CAB320 Assignment 2 – Machine Learning

Adapted by T. Peynot and A. Vanderkop (2024-2025) from a document originally developed by F. Maire

Key Information and instructions

- The submission for this assignment is **due on Week 13 (Friday 30 May)**
- This is a group assignment with an individual component
- Group size: Maximum of 3 people per group (recommended size). 2 person-groups accepted but completion of the same tasks required.
- You need to register your group on Canvas much before the deadline
- Submit your work via Canvas (one submission per group + one report submission per individual) by the deadline
- Make sure to list all the members of your group in the report and the code

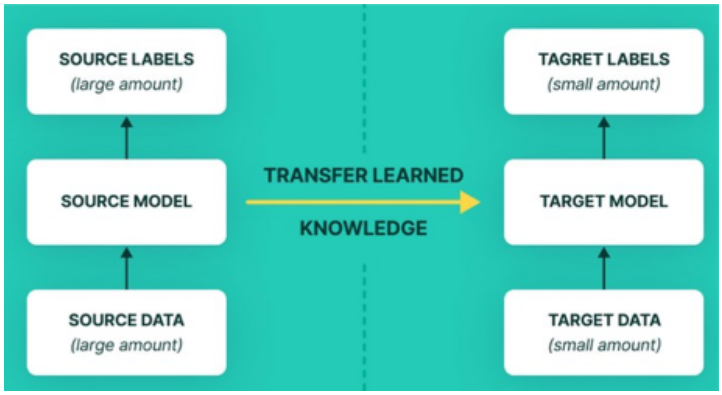

Figure 1: Illustration of the Transfer Learning process

Assignment Overview

Your task is to use machine learning to train a classifier capable of distinguishing different classes of flowers in an image, and to evaluate this classifier's performance. For this we will be using deep learning and a technique called Transfer Learning, which (in this case) consists in reusing a model that was previously trained on a first (usually large) dataset named the *source dataset*, as the starting point for training a model on a second (usually smaller but more specific) dataset named the *target dataset* (see Fig. 1). The source dataset may be a very large dataset such as ImageNet (see Fig. 2) and the target dataset a much smaller one that is directly relevant to a new, or more specific, application domain. In this assignment the aim is to build a flower classifier using transfer learning on a neural network initially trained on the ImageNet dataset.

Figure 2: Illustration of our "Source dataset": ImageNet (left) and our "Target dataset" containing only the specific varieties of flower we want to classify.

Background Theory

Transfer Learning

In practice, in many cases people do not train an entire Convolutional Neural Networks (CNN) from scratch (with weights initialized randomly), because it can be relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a CNN on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the CNN either as an initialisation or a fixed feature extractor for the task of interest. In other words, transfer learning is usually done for tasks where your dataset has too little data to train a full-scale model from scratch.

The most common incarnation of transfer learning in the context of deep learning follows the workflow below:

- Take a slice of layers from a previously trained model.
- Freeze their weights, so as to avoid destroying any of the information they contain during future training rounds on your new dataset.
- Add some new, trainable layers on top of the frozen layers. They will try and learn to turn the old features into predictions on a new dataset.
- Train the new layers on your new dataset.

A last, optional step is *fine-tuning*, which consists of unfreezing the entire model you obtained above (or part of it), and re-training it on the new data with a very low learning rate. This can potentially achieve meaningful improvements, by incrementally adapting the pretrained features to the new data. However, **fine-tuning is not required for this assignment.**

Approach

In a convolutional neural network, lower convolutional layers (i.e. those closest to the input layer) capture low-level image features, e.g. edges or corners, while higher convolutional layers have a larger receptive field and capture more and more complex details, such as body parts, faces, and other compositional features.
The final fully-connected layers are generally assumed to capture information that is relevant for solving common computer vision tasks. For example, the fully-connected layers of the *MobileNet* network can be interpreted as generic computer vision features that are relevant to classify an image into one of the 1000 object categories of the ImageNet dataset.
For this task, we choose *MobileNetV2* as the neural network model architecture because it is lightweight. MobileNetV2 has "only" 3.5 million parameters! It is the smallest of the available pretrained networks in Keras2 (see table of available models at: https://keras.io/api/applications/).
When you perform transfer learning without fine-tuning, you can "accelerate" the training process of the last layers by creating an auxiliary dataset the following way: let us call **F(x)** the function computed

by the frozen layers of the neural network, and let us call **N(x)** the function implemented by the new layers. Your new network implements the function **N(F(x))**. That is, **F** composed with **N**. Assume our dataset is **{(x1,t1), (x2,t2),..., (xm,tm)}**. Standard transfer learning sets the learning rate of the weights of **F** to zero and applies the backpropagation algorithm to the network **x→N(F(x))**. By precomputing the activation arrays **{F(x1),F(x2),...,F(xm)}**, we can create a dataset **{(F(x1),t1), (F(x2),t2),..., (F(xm),tm)}** for the network **z→N(z)**. The gain is that we don't have to recompute the activations **F(x)** when we apply the backpropagation algorithm directly to the network **z→N(z)**.

