CITS5508 Machine Learning
Semester 1, 2020

Lab Sheet 5

Assessed, worth 15%. Due: 11:59pm, Friday 29th May 2020

1 Outline

This lab sheet consists of a large project using the CIFAR-10 dataset. This is a data set containing 10 classes of 32×32 colour images. The training set is perfectly balanced, with 6,000 images per class. The test set contains 10,000 instances. Your task for this labsheet is to train an MLP and a CNN for the classification task and compare their performance. This lab sheet is a good practical exercise to test your understanding of the techniques covered in Chapters 10, 11, and 14.

2 Submission

Name your Jupyter Notebook file as **lab05.ipynb** and submit it together with your pre-trained CNN model file(s) to *cssubmit*: https://secure.csse.uwa.edu.au/run/cssubmit?p=np before the due date and time shown above. You can submit your files multiple times. Only the latest version will be marked.

3 Data Download and Preparation

CIFAR-10 can be downloaded from the website:

https://www.cs.toronto.edu/~kriz/cifar.html

The website supplies very clear description about the dataset. You need to download the **cifar-10-python.tar.gz** file from the link **CIFAR-10 python version**. The direct link is

https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

On Linux, you can unzip the file using the command `tar xvfz cifar-10-python.tar.gz`. On Windows, you may need to unzip twice, firstly to get the **cifar-10-python.tar** file and then unzip to get the data files. If your unzipping operation is successful, you should see the following files in your download directory:

```
batches.meta
data_batch_1
data_batch_2
data_batch_3
data_batch_4
data_batch_5
readme.html
test_batch
```

The training set is split into five *data_batch* files. The test set is in the *test_batch* file. To help you read these files correctly, a Jupyter Notebook file named *lab05-DuHuynh.ipynb* has been provided on LMS. Feel free to rename this file and modify it to form your Notebook file; however, make sure that your Notebook file and these data files are in the same directory. Otherwise, your code would require modifications in order to run in the marking process.

Note that the image pixel values have been normalised to the range 0..1 in *lab05-DuHuynh.ipynb*. You shouldn't need to perform any further feature scaling.

4 Google Colab

The training of your MLP and CNN will take a long time to complete if your computer does not have a GPU. It is very easy to upload the six data files above and your *lab05.ipynb* file onto your Google Drive space. In cell #2 of the supplied *lab05-DuHuynh.ipynb* file, you can insert the lines below to mount your Google Drive so that the data files are accessible in Colab:

```
from google.colab import drive
drive.mount('/content/gdrive')
print(os.listdir('.'))
```

When you run the cell above, you will be directed to a page where you need to authorise the mounting of the drive. After authorisation, you will be given a security code which you need to copy into a text field in Colab. After this is done, your Google drive is mounted. The next thing is to find out where the sub-directory that stores the data files is relative to the mount point. The line `print(os.listdir('.'))` above lists the contents of the mount directory. You should inspect the contents of the directory to figure out the path name to the data files. For example, if you put the data files in the subdirectory named *CITS5508* then the line

```
os.listdir('gdrive/My Drive/CITS5508')
```

might be the correct path name. If this is the case, then change the variable *pathname* in cell #2 to the following:

```
pathname = 'gdrive/My Drive/CITS5508'
```

You should be able to run the supplied *lab05-DuHuynh.ipynb* file if your path is set up correctly.

By default, GPU is not available on Colab. To specify that you need GPU, select the the menu item *Runtime* and then the option *Change runtime type*. In the popped-up window, select "GPU" for *Hardware accelerator*.

5 Tasks

- Design an MLP that has two hidden layers. In each hidden layer, use a suitable number of neurons and an appropriate activation function. Experiment with two possible settings for each of the following:
  – connection weight initialisation,
  – learning rate scheduling (you may need to write a small function for this), and
  – early stopping.
  Feel free to split the training set into a small validation set for these extra experiments. Set the number of training epochs to 10[1].

  Report the extra experiments above in your markdown cells. In your final code, supply only the code for the optimal parameter settings only (so that we can speed up the marking of your Notebook file).

  Report your trained model's classification accuracy and illustrate the confusion matrix on the test set. Show a few correctly classified images and a few failure cases from the test set.

- Design a CNN that has two convolutional layers. You will need to add pooling layers and fully connected layers. Similar to the MLP model above, experiment different parameter setting to optimising the performance of your model. You should explore two possible settings for each of the following:
  – kernel size,

[1] If early stopping kicks in, then the number of training epochs will be fewer.

– number of kernels[2],
– dropout rate, and
– activation function.

Again, report the optimal parameter findings in markdown cells only. Your final code should just use the optimal hyperparameters that you have found. You should structure your code as shown below so that we can have the option of loading your pre-trained model (which must be submitted with your Notebook file) when your labsheet is marked:

```
# ask the user whether retraining should be performed or not
answer = input('Retraining the CNN model (y/n)? ')
if answer == 'y':
    # construct and train the CNN network using the training set
    ...
    ...
    # after training has completed, the model should be saved to disk.
    # If a model file of the same name already exists, ask
    # the user whether to overwrite the file or not
    ...
else:
    # restore the model
    ...

# either the CNN has been trained or the pre-trained model
# has been reloaded.

# code for testing the CNN on the test set
...
```

Obviously, the first time when you run your code, you do not have a trained model yet, so you have to answer 'y' to the first question above. This will result in the start of training and saving of the final model.

In the skeleton code above, save the final checkpoint of your trained model to a file that carries your surname and give name so that your model file(s) do not mix up with another student's (there are no two students having the same surname and given name in the class), e.g., for my name, I could save my trained model to **Huynh_Du_CNN_final**. Note that there are usually more than one file associated with the trained model (i.e., the network architecture and connection weights may be saved to separate files). You need to make sure that you submit all the files so that we can run your code when we mark your labsheet. That is, your submission must be **complete**.

Similar to the MLP model, report the classification accuracy and illustrate the confusion matrix of your trained model on the test set. Show a few correctly classified images and a few failure cases from the test set.

- Compare the performance of your MLP and CNN models.

6 Penalty on late submissions

See the URL below about late submission of assignments:
https://ipoint.uwa.edu.au/app/answers/detail/a_id/2711/~/consequences-for-late-assignment-submission

[2] the term "kernel" and "filter" are often used interchangeably in computer vision