Deep Learning 2025 - Project Assignment

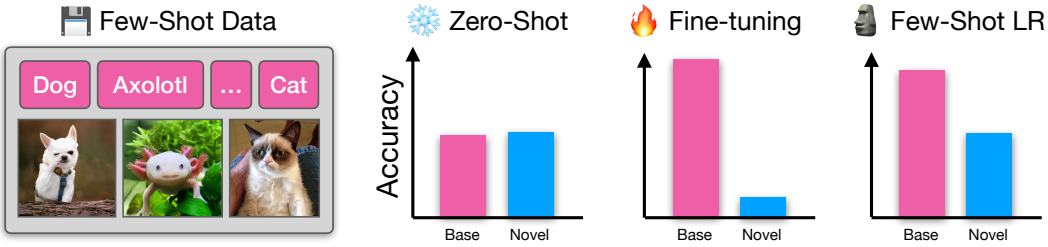


Figure 1: Comparison of Zero-Shot, Fine-Tuning, and Few-Shot Learning. Zero-Shot learning performs poorly on both Base and Novel classes of a fine-grained task, as the model lacks exposure to those data. Fine-Tuning improves Base class accuracy but fails to generalize well, worsening performance on Novel classes. Few-Shot Learning efficiently fine-tunes on limited few-shot data, improving accuracy on Base classes while preserving Novel class performance.

1 Introduction

In traditional deep learning, models achieve impressive performance when trained on large datasets. However, there are scenarios where data is scarce, expensive to obtain, or highly specialized – such as medical imaging, and bird species recognition. Few-Shot Adaptation tackles this challenge by improving generalization when only a *few* examples are available. Few-Shot Adaptation aims to mimic human cognitive abilities, such as the ability to grasp new concepts quickly after seeing just a few instances. For example, a child can recognize a zebra after seeing only a couple of pictures, even if they've never seen one before. Similarly, few-shot learning algorithms strive to leverage prior knowledge, learned from a large base dataset, to adapt to new tasks with minimal data.

Vision-Language Models (VLMs) are a class of models designed to process and integrate visual and textual information. By combining natural language processing with computer vision, VLMs can perform tasks such as image captioning, visual question answering, and text-to-image generation. Among the first VLM, Contrastive Language–Image Pre-training (CLIP) has stood the test of time, with its astonishing zero-shot performance, enabling it to recognize and classify images without explicit task-specific training [10]. As a demonstration of its value, many works throughout the years have built on top of it [2, 8, 6].

Despite CLIP's impressive zero-shot capabilities, challenges arise in fine-grained classification tasks that require distinguishing subtle differences between similar categories. In such scenarios, zero-shot approaches may not suffice, necessitating methods that allow models to adapt to specific tasks using limited labeled examples. For example, CLIP-Adapter [4] fine-tunes lightweight adapter modules, within the pre-trained CLIP model, to enhance its few-shot learning capabilities. By adding a bottleneck layer, CLIP-Adapter learns new features and performs residual-style feature blending with the original pre-trained features. This approach allows for efficient adaptation to new tasks using minimal data. In contrast, CoOp [13] replaces the default prompt template of CLIP, *i.e.*, `a photo of`, with learnable vectors, optimizing the label embeddings rather than the image embeddings.

Current research in Few-Shot Adaptation explores promising solutions, yet the challenge of generalization with data scarcity persists. While progress has been made, fundamental questions remain, keeping few-shot learning a vital and evolving area of study.