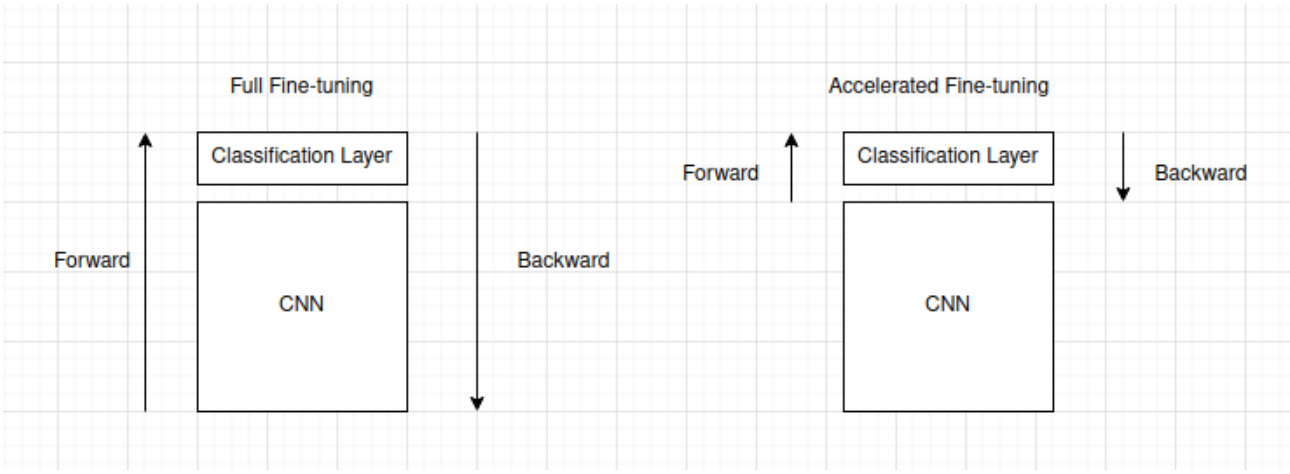

Figure 3: Illustration of accelerated fine-tuning vs. full (or non-accelerated) fine-tuning

Evaluation Metrics

As we have seen in Lecture 8, a number of evaluation metrics and methods can be used to quantify the performance of a classifier. Among those are:

- The Confusion matrix, which indicates the number of correctly classified samples in its diagonal, and the misclassified samples (the 'confusions') in the non-diagonal elements of the array.
- Precision
- Recall
- F1 score (harmonic mean of Precision and Recall)
- MCC score

Please refer to Lecture 8 for more details on those metrics and how to calculate them.

K-fold Cross Validation

K-fold cross validation is a more in-depth method of evaluating a classification model and ensure that every piece of data in the dataset is used in the evaluation process. In k-fold cross validation you split your dataset into k distinct equally-sized (or approximately equal-sized) partitions or groups (i.e. subset of the original dataset). Then, select a group as a 'hold-out' or test dataset, and train your classifier on the data in the (k-1) other groups. Evaluate the results on your hold-out test dataset. Repeat by choosing a different group as hold-out.
You can then average performance metrics across the different test sets to get an idea of how the classifier performs on the dataset overall, as well as variance in the results, which provide some additional insight into how volatile the classifier is.

For example, with k=2 you would split your dataset into two equally-sized partitions (p1 and p2) and then 1) train on p1 and test on p2, and 2) train on p2 and test on p1. With k=5 you would train on 80% of the data and test on the remaining 20% for each of 5 equally-sized partitions.

Deliverables

You need to submit the following on Canvas:

1. **Group Submission (one file per group)**: Your completed Jupyter Notebook `CAB320-Assignment2-Learning.ipynb`. The notebook should implement all functions named in the template and the group-tasks instructions below and provide details about what each of these functions should accomplish, provided in the markdown cells of your notebook. The marker needs to be able to reproduce your work by simply running your code cells in the order they come, and/or maybe uncommenting function calls in the main blocks of your code (only with very clear instructions to do so included in markdown cells and in the corresponding code cells).
2. **Individual Submission: Individual report** (one pdf file) with answers to the questions in the individual component of the assignment (see below).

Section 1 - Your Group Tasks

You main tasks are the following:

- follow the instructions below,
- code the requested functions within the script,
- write a report (directly in the Jupyter Notebook containing your code) that includes some documentation of your developments, a selection of results and analysis and discussion of those results.

