ELEC5307 Deep Learning
Project #2: Challenges in Real-world Image Classification

Due: **08 Nov 2019** 11:59PM

1 Objectives

This project is a group project, which aims to give you a chance to practice your knowledge in neural network. In this project, Task 1 is compulsory, which is to let you design a CNN classification network. Task 2 is an extension of Task 1, which is optional. In task 2, you will add additional modules to reduce the domain gap in the source and target domains. Specifications are listed below:

1. TASK 1 Supervised image classification on **Fruit dataset** (20 marks):
    - Use PyTorch to load images and train a neural network for classification in GPU.
    - Implement pre-trained CNN structures and fine-tune the networks.
    - Try different tricks to achieve good performances on **Fruit validation dataset** (split from training) and **Fruit test dataset** (unavailable, not released).
2. (Optional) TASK 2 Unsupervised domain adaptation on **Office dataset** (**Bonus 5 marks**):
    - (1) Train an additional model or (2) train an unified model (together with the Fruit dataset model to classify all categories from both datasets) by using both **Office source dataset** (labelled) and **Office target dataset** (available but is trained without labels) to achieve good performances on **Office target dataset**.
    - Explore and explain the domain adaptation techniques you used to reduce the domain data distribution mismatch in the **Office source dataset** (source domain) and **Office target dataset** (target doamin).

2 Dataset

The two datasets are collected from the students and the public available Office-31 dataset[1].

For **Task 1**, in the **Fruit dataset**, 18 valid classes are collected include: apple, avocado, banana, blueberry, cherry, dragonfruit, grape, kiwifruit, lemon, orange, papaya, peach, pear, pineapple, plum, pomegranate, rockmelon, strawberry. **Training** dataset and **Validation** dataset are released with 2625 and 356 images, respectively. You can also do your own splitting besides the provided training/val split to better train your network as in Project 1. **Test** dataset is not released, which is used for ranking and marking.

For **Task 2**, in the **Office dataset**, 16 valid classes collected include: back_pack, bottle, calculator, desk_chair, headphones, keyboard, laptop_computer, mobile_phone, monitor, mug,

[1] Ganin, Y., *et al.* (2016). Domain-adversarial training of neural networks. The Journal of Machine Learning Research, 17(1), 2096-2030.

paper_notebook, pen, printer, scissors, speakers, stapler. There are two subset included, which are used for the two domains (**Source** and **Target**). All Source domain images are labelled images without noisy backgrounds and all the target domain images are unlabelled real-world images with different backgrounds. During training, you can use the images in both domains, but only the category labels in the source domain. If you want to evaluate the performance of the adaptation of your model, you can manually label 3-5 images for each class to see whether your models perform well. You are not supposed to train with the labels and to manually label all the images in target domain to see the overall performance.

3 Task 1 Experiments

In Task 1, your goal is to build neural networks to perform classification for the **Fruit dataset**. You can use the predefined neural networks to do the task, including AlexNet, GoogLeNet, ResNet (including ResNet-18, ResNet-50, ResNet-101), etc. However, you should notice that the deeper and larger network will require larger memory, and it is better to utilize GPU instead of CPU to run your network. You can use you own laptop, lab machines, AWS server or google colab.

When you use PyTorch to implement the neural network, you may need to use some techniques that are not discussed before. Here are some hints.

3.1 Predefined Networks

Since you are going to use the predefined networks, 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 should not directly use the network defined in **`torchvision`**. Instead, 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 layers. The Network is similar as the modified network class in Project 1. In addition, you should already know how to save and load model files, which is a '*.pth' file containing all 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

We have used CIFAR-10 for most of our experiments, and CIFAR-10 is a standard dataset included in **`torchvision`**. However in this project, we need to use the images collected by ourselves. In that case, we need to redefine the image dataset and loader.