2 Few-Shot Adaptation

Task formalization. Few-Shot Adaptation adapts a machine learning model f_θ on a downstream task given a limited set of annotated samples per category, called *shots*. Let $\mathcal{D} = \{(\mathbf{x}, y)_i\}_{i=1}^{n}$ be a

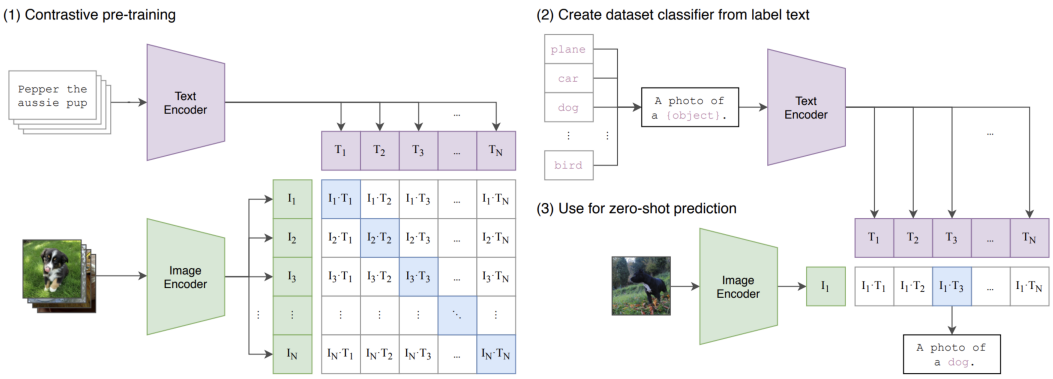


Figure 2: **CLIP [10] pre-training and zero-shot prediction.** After large-scale, contrastive, multimodal pre-training, CLIP can be used as a zero-shot classifier. By specifying class names in natural language, we can exploit CLIP's text encoder to represent categories in a d-dimensional Euclidean space shared with the image encoder. Embedded text representations, then, act as a classifier, thus, the class with the highest similarity with the image embedding is the candidate prediction.

few-shot annotated dataset, where $\mathbf{x}_i \in \mathcal{X}$ is an image, $y_i \in \mathcal{B}$ is the target label, and $\mathcal{X}$ and $\mathcal{B}$ are respectively the image domain and the set of Base categories. The number of shots (*i.e.*, available samples for each category) $k \in \mathbb{N}$ is constant across all categories. Therefore, the dataset size is given by $n = k \cdot |\mathcal{B}|$. The downstream task spans, however, a set of $\mathcal{C}$ categories, which is a superset of the annotated classes, $\mathcal{B} \subset \mathcal{C}$. The remaining, non-annotated, set of classes $\mathcal{N} := \mathcal{C} \setminus \mathcal{B}$ is referred to as Novel categories. The final objective is to improve f_θ accuracy on the Base categories, using $\mathcal{D}$, while preserving (or even improving) the model's original performance on the Novel ones [12, 3], also known as *base-to-novel* generalization.[1] In other terms, you are given a few annotated samples for classes in $\mathcal{B}$ to adapt the pre-trained model on such Base categories while maintaining the network's original performance (*i.e.*, zero-shot accuracy) on the Novel categories $\mathcal{N}$.

Model architecture. For this project assignment, we assume f_θ to be a **pre-trained** contrastive vision-language model (*e.g.*, CLIP [10]), composed of a vision encoder $f_\theta^v : \mathcal{X} \to \mathbb{R}^d$ and a text encoder $f_\theta^t : \mathcal{T} \to \mathbb{R}^d$, mapping images and texts in a shared d-dimensional Euclidean space. By specifying classes in natural language (typically via a prompt like `a photo of {class_name}`), the pre-trained vision-language model (VLM) can be used to classify images in a zero-shot manner:

$$f_\theta(\mathbf{x}_i, \mathcal{C}) = \arg\max_{c \in \mathcal{C}} \left\{ \langle f_\theta^v(\mathbf{x}_i), f_\theta^t(c) \rangle \right\}, \tag{1}$$

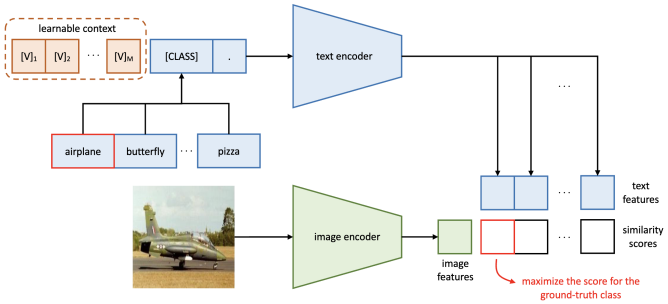
where $\langle \cdot, \cdot \rangle$ is the cosine similarity. Figure 2 visualizes how to compute zero-shot classification with CLIP [10].