Code

All your code should be submitted in **a single Jupyter Notebook file, based on the template provided**. You are required to keep the provided variable and function names exactly as they are in the template, as well as the provided interfaces. In this file, you should structure your code so that your experiments can be clearly identified and easily repeated by the marker. For example, markdowns should clearly explain what to execute for each task, and your code execute under the right Task label.

Code Quality:

Although this will not be marked explicitly, the following gives recommendations for your code quality:

Criteria	- Code is generic, well-structured and easy to follow. - Use of auxiliary functions that help increase the clarity of the code - In-line comments used frequently and well written	- No unnecessary loops where e.g array operations are suitable - Clear header comments and useful in-line comments - Proper use of data structures	- No magic numbers and appropriate variable names - Evidence of testing code and plotting

Tasks and Reporting

Your submission must detail how your code functions and the results of running a number of machine-learning related experiments, in the format of a single Jupyter Notebook, using code (to implement and

execute the required functions) and markdown cells (to give further overview of the function and report on your experiements and results). The following lists the tasks that you need to perform, and the information that must be included in the notebook (a.k.a. the report is the collection of results obtained from running your code associated with the markdown cells providing the analysis).
The total of marks possible for each task is given in parenthesis.

Tasks (Total Marks: 40)
(see also in the Jupyter Notebook template)

1. Implement the `my_team()` function
    - Code: `my_team()` (**1 mark**)
2. Download the `small_flower_dataset` from Canvas and load the data
    - Code: Function `load_data()` (**2 marks)**
3. Prepare your training, validation and test sets for the non-accelerated version of transfer learning.
    - Code: Function `split_data()` **(2 marks)**
    - Report: Include details of how you have split the data to perform this training. Ensure the split is reasonable and does not introduce class imbalance during training. **(2 marks)**
4. Using the `tf.keras.applications` module download a pretrained MobileNetV2 network.
    - Code: Function `load_model()` (**2 marks)**
5. Replace the last layer of the downloaded neural network with a Dense layer of the appropriate shape for the 5 classes of the small flower dataset {(x1,t1), (x2,t2),..., (xm,tm)}.
    - Code: within `transfer_learning()` **(4 marks)**
6. Compile and train your model with an SGD optimizer using the following parameters `learning_rate=0.01, momentum=0.0, nesterov=False`. (NB: The SGD class description can be found at https://keras.io/api/optimizers/sgd/)
    - Code: Function `transfer_learning()` (**2 marks)**
    - Report: Provide an overview of the function and how it works **(1 marks)**
7. Plot the training and validation errors and accuracies of standard transfer learning
    - Code: **(2 marks)**
    - Report: Briefly comment on the results **(1 mark)**
8. Experiment with 3 different orders of magnitude for the learning rate.
    - Code: Plot the results. **(2 marks)**
    - Report: Discuss the results in a markdown cell. **(2 marks)**
9. Run the resulting classifier on your test dataset using results from the best learning rate you experimented with. Compute and display the confusion matrix.
    - Code: **(2 marks)**
10. Compute the precision, recall, and f1 scores of your classifier on the test dataset using the best learning rate.
    - Code: **(2 marks)**
    - Report on the results and comment. (**1 mark)**
11. Perform k-fold validation on the dataset with k = 3. Repeat with two different values for k and comment on the results.
    - Code: **(2 marks)**
    - Comment on the results and any differences with the previous test-train split. (**1 mark)**
12. With the best learning rate that you found in the previous task, add a non-zero momentum to the training with the SGD optimizer (consider 3 values for the momentum).
    - Code: **(1 mark)**
    - Report on how your results change. **(1 mark)**
13. Now using "accelerated transfer learning", repeat the training process (k-fold validation is optional this time). You should prepare your training, validation and test sets based on {(F(x1).t1), (F(x2),t2),...,(F(xm),tm)}, and re-do Task 12.
    - Code: Function `accelerated_learning()` (**4 marks)**

- Report: Plot and comment on the results and differences against the standard implementation of transfer learning. (**2 marks)**

14. Conclusion: use the results of all experiments to make suggestions for future work and recommendations for parameter values to anyone else who may be interested in a similar implementation of transfer learning. (**1 mark)**

Comments on the writing and marking of your report:

- Function descriptions should be written so that the reader could implement them in python from scratch based solely off the report.
- The report should tell a cohesive story about the training method used and so that the results are easily interpretable by the reader.
- Do not forget axis labels and titles/captions for plots you include. Make sure any text within plots is legible and if there are multiple plots on the same axis, make sure they are legible as well.

Section 2 - Your Individual Tasks

Individually, write a short report (**max. 3 pages** in total, with font no less than 11pt) in which you will answer the following questions. Although it is recommended to first progress sufficiently on the group tasks (at least on the 'standard transfer learning' task) before you complete this individual component, you may complete it as you see fit in parallel to the group work. That said, please do not wait until the last minute to work on this report.

