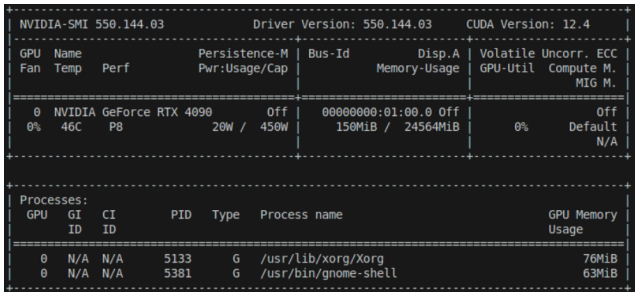
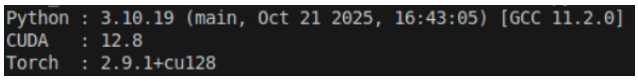
LLaVA

Name: Shankai Wu, Student ID: R14922146

Environment Screenshot



(a) NVIDIA GPU



(b) Python and PyTorch versions

Figure 1. (a) NVIDIA GPU: RXT-4090, (b) Python and PyTorch versions.

Codebase Structure

1. **`prepared_vqarad.py`:**
   This script loads the original VQA-RAD annotations and image files, and converts them into the JSON format required by the assignment. Each example is transformed into `{"image": image_filename, "question": question, "answer": answer}` and saved as `vqa-rad-train.json` and `vqa-rad-test.json` under
2. **`./ms-swift/eval.py` (zero-shot baseline):**
   Use the script runs VQA-RAD zero-shot evaluation using the pre-trained `llava-hf/llava-1.5-7b-hf` model. It loads `vqa-rad-train.json` and `vqa-rad-test.json` created by `prepared_vqarad.py`, performs inference over the whole dataset, and computes BLEU scores.
3. **`zero_shot_llava.py`:**
   This script implements interactive zero-shot inference for the five selected images. It loads the pre-trained Llava-1.5 model and its processor, feeds each image 2 questions, and outputs the model's answers (printed or saved as JSON). The generated examples (image, human question, and model response) are used in the report for qualitative comparison with the LoRA fine-tuned model.
4. **`run_lora_vqarad.py`:**
   This script wraps the long `swift sft` command into a Python program so that training can be easily repeated and hyper-parameters can be tuned. It fixes Llava-1.5-7B as the base model with `train_type=lora`, points to the VQA-RAD train/val JSON files, and sets the main hyper-parameters (such as `num_train_epochs`, batch sizes, `lora_rank`, and `lora_alpha`). The script prints the full command for logging,and then saving LoRA checkpoints under `checkpoints/llava-vqa-rad-lora/`.
5. **`eval_lora_vqarad.py`**
   This script loads the same Llava-1.5 base model as in the zero-shot experiments and attaches the fine-tuned LoRA adapter via `peft.PeftModel.from_pretrained` from the saved checkpoint. It then runs inference on the VQA-RAD test JSON with exactly the same evaluation pipeline as the baseline and computes BLEU scores.
6. **`qualitative_vqarad_3samples.py`:**
   Selects three image–question pairs from the VQA-RAD dataset and runs inference with (i) the zero-shot LLaVA-1.5-7B model and (ii) the LoRA fine-tuned model. The script saves the image, question, ground-truth answer, and both model predictions to disk, providing the qualitative examples used in the report.

1. VQA-RAD Fine-tuning Pipeline

1.1. Data

In this assignment, I use the VQA-RAD dataset, which consists of radiology images paired with question–answer annotations. In my project, images are stored under `data/vqa-rad/images/*.png`, and the annotations are converted into `vqa-rad-train.json` and `vqa-rad-test.json` for Swift.

1.2. Pre-processing for LLM-based VQA

To make the data compatible with LLaVA/Swift, each original `(image, question, answer)` triple is converted into a multimodal conversational `messages` format. For example:

```
{
  "image": "images/train_00001.png",
  "messages": [
    {
```

```
      "role": "user",
      "content": [
        {"type": "image"},
        {"type": "text", "text": "...question..."}
      ]
    },
    {
      "role": "assistant",
      "content": [
        {"type": "text", "text": "...answer..."}
      ]
    }
  ]
}
```

This format encodes the interaction "image + question → answer" and matches Swift's `llava1_5_hf` chat template.