Parameter-Efficient Fine-Tuning. Contrastive VLMs usually count millions if not billions of parameters. As the number of parameters N is much larger than the annotated dataset size (*i.e.*, $N \gg n$), there is a high risk of overfitting during adaptation. Therefore, we suggest using Parameter-Efficient Fine-Tuning [13, 5, 1, 7]. In a nutshell, Parameter-Efficient Fine-Tuning techniques (PEFTs) aim at adapting pre-trained neural networks by either tuning a small subset of parameters (usually $< 5\%$) or by conditioning the model via trainable tokens. Notable examples are prompt-tuning [13], which prepends learnable tokens as input to the text encoder (in the VLM case), and LoRA [5], which applies low-rank decomposition to weight updates.

2.1 Example: CoOp and CoCoOp.

We now introduce CoOp [13] and CoCoOp [12] two seminal works on VLM adaptation that can help you to understand this assignment desideratum. While the former (CoOp) shows an initial tentative of adapting the VLM to Base classes, it fails to generalize to Novel categories of the same dataset. In contrast, CoCoOp shows a simple trick to enable CoOp to generalize to Novel categories, thus, satisfying this assignment's objective.

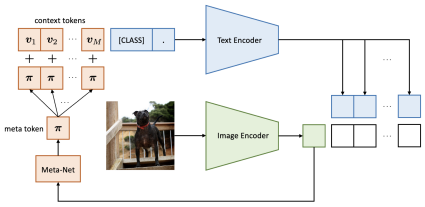
[1] Throughout the literature you might find a slightly different scenario called *all-to-all* Few-Shot Adaptation where $\mathcal{N} := \varnothing$ and $\mathcal{B} = \mathcal{C}$. Feel free to read those papers as well, but keep in mind that your goal is different!



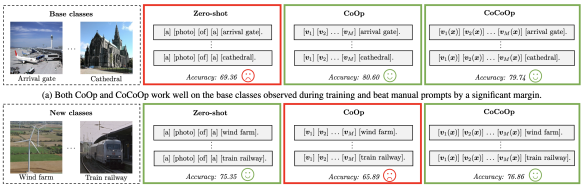
(a) Prompts are substituted by learnable contexts.

(b) Substituting prompt engineering with the learnable context automates the prompt search achieving better results

Figure 3: **CoOp [13] overview.**



(a) Meta-Net generates input-conditional tokens for each image.



(b) Although CoOp [13] improves over zero-shot clip on Base classes, it struggles on Novel ones. CoCoOp [12] solves this problem by conditioning learnable prompts via image-tailored tokens.

Figure 4: **CoCoOp [12] overview.**

CoOp. To avoid searching for the prompt that maximizes CLIP [10] zero-shot performance (*i.e.*, finding a better prompt than `a photo of {class_name}`), CoOp [13] automatizes the process by learning a set of context vectors (*i.e.*, prompts) using few annotated samples. Formally, let $P \in \mathbb{R}^{M \times d}$ be the set of learnable tokens. Then the input to the text encoder for class `class_name` becomes $c = [P_1, ..., P_M, \texttt{class\_name}]$. By forwarding images and corresponding text prompts through the pre-trained VLM, CoOp [13] tunes P by computing the cross-entropy loss:

$$\mathcal{L}_{\text{CE}}(\mathbf{x}_i, y_i) = -\log \frac{\exp(\langle f_\theta^v(\mathbf{x}_i), f_\theta^t(y_i) \rangle)}{\sum_{c \in \mathcal{C}} \exp(\langle f_\theta^v(\mathbf{x}_i), f_\theta^t(c) \rangle)}, \tag{2}$$

where both y_i and c are built by pretending the learnable context to the category name (Fig. 3a visualizes this computation). As Fig. 3b shows, substituting handcrafted prompts easily adapts the pre-trained VLM to a specified dataset. Yet, CoOp [13] cannot generalize to Novel classes of the same dataset, caused by overfitting of Base classes during adaptation.

