COMP309 / AIML421 — *Machine Learning Tools and Techniques*

Project: Image Classification

30% of Final Mark — Due: 23:59pm Tuesday 29th Oct

1 Objectives

It is almost effortless for we humans to quickly distinguish a dog from a cat, a smile from a frown, a person you know from a stranger. However, these are hard problems for a computer, and have involved decades of research into computational vision. Recently, general progress has been achieved in the field of machine learning through advances in deep learning. Deep convolutional neural networks (CNNs) are now able to achieve competitive or even superior performance to human beings on hard vision problems – at least, given plenty of data and the substantial compute power needed to train them. This raises many interesting questions to us: Why does it work? What are the principles that lie behind the success? How can we construct a CNN in practice and make it work properly? What can we do to improve its generalisation ability (i.e. classification accuracy on unseen data)?

The goal of this project is for you to have hands-on experience with building, training and evaluating CNNs, by performing a realistic (and hard!) image classification task. Specific objectives of this project are:

- To understand the basic image classification pipeline.
- To use available data for model training and hyper-parameter tuning, properly.
- Be able to implement CNNs.
- Save a trained CNN model after the training process.
- Load a trained/saved CNN model to classify some unseen test data.
- To understand the subtleties of the different loss functions and optimisers available for training CNNs.
- To learn and be able to use various techniques to improve the CNN performance.
- To write a clear report – an essential skill for a data scientist/ machine learning engineer.

These topics are covered mainly in week 7 to week 11, but also involve content from previous weeks. Research into online resources for image processing, image classification, `PyTorch` / `torchvision` and other related topics is very much encouraged. Make sure you finish reading this whole document before you start working on the project.

2 Dataset

The dataset used in this project is an image classification dataset, which consists of a total of 6,000 RGB images across 3 classes, with 2,000 images per class. The 3 classes are *tomato*, *cherry*, and *strawberry*. Each image belongs to only one class.

Figure 1: Example images: images in the three rows belong to the *tomato*, *cherry*, and *strawberry* classes, respectively.

Figure 1 shows some example images. Notice the wide variety of scales, colours, angles, background clutter, illumination, resolution, numbers of the target item, and so on. There may be outliers to be handled. We think this is a challenging task to do well on! Yet with proper initial data processing a well-designed deep CNN can achieve a classification performance that is much better than random guessing.

We have conducted some initial data pre-processing on the images, such as data cleansing, to ensure a moderate quality of the data. Most of the images have been re-sized to the same size/dimensions, e.g. 300×300 (note this is much larger than,say, MNIST or CIFAR).

From the total 6,000 images, you are given **4,500 images** as the *training* data to train/learn a CNN model. You are free to use this data in your way in order to develop the best model. The other 1,500 images will not given to you, and will be used (by tutors) as the *test* data to evaluate the CNN model that you need to train and submit.

NOTE: All images in the dataset are downloaded from Flickr. Use of these images must respect the corresponding terms of use. Hence, please do **NOT** distribute the data outside of this course, and especially not on the Internet for public access.

3 Instructions

Detailed step-by-step instructions are given below. Please make sure you read them thoroughly.

- Step 1: Conduct exploratory data analysis (EDA).
  By performing EDA, you can have a better understanding about the data, which helps you decide whether to perform pre-processing (such as image filtering or feature engineering) and which methods to perform in order to improve the data quality. Check they're all the right size.

- Step 2: Apply pre-processing techniques to improve quality of the data.
  For simple pre-processing, you can just simply remove some noisy images/instances. Alternatively, you could apply feature engineering techniques (e.g. image feature descriptors to extract high-level features, feature extraction, or feature constructions) to try and improve the quality of the data and hence classification accuracy of your final model.
- Step 3: Build a simple **Baseline** model.
  You should build a simple/standard neural network, i.e. multilayer perceptron (MLP) trained by back-propagation for this step, which helps you have a baseline intuition / idea of the task. It is probably easiest to just do this in `PyTorch`. Don't use a CNN for this part! The point is to get a sense of how well a "vanilla" neural net can do, without convolutions. Because it is "dense", this probably can't be too large, as it has to be trained in a reasonable amount of time. Start small and see how you go.
- Step 4: Next, build your own **CNN** model.
- Step 5: Tuning the CNN model.
  You should tweak the model built from Step 4 by using the knowledge you have learned so far. Some suggestions are given below and you should try at least five of the following points:
  1. Use cross-validation on the given images to tune the model or hyper-parameters.
  2. Investigate loss functions.
  3. Investigate minibatch sizes.
  4. Investigate optimisation techniques.
  5. Investigate the regularisation strategy.
  6. Investigate the activation functions used.
  7. Get more data, i.e. you can obtain more data (e.g. download from internet, or augment the training data you're given) to enrich the training set.
  8. Use "transfer learning" by building upon an existing model that has been pre-trained by more wide-ranging data, like ImageNet.
  9. Use ensemble learning (e.g., stacking different models) to boost your model.
  10. Some other option of your own choosing!
- Step 6: Write a **report**.
  For this step, you are required to complete a formal written report. Please check the detailed task description and submission requirements below.

