A. MegaMedical

Preprocessing. Medical images involve large variations of voxel or pixel values. For example, MRI intensities in MegaMedical range from [0, 800], CT intensities range from [-2000, 2000], while other modalities might already be in the $[0, 1]$ range.

To normalize data across the diverse datasets, we apply several preprocessing steps for each modality. For MRI datasets, we clip the intensity to $[0.5, 99.5]$ percentiles for non-zero voxels. For CT images, we clip intensity values to the range $[-500, 1000]$. We min-max normalize all resulting volumes to $[0, 1]$ and resize them to $128 \times 128 \times 128$. From any 3D volumes, we extract two different kinds of slices: *mid-slices* and *max-slices*.

Slicing. For *mid-slices*, from any 3D image and label volumes we extract the middle slice along each axis, resulting in a representative 128×128 slice. This strategy avoids biasing the data toward knowing the location of labels in the scans. This is especially important for inference, where the location of the foreground label would not be known in a 3D volume.

For training, we also extract *max slices*. For each label l of a dataset, we find the slice (along each axis) in each volume that contains the most voxels with that label. We extract this slice from both volume and label map and repeat this for all labels in the dataset. These slices provide additional training data, and we do not use them during evaluation.

Label Maps. Most datasets include label maps that were either manually obtained, or manually curated after being obtained using an automatic tool. For adult brain datasets [28, 70, 71], we follow recent large-scale analyses [7, 38] and obtain semantic sub-cortical segmentations using FreeSurfer [26].

Datasets can often contain multiple tasks – such as segmenting both lesions and anatomy – and the same task can appear in different datasets – like segmenting the hippocampus in different MRI collections. Certain labels can sometimes tackle the same anatomical region of interest but be defined differently in two different datasets. In this work, we focus on single-label, single-modality, and 2D segmentation.

Medical Task Creation. To create a task, the subjects of dataset d can contain labels for either a particular biomedical target (e.g. eye-vessels [99], vertebrae [114], white blood cells [115]) or a set of targets (e.g. abdominal organs [49], brain regions [70]), and an imaging modality $m \in M_d$ (e.g. CT, MRI, X-Ray). If d is multi-class, we split it into several single-label $l \in L_d$ tasks. If d is a 3D dataset, we extract different axes $a \in A_d$ as different tasks. Following this construction, each task can be described using a unique tuple $t = (d, m, l, a)$.

Table 3: We assembled the following set of datasets to train UniverSeg. For the relative size of datasets, we have included the number of unique scans (subject and modality pairs) that each dataset has.

Dataset Name	**Description**	**# of Scans**	**Image Modalities**
AbdomenCT-1K [68]	Abdominal organ segmentation (overlap with KiTS, MSD)	1000	CT
ACDC [10]	Left and right ventricular endocardium	99	cine-MRI
AMOS [43]	Abdominal organ segmentation	240	CT, MRI
BBBC003 [65]	Mouse embryos	15	Microscopy
BrainDevelopment [29, 30, 56, 92]	Adult and Neonatal Brain Atlases	53	multi-modal MRI
BRATS [5, 6, 73]	Brain tumors	6,096	multi-modal MRI
BTCV [58]	Abdominal Organs	30	CT
BUS [111]	Breast tumor	163	Ultrasound
CAMUS [59]	Four-chamber and Apical two-chamber heart	500	Ultrasound
CDemris [47]	Human Left Atrial Wall	60	CMR
CHAOS [48, 50]	Abdominal organs (liver, kidneys, spleen)	40	CT, T2-weighted MRI
CheXplanation [89]	Chest X-Ray observations	170	X-Ray
CoNSeP [31]	Histopathology Nuclei	27	Microscopy
DRIVE [99]	Blood vessels in retinal images	20	Optical camera
EOphtha [19]	Eye Microaneurysms and Diabetic Retinopathy	102	Optical camera
FeTA [83]	Fetal brain structures	80	Fetal MRI
FetoPlac [9]	Placenta vessel	6	Fetoscopic optical camera
HMC-QU [20, 54]	4-chamber (A4C) and apical 2-chamber (A2C) left wall	292	Ultrasound
I2CVB [60]	Prostate (peripheral zone, central gland)	19	T2-weighted MRI
IDRID [84]	Diabetic Retinopathy	54	Optical camera
ISLES [36]	Ischemic stroke lesion	180	multi-modal MRI
KiTS [35]	Kidney and kidney tumor	210	CT
LGGFlair [15, 72]	TCIA lower-grade glioma brain tumor	110	MRI
LiTS [11]	Liver Tumor	131	CT
LUNA [93]	Lungs	888	CT
MCIC [28]	Multi-site Brain regions of Schizophrenic patients	390	T1-weighted MRI
MSD [96]	Large-scale collection of 10 Medical Segmentation Datasets	3,225	CT, multi-modal MRI
NCI-ISBI [13]	Prostate	30	T2-weighted MRI
OASIS [38, 70]	Brain anatomy	414	T1-weighted MRI
OCTA500 [61]	Retinal vascular	500	OCT/OCTA
PanDental [2]	Mandible and Teeth	215	X-Ray
PROMISE12 [63]	Prostate	37	T2-weighted MRI
PPMI [71, 18]	Brain regions of Parkinson patients	1,130	T1-weighted MRI
ROSE [69]	Retinal vessel	117	OCT/OCTA
SCD [85]	Sunnybrook Cardiac Multi-Dataset Collection	100	cine-MRI
SegTHOR [57]	Thoracic organs (heart, trachea, esophagus)	40	CT
SpineWeb [114]	Vertebrae	15	T2-weighted MRI
STARE [40]	Blood vessels in retinal images	20	Optical camera
TUCC [1]	Thyroid nodules	167	Ultrasound cine-clip
WBC [115]	White blood cell and nucleus	400	Microscopy
WMH [55]	White matter hyper-intensities	60	multi-modal MRI
WORD [67]	Organ segmentation	120	CT

B. Additional Implementation Details

Data Storage. For each gradient step, a UniverSeg model needs to load $B \times (N+1) \times 2$ images, where B is the batch size, N is the support size, and the factor of 2 corresponds to the combination of the image and label map. This can pose a serious challenge for traditional data loading strategies, especially as N increases. Therefore, we store data samples in a highly optimized way to ensure that I/O does not bottleneck the training process, using LMDB data stores that are optimized for read-only access. Within the database, data is encoded using msgpack and compressed with the LZ4 codec for fast decompression. We find that this setup exceeds regular file-system random access by over two orders of magnitude.

Task Sampling. To ensure task and data heterogeneity during training, we do not sample all tasks equally. Some datasets contain substantially more tasks than others, and we aim to avoid overfitting medical domains where tasks are abundant (such as neuroimaging tasks). Instead, we perform hierarchical uniform sampling with multiple stages: dataset, subject group, acquisition modality, axis, and label. We first sample the dataset uniformly from all datasets, then sample a task among the tasks from that dataset, and so on.

Model. We implemented UniverSeg in PyTorch [82] and used the official implementations for the baselines (ALPNet, PANet, and SENet) and supervised network nnUNet. Based on the experimental details in the ALPNet work, we used an off-the-shelf ResNet101 [34] for both the pre-trained encoder for ALPNet and PANet. For these two methods, because their feature encoder expects three-channel inputs, we duplicate the input dimension $1 \times 128 \times 128$ three times channel-wise to get inputs of dimension $3 \times 128 \times 128$.

We efficiently perform the CrossConvolution operation by exploiting the batch dimension. Instead of performing N convolutions with the same learnable parameters, we perform a single convolution by tiling the inputs along the batch dimension. We use the same strategy for the convolutions predicting the CrossBlock outputs V'.

Optimization. For all models during training, we minimize the soft Dice loss:

$$\mathcal{L}_{\text{Dice}}(y_t, \hat{y}) = 1 - \frac{2\sum y_t \odot \hat{y}}{\sum y_t^2 + \sum \hat{y}^2}, \tag{4}$$

using a learning rate of $\eta = 10^{-4}$, the Adam optimizer[53], and a batch size of 1. We searched learning rates over the range $[10^{-5}, 10^{-2}]$ and found the best results on the validation split of the training datasets with learning rates around 10^{-4} and set on 10^{-4} for comparison and reproducibility purposes.

Evaluation. We evaluate predicted label maps $\hat{y}$ using the Dice score [21], which quantifies the overlap between two regions and is widely used in the segmentation literature:

$$\text{Dice}(y_t, \hat{y}) = 100 * \frac{2|y_t \cap \hat{y}|}{|y_t|^2 + |\hat{y}|^2} \tag{5}$$

where y is the ground truth segmentation map and $\hat{y}$ is the predicted segmentation map. A Dice score of 100 indicates perfectly overlapping regions, while 0 indicates no overlap.

Task-Specific Networks The nnUNet framework trains 5 networks per task using multiple folds of the support data for training, and ensemble their predictions at inference. We apply the nnUNet framework independently for each held-out task, which corresponds to a set of subjects and the segmentation labels for a particular binary task.

We also designed and trained additional individual U-Net networks. For the majority of the tasks, we found the best results after searching batch sizes and augmentation policies. We omitted these as we found that the nnUNets performed very similarly.

C. Data Augmentation

During UniverSeg training, we found that using substantial data augmentation was important. Augmentation techniques enable UniverSeg to see effectively both a greater diversity of tasks as well as a greater number of examples of each. We separate these two kinds of augmentations into *Task* and *In-Task*.

In Table 4, we detail included augmentations. During model development, we experimented to find the hyperparameters which worked best for each kind of augmentation. Several augmentations are repeated (although with different parameters) across task and in-task sections of Table 4.

Table 4: **List of augmentations used in model training.**

Augmentation	Aug Type	Parameter Details
Flip Intensities	Task	$\mathbf{p} = 0.50$
Flip Labels	Task	$\mathbf{p} = 0.50$
Horizontal/Vertical Flip	Task	$\mathbf{p} = 0.50$
Sobel-Edge Label	Task	$\mathbf{p} = 0.50$
Task Affine Shift	Task	$\mathbf{p} = 0.50$ **degrees** $= [0, 360]$ **translate** $= [0, 0.2]$ **scale** $= [0.8, 1.1]$
Task Brightness Contrast Change	Task	$\mathbf{p} = 0.50$ **brightness** $= [-0.1, 0.1]$ **contrast** $= [0.8, 1.2]$
Task Elastic Warp	Task	$\mathbf{p} = 0.25$ $\boldsymbol{\alpha} = [1, 2]$ $\boldsymbol{\sigma} = [6, 8]$
Task Gaussian Blur	Task	$\mathbf{p} = 0.50$ **k-size** $= 5$ $\boldsymbol{\sigma} = [0.1, 1.1]$
Task Gaussian Noise	Task	$\mathbf{p} = 0.50$ $\boldsymbol{\mu} = [0, 0.05]$ $\boldsymbol{\sigma^2} = [0, 0.05]$
Task Sharpness Change	Task	$\mathbf{p} = 0.50$ **sharpness** $= 5$
Example Affine Shift	In-Task	$\mathbf{p} = 0.50$ **degrees** $= [0, 360]$ **translate** $= [0, 0.2]$ **scale** $= [0.8, 1.1]$
Example Brightness Contrast Change	In-Task	$\mathbf{p} = 0.25$ **brightness** $= [-0.1, 0.1]$ **contrast** $= [0.5, 1.5]$
Example Gaussian Blur	In-Task	$\mathbf{p} = 0.25$ **k-size** $= 5$ $\boldsymbol{\sigma} = [0.1, 1.1]$
Example Gaussian Noise	In-Task	$\mathbf{p} = 0.25$ $\boldsymbol{\mu} = [0, 0.05]$ $\boldsymbol{\sigma^2} = [0, 0.05]$
Example Sharpness Change	In-Task	$\mathbf{p} = 0.25$ **sharpness** $= 5$
Example Variable Elastic Warp	In-Task	$\mathbf{p} = 0.80$ $\boldsymbol{\alpha} = [1, 2.5]$ $\boldsymbol{\sigma} = [7, 8]$

We briefly describe each augmentation and its parameters. Each augmentation also has a parameter **p** which controls the probability that augmentation is applied at each iteration. For in-task augmentation, this probability controls whether or not **all** of the support set entries are individually augmented or not. For operations that we developed, we include examples in Figure 10.

