Appendix: Keypoint-Augmented Self-Supervised Learning for Medical Image Segmentation with Limited Annotation

Anonymous Author(s)
Affiliation
Address
email

A Additional Experiments

Ablation on UNet Channels (Double Channels). In our implementation, we augment the convolutional UNet features by concatenating them with features learned from KAF. Introducing KAF layers leads to an increase in the total number of parameters of the final KAF-enhanced UNet. Specifically, the number of channels for the second, third, and fourth blocks in the KAF-enhanced UNet become twice as large as the original UNet.

For a more fair comparison, we construct a non-KAF UNet (`'UNet(c2)'` in Tab. 1) with the same amount of parameters in convolutional layers as our model by duplicating the features of the first, second, third, and fourth blocks from the UNet baseline (`'UNet(c1)'` in Tab. 1) and concatenated each with itself.

The segmentation results are presented in Table 1. `UNet(c1)` indicates the UNet backbone, and `UNet(c2)` denotes the larger UNet whose convolutional parameters matched our model. They indicate that widening the network's architecture by increasing the input channel size can improve its performance. However, the performance enhancement is even more substantial with our modified layer. This suggests that incorporating features beyond simple convolution into the network architecture can further enhance the network's performance.

Sample M	dataset	Method	mean/std	#params
15	CHD	UNet(c1)	0.627(.05)	7.8 M
15	CHD	UNet(c2)	0.646(.04)	27.9 M
15	CHD	Ours	0.712(.03)	71.7 M
6	ACDC	UNet(c1)	0.782(.03)	17.5 M
6	ACDC	UNet(c2)	0.796(.03)	62.8 M
6	ACDC	Ours	0.873(.01)	106.8 M

Table 1: Performance comparison among standard UNet (`UNet(c1)`), a larger UNet with duplicated input channels (`UNet(c2)`), and UNet augmented with features derived from KAF (Ours). The results indicate that using a larger UNet slightly improves the segmentation performance. Our proposed KAF-enhanced UNet further boosts the performance significantly compared with `UNet(c2)`.

Ablation on Correspondence Weights. To further investigate the contribution of the correspondence loss to the performance of the pretraining weights, we conducted a study on the various combinations of weights w_1, w_2, and w_3, as defined in Sec. 3.2, in Table A.

First, we study the effect of varying w_3 by setting both w_1 and w_2 to 1. We experimented with values of 0.1, 0.01, and 0.001 for w_3. Among these, 0.01 yielded the best performance.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

In a subsequent series of experiments, we set w_1 to 0, intending to exclusively investigate the impact of the local correspondence loss on the global KAF self-supervised learning (SSL) loss. When only w_2 is set to 1, the model achieves an dice of 0.701. Upon varying w_3 while keeping w_2 fixed at 1, in most instances, incorporating $\mathcal{L}_{local}$ yields performance better than with $\mathcal{L}_{global}$ only. This implies that our local KAF correspondence SSL loss indeed offers a superior local minimum for downstream tasks.

When the weights are set to 1, 1, and 0, our model's performance, at 0.689, still surpasses that of a model trained from scratch (0.686). This suggests that using only $\mathcal{L}_{local}$ could assist in finding a more optimal starting point for fine-tuning. However, when compared to having only local losses, global pretraining losses contribute more to enhancing performance.

w_1	w_2	w_3	Dice
1	1	0.1	0.702(.04)
1	1	0.01	0.712(.03)
1	1	0.001	0.708(.04)
0	1	0.08	0.705(.03)
0	1	0.04	0.705(.03)
0	1	0.02	0.707(.03)
0	1	0.01	0.700(.03)
0	1	0.005	0.700(.03)
0	1	0.001	0.705(.03)
0	0	1	0.689(.04)

Table 2: Additional assessment of weight of $\mathcal{L}_{local}$ in pretraining to supplement Tab. 2 in the main text. The results are all from pretraining on CHD and finetuning at $M = 15$.

Comprehensive 5-Fold Results. Our detailed examination of the five folds in the CHD and ACDC datasets is exhibited in Table 3. We adhere strictly to the divisions as described in the Positional Coding Learning (PCL) study [7], ensuring consistency and reliable comparison of results.

