Supplementary material

A. Deformable transformer encoder

The deformable encoder [44] enriches the input mainly by the Deformable Attention (DA) module and the Feed-Forward Network (FFN). The detailed architecture can be seen in Figure 7. The DA module sums the selected features at deformable sampling locations across multi-scales with learned attention weights. The ourput of this module is passed through the FFN.

Suppose the encoder takes as inputs the flattened feature map $\boldsymbol{m_f} \in \mathbb{R}^{C \times N_{in}}$($C$: number of channels, $N_{in} = \sum_l H_l W_l$), positional and level embedding information $\boldsymbol{E} \in \mathbb{R}^{C \times N_{in}}$ and reference points $\boldsymbol{P} \in \mathbb{R}^{N_{in} \times 2}$. The output of the deformable attention layer is formulated as:

$$\boldsymbol{O_{DA}} = f(\boldsymbol{O_s}) \tag{7}$$

where f is the linear layer and $\boldsymbol{O_s}$ is the weighted summation:

$$\boldsymbol{O}_s = \sum_{l,p} \boldsymbol{W}_{nhlp} \boldsymbol{V}_{nh}(\boldsymbol{P}_n + \Delta \boldsymbol{P}_{nhlp}) \rightarrow [C, N_{in}] \tag{8}$$

where n, h, l and p index the pixel in the flattened feature map, attention head, feature level and sampling point respectively. The rightarrow $\rightarrow$ represents reshaping to the dimensions in the brackets. The value feature $\boldsymbol{V}$, the predicted sampling offsets $\Delta \boldsymbol{P}$ and attention weights $\boldsymbol{W}$ are defined as:

$$\boldsymbol{V} = f(\boldsymbol{m} \rightarrow [N_{in}, N_h, C/N_h]) \tag{9}$$

$$\Delta \boldsymbol{P} = f(\boldsymbol{Q}) \rightarrow [N_{in}, N_h, N_l, N_p, 2] \tag{10}$$

$$\begin{aligned} \boldsymbol{W} = Softmax(f(\boldsymbol{Q}) &\rightarrow [N_{in}, N_h, N_l N_p]) \\ &\rightarrow [N_{in}, N_h, N_l, N_p] \end{aligned} \tag{11}$$

where the query feature $\boldsymbol{Q}$ is the element-wise addition:

$$\boldsymbol{Q} = \boldsymbol{m} + \boldsymbol{E} \tag{12}$$

It should be noted that $\boldsymbol{W}$ is normalized in the last dimension to provide weights that sum up to 1. The encoder finally outputs $\boldsymbol{O} \in \mathbb{R}^{C \times N_{in}}$ as:

$$\boldsymbol{O} = FFN(LN(Dropout(\boldsymbol{O}_{DA}) + \boldsymbol{m})) \tag{13}$$

where FFN and LN are short for Feed-Forward Network and Layer Normalization layer respectively.

B. Dataset explanations

To acquire the weak annotations of polyps, we manually annotate the simple sketches with the help of the PaintTool SAI, which is a painting tool for drawing. Annotators are asked to relabel the dataset according to their first impressions without a fixed drawing style. These simple sketches only cost 2 seconds to label an image.

More visualizations of our annotated dataset can be seen in Figure 9. Column 1 shows the original image, column 2 shows the original ground truth segmentation map, column 3 shows only the foreground annotation, and column 4 shows both the background and foreground annotations. Only the annotations shown in the last 2 columns were used in training our model.

C. Visualizations

Figure 10 shows qualitative results between our method and other state-of-the-art methods. It should be noted that all the other methods shown were trained in a fully supervised way. Impressively, in some cases such as in rows 1, 2 and 3, the fully supervised methods completely fail while our method manages to recover the main polyp part. In order to provide fair visualizations and avoid cherry-picking we also provided more cases where other methods such as PraNet [11] perform better such as rows 4, 5 and 7. However, our method still outperforms all other fully supervised methods beyond Pranet in these qualitative visualizations. In other words, visual maps in Figure 10 demonstrate that the proposed method has a better generalization ability that can achieve satisfactory detection results in different scenarios.