CoCoOp. To address this problem, CoCoOp [12] improves over its predecessor by combining the context vectors with an image-conditioned token, which shifts the focus away from a specific set of classes to a specific input instance, reducing overfitting. To generate the image-conditioned token, CoCoOp [12] trains a lightweight MLP called Meta-Net alongside learnable prompts (see Fig. 4a). Formally, let h_φ be the Meta-Net, then each conditional token is obtained as $P_m(\mathbf{x}_i) = P_m + h_\varphi(\mathbf{x}_i)$, where P_m is the m-th learnable context vector in the sequence and $P_m(\mathbf{x}_i)$ is the m-th conditional token.[2] Both the Meta-Net and the context vectors are trained end-to-end using the cross-entropy loss as in Eq. (2). As Fig. 4b shows, combining context vectors with an input-conditioned token, has the joint benefit of adapting the VLM to Base classes while keeping high accuracies for Novel ones.

3 Assignment

Project Objective. We request that you develop a Few-Shot Adaptation method that adapts the model to Base categories $\mathcal{B}$ using the provided few-shot annotated dataset $\mathcal{D}$ while preserving the original network's zero-shot accuracy (or even improve it) on Novel classes $\mathcal{N}$.

[2]Note that to standardize CoOp and CoCoOp notations we used P instead of $[\text{V}]$ and v for prompts.

Model	Base (↑)	Novel (↑)	Harmonic Mean (↑)
RN-50	69.43	73.80	71.55
ViT-B/16	71.33	78.24	74.62

Table 1: **Our reproduced CLIP zero-shot accuracy on Oxford Flowers [9].** Minor discrepancies with published papers are due to different train-val-test splits.

Suggestions. We suggest using CLIP ViT-B/16 or RN-50 as they are the most used models for Few-Shot Adaptation. Nevertheless, as long as pre-trained weights are available and you re-compute zero-shot baselines performance, you can use any available model you like (*e.g.*, SigLIP [11]).

Additionally, we suggest you use an existing work in the literature as a starting point. We encourage you to propose, implement, and evaluate modifications to the original method you choose to demonstrate improvements in accuracy or efficiency terms. Remember that you are constrained to only use the few-shot samples provided. Therefore, you cannot use more than the given shots or train on Novel categories (these are assumed to be unknown at train time). Of course, you must use the entire validation set (Base + Novel categories) to evaluate your model.

3.1 Datasets and evaluation metrics

For this year's assignment, you will evaluate your proposed method on Oxford Flowers [9]. You can download, load, and split the dataset using the provided code.[3] As Oxford Flowers [9] counts ten images per class in the training set, **the number of available shots is set to 10**, that is, $\mathbf{k = 10}$.

Oxford Flowers [9] is a vision dataset consisting of 102 flower categories. The flowers are chosen from those commonly occurring in the United Kingdom. Each class consists of between 40 and 258 images. The images have large scale, pose, and light variations. In addition, some categories have large variations within the category and several very similar categories.

Your method's performance should be separately measured with the top-1 accuracy (*i.e.*, the ratio of samples for which the predicted label matches the target label) on the Base categories, the top-1 accuracy on the Novel categories, and the harmonic mean of the two. Table 1 shows CLIP RN-50 and ViT-B/16 zero-shot performance on Oxford Flowers [9]. If you decide to use one of these models, feel free to reuse these numbers as a zero-shot baseline for your method, otherwise, you are strongly encouraged to recompute them with the model of choice, which will serve as a baseline.

