COMP 309 — *Machine Learning Tools and Techniques*

Project: Image Classification

36% of Final Mark — Due: 23:59pm 24 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 many decades of research into computational vision, specifically. Recently, general progress has been achieved in the field of machine learning through advances in deep learning, which makes these problems seem solvable. Deep convolutional neural networks (CNNs) are now able to achieve competitive or even superior performance to human beings on hard vision problems. This raises many interesting questions to us: Why does it work? What are the principles behind it? How can we construct a CNN? How can we make it work properly? What can we do to improve the recognition rate (i.e. classification accuracy) of the CNN model?

The goal of this project is for you to have hands-on practice to better understand CNNs and the above questions, by performing a realistic image classification task. Specific objectives of this project are:

- To understand the basic image Classification pipeline.
- To properly use available data for model training and hyper-parameter tuning.
- Be able to implement CNNs.
- Be able to save a trained CNN model after the training process.
- Be able to load a trained/saved CNN model to classify unseen test data.
- To understand the influences of different loss functions and optimisers for training CNNs.
- To learn and be able to use techniques to improve the CNN performance.
- To write a clear report, which is an essential skill for a data scientist/a 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*,

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

and *strawberry*. The three classes of images have been manually made to be completely mutually exclusive, i.e. each image belongs to only one class.

Figure 1 shows some example images, where each of the three rows shows four images from the *tomato* class, the *cherry* class, and the *strawberry* class, respectively.

We have conducted some initial data pre-processing on the images, such as data cleansing, to ensure a moderate quality of the data. All 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). However, as you can see from Figure 1, the images show different properties in terms of the background, light condition, number of objects, resolution, etc. These images may also be noisy and there may be outliers, which need to be handled properly. This task looks difficult, but a well-designed deep CNN with proper initial data processing should be able to achieve a classification performance that is much better than random guessing on this dataset.

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 trained/submitted.

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 COMP309, especially on 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.

- 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 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`.

- Step 4: Build a CNN model.
  Here, you need to build a CNN based model to classify the images.

- 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) to enrich the training set.
  8. Perhaps use existing models pre-trained by data like ImageNet to conduct "transfer learning" to fine-tune your model.
  9. Use ensemble learning (e.g., stacking different models) to boost your model.
  10. Some other option of your own choosing.

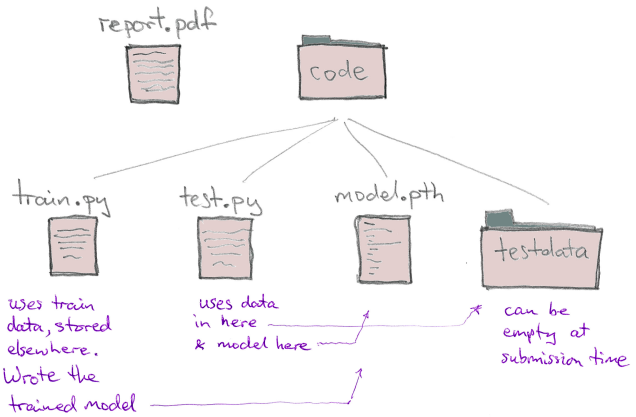
- Step 6: Write a report.
  For this step, you are required to complete a formal written report. Please check the detailed task description and submission requirements below.

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:

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 do not 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 remove any images we find in that directory and place the unseen test images there, at test time. If applicable, you should submit any other Python files that are needed for running your *test.py*, with 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.

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 (unless we need to choose online demonstration of the codes, where you can

run the codes in your laptop). If you 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 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:

- 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.
- Result discussions [7 marks]: compare the results of your CNN with the baseline method MLP in terms of the training time and the classification performance. Analyse why differences occurred. You should also describe the settings of your MLP here.
- Conclusions and future work [4 marks]: describe the conclusions, potential pros and cons of your approach, and list any possible future work.

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

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_2021T2/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.

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.