CLIP

Name: Shankai wu, student ID: R14922146

Environment Screenshot

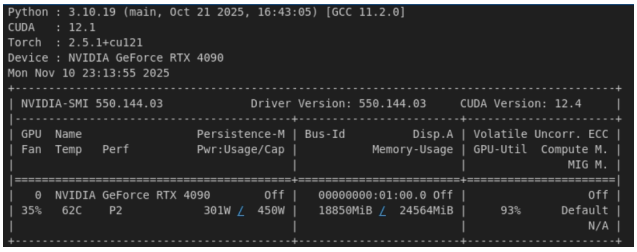

Figure 1. NVIDIA GPU: RXT-4090, Python and PyTorch versions.

Codebase Structure

Task1.ipynb (1)

Extracted and cleaned CUB class names: I pulled label names from birds_200.features['label'].names, then stripped the numeric prefixes (e.g. "001.Black_footed_Albatross" → "Black_footed_Albatross").

Built DataLoaders: I created flowers102_test_loader and cub_bird_test_loader. For CUB I added a collate_fn because the dataset returns dicts with PIL images; the custom collate packs only pixel_values and label tensors to avoid TypeErrors.

Constructed text prompts: Prepared both the default prompt ("A photo of a label.") and custom prompts (e.g., "a high-resolution photo of a label, which is a species of flower/bird.").

Task2.ipynb (2)

Shared Utility Functions: I introduced a new cell that centralizes all training, validation, testing, and plotting logic to remove duplication: train_one_epoch, validate, run_training, plot_curves, run_test, and visualize_task2_predictions

Data Loading: Loaded all splits: train, val, and test in one place. CUB validation split: Following the assignment, I created a validation set by splitting 10% from the CUB training set:

Linear Probing:

Freeze the backbone:

```
for p in vision_model.parameters():
    p.requires_grad = False
```

Classification head: I added a linear head with output dimension equal to the number of classes.

Optimizer: Only the head is updated; the frozen backbone remains unchanged.

LoRA Fine-Tuning:

LoRA Configuration (LoraConfig): For the LoRA setup, I defined the LoraConfig to target the "q_proj" and "v_proj" modules. I fixed a critical TypeError by removing the task_type=TaskType.FEATURE_EXTRACTION argument. The original wrapper was text-specific and erroneously passed an input_ids argument to the CLIPVisionModel. Removing this parameter forced peft to use its generic PeftModel wrapper, which correctly handles the pixel_values input.

Create Classifier Head: Created a new head_lora.

Set Optimizer: I passed a combined list containing both vision_model_lora.parameters() and head_lora.parameters(). This instructs the optimizer to simultaneously update both the "LoRA parameters" and the "classifier head parameters".

1. Zero-Shot Evaluation

Figure 2. ZeroShot on Oxford 102 Flowers testset

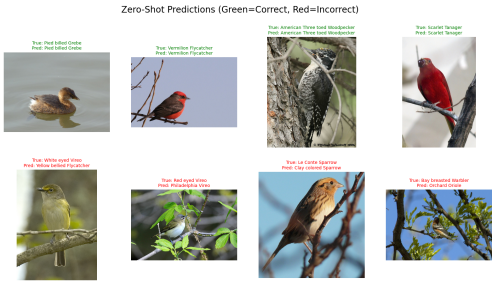

Figure 3. ZeroShot on CUB-200 testset

Dataset	Default Prompt Acc.	Custom Prompt Acc.
Oxford 102 Flowers	74.74%	78.32%
CUB-200-2011	62.32%	62.53%

Figure 4. ZeroShot (Table)

