7. Implementation Details

During training, we randomly crop images to 512×512 pixels and apply data augmentation with random scaling in $\{0.5, 0.75, 1.25\}$ and random flipping. The whole pipeline is trained for 80 epochs with a batch size 32. The number of warm-up epochs in N is chosen to be 4. Similar to [20], we utilize the SGD and the polynomial learning-rate decay $(1 - iter/max\_iter)^{0.9}$ to train the student network. The teacher network is updated with the student network parameters in the same LG after each training epoch,

$$\theta_{t^1} = \gamma \times \theta_{t^1} + (1 - \gamma) \times \theta_{s^1}, \tag{8}$$

where γ is chosen as 0.996. We initialize the backbone with ImageNet pre-trained checkpoint and the head with random numbers. The loss weight for unlabelled data (α) is 1.5. During inference, the final segmentation mask y is produced by the ensemble of the two teachers,

$$y = \sigma(0.5 \times (g^1(x) + g^2(x))), \tag{9}$$

where $\sigma(.)$ denotes the Softmax function, $g^1(.)$ denotes the teacher output from the first LG, and $g^2(.)$ denotes the teacher output from the second LG.

8. Noise Generation

We generate three types of noise for semi-supervised semantic segmentation, including the pseudo label (PL), random dilation and erosion (RDE) and similar class perturbation (SCP). The first two types of noise are easier to understand. Here, we explain the generation process of SCP and show the visualization of all types of noise in Fig.9.

The key point of SCP is to get the classes resemble the texture with each other. We utilize the model prediction of [34] to estimate the visual similarity among different classes. Specifically, we first initialize a memory $\mathcal{M} := [0]^{\ell \times \ell}$ (ℓ is the total number of classes) to save the similarity values. Then, we get the model prediction $h(x_i, j) \in [0, 1]^{\ell}$ for each pixel in the labelled dataset $\mathcal{D}_L = \{(x_i, j, y_{i,j})\}$, $y_{i,j} \in \mathcal{C}$ based on [34]. After that, the memory can be updated by,

$$\mathcal{M}_c = \frac{1}{n_c} \sum_{i,j} \{h(x_i, j) \mid y_{i,j} = c\}, \tag{10}$$

where $\mathcal{M}_c \in [0, 1]^{\ell}$ is the $c\text{-}th$ row of the memory $\mathcal{M}$, n_c is the pixel number of class c in the dataset and $h(.)$ denotes the model mapping to softmax scores. After updating the memory $\mathcal{M}$ with all the data, we set the diagonal element of $\mathcal{M}$ as zeros. If we need to add pertubation to the label of class c, we select another class c' according to $\mathcal{M}_c$. The $k\text{-}th$ element of $\mathcal{M}_c$ are used as sampling probability to select class k as a replacement of class c.

Table 5: The mIoU (%) of our pipeline on Pascal VOC 2012.

	Labelled Data Ratio			
	1/16	1/8	1/4	1/2
PS-MT [20]	72.83	75.70	76.43	77.88
Ours − FM	78.41	78.61	79.82	79.90
Ours	77.75	79.31	79.14	79.54

(a) Input image (b) Clean Annotation (c) Noisy Annotation

Figure 9: The visualization of three types of noise, including PL (the $1st$ row), RDE (the $2nd$ row) and SCP (the $3rd$ row).

9. Comparison on SegTHOR

We compare our method with Base-ADELE and ADELE on the SegTHOR dataset. The Base-ADELE consists of a UNet which is trained with multi-scale inputs. ADELE introduces the label correction to Base-ADELE, which can automatically correct the noisy annotation based on model prediction in an early training stage. In our framework, we utilize the filter model to detect the noisy annotations and use them in unsupervised setting.

10. Performance of FM on Clean Data

In Table 1, we remove the Filter Module ('Ours − FM') when training models with clean data. We also show the performance of the whole pipeline ('Ours') including the filter module on clean data in Table 5. The filter module has only a small adverse effect when applied on the clean data. Hence, 'Ours' produces similar performance with 'Ours − FM' and much better than the baseline model ('PS-MT').

11. Qualitative Results

We show the qualitative results of our pipeline on Pascal VOC 2012 in Fig.10 and SegTHOR in Fig.11. We can see that the proposed approach can get good segmentation results on the two datasets.

