Lightweight Vision-Language Modeling for Fine-Grained Bird Species Reasoning

Shufan He (she68), Yitong Liu (yliu336), Yunjia Zhao (yzhao291)

Github Link

Introduction

This project aims to develop a domain-specific, lightweight Vision-Language Model (VLM) for Visual Question Answering (VQA) focused on bird species identification. This project explores training a compact ViT-based image encoder on a bird-only dataset and connecting it with a language model through a simple and interpretable modality integration strategy.

Our motivation is to study whether lightweight VLMs, using minimal connection mechanisms such as a linear projection-based integration, can still achieve strong performance in fine-grained visual reasoning tasks. We focus on the bird domain to take advantage of its rich attribute annotations (e.g., color, shape, part-level details) and to explore VLM interpretability in specialized settings.

Challenges and Insights

Initially we tried training our model locally but it was extremely slow (~30min/ 2 epochs). We then switched to using Oscar, but had some difficulties with the connection.

The main challenge we faced was severe overfitting. After connecting to the GPU, we trained our model for 100 epochs. The training accuracy reached 100% but validation accuracy remained extremely low (~10%). Our first attempt modifying and tuning our model only resulted in a 3% increase in validation accuracy. When we used a pre-trained ViT, the accuracy was 70%.

We suspected that overfitting was primarily due to the limited size of the CUB-200-2011 dataset, which contains 11,788 images across 200 categories — relatively small for training a ViT model. To address this, we attempted to switch to a backup dataset with 126.7k images and over 500 categories. However, the download attempts failed, and we later discovered the source appeared to be fraudulent. We then identified another dataset with 48,000 images and 400 categories. Surprisingly, despite its larger size, this dataset led to worse validation accuracy.

Some other things we tried include regularization, augmentation, lr scheduling, gradient clipping, and implementing a weighted attention pooling strategy over patch tokens, rather than just using the CLS tokens alone for classification. None of the above methods resulted in significant improvements.

We expected the validation accuracy to be low because training a ViT from scratch typically doesn't yield great results. However, we still decided to give it our best effort, partly because the

TA emphasized that focusing on the model's design and learning process was more important than immediate accuracy.

Later in the project, we decided to integrate the pre-trained Tiny ViT model that trained on the CUB-2011 data set into a lightweight VLM framework. Specifically, we designed a VLM by connecting a Tiny ViT based encoder to a small language model-Ph-3-mini, connected via a two-layer MLP projection. However, we have more problems when training the vlm, initially we attempted to train the vlm directly on the cub-2011 dataset with our image-yexy question-answer jsons, and there are several new challenges emerge:

Overfitting Too Soon:
When training directly on the bird QA dataset, we observed the model quickly overfitting — training loss plummeted while validation loss stagnated. We suspected this was because the dataset, although carefully annotated, lacked the diversity needed for robust vision-language grounding.

Pretraining Attempts:
We hypothesized that initializing the VLM with some general vision-language pretraining could help. We attempted pretraining on the **LLaVA-CC3M-Pretrain-595K** dataset available on Hugging Face (link) to give the model a broader foundation before fine-tuning on birds.

Out-of-Memory (OOM) Issues the desk memory is 20GB:
During pretraining, we encountered memory bottlenecks — especially when using mixed-precision(fp-16) training combined with large batch sizes. Lowering the learning rate to 1e - 5 helped stabilize training slightly, but going too small resulted in the optimizer stalling, and larger learning rates 5e-5 led to faster but unstable convergence.

Small Subset Pretraining:
To debug and iterate faster, we experimented with pretraining on smaller random subsets of CC3M, which helped slightly reduce OOM risks but still couldn't fully prevent crashes at certain steps.

Image Padding (img_pad) Handling:
We explored increasing the number of <|image_pad|> tokens (e.g., up to 49 tokens) to better fit the output structure of the ViT feature maps. This allowed more fine-grained patch-level visual information to be aligned with the text sequence. However, larger image padding sizes slightly increased memory consumption.

Crucial Discovery:
We realized a critical mistake: the Tiny ViT weights we initially used were trained specifically on the CUB-200-2011 bird dataset. Thus, even though we "froze" the vision encoder, it was leaking task-specific features, artificially inflating early-stage accuracy. To correct this, we

discarded those weights and retrained using a Tiny ViT initialized from ImageNet or trained from scratch.

Training Dynamics:
When correctly initializing the vision backbone, we observed that during pretraining on CC3M, the loss consistently decreased to around 0.9 within the first 1000 steps (out of 5000 planned steps), suggesting reasonable convergence. However, true generalization was still limited, and full pretraining would require significantly longer training times and larger compute resources.

