A. Appendix

A.1. Network architectures and training parameters

The structure of the U-Net embedding network used for each dataset is described using the buliding block shown in Figure 1. The number of ConvBlocks in the encoder/decoder part of U-Net is chosen such that the receptive field of features in the last encoder layer is equal to or slightly bigger than the input size. We used group normalization [14] for 3D and electron microscopy experiments and batch normalization for CVPPP and Cityscapes datasets [6].

In the tables describing the architecture: second number after the comma corresponds the number of output channels from each layer, *Upsample* denotes nearest-neighbor upsampling and *Concat* denotes channel-wise concatenation of the output for a given decoder layer with the output from the corresponding encoder layer.

Unless otherwise specified, Adam optimizer [7] with an initial learning rate of 0.0002, weight decay 10^{-5}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ was used for training. Learning rate was reduced by a factor of 0.2 when the validation loss stopped improving after a dataset-dependent number of iterations. Training was stopped when the learning rate dropped below 10^{-6} or maximum number of iterations was reached. In all our experiments we use 16-dimensional embedding space, i.e. the output from the U-Net after the last 1×1 convolution has 16 channels. Input images were globally normalized to zero mean and a standard deviation of one unless stated otherwise.

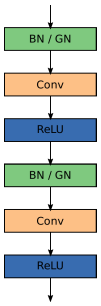

Figure 1. ConvBlock architecture. We use *ConvBlock*, *ConvBlockBN* or *ConvBlockGN* to refer to the block where *no*, *batch* or *group* normalization is used respectively.

CVPPP. Table 1 shows the 2D U-Net architecture used in the experiment. All networks were trained for up to 80K iterations (unless the stopping criteria was not satisfied before) with a minibatch size of 4. Input images were randomly scaled, flipped horizontally and vertically and cropped to 448×448 pixels. Before passing to $f(\cdot)$ and $g(\cdot)$ networks, random color jitter and Gaussian blur were applied.

3-channel image $x \in \mathbb{R}^{M\times N\times 3}$
ConvBlockBN, 16, MaxPool 2×2
ConvBlockBN, 32, MaxPool 2×2
ConvBlockBN, 64, MaxPool 2×2
ConvBlockBN, 128, MaxPool 2×2
ConvBlockBN, 256, MaxPool 2×2
ConvBlockBN, 512, Upsample 2×2
Concat, 256 + 512
ConvBlockBN, 256, Upsample 2×2
Concat, 128 + 256
ConvBlockBN, 128, Upsample 2×2
Concat, 64 + 128
ConvBlockBN, 64, Upsample 2×2
Concat, 32 + 64
ConvBlockBN, 32, Upsample 2×2
Concat, 16 + 32
ConvBlockBN, 16, conv 1×1, d

Table 1. U-Net architecture for CVPPP and Cityscapes datasets. $(M, N) = (448, 448)$, $d = 16$ for CVPPP and $(M, N) = (384, 768)$, $d = 8$ for Cityscapes.

Cityscapes. See Table 1 for an overview of 2D U-Net architecture for the Cityscapes semantic instance segmentation task. All networks were trained for up to 90K iterations with a minibatch size of 16. The network output dimension was set to 8. Input images were randomly cropped to 358×768 patches. Random flipping and scaling (ratio in [0.5, 2.0]), Gaussian blurring, color jitter and random conversion to grayscale was applied to the input before passing it to $f(\cdot)$ and $g(\cdot)$ networks.

Light microscopy datasets. 3D U-Net architecture used for the light microscopy datasets is shown in Table 2. Ovules networks were trained for up to 200K iterations (or until the stopping criteria was satisfied) with a minibatch size of 8. Stem networks were fine-tuned with a fixed, reduced learning rate of 0.00002 for 100K iterations. 3D patches of shape 40×64×64 (*ZYX* axes ordering) were used. Patches were augmented with random rotations, flips and elastic deformations. Gaussian noise was added to the input before passing through $f(\cdot)$ and $g(\cdot)$ networks.
For a fair comparison with other methods we do not stitch the patches to recover the whole volume, but evaluate on the patch-by-patch basis.