- *Flip Intensities* (Task): Flip the intensity values for all images (query and support), but not the label maps, using 1 - image for each.
- *Flip Labels* (Task): Reverse the foreground and background in the segmentation maps.
- *Horizontal/Vertical Flip* (Task): Flip all entries in the support horizontally or vertically (all flipped in the same way).
- *Sobel-Edge Label* (Task): We propose an operation that increases the number of tasks with thin segmentation structures. We apply a Sobel filter to each label map in the x and y directions, compute the squared norm, which becomes our new label map.
- *Affine Shift* (Task, In-Task): Apply a consistent random affine transformation to all entries in the support set; **degrees** controls how much to randomly rotate, **translate** controls how far the images and labels can shift, and **scale** controls the amount of zoom.
- *Brightness Contrast Change* (Task, In-Task): Apply a random brightness and contrast change to all images; how much brightness can change is controlled by **brightness** and contrast is controlled by the parameter **contrast**.
- *Elastic Warp* (Task, In-Task): Apply a consistent elastic deformation warp to all entries in the support and to the query; $\boldsymbol{\alpha}$ controls the strength of the warp and $\boldsymbol{\sigma}$ controls the smoothness of the warp.
- *Gaussian Blur* (Task, In-Task): Apply a convolutional Gaussian blur to each image in the support set and the query with a certain kernel size, **k-size**, and standard-deviation $\boldsymbol{\sigma}$.

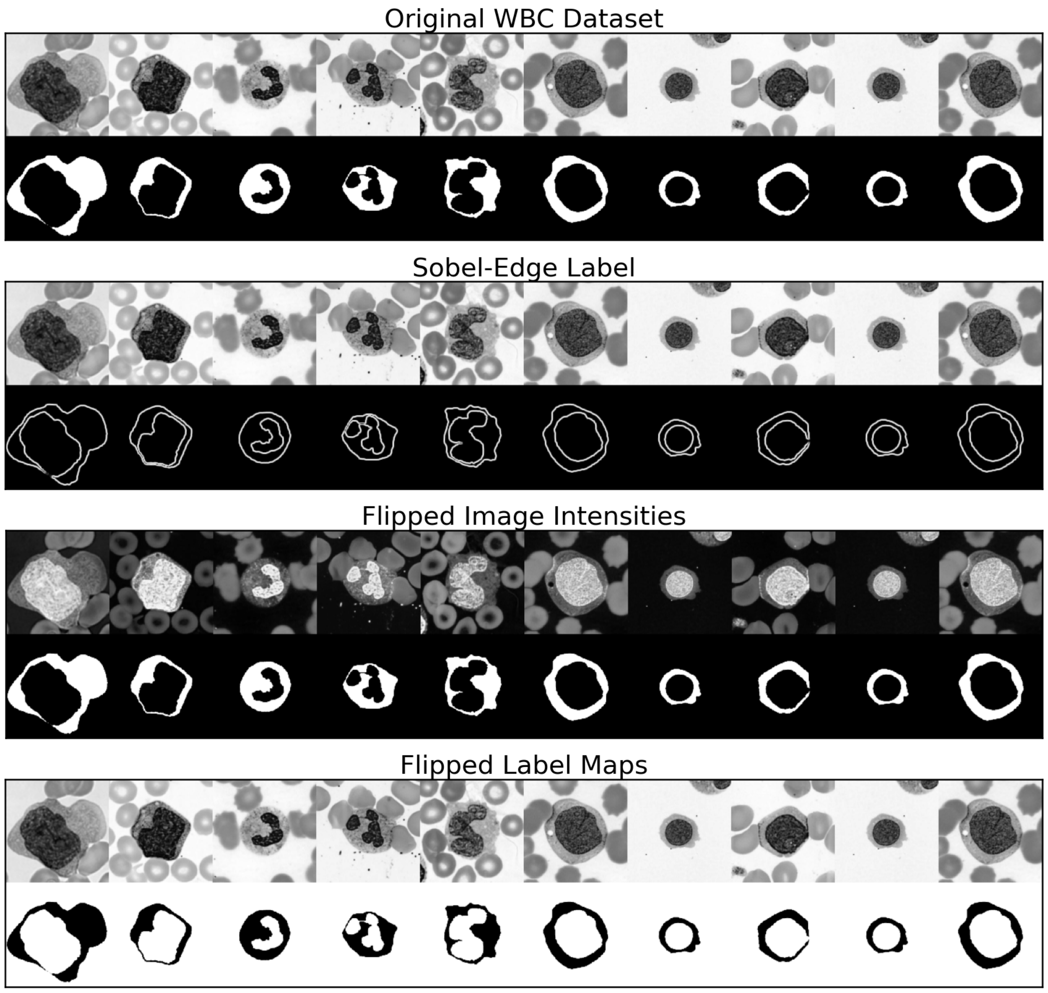


Figure 10: **Example augmentation operations applied to the WBC Dataset.** We visualize several examples of unique task augmentations we apply during training.

- *Gaussian Noise* (Task, In-Task): Apply Gaussian noise to all images in the support set and query with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma^2}$.
- *Sharpness Change* (Task, In-Task): Apply a sharpness filter to the images (query and support), where the sharpness strength is controlled by **sharpness**.

D. Synthetic Tasks

We found improvement in held-out performance by introducing synthetic tasks during training, building on recent methods that use synthetic medical images to solve specific tasks [12, 37, 39], especially the synthetic shapes in SynthMorph [37]. We generate 1,000 new tasks with high diversity (Figure 12). As shown in Figure 11, for each task, we first synthesize a label map of 16 random shapes, representing 16 regions of interest. We deform this label map with 100 random smooth deformation fields, representing 100 subjects with the same simulated anatomy. We then add texture to the resulting images by filling in each region of interest with slightly varied intensities around a sampled mean and adding Gaussian and Perlin noise.

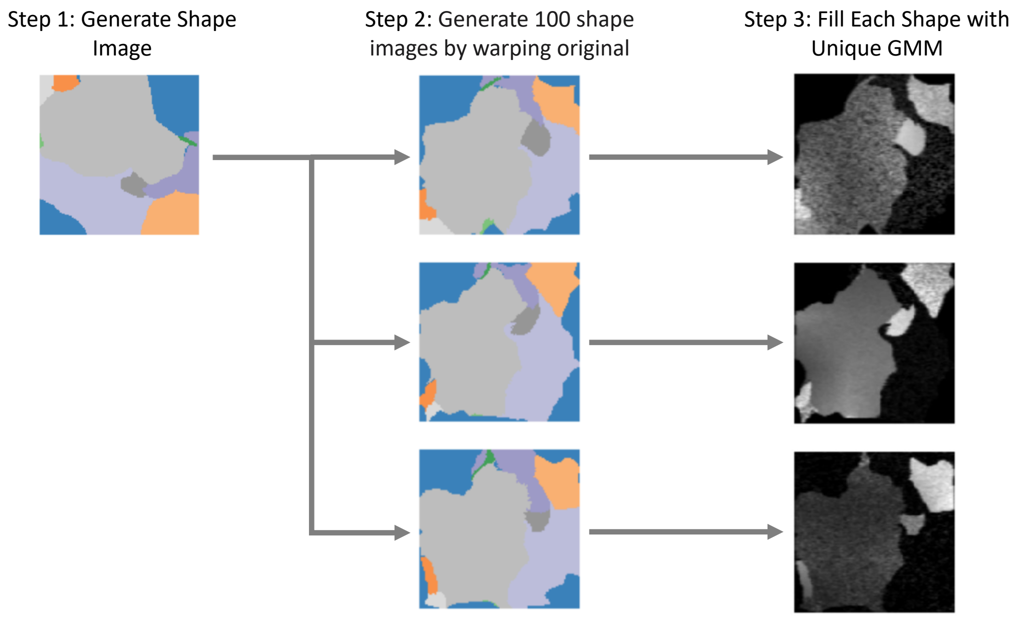

Step 1: Generate Shape Image
Step 2: Generate 100 shape images by warping original
Step 3: Fill Each Shape with Unique GMM

Figure 11: **Generation process for synthetic tasks.** For a new synthetic task, we first generate random shapes to obtain a label map, then synthesize 100 spatial variations on this label map, and finally synthesize resulting intensity images. We repeat this process for 1000 tasks.

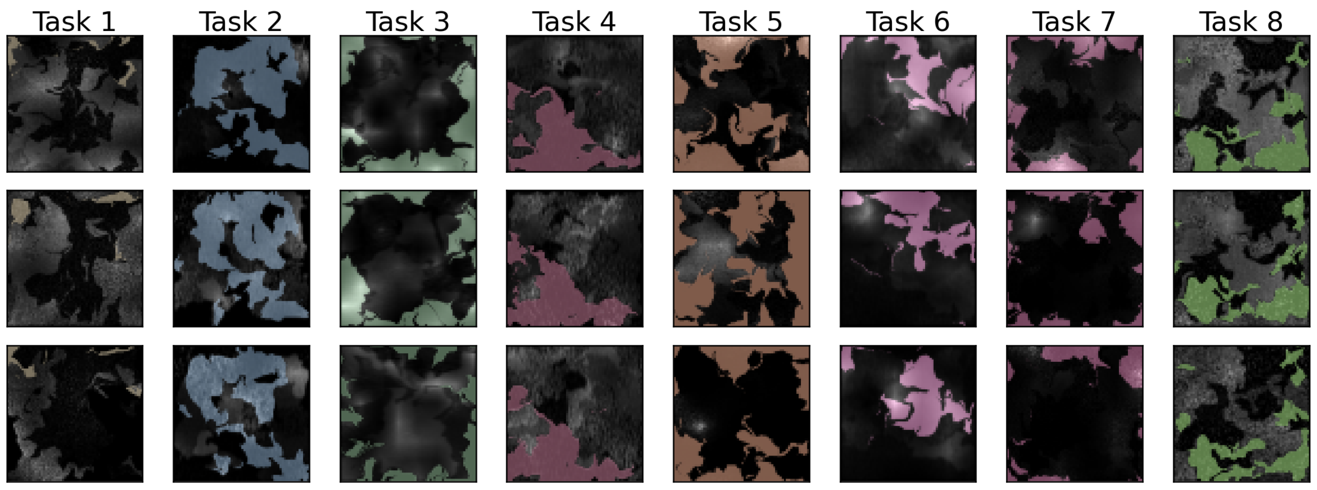


Figure 12: **Examples of Synthetically Generated Tasks.** We visualize 10 of the 1000 synthetically generated tasks, involving varying shapes, textures, and label shapes.

E. Extended Results

E.1. Main Results

We include detailed numbers corresponding to figures in the main body of the paper.

