CITS5508 Machine Learning
Semester 1, 2021

Lab Sheet 5

Assessed, worth 15%. Due: 11:59pm, Friday 21st May 2021

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 **Surname_FirstName-lab05.ipynb** (please replace **Surname** and **FirstName**[1] by your own surname and first name). For your trained MLP and CNN models, name them as **Surname_FirstName-MLP** and **Surname_FirstName-CNN**. Your models are likely to be saved as a small number of files so **Surname_FirstName-MLP** and **Surname_FirstName-CNN** would be subdirectories containing those files. You need to make sure that these subdirectories were created in the same directory with your Notebook file, as this is what we will assume in our marking process.

You should zip[2] your Notebook file and the two subdirectories together and submit a single **lab05.zip file**[3] to LMS before the due date and time shown above. You can submit your zip file multiple times before the deadline. Only the latest version will be marked. After the deadline, you can still submit your Jupyter Notebook file but resubmission would not be possible.

NOTE 1: When you create your zip file for submission, ensure that

- **no data files** are included. The dataset for this labsheet is very large and we do not need you to supply the dataset back to us.
- **no checkpoint files** are included. You can create these files to monitor the training progress of your models but these checkpoint files should be for yourself only. You should submit only the final trained models, as including all the checkpoint files would significantly increase the size of your zip file (it may even be too large to be uploaded onto LMS).
- **no hidden files/directories** are included. Jupyter-Lab and Jupyter-notebook create backup files in the **.ipynb_checkpoints** directory. This is a hidden directory that might not show up (on Windows, it will depend on whether you turn on the option of showing hidden files/directories; in Linux/MacOS, you can type: `ls -al` to list the contents (including all hidden files) of the current

[1] You can include your middle name if you like. It does not matter. We just want to be able to distinguish students' files when we unzip the whole class's submissions together in one directory. **NOTE:** If your surname and/or given name contain special characters, such as apostrophes, '/', etc, you may have to omit them or replace them by underscores or hyphens as these characters have special meanings in Python: apostrophes are considered as single quotes; '/' is used to denote directories.

[2] On Linux and MacOS, type in a terminal window:

```
zip -r lab05.zip Surname_FirstName-lab05.ipynb Surname_FirstName-MLP Surname_FirstName-CNN
```

On Windows, see examples on the web, e.g.,
https://support.microsoft.com/en-us/windows/zip-and-unzip-files-8d28fa72-f2f9-712f-67df-f80cf89fd4e5

[3] When we download the whole class's submissions, Turnitin will insert your surname and given name in front of your zip file name.

directory). On the MacOS, there are further hidden files/directories starting with dot-underscore (._) characters. None of these hidden files should be included in your submission.

Any of the above files, if found in your submission, will incur a 5% penalty to your mark.

NOTE 2: You must submit your zip file onto LMS. You should reserve at least half an hour for the submission procedure as Turnitin carries out virus checking (which can't be disabled) on each submitted file. You should wait until you get an acknowledgement screen confirming that your submission has completed successfully. This is your digital receipt that you should keep as evidence of your submission. Do **not** send your zip file via email to the Unit Coordinator. Do **not** upload your zip file onto Google Drive or OneDrive and then send the link to the Unit Coordinator. Only files uploaded on LMS will be marked.

NOTE 3: Do not blindly copy examples that you found on the internet. Many of these examples are too complex and can only be trained on huge datasets such as ImageNet. These complex networks may make your zip file too large to be uploaded onto LMS. **Significant penalty** will be imposed on your mark if your MLP and CNN are too complex and do not match the specification stated in the labsheet. You should be able to complete this labsheet by following examples in the lectures, textbook, and sample codes from the author.

3 Data Download and Preparation

CIFAR-10 can be downloaded from the website: https://www.cs.toronto.edu/~kriz/cifar.html which covers 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 a subdirectory named *cifar-10-batches-py*. Under there, you should find the following files:

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

You should move all these files to the same directory with your **Surname_FirstName-lab05.ipynb** file. Here, the dataset has been organised as follows: 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, two additional files are supplied with this labsheet:

- **data_loader.py**. This file contains a Python class called `DataLoader`. The function `load_batch` can be used to read the dataset.
- **lab05-sample.ipynb**. This file shows you how to call the `load_batch` function mentioned above. Feel free to modify this file to form your Surname_FirstName-lab05.ipynb file.

