Tutorial

>>Wenxin Liu<<

This file aims to guiding how to use the related plant counting repository

➢ Introduction

There are totally 16 .py documents and they can be classified into 6 groups:

1. {V3lite.py, V3liteNet.py, V3litedatset.py},
2. {V3segtraval.py, V3seg_net.py, V3segdataset.py},
3. {V3segplustraval.py, V3segplus.py, V3segdataset.py},
4. {traval.py, IntegrateNet.py, Netdataset.py},
5. {error.py, mixnetL.py, CARAFE.py},
6. {Visualization.py, ScatterPlot.py}.

For the first 4 groups, take V3lite as an example. V3liteNet.py and V3litedatset.py are respectively generating the model of V3lite and related data for learning. V3lite.py is to run the model on the generated training data and testing data. Group 2,3,4 are quite similar to group 1.

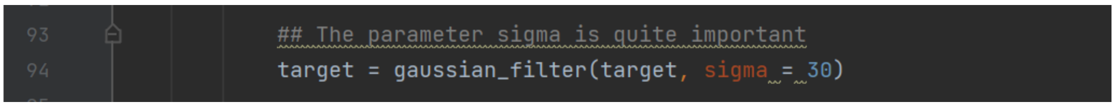
In the group 5, error.py is used to generate functions for evaluation. E.g. the function to compute MAE, RMSE, R^2. mixnetL.py and CARAFE.py store two related modules which are part of the mentioned network (V3 versions and IntegrateNet). To be more specific, mixnetL.py corresponds to Mixnet (Encoder module) and CARAFE.py corresponds to CARAFE upsampling operator.

Visualization.py and ScatterPlot.py in group 6 are used to generate results.

➢ Details

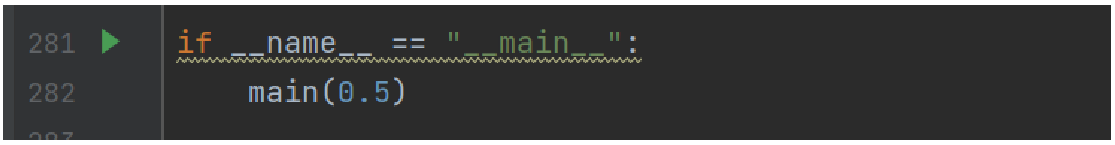
This part focuses on the operation of IntegrateNet.

1) The quite important parameter σ can be tuned in Netdatset.py file. It is in the MaizeDataset class.



```
        ## The parameter sigma is quite important
        target = gaussian_filter(target, sigma = 30)
```

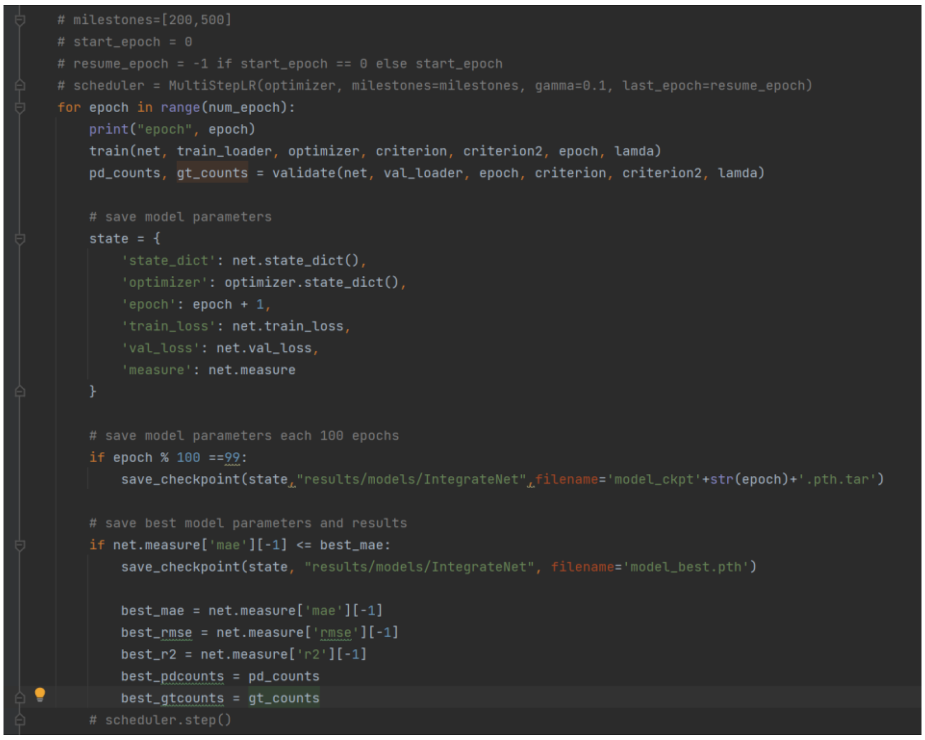
2) λ can be tuned in the last line of traval.py.



```
if __name__ == "__main__":
    main(0.5)
```

3) The current learning rate is 0.01 through the hole 1000 epochs. If you want to tune the learning rate, e.g. 0.01 for the learning rate through first 200 epochs, then 0.001 for the next 300 epochs and 0.0001 for the remaining epochs, you can use

"scheduler" in traval.py. Please uncomment the "scheduler" comments and tune the milestones and other related parameters.



```python
# milestones=[200,500]
# start_epoch = 0
# resume_epoch = -1 if start_epoch == 0 else start_epoch
# scheduler = MultiStepLR(optimizer, milestones=milestones, gamma=0.1, last_epoch=resume_epoch)
for epoch in range(num_epoch):
    print("epoch", epoch)
    train(net, train_loader, optimizer, criterion, criterion2, epoch, lamda)
    pd_counts, gt_counts = validate(net, val_loader, epoch, criterion, criterion2, lamda)

    # save model parameters
    state = {
        'state_dict': net.state_dict(),
        'optimizer': optimizer.state_dict(),
        'epoch': epoch + 1,
        'train_loss': net.train_loss,
        'val_loss': net.val_loss,
        'measure': net.measure
    }

    # save model parameters each 100 epochs
    if epoch % 100 ==99:
        save_checkpoint(state,"results/models/IntegrateNet",filename='model_ckpt'+str(epoch)+'.pth.tar')

    # save best model parameters and results
    if net.measure['mae'][-1] <= best_mae:
        save_checkpoint(state, "results/models/IntegrateNet", filename='model_best.pth')

        best_mae = net.measure['mae'][-1]
        best_rmse = net.measure['rmse'][-1]
        best_r2 = net.measure['r2'][-1]
        best_pdcounts = pd_counts
        best_gtcounts = gt_counts
    # scheduler.step()
```

4) All the results are saved in the result directory, including the dotimages, vsualization results and model parameters. Visualization.py is just calling the best model parameters and visualize the results.