ELEC 4806/5806 Introduction to Deep Learning and PyTorch

**Lab 3 Customize a Convolutional Neural Network for image classification tasks**

1. **Experiment goal:**

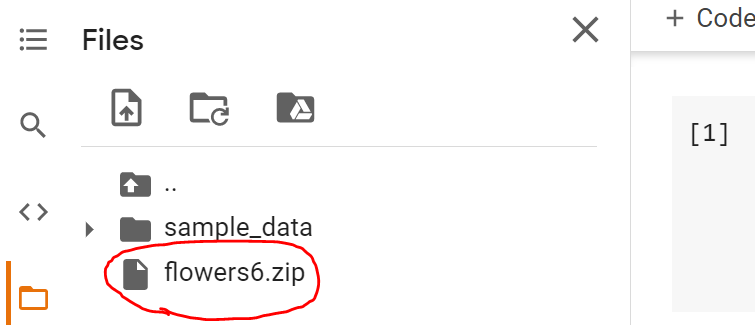
* Understand the structure of Convolutional Neural Networks.
* Master the use of hyperparameters.
* Master various optimization, regularization and weight initialization methods.
* Improve the performance of the model by fine-tuning the CNN network

1. **Introduction:**

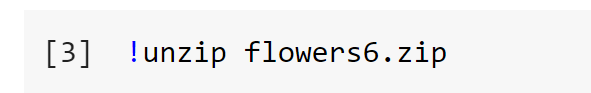
In this lab, you will customize a convolutional neural network for image classification tasks. You will use the provided flowers-6 dataset. It consists of 480 color images with different sizes in 6 classes (80 images per class). For each class, there are 73 training images and 7 test images.



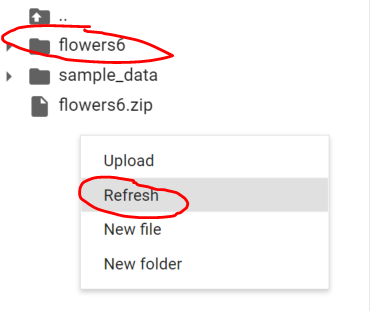
Please download the dataset “**flowers6.zip**” from Canvas and then upload it to the Google Colab online computer as shown below.



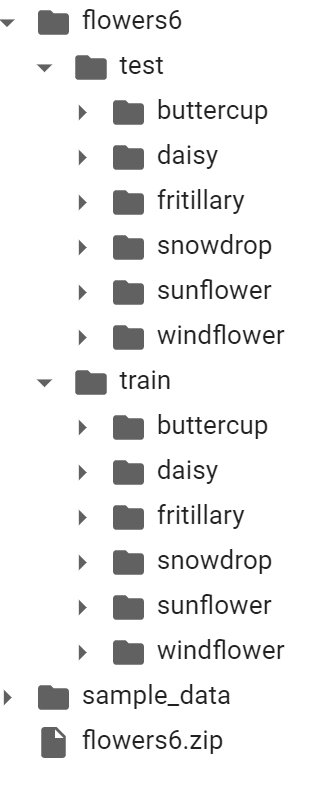
Then extract the zip file using the following command:



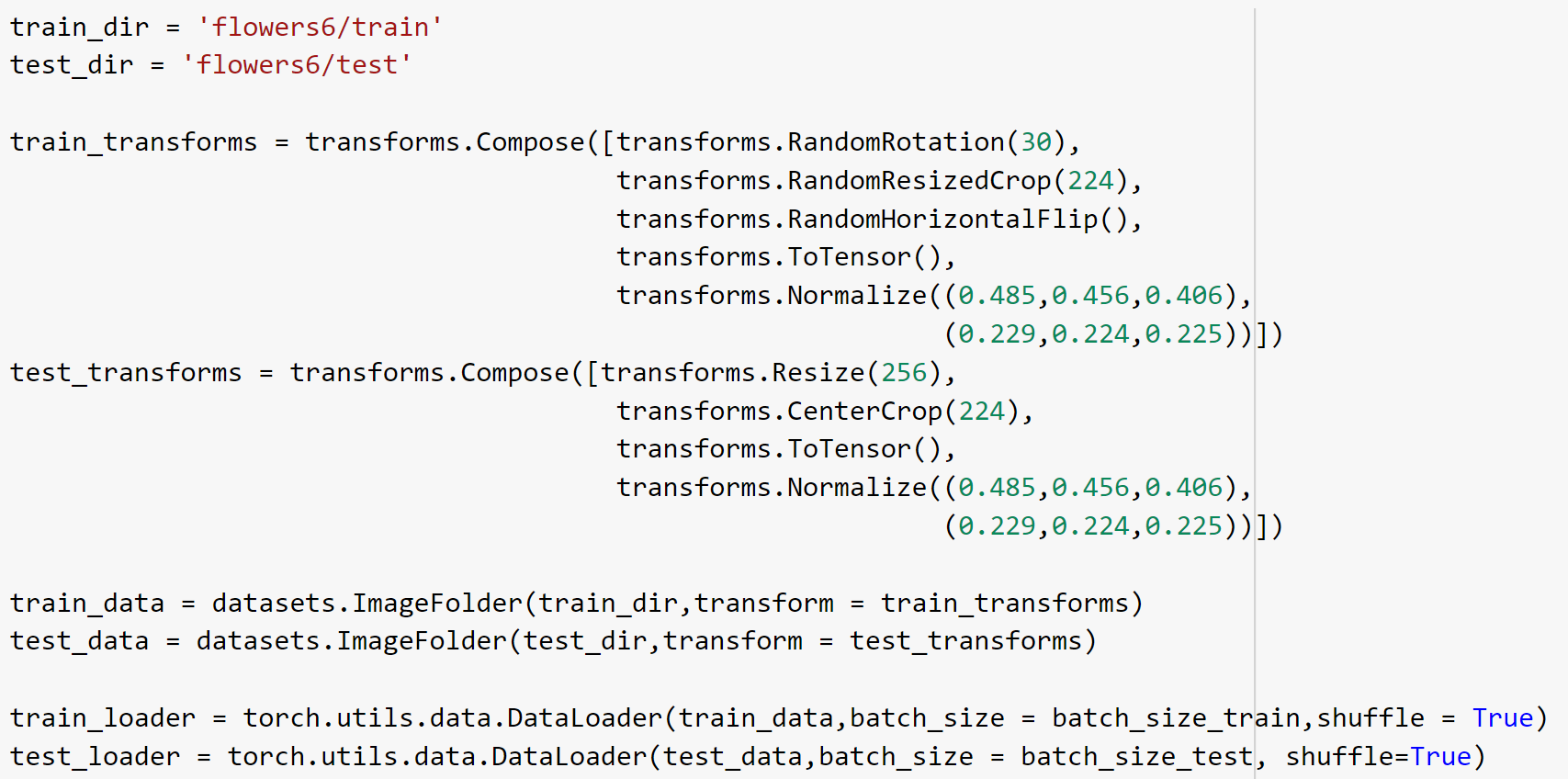
You can find the “**flower6**” folder under the current directory. If you cannot see what shows below, please refresh it.



And its sub-folders are as shown in the figure below. The first-level contains two subfolders "train" and "test", which are used to store data for training and testing respectively. Each of them has 6 secondary sub-folders, which are used to store 6 kinds of flower images.



**You can use the following code to construct the *train\_loader* and the test\_loader**. In this lab, the provided dataset is relatively small. However, training deep learning neural network models on more data can result in more skillful models. So here we introduce a powerful technique which is called “Data Augmentation”. Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. It can improve the ability of the fit models to generalize what they have learned to new images.