Results:

Vit training from scratch

```
Mon Apr 21 10:02:56 2025
+-----------------------------------------------------------------------------
---------+
Running on: gpu1602
[Epoch 1] Train Loss: 1010.2588 | Train Acc: 0.0078
[Epoch 1] Validation Accuracy: 0.0143
[Epoch 2] Train Loss: 970.5503 | Train Acc: 0.0120
[Epoch 2] Validation Accuracy: 0.0200
[Epoch 3] Train Loss: 954.7608 | Train Acc: 0.0167
[Epoch 3] Validation Accuracy: 0.0178
……
[Epoch 97] Train Loss: 1.4996 | Train Acc: 0.9998
[Epoch 97] Validation Accuracy: 0.0892
[Epoch 98] Train Loss: 1.5297 | Train Acc: 0.9998
[Epoch 98] Validation Accuracy: 0.0885
[Epoch 99] Train Loss: 1.6232 | Train Acc: 0.9998
[Epoch 99] Validation Accuracy: 0.0887
[Epoch 100] Train Loss: 1.5348 | Train Acc: 1.0000
[Epoch 100] Validation Accuracy: 0.0887
Model saved
Finished training at: Mon Apr 21 11:28:28 AM EDT 2025
```

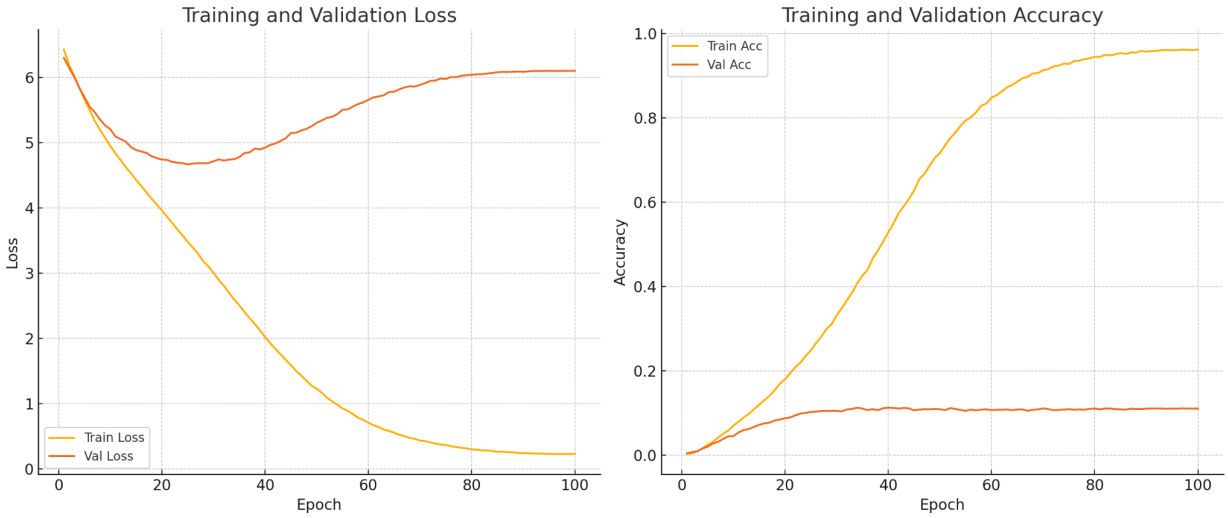
Training with a weighted attention pooling strategy over patch tokens, not just using the CLS tokens alone for classification and other methods:

```
Mon Apr 21 10:24:13 2025
+-----------------------------------------------------------------------------
---------+
| NVIDIA-SMI 535.216.01             Driver Version: 535.216.01   CUDA
Running on: gpu1602
[Epoch 1] Train Loss: 1047.8737 | Train Acc: 0.0068
[Epoch 1] Validation Accuracy: 0.0072
[Epoch 2] Train Loss: 1002.9758 | Train Acc: 0.0093
```

```
[Epoch 2] Validation Accuracy: 0.0121
[Epoch 3] Train Loss: 981.3098 | Train Acc: 0.0100
[Epoch 3] Validation Accuracy: 0.0173
……
[Epoch 98] Train Loss: 169.1392 | Train Acc: 0.8183
[Epoch 98] Validation Accuracy: 0.1350
[Epoch 99] Train Loss: 164.5387 | Train Acc: 0.8270
[Epoch 99] Validation Accuracy: 0.1348
[Epoch 100] Train Loss: 166.7099 | Train Acc: 0.8268
[Epoch 100] Validation Accuracy: 0.1348
Model saved
Finished training at: Mon Apr 21 11:57:13 AM EDT 2025
```