If you have to resize the images to keep compute time down, do so.

4 Task Description and Submission Requirements

You are given **4,500 images** out of the 6,000 images, and a sub-folder named `data` that includes a small number of example images.

The images in the `data` sub-folder can be placed in your `code/testdata` directory (see below) for the sake of development, and are just used as a faked example to show you what will happen when

evaluating your submitted model. After submission, we will put the 1,500 unseen test data/images that are *not* given to you into the `testdata` sub-folder, and then run the `test.py` with your submitted model. You should use some images in the `testdata` sub-folder while developing `test.py`, to ensure your code functions properly.

Submit the following (see Figure 2) :

1. `report.pdf` - your report.
2. a directory, `code`, containing:
   (a) `train.py` - this is code to build your CNN model, train it, and save the trained model. Clean and well commented code is strongly encouraged, and simple is usually best. If you add any extra pre-processing steps, please make sure you also implement them in `test.py` so that they can later be applied to the unseen test images.
   (b) `model.pth` - this is produced by your own run of *train.py*, and is the stored, trained model that you wish to be evaluated on.
   (c) `test.py` - this reads in `model.pth`, and whatever images are present in the `testdata` directory. It runs the model on those images, and reports its performance.
   (d) `testdata` - this is a directory, and you should check that `test.py` runs model `model.pth` on images placed in here. It can be empty at submission time.

Please don't submit the training images back to us! - you can keep those anywhere handy, but not in the submission itself. For testing, we do not need to re-train your model or re-run any other program except for `test.py`, and this program **only** runs your your submitted model `model.pth` on the images in a subfolder `testdata`.

We will place the unseen test images there, at test time. Only if really necessary should you submit any other Python files for running your `test.py`. If you do need this, include good documentation.

Submit both the **code** and the **report**, which have 25 and 60 marks, respectively. The other 15 marks of this project is based on the testing performance of your trained/submitted model.

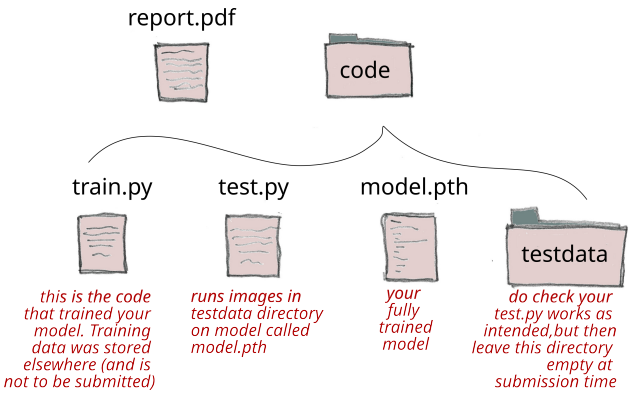

Figure 2: Structure of your submission – files and folders.

4.1 Code [25 marks]

You need to make sure your codes can run on the **ECS School machines** since we plan **in-person marking** of the codes. If your programs cannot run properly, you should provide a `buglist` file. Please follow the coding instructions here strictly, as otherwise some marks will be deducted.

ie. you really do want this to run smoothly on ECS machines!

4.2 Report [60 marks]

You must submit a written report in PDF, describing the methodology you used and the results.

The report should include the following information, and it probably makes sense to make these section headings as it helps the reader to navigate.