Lastly Figures 11, 12, 13, and 14 show more qualitative examples ablating our method in a similar way to Figure 2. Column 1 shows the original RGB image, column 2 shows the prediction when trained only with L_p, column 3 shows the predictions when trained with sparse foreground loss, column 4 shows the predictions when trained with L_{semi}, column 5 shows the predictions when trained using DTEN and column 6 shows the original ground truth segmentation maps. It is evident that each proposed idea of our method provides performance improvement evident from these visualizations.

D. Hyperparameter optimization

In order to find proper α (equation 3), β_1 and β_2 (equation 6) for our training regime, we carried out hyperparameter optimization. We investigated the performance of our regime on five polyp datasets with different hyperparameter settings as presented in Tables 6 and 7. Results show that the most accurate segmentation is achieved on all datasets with $\alpha = 0.5$. For (β_1, β_2), the combination of $(0.1, 0.5)$ produces the best results on three out of the five datasets. To generalize the regime, we use the aforementioned settings for most robust and accurate segmentation.

Table 6. Comparisons with different α in weakly-supervised training.

α	ColorDB		ETIS		Kvasir		CVC-300		ClinicDB	
	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU
0	0.327	0.263	0.218	0.168	0.555	0.488	0.240	0.174	0.479	0.448
0.5	0.539	0.503	0.442	0.415	0.700	0.668	0.662	0.658	0.740	0.708
1	0.124	0.089	0.064	0.026	0.209	0.133	0.060	0.029	0.126	0.082

Table 7. Comparisons with different combinations of β_1 and β_2 in semi-supervised training. 'Baseline' represents the performance of the model trained only with L_{weak} using equation 3.

(β_1, β_2)	ColorDB		ETIS		Kvasir		CVC-300		ClinicDB	
	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU
Baseline	0.539	0.503	0.442	0.415	0.700	0.668	0.662	0.658	0.740	0.708
$(0.5, 0.5)$	0.559	0.513	0.483	0.439	0.716	0.668	0.702	0.667	0.748	0.701
$(0.3, 0.5)$	0.579	0.527	0.497	0.444	0.718	0.668	0.722	0.692	0.760	0.716
$(0.1, 0.5)$	0.604	0.544	0.501	0.442	0.730	0.677	0.729	0.678	0.771	0.718
$(0.0, 0.5)$	0.582	0.511	0.424	0.359	0.759	0.690	0.648	0.585	0.756	0.690

Table 8. Quantitative results with mDice and mIoU on DiNO.

Method	ColorDB		ClinicDB	
	mDice	mIoU	mDice	mIoU
DiNO+L_{weak}	0.577	0.489	0.756	0.670
DiNO+$L_{weak} + L_c$	0.623	0.527	0.821	0.747

E. DINO backbone

Furthermore, the generality of our method can be seen (Table 8) beyond convolutional-based backbones. Using our framework we fine-tune a transformer-based backbone, DiNO [7], and a convolutional-based segmentation head surpassing the performance of other fully supervised polyp segmentation methods.