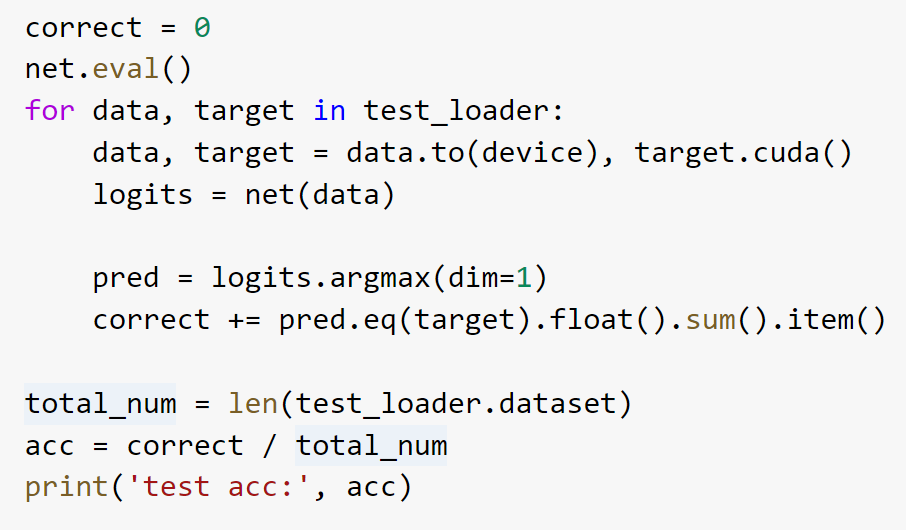
Pytorch provides very convenient API functions. You only need to add what the data augmentation you want in the *transpose.compos*e function. Here we use three forms: random rotation, random crop and random horizontal flip.

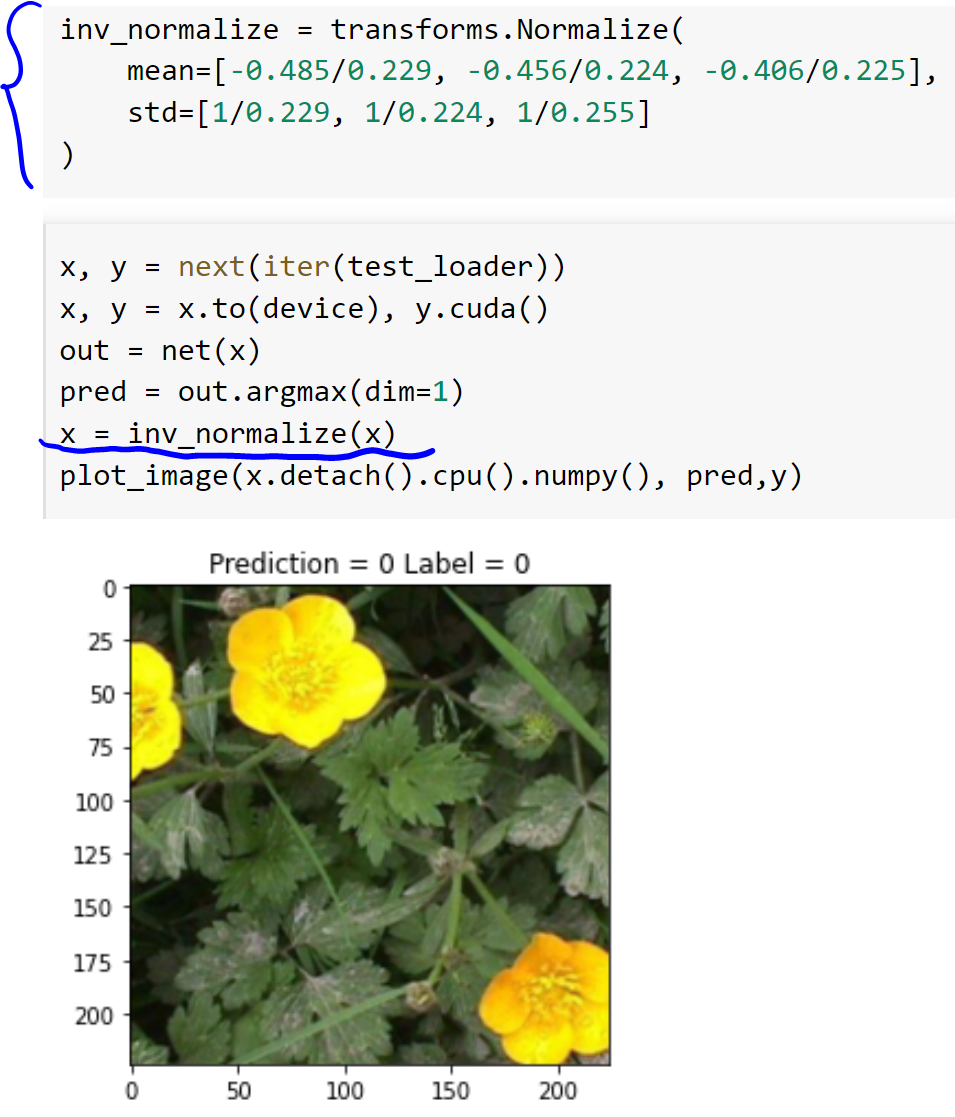
*ImageFolder* function assumes that all files are stored in folders, each folder stores images of the same category, and the folder name is the category name.