- Introduction [4 marks]: briefly describe the problem and summarise your approach (1 paragraph).
- Problem investigation [15 marks]: describe and justify what you have done in terms of EDA, data pre-processing, feature design/selection methods, and any other methods/algorithms you used to improve your data. A good way to report this part is to describe your findings in EDA, the (pre-)processing you have done based on the findings (if applicable), why you did such processing, and whether such processing helped improve the model.
- Methodology [30 marks]: you are required describe and justify what you have done for training the CNN in terms of the following aspects if applicable, but you should include at least five of them:
    1. how you use the given images (e.g. how you split them into training and validation sets or k-fold cross validation, and how you use them for training),
    2. the loss function(s),
    3. the optimisation method(s),
    4. the regularisation strategy,
    5. the activation function,
    6. hyper-parameter settings,
    7. how many more images obtained (should be at least 200 in order to get marks) and how you got them to enrich the training set,
    8. the use of existing models. For example, you may use transfer learning (e.g. models pre-trained by ImageNet).
    9. any extra technical advancements you have made to improve your model. For example, you may use ensemble learning (e.g., stacking different models) to help boost the performance.
- Summary and discussion [7 marks]: This should take the form of a self-contained summary, in which you clearly and succinctly
    - describe the structure and overall settings of your chosen baseline MLP, and your best CNN

  - compare the results of your MLP and CNN, in terms of the training time and the classification performance (for validation/test data). Plots of loss and accuracy versus time, for train and test sets, would be useful graphics to show.
  - note any striking differences, and discuss why you think they occurred.

- Conclusions and future work [4 marks]: describe the *overall conclusion(s)* that can be drawn from your study, show insight into the potential pros and cons of whatever approach you took, and list next steps / future work. that you would suggest could be undertaken to take things further. Aim to exhibit insight and understanding here - simply saying 'do more of the obvious things' for next steps doesn't amount to insight :)

  (note: it's not a 'Reflection' about your own experience, but a technical conclusion to a technical study).

About report writing

See the accompanying `COMP309_report_writing_guide.pdf`.

As a general rule, try to avoid too much 'I...' language: it's not a diary about you but a technical report about a technology, so aim for that more formal style. If you find yourself inserting lots of numbers into a long paragraph of text... consider replacing that with a nice table or a figure, and some brief lines of explanation instead. The reader will thank you.

Look to insert *conceptual* content: don't say 'My conclusion is that Method B is better...' - that is a bald statement of fact, as opposed to a *conclusion* based on *reasoning*.

Figures need proper axis labels, font large enough to read, and a caption summarising what the take-home message is.

Document the results you obtained in a structured way for clarity.

The report should *not exceed **10** pages* with font size no smaller than 10.

Take care with figures - if you shrink them, the axes and key can become illegible.

Don't forget to include your name on the first page.

4.3 Relevant Data Files and Program Files

A soft copy of this assignment, the relevant data and code files are available from the course homepage: http://ecs.victoria.ac.nz/Courses/COMP309_2024T2/Assignments

5 Some Torch details to keep in mind

- a good place to start:
  `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`
- You will have to do some up-front work to figure out how to read the data in - this is a good job for `torchvision`, see `https://pytorch.org/vision/stable/io.html#image`.
- PyTorch for saving, and then reloading, models:
  `https://pytorch.org/tutorials/beginner/saving_loading_models.html`

- saving a model: `torch.save(model, PATH)`
- A common PyTorch convention is to save models using a `.pth` file extension.
- loading a model: `model = torch.load(PATH)`
- If you use dropout or batch normalisation, note that you must call `model.eval()` to set these layers to evaluation mode before running inference (i.e. `test.py`). Failing to do this will yield inconsistent inference results.
- There may be caveats regarding the use of dropout in convolution layers.
- If you see silly results don't ignore them - they're telling you something. For instance if you see 100% accuracy on one class *something is wrong! Find it.* Just for example, apparently combining an NLLLoss function with convolutions can cause the PyTorch classification Tensors to equate to NaN!

6 Assessment

We will endeavour to mark your work and return it to you as soon as possible. The tutor(s) will run a number of helpdesks to provide assistance to answer any questions regarding what is required.

6.1 Submission Method

Both the program *code* and the PDF version of the *report* should be submitted through the web submission system from the COMP309 course web site **by the due time**.
There is no need to supply hardcopy of the documents.

KEEP a backup and receipt of submission.

Submission should be completed on School machines, i.e. problems with personal PCs, internet connections and lost files, which although eliciting sympathies, will not result in extensions for missed deadlines.

6.2 Late Penalties

The assignment must be handed in on time unless you have made a prior arrangement with the course coordinator or have a valid medical excuse (for minor illnesses it is sufficient to discuss this with the coordinator). The penalty for assignments that are handed in late without prior arrangement is one grade reduction per day. Assignments that are more than one week late will not be marked.