KAF Layer in FCN. To validate the general usefulness of the KAF layer, we injected KAF layer into a different segmentation backbone Fully Convolutional Network (FCN) [3] and compare the results between the original FCN versus the KAF-enhanced FCN. Note that in the original FCN implementation, VGG [5] was used as the encoder to gather multi-resolution features from different layers. However, given the small training size of our dataset and the relatively high complexity of VGGwe substitute the VGG encoder with a shallower CNN. More specifically, we replaced each block in VGG with two convolution layers, aiming to create a more efficient model that better suits our dataset and task.

Dataset	Sample M	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean/Std
CHD	2	0.3085	0.3292	0.4649	0.4405	0.4178	0.392(.062)
	6	0.5370	0.6527	0.6707	0.7076	0.6119	0.636(.058)
	10	0.6382	0.6797	0.7252	0.7326	0.6900	0.693(.034)
	15	0.6668	0.6892	0.7519	0.7458	0.6844	0.712(.035)
	20	0.6738	0.7204	0.7629	0.7766	0.7051	0.728(.038)
	30	0.7291	0.7324	0.8001	0.8014	0.7093	0.754(.039)
	51	0.7385	0.7594	0.8148	0.8234	0.8048	0.788(.033)
ACDC	2	0.7975	0.7027	0.7510	0.7097	0.7458	0.741(.034)
	6	0.8827	0.8941	0.8620	0.8596	0.8682	0.873(.013)
	10	0.9101	0.9086	0.8919	0.8709	0.8914	0.895(.014)
	15	0.9175	0.9076	0.9140	0.8932	0.9091	0.908(.008)
	20	0.9173	0.9152	0.9168	0.9101	0.9143	0.915(.003)
	30	0.9224	0.9232	0.9252	0.9162	0.9187	0.921(.003)
	80	0.9313	0.9285	0.9336	0.9255	0.9328	0.930(.003)

Table 3: The complete five-fold Dice results for CHD and ACDC are presented in our Table 1.

Sample M	With KAF	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean/Std
2	-	0.2259	0.2133	0.3297	0.311	0.3516	0.286(.056)
	✓	0.2517	0.2133	0.3392	0.3034	0.3794	0.297(.059)
6	-	0.4441	0.5462	0.5427	0.5918	0.4854	0.522(.052)
	✓	0.4495	0.5603	0.5652	0.6286	0.5535	0.551(.058)
10	-	0.4866	0.5584	0.6393	0.6613	0.6094	0.591(.063)
	✓	0.5632	0.6209	0.6381	0.6718	0.6383	0.626(.036)
15	-	0.5938	0.6356	0.6702	0.6926	0.6499	0.648(.033)
	✓	0.6157	0.6163	0.7117	0.6936	0.6537	0.658(.039)
20	-	0.6318	0.6422	0.7102	0.7383	0.6356	0.672(.044)
	✓	0.6382	0.6541	0.7112	0.7309	0.6806	0.683(.034)
30	-	0.6339	0.6558	0.7498	0.7701	0.6685	0.696(.054)
	✓	0.6841	0.6671	0.7507	0.7708	0.6973	0.714(.040)
51	-	0.7125	0.7287	0.7693	0.7755	0.7559	0.748(.024)
	✓	0.7087	0.7324	0.7711	0.7813	0.7661	0.752(.027)

Table 4: Segmentation results on CHD dataset from a random initialized FCN backbone. The column labeled 'With KAF' indicates whether the proposed KAF layer is inserted to the backbone. The results demonstrate that the integration of the KAF layer tends to improve the mean values across different sample sizes, indicating an enhanced performance of the FCN when augmented with the KAF layer.

The results of our experiment are reported in Table A. We conduct trials with different training sample sizes. The results verify that appending KAF layers to FCN boosts performance across all sample sizes, with an average improvement exceeding 1%. Interestingly, we observed that FCN outperforms UNet in tasks involving training from scratch. However, when we incorporated the KAF layer into FCN, it did not surpass the performance of our layer applied to UNet. This could potentially be attributed to our approach in this experiment; we did not undertake hyperparameter optimization but instead directly added the layer we used in our main text to FCN. Therefore, compared to UNet, the performance improvement is relatively smaller.