Swift automatically joins `$ROOT_IMAGE_DIR` with the relative path to load images. I set `max_length = 512` to control the total number of text tokens plus image tokens. Rare over-long samples are truncated or skipped by Swift so that training does not crash due to excessive sequence length.

1.3. Fine-tuning with Swift (LoRA)

After pre-processing, I fine-tune the **LLaVA-1.5-7B** model using Swift's `swift sft` command with LoRA. The main configuration (invoked via `run_lora_vqarad.py`) is summarized below:

- Model:
  `-model llava-hf/llava-1.5-7b-hf`
  `-model_type llava1_5_hf`
- Training type:
  `-train_type lora`
  `-target_modules all-linear`
  `-freeze_vit true`
- LoRA hyperparameters:
  `lora_rank = 8`
  `lora_alpha = 32`
- Datasets:
  `-dataset vqa-rad-train.json`
  `-val_dataset vqa-rad-test.json`
- Optimization:
  `num_train_epochs = 3`
  `learning_rate = 1e-4`
  `per_device_train_batch_size = 2`
  `gradient_accumulation_steps = 4`
  `max_length = 512 torch_dtype = bfloat16`
- Hardware / output: training is run on an RTX 4090, and LoRA checkpoints are saved under `checkpoints/llava-vqa-rad-lora` (e.g. version folder `v4/`) for later evaluation.

2. LLaVA Inference

The result will save as `zero_shot_results.md` in the project. The table is shown below (`figure 6 to 10`).

3. Fine-tune

```
cmd = [
    "swift", "sft",
    "--model", "llava-hf/llava-1.5-7b-hf",
    "--model_type", MODEL_TYPE,
    "--train_type", "lora",
    "--dataset", train_json,
    "--val_dataset", val_json,
    "--num_train_epochs", str(NUM_TRAIN_EPOCHS),
    "--per_device_train_batch_size", str(PER_DEVICE_BATCH_SIZE),
    "--per_device_eval_batch_size", str(PER_DEVICE_BATCH_SIZE),
    "--gradient_accumulation_steps", str(GRAD_ACC_STEPS),
    "--learning_rate", "1e-4",
    "--lora_rank", str(LORA_RANK),
    "--lora_alpha", "32",
    "--target_modules", "all-linear",
    "--freeze_vit", "true",
    "--max_length", str(MAX_LENGTH),
    "--eval_steps", "100",
    "--save_steps", "200",
    "--save_total_limit", "2",
    "--logging_steps", "20",
    "--torch_dtype", "bfloat16",
    "--output_dir", output_dir,
]

env = os.environ.copy()
env["CUDA_VISIBLE_DEVICES"] = CUDA_VISIBLE_DEVICES
# ✅ 用環境變數告訴 swift：圖片根目錄在哪裡
env["ROOT_IMAGE_DIR"] = root_image_dir
```

Figure 2. Enter Caption

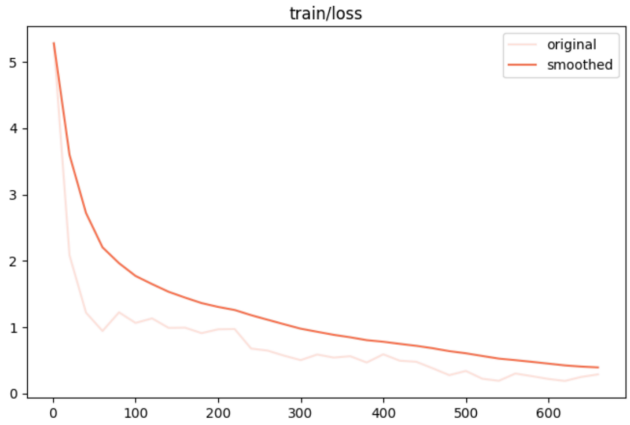
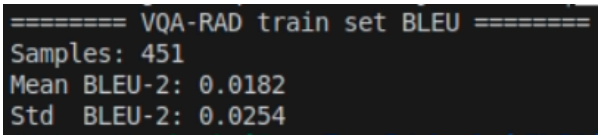
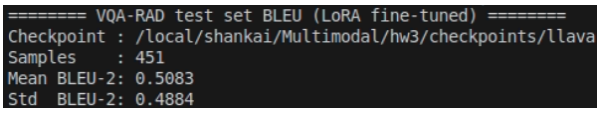


