GOARS is modified from nnU-Netv1. Therefore, its training method is the same as that of nnU-Netv1. When training different network structures, simply modifying the corresponding folder names will suffice to achieve the goal.

|  |  |
| --- | --- |
| nnunet\_organ\_seg.zip(Coarse segmentation) | |
| network\_architecture\_origin | Network structure folder, among which the network structure file is generic\_UNet.py, which is the original nnU-Net network structure |
| network\_training\_origin | Training program folder. The number of training rounds can be modified in the nnUNetTrainerV2.py file. |
| When using it, you need to change "network\_architecture\_origin" to "network\_architecture" and "network\_training\_origin" to "network\_training". | |

|  |  |
| --- | --- |
| nnunet\_organ\_seg.zip(location segmentation) | |
| network\_architecture\_heat | The network architecture still adopts the nnU-Net structure, and the network files are located in the generic\_UNet.py file. |
| network\_training\_heat | The parameter modifications for training are made in nnUNetTrainerV2.py. |
| data\_augmentation\_heat | The folder containing the data augmentation code for training heatmap regression. |
| dataloading\_heat | We have modified the dataloading method to present the input data in the form of a heatmap. The specific configuration file is located in dataset\_loading.py. |
| inference\_heat | We have modified the code for the reasoning process to enable it to output the heatmap results. The modification was implemented in the predict.py file. |
| When using it, you need to change "network\_architecture\_heat" to "network\_architecture" , "network\_training\_heat" to "network\_training", “data\_augmentation\_heat” to “data\_augmentation” , “dataloading\_heat” to “dataloading” and “inference\_heat” to “inference”. | |

|  |  |
| --- | --- |
| nnunet\_organ\_seg.zip(fine segmentation for small organ) | |
| network\_architecture\_25D | The folder of the small organ segmentation network structure, with the main functions located in nnunet\_2\_5D\_coarse\_delete2D.py. |
| network\_training\_25D | The training program folder for the multi-dimensional precise segmentation network, the main parameter settings are located in nnUNetTrainerV2.py. |
| data\_augmentation\_25D | Data augmentation folder for the network for precise segmentation of small organs. |
| dataloading\_25D | We made modifications in the dataset\_loading.py file to enable the network to receive input data with fixed three-layer slices. |
| When using it, you need to change "network\_architecture\_25D" to "network\_architecture" , "network\_training\_25D" to "network\_training", “data\_augmentation\_25D” to “data\_augmentation” and “dataloading\_25D” to “dataloading”. | |

|  |  |
| --- | --- |
| nnunet\_organ\_seg.zip(fine segmentation for large organ) | |
| network\_architecture | The multi-dimensional fine segmentation network file for large organs is nnunet\_coarse\_bottle\_  upsample\_delete3D\_connect5\_linux.py |
| network\_training\_32D | We made modifications to nnUNetTrainerV2.py to enable it to train 2D + 3D semantic segmentation networks. |
| When using it, you need to change "network\_training\_25D" to "network\_training". | |

For the segmentation and cropping program seg\_crop\_recover.py, the main function is crop\_data\_sum. The specific parameter settings are as follows:

|  |  |
| --- | --- |
| coarseg\_in | The storage path for the coarse segmentation results |
| image\_in | The storage path of the original image |
| label\_in | The storage path for image labels |
| coarseg\_out | The storage path for the results of the post-reduction coarse segmentation |
| image\_out | The storage path for the image results after reduction |
| label\_out | The storage path for the result of the label after reduction |
| label\_wanted | The label values of the organs that need to be trimmed |