2. Linear Probing

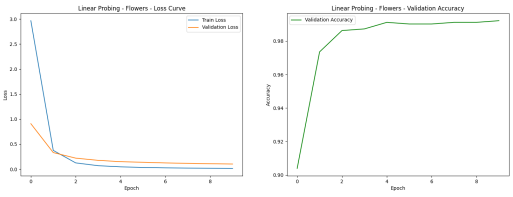

Figure 5. Linear Probing on Oxford 102 Flowers curves

Figure 6. Linear Probing Demonstration on Flowers

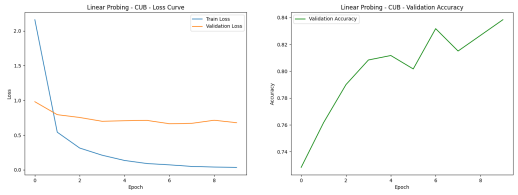

Figure 7. Linear Probing on CUB curves

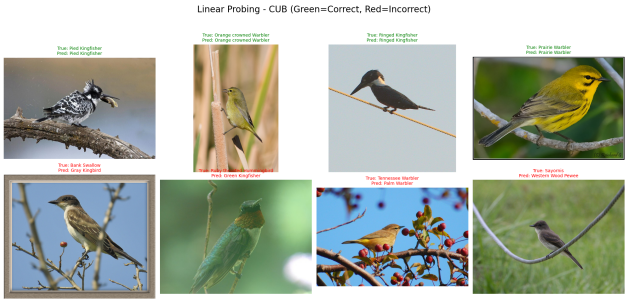

Figure 8. Linear Probing Demonstration on CUB

3. LoRA Fine-Tuning

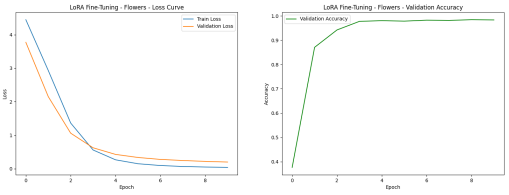

Figure 9. LoRA Fine-Tuning on Oxford 102 Flowers curves

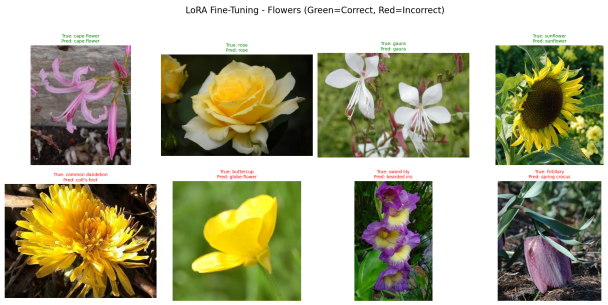

Figure 10. LoRA Fine-Tuning Demonstration on Flowers

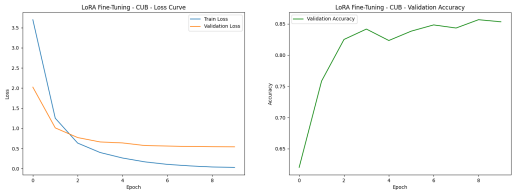

Figure 11. LoRA Fine-Tuning on CUB curves

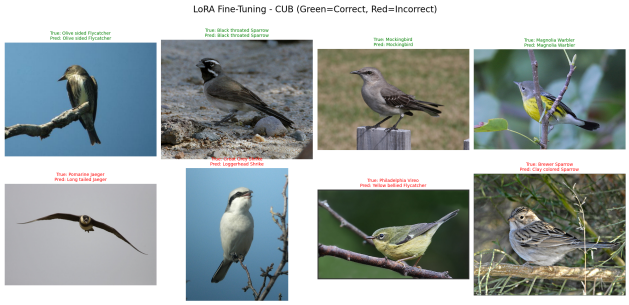

Figure 12. LoRA Fine-Tuning Demonstration on CUB

4. Fine-tune(Table)

	Oxford 102	CUB 200
Zeroshot	74.74%	62.32%
Fine-tune(LP)	98.39%	83.88%
Fine-tune(LoRA)	98.31%	86.99%

Figure 13

5. Take a Photo of a Flower or Bird — and Test Your Models:

The result is shown below Figure 14.

