COL8628/COL828 Assignment 1
Transformer models for implant classification

Instructions:

- **Deadline: 11.59PM, 30th September, 2025.**
- **Anti-Plagiarism Policy**: Any form of plagiarism will result in an F-grade.
- **Late submission policy**: **No late submissions allowed.**
- Cite your sources properly in the report.
- Use Python 3.9 and PyTorch for experiments.
- Submission: Via Moodle. This assignment is to be done **individually**.

Overview

This assignment introduces you to the task of image classification using Transformer-based models. You will experiment with Vision Transformers[1] pretrained under different paradigms and evaluate how their representations transfer to implant classification using medical images. In addition, you will explore Prompt-tuning techniques and assess their effectiveness for this application. The assignment focuses on two medical imaging datasets: `Pacemakers` and `Orthonet`, both of which consist of X-ray images of implants corresponding to different organs. You are encouraged to familiarize yourself with the characteristics of these datasets before beginning the assignment. The assignment is divided into two parts: Task 1, which involves classification on the `Orthonet` dataset, and Task 2, which involves classification on the `Pacemakers` dataset.

Task 1: Image Classification on Orthonet

This task focuses on evaluating the transfer learning capabilities of Vision Transformers and prompt-learning methods on the **Orthonet** dataset, which consists of X-ray images of different types of orthopedic knee implants. The objective is to compare the effectiveness of various pretraining strategies, explore zero-shot classification, and investigate advanced prompt-learning approaches.

1.1: Fine-tuning Pretrained ViT Variants

In this subtask, you will fine-tune Vision Transformer (ViT) models that have been pretrained using different strategies. Rather than pretraining the models from scratch, you will leverage publicly available pretrained weights and adapt them to the implant classification task on the Orthonet dataset. Specifically, you will experiment with three variants of pretrained weights for the ViT:

a. standard ImageNet-21k pretrained weights,

b. CLIP weights [4], and

c. DINOv2 weights [3].

For each variant, initialize the ViT with the respective pretrained weights and fine-tune it on the Orthonet dataset. You will also need to modify the final MLP classification head of the ViT so that the output dimensionality matches the number of classes in the dataset. After training, evaluate each model and compare their performance in terms of **Top-1 Accuracy**, **F1-score**, and **AUC-ROC** on the test set.

1.2: Zero-Shot Classification with CLIP

In this subtask, you will evaluate the zero-shot classification capability of the CLIP model [4]. Rather than training a dedicated classifier, this task relies on the image–text alignment learned during CLIP pretraining. For each class in the Orthonet dataset, you should design simple textual prompt templates (e.g., `‘‘an X-ray of a {class} implant’’`) and compute the corresponding text embeddings. Each image will then be classified by comparing its embedding with all class template embeddings and assigning it to the class with the highest cosine similarity. The publicly available weights for CLIP can be utilised for this subtask. You are encouraged to read CLIP paper [4] to familiarize yourself with the zero-shot capabilites of CLIP.

The performance of this zero-shot pipeline should be evaluated using the same metrics as in Subtask 1.1: **Top-1 Accuracy**, **F1-score**, and **AUC-ROC**. Finally, compare these results with those obtained from fine-tuning in Subtask 1.1. This analysis will help illustrate the trade-offs between task-specific adaptation through fine-tuning and the generalization ability of large-scale pretrained vision–language models for medical imaging tasks.

1.3: Prompt Learning Methods on CLIP

While handcrafted templates in CLIP are simple to design, they are often limited in expressiveness and adaptability. Prompt learning methods address these limitations by introducing learnable, task-specific prompt embeddings, thereby enabling adaptation to the target dataset without requiring full model fine-tuning. In this subtask, you will experiment with three such methods: CoOp [7], CoCoOp [6], and MaPLe [2]. Each method progressively increases the flexibility of prompt design and the degree of adaptation to the downstream task. You are encouraged to read the original papers for implementation details. Throughout this subtask, you will use the pretrained CLIP as your base model.

CoOp (Context Optimization). CoOp [7] learns continuous context tokens that either replace or augment handcrafted prompts. These tokens will be optimized using the Orthonet training data while keeping the CLIP backbone frozen. The learned prompts allow the model to adapt to the distribution of medical images without requiring end-to-end fine-tuning. Evaluate CoOp on the Orthonet dataset and report performance in terms of **Top-1 Accuracy**, **F1-score**, and **AUC-ROC** on the test set.

CoCoOp (Conditional CoOp). CoCoOp [6] extends CoOp by conditioning prompt embeddings on image features, thereby introducing dynamic adaptation to the input. This enables the model to better handle class imbalance and generalize to unseen categories. Train CoCoOp on the Orthonet dataset and evaluate its performance, comparing it against both CoOp and the zero-shot baseline. Report **Top-1 Accuracy**, **F1-score**, and **AUC-ROC** on the test set.

MaPLe (Multi-modal Prompt Learning). MaPLe [2] further enhances prompt learning by distributing learnable prompts across both the image and text encoders, as well as at multiple transformer layers. This multi-level adaptation promotes stronger alignment between modalities and improves performance on both base and novel categories. Train MaPLe on the Orthonet dataset and evaluate its performance, comparing results against CoOp, CoCoOp, and the zero-shot baseline. Report **Top-1 Accuracy**, **F1-score**, and **AUC-ROC** on the test set.

Task 2: Image Classification on Pacemakers

The second task focuses on the **Pacemaker** dataset, which contains chest radiographs with implanted cardiac devices spanning multiple manufacturers and models. Unlike Task 1, this problem is more fine-grained, as each device is annotated according to a hierarchical taxonomy consisting of three levels: *manufacturer*, *series*, and *model*. The objective of this task is to investigate how pretrained Vision Transformers and prompt-learning methods perform in such fine-grained and hierarchically structured classification settings.