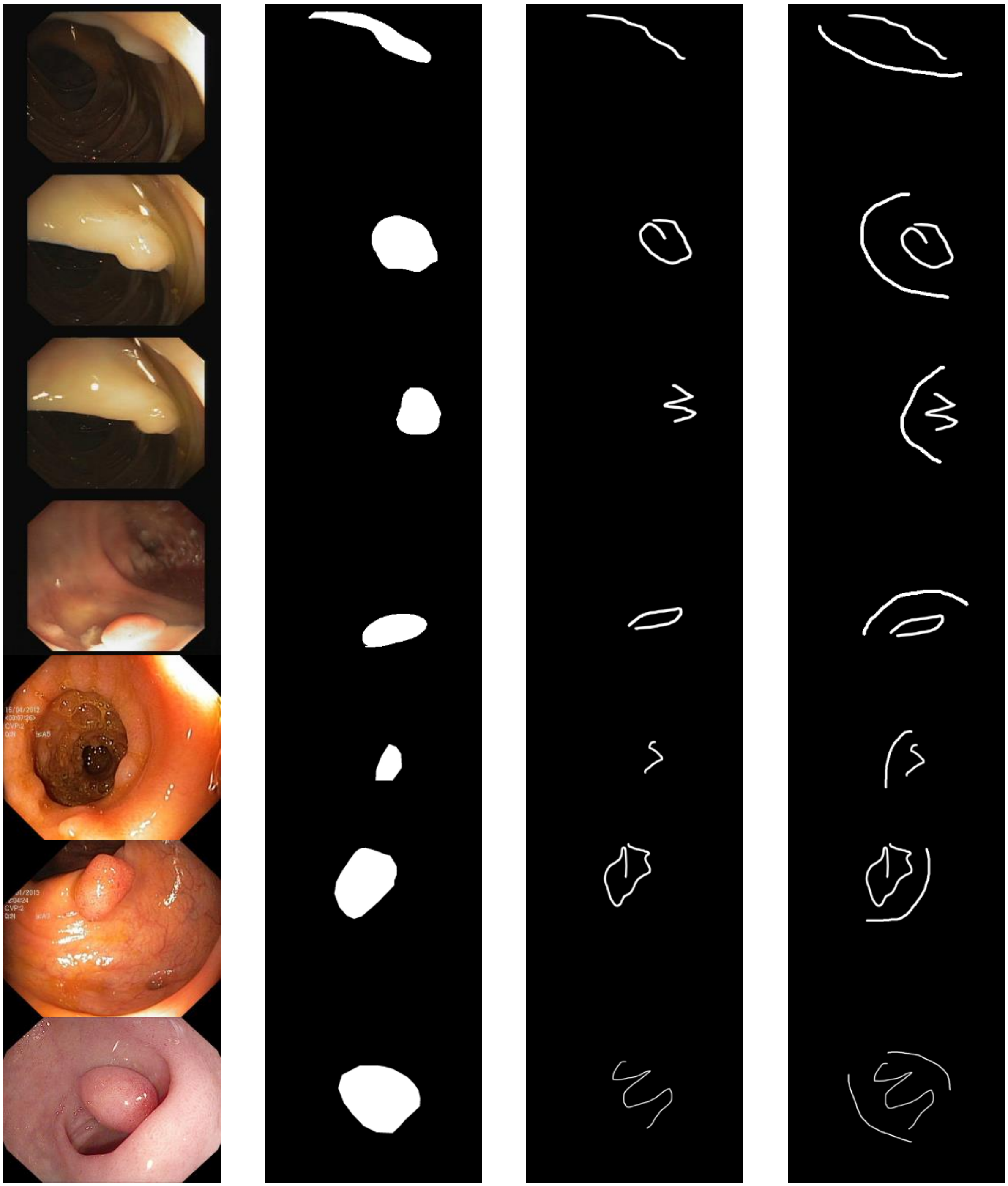

Figure 9. Training samples.

