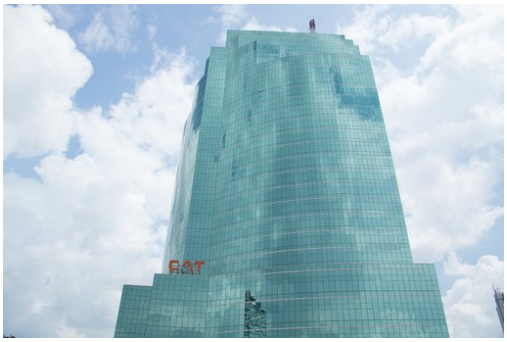
Poor/Failure Cases (or areas for improvement):

1. Image: 100001.jpg:

Prediction: a photo of a person holding a knife on a beach

Comment: Partially correct but potentially misleading or incomplete. The image shows a hand with a watch and wallet, with two large knives stuck upright in a piece of wood on a beach. The "person holding a knife" is not entirely accurate; the knives are displayed. The other prominent items (watch, wallet, beach setting) are captured by the model's likely understanding but not fully articulated in this specific caption. This highlights a case where the model might focus on one salient object ("knife") and simplify the scene.

2. Image: 100005.jpg:




Prediction: a photo of a tall building with a sky background
Comment: Too generic. While true (it's a tall, modern glass building against a cloudy sky with "CAT" logo), the caption lacks specificity. It doesn't mention the reflective glass, the modern architecture, or the visible logo, which could make the caption more informative.

3. Image: 100013.jpg

Prediction: a photo of poker chips and dice
Comment: Accurate but incomplete. The image shows a royal flush in hearts along with poker chips and a die. The caption identifies "poker chips and dice" but misses the very significant element of the playing cards forming a strong poker hand. This shows a limitation in recognizing more complex or specific arrangements of common objects.

---

## 5. References


- Li, J., Li, D., Savarese, S., & Hoi, S. (2023). BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. arXiv preprint arXiv:2301.12597. (Accessed from: https://arxiv.org/abs/2301.12597)
- Hugging Face Transformers Library: https://huggingface.co/docs/transformers/index
- Salesforce/blip2-opt-2.7b model card: https://huggingface.co/Salesforce/blip2-opt-2.7b
- BitsAndBytes Library: https://github.com/TimDettmers/bitsandbytes