- **Method Comparison**. Table 5 reports test performance numbers of the results from Figure 1 and Table 1, comparing the segmentation results of UniverSeg to the FS baselines and the supervised nnUNet upper bounds.
- **Training Strategies Ablation**. Table 6 reports per-dataset test performance numbers for the results of Table 2 comparing several ways of augmenting the task diversity artificially. While the overall trend holds for most datasets, we find that the increase in task diversity has a detrimental effect on the STARE eye vessel segmentation task.
- **Model Support Size**. Table 7 reports held-out test performance numbers of the results from Figure 7 along with per-dataset breakdowns. We find that the global trend holds for each individual dataset, with larger support sizes achieving better results and ensembling (with $K = 10$) consistently improving predictions.
- **Available Data for Inference Ablation**. Table 8 reports extended results from Figure 8 with per-task results as we change the size of the support example pool. All tasks showcase the same trend with consistent improvements as more support examples are used during inference and with a reduced variance across random subsets of the support split.
- **Support Set Ensembling**. Table 9 reports results for the support set ensembling experiment. We observe a clear difference between $N = 1$ and $N > 1$ for ensembled predictions. For $N = 1$ ensembling leads to small improvements that eventually decline as K grows. In contrast for $N > 1$, ensembling leads to substantial improvements that also reduce the variance of the distribution, limiting the dependence on the specific subset used for the support set.
- **Number of Tasks Ablation**. Table 10 reports the per-dataset and global dice numbers for the models trained with a subset of the training datasets.
- **Effect of Support Size on Training and Inference Costs**. Table 11 reports per-iteration costs for training and inference as we vary the number of examples in the support set. Figure 13 illustrates the inference trend, where we see that for $N > 4$, inference latency and peak memory requirements scale linearly as we increase the support set size.

Table 5: **Method Comparison**. Test Dice Score for the baselines, UniverSeg, and the nnUNet upper bounds in each of the held-out datasets. Standard deviation is computed by bootstrapping subjects before hierarchically averaging the data.

Model	**ACDC**	**PanDental**	**SCD**	**STARE**	**SpineWeb**	**WBC**	**All (avg.)**
ALPNet	34.6 ± 2.4	72.9 ± 0.8	53.4 ± 3.0	17.8 ± 1.9	31.6 ± 4.6	76.2 ± 1.1	47.8 ± 1.1
PANet	27.8 ± 4.3	67.7 ± 0.8	58.9 ± 3.4	20.1 ± 3.2	21.8 ± 0.4	54.7 ± 1.6	41.8 ± 1.3
SENet	40.1 ± 2.0	81.1 ± 0.9	55.4 ± 3.3	35.2 ± 2.2	18.3 ± 4.0	70.8 ± 1.3	50.1 ± 1.3
UniverSeg (ours)	$\mathbf{70.9 \pm 2.9}$	$\mathbf{87.5 \pm 0.9}$	$\mathbf{69.0 \pm 2.9}$	$\mathbf{48.1 \pm 2.0}$	$\mathbf{64.6 \pm 5.4}$	$\mathbf{90.6 \pm 1.1}$	$\mathbf{71.8 \pm 0.9}$
nnUNet (sup.)	82.5 ± 2.3	92.9 ± 1.1	75.0 ± 3.4	65.5 ± 1.1	91.2 ± 2.3	95.1 ± 0.7	84.4 ± 1.0

Table 6: **Training Stategies Ablation**. Per dataset held-out Dice for UniverSeg models trained with different combinations of the proposed techniques to increase task diversity: in-task augmentation, task augmentation, and synthetic tasks.

Synth	**Medical**	**In-Task**	**Task**	**ACDC**	**PanDental**	**SCD**	**STARE**	**SpineWeb**	**WBC**
✓				55.4 ± 3.4	80.6 ± 1.3	55.7 ± 2.4	42.6 ± 2.5	50.1 ± 6.5	86.0 ± 1.4
	✓			44.9 ± 1.8	85.3 ± 0.9	59.9 ± 1.9	**63.8** ± **0.9**	40.3 ± 6.0	82.0 ± 1.6
✓	✓			50.6 ± 2.9	85.7 ± 0.9	59.0 ± 1.9	61.9 ± 1.6	45.6 ± 4.8	84.2 ± 1.4
	✓	✓		52.3 ± 4.3	86.5 ± 0.9	64.9 ± 2.7	56.0 ± 2.3	57.2 ± 3.7	85.1 ± 1.4
	✓		✓	68.0 ± 3.0	**87.5** ± **1.0**	63.5 ± 2.3	56.6 ± 2.1	57.8 ± 6.6	89.2 ± 1.3
	✓	✓	✓	**70.0** ± **2.8**	**88.0** ± **0.9**	**71.2** ± **3.1**	42.2 ± 2.1	58.4 ± 8.5	**90.3** ± **1.2**
✓	✓	✓	✓	**70.9** ± **2.9**	**87.5** ± **0.9**	**69.0** ± **2.9**	48.1 ± 2.0	**64.6** ± **5.4**	**90.6** ± **1.1**

Table 7: **Model Support Size**. Comparison of predictions for models trained with various of support sizes N and evaluated with and without ensembling $K = 10$ predictions. We report results on each held-out dataset as well as the global average. Standard deviation is computed by bootstrapping subjects before hierarchically averaging the data. For all datasets, we find that increasing the support size leads to better predictions, with diminishing returns after $N > 16$. Ensembling predictions significantly improve performance in the majority of settings (paired t-test).

N	**K**	**ACDC**	**PanDental**	**SCD**	**STARE**	**SpineWeb**	**WBC**	**All (avg.)**
1	1	41.3 ± 1.3	76.3 ± 0.9	60.2 ± 1.8	37.4 ± 3.8	30.4 ± 5.5	74.0 ± 1.2	53.3 ± 1.0
	10	44.5 ± 2.4	79.1 ± 1.0	60.0 ± 1.9	38.5 ± 4.0	32.4 ± 6.6	79.4 ± 1.4	55.7 ± 1.1
2	1	41.3 ± 2.6	80.0 ± 1.0	63.5 ± 2.0	40.4 ± 2.1	38.0 ± 4.3	77.6 ± 1.1	56.8 ± 1.0
	10	42.8 ± 3.2	82.4 ± 1.1	68.0 ± 2.5	40.7 ± 2.3	43.4 ± 4.1	82.3 ± 1.4	60.0 ± 1.2
4	1	53.9 ± 1.9	83.9 ± 1.0	64.7 ± 1.7	47.9 ± 2.9	45.5 ± 4.0	82.7 ± 1.4	63.1 ± 0.8
	10	57.0 ± 2.6	84.6 ± 1.1	66.4 ± 2.8	48.6 ± 2.9	50.8 ± 4.1	85.7 ± 1.5	65.5 ± 0.8
8	1	57.0 ± 2.5	85.0 ± 0.9	66.9 ± 3.2	45.9 ± 3.5	57.3 ± 6.5	83.7 ± 1.5	66.0 ± 1.3
	10	61.6 ± 3.3	86.1 ± 0.9	69.0 ± 4.1	47.1 ± 3.5	62.3 ± 6.0	85.9 ± 1.5	68.6 ± 1.3
16	1	64.1 ± 2.4	86.1 ± 0.9	69.1 ± 3.1	48.8 ± 3.0	64.4 ± 5.8	86.9 ± 1.4	69.9 ± 1.0
	10	66.8 ± 2.5	86.7 ± 0.9	68.7 ± 3.5	**49.7** ± **2.8**	**66.8** ± **5.7**	88.3 ± 1.5	71.2 ± 1.0
32	1	65.6 ± 3.0	87.1 ± 0.9	69.0 ± 2.0	45.7 ± 2.2	65.8 ± 4.6	87.6 ± 1.3	70.1 ± 0.9
	10	**69.3** ± **2.9**	**87.6** ± **0.9**	**69.5** ± **1.9**	**46.4** ± **2.1**	**66.4** ± **4.3**	88.9 ± 1.4	**71.4** ± **0.8**
64	1	69.0 ± 2.9	87.2 ± 0.9	68.7 ± 2.9	47.2 ± 2.2	64.2 ± 5.5	89.7 ± 1.1	71.0 ± 1.0
	10	**70.9** ± **2.9**	**87.5** ± **0.9**	**69.0** ± **2.9**	**48.1** ± **2.0**	**64.6** ± **5.4**	**90.6** ± **1.1**	**71.8** ± **0.9**

Table 8: **Limited Example Data**. UniverSeg predictions using a limited d_{support} example pool for each held-out task. For each size, we perform 100 repetitions using different random subsets, reporting the mean and standard deviation across them. Since some tasks do not have enough subjects to be evaluated for all values of N, we report $\min(N, |d_{\text{support}}|)$ and omit repeated settings where $N > |d_{\text{support}}|$.

Task	**N = 1**	**N = 2**	**N = 4**	**N = 8**	**N = 16**	**N = 32**	**N = 64**
ACDC	22.9 ± 5.5	38.5 ± 6.9	51.4 ± 4.7	59.1 ± 3.0	64.4 ± 2.2	68.6 ± 1.4	71.0 ± 0.0
PanDental$_0$	59.1 ± 7.4	73.3 ± 4.2	77.6 ± 1.6	80.1 ± 0.8	82.1 ± 0.5	83.2 ± 0.3	83.7 ± 0.1
PanDental$_1$	65.5 ± 3.9	84.1 ± 2.2	87.5 ± 2.7	89.5 ± 1.2	90.6 ± 0.5	91.1 ± 0.3	91.3 ± 0.0
SCD$_0$	34.3 ± 9.2	63.1 ± 5.1	70.8 ± 2.5	73.1 ± 1.2	74.3 ± 0.4	74.2 ± 0.0	
SCD$_1$	33.0 ± 10.4	61.8 ± 9.4	72.8 ± 5.0	76.7 ± 2.4	78.5 ± 0.8	78.6 ± 0.0	
SCD$_2$	45.0 ± 11.4	71.5 ± 12.7	80.6 ± 7.2	84.8 ± 0.0			
SCD$_3$	30.5 ± 9.3	47.1 ± 6.5	54.9 ± 4.5	63.0 ± 2.3	64.2 ± 0.0		
SCD$_4$	9.2 ± 4.4	13.3 ± 8.3	25.9 ± 7.8	39.0 ± 3.1	41.0 ± 0.0		
STARE	25.5 ± 3.5	33.5 ± 2.0	40.2 ± 1.2	45.2 ± 0.5	47.7 ± 0.0		
SpineWeb	28.1 ± 2.1	39.3 ± 6.1	52.1 ± 7.0	63.1 ± 3.3	64.7 ± 0.0		
WBC$_0$	49.4 ± 4.5	65.0 ± 4.3	74.8 ± 3.0	81.0 ± 1.9	85.0 ± 1.2	87.5 ± 0.8	88.8 ± 0.0
WBC$_1$	57.4 ± 4.8	75.2 ± 3.5	83.0 ± 2.1	87.4 ± 1.0	89.9 ± 0.4	91.3 ± 0.3	91.9 ± 0.2

Table 9: **Ensembling predictions at different inference support sizes.** For each inference support size N, we report the results (in average held-out Dice Score) of taking 100 predictions ($K = 1$) and ensembling by averaging in groups of size K, performing 100 repetitions for each K. We report the mean and standard deviation across the 100 values for each setting and find that increasing either K or N leads to improved model performance, with N having a significantly larger effect than K.