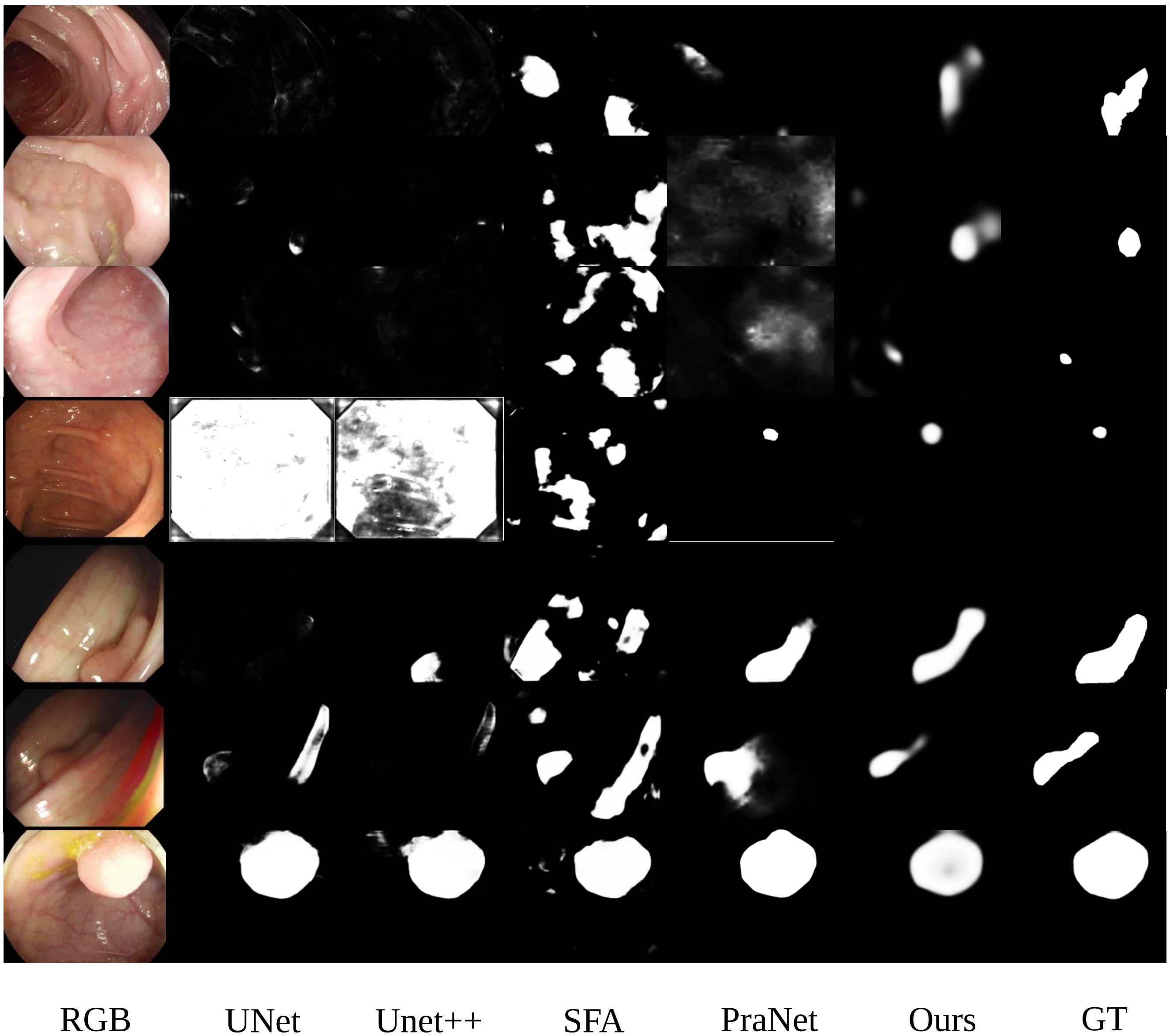

Figure 10. Comparisons with other state of the art methods.