Computation analysis. As transformers [6] also incorporate long-range dependencies by learning self-attention among uniformly distributed patches within the image, we compare the computational differences of a SwinTransformer [2] and our model. Fig. 1 displays a comparison of GFLOPs and GPU memory usage between our method and the SwinTransformer, given two specific variations: the edge length of the input image and the number of self-attentions within each transformer.

In the left-side plot, the GFLOPs of our method vary by a constant value (around 130 GFLOPs) as the depth of the attention map escalates. Conversely, this metric grows exponentially in the SwinTransformer. The disparity arises because our method keeps the number of keypoints constant, meaning that even as the edge length of the input image enlarges, the computation surge in the attention map remains consistent. In contrast, transformer-based methods like the SwinTransformer require a quadratic increase in computation to generate the attention map.

Our method demonstrates a slower growth rate in memory usage, particularly as the number of self-attentions increases. In the SwinTransformer, the size of the attention map correlates quadratically with the edge length of the image. However, our approach maintains a fixed number of input keypoints, which stabilizes the attention map as a constant factor in memory usage.

B Additional Implementation Details

Keypoint Preprocessing Details. We compute the keypoint positions on the original image using SIFT. To obtain the keypoint positions on augmented images, we propose three potential solutions: (1) Extract the translation matrix from the augmented image and apply this translation to the keypoint positions. (2) Recompute SIFT on the augmented image to determine the new keypoint positions. (3) Consider each keypoint position as a label, and augment these labels alongside the original image.

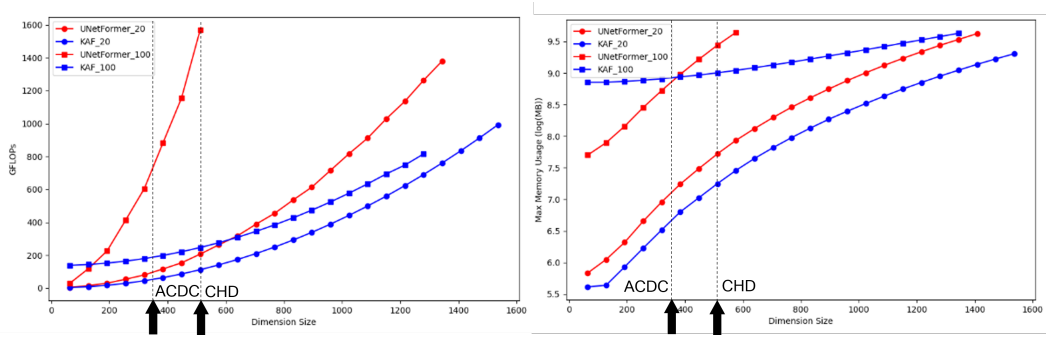

Figure 1: Comparison of GFLOPs and GPU memory usage between our method and the SwinTransformer [2]. The x-axis indicates the size of the image. We test models with different numbers of self-attention blocks (20 or 100) within the transformer, represented by different colors. The results illustrate that the GFLOPs of our method vary minimally with increasing attention map depth, and the memory usage of our method increases at a slower rate with a growing number of self-attentions. This highlights the efficiency of our method, especially with larger input image edge lengths and more complex attention maps. We also denote the actual input size of CHD dataset (512×512) and ACDC dataset (352×352), where our model is more computationally efficient than using a transformer in both the GFLOPs and memory consumption.

The first solution involves retrieving the translation matrix from the augmentation package, which is not easily accessible in popular data augmentation packages currently available. Moreover, this approach requires careful handling of the cropping of the translated keypoints. The second solution poses the challenge of translating the correspondence from the original image pair to the augmented image pair. The third solution, on the other hand, retains the index of the keypoints, thus preserving the correspondence from the original to the augmented image. Consequently, we choose the third solution as our preferred approach.