Figure 3. LLaVA LoRA fine-tune loss curve

Figure 4. training cmd



(a) w/o Fine-tune



(b) w/ Fine-tune

Figure 5. BLEU with and without fine-tune

4. Findings and Insights

In this section, I summarize the main observations from my experiments on VQA-RAD with LLaVA-1.5-7B and Swift LoRA fine-tuning[1].

4.1. Zero-shot performance is already strong

One of the most surprising findings is that the *zero-shot* performance of the base LLaVA-1.5-7B model[1] on VQA-RAD is already quite strong. On the full test split, the fine-tuned model only improves the accuracy by a small margin compared to the zero-shot baseline, rather than providing a dramatic jump in performance. In other words, most of the time the base model already answers reasonably well, and fine-tuning mainly provides incremental gains.

4.2. Fine-tuning improves stability and style rather than radically changing accuracy

From the quantitative results, the fine-tuned model is consistently but only slightly better than the zero-shot model. The more noticeable difference appears in the qualitative analysis:

- The fine-tuned model tends to give shorter, more direct answers (e.g. a single word or a clear "Yes/No"), which is easier to evaluate and matches the annotation style of VQA-RAD.
- The zero-shot model sometimes adds extra explanations or slightly off-topic text, even when the core answer is correct. Fine-tuning reduces this verbosity and makes the outputs more consistent.

Overall, the fine-tuned model feels more *stable* and aligned with the dataset format, but the underlying visual and language understanding is already very good in the zero-shot model.

4.3. Quantitative vs. qualitative perspective

Finally, the three selected qualitative examples help illustrate differences that pure accuracy numbers cannot fully capture:

- In many cases, the zero-shot model already captures the key visual cues and the semantics of the question.
- The fine-tuned model produces answers that are more concise and better aligned with the expected label space of VQA-RAD.

Therefore, my overall takeaway is that fine-tuning brings small but consistent improvements, mainly in output behavior, while the zero-shot baseline of a modern multimodal LLM is already surprisingly competitive on VQA-RAD.

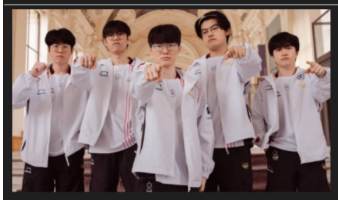
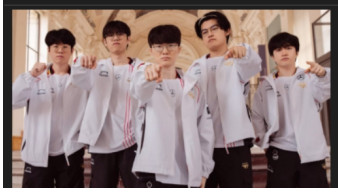
Image	Question	Answer	Comment
	How many people in the image are pointing at the camera?	There are three people in the image pointing at the camera.	Answer is wrong (the answer is four)
	What is the dominant color of the uniforms worn by the people?	The dominant color of the uniforms worn by the people is white.	Answer is correct

Figure 6. 01.jpeg

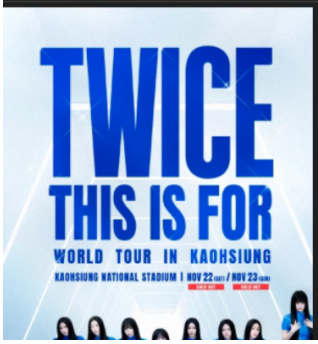
	According to the poster, at which venue will the group perform?	The group will perform at the Kahshung National Stadium.	Answer is correct
	What represent the dates of the event listed on the poster?	The dates of the event listed on the poster are November 22nd and 23rd.	Answer is correct