N	**K = 1**	**K = 2**	**K = 4**	**K = 8**	**K = 16**	**K = 32**	**K = 64**
1	36.9 ± 2.0	39.4 ± 2.3	40.7 ± 2.0	40.9 ± 1.6	40.3 ± 1.1	39.4 ± 0.7	38.3 ± 0.4
2	51.0 ± 3.2	56.3 ± 2.2	59.5 ± 1.6	61.0 ± 1.2	61.9 ± 0.9	62.3 ± 0.6	62.4 ± 0.3
4	59.4 ± 2.3	63.7 ± 1.5	66.2 ± 1.0	67.5 ± 0.7	68.2 ± 0.4	68.6 ± 0.3	68.8 ± 0.2
8	64.8 ± 1.9	68.0 ± 1.1	69.6 ± 0.6	70.5 ± 0.4	71.1 ± 0.2	71.3 ± 0.2	71.4 ± 0.1
16	68.4 ± 1.1	70.1 ± 0.5	71.0 ± 0.4	71.5 ± 0.3	71.8 ± 0.2	71.9 ± 0.1	72.0 ± 0.1
32	70.1 ± 0.6	71.0 ± 0.3	71.5 ± 0.2	71.7 ± 0.1	71.8 ± 0.1	71.9 ± 0.1	71.9 ± 0.0
64	71.0 ± 0.3	71.4 ± 0.2	71.6 ± 0.2	71.7 ± 0.1	71.8 ± 0.1	71.8 ± 0.1	71.8 ± 0.0

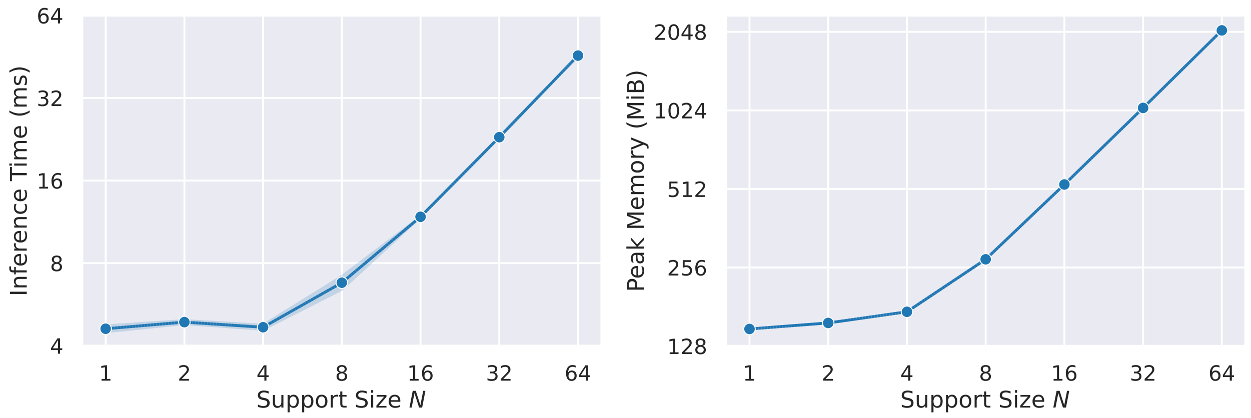


Figure 13: **Effect of Support Set Size on Inference Computational Cost**. UniverSeg inference latency (left) and peak GPU memory (right) both linearly increase with the support set size. Measurements are performed on an NVIDIA V100 GPU with a batch size of 1.

Table 10: **Number of Training Datasets and Tasks**. Test Dice score results for models trained with N_D datasets comprising N_T tasks. The subsets of the training datasets are chosen randomly so we report three realizations for each N_D, except for the case where all datasets are included. Each row corresponds to a separate UniverSeg model.

N_D	N_T	ACDC	PanDental	SCD	STARE	SpineWeb	WBC	All (avg)
	25	24.7 ± 2.8	82.2 ± 0.8	43.7 ± 3.3	7.2 ± 2.5	0.2 ± 0.2	61.1 ± 1.3	36.5 ± 0.8
1	29	3.3 ± 2.4	18.4 ± 0.7	0.2 ± 0.1	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3.7 ± 0.4
	156	63.1 ± 2.7	80.0 ± 1.8	53.5 ± 5.3	17.5 ± 2.3	46.2 ± 1.6	86.2 ± 1.3	57.8 ± 1.1
	33	41.7 ± 3.7	83.8 ± 1.0	43.9 ± 2.3	16.9 ± 1.4	22.2 ± 3.3	80.0 ± 1.5	48.1 ± 1.1
2	85	49.5 ± 3.5	83.1 ± 1.1	59.9 ± 2.5	22.9 ± 2.1	52.9 ± 2.1	85.5 ± 1.4	59.0 ± 1.0
	131	31.9 ± 2.5	82.0 ± 0.8	42.3 ± 3.4	0.0 ± 0.0	10.2 ± 3.2	69.5 ± 1.3	39.3 ± 0.6
	237	43.6 ± 5.5	75.3 ± 1.3	52.1 ± 2.8	26.8 ± 3.4	28.3 ± 10.1	86.3 ± 1.2	52.1 ± 2.2
5	710	63.9 ± 3.6	86.6 ± 1.0	63.5 ± 2.2	27.2 ± 3.3	61.1 ± 5.0	87.4 ± 1.6	64.9 ± 1.3
	1117	67.4 ± 3.1	86.5 ± 1.0	62.9 ± 4.5	51.6 ± 2.4	58.4 ± 8.2	89.9 ± 1.0	69.4 ± 2.0
	174	51.2 ± 2.8	84.0 ± 0.9	66.5 ± 2.7	22.8 ± 1.3	52.2 ± 0.4	84.5 ± 1.1	60.2 ± 0.8
11	1223	60.6 ± 3.6	86.8 ± 1.0	59.0 ± 4.9	29.7 ± 2.4	58.4 ± 5.6	85.5 ± 1.3	63.3 ± 1.7
	1457	69.7 ± 2.8	87.5 ± 0.9	65.4 ± 3.3	40.5 ± 3.0	59.6 ± 7.0	88.6 ± 1.1	68.6 ± 1.7
	1320	66.9 ± 3.5	85.3 ± 0.9	68.1 ± 2.4	31.2 ± 0.7	57.2 ± 5.9	88.4 ± 1.3	66.2 ± 1.1
23	2157	66.5 ± 3.0	86.1 ± 1.0	64.0 ± 2.3	36.9 ± 1.1	64.9 ± 5.4	89.6 ± 1.3	68.0 ± 1.1
	2276	66.5 ± 3.8	86.7 ± 0.9	65.7 ± 2.8	28.1 ± 2.4	52.4 ± 6.6	89.1 ± 1.2	64.8 ± 1.4
	3008	69.8 ± 3.0	88.8 ± 0.9	68.3 ± 2.6	42.5 ± 3.2	62.4 ± 6.1	90.0 ± 1.2	70.3 ± 1.4
34	3483	70.7 ± 2.9	87.1 ± 1.0	67.2 ± 3.5	46.7 ± 3.6	63.0 ± 5.0	90.7 ± 1.4	70.9 ± 1.0
	3854	65.3 ± 3.6	88.2 ± 1.0	65.1 ± 1.6	43.1 ± 3.1	62.2 ± 5.5	89.0 ± 1.1	68.8 ± 1.0
46	4432	71.3 ± 2.6	87.9 ± 0.9	67.9 ± 2.5	44.9 ± 2.9	65.5 ± 4.7	91.0 ± 1.1	71.4 ± 1.1

Table 11: **Effect of Support Size on Training and Inference Costs**. We measure the cost of performing a training iteration (forward, backward pass, and optimizer step), and an inference step (only forward pass) for UniverSeg models with varying support set size N. We report wall-clock time of each operation and the peak GPU memory required. For small support set sizes ($N \leq 4$), the costs are similar, and beyond that, they increase linearly with N. Measurements are performed on an NVIDIA V100 GPU with a batch size of 1.

	Iteration Time (ms)		Peak Memory (MiB)	
N	Inference	Training	Inference	Training
1	4.6	10.4	149	231
2	4.9	12.8	157	288
4	4.7	13.3	173	477
8	6.8	21.2	275	917
16	11.8	35.9	533	1797
32	23.0	69.7	1047	3555
64	45.7	132.2	2074	7073

E.2. Additional Results

Few-shot Baseline Model Variants. The FS baselines (ALPNet, PANet, and SENet) were introduced in a few-shot setting where the underlying assumption is that any new task can only have very few examples, rather than our setting where we avoid re-training due to the limitations of the clinical settings. These baselines were therefore presented with a support size of 1 example. They also involved no data or task augmentation. Our ablations show that UniverSeg models performed best with large support set sizes and increased data and task diversity from augmenting examples. Consequently, we test whether incorporating these changes to the baseline methods leads to improved performance in the held-out datasets in our setting, where more data *might* be available for some datasets. Similarly, we also test whether ensembling predictions from several support sets lead to better predictions, as we do for UniverSeg.

Table 12 and Table 13 report results of the hyperparameter grid search for all the few-shot baseline models and UniverSeg. Table 12 shows that ensembling ($K = 10$) and an increased support size ($N = 64$) leads to held-out improvements for all methods. In contrast, augmentation strategies do not benefit all methods. While UniverSeg and SENet improve when using augmentation strategies, PANet and ALPNet experience a decrease in performance. Table 13 shows that the best hyperparameter setting is not consistent across held-out datasets for the baseline methods.

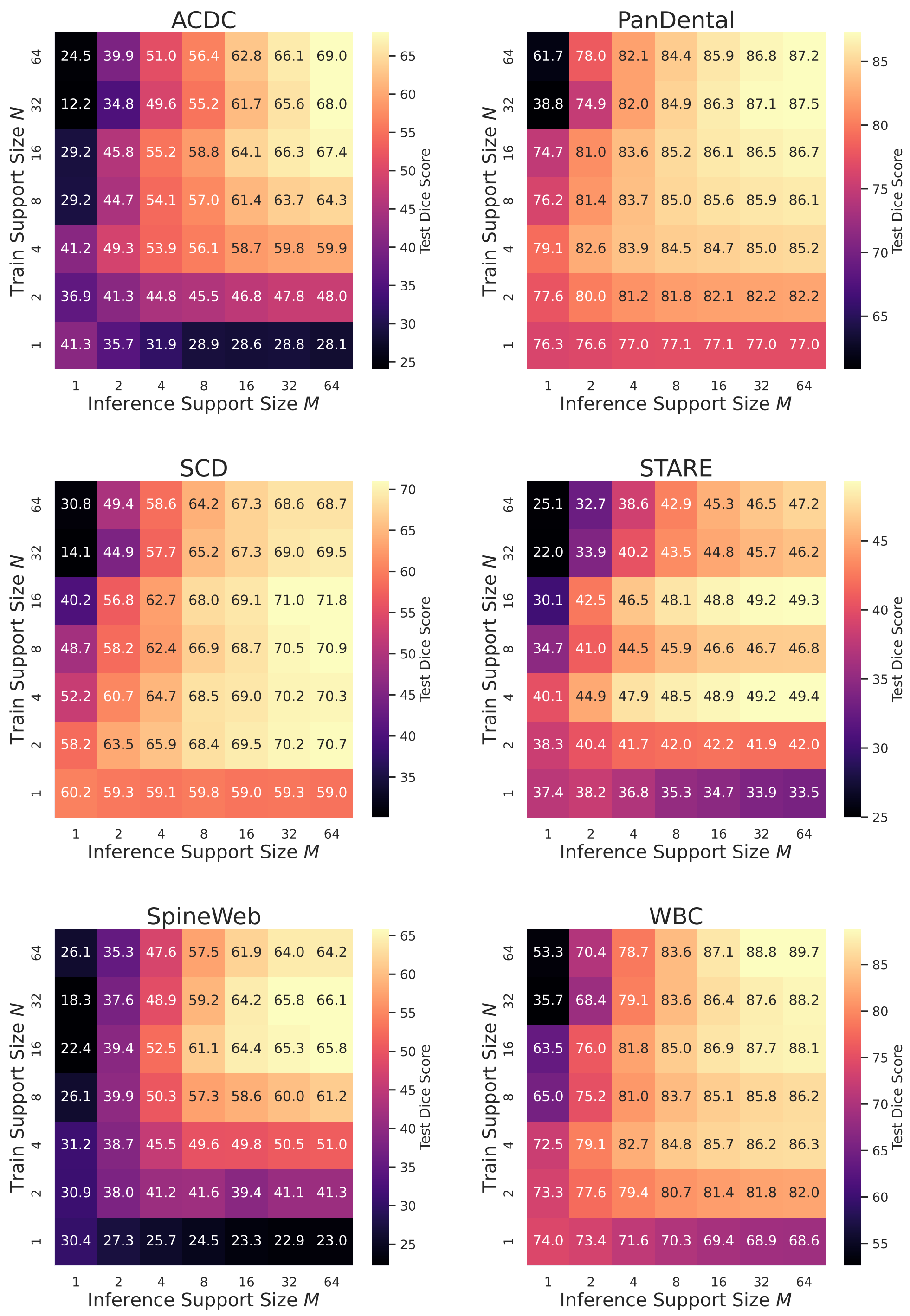
Table 12: **FS baseline hyperparameter search**. For each method, we report results for models trained with a support size N, ensemble size K, and with and without data and task augmentation. Dice scores are averaged across all datasets and the standard deviation is computed via subject-level bootstrapping.

