ELEC 475
Lab 4

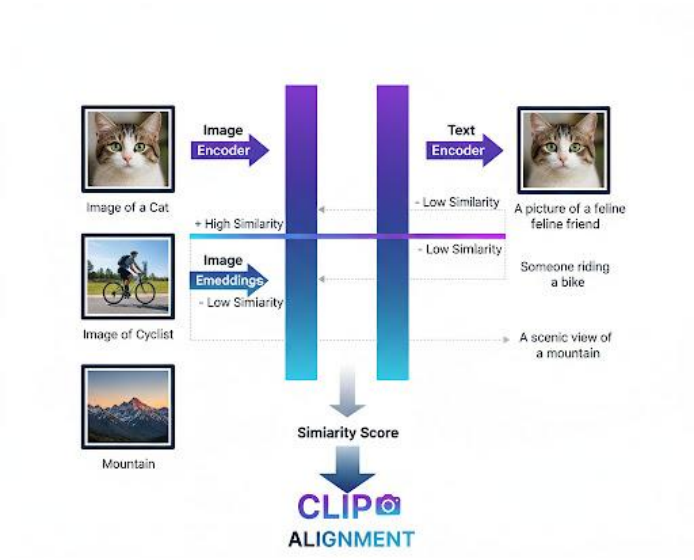

figure credit: M. Greenspan and Google Gemini

Lab 4: Fine-tuning ResNet50 for CLIP

Ali Zavareh, Justin Jacob,
and Michael Greenspan

11 November 2025

Table of Contents

1. Introduction

In this lab, you will explore one of the most influential systems in modern vision–language modelling: Contrastive Language–Image Pre-training (CLIP). The CLIP framework learns a joint embedding space where images and text captions that describe similar concepts lie close together. By training a model to align these modalities, CLIP enables flexible zero-shot classification, retrieval, and image–text reasoning.

You will implement and train a CLIP-style model where:

- The **text encoder** (a pretrained Transformer from OpenAI's CLIP model) remains **frozen**, and;
- The **image encoder** (a ResNet50 backbone) is **fine-tuned** to align its output embeddings with those from the text encoder.

The model will be trained on the **COCO 2014 dataset**, which provides paired image–caption annotations.

Through this lab, you will learn how to:

- Construct and train a contrastive vision–language model;
- Understand the mathematical formulation of CLIP's InfoNCE loss;
- Evaluate retrieval performance with quantitative metrics such as Recall@K;
- Analyze qualitative retrieval results on unseen examples.

2. Task Overview

2.1 Dataset Preparation – MS COCO

Use the **COCO 2014** dataset (available from Kaggle here: www.kaggle.com/datasets/jeffaudi/coco-2014-dataset-for-yolov3), including:

- Training images from `train2014/`
- Validation images from `val2014/`
- Captions from `captions_train2014.json` and `captions_val2014.json`

Implement a preprocessing step to:

- Load all images and captions
- Normalize images using the CLIP mean and standard deviation\
- Encode captions using the pretrained CLIP text encoder (available from huggingface, here: https://huggingface.co/openai/clip-vit-base-patch32)

Verify dataset integrity by displaying random image–caption pairs.

***Hint*:** Images must be resized to 224×224 and normalized using CLIP's mean values (0.48145466, 0.4578275, 0.40821073) and std values: (0.26862954, 0.26130258, 0.27577711).

***Hint*:** For easier training, you can encode all of the captions and save the text embeddings along with their respective image_id to .pt cache files for the train and val sets. Caching text embeddings can reduce training time and minimize GPU memory usage.

Hint: You can alternately download the dataset from the links below:

Link to cloud storage containing COCO dataset

Link to the official COCO dataset website

You must document:

- Dataset split sizes (If using a subset of the dataset);
- Image preprocessing (Resize → 224 × 224, normalization values);
- Text preprocessing (tokenization method, max length, padding strategy).

2.2 Model Design – CLIP

- Implement a **ResNet50** image encoder using PyTorch and initialize it with pretrained ImageNet weights;
- Add a **projection head** (two linear layers with GELU activation) that maps image features into a 512-dimensional CLIP embedding space;
- Freeze all parameters in the text encoder; only the image encoder and projection layers are trainable.

2.3 Training and Experimentation

- Define the **InfoNCE loss** for the CLIP model.

***Hint*:** LLMs can help you with the definition of the InfoNCE loss.

- Train your model to maximize alignment between paired images and captions. You may experiment with different learning rates, optimizers, and batch sizes. LLMs can help select the right parameters.

You must record and report:

- Training and validation loss curves.
- Total training time and hardware used.
- Observed issues (divergence, instability, etc.) and how they were addressed.

2.4 Evaluation and Visualization

During validation, compute the **cosine similarity matrix** between all image and caption embeddings.
Derive **Recall@1, Recall@5, and Recall@10** for both:

- Image to Text retrieval;
- Text to Image retrieval.

Visualize example retrievals:

- Given a text query (such as 'sport'), display the top-5 retrieved images;
- Given an image and a list of classes (such as ['a person', 'an animal', 'a landscape']), classify the image.

Discuss performance trends and training dynamics.

Recall@K measures how often the correct match (e.g., the true caption for an image, or the true image for a caption) appears in the top K retrieved results ranked by similarity. It's a way to evaluate how good your model's ranking is.

2.5 Modifications

Modify your system to improve accuracy. Modifications can include combinations of regularizations (e.g. normalization layers), architectural changes, data augmentation, and other. Implement at least two such modifications, and repeat the above (Section 2.4) evaluations on the modified system as an ablation study, to show which modifications (and their combinations) had the biggest impact on performance.

2.6 Analysis and Discussion

In your lab report:

- Describe all hyperparameters used in training, and describe the hardware used. How long did the training take? Include the loss plot;
- Explain the mathematical intuition behind the CLIP loss (anchors, positives, and negatives);
- Provide qualitative retrieval examples demonstrating semantic alignment between modalities;
- Reflect on your use of LLMs or chatbots during the lab; e.g., what queries were helpful and how you validated their outputs. Include a link to your conversation.

3. Deliverables

3.1 Completed Code

Submit a compressed project directory containing:

- All model, dataset, and training scripts (.py files);
- Two text files as below:
    - Train.txt (containing the command to start the training process;
    - Test.txt (containing the command to start the evaluation process;
- Include any generated qualitative results (retrieval samples).

3.2 Report

The report should include a title (e.g. minimally ELEC 475 Lab 4), your name and student number. If you are working in a team of two, then include the information for both partners in the report.
It is encouraged that your report contain below sections:

1. **Introduction**: Explain the motivation and structure of CLIP.
2. **Methodology**: Summarize your model and training design.
3. **Results**: Present quantitative metrics (Recall@K, loss curves).
4. **Discussion**: Interpret results and analyze model behavior.
5. **Conclusion**: Reflect on what was learned and potential improvements.
6. **Appendix**: Include relevant code snippets or diagrams.

4. Submission

You must submit a single .zip file named <studentID>.zip (or <id1>_<id2>.zip for teams) containing code, weights, plots, and a report.

5. Evaluation Rubric

Component	**Marks**
Implementation & Model Design	4
Training	1
Modifications and Ablations	2
Evaluation	2
Report	6
Submission Format	1
Total	**16**