2.1: Fine-tuning Pretrained ViT Variants

As in Task 1.1, begin by loading a Vision Transformer (ViT) pretrained on **ImageNet-21k** [5] and fine-tune it using the Pacemaker training set. Evaluate the model in terms of **Top-1 Accuracy**, **Top-3 Accuracy**, **F1-score**, and **AUC-ROC**. Since the Pacemaker dataset contains 45 classes, where each implant belongs to one of five manufacturers, reporting **Top-3 Accuracy** provides additional insight into whether misclassifications occur within the same manufacturer group.

Repeat the same procedure with ViT backbones pretrained using **CLIP** [4] and **DINOv2** [3], and compare their performance on the test set.

2.2: Zero-Shot Classification with CLIP

In this subtask, you will evaluate the zero-shot classification capability of CLIP on the Pacemaker dataset. Following the approach in Task 1.2, design textual prompts for each of the 45 implant classes (e.g., `‘‘an image of a {class} implant’’`) and use them to obtain text embeddings. Each test image will then be classified by computing the cosine similarity between its embedding and the candidate text embeddings, and assigning it to the most similar class.

Evaluate this zero-shot pipeline by reporting **Top-1 Accuracy**, **Top-3 Accuracy**, **F1-score**, and **AUC-ROC**. Finally, compare these results against the fine-tuned baselines from Subtask 2.1.

2.3: Hierarchical Prompt Learning

Moving beyond handcrafted prompts, we now explore **hierarchical prompt learning** inspired by fine-grained image classification. The `Pacemaker` dataset contains 45 implant classes, which can be grouped into 5 manufacturers as follows: `BIO, BOS, MDT, SOR, STJ`. The goal is to leverage this hierarchical structure when adapting prompt learning techniques.

1. **Stage 1: Manufacturer-level classification.** First, collapse the 45 fine-grained classes into 5 manufacturer-level categories. Train prompt learning models using pretrained CLIP with the following methods:
    - CoOp [7]
    - CoCoOp [6]
    - MaPLe [2]

Evaluate classification performance on the 5-class manufacturer task.

2. **Stage 2: Fine-grained classification.** Next, expand back to the original 45 implant classes. Initialize the prompt embeddings with the learned manufacturer-level embeddings from Stage 1, and then finetune them for the 45-class classification task. This ensures that the learned prompts encode coarse manufacturer distinctions while adapting to finer model-level differences.

3. **Analysis.** Repeat Stage-2 for all three methods (CoOp, CoCoOp, MaPLe). Compare their performance and analyze the effect of hierarchical initialization on the classification accuracy. Check Whether hierarchical prompts lead to better generalization than directly learning flat prompts and compare the relative advantages of CoOp, CoCoOp, and MaPLe in this hierarchical setting. Report the top-1 accuracy, top-3 accuracy, F1 score and AUC ROC for comparison.

You will be using the pretrained CLIP model as the backbone throughout this task. Please note that for the Prompt-learning experiments in Tasks 1 and 2, the image encoder and the text encoder remain frozen at all times and only the prompt embeddings are trained.

Note: In Step 1, you may choose to learn a single prompt for the five class classification problem for using it further in Stage 2. For Stage 2, you may learn a single prompt for the 45-class prompt or class-specific prompts for all the three methods. You may justify your chosen design choice in the report.

Submission Format and Instructions

1. **Code submission.** For each subtask, submit two scripts named in the format: `train_subtask_<No.>.py` and `test_subtask_<No.>.py`.
   - For Subtask 1.1 and 2.1, a single script for training all three models and a single script for testing all three models sequentially needs to be submitted.
   - For Subtask 1.2 and Subtask 2.2, only the testing script needs to be submitted.
   - Each testing script should evaluate the model on the test set and report the scores for all the metrics specified in the respective subtask.

2. **Environment.** Submit a `requirements.txt` file for setting up the environment.

3. **Documentation.** Provide a well-written `README.md` that contains clear instructions on how to run your code for each subtask. You are advised NOT to hard code paths in your code and instead use command line arguments to pass the required paths. Please mention detailed instructions in your `README.md` file on how to run each of your scripts in detail.

4. **Model weights.** If required, upload the trained model weights to OneDrive and include the link in your report, if it exceeds the ZIP file limit of 25MB for Moodle. **Note:** The timestamp of the last update for the weights must be strictly before the submission deadline.

5. **Use of external repositories.** You may use publicly available repositories for ViT, CLIP, CoOp, CoCoOp, and MaPLe. Any other external resource must be justified and appropriately cited in the report.

6. **Final submission.** Submit a single zip file on **Moodle** named `<EntryNumber>.zip`, containing:
   - all training and testing scripts (`.py` files),
   - `report.pdf`,

- `requirements.txt`,
- `README.md`.

Do **not** include the datasets in your submission.

Evaluation Rubrics

- **Task 1: 45 marks**
  - Subtask 1.1: 15 marks
  - Subtask 1.2: 10 marks
  - Subtask 1.3: 20 marks
- **Task 2: 45 marks**
  - Subtask 2.1: 15 marks
  - Subtask 2.2: 10 marks
  - Subtask 2.3: 20 marks
- **Report: 10 marks**

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[2] Muhammad Uzair Khattak, Imran Ahmed, Salman H Khan, Fahad Shahbaz Khan, and Bernard Ghanem. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[3] Maxime Oquab, Timothée Darcet, Theo Moutakanni, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[5] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

[6] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[7] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 2022.