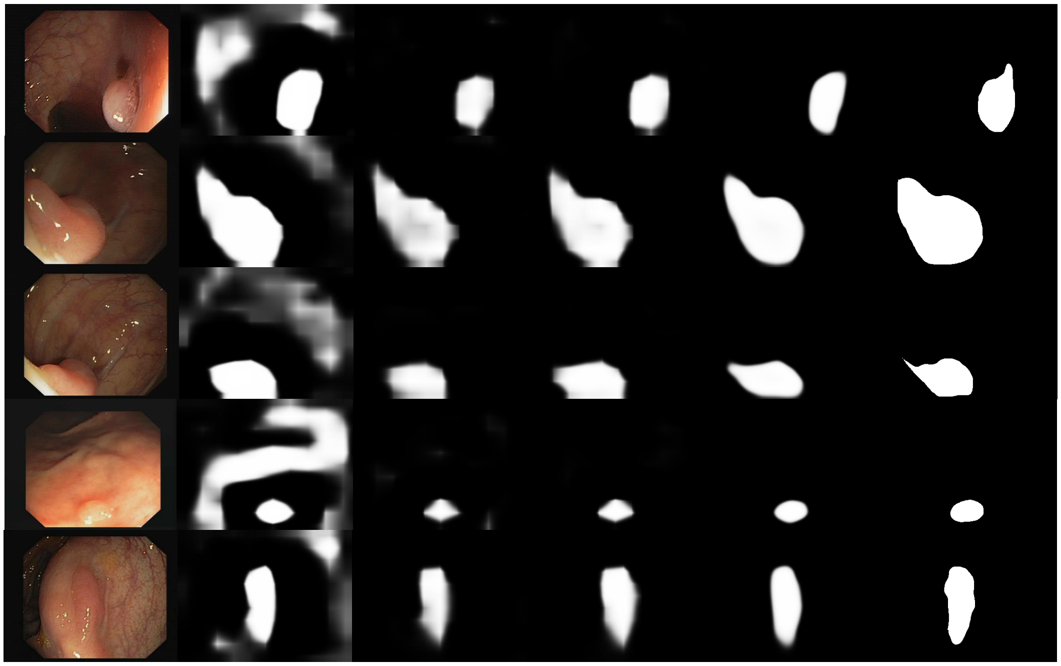

Figure 11. ClinicDB

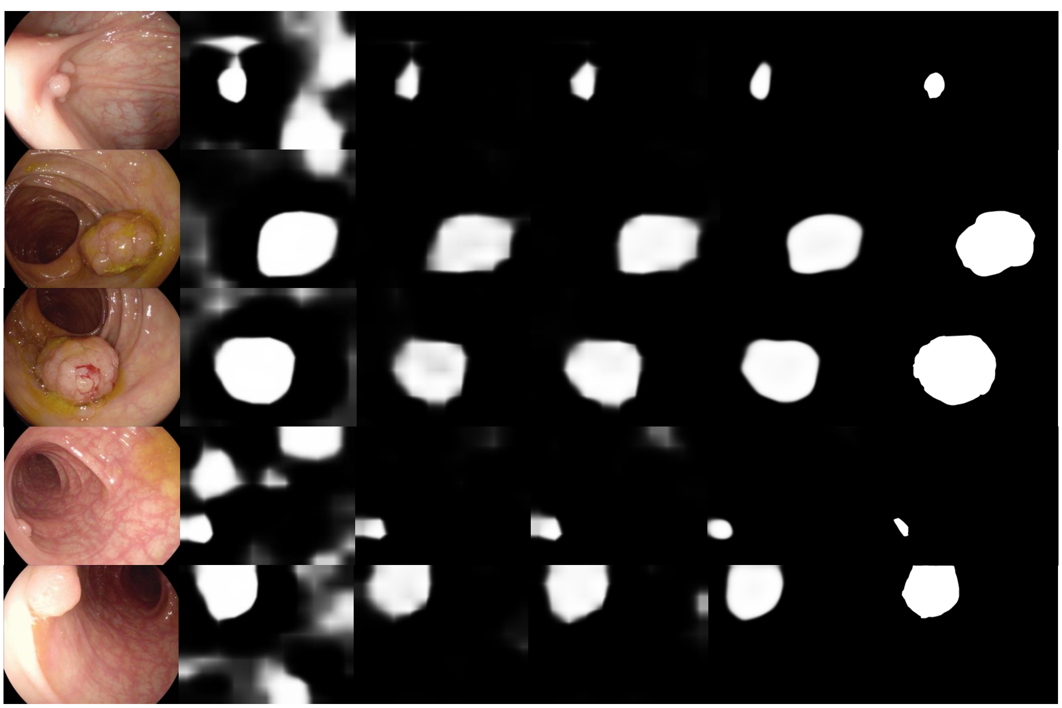

Figure 12. ETIS

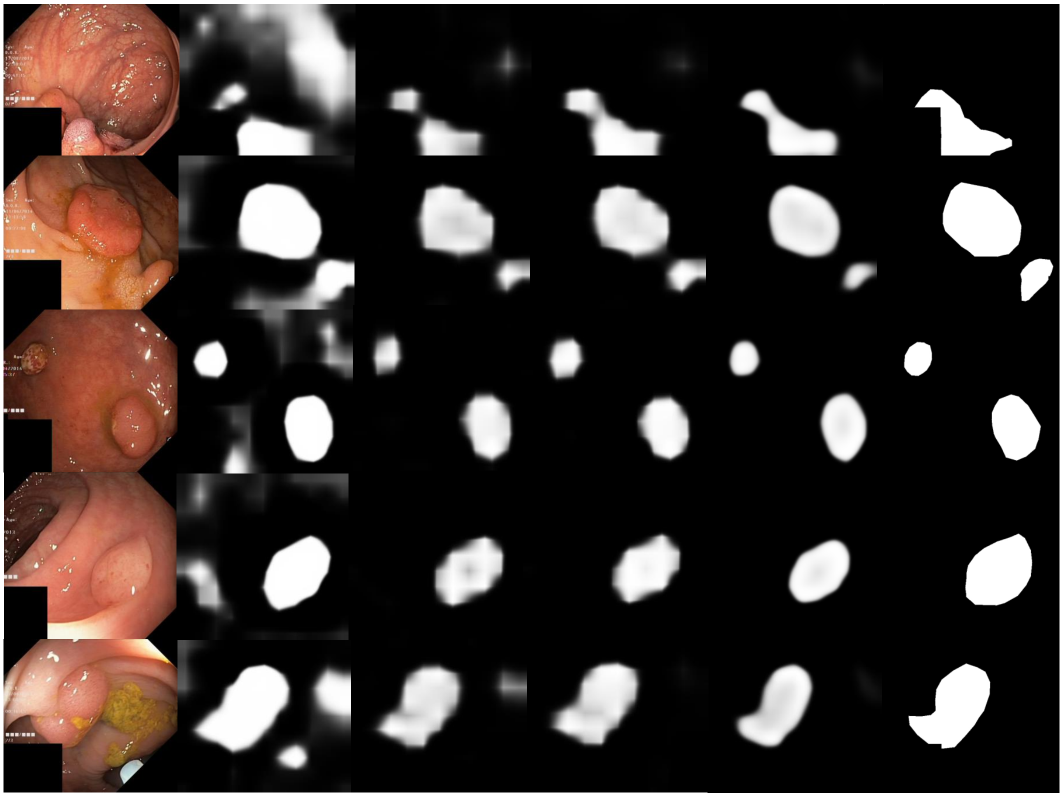

Figure 13. Kavsir

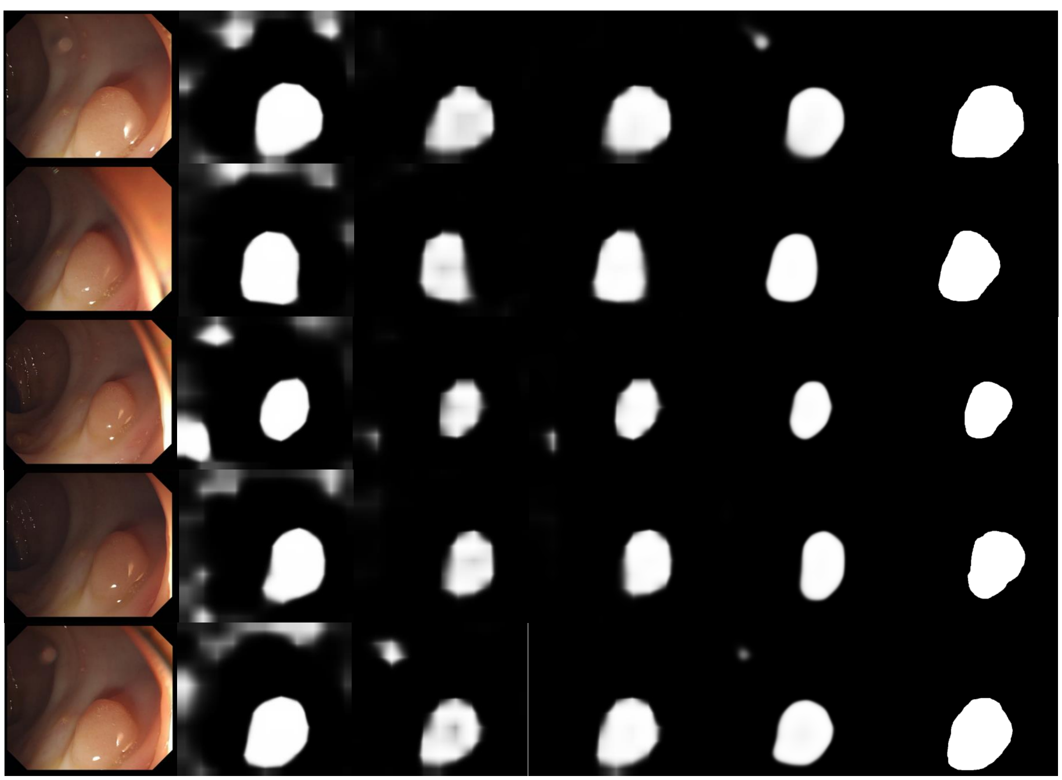

Figure 14. ColorDB

Table 9. Notations lookup table.

Notation	Description
X	Whole dataset
X_l	Weakly-annotated subset
Y_l	Weakly-annotated subset's ground truth
X_u	Unlabeled subset
θ	Learnable parameters
M_θ	Model
M_θ^t	Teacher model
M_θ^s	Student model
x_i	The ith image from X
$\hat{y}_i$	Predicted segmentation map of x_i
y_i	Ground truth map of x_i
y_i^f	The ground truth map of x_i with only the foreground annotation
L_p	Partial cross-entropy loss
