CAB320 Assignment 2 – Machine Learning

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

Key Information and instructions

- The submission for this assignment is **due Week 13 (Friday 31 May)**
- Register your group on Canvas much before the deadline
- Submit your work via Canvas (one submission per group) by the deadline
- Group size: 3 people per group recommended (and maximum). 2 person-groups accepted but completion of the same tasks required. (NB: a group of 1 may be accepted but only in exceptional circumstances and with approval from the unit coordinator on a case-by-case basis).
- Make sure to list all the members of your group in the report and the code

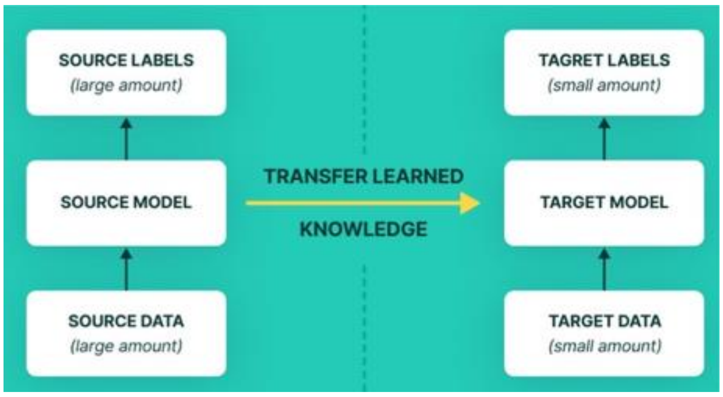

Figure 1: Illustration of the Transfer Learning process

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)**.

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)

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.

Your tasks

You main tasks are the following:

- follow the instructions below,
- code the requested functions within the script,
- And write a report that includes some documentation of your developments, a selection of results and analysis and discussion of those results.

Deliverables

You should submit via Canvas only two files

1. A **report** in **pdf** format **strictly limited to 8 pages in total** (be concise!) containing:
    a. a description of your investigation with results clearly presented in tables and figures,
    b. a recommendation for the values of the investigated parameters.
2. A Python script to reproduce your experiments, in `TransferLearning.py`
   The script should implement all functions named in the following section. Details about what each of these functions should accomplish are provided in the script when downloaded from Canvas. The marker needs to be able to reproduce your work by simply uncommenting function calls in the main block of your script.

Code

The functions that need to be coded and the mark breakdown for the code are as follows:

Implementing functions of `TransferLearning.py`: 20 marks
Each function is worth up to the following number of marks depending on functionality.

- `my_team()`: 1 mark
- `load_model()`: 2 marks
- **Transfer Learning Functions: 10 marks**
    - `split_data()`: 2 marks
    - `transfer_learning()`: 4 marks
    - `accelerated_learning()`: 4 marks
- **Evaluation Metrics Functions: 7 marks**
    - `confusion_matrix()`: 1 mark
    - `recall()`: 1 mark
    - `precision()`: 1 mark
    - `f1()`: 1 mark
    - `k_fold_validation()`: 3 marks

All your code should be submitted in **a single Python script file**. In this file, you should structure your code so that your experiments can be easily repeated by uncommenting function calls in the main block of your script.
That is, your code should look like the following (note here we combined simple tasks in one section, e.g. Tasks 1, 2, 3):

```
if __name__ == "__main__":
    pass
    # task_1_2_3()
    # task_4()
```

```
        # task_5()
        # task_7()
      :
```

Your code should calculate all metrics by itself without using imported libraries (apart from `numpy`). You cannot for example use keras to calculate recall, you must implement `recall()` yourself.

Code Quality:

For each function, code quality will be applied as a scaling factor to the mark achieved for functionality. The scale will be applied as followed:

Scaling factor	**100%**	**80%**	**60%**	**40%**	**20%**
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 - Appropriate variable names	- 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 reasonable variable names for the most part - Evidence of testing code and plotting	- Some header comments but lacking in-line comments - Difficult to understand code or variable names	- Code gives readers headaches - Unclear variable names - No evidence of testing

Note the quality factor is cumulative: to get "i %", the report needs to satisfy all the positive items of the column "j %" for all j ≤ i. For example, to obtain a quality factor of 40%, your code needs to satisfy all items of the 20% column AND all items or the 40% column.

Report: Total of 20 marks

The report must detail how your code functions and the results of running a number of machine-learning related experiments, in the format of a laboratory report. The following lists the tasks that you need to perform, and the information that must be included in the report. The specified marks are the marks contributing to the report total mark (out of 20).

1. Download the `small_flower_dataset` from Canvas.
2. Using the `tf.keras.applications` module download a pretrained MobileNetV2 network.
3. 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)}.
4. Prepare your training, validation and test sets for the non-accelerated version of transfer learning.
5. 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/)
6. **6.** Include an overview of functions and how each works in your report (NB: this applies to all the functions named above in the Code section of this document): **(4 marks)**
7. **7.** Plot the training and validation errors and accuracies of standard transfer learning (**3 marks)**
   - 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.
8. Experiment with 3 different orders of magnitude for the learning rate. Plot the results and discuss. **(3 marks)**
9. **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. **(2 marks)**

10. Calculate the precision, recall, and f1 scores of your classifier on the test dataset using the best learning rate. Report on the results and comment. (**2 marks)**
11. Perform k-fold validation on the dataset with k = 3. Comment on the results and any differences with the previous test-train split. Repeat with two different values for k and comment on the results. (**2 marks)**
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).
13. Report how your results change. (**1 mark)**
14. 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. Plot and comment on the results and differences against the standard implementation of transfer learning. (**2 marks)**
15. 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.

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 the unit's MS Teams
- **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.
- **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