ELEC5307 Deep Learning
Project #2: Challenge in Image Classification

Due: **29 Oct 2023 11:59PM**

The late penalty for project 2 is **5%** per day. However, if you submit the required zip file after **23:59 29 Oct 2023**, you will get zero.

1 Objectives

This laboratory aims to give you a chance to practice your knowledge in neural networks. You will build up a pipeline for fruit classification and try your best to improve the performance.

2 Dataset

The dataset (fruits) is defined by all the students and tutors. Each group takes more than 120 pictures for one kind of fruit. Tutors might add some new images and delete some your images.

After preprocessing operations, some images are released as the training-validation data, and some are used as the test set. The test data will not be released and used for testing your accuracy and inference time. You should split the released dataset into train and validation sets by yourself.

3 Experiments

Your task is to build a pipeline to classify fruits. You can use the predefined CNNs (AlexNet, GoogLeNet, ResNet, DenseNet, etc) to do the task, but the network structure for vision is not limited to CNNs nowadays. You can also use Transformer, MLP-Mixer, etc. Notice that deeper and larger networks will require larger memory and longer running time, so it is better to utilize GPU instead of CPU to run your network. Here are some hints (You can always come up with your own way).

3.1 Predefined Networks

Take AlexNet as an example. You may need to use the classes in **`torchvision`** for experiments. The source code can be found in https://github.com/pytorch/vision/tree/master/torchvision/models. For example, you can call the AlexNet class by:

```
net = torchvision.models.alexnet(), or
net = torchvision.models.alexnet(pretrained=True)
```

if you use **`pretrained=True`**, the code will download and use the model pretrained on ImageNet. For submission, you need to copy the network class from **`torchvision`** to a new file named 'network.py', name it as a new class called 'Network' and call it by:

```
from network import Network
net = Network().
```

In the 'network.py', you can modify the predefined network structure. However, if you want to use the pretrained models with your modified structure, you need to write a few lines of code to handle this case.

You should also know how to save and load model files, which contain all of the trained parameters. We prefer to use the following functions:

```
#  save the network to some file:
torch.save(net.state_dict(), 'filename.pth')
#  load the parameters to the defined network:
net = Network() # first define the network structure
net.load_state_dict(torch.load('filename.pth')).
```

3.2 Load ImageData

CIFAR-10 is a built-in dataset in **`torchvision`**. You can use it to build up your baseline before our dataset (fruit) release.

Our fruit dataset will be stored in one folder named '**5307Project2**', and under this folder, each category is stored in one folder with the name of the category's name. This is a standard structure for the class **`torchvision.datasets.ImageFolder`**. The basic usage of this command is:

```
from torchvision.datasets import ImageFolder
TrainSet = ImageFolder('./path/to/trainset', transform=train_transform)
ValSet = ImageFolder('./path/to/valset', transform=val_transform)
```

You should indicate at least the file path and the transformation requirements here. For more attributes, please check https://pytorch.org/vision/stable/datasets.html. The **`TrainSet`** and **`ValSet`** defined here could be taken as inputs by **`DataLoader`**. You are expected to split train and validation sets by yourself.

3.3 Use GPU

For the new task, you may need to train a deeper network with GPU. Using the GPU can accelerate your program. When using GPU in PyTorch, first you need to make sure your computer is capable of running CUDA (the package to enable running codes on NVIDIA graphic cards). If the following command prints '**`True`**', then you can use GPU to accelerate your program.

```
print(torch.cuda.is_available())
```

When using GPU, your whole network, the input images and labels should be moved to the GPU. This can be achieved by **`.cuda()`** method.

```
net.cuda()
images, labels = images.cuda(), labels.cuda()
```

If you need to move the tensors back to CPU (for example, you need to store the loss or print some weights on the screen), then you can use the method **`.cpu()`**.

4 Submission

You are supposed to finish this project within a group. Your submission should include the following files and follow the instructions:

- The python file "**network.py**", which contains the class 'Network' that defines your best network (method **`__init__`**) and forward process (method **`forward`**). You can include other network structures that you have tried.
- The python file "**project2_train.py**", which is used to train your network. We will not run it, but we will check your code. You can include tricks that you have tried.
- The python file "**project2_test.py**", which is used to test your result. We will **run it** with the trained

model (.pth) to test your accuracy and inference time.

- Other python files '*.py' (Optional).
- If you train your model on Colab, you can include **.ipynb files** instead of .py files, but you must include "**project2_test.py**" to make sure we can test with your trained model (.pth).
- The trained network parameter file '**project2.pth**'. This file should be too large (hundreds of MBs) to submit through *Canvas*, so you need to submit a text file named '**project2.txt**' that only contains a Google Drive/Dropbox link to the file '**project2.pth**'. We are able to check the timestamp of the file on Google Drive/Dropbox, so do not update your file after the deadline, otherwise, you will be penalised for late work.
- Your report '**project2.pdf**'. The report should have no more than 5 pages (main contents) for introduction, previous work, method, experiments and discussion, conclusion and future work. There is no page limit for the title page, reference, appendix, etc. When you write the authors, please write your names and your SIDs. You should also indicate group member contributions in the appendix. You can use your own template for the report, but please make sure they look reasonable.
- Your presentation video '**project2.mp4**'. The presentation should be no longer than 6 minutes which contains the main contents in your report (introduction, previous work, method, experiments and discussion, conclusion and future work). Please show all your group members' information (names and IDs) in the beginning of the presentation.

The files, including the python files, one text file (link to *.pth), one pdf file and one mp4 file, should be contained in a .zip file named '**project2_YourGroupNumber.zip**' with no spaces in the file name and submitted through *Canvas*. Please use two digits in the file name. For example, group 1 should write 'project2_01.zip' and group 20 should write 'project2_20.zip'. Your marks will be given according to your codes, written report, and presentation. For a detailed marking scheme, please refer to **Appendix: Marking Scheme.**

After your submission, we will run your codes and see the results of your model on the test dataset. We will release a ranking (top ten) of the **test accuracy** on *Canvas*, and the top three teams will be **given bonus scores** (five marks for the 1st group, four marks for the 2nd group, three marks for the 3rd group). The bonus count into your total mark for this course.

The Marking Scheme is on the next page

Appendix: Marking Scheme

The total mark for Project 2 is 20 in your final score, and your code, report and presentation account for 35%, 35% and 30% respectively.

code	correct submission files and correct naming	5%
	well commented codes & good coding style	5%
	good accuracy*	15%
	short inference time**	10%
report	introduction & previous work	5%
	method & experimental evaluation	10%
	discussion & conclusion and future work	5%
	good references & no writing typos	5%
	novel pipelines, networks, modules and tricks***	10%
presentation	introduction & method & experiment & conclusion	15%
	novel pipelines, networks, modules and tricks***	10%
	clear slides & good presentations	3%
	good time management	2%
punishment	cheating	-100%
	late submission per day	-5%
	insufficiently taken pictures for Pre-project2	-20%

* You can get a full mark (15%) if your accuracy is higher than 50% of the class.

** You can get a full mark (10%) if you inference time is shorter than 50% of the class. The inference time is the running time of your **project2_test.py** on a PC with a GPU.

*** We expect that you use new networks, modules and tricks to build up your pipeline. You can still include your reasonable attempts in your codes, report and presentation, even if they do not help the performance.

Extra bonus for top 3 groups (5 marks for 1st, 4 marks for 2nd, 3 marks for 3rd; marks are added to your final score of this course). This ranking is according to your testing accuracy.