4 Project evaluation

We expect the project to be self-contained within a Jupyter Notebook hosted on Google Colab. The notebook must contain code and text cells, which are meant to integrate the code with comments, and they should form the final project report. The report must include:

- A thorough description of the solution you adopted. You must include a detailed overview of the architecture, the losses, and the overall pipeline. Make sure to highlight your original contributions and cite the literature if needed. It is important you carefully motivate each design decision you take, particularly when their motivation is nontrivial. You are also encouraged to enhance the description by incorporating images within text cells;
- An extensive presentation and discussion of the results obtained with your solution. Organize the scores obtained in tables, including charts depicting learning curves, confusion matrices, and other useful visual representations you believe are relevant. Structure the discussion in such a way that emphasizes the take-home message of your findings;
- **Add comments to the code**. Comments must be clear, concise, and nontrivial, proving that you are aware of all the steps that you go through during the development and evaluation of the code;

[3] `https://drive.google.com/file/d/15kAm36VMrqnV1JqGqgH3PGvP2Eijy-gu/view?usp=sharing`

- We expect to run the notebook from the first to the last code cell without any additional steps required and without errors or crashes. Should your code require extra steps to run, please include them in the report.

Your project will be evaluated according to the following metrics [4]:

- originality of the solution;
- methodological thoroughness;
- clarity of the report;
- performance on validation accuracy;
- quality of the code.

Note that a well-motivated and original idea that is also clearly described will likely result in a higher grade than a solution that beats the state-of-the-art but adds minor, poorly motivated, contributions. Therefore, as a general suggestion, be creative and "*Impress us!*".

In order to deliver the solution, send an email with object `[DL2025 - Project Delivery] {student id 1} {student id 2} {student id 3}`, with your self-contained notebook in `.ipynb` format attached to it.[5] The notebook name must follow the following format: `{student id 1}_{student id 2}_{student id 3}.ipynb`. **IMPORTANT**: make sure that the file can be seamlessly loaded and executed into Google Colab, any exception might result in a lower grade. **The solution must be delivered no later than 7 days before the day you decide to take the oral exam** (around 15 to 7 days before the oral exam would be best). **This project assignment has validity until the end of February 2026** (*i.e.*, the fifth exam session). If you do not submit this year's assignment within seven days before the last exam session, you must refer to the next year's project assignment.

5 Group Registration

The project is intended for groups of 2 or 3 people. You can register your group by filling out the Google Form.[6] **The registration deadline is *April 20th***.

6 Policies

We require you **not to start from an existing GitHub repository** containing code developed by others. Of course, you can install common libraries such as `matplotlib`, `scikit-learn`, `pillow`, and many more. You are allowed to choose any existing contrastive vision-language model architecture as your backbone as long as pre-trained weights are available. It is strictly forbidden to share the source code with other groups. Also, keep in mind that your code, along with the project report, will be checked with plagiarism tools. **Any violation of these policies will result in actions being taken toward all group participants**: it is each group's responsibility to comply with the rules.

References

[1] Samyadeep Basu, Shell Hu, Daniela Massiceti, and Soheil Feizi. Strong baselines for parameter-efficient few-shot fine-tuning. In *AAAI*, 2024.

[2] Alessandro Conti, Enrico Fini, Massimiliano Mancini, Paolo Rota, Yiming Wang, and Elisa Ricci. Vocabulary-free image classification. *NeurIPS*, 2023.

[3] Matteo Farina, Massimiliano Mancini, Giovanni Iacca, and Elisa Ricci. Rethinking few-shot adaptation of vision-language models in two stages. In *CVPR*, 2025.

[4] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *IJCV*, 2024.

[4] The metrics have not equal importance. For example, code quality is of *slightly* less importance than the clarity of the report. However, remember that the maximum score is obtained by maximizing all the metrics.

[5] Email address: `deep-learning-2025-rnoos72391n9-unverified@unitn.it`

[6] `https://forms.gle/GysH7yYj1RYeoqtA6`

[5] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 2022.

[6] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

[7] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *NeurIPS*, 2022.

[8] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *NeurIPS*, 2023.

[9] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[11] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023.

[12] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, 2022.

[13] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 2022.