CITS5017 Deep Learning
Semester 2, 2024

Project 1

Assessed, worth 15%. Due: 11:59pm, Friday, 13th September 2024

1 Outline

In this project you will deal with a small colour image dataset containing 10 classes. Each image has 64×64 pixels. Although both the training set and validation set are perfectly balanced, there are only 200 training instances available for each class. Your tasks for this project are to train a CNN model, to retrain a pre-trained model through *transfer learning*, and compare the performances of these two models. 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). Please save your two trained models to the files **Surname_FirstName-CNN.keras** and **Surname_FirstName-MobileNetV2.keras**. You need to make sure that your models were saved in the same directory (i.e., not under a subdirectory) with your Notebook file, as this is what we will assume in our marking process.

Submit your three files above (a `.ipynb` file and two `.keras` model files 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 and your two model files for the project can be quite large. You need to ensure that your internet connection is stable during the entire submission 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 submission via email to the Unit Coordinator. Do **not** upload your submission onto Google Drive or OneDrive and then send the link to the Unit Coordinator. **Only files submitted to cssubmit will be marked.**

NOTE: Do not blindly copy examples that you found on the internet. Many of these examples are often too complex and they do not meet the project specification here. **Significant penalty** will be imposed on your mark if your code is found to be copied from the internet. You should be able to complete this project by following examples in the lectures, textbook, and sample codes from the author Géron.

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

3 The Image Dataset

Two pickle files, *train.pkl* and *val.pkl*, are supplied on the LMS page of the unit. Download these two files and save them in the same directory where you would develop your code. Use the following code to load the data:

```
def load_data(pickle_file):
    import pickle
    with open(pickle_file, "rb") as f:
        data_set = pickle.load(f)
    return data_set[b"data"], data_set[b"labels"], data_set[b"class_names"]

# now call load_data to form training and validation sets
X_train, y_train, class_names = load_data("train.pkl")
X_val, y_val, _ = load_data("val.pkl")
```

You should find that the shapes of `X_train` and `X_val` are $(2000, 64, 64, 3)$ and $(500, 64, 64, 3)$ respectively, i.e., there are 200 instances per class in the training set and 50 instances per class in the validation set. Both `X_train` and `X_val` are of type `uint8` (8-bit unsigned integer, just enough to represent any pixel value in the range $0 \cdots 255$). In the training (or validation) set, the first 200 (or 50) instances are for class 0, the next 200 (or 50) instances are for class 1 and so on. You should randomly shuffle[2] the training set (and optionally the validation set) to make sure that the minibatches used in the training process later on have a good mix of the 10 classes.

4 Tasks

(i) Write a small function called `displayImages` that would take in appropriate arguments so that it can be used to display 20 randomly sampled images from the arguments. This function should be called 2 times – for the training set and for the validation set. The figure displayed by the function should show the corresponding class name of each image.

(ii) **Implementation of a CNN**

Design a CNN that has 3 to 4 convolutional layers with appropriate activation function(s) and some pooling 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. Add one or two DropOut layers to your network. Use an appropriate function from `tensorflow.keras` to show a summary of your CNN architecture.

You should manually explore **two** possible settings for each of the following three hyperparameters:

- kernel size;
- number of kernels[3];
- the dropout rate.

Choose a suitable optimizer and a suitble learning rate scheuler to train your CNN. Use your validation set to help you find the optimal values for the hyperparameters above.

You should use the validation set above to help you find the optimal value for each of three hyperparameters above. In the final version of your Notebook file, copy your hyperparameter tuning code to a markdown cell and describe the hyperparameter values that you experimented and your finding for the optimal hyperparameter values in the markdown cell.

[2]for instance, using `numpy.random.permutation()` or `numpy.random.shuffle`.

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

(iii) **Structure of your code**

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

(a) train your network from scratch for 100 epochs using the optimal hyperparameter values that you have found above; or

(b) load your trained CNN model and train it for just 1 more epoch with a tiny learning rate (e.g., 10^{-5}).

For option (a), we would test your code by temporarily moving your CNN keras model file to somewhere else.

So, after describing your hyperparameter tuning process, your Python code should look something like this:

```
if the CNN keras model file 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 Surname_FirstName-CNN.keras
    show the learning curve plot for the training and validation set

use the model to perform predictions on the training and validation sets.
```

(iv) **Transfer learning using MobileNetV2**

- Use the pretrained model *MobileNetV2* from the `tf.keras.applications` package as your base model. Discard the last layer (or last couple of layers) and add on your own suitable layer(s) for the image dataset mentioned above. To decide what layers to keep and what you should add, refer to the example code in the Jupyter Notebook file for Chapter 14 from the author Géron of the textbook. You will need to inspect the pretrained model (using `model.summary()`, for instance) to find out its architecture, number of layers, name of each layer, etc.
- Since the base model is a well-trained model for the same image classification task, no hyperparameter tuning work is required. After constructing your deep network using the base model, display the **first and last 5 layers** of your network using the `summary()` function[4].
- Follow the training strategy as described in Géron's code. You will need to freeze layers from the base model, try your model for a few epochs first. After that, try unfreezing layer 90 onward of the base model and training for 30 epochs with a very low learning rate (e.g., 10^{-3}).
- Save your trained model to **Surname_FirstName-MobileNetV2.keras**.
- Apply the trained model to the training set and validation set and show the prediction accuracies and confusion matrices. Illustrate 10 examples from the validation set where the trained model made correct predictions and 10 examples from the validation set where it did not. You should make use of the `displayImages` function that you wrote earlier on for this task.

[4] You will need to use an appropriate optional argument.

(v) **Comparisons**

- Finally, compare and comment on the classification performances of your CNN model and the retrained MobileNetV2 model on the training and validation sets, in terms of: accuracy, F1 score, and precision per class. You should also compare the two models' complexity (e.g., number of trainable parameters), and training time per epoch[5] – You can use any of the Python modules, such as `time` or `timeit`, to help you.
- Also, compare the two models by showing up to ten example images from the validation set where[6] (i) both models correctly classified, (ii) both models incorrectly classified; (iii) CNN classified correctly but retrained MobileNetV2 did not; and (iv) retrained MobileNetV2 classified correctly but CNN did not.

5 Google Colab

The training of your models can 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, unfortunately, you may have to pay a small fee to use the resources there. You will need to have a Google account (it is free to register for one). You will also need to upload the two pickle files onto your Google Drive directory and then link the directory with your Jupyter Notebook file. Any extra code for doing the directory linking should be commented out before submission, as we can't access your Google Drive when we mark your Notebook file. Once you have finished the project, download your Notebook file and the two keras model files and submit them to cssubmit.

Only CPU is available in the default setting on Colab. To specify that you need a GPU or TPU, select the menu item *Runtime* and then the option *Change runtime type*. In the popped-up window, select "GPU" or "TPU" 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.

[5]As it is likely that the CNN model would need to be trained longer than the retrained MobileNetV2, it is more meaningful to compare the training times per epoch rather than the total training times. You can record the total training time for each model and then divide by the number of training epochs to get the training time per epoch.

[6]Note that it may happen that some of the four cases may have no examples, e.g., if your MobileNetV2 model did not misclassify any images that were correctly classify by your CNN model, then case (iii) would not exist.