The dataset now is stored in one folder named '**Task1**'(similar for Task2), 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/train/set', transform=train_transform)
TestSet = ImageFolder('./path/to/test/set', transform=test_transform)
```

You should indicate at lease the file path and the transformation requirements here. For more attributes, please check `https://pytorch.org/docs/stable/torchvision/datasets.html#imagefolder`. The **`TrainSet`** and **`TestSet`** defined here could be utilized using **`DataLoader`**.

3.3 Use GPU

Previously we mainly train the network on CPU. However, for the new task, you may need to train a deeper network, which could be suitable to use the GPU to 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, and 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 screen), then you can use the method **`.cpu()`**.

4 Task 2 Experiments (Optional)

The second task is optional, which is a much more challenging task for you to have a chance getting bonus marks. In this task, you will need to build a **domain adaptive** neural networks to improve the performance on the (target set of) Office dataset. You can use any techniques introduced in the lecture, in the tutorial or you find in the Internet. Please refer to the '**project2_da.py**' code for more details. As this is a bonus task, not many instructions will be provided.

5 Submission

You are supposed to finish this project within a group of 2 to 3 people. Your submission should include the following files and follow the instructions:

- The python file '**network.py**' (optional '**network_da.py**') which contains the class 'Network' that defines your network layers (method `__init__`) and forward process (method `forward`).
- The python file '**project2_train.py**' (optional '**project2_train_da.py**') which is used to train your network.
- The python file '**project2_test.py**' which is used to test your result. Please follow the given template to finish your test file.
- The trained network paramater file '**project2.pth**'(optional '**project2_da.pth**'). This file might be too large (hundreds of MBs) to submit through *Canvas*, so if you can't submit, you need to submit a text file named '**project2.txt**' that only contains a Google Drive link to the file '**project2.pth**'(optional '**project2_da.pth**') in your google drive. We are able to check the time stamp of the file on Google Drive, so do not update your file after the deadline, otherwise you will be punished as late work.
- Your report '**project2.pdf**'. The report should be no more than 6 pages (excluding references), which should include introduction, previous work, your backbone network definition and your improvement, experiments and analysis, and your conclusion and future work, etc. When you write the authors, please write your names and your SIDs. You should also indicate your contributions in the appendix. You can use the latex template in CVPR 2018 as your formatting reference: `http://cvpr2018.thecvf.com/files/cvpr2018AuthorKit.zip`.

The files (including the three python file, one text file, and one pdf file) should be contained in a .zip file named as '**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, your written report and your presentation. For 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. Before the last course, we will release a ranking on *Canvas*, and the top three teams will be given bonus scores. In Week 13, you will be given 3 minutes to do a presentation and answer questions.

Appendix: Marking Scheme

The general total marks for Project 2 is 20 in your final score, and your codes, report and the presentation account for 30%, 40% and 30% respectively. If you also complete Task 2, well demonstrate and analyze the domain adaptation techniques you use. You might get up to additional 25%, which equals to 5 bonus marks. The numbers in the following chart are in percentages.

Please note that the marking scheme may be updated in details.

codes	correct submission files and correct naming	10
	good performance [2]	20
	good domain adaptation performance [3]	20
	well commented codes	5
	proper improvement	5
report	introduction	5
	previous work	5
	implement the network and report results	10
	modifications to get better performance	5
	conclusion and future work	5
	good references	5
	no writing typos	5
	domain adaptation techniques you used	5
presentation	analyse of the network you are using	5
	clear slides and descriptions	10
	good answers to the questions (if any)	5
	to the impression of the lecturers [4]	10
punishment	cheating	-100
	late submission per day	-15
	bad coding	-20
	insufficient taken pictures for Pre-project2	-20

[2]For Task 1, You can get the full mark if you can get the performance better than 50% of the class.

[3]For Task 2, if you can demonstrate the effectiveness of using the domain adaptation techniques (better than not using it), you can get up to bonus 25% marks together with the report

[4]The bonus 10% will be added to the final score if there are any impressive presentation performance, i.e. online presentation, videos or animations.