		No Aug		**Aug**	
Model	**N**	K=1	K=10	K=1	K=10
ALPNet	1	40.2 ± 0.9	42.3 ± 1.3	35.4 ± 0.6	37.0 ± 0.8
	64	46.3 ± 1.3	**47.8 ± 1.1**	42.3 ± 1.0	45.2 ± 1.2
PANet	1	37.4 ± 0.7	39.3 ± 0.8	33.2 ± 1.3	34.3 ± 1.4
	64	**41.6 ± 1.3**	**41.8 ± 1.3**	38.7 ± 0.9	40.8 ± 0.8
SENet	1	40.0 ± 0.9	41.2 ± 0.9	40.1 ± 1.2	41.1 ± 1.4
	64	42.1 ± 0.7	42.4 ± 0.8	**50.2 ± 1.1**	**50.1 ± 1.3**
UniverSeg (ours)	1	49.7 ± 0.9	53.4 ± 1.1	51.9 ± 0.8	54.0 ± 1.0
	64	64.0 ± 1.1	64.5 ± 1.0	71.0 ± 1.0	**71.8 ± 0.9**

Table 13: **Few-shot baseline hyperparameter search per dataset**. For each method, we report results for models trained with a support size N, ensemble size $K = 10$, and with and without data and task augmentation. Dice score values are averaged across all datasets and the standard deviation is computed via subject-level bootstrapping. For each dataset and model, we highlight the setting with the best performance

Model	**N**	**Aug**	**ACDC**	**PanDental**	**SCD**	**STARE**	**SpineWeb**	**WBC**
ALPNet	1	No	22.1 ± 3.2	66.8 ± 1.0	49.1 ± 3.8	**22.7 ± 2.0**	29.7 ± 3.7	63.2 ± 0.9
		Yes	26.7 ± 2.9	51.8 ± 1.5	41.5 ± 1.9	11.0 ± 2.8	19.7 ± 4.9	71.0 ± 1.6
	64	No	34.6 ± 2.4	**72.9 ± 0.8**	53.4 ± 3.0	17.8 ± 1.9	**31.6 ± 4.6**	**76.2 ± 1.1**
		Yes	**38.3 ± 2.5**	71.1 ± 1.0	**56.1 ± 1.6**	6.3 ± 2.2	25.5 ± 6.4	73.9 ± 1.2
PANet	1	No	**33.4 ± 2.5**	**69.8 ± 1.3**	48.7 ± 3.5	17.4 ± 4.3	25.4 ± 3.9	40.9 ± 1.8
		Yes	30.3 ± 3.0	63.2 ± 1.3	48.4 ± 3.3	4.6 ± 2.9	**28.6 ± 5.6**	31.0 ± 2.1
	64	No	27.8 ± 4.3	67.7 ± 0.8	**58.9 ± 3.4**	**20.1 ± 3.2**	21.8 ± 0.4	54.7 ± 1.6
		Yes	29.6 ± 2.3	66.4 ± 1.4	46.8 ± 2.3	15.1 ± 2.1	27.9 ± 5.8	**58.8 ± 1.5**
SENet	1	No	17.0 ± 2.9	61.7 ± 1.1	47.5 ± 2.3	**41.3 ± 2.7**	**21.7 ± 3.7**	58.1 ± 0.9
		Yes	32.2 ± 2.8	62.4 ± 1.3	48.2 ± 2.9	31.4 ± 2.1	16.8 ± 7.8	55.5 ± 1.3
	64	No	32.0 ± 2.4	79.1 ± 0.8	43.8 ± 2.9	37.5 ± 2.9	3.2 ± 2.4	58.7 ± 1.1
		Yes	**40.1 ± 2.0**	**81.1 ± 0.9**	**55.4 ± 3.3**	35.2 ± 2.2	18.3 ± 4.0	**70.8 ± 1.3**
UniverSeg	1	No	29.1 ± 2.0	76.1 ± 0.9	58.0 ± 2.1	54.5 ± 2.7	31.6 ± 6.4	70.9 ± 1.7
		Yes	37.5 ± 2.0	76.8 ± 1.1	62.9 ± 2.6	33.9 ± 3.7	36.0 ± 5.0	76.8 ± 1.6
	64	No	50.6 ± 2.9	85.7 ± 0.9	59.0 ± 1.9	**61.9 ± 1.6**	45.6 ± 4.8	84.2 ± 1.4
		Yes	**70.9 ± 2.9**	**87.5 ± 0.9**	**69.0 ± 2.9**	48.1 ± 2.0	**64.6 ± 5.4**	**90.6 ± 1.1**

Using different training and inference support sizes. In Figure 14, we report dataset-level results of performing inference with a support size of M using a UniverSeg model trained with a support size of N examples. We find that using support sets larger than those seen in training ($M > N$, lower quadrant of heat-maps) leads to improvements for $N \geq 2$, which demonstrates the model is learning to interact the elements of the support set and benefits from larger amounts of examples.



ACDC

Train Support Size N \ Inference Support Size M	1	2	4	8	16	32	64
64	24.5	39.9	51.0	56.4	62.8	66.1	69.0
32	12.2	34.8	49.6	55.2	61.7	65.6	68.0
16	29.2	45.8	55.2	58.8	64.1	66.3	67.4
8	29.2	44.7	54.1	57.0	61.4	63.7	64.3
4	41.2	49.3	53.9	56.1	58.7	59.8	59.9
2	36.9	41.3	44.8	45.5	46.8	47.8	48.0
1	41.3	35.7	31.9	28.9	28.6	28.8	28.1

PanDental

Train Support Size N \ Inference Support Size M	1	2	4	8	16	32	64
64	61.7	78.0	82.1	84.4	85.9	86.8	87.2
32	38.8	74.9	82.0	84.9	86.3	87.1	87.5
16	74.7	81.0	83.6	85.2	86.1	86.5	86.7
8	76.2	81.4	83.7	85.0	85.6	85.9	86.1
4	79.1	82.6	83.9	84.5	84.7	85.0	85.2
2	77.6	80.0	81.2	81.8	82.1	82.2	82.2
1	76.3	76.6	77.0	77.1	77.1	77.0	77.0

SCD

Train Support Size N \ Inference Support Size M	1	2	4	8	16	32	64
64	30.8	49.4	58.6	64.2	67.3	68.6	68.7
32	14.1	44.9	57.7	65.2	67.3	69.0	69.5
16	40.2	56.8	62.7	68.0	69.1	71.0	71.8
8	48.7	58.2	62.4	66.9	68.7	70.5	70.9
4	52.2	60.7	64.7	68.5	69.0	70.2	70.3
2	58.2	63.5	65.9	68.4	69.5	70.2	70.7
1	60.2	59.3	59.1	59.8	59.0	59.3	59.0

STARE

Train Support Size N \ Inference Support Size M	1	2	4	8	16	32	64
64	25.1	32.7	38.6	42.9	45.3	46.5	47.2
32	22.0	33.9	40.2	43.5	44.8	45.7	46.2
16	30.1	42.5	46.5	48.1	48.8	49.2	49.3
8	34.7	41.0	44.5	45.9	46.6	46.7	46.8
4	40.1	44.9	47.9	48.5	48.9	49.2	49.4
2	38.3	40.4	41.7	42.0	42.2	41.9	42.0
1	37.4	38.2	36.8	35.3	34.7	33.9	33.5

SpineWeb

Train Support Size N \ Inference Support Size M	1	2	4	8	16	32	64
64	26.1	35.3	47.6	57.5	61.9	64.0	64.2
32	18.3	37.6	48.9	59.2	64.2	65.8	66.1
16	22.4	39.4	52.5	61.1	64.4	65.3	65.8
8	26.1	39.9	50.3	57.3	58.6	60.0	61.2
4	31.2	38.7	45.5	49.6	49.8	50.5	51.0
2	30.9	38.0	41.2	41.6	39.4	41.1	41.3
1	30.4	27.3	25.7	24.5	23.3	22.9	23.0

WBC

Train Support Size N \ Inference Support Size M	1	2	4	8	16	32	64
64	53.3	70.4	78.7	83.6	87.1	88.8	89.7
32	35.7	68.4	79.1	83.6	86.4	87.6	88.2
16	63.5	76.0	81.8	85.0	86.9	87.7	88.1
8	65.0	75.2	81.0	83.7	85.1	85.8	86.2
4	72.5	79.1	82.7	84.8	85.7	86.2	86.3
2	73.3	77.6	79.4	80.7	81.4	81.8	82.0
1	74.0	73.4	71.6	70.3	69.4	68.9	68.6

Figure 14: **Cartesian Product of Training and Inference Support Sizes.** Test results for using a UniverSeg trained with a support size of N examples and performing inference with a support size of M examples. No ensembling is performed ($K = 1$), but we perform 10 repetitions with varying support sets and report the average.

Multi-Instance Generalization. In some segmentation tasks, such as WBC, images can contain several instances of a particular class. Although instance segmentation is beyond the scope of our work, in Figure 15 we visualize the segmentation of five different examples of WBC cells, which have different numbers of nuclei (blue). For these target images, UniverSeg was given the same support set, with each support example having only one nucleus. We find that UniverSeg is able to segment all visible nuclei, suggesting that UniverSeg could be used or adapted to multi-instance segmentation tasks.

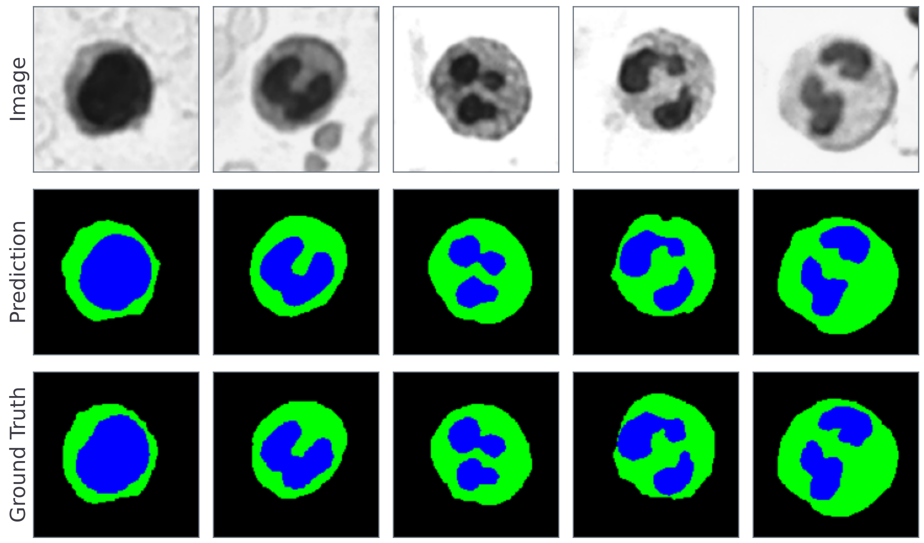


Figure 15: **Multi-label UniverSeg predictions for WBC images with one or two instances of nuclei (blue), cytoplasm (green), and background (black). The support set consists of mono-nucleic cells.**

F. Additional Visualizations

Visualization of Held-Out Support Sets. UniverSeg networks take advantage of large support sets of (image, label-map) pairs, which can be very diverse. In Figure 16, we visualize a random subset of 10 pairs for each held-out dataset. The diversity of subjects within support sets differs between tasks, which likely plays a role in the number of examples required to perform well.

