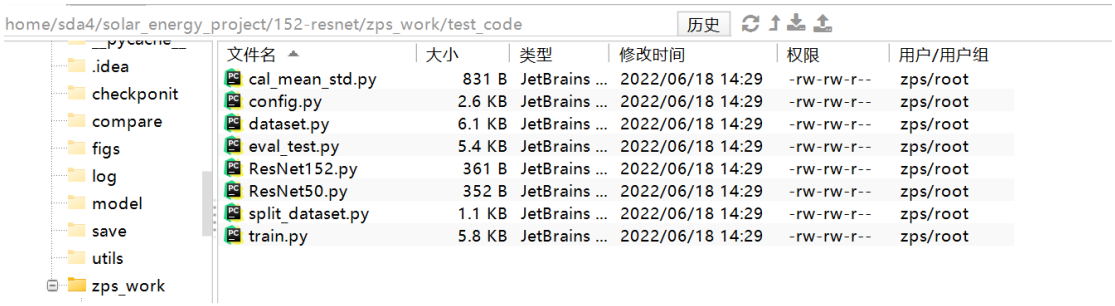
Solar project code user manual

First, move to the directory where the current code is located.



1. Calculate the mean and standard deviation of the original training set

Command：

python cal_mean_std.py --cal_mean_std_path /home/sda4/data/0min-300s-regular/DNI/LR/train --mask 0

Where: cal_mean_std_path is followed by the path of the original data training set, and mask is followed by whether the image needs to be masked (0 is not masked, 1 is masked).

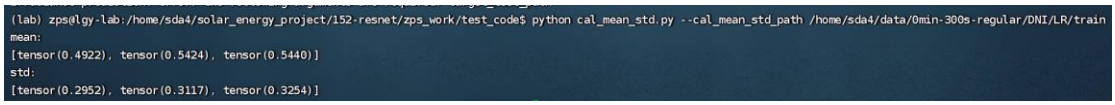
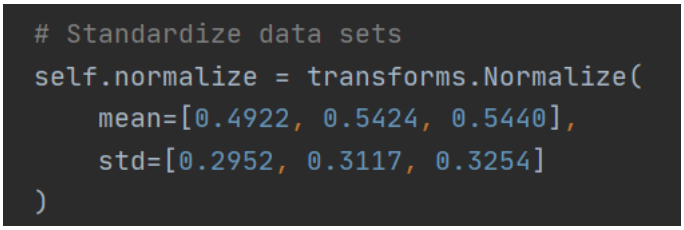


Figure 1. calculate mean and std values

2. Modify mean and std of dataset.py file

Enter dataset.py, modify the mean and std in the MyDataSet and MyDataSetTest classes to the values shown in Figure 1. Remember that both classes need to be modified.



```
# Standardize data sets
self.normalize = transforms.Normalize(
    mean=[0.4922, 0.5424, 0.5440],
    std=[0.2952, 0.3117, 0.3254]
)
```

Figure 2. location of dataset.py to be modified

3. Re divide the data set

A folder “data” for storing new data sets can be created in the current directory, and

three subfolders of train, val and test can be created in the “data” folder to store new training, verification and test sets.

Command：

python split_dataset.py --source_train_path /home/sda4/data/0min-300s-regular/DNI/LR/train --source_test_path /home/sda4/data/0min-300s-regular/DNI/LR/test --target_train_path data/train --target_val_path data/val --target_test_path data/test

Where: source_train_path followed by the location of the original data training set, source_test_path followed by the location of the original data test set (the original data has only two folders of training and test sets), target_train_path followed by the path where the new training set is stored, target_val_path followed by the path where the new verification set is stored, target_test_path followed by the path where the new test set is stored.

