Evagelos Petropoulos

Project 2 Part 2

Choice of method (prompt engineering)

For this project, I decided to go with OpenAI's CLIP. I chose this over BLIP as CLIP can output evaluation and classification, compared to BLIP, which is better at generating captions and text generation from the images themselves.

CLIP is a neural network created by OpenAI that can learn and recognize images by using natural language supervision(*Clip: Connecting Text and Images / Openai*). Unlike other neural network models that are available, CLIP is pre-trained from a wide variety of images available on the internet along with natural language supervision(*Clip: Connecting Text and Images* | *Openai*). This lets it not have to be fine-tuned specifically with a dataset to perform favorably, although that option can be explored to increase performance.

Prompt engineering was the key to allowing the AI model I chose to be able to function correctly. It includes instructions or requests to the model that will yield the best image classifications from the CLIP model("Prompt Engineering: The Magic Words to Using OpenAI's Clip."). Originally, I included instructions for the classes that were more of what the objects of the class inside the images could do(i.e. for a fork, "used to hold meat when cutting it"). This did not provide favorable results. It took a bit of trial and error to be able to find instructions that worked better with the model. After doing some more research, I found that using more descriptive language of the object itself provided better results(for the same fork, "a metal fork with a reflective surface").

Results, successes, and limitations

My final accuracy result given from the model I used, patch16 of CLIP, was 33.33%. While this does not seem high, this is actually a substantial improvement from my initial results and what I was expecting from a dataset of just 30 image-instruction pairs. My initial accuracy result came out to just 10%. A completely random assignment of instruction to image guessing would be 1/30 which is about 3.33%, so this was still better than that. I felt this was not enough. The first change I made was, since originally I was using patch32 of the CLIP model, I switched to patch16 as I read that it could work better on small datasets and it did. It doubled the accuracy of my model, bringing it up to 20%. I was still confused as to why it was so low, so I did more research into prompt engineering, and saw that my instructions I was giving to the model could be greatly improved. After I fixed my instructions up in my json file, the performance shot up to 33% accuracy.

There are some limitations with the model within the scope of this project. First was the amount of image-instruction pairs. A more robust version of the model could have been used for greater accuracy but trying to use one with this small dataset actually lowered the accuracy. Second was the lack of fine tuning. Since it was not specified in the instructions of the project, I did not finetune the model with images from the original database I used to grab image classes from(/kitchenware_5000 from https://huggingface.co/datasets/RitaSha/kitchenware_5000).

Works Cited

Clip: Connecting Text and Images | Openai, openai.com/index/clip/. Accessed 27 Oct. 2025.

CLIP, huggingface.co/docs/transformers/en/model_doc/clip. Accessed 27 Oct. 2025.

Nelson, Joseph. "Prompt Engineering: The Magic Words to Using OpenAI's Clip." *Roboflow Blog*, Roboflow Blog, 18 May 2021, blog.roboflow.com/openai-clip-prompt-engineering/.