**Name: LI Yuyao**

**Student ID: 12615034**

**Tutorial Session: T01 Thu 14:00-16:00**

* **train.ipynb:** data preparation and training process implement code
* **test\_model.ipynb:** testing process implement code
* **best\_model.path:** train model path
* **model\_best\_checpoint.path:** best checkpoint in the model
* **test\_images folder:** include 6 images to test the model(2 nike, 2 converse, 2 adidas)
* **Nike\_Adidas\_converse\_Shoes\_image\_dataset:** train and test dataset
* **results.txt:** predicted result
* **code.txt:** testing and training commands
* **Project Report**

**Data preparation:**

**Install the Pytorch in the colab and necessary modules.**

**Connect to the google drive to read the dataset:**

1. ‘train’ folder as the training dataset path
2. ‘test’ folder as the test dataset path.

**Calculate the mean and standard deviation of the dataset:**

1. Define the transform size(240 \* 240) and convert it to tensor
2. Make training transform to equal to the training data path
3. Define a function(get\_mean\_and\_std) to calculate the mean and std.

**Dataloader and transform:**

1. Transform the train and test dataset using transforms.Compose(reduce the average size(240\*240), randomly flip the image horizontally, apply the random rotation to a 10 degrees, use mean and tad to normalisation)
2. Load the train and test dataset by specifying the path and apply the transform(torchvision.datasets.ImageFolder)

**Show the random transform images:**

1. Set the batch size equal to 32
2. Define a function(show\_transformed\_images) to show the 32 random images

**Training process implementation:**

**Define the set the device function**(set\_device)

**Model using:**

1. Resnet18 model
2. models.resnet18(pretrained = False) start with the random weight
3. Resnet18\_model.fc.in\_features size of each input sample
4. 3 classes(nike, converse, adidas)
5. nn.Linear(num\_firs, number\_of\_classes):Linear function, take the number of inputs and features as parameters and prepare the necessary matrices for forward propagation
6. Set the device.
7. Set the cross entropy as loss function nn.CrossEntropyLoss()
8. Optimizer: SGD optimizer, set the learning rate, optimization algorithms and extra error to loss function, prevent the overfitting

**Train the neural network:**

1. Define the train neural network function(train\_nn):
   1. For every epoch track all the information
   2. For data in train loader, make images and labels to device and calculate how many images are there in the batch（total += labels.size(0)）, set the gradients to zero(optimizer.zero\_grad()), get all the output from minimal badge(outputs = model(images)).
   3. Prediction 1 dimension to prejudice
   4. Set the loss function
   5. Back propagate to calculate the weight gradients
   6. Calculate the epoch accuracy
   7. Print the training dataset accuracy message
   8. Find the checkpoint
   9. Return model

**Define the checkpoint function**(save\_checkpoint(model, epoch, optimizer, best\_acc)) to save the best accuracy.

**Evaluate the model on the test dataset:**

1. Define the (evaluate\_model\_on\_test\_set(model, test\_loader))function, reduce the memory usage and speed up the computations.
2. Prediction
3. Print the test dataset accuracy message.
4. Return the dataset accuracy

**Train the model 150 times**

train\_nn(resnet18\_model, train\_loader, test\_loader, loss\_fn, optimizer, 150)

Find the best accuracy is 73.6842105263158

Epoch is 112

**Save the model in google drive**

**torch.save(resnet18\_model, './gdrive/My Drive/COMPS492F/Project/best\_model.path')**

**Testing process implementation:**

Test the model using the image folder named **test\_image**

**Connect to the google drive**

from google.colab import drive

drive.mount('/content/gdrive')

**Load the model**

model = torch.load('./gdrive/MyDrive/COMPS492F/Project/best\_model.path')

model.eval() # evolation model

**Image preprocessing, load the images and predict**

Using transforms.Compose to set the transform of image dataset.

For test the dataset load the images in the test\_images folder.

**Save result to txt file(result.txt)**

Using the np.savetxt to save the predicted result.