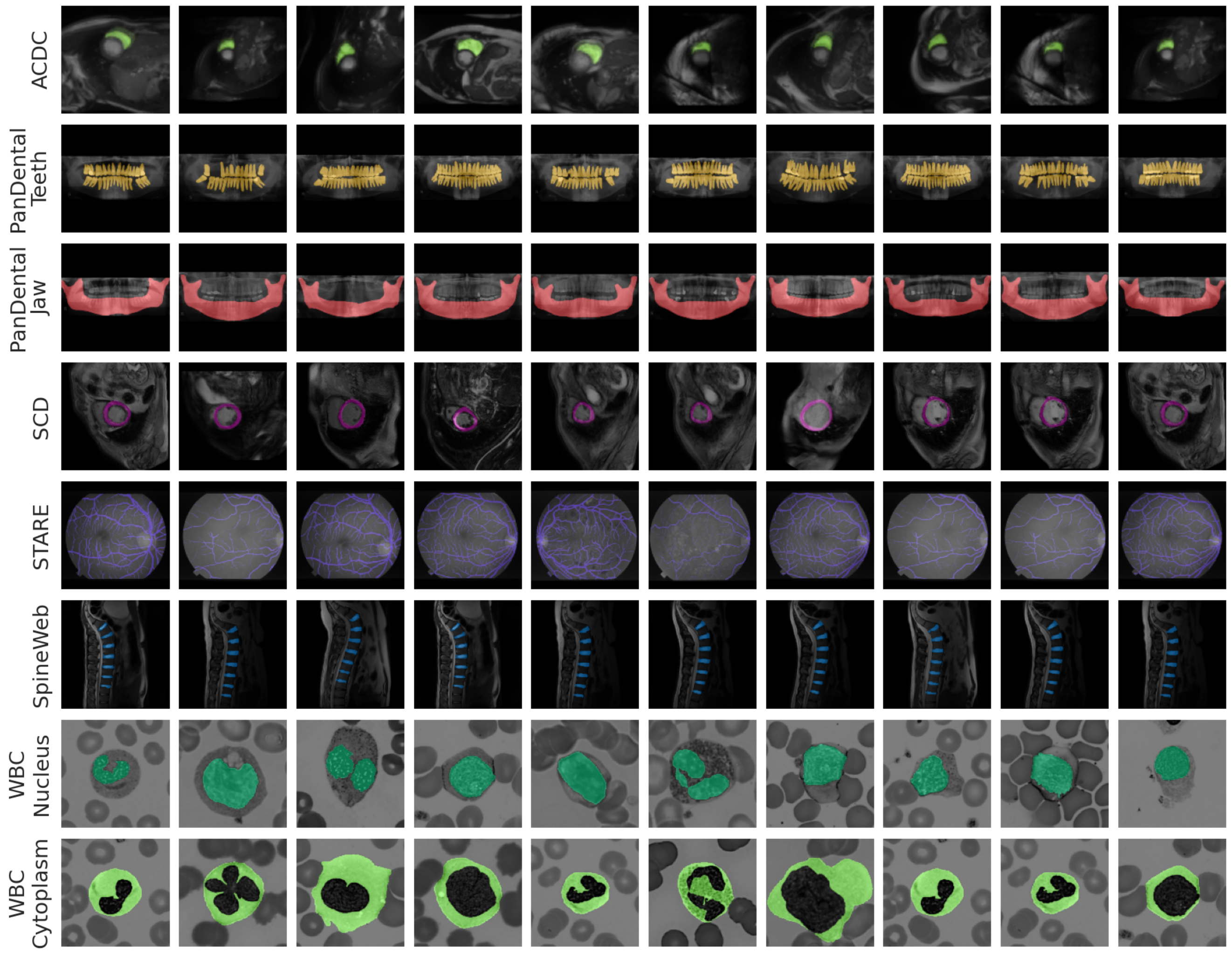


Figure 16: **Example Support Sets for Held-Out Datasets.**

Visualization of Soft Predictions. Thresholding segmentation predictions (at 0.5) provides a binary segmentation and enables computation of well-known metrics such as the (hard) Dice score. However, for certain regions of interest, like thin structures, thresholding can hide network performance. In Figure 17, we show this effect visually. For example, focusing on STARE, we see that UniverSeg networks can capture the thin structures very well, which is lost when thresholding the predictions to create a binary segmentation.

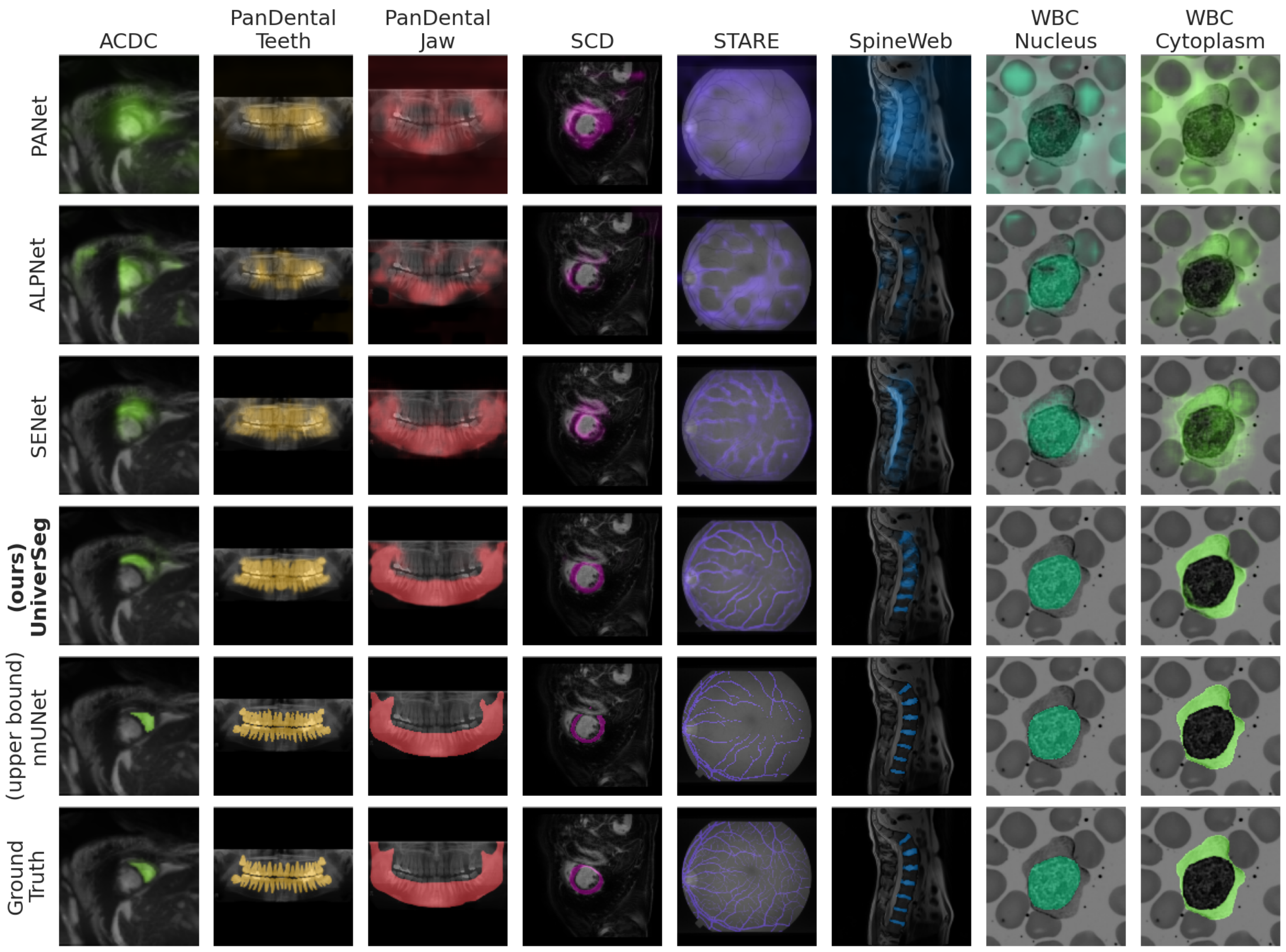


Figure 17: **Visualization of Soft (Non-Thresholded) Predictions for All Methods.**

WBC task visualizations. We include some visualizations of UniverSeg's capability to adapt based on the support set specification. We use the WBC dataset, which presents substantial variability between support set examples.

- Figure 18 presents support set examples for the WBC Cytoplasm label as well as held-out predictions, showing that UniverSeg closely matches the ground truth.
- Figure 19 shows how UniverSeg is equivariant with respect to the support set labels. Given the same images as in Figure 18 but different labels, UniverSeg adapts its predictions to the nucleus label.
- Figure 20 showcases UniverSeg's invariance to image transformations. Using the same images and label examples from Figure 18, we invert the image data (i.e. $1 - x$) for both the query and support set images. UniverSeg correctly segments the label regardless of the image transformation.
- Figure 21 shows that while UniverSeg is trained on binary segmentation tasks, it can adequately perform multi-label segmentation. To produce multi-label predictions, we treat each label independently, and then combine the predictions for each label using a softmax operation.
- Figure 22 shows the effect of the support set size N in the prediction results. We observe that segmentation mask quality substantially improves as we increase the number of support set image-label pairs.
- Figure 23 shows prediction variability for predictions performed with support size $N = 8$ along with an ensembled prediction.

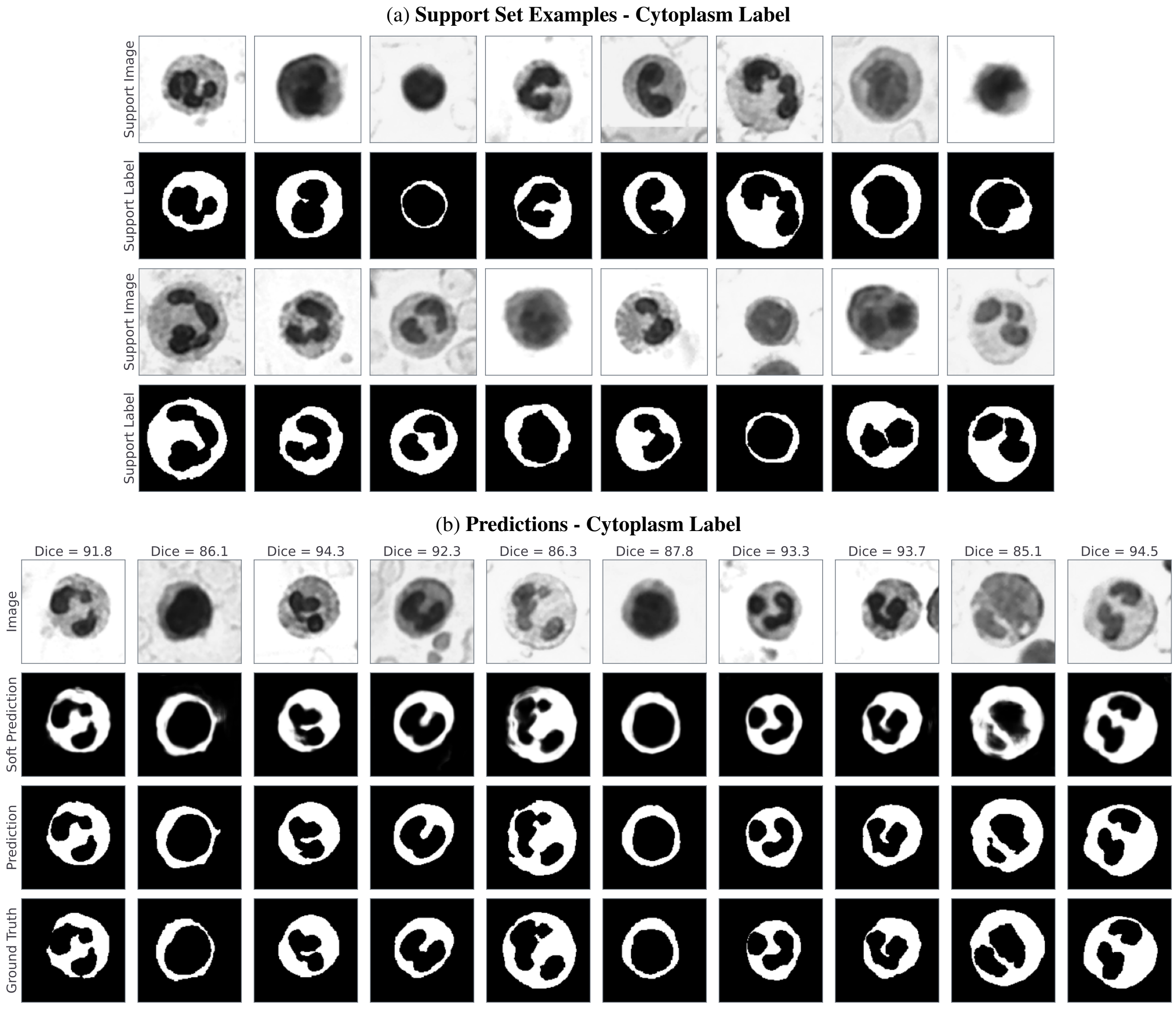
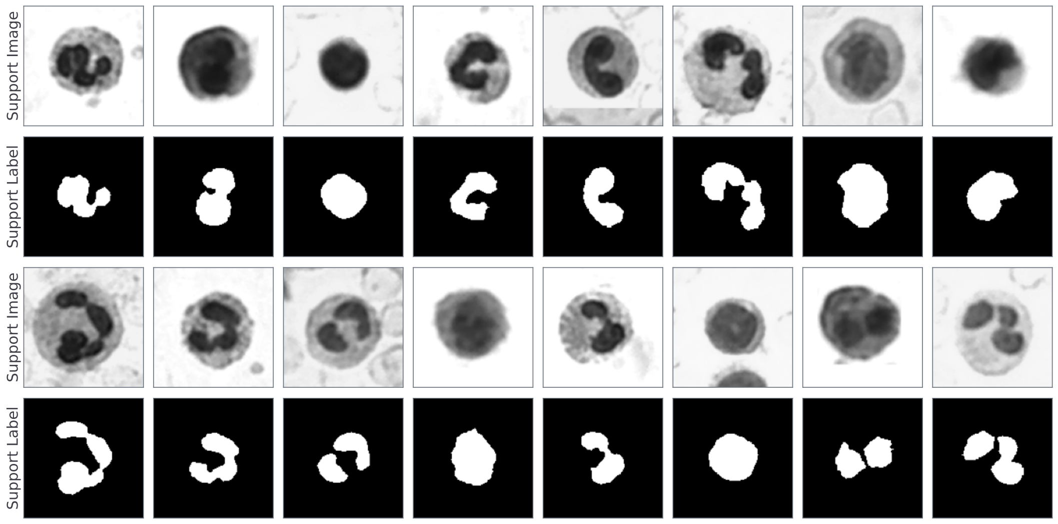


