University of Tehran

College of Interdisciplinary Science and Technology

**Course Name:**

Artificial Intelligence Course

**Final Title:**

# Understanding and Fine-Tuning the CLIP Model

**Instructor:**

Dr. Hanieh Naderi

**Teaching Assistant:**

Omid Ostovari

**Submission Deadline:**

July 9, 2025 at 11:59pm

Spring 2025

# TABLE OF CONTENT

# INTRODUCTION

The goal of this project is to familiarize students with the use of **pre-trained deep learning models** and their fine-tuning for different datasets. In this exercise, we focus on **Contrastive Language-Image Pre-Training (CLIP)**, a multimodal model developed by OpenAI (Radford et al., 2021). CLIP learns visual concepts by aligning images with natural language descriptions, enabling zero-shot learning and efficient fine-tuning. This project will guide you through loading the CLIP model, evaluating its baseline performance, and performing various fine-tuning techniques.

- Pay attention to the points mentioned at the end of the file.

# PROJECT BREAKDOWN

## PART 1: LOADING CLIP AND PREPROCESSING THE DATASET

1. **Installing Required Libraries**
   - Install the open_clip_torch package and import necessary modules.
2. **Loading the Dataset**
   - Use the ceyda/fashion-products-small dataset, which consists of labeled fashion product images.
   - Load the dataset and visualize a few sample images.
3. **Exploring Pre-Trained Models**
   - List the available pre-trained CLIP models.
   - Instantiate the convnext_base_w model with pretrained='laion2b_s13b_b82k_augreg'.
   - Print the model structure and preprocessing transforms.
4. **Dataset Preparation**
   - Split the dataset into **train, validation, and test sets**.
   - Implement a **custom PyTorch dataset class** with appropriate augmentations.
   - Create **DataLoaders** for each split.
   - Perform **k-fold cross-validation** and visualize batch samples.
5. **Cosine Similarity Analysis**
   - Select **8 sample images** and generate **custom textual descriptions**.
   - Compute **cosine similarity** between images and descriptions using CLIP's embeddings.
   - Visualize the similarity matrix.

## PART 2: EVALUATING THE BASELINE CLIP MODEL

1. **Modifying the CLIP Model for Image Classification**
    - Freeze all parameters except the final classifier layer.
    - Replace CLIP's last linear layer in the visual encoder with a **custom classifier (1, 2, and 3 linear layers)** and train models and comparing performance
2. **Fine-Tuning Optimization**
    - Train the model using two different optimizers:
        - **Adam** optimizer with CLIP's last two visual layers.
        - **AdamW** optimizer with CLIP's last two visual layers.
    - Compare the performance of both optimizers.
3. **Training and Evaluation**
    - Report **training, validation, and test losses** after each epoch.
    - Use **TensorBoard** for logging results.
    - Save logs for analysis and submission.
4. **Visualization of Predictions**
    - Display **predicted vs. actual labels** for validation images.
    - Compare classification performance with and without fine-tuning.
5. **Using an Alternative CLIP Model**
    - Repeat the steps above using ViT-B-32 with pretrained='openai'.

## PART 3: ADVANCED FINE-TUNING STRATEGIES

1. **Full Model Fine-Tuning**
    - Train all layers of CLIP on the dataset.
    - Report **training, validation, test loss and accuracy**.
    - Analyze the impact of fine-tuning all layers on model performance.
2. **Fine-Tuning Only the Language Model**
    - Freeze the visual model and train only the text encoder.
    - Explain how this method is useful when **labels are in different languages**.
    - Report training loss and accuracy.
    - Compare results with full fine-tuning.
3. **Fine-Tuning Only the Vision Model**
    - Freeze the text model and train only the visual encoder.
    - Explain when this approach is beneficial (e.g., domain-specific images).
    - Report training, validation and test loss and accuracy.
    - Compare results with full fine-tuning.
4. **Fine-Tuning the Last 30% of Layers**
    - Fine-tune only the last layers of both encoders.
    - Report training, validation and test loss and accuracy.
    - Report training, validation and test loss and accuracy.
    - Compare results with full fine-tuning.
5. **Parameter-Efficient Fine-Tuning (PEFT) with LoRA**
    - Implement **Low-Rank Adaptation (LoRA)** from Hugging Face's PEFT library.
    - Fine-tune **only 1-5% of model parameters**.
    - Report training, validation and test loss and accuracy.

- Compare LoRA’s efficiency vs. full fine-tuning.

## CONCLUSION AND REPORT SUBMISSION

1. **Summarizing Results**
    - Compare **zero-shot CLIP, full fine-tuning, and different tuning strategies**.
    - Create a table summarizing model performance for each fine-tuning method.
    - Identify the best-performing model and justify the findings.
2. **Submission Requirements**
    - Submit a structured **report** with logs, results, and code.
    - Include **TensorBoard visualizations**.
    - Discuss challenges encountered and solutions applied.

## ASSESSMENT CRITERIA

**Please read the following points carefully. Failure to adhere to any of them will result in a deduction of your score.**

- **Code Implementation (40%)**: Correctness and efficiency of model training and fine-tuning.
- **Understanding of CLIP (20%)**: Demonstration of understanding through explanations.
- **Analysis & Interpretation (20%)**: Insightful comparisons and justification of results.
- **Report Quality (20%)**: Clear presentation, well-structured content, and proper documentation.

**Important Guidelines:**

- The deadline for submission **will not be extended**.
- Your report is crucial for the evaluation process. Please include all key points and assumptions considered in your implementation and calculations.
- **Do not include code snippets as images** in the report. Instead, present the code in a suitable format (e.g., writing the entire report and explanations in a Jupyter Notebook).
- Ensure that your code contains **necessary comments** and submit all required files for proper execution.
- While a detailed explanation of the code is not required in the report, you must analyze and discuss the obtained results.
- The report must follow the **specified format** uploaded to the course website (eLearn).

- You may use **LaTeX** to write the report if you prefer, but it must conform to the main format and structure.
- Add **captions** for all images and tables in your report.
- If you use code from the internet that is not part of the main task, you must **cite the source** in both the report and the code. Failure to do so will be considered plagiarism, resulting in a **zero score** for the assignment. However, you are free to use online resources for aspects **not explicitly restricted** in the exercise.

• You are **only allowed** to use the Python programming language and the PyTorch library to implement neural networks.
• Any **similarity** in the report and code is considered fraud, and the score for all individuals involved in the fraud will be **zero**.
• Please upload the **code**, **report**, and any other required attachments in the following format on the lesson page in the eLearn system:
**PRJ_[Lastname1]_[StudentNumber1].zip**
• If you have any questions or doubts, please **contact the teaching assistants** at:
omidostovary@gmail.com