To elaborate further, we initially assign a unique index to each detected keypoint. Subsequently, this index is attributed to the image space, resulting in a 2D matrix. In our implementation, the background is designated as 0, while the keypoint index starts from 1. This matrix is then transformed concurrently with the original image, resulting in a translated keypoint index matrix. We further process the keypoints under the following two circumstances: (1) If a keypoint is present in the original image gets cropped out, we simply disregard that keypoint. (2) The keypoints might project onto one or multiple nearby positions. In this scenario, we compute the mean of all these positions, and this average becomes the new keypoint position in the augmented image.

C Additional Results

Keypoints Detection Results. In Fig. 2 and 3, we present the input images and detected keypoints using the Scale-Invariant Feature Transform (SIFT) [4]. The results suggest that the majority of the detected keypoints are located in the foreground rather than the background. This helps the KAF layer to concentrate more on the regions that are crucial for segmentation tasks. Furthermore, we display the transformed keypoint positions after performing data augmentation on the input image in the third column. As described in the preprocessing details in Sec. B, during the augmentation, we transform the keypoint positions from the original image to the augmented one with the transformation matrix.

Keypoint Correspondence on Images. In columns (b) and (d) of Fig. 2 and Fig. 3, we provide a visual representation of the detected image correspondence. The results indicate that our heuristic distance metrics effectively identify correspondences between neighboring slices in biomedical images. During data augmentation, rather than recomputing the correspondences between the augmented slices, we translate the found correspondences from the original slices to the augmented ones, similar as the keypoint processing above.

Additional Self-attention Map. In Fig. 4, we provide a visualization of the self-similarity maps from two neighboring slices derived from various layers of our UNet. Similar to Fig. 3 in the main text, the