Electron microscopy datasets. 2D U-Net architecture for the VNC and MitoEM datasets is shown in Table 3. The source VNC network was trained for up to 100K iterations with a minibatch size of 4. MitoEM networks were

1-channel 3D patch $x \in \mathbb{R}^{K \times M \times M \times 1}$
ConvBlockGN, 64, MaxPool 2×2
ConvBlockGN, 128, MaxPool 2×2
ConvBlockGN, 256, MaxPool 2×2
ConvBlockGN, 512, Upsample 2×2
Concat, 256 + 512
ConvBlockGN, 256, Upsample 2×2
Concat, 128 + 256
ConvBlockGN, 128, Upsample 2×2
Concat, 64 + 128
ConvBlockGN, 64, conv 1×1, $d = 16$

Table 2. U-Net architecture for Ovules and Stem datasets, $K = 40, M = 64$. All convolutions and max pooling operations are 3D.

1-channel image $x \in \mathbb{R}^{M \times M \times 1}$
ConvBlockGN, 16, MaxPool 2×2
ConvBlockGN, 32, MaxPool 2×2
ConvBlockGN, 64, MaxPool 2×2
ConvBlockGN, 128, MaxPool 2×2
ConvBlockGN, 256, MaxPool 2×2
ConvBlockGN, 512, MaxPool 2×2
ConvBlockGN, 1024, Upsample 2×2
Concat, 512 + 1024
ConvBlockGN, 512, Upsample 2×2
Concat, 256 + 512
ConvBlockGN, 256, Upsample 2×2
Concat, 128 + 256
ConvBlockGN, 128, Upsample 2×2
Concat, 64 + 128
ConvBlockGN, 64, Upsample 2×2
Concat, 32 + 64
ConvBlockGN, 32, Upsample 2×2
Concat, 16 + 32
ConvBlockGN, 16, conv 1×1, $d = 16$

Table 3. U-Net architecture for VNC and MitoEM datasets, $M = 448$.

fine-tuned with a fixed, reduced learning rate of 0.00002 for 100K iterations. 2D patches of shape 448×448 were used. Patches were augmented with random rotations, flips and elastic deformations. Gaussian noise was added to the input before passing through $f(\cdot)$ and $g(\cdot)$ networks.

Adversarial training. As mentioned in Sec.3.1 our approach can be used for adversarial training, where the pixel embedding network is a generator of object masks, which the discriminator learns to distinguish from the ground truth object segmentation masks. In this case the embedding network (generator) is trained with the following objective:

$$L_{adv} = L_{SO} + \lambda L_{wgan} \tag{1}$$

with L_{SO} defined in Eq. 5. For adversarial training we use Wasserstein GAN with gradient penalty (WGAN-GP) [4] objective function given by:

$$\begin{aligned} V_D(G, D) = \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g}[D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r}[D(\mathbf{x})] \\ + \lambda \mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\| - 1)^2] \end{aligned} \tag{2}$$

$$L_{wgan} = V_G(G, D) = -\mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g}[D(\tilde{\mathbf{x}})] \tag{3}$$

for the critic and the embedding network (generator) respectively. $\mathbb{P}_r$ is the distribution of ground truth mask, $\mathbb{P}_g$ is the distribution of predicted "soft" masks and $\mathbb{P}_{\hat{\mathbf{x}}}$ is the sampling distribution. Table 4 shows the architecture of the critic used in the Ovules dataset experiments. Our final objective for the embedding netowrk is given by: $L_{SO} + \zeta L_{wgan}$ or $L_{SSO} + \zeta L_{wgan}$ depending on whether full or sparse supervision is used. Training of the embedding network and the critic is done using the Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.9$ and initial learning rate of 0.0001 for both networks. We use $n_{critic} = 5$ iterations per each iteration of the embedding network. We use $\lambda = 10$ (GP weight) and $\zeta = 0.1$ (L_{wgan} weight) in our experiments. In order to prevent uninformative gradients from the critic at the beginning of the training process, L_{wgan} is enabled after the warm-up period of 50K iterations.

