README

Section 1: Running the Experiments

Requirements

1. Python 3.6
2. Pytorch 0.4.0 or 0.4.1
3. CUDA 8.0 or higher

Guide

- After downloading the code zip file, upload the DCL code folder to your Google Drive.
- Change the runtime type to V100 while training the DCL model for 60 epochs and select the High RAM option.
- Open the AI_Proj_DCL.ipynb file in Google Colab and run the cells according to the instructions provided for the cells.
- The dataset will be downloaded by running the Colab cells.
- Once the dataset is moved to the destination location - dataset/CUB_200_2011, make sure that the dataset is organized such that
    - The annotations are in dataset/CUB_200_2011/anno and
    - The images are in dataset/CUB_200_2011/data.
- Open the terminal and cd to the location of the code folder.
- Use the training the following command to train the model:
    - To train ResNet-50 for 60 epochs with a training batch size equal to 16 and a validation batch size equal to 16, run the following command on the terminal:

```
python train.py --data CUB --epoch 60 --backbone resnet50 \
                --tb 16 --tnw 16 --vb 16 --vnw 16 \
                --lr 0.001 --lr_step 10 \
                --cls_lr_ratio 10 --start_epoch 0 \
                --detail training_descibe --size 512 \
                --crop 448 --cls_mul --swap_num 7 7
```

- To train SE ResNet-50 for 60 epochs with a training batch size equal to 16 and a validation batch size equal to 16, run the following command on the terminal:

```
python train.py --data CUB --epoch 60 --backbone se_resnet50 \
                --tb 16 --tnw 16 --vb 16 --vnw 16 \
                --lr 0.001 --lr_step 10 \
                --cls_lr_ratio 10 --start_epoch 0 \
                --detail training_descibe --size 512 \
                --crop 448 --cls_mul --swap_num 7 7
```

- To run the test.py file after training the DCL model, run the following command in the terminal by replacing the location of the weights you intend to use for testing:

```
Python test.py --save
/content/drive/MyDrive/ECE570/DCL-master/net_model/training_des
cibe_111522_CUB/weights_38_312_0.0403_0.1060.pth --acc_report
```

Section 2: Description

a. Which code files have been copied from other repositories with references to these repositories

- I used the DCL codebase on GitHub by JDAI-CV.
  Link: https://github.com/JDAI-CV/DCL

- The choice of a batch size of 16 for training the CUB dataset and the hue value of 0.4 for ColorJitter transformation was made based on the parameters used in configs/cub.yaml in the GitHub Repository, Mobulan in the IELT project which implements the Fine-Grained Visual Classification via Internal Ensemble Learning Transformer paper.
  Link: https://github.com/mobulan/IELT/tree/main

b. Which code files have been modified and how they have been modified?

The changes I made to the existing code are as follows:

- I experimented with the **swap function** in the **transforms/functional.py** file to see how the different ways of shuffling would affect the performance of the DCL model and visualized the images before and after swapping.
    - The first method I used for this comparison is the algorithm used by the authors in their codebase where they use a fixed value of k equal to 1 and RAN equal to 2 (where k is a tunable parameter that defines the neighborhood range and *RAN by definition provided in the paper for r, is a random variable from a uniform distribution between -k and k)
    - To introduce randomness, I modified RAN by assigning it a random integer between -k and k for k=1.
    - I wrote a shuffling mechanism from scratch using the paper's explanation for RCM hoping to improve the accuracy of the DCL model.
- The authors have a **ColorJitter class** written in **transforms/transforms.py** that randomly changes the brightness, contrast, and saturation of the images. However, they never use this class to transform their training datasets. To understand if they excluded it because it slows the training or it is not useful for learning, I changed the training and validation dataset transformation in **config.py's load_data_transformers()** to include ColorJitter under the **"common aug" transformation**. I provided the brightness, saturation, contrast, and hue inputs as 0.6 for the ColorJitter class since it was written in a way that it does not accept range values for the above four parameters.
- To optimize certain parts of the code, I performed the following experiments with the code:
    - To optimize the ColorJitter class mentioned above, I replaced this class with the in-built ColorJitter function from **torchvisions.transforms** and provided the parameters for brightness, saturation, and contrast as a range [0.6, 1.4] and 0.6 for hue (since hue does not accept a range input. The values I used for the above parameters are based on the studies shown in the paper, Bag of Tricks for Image Classification with Convolutional Neural Networks, where under the Baseline Training Procedures section they mention the values that should be used for scaling brightness, saturation, and contrast for improved results of ResNet-50 trained on the ImageNet dataset.

    - In the **transforms/functional.py** file's **to_tensor function**, the authors mention in their comments that the Height-Width-Channel (HWC) transformation to Channel-Height-Width (CHW) takes up 80% of the loading time. I tried to optimize this conversion by replacing their code that uses transpose() twice followed by contiguous() with a single permute() followed by a contiguous().
    - I tried improving the results of the DCL model by using the SeResnet-50 model pretrained on the ImageNet dataset. This model uses squeeze and excitation blocks which are known to provide better results than ResNet50. I added the pretrained SE ResNet-50 model in the **models/pretrained directory** after downloading it from this link and changed the "pretrained_model" dictionary in the **config.py file** to include this model.
- I changed the test code based on the code provided by a GitHub user, Lamborghini1709, under the "Issues" called "**New test.py**". This was done because the author's **test.py** file seems to be buggy and deprecated.

c. Which code files are the student's original code?

The code that I implemented from scratch:

- I have written a setup code in the **AI_DCL_Proj.ipynb** file that has to be executed before running the author's code. The setup code includes changing package versions, importing certain packages, downloading the CUB dataset, and extracting all its files to a specified location on my Drive folder. I have also written a code snippet to count the number of files in the Dataset folder to ensure the issue I had in my initial implementation of losing images every time I trained can be avoided by checking if I have the correct number of files before starting the training.
- I implemented the swap function in the **transforms/functional.py file** to perform the Region Confusion Mechanism (RCM).
- I wrote the code to visualize the images before and after performing RCM, i.e., shuffling the images, inside the **swap function** in the **transforms/functional.py** file.

- I added the pretrained SE ResNet-50 model in the **models/pretrained** directory after downloading it from this link and changed the **"pretrained_model"** dictionary in the **config.py** file to include this model.

Section 3: Description of the Dataset Used

1. Download the **CUB_200_2011** dataset zip file from the Caltech Data by Caltech Library using this link. The zip file size is 1.2 GB.
2. Upload the downloaded zip file to your designated location in Google Drive.
3. Execute the code in the **AI_Proj_DCL.ipynb** notebook to extract the files from the uploaded zip file.