Machine learning Projects Analysis

1. Local image folders download methods:

In our projects, I used two different download methods:

(1). Iterate through the photo folders such as MNIST by Python.

(2). Use torchvision.datasets.ImageFolder method to download the flower folders.

Compared with the second one, the first method is very time-consuming and inefficient.

2. Data indices for training and validation splits:

In our data folders, we don't have any validation data set. So I just split the original training data set to new training dataset and validation dataset.

And it has been found necessary to make the validation data set for the training process.

3. Scratch CNN architecture VS Transfer Learning CNN architecture:

I used two different deep learning algorithms to solve the Flower Classification projects. From the results, it's clear that the Transfer Learning CNN model (such as VGG 16 or VGG 19) can perform much better than the Scratch CNN architecture.

4. The procedure for preprocessing the data

(1). For the Transfer Learning Model:

I resized all the trainloader, validationlocader and testloader images to 224 * 224 pixel as required by the pre-trained networks (VGG 16).

(2). For the Scratch CNN Model:

I resized all the trainloader, validationloader and testloader images to 224 * 224 pixel, center cropped, to avoid overfitting of the model. The input data should be resized to 224x224 pixels as required by the pre-trained networks (VGG 16). I also nomalized the images for each color channel, the means are [0.485, 0.456, 0.406] and the standard deviations are [0.229, 0.224, 0.225].

In fact, we should use a little bit different Transforms Strategies for the training, validation and test data set. Such as, we should use a little bit more Transforms methods for training data to avoid overfitting. However, the validation data set in this case also came from the training data set, so that's why I just use same Transforms Strategies for all three data sets.

5. The steps for building Transfer Learning Model

(1). I Load in a pre-trained VGG16 model. VGGNet is great because it's simple and has great performance.
(2). I "Freeze" all the parameters, so the net acts as a fixed feature extractor. I keep all the convolutional layers and Freeze training for all "features" layers.
(3). I remove the last layer.
(4). I replace the last layer with a linear classifier of our own. I only replace the final fully-connected layer with our own classifier. I replace the last fully-connected layer with one that produces the appropriate number of classes (5 in this case).

VGG13 and VGG16 work very well for feature detectors of images they were not trained on. So that's why I'll use transfer learning to train a network that can classify the flowers images.

7. Three possible points of improvement for my algorithm in the future

(1). Increase the training images, only the flower is inside the image and the background has not so much details

(2). Try different pre-trained models, and expand / modify the classifier full connected layers to train the images better (dropout layers, dimensions, ...)

(3). Increasing number of epochs (such as 10 for transfer learning model)

My computer is not compatible with NVIDIA driver, so I can't use GPU in my computer. I can use Google Colab free GPU, however, Pytorch torchvision.datasets.ImageFolder download method can't use the Google Drive Path. So I have to use CPU in my computer.

In terms of the Transfer Learning Algorithm for flower classifications, I only set the epochs equal to 4 due to the long training time. In my experience, we can get testing accuracy which is more than 83% if we set epochs equal to 10.

8. I also sent the Jupyter Notebook files for the visualization.