COMP 3105A Introduction to Machine Learning
Assignment 4

Instructor: Junfeng Wen (junfeng.wen [AT] carleton.ca)

Fall 2025
School of Computer Science
Carleton University

Deadline: 11:59 pm, Sunday, Nov. 30, 2025

Instruction: **Submit the following three files** to Brightspace for marking

- A Python file `A4codes.py` that includes all your implementations. **Do not include any top-level code in this file** other than the required functions.
- A pip requirements file named `requirements.txt` that specifies the running environment including a list of Python libraries/packages and their versions required to run your codes.
- A PDF file `A4report.pdf` that includes all your answers to the written questions. It should also specify your team members (names and student IDs). Please clearly specify question/sub-question numbers in your submitted PDF report so TAs can see which question you are answering.

Do not submit a compressed file, or it will result in a mark deduction. We recommend trying your code using Colab or Anaconda/Virtualenv before submission.

Rubrics: This assignment is worth 15% of the final grade. Your codes and report will be evaluated based on their scientific qualities including but not limited to: Are the implementations correct? Is the analysis rigorous and thorough? Are the codes easily understandable (with comments)? Is the report well-organized and clear?

Policies:

- You can finish this assignment in groups of two. All members of a group will receive the same mark when the workload is shared.
- You can resubmit your work on Brightspace and the old submissions will be overwritten. However, please **submit all required files** (as opposed to only submitting the file that you want to overwrite) on Brightspace as we will only evaluate your latest submission.
- You may consult others (classmates/TAs) about general ideas but don't share codes/answers. Please specify in the PDF file any individuals you consult for the assignment. Any student found to cheat or violate this policy will receive a score of 0 for this assignment.
- For this assignment only, you can use large language models (LLMs) as much as you want to without providing chat history.
- Remember that you have **three** excused days *throughout the term* (rounded up to the nearest day), after which no late submission will be accepted.

- Specifically for this assignment, you can use **any** Python libraries you want, as long as the program can complete within a reasonable time and space limit. **However, you must not use any additional/external data sources**.

Question 0: The Task

The goal of this assignment is to address a real-world machine learning problem.

Suppose that we have gathered an `in-domain` image dataset that consists of everyday objects (see a few examples in Fig. 1). We would like to build a classifier that can predict the object within the image as accurately as possible. However, we do not want our classifier to be misused or abused by others. This means that we want our classifier's performance on the out-of-domain (`out-domain`) data (see Fig. 2) to be **poor** instead. Assuming the datasets are balanced (i.e., each class has approximately the same amount of data), the closer the classifier is to random guessing, the poorer it performs.

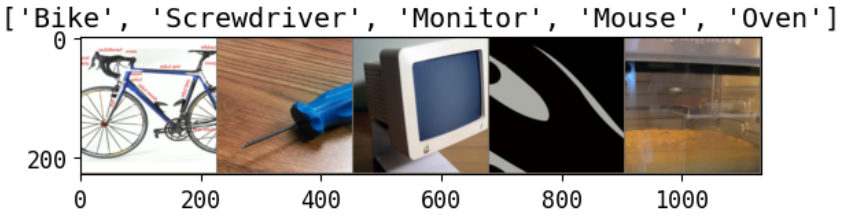

Figure 1: `In-domain` images

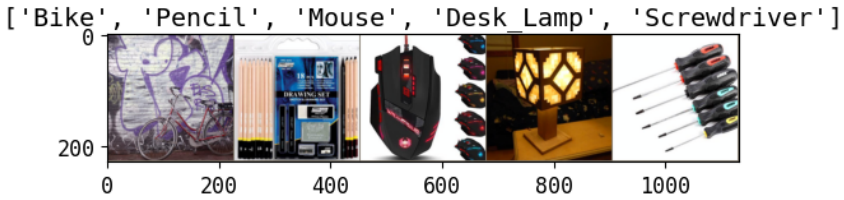

Figure 2: `Out-domain` images

In this assignment, you are given a subset of data from the Office-Home dataset:

	Training	Evaluation
`in-domain`	Labelled	Labelled
`out-domain`	Unlabelled	Labelled

Table 1: Datasets given in this assignment

where the training data can be used to train the classifier, while the evaluation sets are only used for your own evaluation. All images are from ten fixed classes, including bike, mouse, etc.