L_f	Sparse foreground loss
L_{weak}	Total loss for weakly-supervised learning
α	Weight of the foreground loss
$\hat{y}_i^t$, $\hat{y}_i^s$	The prediction of the teacher and student models with input x_i
B	Batch
B_l	The batch of labeled samples
B_l^f	The batch of foreground annotations
L_{semi}	Total loss for semi-supervised learning in each B
β_1, β_2	Weights of L_c in L_{semi}
$\boldsymbol{m}$	Feature map output by the last stage of the backbone
l	Index of the feature level
n	Index of the pixel in $\boldsymbol{m_f}$
h	Index of the attention head
p	Index of the sampling point
$\boldsymbol{m}_l$	Feature map at l-th level
H_l, W_l	Height and width of $\boldsymbol{m}_l$
W_l	Width of $\boldsymbol{m}_l$
$\boldsymbol{m_f}$	Feature map after concatenation and flatten
$\boldsymbol{o}_l$	Output feature map by the encoder at l-th level
C	Number of channels
N_{in}	Number of pixels in $\boldsymbol{m_f}$
N_h	Number of attention heads
N_l	Number of levels
N_p	Number of sampling points
$\mathbb{R}$	Real number
$\boldsymbol{P}$	Reference points
$\boldsymbol{E}$	Position and level embedding information
$\boldsymbol{O}$	Output of the encoder
f	Linear layer
$\boldsymbol{W}$	Attention weights
$\boldsymbol{V}$	Value tensor
$\boldsymbol{\Delta P}$	Sampling offsets
$\boldsymbol{Q}$	Query tensor
$\rightarrow [d_1,\ d_2]$	Reshape to dimension $d_1 \times d_2$