Training with nabirds(another dataset that is larger than cub-2011):

```
Training configuration: {'data_root': 'nabirds_split', 'num_classes': 555,
'batch_size': 32, 'epochs': 100, 'lr': 0.0001, 'weight_decay': 0.05,
'warmup_epochs': 5, 'img_size': 224, 'embed_dim': 192, 'depth': 12,
'num_heads': 3, 'num_workers': 4}

2025-04-23 10:03:12,980 - Using device: cuda

2025-04-23 10:03:23,113 - Dataset sizes - Train: 23929, Val: 24633

2025-04-23 10:03:25,980 - Starting training

2025-04-23 10:15:20,502 - Epoch 1/100, train_loss: 6.4264, train_acc: 0.0023,
val_loss: 6.2982, val_acc: 0.0044, lr: 0.000020

…

2025-04-23 17:53:10,815 - Epoch 95/100, train_loss: 0.2316, train_acc: 0.9608,
val_loss: 6.0994, val_acc: 0.1103, lr: 0.000001

2025-04-23 17:58:03,600 - Epoch 96/100, train_loss: 0.2281, train_acc: 0.9611,
val_loss: 6.0989, val_acc: 0.1101, lr: 0.000000

2025-04-23 18:03:05,186 - Epoch 97/100, train_loss: 0.2274, train_acc: 0.9622,
val_loss: 6.0994, val_acc: 0.1105, lr: 0.000000

2025-04-23 18:07:55,911 - Epoch 98/100, train_loss: 0.2281, train_acc: 0.9616,
val_loss: 6.1002, val_acc: 0.1103, lr: 0.000000

2025-04-23 18:12:44,752 - Epoch 99/100, train_loss: 0.2283, train_acc: 0.9613,
val_loss: 6.1004, val_acc: 0.1101, lr: 0.000000

2025-04-23 18:17:32,265 - Epoch 100/100, train_loss: 0.2296, train_acc: 0.9620,
val_loss: 6.1003, val_acc: 0.1102, lr: 0.000000

2025-04-23 18:17:32,631 - Training completed.

Model saved as tinyvit_nba.pth
```



The classic overfitting when training on the smaller dataset

When training vlm with cc3m data:

```
Loading checkpoint shards:
100%|████████████████████████████████████████████████████| 2/2 [00:03<00:00,
1.75s/it]
Number of trainable parameters: 5024256
2025-04-25 12:44:20,978 - INFO - Loading dataset from:
llava_dataset/subset_chat.json
2025-04-25 12:44:22,394 - INFO - Using 200000 samples for training
Epoch 1/2:   1%|                                    | 289/50000
[00:52<2:26:11,  5.67it/s, loss=10.5, lr=7.2e-8]2025-04-25
12:45:14,742 - INFO - Sample 1 - Input IDs length: 73
2025-04-25 12:45:14,742 - INFO - Sample 1 - Labels length: 73
2025-04-25 12:45:14,742 - INFO - Sample 1 - First 10 labels: [-100,
-100, -100, -100, -100, -100, -100, -100, -100, -100]
2025-04-25 12:45:14,742 - INFO - Sample 1 - Last 10 labels: [915,
1300, 6845, 6337, 284, 373, 263, 6473, 32007, 32000]
Epoch 1/2:   2%|▏                                   | 799/50000
[02:25<2:20:57,  5.82it/s, loss=9.25, lr=2e-7]2025-04-25 12:46:47,554
- INFO - Step 100: Loss: 9.2463, LR: 0.000000
2025-04-25 12:46:47,555 - INFO - Recent loss trend - Min: 8.0759,
Max: 10.6954, Mean: 9.4781
Epoch 1/2:   2%|▏                                   | 800/50000
[02:25<4:06:48,  3.32it/s, loss=9.94, lr=2e-7]2025-04-25 12:46:47,755
- INFO - Step 100: Loss: 9.9352, LR: 0.000000
2025-04-25 12:46:47,756 - INFO - Recent loss trend - Min: 8.0759,
Max: 10.6954, Mean: 9.4839
```

```
Epoch 1/2:   2%|▌                                  | 801/50000
[02:25<3:42:11,  3.69it/s, loss=9.59, lr=2e-7]2025-04-25 12:46:47,952
- INFO - Step 100: Loss: 9.5877, LR: 0.000000
2025-04-25 12:46:47,952 - INFO - Recent loss trend - Min: 8.0759,
Max: 10.6954, Mean: 9.4821
…
[12:02<3:51:08,  3.32it/s, loss=0.752, lr=1e-6]2025-04-25
12:56:25,177 - INFO - Step 500: Loss: 0.7521, LR: 0.000001
2025-04-25 12:56:25,177 - INFO - Recent loss trend - Min: 0.2868,
Max: 1.3286, Mean: 0.8835
2025-04-25 12:56:25,177 - WARNING - Loss is converging unusually
fast. Check data and labels!
Epoch 1/2:   8%|██                                 | 4001/50000
[12:02<3:21:53,  3.80it/s, loss=0.914, lr=1e-6]2025-04-25
12:56:25,348 - INFO - Step 500: Loss: 0.9135, LR: 0.000001
2025-04-25 12:56:25,348 - INFO - Recent loss trend - Min: 0.2868,
Max: 1.3286, Mean: 0.8821
2025-04-25 12:56:25,348 - WARNING - Loss is converging unusually
fast. Check data and labels!
```

Plan

Given time constraints and the continued low accuracy issues with our self-trained ViT, we plan to move forward by finalizing the rest of the model architecture — specifically, the language model components for image captioning and VQA-style generation — to ensure timely completion. Instead of continuing to train a vision encoder from scratch, we will now use a pre-trained ViT model to generate image embeddings. These embeddings will serve as inputs to a language model for both image captioning and VQA-style generation tasks.

Regarding what we generally expect you to have **done** by this time:

- You should have collected any data and preprocessed it.
- You should have shared the Github repo link with your mentor TA
- You should have almost finished implementing your model, and are working on training your models and ablation experiments.
- Please make sure you are keeping your list of public implementations you've found up-to-date.