Figure 7. 02.jpeg

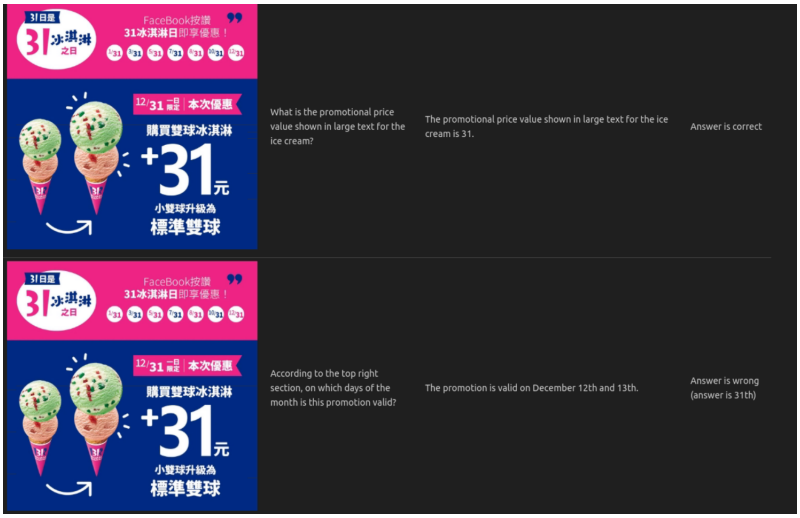


Figure 8. 03.jpeg

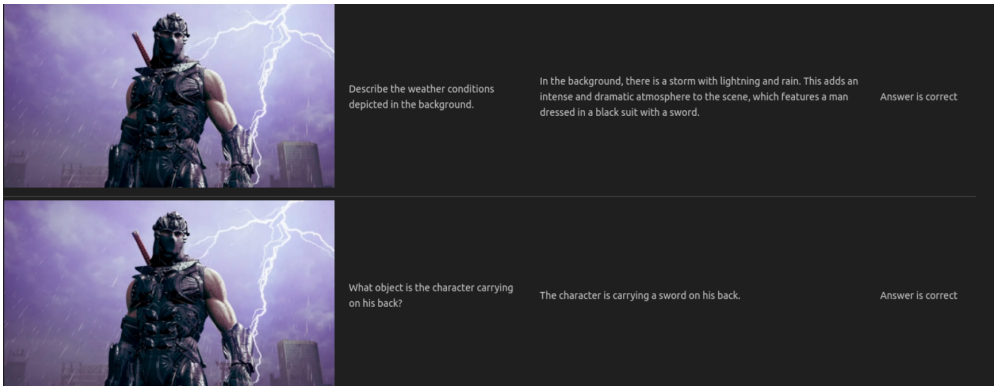


Figure 9. 04.jpeg

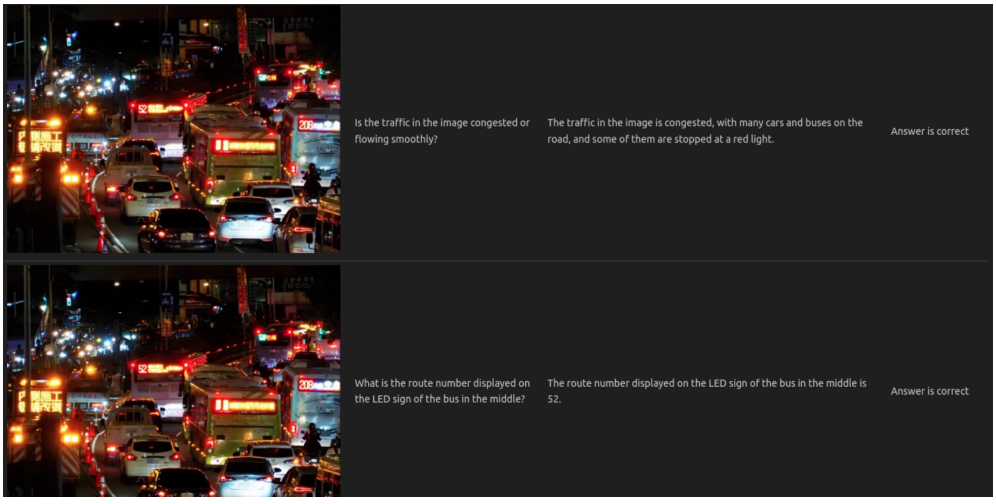


Figure 10. 05.jpeg

References

[1] Haotian Liu. Visual instruction tuning, 2023.