Figure 3. dividing data sets

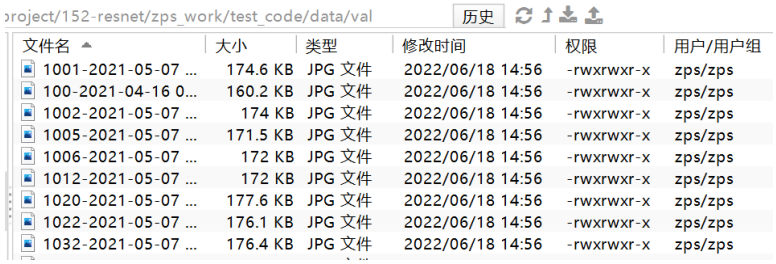


Figure 4. new data set

4. Model training

Command：

python train.py --data_type ghi --train_path data/train --val_path data/val --model_name ResNet152 --save_path result/train_info --mask 0 --num_workers 4

Where: data_type followed by the label to be trained (GHI, DHI, DNI), save_path followed by the path to save the model and training information, mask followed by whether the image needs to be masked (0 is not masked, 1 is masked), num_workers is followed by how many threads are used to read data. The more threads are used, the

faster the reading speed is.

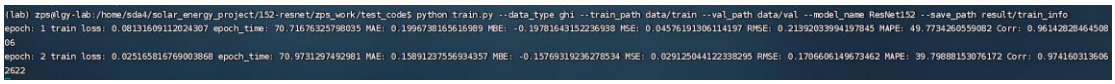


Figure 5. model training

5. Model test (prediction)

In the model saving path (such as result/train_info), find the latest saved model (the one with the highest round), such as result/train_info/ResNet152_model_5.pt (stop training after only 7 times, for demonstration purposes).

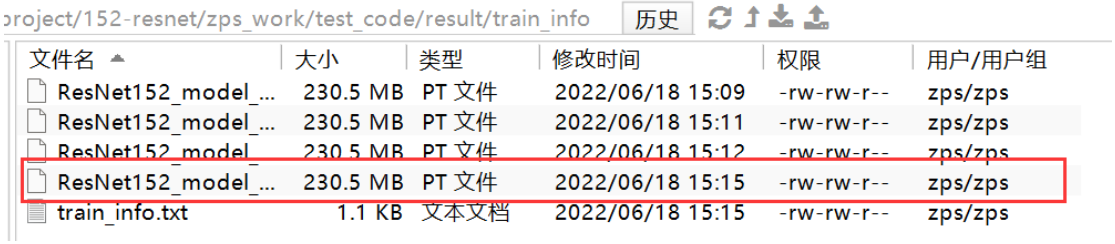


Figure 6. The later the model is saved, the better.

Command：

python eval_test.py --load_model result/train_info/ResNet152_model_5.pt --data_type ghi --test_path data/test --model_name ResNet152 --save_path result/test_info --mask 0 --num_workers 4

Where, load_model followed by the path of the model saved, data_type followed by the predicted label (consistent with the training), save_path followed by the path where the prediction results and indicators are saved, test_path is followed by the path of the test set, mask is followed by whether the image needs to be masked (the same as the setting during training), num_workers followed by the number of threads used to read data.

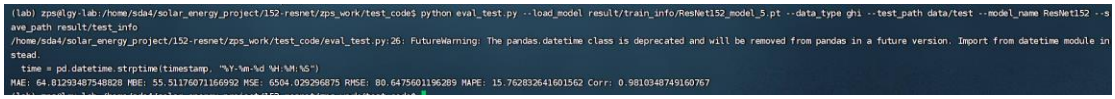


Figure 7. model test (prediction)

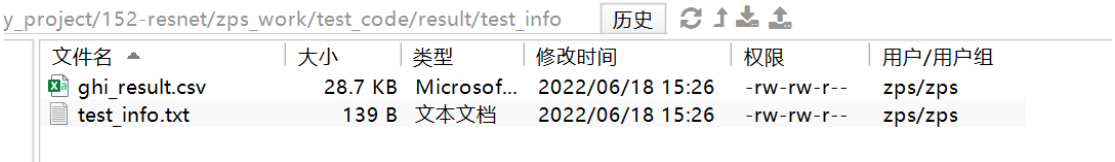


Figure 8. stored results