2.1 Delivery to a non-technical client (recommended length: max. 1 page)

You have successfully trained your flower classification model, and it achieves a certain level of accuracy on your test set. Now, consider you need to present this solution to a client who is enthusiastic about using it for their business (e.g., a nursery wanting to automate plant labelling or a landscaping company wanting to identify plants in client gardens) but has little to no understanding of machine learning concepts like accuracy, error rates, or the inherent non-stochastic nature of modern machine learning predictions.

Question: Describe how you would effectively communicate the capabilities and, crucially, the limitations of your flower classification system to this non-technical client.

You may consider the following points:

- What specific language and analogies would you use to explain concepts like the possibility of misclassifications, the factors that might affect the model's performance in real-world conditions (e.g., image quality, lighting, angle), and the difference between achieving high accuracy on a curated dataset versus in uncontrolled environments?
- What steps would you take to ensure the client has realistic expectations and understands the potential for errors or the need for ongoing monitoring and refinement of the model, thus minimizing the risk of them being misled or over-relying on the technology?

2.2 Ethical Considerations

As future engineering professionals working developing machine learning solutions for clients, your technical decisions carry ethical weight. The aim of this section is to develop your *ethical compass* in relation to your work as a future engineer.

After completing some of the technical component of your assignment, choose **TWO** of the following questions to explore in no more than **1 page each**. Your response should draw on relevant literature, technical reasoning, and personal reflection. This is an individual submission, for your own thinking.

You may use GenAI tools to help you brainstorm and reflect if you need, but **the answers in the report should be your own, formulated by yourself**.

Note there is no right or wrong answer to those questions as such. The purpose of this exercise is not for you to solve complex ethical dilemma but to make you think about it in the context of the specific machine learning task conducted in this assignment. To show your own thinking your answers should be specific and contextualised.

Proposed Questions:

1. Who owns nature in datasets?
Large datasets used to train deep-learning models often originate from photographs taken by various individuals or obtained from various online sources.

- Explore the implications of using publicly available images of natural objects like flowers for machine learning.
- Should nature be considered "free data" in the context of ML training?
- How do copyright, consent, and/or Indigenous sovereignty relate to the use of natural imagery in dataset construction?

2. Bias in pretrained models: Hidden patterns in flowers or humans?
MobileNet was pretrained on ImageNet—a large and broad dataset not specific to flowers.

- Discuss the potential biases or limitations that can be inherited from this pretraining when applying it to classify flower species.
- What are the risks of reusing pretrained models in real-world contexts such as environmental monitoring or agricultural robotics if underlying biases persist?
- How can you detect and address biases in your transfer learning pipeline?

3. When models see more than flowers: Dual-use and unintended applications
While this task is limited to flower classification, the same pipeline can be used for face recognition, plant surveillance, or invasive species tracking.

- Reflect on the dual-use nature of transfer learning architectures.
- How can your work in this assignment translate (intentionally or not) into surveillance tools or ecological intervention technologies?
- What responsibility do engineers have in anticipating misuse or unintended applications of such workflows?

For each question, we recommend the following template:

A) Clearly state which question you are addressing.
B) Ethical Analysis & Technical Context

- Clearly define the ethical dilemma or issue.
- Explain how it connects to your technical workflow (e.g., dataset, model choice, deployment implications).
- Discuss perspectives or tensions (e.g., innovation vs responsibility, open data vs consent, sustainability vs accuracy).
- Use relevant examples or scenarios to illustrate your points. Cite supporting evidence such academic or technical resources as appropriate.

C) Personal Reflection
The following are examples of discussions points, but you may also choose your own.

- Reflect on how this ethical issue might shape your values and decisions as a future engineer.
- How might it affect how you design, deploy, or discuss AI systems in future roles?
- What is your ethical responsibility when working with powerful tools like deep learning and pretrained models?

Additional Tips and Frequently-Asked Questions

- **How many submissions can I make?**
  - You can make multiple submissions, however, only the last one will be marked.
- **How do I find team-mates and form a group?**
  - Post a message in the 'Group Searching' channel on Canvas Discussions
- **Do I need to be in the same group as for Assignment 1?**
  - You can stay in the same team as for Assignment 1, however, this is not compulsory, and you are welcome to change for Assignment 2. NB: if you do change groups, make sure you communicate this as soon as possible with your previous (and new) group members.
- **How do we get organised as a group?**
  - Make sure you discuss early with your team-mates some of the fundamental workings of the group, including:
    - How are you planning to split the workload, and make sure to work together consistently
    - How will you be communicating; how often will you be meeting and how/where?
    - How will you split the workload? Make sure to revisit this as regularly as needed, without waiting for the last week before the deadline