You can use any algorithm (even those that have not been taught in this course). You can even try ensemble methods (combining multiple models) or apply any pre-processing steps before training. You may also try different regularizations or controlling model complexities.

The constraints are

1. The implementation must be your own, although you can use **any** Python library, including ML libraries such as scikit-learn, PyTorch and TensorFlow. It is highly recommended to test your code in Google Colab or Anaconda environment. Marks will be deducted if we cannot run your code.
2. Your program can run within a reasonable time and space limit. You should aim for a running time that is within 30 minutes when training in Google Colab and this is a soft constraint.
3. **You must not use any additional/external data sources (including the original Office-Home dataset).**

Note: If you want to try out different deep learning models, Colab offers free GPU access that may speed up your method (Runtime → Change runtime type)

In the zip file, we provide the following folders

- `in-domain-train`: labelled `in-domain` training data;
- `out-domain-train`: unlabelled `out-domain` training data;
- `in-domain-eval`: labelled `in-domain` evaluation data;
- `out-domain-eval`: labelled `out-domain` evaluation data.

The sub-folders' names correspond to the class labels of those images. The folder structure will be similar to the new training and evaluation data used for evaluating your method (see Question 3 for more detail). You should only use the training datasets for training and the evaluation datasets to check the generalization capability of your model (as this will be how we call your functions).

Be creative :)

Question 1 (5%) Learning and Classifying

Implement a Python function

```
model = learn(path_to_in_domain, path_to_out_domain)
```

that takes the path to the training `in-domain` data folder and the path to the training `out-domain` data folder, and returns your `model` of any type, as long as it can be used to classify new data (see next). Your functions shouldn't print additional information to the standard output.

Note that the images have different sizes, but you can always standardize/transform them in your preprocessing steps (e.g., using PyTorch transforms).

Implement a Python function

```
accuracy = compute_accuracy(path_to_eval_folder, model)
```

that takes the path to an evaluation data folder (can be `in-domain` or `out-domain`) and a `model` learned by your algorithm, and returns a scalar that is the accuracy of the model on the given data.

Hint: To start, it may be a good idea to train a classifier solely from the `in-domain` data, and make sure that the model achieves a reasonable training accuracy. Then, you can consider how to utilize the `out-domain` data so that the model will perform poorly for them. For example, PyTorch provides an example of loading images and learn a classifier here (although for a different problem).

Note: Please do not attempt to "hack" the accuracy without learning a prediction model. Anyone found doing so will receive a grade of zero for this assignment.

Question 2 (5%) Explanation

In the PDF file, explain your learning algorithm and more importantly, why it is designed this way. The report should be at least 2 pages, but no longer than 8 pages in length.

Your report will be evaluated mainly on how well you justify your design choices. In general, each modelling step should be justified by reasons and/or empirical evidence. For example, if you tried several different algorithms, how did you decide to submit one algorithm over the others? The same applies to other modelling strategies such as optimization schemes, hyper-parameter selection, pre-processing and/or post-processing steps. One possible supporting evidence might be tables of accuracies with different algorithm configurations. Note that these are only guidelines and you do not have to follow them strictly. Feel free to discuss any findings that you feel are interesting or relevant. We will also take into account the overall quality of your report.

Question 3 (5%) Evaluation

For this question, you don't need to submit/write an answer. We will apply your learning method to **different** training sets and evaluate the performance of the learned model on hold-out test datasets:

```
path_to_in_domain_train = ...
path_to_out_domain_train = ...
path_to_in_domain_eval = ...
path_to_out_domain_eval = ...

model = learn(path_to_in_domain_train, path_to_out_domain_train)
in_domain_accuracy = compute_accuracy(path_to_in_domain_eval, model)
out_domain_accuracy = compute_accuracy(path_to_out_domain_eval, model)
```

These new data will follow the same folder structure as in Question 1 with ten classes, but they may not be the same classes as the handout data, so your algorithm should be general. The main criteria to evaluate your method are

- How high the `in-domain` accuracy is
- How close to 0.1 the `out-domain` accuracy is

Your method will be also evaluated based on the following minor criteria

- Time-complexity: the overall run-time of your algorithm
- Space-complexity: whether your algorithm can finish within a reasonable memory limit