IT5005 Project Description

Rajendra Prasad Sirigina

AY 2025/26 Sem 1

Instructions

- **Submission:** You must submit your solutions online on the course canvas site. Refer Section 3 to prepare your submission material and for deadlines.
- **Generative AI Policy:** You are free to use any generative AI, but you are required to document the usage: which AI do you use, and what's the query to the AI? You are responsible for checking the correctness.
- **Late Policy:** No late submission is allowed.

1 Project Overview

In this project, you will learn the basics of multi-modal understanding, that is, integrating multiple modality types (e.g., images, text, etc.) as a single prediction task. In Part 1, you will learn the fundamental ideas of contrastive learning and why it works. Then, you will be tasked with implementing a zero-shot image classifier based on text labels and logistic regression with CLIP as a feature extractor.

Secondly, in Part 2, you will advance from using a pre-trained model as a fixed feature extractor to fine-tuning it for a new task: image-text retrieval. Using the Flickr8k dataset, you will implement a parameter-efficient fine-tuning loop where only the model's projection layers are trained. The core of this part involves implementing the InfoNCE contrastive loss function from scratch. Finally, you will evaluate your fine-tuned model's ability to retrieve correct images from text captions (and vice versa) using the Recall@K metric, comparing its performance to the zero-shot baseline from Part 1.

2 Project Tasks

Early AI models focused on single modalities, e.g., image-only or text-only predictions. However, a single modality often lacks the expressive power to encapsulate complicated ideas (imagine describing what an alien looks like to a vision-impaired friend). As different modalities are complementary and better mimic the way humans interact with the world (5 senses, not just one), multi-modal models have gained traction recently with immense progress, especially in the vision-language field. In this project, you will explore some foundational ideas in the intersection of vision and language.

2.1 Part 1

The project tasks for this part are as follows:

1. Read the paper titled "Learning Transferable Visual Models From Natural Language Supervision" to understand the foundation of the CLIP model. Answer the questions in Sec 2.1.1 in your report.

2. Set up your Python environment as shown in Section 4. You may use alternative methods at your own risk, but your submitted notebook must run successfully from top to bottom. Non-runnable code will receive a penalty.

3. Complete the Jupyter notebook `part_1.ipynb`. You are required to:

   (a) Implement the core components of the `CLIPEncoder` class to perform zero-shot image classification.

   (b) Implement evaluation metrics (`accuracy_score` and `f1_score`).

   (c) Conduct zero-shot classification experiments and analyze the effects of prompt engineering and similarity metrics.

   (d) Implement a simple linear classifier (`ClsNetwork`) for linear probing and complete the training logic in `Trainer`.

   (e) Compare the performance of zero-shot and linear-probing approaches through accuracy and F1-score.

4. Answer the analysis questions embedded in the notebook (re-iterated in Sec. 2.1.2). Your answers should be included in the report. These include discussions on prompt effects, similarity criteria, and comparisons between zero-shot and linear-probing methods.

5. Summarize your findings in your report, clearly stating your experimental setup, results, and conclusions. You are encouraged to relate your discussion to the CLIP paper and your implementation experiences.

2.1.1 Questions for Paper Reading

Read the paper titled "Learning Transferable Visual Models From Natural Language Supervision" and answer the following questions. The answers must be included in your report.

1. What does the CLIP model learn?

2. Explain in at most 3 sentences what "contrastive learning" means.

3. Why do you think CLIP's zero-shot performance can sometimes surpass supervised baselines? What does this say about the generalization abilities of representation learning?

4. How do the labels in CLIP-based zero-shot classification differ from traditional image classification models from supervised learning?

5. What do you think CLIP falls short in, and why do you think this happens?

2.1.2 Followup questions

Answer the follow-up questions (from the Jupyter notebook). The answers must be included in your report.

1. Try different prompts and record them. Which works best?
2. Instead of using a simple prompt like "an image of {class name}", what happens if we add random things to the front? E.g., "Beneath the fading sunset, the curious child wandered along the winding path, humming a gentle tune that echoed through the quiet forest, where birds whispered secrets from the branches above, and the air carried the fragrance of wildflowers, while dreams of adventure swirled within the stillness of twilight's embrace. *This is an image of {class name}*".
3. Does using cosine similarity differ from using a dot product as the decision criterion? Why or why not?