The CNN model requires the input images to have the same size. So the images here are all resized to 224x224.

Please refer to the **Lecture 7 and 8** in the Week 4 module to complete this lab. But you need to pay attention to the following:

* **The minimum requirement for model performance is that the test accuracy is greater than 82% (students registered ELEC 4806) and 85% (students who registered ELEC 5806)**
* To achieve good performance, you can consider:
  + Add more convolution layers and pooling layers.
  + You may have only 2-3 fully connected layers at the end of your model.
  + You can try to use a relatively smaller learning rate and train more epochs.
  + Choose a smaller batch size, otherwise you will encounter out-of-memory issue.
  + Try different combination of activation function, optimization method, and regularization method.
  + Please be patient fine tuning the model. This is the most basic requirement for engaging in deep learning work.
* Since the image data in this lab are all color images, you also need to modify some other codes, such as the “**plot\_image**” function, etc.
* When evaluating the model, you must first use the .eval() function as shown in the figure below, otherwise, batch normalization and dropout will also be used in the test. The performance of the model will be very bad.



* Since the three channels (RGB) of the image are all normalized when constructing the *test\_loader*, thus, it is necessary to inversely transform the data when displaying the image. Please refer to the code in the figure below.
* **Please use only what you learned from Lecture 7 and Lecture 8 to complete this lab**. **Plagiarizing code directly from other websites is not allowed**. At the same time, methods to improve model performance **not mentioned in Lecture6, 7 and 8 are not allowed to be used**. The purpose of this is to prevent plagiarism.
* Please try to use Google Colab to complete this Lab. Without GPU, the training process may become unacceptably long.

**Submission requirements (all team members need to submit):**

1. Please submit the **.ipynb file** and make sure the TA or the instructor can easily run your code. If you use google colab, you can download the .ipynb file by clicking “File🡪Download🡪Download .ipynb”. **Please include the names of all team members in the file name**.
2. Please submit a **pdf version** as well. You can use this website (<https://htmtopdf.herokuapp.com/ipynbviewer/> ) to convert your .ipynb file to pdf.
3. Please **record a video** to demo your work. In the video, **please indicate where you have changed the code and state the reason for the change.** **In addition, please describe in detail the process of modifying the model, adjusting the parameters, the problems encountered, etc.** And all team members must participate in the recording. If the team members cooperate remotely, they can submit multiple videos.
4. **When you are recording a video, if you just read the comments you wrote in the code, or read according to a script, then you will not be able to prove that the work was done independently by you. Therefore, the lab will be judged as plagiarism.**