COMP 309 — *Machine Learning Tools and Techniques*

Project: Image Classification

36% of Final Mark — Due: 2020-10-21 23:59:00

1 Objectives

Thanks to our complex and powerful brain, it is almost effortless for us to quickly distinguish a dog from a cat, or instantly detect a traffic sign while driving on the motorway at a high speed, or immediately recognise our friends by their Facebook avatars. However, these are hard problems for a computer. Recently, tremendous progress has been achieved in the field of machine learning because of the significant development of deep learning, which makes these problems seem solvable. More specifically, deep convolutional neural networks (CNNs), one of the deep learning models, achieve competitive or even better performance than human beings on these 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 a hands-on practice to help better understand CNNs and the above questions by performing an image classification task, which is the most promising application of CNNs. You are required to use Keras to accomplish the task, since Keras, which is based on the most popular deep learning framework – TensorFlow, is a state-of-the-art and user-friendly tool designed to enable fast experimentation with deep CNNs. 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 using Keras.
- 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, e.g., feature engineering and transfer learning.
- 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 Keras, image classification, image processing and other related topics is encouraged. Make sure you finish reading this whole document before you start working on the project.

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

2 Dataset

The dataset used in this project is an image classification dataset, which consists of totally 6,000 RGB images of 3 classes with 2,000 images per class. The 3 classes are *tomato*, *cherry*, 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. 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. Theose images may also include noisy images, noisy objects or outliers, which need to be handled properly. This task looks difficult, but as the state-of-the-art approach for image classification, a well-designed deep CNN with proper initial data processing should be able to achieve a good classification performance or 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. Furthermore, the dataset is an important part of our research work in preparation. 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 have read them thoroughly and carefully.

- 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.

- Step 4: Build a CNN model.
  Here, you need to build a CNN based model using Keras 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 tips 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 optimisation techniques.
  4. Investigate the regularisation strategy.
  5. Investigate the activation function.
  6. Get more data, i.e. you can obtain more data (e.g. download from internet) to enrich the training set.
  7. Implement ways to set hyper-parameters
  8. 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.

- 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 folder named *template* which includes *test.py*, *train.py*, a sub-folder named *model* that includes an example trained model named *model.h5*, and a sub-folder named *data* that includes a small number of examples images. Note there are two versions of the template: one for Python 3.7 and one for Python 3.8, but we recommend you to use Python 3.8, since it is the default in ECS School machines.

- *train.py* is the simple template for you to build your CNN model, train the model, and save the model for submission (and marking). Feel free to change *train.py* according to your own style, where clear comments are encouraged However, if you have added any extra pre-processing steps, please make sure you also implement them in *test.py* so that they can later be applied to unseen test images. More details about what you should and should not change are given in *train.py* and *test.py*. Make sure you read them carefully before you start coding.
- *model.h5* in *model* is an example/baseline model obtained via simple training, which is just to show you what a trained model looks like.
- The images in the *data* sub-folder is just used as a faked example to show you what will happen when evaluating your submitted model, i.e. we will put the 1,500 unseen test data/images that are **not** given to you into the *data* sub-folder, and then run the *test.py* with your submitted model. You may use the images in the *data* sub-folder to evaluate your model in the same manner as a tutor before submission to ensure your code functions properly.

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

4.1 Code [25 marks]

As stated above, you should modify the template *train.py* and *test.py*. More detailed coding instructions, which specify the part you should or should not change, are given in the template *train.py* and *test.py*. Tensorflow 2.0 is recommended. and you may need to install the *imutils* package (pip3 install –user imutils) to run on GPU.

You are required to submit a folder named *code*. The *code* folder should include (at least) three parts: your modified *train.py*, your modified *test.py*, and a sub-folder named *model* that includes your final model named as *model.h5*. You do not need to submit the training images, since for testing, we do not need to re-train your model or re-run any other program except for *test.py*, but **only** run your submitted *test.py* with *model.h5* on the unseen test data/images. However, if applicable, you should submit other Python files that are needed for running your *test.py* with good documentation.

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 strictly the coding instruction in the provided template and submission requirement, otherwise, some marks will be deducted.

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. Analysis 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.

5 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_2020T2/Assignments

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 required hard copy 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 lecturer or have a valid medical excuse (for minor illnesses it is sufficient to discuss this with the lecturer). 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.