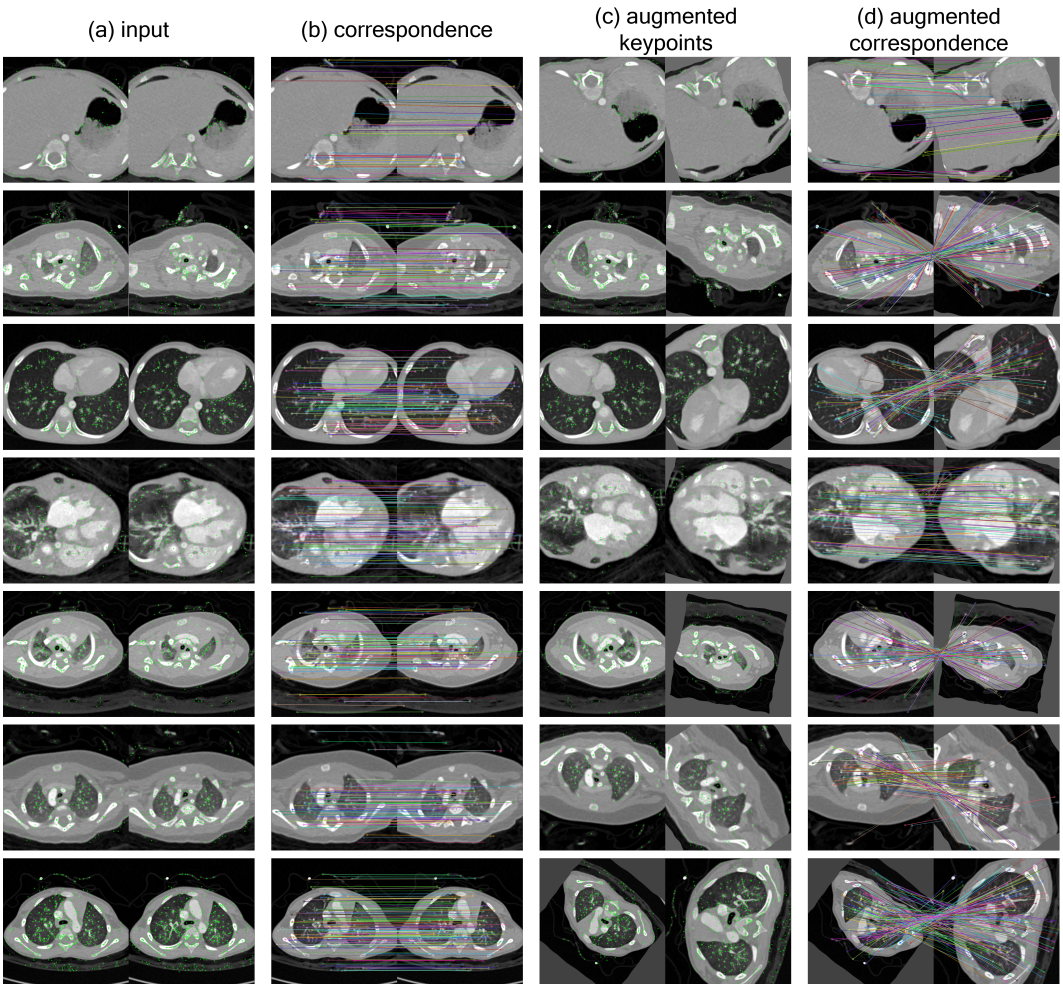

Figure 2: Visualizatin of the detected keypoints and their correspondence on images sampled from CHD dataset. The green dots represent the keypoints detected by SIFT. In column (a), we illustrate two adjacent slices. In column (b), we showcase the correspondence between these two slices, applying the heuristic described in section 3.2 from the main text. For the actual training, we apply data augmentation into the input image, and examples are shown in (c) with the augmented keypoints. We also display the transferred correspondence in column (d). Please zoom in to view the details.

similarity map of our method demonstrates better resilience to augmentations and maintains localized consistency among keypoints, compared to other pretrained models such as PCL and GLCL [1].

Additional Segmentation Results. To supplement Sec. 4, we present additional segmentation results comparing our method with other baselines in Fig. 5. Models trained and/or finetuned with different numbers of subjects from both ACDC and CHD datasets are present.