12. Class Perturbation

In Section 3.2, we introduced Similar Class Perturbation (SCP) as a technique for adding noise to ground-truth labels. SCP involves adding class noise to the ground-truth label by considering the visual similarity between different classes. To generate this noise, we first create a class-similarity array $\mathcal{A} \in \mathbb{R}^{21\times21}$, where each element in the i-th row and j-th column of $\mathcal{A}$ represents the visual similarity between the i-th and j-th classes. We initialize $\mathcal{A}$ with zeros. We then use SEAM [34] to predict the classification probabilities for each image in the val dataset of Pascal VOC 2012. For each pixel in an image, we obtain a classification probability vector $a \in \mathbb{R}^{1\times21}$ over the 21 classes. Given the ground-truth label, we compute the mean classification probability vector for pixels belonging to each class c and add this vector to the c-th row of $\mathcal{A}$. We repeat this process for all images in the dataset. To select similar classes, we set the diagonal elements of $\mathcal{A}$ to zero and normalize the elements in each row. $\mathcal{A}$ is used to determine the similarity between classes. In this way, we obtain a noisy ground-truth label that reflects the visual similarity between classes in the dataset.

Figure 10: The visualization of the predicted segmentation masks on the validation set of Pascal VOC 2012. The model is trained with $1/8$ labelled data and 9% PL noise. The first line shows the input images. The second line presents the ground truth. The last line denotes our model prediction. Our model can get good performance for different classes.

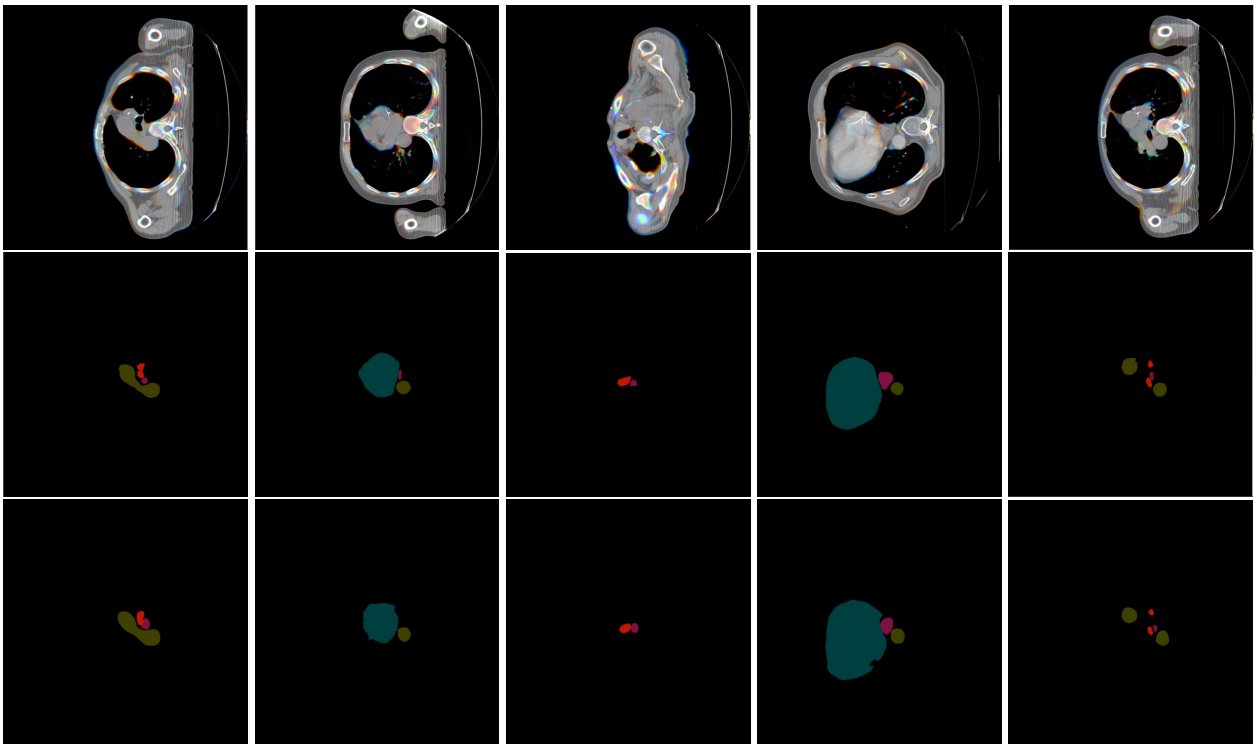

Figure 11: The qualitative results of our method on SegTHOR. The first line shows the input images. The second line presents the ground truth. The last line denotes our model prediction. Our model achieves good performance.