Online Contactless Palmprint Identification using Deep Learning (Matching Part)

This is a Pytorch implementation for matching part of Online Contactless Palmprint Identification using Deep Learning.

We provide two architecture based on the proposed Residual Feature Network (RFN), which we name them as RFN-32 and RFN-128. The RFN-32 is designed with larger receptive field on first conv layer while RFN-128 is designed with wider residual block. It is found that RFN-32 with larger receptive field at first has a better performance on CASIA cross database test. So, in short, **please use `--model RFN-32` when training for CASIA crossdb test and uise `--model RFN-128` for others**.

1.Setup Requirement

- Pytorch
- Numpy
- Pillow
- Scipy

It is recommended to use Anaconda Python2.7, since you only need to install Pytorch manually.

Steps:

1. Install anaconda python2.7 from https://www.continuum.io/anaconda-overview
2. Install pytorch with version 0.2.0 (conda install pytorch=0.2.0 -c soumith), earlier version may not work

2.Training Procedure

- Change directory to code folder

```
cd /palmprint-recog/code/
```

By default, you can trained with **IITD Left** and tested on **IITD Right** (WithinDB test), **CASIA, PolyU 2D/3D, 300 Subject, 600 Subject** (CrossDB test) The IITD Left training command line is as follows:

```
python main.py --db_prefix IITD --train_path ../dataset/IITD5/IITD\ Left
--shifted_size 5 --alpha 10
```

The best trained model for each dataset is stored in `./checkpoint/selected-best-ckpt/`, you can directly run protocol test based on these `.pth` files.

It is worth to mention that the best model for CASIA is from RFN-32 architecture, the command line is as follows:

```
python main.py --db_prefix IITD --train_path ../dataset/IITD5/IITD\ Left
--shifted_size 5 --alpha 10 --model RFN-32
```

Other Options

- `--checkpoint_dir` allows you to specifically set the saving directory of your model, which by default is './checkpoint/'
- `--start_ckpt` allows you to train on dataset based on the chosen the checkpoint model
- `--epoch` specifies the maximum epoch number during training.
- `--log_interval` specifies the **epoch frequency** to logging the training loss (default is 1)
- `--checkpoint_interval` specifies the frequency of saving the checkpoint model (default is 10)

You need to specify the path of training dataset if you want to train with other dataset. please run `python main.py -h` to see a list of all options

3.Protocol Test

There are four pretty designed protocols for the testing dataset under protocol folder. It is worth to mention that `polyu2d_roc.py` is specially designed for PolyU 2D/3D Contactless dataset due to its special data formating. You can also use `twos_roc.py` if you have formatted the dataset correctly (first 5 in session1, second 5 in session2).

For all the protocol files, **you need to specify the path to the testing dataset**, please use `python ./protocols/XXX.py -h` for more details.

`twos_roc.py`: Two Sessions protocol (**300 Subject, 35 Subject**)

`iitdlike_roc.py`: IITD-like protocol (**600 Subject, IITD Right**)

`nfolds_roc.py`: All-to-All protocol (**CASIA**)

`polyu2d_roc.py`: Specially designed for PolyU 2D/3D contactless protocol (**PolyU 2D/3D**)

The `path-to-model-ckpt` is the `.pth` file that you have stored during your training, and `dest-to-store-npy` is the file that the stored score file from each protocol test, which you can later plot with the plot function below. For example, you can name your `dest-to-store-npy` as `./test.npy`.

[1] **Test on IITD Right:**

```
python protocols/iitdlike_roc.py --test_path ../dataset/IITD5/IITD\ Right/
--model_path ./path-to-model-ckpt/ --out_path ./dest-to-store-npy --save_mmat True
```

[2] **Test on CASIA:**

```
python protocols/nfolds_roc.py --test_path ../dataset/n4CASIA --model_path
./path-to-model-ckpt/ --out_path ./dest-to-store-npy --save_mmat True
```

[3] **Test on PolyU 2D/3D Contactless:**

```
python protocols/polyu_roc.py --test_path ../dataset/PolyU2D --model_path
./path-to-model-ckpt/ --out_path ./dest-to-store-npy --save_mmat True
```

[4] **Test on the left part of unreleased dataset with 600 subjects:**

```
python protocols/iitdlike_roc.py --test_path ../dataset/600-subject/L/ --model_path
./path-to-model-ckpt/ --out_path ./dest-to-store-npy --save_mmat True
```

[5] **Test on the left part of unreleased dataset with 300 subjects:**

```
python protocols/twos_roc.py --test_path ../dataset/300-subject/L/ --model_path
./path-to-model-ckpt/ --out_path ./dest-to-store-npy --save_mmat True
```

[6] **Test on unreleased dataset with 35 subjects:**

```
python protocols/twos_roc.py --test_path ../dataset/35-subject --model_path
./path-to-model-ckpt/ --out_path ./dest-to-store-npy --save_mmat True
```

The option `--save_mmat` True means saving matching matrix

4.Plot

For each of the targeting dataset, we prepare a plot function for each of the test dataset(*specifies the x,y coordinates*), which you can plot choose to plot by your own. The example of plotting for IITD using the score file `./test.npy` stored by protocol test is as follows:

```
python plot/roc_iitd.py --src_npy ./test.npy --dest ./test.pdf --label RFN-128
```

It is worth to mention that there are no differences between each plotting function except the difference choose of the x, y coordinates. Please use the correct plotting files for plot protocol test result on different dataset, i.e. `roc_casia.py` for plotting casia file.

5.Put It Together

It would be nice if we put all the procedure together. So let's take training with **IITD Left** and testing on **IITD Right** as an example.

- Change directory to code folder

```
cd /palmprint-recog/code/
```

- Training with prepared dataset

Please make sure there are `IITD5` under `/palmprint-recog/dataset/` folder

```
python main.py --db_prefix IITD-Left --train_path ../dataset/IITD5/IITD\ Left
--epochs 500 --alpha 10 --model RFN-128 --shifted_size 5
```

There will be a folder name `IITD-Left_a10s5mRFN-128_XXX` in the `./checkpoint/` folder, which `XXX` is the datetime.

- Use training checkpoint to test on IITD Right with IITD-like protocol

Let's take epoch 300 as an example, `--save_mmat` is used when plotting CMC

```
python protocols/iitdlike_roc.py --test_path ../dataset/IITD5/IITD\ Right
--model_path ./checkpoint/IITD-Left_a10s5mRFN-128_XXX/ckpt_epoch_300.pth --out_path
./output/IITD-Right-test.npy --shift_size 5 --save_mmat True
```

- Using the generated .npy file to plot ROC and CMC

ROC:

```
python plot/roc_iitd.py --src_npy ./output/IITD-Right-test.npy --dest
./output/IITD-Right-test.pdf --label RFN-128
```

CMC:

```
python plot/cmc_iitd.py --src_npy ./output/IITD-Right-test.npy --dest
./output/IITD-Right-test_cmc.pdf --label RFN-128
```