In our experiments, adversarial training does not by itself bring a significant performance improvement over the Dice-based loss. Both losses can also be used in combination which can be beneficial: as we show in Table 3, the combined loss outperforms a much more complex 3-step state-of-the-art segmentation pipeline. This finding is similar to [10] where authors use an adversarial approach to train a semantic segmentation model. Our approach differs, as in our setup the discriminator focuses more on the individual object properties instead of the global statistics of the semantic mask predicted by the network.

1-channel image $x \in [0, 1]^{K \times M \times M \times 1}$
ConvBlock, 64, MaxPool 2×2
ConvBlock, 128, MaxPool 2×2
ConvBlock, 256, MaxPool 2×2
ConvBlock, 512, Upsample 2×2
dense layer, 1

Table 4. WGAN-GP critic architecture used in the adversarial setting on Ovules dataset, $K = 40, M = 64$. All convolutions and max pooling operations are 3D. No batch or group normalization was used. ReLU was replaced by leaky ReLU activation function with $\alpha = 10^{-2}$.

A.2. Ablation study of loss functions

In this section, we study the impact of different loss variants and the choice of $g(\cdot)$ network on the final performance of our method. Table 5 presents segmentation and counting scores (CVPPP test set) with different loss variants. HDBSCAN with $min\_size = 200$ and no foreground mask was used for clustering the network outputs in all cases. We see that when only a few ground truth objects are used for training (10%, 40%) the consistency term L_{U_con} (Eq. 7) has a much stronger impact on the final segmentation performance than the unlabeled "push" term L_{U_push} (Eq. 6). The absence of L_{U_push} term worsens the segmentation and counting scores in all experiments.

Loss function	SBD	$\lvert DiC\rvert$
@0.1	0.788 ± 0.017	5.4 ± 0.3
@0.1 w/o L_{U_push}	0.734 ± 0.042	8.5 ± 0.4
@0.1 w/o L_{U_con}	0.720 ± 0.037	6.3 ± 0.1
@0.4	0.824 ± 0.003	3.2 ± 0.5
@0.4 w/o L_{U_push}	0.779 ± 0.045	3.0 ± 0.7
@0.4 w/o L_{U_con}	0.738 ± 0.019	2.1 ± 0.1
@0.8	0.828 ± 0.010	1.6 ± 0.2
@0.8 w/o L_{U_push}	0.797 ± 0.014	1.9 ± 0.4
@0.8 w/o L_{U_con}	0.810 ± 0.010	2.1 ± 0.2

Table 5. Ablation study of different loss variants. We report segmentation (SBD) and counting ($|DiC|$) scores on the CVPPP test set. Ablation of the L_{U_con} L_{U_push} term in the semi-supervised setting is reported for 10%, 40% and 80% of randomly selected ground truth objects. Mean $\pm$ SD are reported across 3 random samplings of the ground truth objects.

To finalize the ablation study, we trained the network in a fully-supervised setting using the consistency regularization from the weakly-supervised setup. Tab 6 shows the comparison between all four variants. Using the consistency term L_{U_con} together with the instance-based term L_{obj} on the Cityscapes validation set gives the highest mAP@0.5 score of 0.459 as compared to 0.387 (discriminative loss [1]), 0.418 ([1] + L_{obj}) and 0.429 ([1] + L_{U_con}). For CVPPP the SBD metric on the validation set improves from 0.847 (full supervision as in [1]) to 0.852 ([1] + L_{obj}), to 0.853 ([1] + L_{obj}) and to 0.849 ([1] + L_{obj} + L_{U_con}).

