CITS5017 Deep Learning
Semester 2, 2023

# Project 1

Assessed, worth 15%. Due: 11:59pm, Friday, 15th September 2023

## 1 Outline

In this project you will deal with the CIFAR-10 dataset. This is a data set containing 10 classes of colour images of size $32 \times 32$ pixels. The training set is perfectly balanced, with 5,000 images per class. The test set has $1,000$ images per class. Your task for this project is to train an MLP and a CNN for the classification task and compare their performances. This project 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-proj1.ipynb** (please replace **Surname** and **FirstName**[1] by your own surname and first name). For your trained MLP and CNN models, use the `"tf"` format to save each model as a directory. Please name the directories for the two models as **Surname_FirstName-MLP** and **Surname_FirstName-CNN**. Under these directories should be files and subdirectories containing the connection weights and the meta-data of the models. You need to make sure that your models were saved in the same directory with your Notebook file, as this is what we will assume in our marking process.

**NOTE 1:** You should zip[2] your Notebook file and the two model directories together and submit **a single proj1.zip file**. When you create your zip file for submission, ensure that

- **no data files** are included. The dataset for this project 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 will significantly increase the size of your zip file.
- **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 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.

[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 proj1.zip Surname_FirstName-proj1.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

**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 to **cssubmit** (https://secure.csse.uwa.edu.au/run/cssubmit). You will need to login using your Pheme account to do the submission. You should reserve at least half an hour for the submission procedure as csmarks carries out virus checking (which can't be disabled) on each submitted file. Your zip file for the project can be quite large as it needs to include two trained models. You need to ensure that your internet connection is stable during the entire uploading process. If possible, have your computer wire-connected to your modem before you start your submission.

You should wait until you get an acknowledgement screen confirming that your submission has completed successfully. You can take a screen shot and keep the image 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 submitted to cssubmit 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 csmarks. **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 project specification. You should be able to complete this project by following examples in the lectures, textbook, and sample codes from the author Géron.

## 3 Data Download and Preparation

CIFAR-10 can be downloaded from the TensorFlow library. Details can be found on the webpage below:

https://www.tensorflow.org/api_docs/python/tf/keras/datasets/cifar10/load_data

The `load_data` function will return the training set and test set. You will need to normalize the pixel values so that they are in the range $0..1$.

## 4 Tasks

(i) Use an 85/15 random split on the training set to form a validation set. You can use the `train_test_split` function from the *scikit-learn* library with the `stratify` parameter so that the classes remain balanced in the resultant (smaller) training set and validation set.

Use the same training, validation, and test sets for training, validating, and testing both of your MLP and CNN models 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.

Manually experiment with **two** possible settings for each of the following hyperparameters:

- connection weight initialisation;
- learning rate scheduling (you can either use *callback* or use an appropriate optimizer);
- dropout rate.

You should use the validation set created above to help you find the optimal value for each of three hyperparameters above. In the final version of your Notebook file, do not include your hyperparameter tuning code; instead, you should describe the hyperparameter values that you experimented and your finding for the optimal hyperparameter values in a markdown cell.

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

(iv) **Structure of your code**

In the marking process, we want to have the following options to test your implementation of the MLP model:

(a) train your network from scratch for 100 epochs using the optimal hyperparameter values that you have found; or
(b) load your trained MLP model and train it for just 1 more epoch.

For option (a), we would temporarily move your MLP model directory to somewhere else. So your Python code should look something 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 with early stopping
      and the validation set using the optimal hyperparamter values;
    save the model to the directory described earlier.

use the model to perform predictions on the test set.
```

You should report and compare your trained MLP model's classification accuracies, F1 scores, and show the confusion matrices on both the training and test sets.

(v) **Implementation of a CNN**

Design a CNN that has 2 to 3 convolutional layers and a pooling layer between consecutive convolutional layers. Before the output layer, you will need to have 1 to 2 fully connected layers and maybe *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 manually explore **two** possible settings for each of the following three hyperparameters:

- kernel size;
- number of kernels[3];
- 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.

Same as before, describe the hyperparameter values that you experimented and the optimal values that you found in a markdown cell.

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

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

Follow the code structure mentioned above and repeat the same steps for your CNN model.

(vi) **Comparisons**

Finally, compare and comment on your MLP and CNN models on the test set, in terms of: classification performance (accuracy, F1 score, precision per class), model complexity (e.g., number of trainable parameters), and computation time[4].

Also, compare the two models by showing some example images from the test set that (i) both models correctly classified, (ii) both models incorrectly classified; (iii) MLP classified correctly but CNN didn't; and (iv) CNN classified correctly but MLP didn't.

## 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 complete your project in Google Colab. You will need to have a Google account (it is free to register for one). Upon completing the project, you will need to download your Notebook file and the two model directories, zip them, and submit the zip file to cssubmit.

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 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

Late submissions will be calculated as 5% of the total mark per day for the first 7 days (including weekends and public holidays) after which the project is not accepted.

## 7 Plagiarism

You should attempt the project by yourself. Collusion with (an)other student(s) is considered to be serious academic misconduct and can cause you to be suspended or expelled from the unit. Please see https://www.uwa.edu.au/students/my-course/student-conduct for more details.

[4] You can use any of the Python modules, such as `time` or `timeit`, to help you.