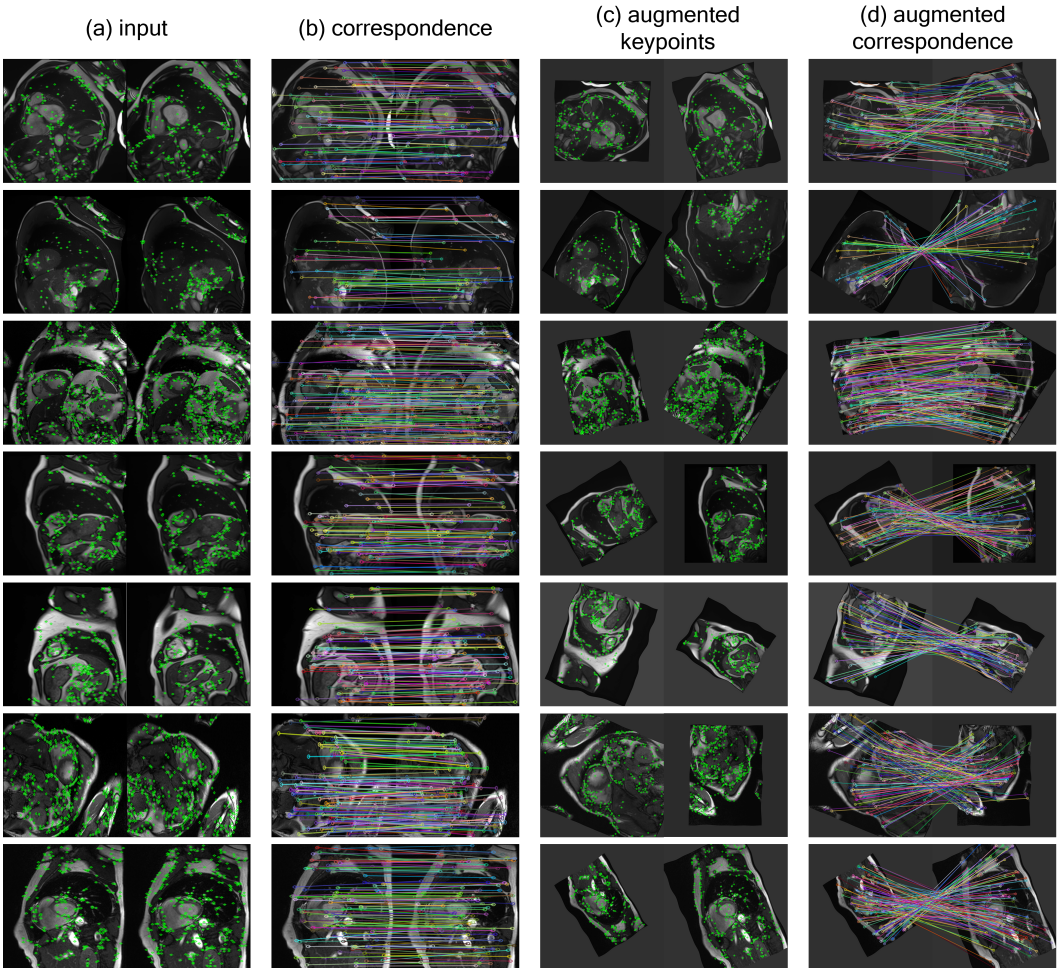

Figure 3: Visualization of slices, keypoints, and correspondences on the ACDC dataset.

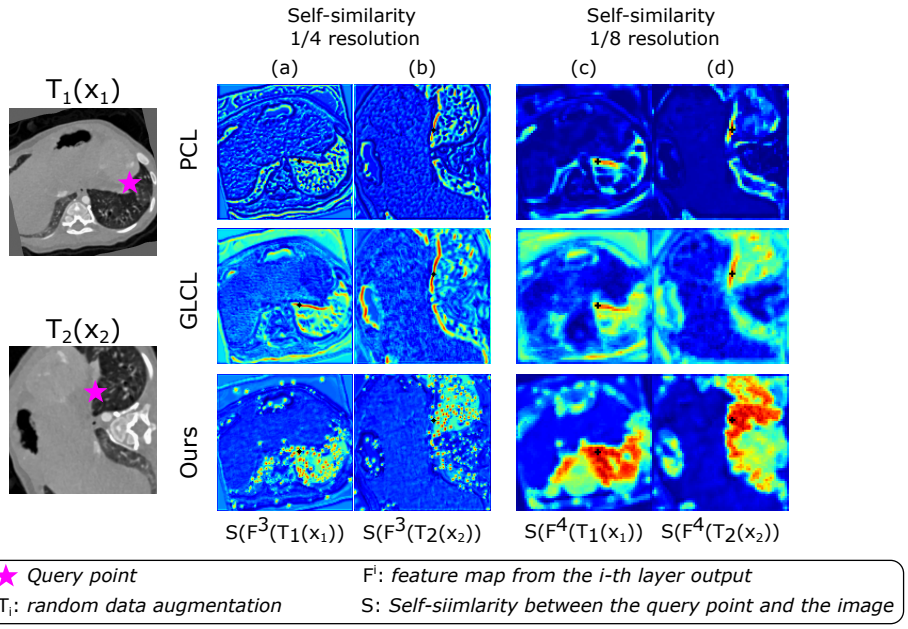

Figure 4: Learned self-similarity from two adjacent slices. Each map indicates the feature similarity between the query point feature (star) and other points within the image. We display similarity at two different scales within the UNet encoder. The comparison between (a) and (b), (c) and (d) indicates that ours is more resilient in maintaining the self-similarity of the features under various transformations.