Deliverables You are to submit a ZIP file `group_<group_id>_part_1.zip` containing the completed Jupyter notebook `part_1.ipynb` and a report of your answers to the questions, in PDF format `part_1.pdf`.

2.2 Part 2

The project tasks for this part are as follows:

1. Complete the Jupyter notebook `part_2.ipynb` to implement a fine-tuning pipeline for the model using the Flickr8k dataset, improving its performance on cross-modal retrieval tasks. Note that your code must be runnable from top to bottom.
2. Answer the analysis questions embedded in the notebook (re-iterated in Sec. 2.2.1). Your answers should be included in the report. These include discussions on prompt effects, similarity criteria, and comparisons between zero-shot and linear-probing methods.

2.2.1 Followup questions

Answer the follow-up questions (from the Jupyter notebook). The answers must be included in your report.

1. **Performance Comparison:** Compare the final Recall@K performance of your fine-tuned models (both with InfoNCE and Triplet loss) with the initial zero-shot baseline. Did parameter-efficient fine-tuning improve the model's performance? Which loss function performed better on this dataset?
2. **Training Loss Analysis:** Plot the training loss curves for both experiments over epochs (you can view this using `tensorboard --logdir=runs`). Compare the convergence behavior of InfoNCE and Triplet loss. Did one converge faster or more smoothly than the other?
3. **Hyperparameter Experimentation (Learning Rate):** In Experiment 1, you used a learning rate of 1×10^{-5}. Rerun the InfoNCE experiment with:
   - a higher learning rate (e.g., 1×10^{-4}), and
   - a lower learning rate (e.g., 1×10^{-6}).

How do the training loss curves and final Recall@K metrics change? Discuss the effects of learning rate on model training.

4. **Overfitting Analysis:** Train the model with the best-performing setup from your previous experiments for more epochs (e.g., 15 or 20). Plot both the training loss and the test Recall@1 metric on the same graph against the epoch number. Do you see signs of overfitting (i.e., training loss continues to decrease while test performance stagnates or drops)? Based on the graph, what would be the optimal number of epochs to train for?

5. **Qualitative Analysis:** Provide one qualitative example from your best-performing model:

   - Show one image and its top-3 retrieved captions.
   - Show one text query and its top-3 retrieved images.

   Include at least one successful and one unsuccessful example. Briefly analyze why the model might have failed in the unsuccessful case.

6. **Discussion Question (Bonus):** Instead of training both projection layers, what would happen if you only trained the image encoder's projection layer while keeping the text side completely frozen, or vice versa? Try one of these experiments and discuss the results. What does this suggest about which modality adapts more effectively to the new dataset?

Deliverables You are to submit a ZIP file `group_<group_id>_part_2.zip` containing the completed Jupyter notebook `part_2.ipynb` and a report of your answers to the questions, in PDF format `part_2.pdf`.

3 Summary of Deliverables

The list of deliverables is presented in Table 1, which are to be submitted on Canvas.

Table 1: Deliverables

Tasks	**Deliverables**
Part 1	Jupyter notebook of your code (`part_1.ipynb`) and PDF report of answers (`part_1.pdf`) from sections 2.1.1 and 2.1.2. Submit a Zip file of both your notebook and report with the name `group_<group_id>_part_1.zip`.
Part 2	Jupyter notebook of your code (`part_2.ipynb`) and PDF report of answers (`part_2.pdf`) from section 2.2.1. Submit a Zip file of both your notebook and report with the name `group_<group_id>_part_2.zip`.

4 Environment Setup

This section entails the environment setup instructions.

4.1 Install Conda

We will use `conda` as our environment manager to isolate the packages used by this project from any other packages you may have installed. This also ensures that the correct versions of packages are installed, minimizing dependency conflicts.

We recommend installing `miniconda` (guide here), a command-line tool. The necessary commands for setting up a conda environment using `miniconda` are provided below. Alternatively, you may use `anaconda`, which provides a graphical user interface (guide here). However, we recommend getting familiar with terminal commands for your future benefit.

4.2 Creating the Environment

After installing conda, create a new virtual environment using the following commands:

```
conda create -n it5005 python=3.12 -y
conda activate it5005
conda install pip
```

4.3 Installing Necessary Packages

A list of required packages for this project is provided in `requirements.txt`. You can install them by running the command below:

```
pip install -r requirements.txt
```

Note that this installation process may take some time to complete.