Lab 2: Classifying Remote Sensing Images with Data Augmentation and Transfer Learning

In this lab, you will perform data augmentation and transfer learning yourself to classify remote sensing images.

If you cannot familiar with using PyTorch to develop CNN models for image classification, please read this good PyTorch code example - PyTorch CIFAR10 tutorial:

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

Task 1: Dataset download and preparation (code already included in the template). In Google Colab, use *wget* to download the remote sensing image dataset from http://weegee.vision.ucmerced.edu/datasets/UCMerced_LandUse.zip

Task 2: Training and test data splitting (10 pts). Use *torchvision.datasets.ImageFolder()* to load the data, and split the data into 80% for training and 20% for testing (set the image size to 96 x 96). Plot out any 9 images of your choice from the training data to give a preview of the dataset.

[Chintan: Please write the answer key based on https://github.com/GeneralLi95/PyTorch_UCMerced_LandUse/blob/master/main.py]

Task 3: Data augmentation (40 pts). Use at least four different ways of your choice to augment the training dataset. Feel free to set your own probabilities for different augmentation methods (e.g., you may prefer to apply one image processing method more frequently than another). After the augmented dataset is generated, concatenate it with the initial training data.

The code might looks like this:

```
# img_size is the size of an input image
import torchvision.transforms as transforms
import torchvision.datasets as datasets
augmentation = [
    transforms.Resize(img_size*2),
    transforms.RandomResizedCrop(img_size, scale=(0.2, 1.)),
    transforms.RandomApply([
        transforms.ColorJitter(0.4, 0.4, 0.4, 0.1)  # not strengthened
    ], p=0.8),
    transforms.RandomGrayscale(p=0.2),
    transforms.RandomApply(
        [moco.loader.GaussianBlur([.1, 2.])], p=0.5),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    normalize
```

```
]
transmform = transforms.Compose(augmentation)
```

Task 4: Train a simple CNN using the augmented data (30 pts). You can design the CNN of your own choice but it should contain at least 2 convolutional layers, 2 max pooling layers, 1 dropout layer, and 1 dense layer. Use 'adam' optimizer, cross entropy loss, and accuracy for metrics (Check out the demo code provided in Lecture 5: https://colab.research.google.com/drive/1JNIlZvKnIOE0k93UjrQgHTN8M2qqkAd3?usp=sharing). Train your model on the augmented dataset for 10 epochs. Evaluate the performance of your CNN model on the test dataset by measuring its accuracy.

Helpful notes:

1) Check out this link: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

Task 5: Apply Inception V3 to do transfer learning on augmented data (20 pts). Find Inception V3 model on torchvision model Hub: https://pytorch.org/vision/stable/models/inception.html. Use the same optimizer, loss function, and accuracy setting, as used in Task 4. Train your transfer learning model on the augmented dataset for the same number of epochs (see demo code in Lecture 6: https://colab.research.google.com/drive/1hDVV38VoTdJdNnX-zSewagL94ajVUzPX?usp=sharing). Evaluate the performance of this transfer learning model on the test dataset by measuring its accuracy.

Helpful notes:

1) If you want to learn more about the model architecture of Inception V3, you can read their original CVPR 2016 paper: https://arxiv.org/abs/1512.00567v3.
2) This PyTorch ResNet18 Transfer Learning tutorial will be very helpful: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html, your code will be very similar to those code in "ConvNet as fixed feature extractor" section. The only difference is that we use Inception V3 instead of resnet18.

To submit:

- Put your Google Colab link in Github Classroom and then submit