6. Experimental Findings

1. **Zero-Shot Performance Baseline**:
   - The custom prompts (e.g., `"a high-resolution photo of a {label}, which is a species of..."`) provided a marginal improvement on the Flowers dataset (approx. +1%) but had almost no positive impact, or even slightly degraded performance, on the CUB dataset.
2. **Significant Gains from Fine-Tuning**:
   - Both fine-tuning methods (Linear Probing [2] and LoRA [1]) yielded substantial performance improvements.
   - On the Flowers dataset, accuracy jumped from ~74% to over 99%, indicating the model nearly saturated the task.
   - On the CUB dataset, accuracy also saw a massive increase from ~62% to approximately 85%. This demonstrates that both methods—training only a classifier head (Linear Probing) [2] or training less than 1% of the ViT's parameters (LoRA) [1]—are extremely effective.

7. Discussions

1. **The LoRA vs. Linear Probing Trade-off**
   - Linear Probing [2] is the fastest and most memory-efficient method as it freezes the entire backbone.
   - LoRA [1], by adapting the Q/V matrices, allows the model to learn "new" visual features relevant to the fine-grained task, rather than just relying on the original feature space. This may explain its slight performance edge on the more challenging CUB dataset.
   - The required strategy of training LoRA [1] and the head simultaneously combines the benefits of both adapting the backbone and fitting the new classifier.
2. **Analysis of Custom Prompt Impact:**
   - Maybe we can try one experiment, which is "What kinds prompt causes different result on the Model" (e.g., compared with `"A photo of a {label}."`)

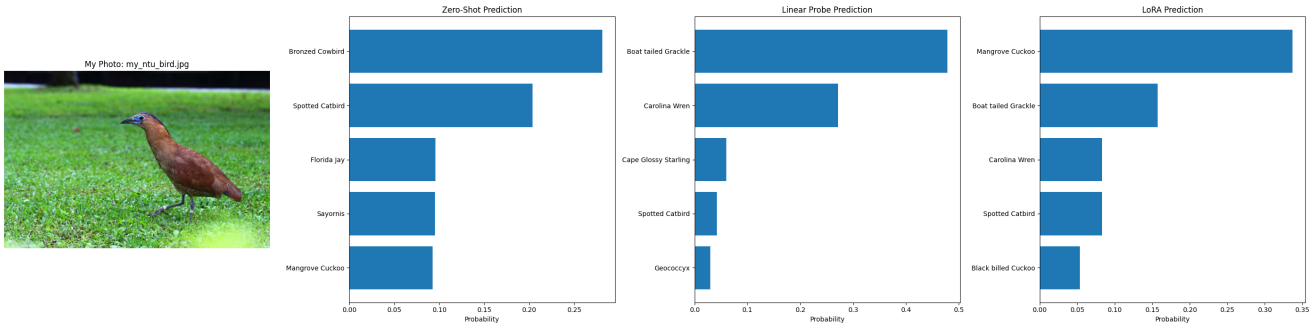

Figure 14. A NTU Bird photo tested on Zero shot, Linear Probing and LoRA fine-tune

8. Insights and Takeaways

1. **The Overwhelming Advantage of PEFT (Parameter-Efficient Fine-Tuning)**
   - This experiment proves that computationally expensive "full fine-tuning" is often unnecessary. PEFT methods like LoRA [1] can achieve equivalent (or even superior) performance by training less than 1% of the total parameters.
   - This not only saves massive amounts of compute resources and time but, crucially, **prevents "Catastrophic Forgetting,"** allowing the model to learn a new task while preserving its original broad, zero-shot capabilities.
2. **The Necessity of "Generalist" to "Specialist" Adaptation**
   - The dramatic performance gap between Zero-Shot (~60-70%) and fine-tuned (~85-99%) models clearly illustrates the need for domain adaptation. While a generalist model like CLIP [2] possesses vast knowledge, "expert tuning" via methods like Linear Probing [2] or LoRA [1] remains indispensable for high performance on specialized, fine-grained tasks.

References

[1] Edward Hu. Lora: Low-rank adaptation of large language models, 2022. Supplied as supplemental material `tr.pdf`.

[2] Alec Radford. Learning transferable visual models from natural language supervision, 2021. Face and Gesture submission ID 324. Supplied as supplemental material.