Figure 18: Visualization of support set examples (a) and predictions (b) for the WBC Cytoplasm label

(a) **Support Set Examples - Nucleus Label**



(b) **Predictions - Nucleus Label**

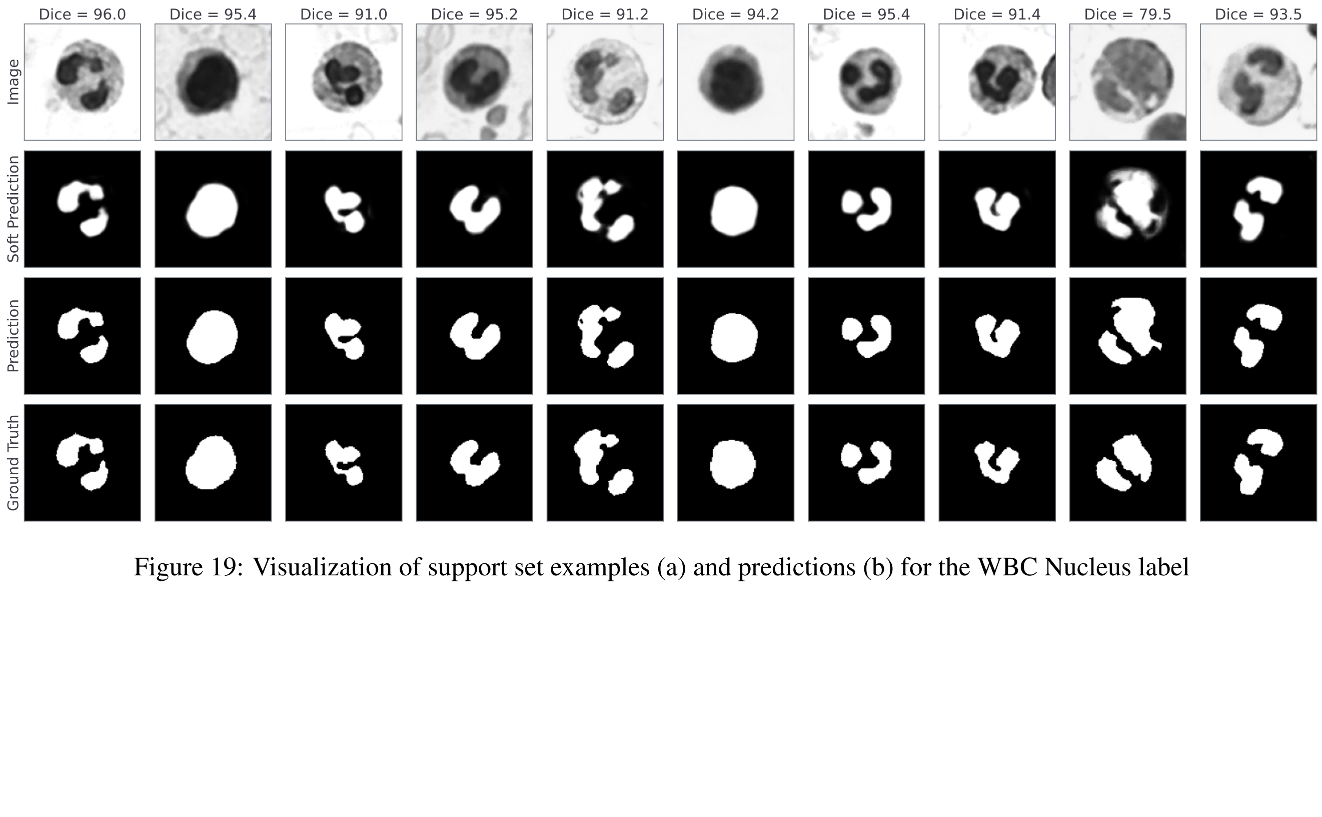
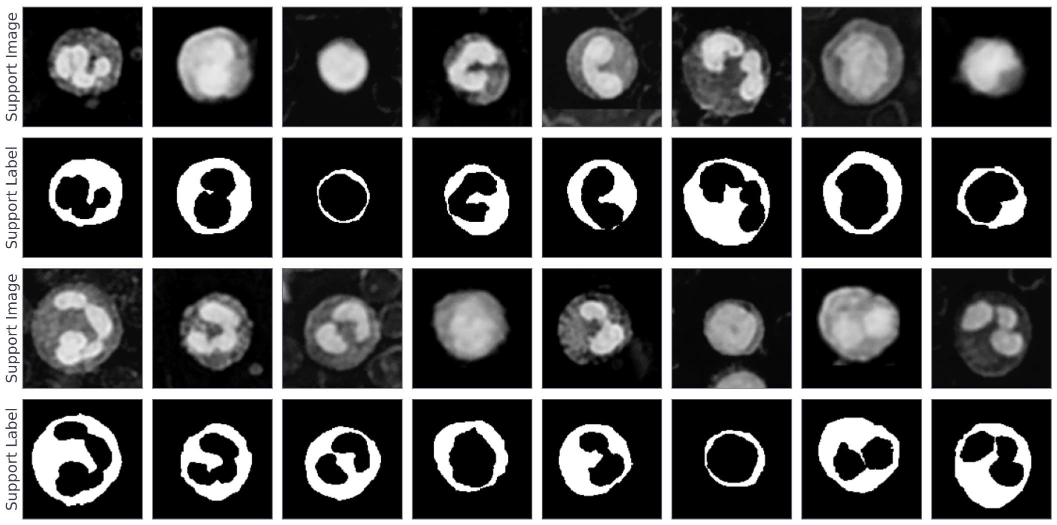


Figure 19: Visualization of support set examples (a) and predictions (b) for the WBC Nucleus label

(a) **Support Set Examples - Inverted Images**



(b) **Predictions - Inverted Images**

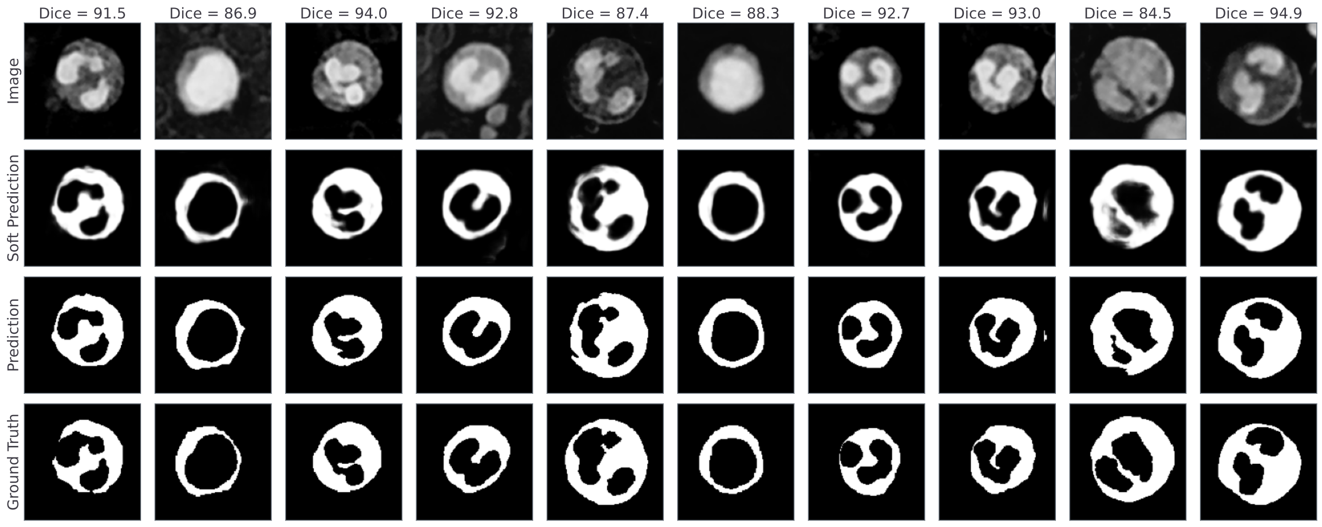
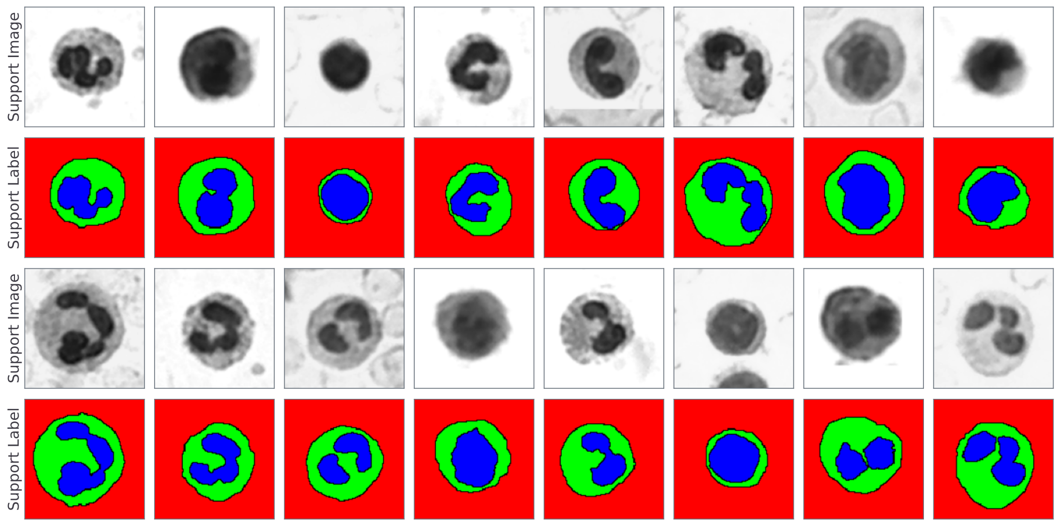