Loss function	CVPPP	Cityscapes
DL [1]	0.847	0.387
DL + L_{obj}	0.852	0.418
DL + L_{U_con}	**0.853**	0.429
DL + L_{obj} + L_{U_con}	0.849	**0.459**

Table 6. Weakly-supervised regularization in the fully-supervised setting. SBD (CVPPP datasets) and mAP@0.5 (Cityscapes dataset) computed on the validation sets.

A.3. g-network ablations

We experiment with different types of the g network used in the consistency loss to better understand its effect. Apart from the momentum g described in Sec. 3.2 we consider three other variants: (1) weights are *shared* between f and g, i.e. $\theta_g = \theta_f$, (2) g shares the weights with f, but uses spatial *dropout* [11] in the bottleneck layer of the U-Net architecture, (3) g uses independent set of weights *trained* by back-propagation.
For the purpose of this ablation, we split the CVPPP A1 training set into 103 randomly selected images used for training and report the results on the remaining 25 images.

Table 7 shows the segmentation and counting scores for the HDBSCAN-clustered embeddings together with training dynamics for different variants of g. "Dropout" variant shows a good initial convergence rate, but overfits quickly and has the worse final performance. "Trained" and "shared" variants show comparable average scores, however the variance is much larger for the "trained", which is prone to training instabilities. "Momentum" outperforms the others by a large margin and has the fastest convergence speed. Figure 2 shows the PCA-projected netork outputs for two randomly selected CVPPP images (test set) and five different settings. One can see that sparse training without the consistency loss (column 2) fails to separate the background. Using dropout g leads to artifacts in the unlabeled region. "Shared" and "trained" variants of g (columns 4 and 5) provide limited background separation, but fail to produce crisp embeddings. The "momentum" variant (column 6) is able to correctly separate the background.

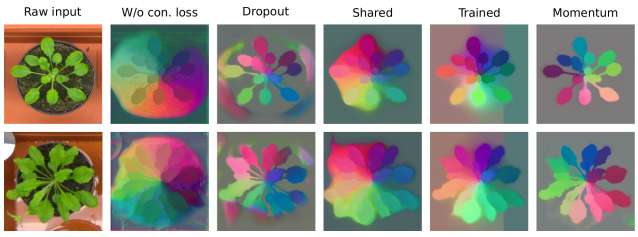

Figure 2. Qualitative comparison of the PCA-projected outputs from the network f for different training setups: (col 2) no consistency term, (col 3) dropout g, (col 4) shared g, (col 5) trained g, (col 6) momentum g. Two images from the CVPPP test set were randomly selected. SPOCO@0.1 was used for training.

In transfer learning setting, the embedding network trained on the source domain is fine-tuned on the target domain with just a few groundtruth objects. Results on the EM data, where VNC dataset [3] is the source and the MitoEM [12] is the target domain (Tab 8) show significant drop in segmentation scores across all experiments if the embedding consistency is removed from the loss.

g-network	SBD	$\lvert DiC \rvert$
Shared	0.602 ± 0.016	6.0 ± 0.6
Dropout	0.507 ± 0.060	7.9 ± 0.9
Trained	0.591 ± 0.131	5.7 ± 2.0
Momentum	$\mathbf{0.649 \pm 0.045}$	$\mathbf{4.5 \pm 1.6}$

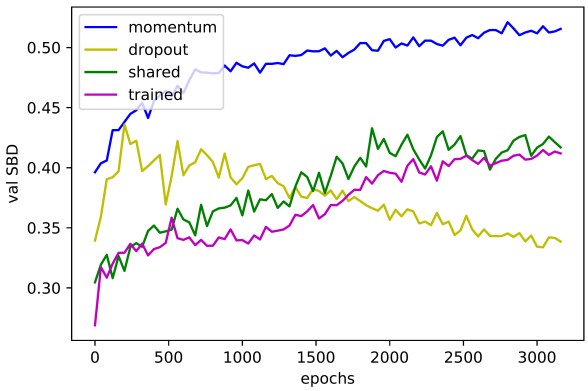

Table 7. (Top) Segmentation performance computed for different types of g-network on the CVPPP validation set. (Bottom) Comparison between g-network variants during training. SPOCO@0.1 used for training, mean $\pm$ SD across 3 random samplings of the groundtruth objects is shown.