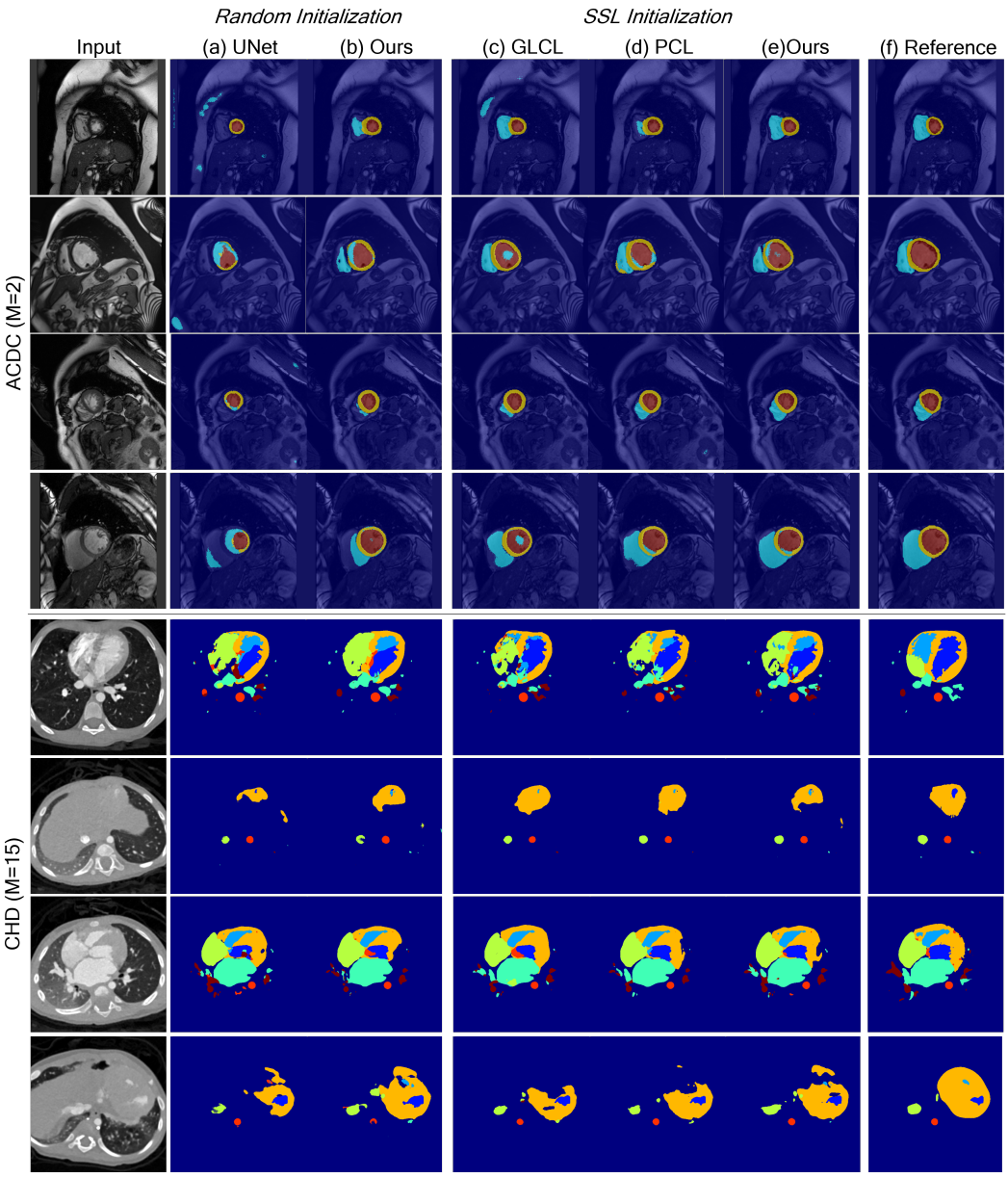

Figure 5: Additional segmentation results to supplement Fig. 2 in the main text.

References

[1] Krishna Chaitanya, Ertunc Erdil, Neerav Karani, and Ender Konukoglu. Contrastive learning of global and local features for medical image segmentation with limited annotations. *Advances in Neural Information Processing Systems*, 33:12546–12558, 2020.

[2] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[4] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[7] Dewen Zeng, Yawen Wu, Xinrong Hu, Xiaowei Xu, Haiyun Yuan, Meiping Huang, Jian Zhuang, Jingtong Hu, and Yiyu Shi. Positional contrastive learning for volumetric medical image segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part II 24*, pages 221–230. Springer, 2021.