**Project: Finetune OpenAI’s gpt-4o-2024-08-06 for medical image data**

**Objective:**

The gpt-4o-2024-08-06 model has been trained on a wide range of image data. Therefore, it is not accurate enough to identify organ positions in medical images, such as MRI. Therefore, the objective of this project was to fine-tune the GPT-4o-2024-08-06 on a medical image dataset called MISR to improve its accuracy in identifying the relative positions of organs.

**Dataset:**

For this finetuning MISR RQ2 - images\_letters dataset was used. In this dataset, each organ in each image is labeled with a specific letter. The model is then fine-tuned to identify the relative positions of organs via these specific letters. There are a total of 4878 images available. Out of which 1220 images were used to finetune the model, and the rest of the images were used to test and compare the base model and the finetuned model accuracy.

**Finetuning pre-processing:**

There is a README file named as README\_finetune\_chatgpt.txt in the finetune\_chatgpt folder, which describes the overall pipeline, including preprocessing and finetuning.

**1.File split\_data.py:**

The purpose of this script is to randomly select 1220 images from the complete dataset for fine-tuning. The output of this script is that the images for fine tuning are saved in the test\_data folder. These images are then deleted from the main image folder.

To use this script, the following things need to be changed:

1. source\_dir- This is the path for the main images folder.
2. dest\_dir - The path for the destination data folder e.g., test\_data.
3. log\_dir - Place to save the log\_file.
4. json\_file - Path to image metadata file e.g., qa\_letters.json.
5. log\_file - The text file to save the images name used for finetuning.
6. num\_images - Number of images to be used for fine tuning.

**2. File split\_qa\_letters -**

The purpose of this file is to filter the original QA JSON (qa\_letters.json) to exclude the 1220 fine tuning images. The output of this script is that the remaining QA metadata is saved in the qa\_letters\_ inference.json file. This file is used as test data with all\_experiments\_chatgpt.py. Thai script also includes the verification step to ensure all reference images files are present.

To use the script, the following things need to be changed:

1. qa\_json\_path - Path to the full QA dataset.
2. Image\_list\_path - Path to the text file containing image file names used for fine-tuning.
3. image\_folder - Path to the folder containing images.
4. image\_key - the key in each JSON entry used to match image file names.

**3. File generate\_jsonl.py -**

The purpose of this file is to convert the fine tuning subset of QA data into OpenAI’s finetuning “.jsonl” format. It takes the original QA JSON (qa\_letters.json) and fine\_tuning\_images.txt file. Each image is converted to a base64-encoded image. The output file is vision\_train\_1220.jsonl.

Here is the example for one image and its .jsonl entry.

{

"messages": [

{

"role": "system",

"content": "You are an assistant that answers positional queries about organs in MISR images. Answer only Yes or No."

},

{

"role": "user",

"content": "Is the left kidney (AC) below the inferior vena cava (CK)?" // Question from qa\_letters.json

},

{

"role": "user",

"content": [

{

"type": "image\_url",

"image\_url": "https://your-image-url-here.com/image.png"

}

]

},

{

"role": "assistant",

"content": "Yes" // Real output from qa\_letters.json

}

]

}

To use this file, the following things needed to be changed:

1. IMAGE\_DIR - Directory containing image files, e.g., test\_data.
2. META\_PATH - Path to the metadata JSON file e.g., qa\_letters.json.
3. OUTPUT\_PATH - Path to the output JSONL file.
4. SYSTEM\_PROMPT - IT can be kept as it is or can be changed based on the need.

**4. File finetune\_chatgpt.py -**

The purpose of this script is to upload the ‘.jsonl’ file and launch a finetuning job using the OpenAI API. It uploads the data start fine-tuning on the base model gpt-4o-2024-08-06. It also monitors job ID and status.

The fine-tuning was performed using Supervised Fine-Tuning (SFT), specifically for vision-based tasks using GPT-4o. The documentation can be found [here](https://platform.openai.com/docs/guides/vision-fine-tuning). The dataset was prepared in .jsonl format and uploaded via the OpenAI API for training. While OpenAI does not publicly disclose whether the fine-tuning involves full model training or a parameter-efficient approach such as LoRA (Low-Rank Adaptation), several key indicators suggest the latter. For example, Microsoft Azure’s OpenAI service explicitly uses LoRA for fine-tuning the GPT-3.5 Turbo model ([link](https://learn.microsoft.com/en-us/answers/questions/1689180/what-fine-tune-method-is-used-under-the-fine-tune?utm_source=chatgpt.com)),, and OpenAI’s own open-source models (e.g., gpt-oss) support LoRA-based fine-tuning. Additionally, in this project, the fine-tuning process took only approximately 95 minutes for 1220 image-text pairs, which strongly suggests that the process involved partial fine-tuning (e.g., adapter-based methods like LoRA, IA3, Adapter Tuning) rather than full parameter updates. LoRA is **most common and standard adapter-based fine-tuning method**.

**LoRA (Low-Rank Adaptation):**

It is an efficient fine-tuning method for large language models. Instead of updating all of the model’s parameters, LoRA inserts small trainable matrices into certain layers while keeping the original weights frozen. This significantly reduces memory usage and training cost, making it ideal for domain-specific adaptation.

LoRA modifies the model by adding low-rank matrices to key weight projections (typically in the attention layers). The original weight matrix W remains frozen, and the update is modeled as:

Where, The is the dimensionality of the model layer, i.e., the number of hidden units (usually several hundred or thousands). The is the rank of the LoRA adaptation — a small integer (like 4, 8, 16, etc.) that controls the compression level and the number of trainable parameters.These matrices A and B are the only parameters trained during fine-tuning. By doing so, LoRA enables scalable and efficient training with minimal impact on inference speed or memory, while maintaining performance comparable to full fine-tuning.

**5. File status.py -**

The purpose of this file is to retrieve and display the status of the finetuning job using jobID. In the end of this pipeline we get a finetuned GPT4o model tailored for our dataset. The fine tuned model name is

ft:gpt-4o-2024-08-06:viscom::Bp0Y8FmM.

This model can only be accessed using our specific OpenAI’s API.

**Results and Discussion:**

The remaining images (3658) were used in inference using all\_experiments\_chatgpt.py as the test data using both the base model and fine tuned model. The outputs of both models were later evaluated and compared. These are shown in the following table. It can be clearly seen that the accuracy and F1 score has been improved drastically when using fine fine-tuned model compared to the base model. [Results](https://github.com/Gunjan740/finetune_gpt4o/blob/main/results/1_model_answers/Results_Images.xlsx) can be seen here.

| Parameters | Base model | Finetuned model |
| --- | --- | --- |
| Overall Accuracy | 59% | 96% |
| Accuracy left right | 58% | 96% |
| Accuracy above below | 64% | 98% |
| F1 score mean | 50% | 97% |
| F1 score left right | 49% | 97% |
| F1 score above below | 50% | 98% |
| Correct answers all runs | 2150, 2138, 2181 | 3539, 3551, 3528 |
| Wrong answer all run | 1508, 1520, 1477 | 119, 107, 130 |