Figure 20: Visualization of support set examples (a) and predictions (b) for the WBC Cytoplasm label with inverted images

(a) **Support Set Examples - Multi Label**



(b) **Predictions - Multi Label**

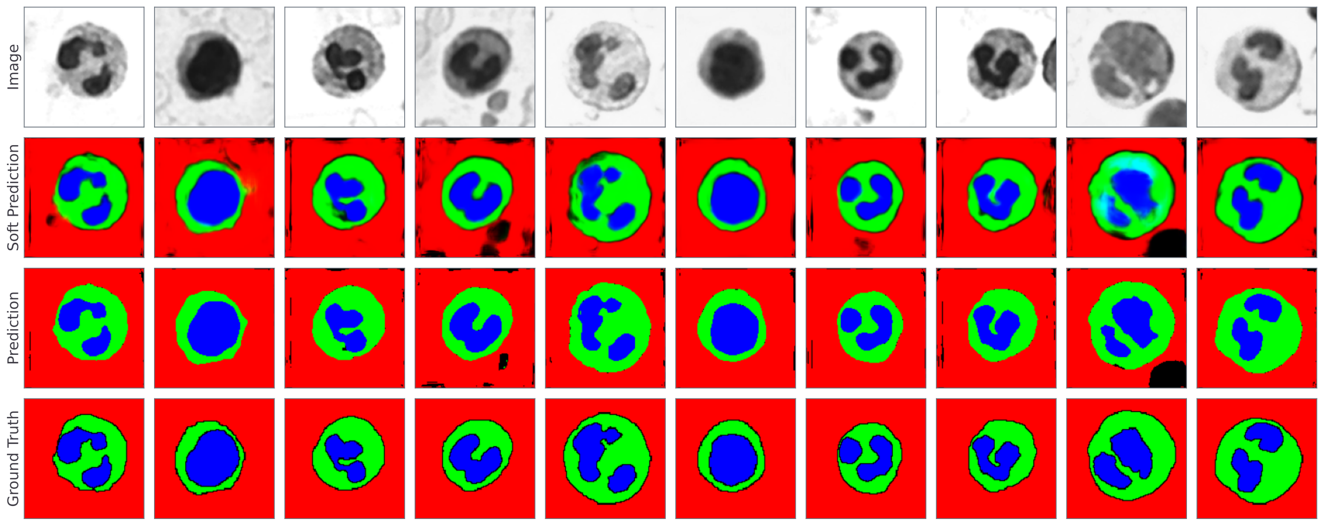


Figure 21: Visualization of support set examples (a) and predictions (b) for the WBC task with multiple labels being predicted independently. Each label is encoded using a RGB channel (Red=backgroud, Green=Cytoplasm, Blue=Nuclues), we only see some mild nucleus-cytoplasm overlaps in cyan for one example.

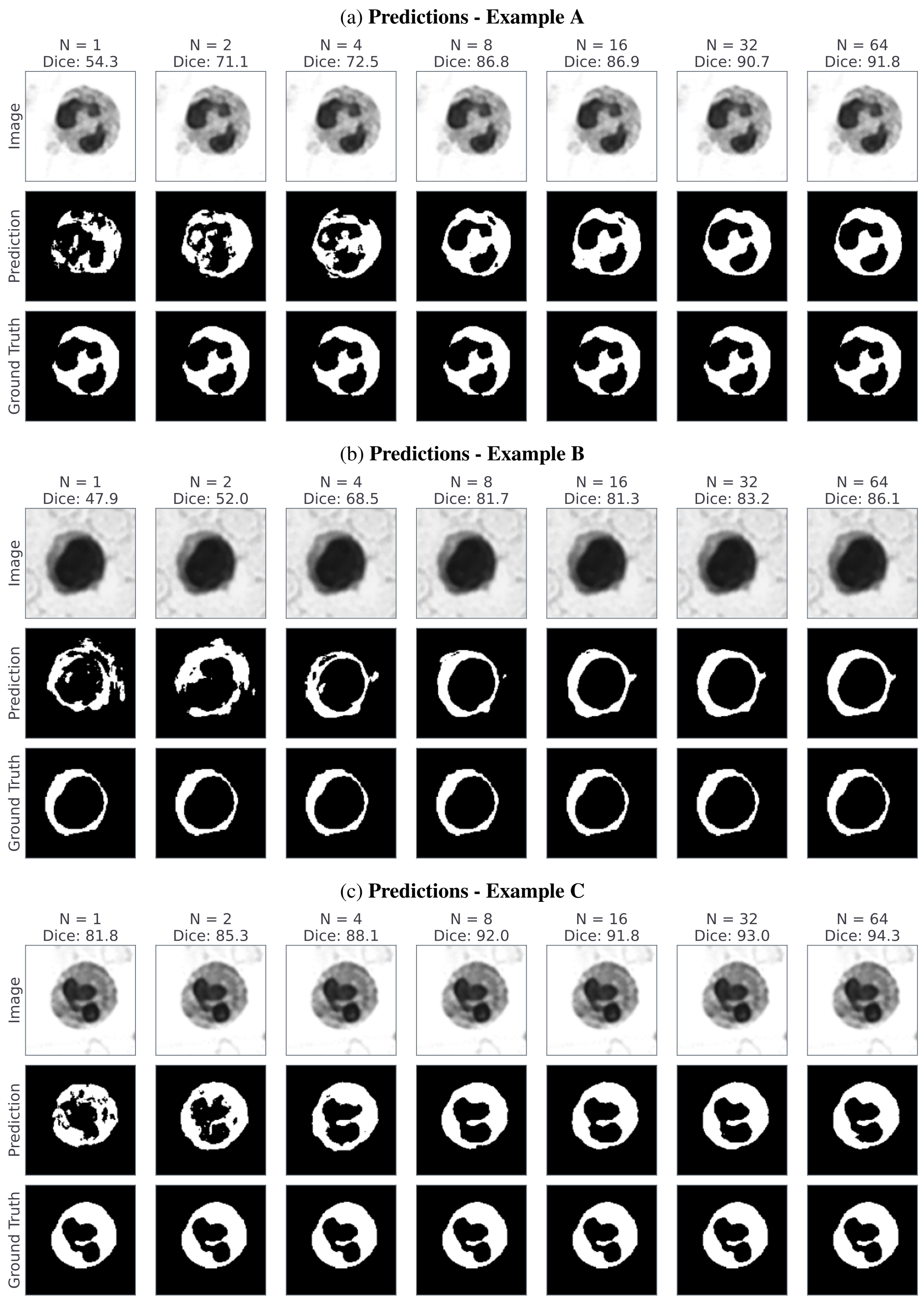
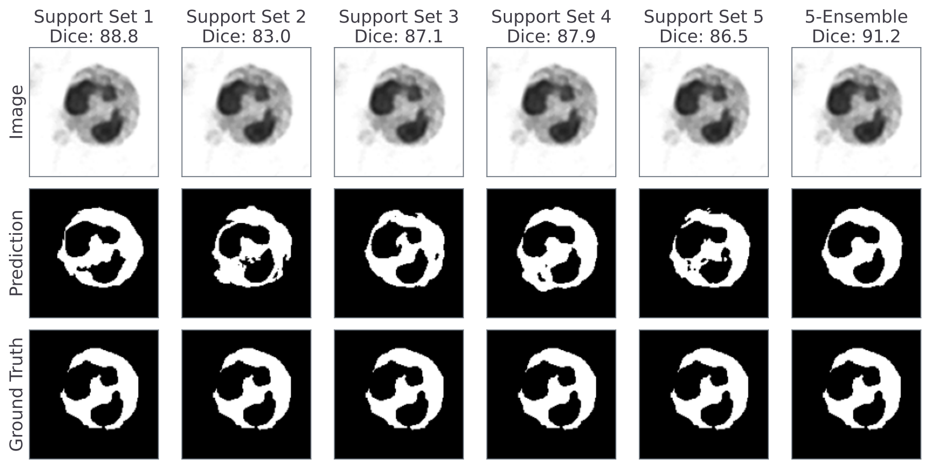
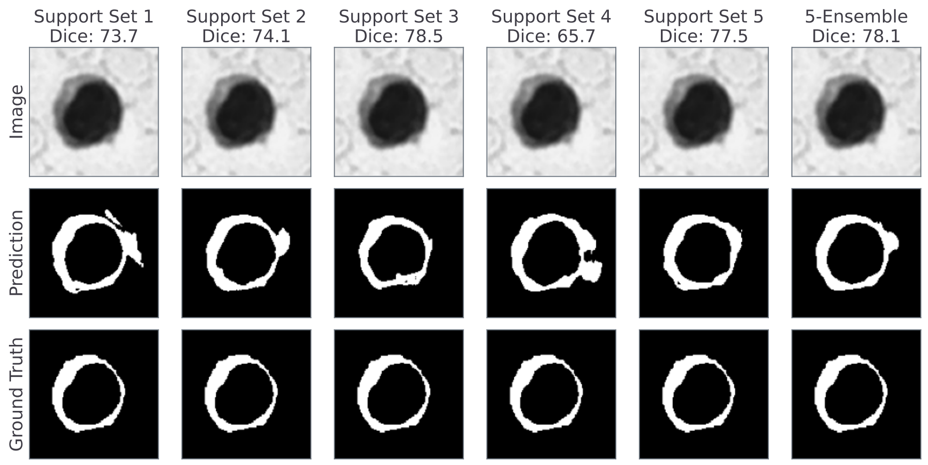


Figure 22: Visualization of predictions for the WBC Cytoplasm task with varying number of support set examples N. Larger support sets lead to better segmentation masks.

(a) **Predictions - Example A**



(b) **Predictions - Example B**



(c) **Predictions - Example C**

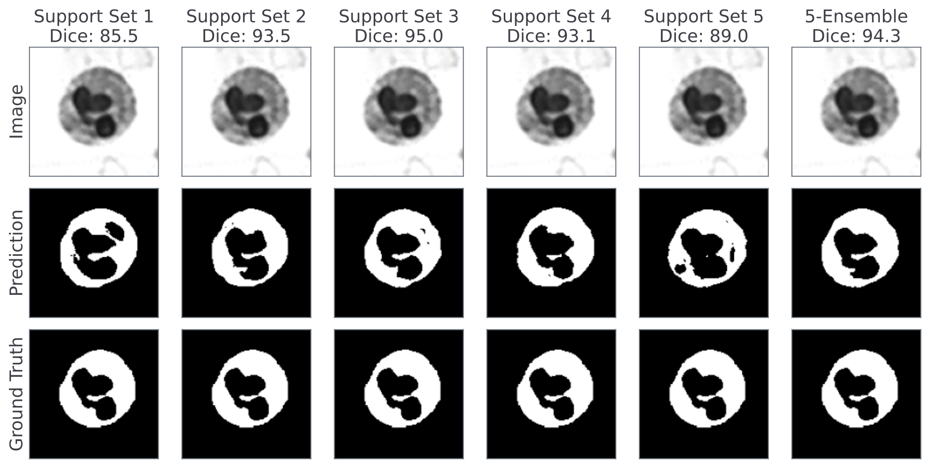


Figure 23: Visualization of predictions for the WBC Cytoplasm task with various choices of support set ($N = 8$) as well as the ensembled prediction (last column). Ensembling reduces the variance of predictions.