Note that the image pixel values have been normalised to the range 0..1 by the `load_batch` function. You shouldn't need to perform any further feature scaling.

4 Tasks

(i) Use an 80/20 random split on the training set to form a validation set. After splitting, your training set should be 80% of its original size. Use these training, validation, and test sets for training and testing both the MLP and CNN later on.

(ii) Write a small function that takes in appropriate arguments so that it can be used to display 20 randomly sampled images from the arguments. This function should be called 3 times – for the training, validation, and test sets. The figure displayed by the function should show the corresponding class name of each image.

(iii) **Implementation of an MLP**

Design an MLP that has 2 to 3 hidden layers and an output layer. Use an appropriate function from `tensorflow.keras` to show a summary of your MLP architecture.

You should try to design your MLP in such a way that the hidden layers have fewer neurons than the input layer. This would help to avoid overfitting. You should train the network for 100 epochs but use the validation set for *early stopping* (which also helps to overcome overfitting).

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 will need to use *callback* and write a small function for this), and
– dropout rate.
You can use grid search together with the validation set created above to find the optimal value for each of them. The code for this hyperparameter fine tuning process should either be commented out or be moved to markdown cells in the final version of your Notebook file.

Your trained MLP model should be saved to the directory named Surname_FirstName-MLP.

(iv) **Implementation of a CNN**

Design a CNN that has 2 to 3 convolutional layers. You will need to add a pooling layer between consecutive convolutional layers. Before the output layer, you will need to have 1 to 2 fully connected layers and *batch normalisation* layers to help control the numerical values of the network weights. Use an appropriate function from `tensorflow.keras` to show a summary of your CNN architecture.

Similar to the MLP model above, you should explore **two** possible settings for each of the following:
– kernel size,
– number of kernels[4],
– activation function.

For your CNN, you can use the optimal way to initialise the network, the optimal learning rate scheduling, and the optimal dropout value that you found from the training of your MLP model above.

Once you have got the optimal values of these hyperparameters, either comment out the code or put the code into markdown cells.

Your trained CNN model should be saved to the directory named Surname_FirstName-CNN.

(v) **Structure of your code**

In the marking process, we want to have the following options in your implementation of both the MLP and CNN:

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

a) train each of your networks from scratch for 100 epochs using the optimal hyperparameter values that you found; or
b) load each of your trained models and train it for just 1 more epoch.

After that, the model (obtained from a) or b)) should be used to do the prediction on the test set. So your Python code should look like this:

```
if the MLP model subdirectory is present in the current directory
    load the model,
    display its architecture,
    train for one epoch.
else
    set up the model and display its architecture,
    train the model from scratch for 100 epochs using the optimal hyperparamter values.
test the model on the test set.
```

After testing your MLP model on the test set, you should report your trained model's classification accuracy and F1 score and show the confusion matrix on the test set. Display also a few correctly classified images and a few failure cases from the test set.

Similar to the code for your MLP, repeat the above steps for your CNN.

(vi) Compare and commment on the performance of your MLP and CNN models.

5 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 all the batch files for the dataset, data_loader.py, and your Surname_FirstName-lab05.ipynb file onto your Google Drive. If you decide to use Google Colab, then you should find out how to mount the data files on your Google drive so that they are visible in your code.

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

6 High performance computing (HPC) facilities at UWA

UWA has a number of multi-core computers, some of which have GPUs. It is not compulsory that you must use these facilities for this labsheet. However, they will certainly help you cut down the training time of your deep networks. An account should be set up for each student on **Kaya** (the name of the GPU host) at the end of week 9 or early week 10. Refer to the instructions given in the file **Using-Kaya.pdf** and watch the related video (available on LMS) if you want to use the UWA HPC facilities.

7 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