Method	AP@0.5	mAP
@0.01	0.368 ± 0.022	0.247 ± 0.022
@0.01 w/o L_{U_con}	0.306 ± 0.014	0.210 ± 0.008
@0.05	0.398 ± 0.007	0.277 ± 0.006
@0.05 w/o L_{U_con}	0.319 ± 0.002	0.227 ± 0.002
@0.10	0.389 ± 0.013	0.268 ± 0.007
@0.10 w/o L_{U_con}	0.301 ± 0.012	0.212 ± 0.007

Table 8. Ablation of the consistency term L_{U_con} in the transfer learning setting with 1%, 5%, 10% of groundtruth objects (target domain). Average precision measured on the target task of MitoEM mitochondria segmentation is reported. VNC dataset serves as a source domain. Mean $\pm$ SD are reported across 3 random samplings of the instances from the target dataset.

A.4. Comparison of clustering algorithms

Apart from the standard mean-shift and HDBSCAN, we introduce two additional clustering methods: (1) a hybrid scheme called *consistency clustering* and (2) a fast affinity graph partitioning. Consistency clustering (Algorithm 1) works by passing two augmented versions of the input through the networks f and g, producing embeddings $\mathcal{E}_f$ and $\mathcal{E}_g$ respectively. We cluster $\mathcal{E}_f$ using mean-shift with bandwidth set to the pull force margin δ_v. Then for each segmented object S_k, we randomly select M anchor points and for each anchor we extract a new object $\hat{S}_k^m$ by taking a δ_v-neighborhood around the anchor in the $\mathcal{E}_g$ space. If the median intersection-over-union (IoU) between S_k and each of the $\hat{S}_k^m$ objects is lower than a predefined threshold, we discard S_k from the final segmentation. This is based on the premise that clusters corresponding to the real objects should remain consistent between $\mathcal{E}_f$ and $\mathcal{E}_g$.

Algorithm 1: Consistency clustering

Input: Set of mean-shift segmented objects $\mathcal{S}$, embeddings from the g-network $\mathcal{E}_g = \{\boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_N\}$, IoU threshold t_{IoU}, number of anchors per object to sample M

Output: New set of segmented objects $\hat{\mathcal{S}}$

$\hat{\mathcal{S}} = \{\}$;

for $S_k \in \mathcal{S}$ **do**

  $\mathcal{A}_k = \{\boldsymbol{a}_k^1, ..., \boldsymbol{a}_k^M \mid \boldsymbol{a}_k^m \in \mathcal{E}_g\}$ - anchors of S_k;

  $I_{IoU} = \{\}$;

  for $\boldsymbol{a}_k^m \in \mathcal{A}_k$ **do**

    $\hat{S}_k^m = \{s_i \mid s_i = \|\boldsymbol{e}_i - \boldsymbol{a}_k^m\| < \delta_v\}$;

    $I_{IoU} \cup \text{IoU}(\hat{S}_k^m, S_k)$;

  if $med(I_{IoU}) > t_{IoU}$ **then**

    $\hat{\mathcal{S}} = \hat{\mathcal{S}} \cup \{S_k\}$;

return $\hat{S}$;

The affinity graph-based method proceeds similar to [8]: we convert the embedding space into a graph partitioning problem by introducing a grid-graph that contains a node for each pixel and connects all direct neighbor pixel via edges. Following [9] and [13] we introduce additional long-range edges that connect pixels that are not direct neighbors in a fixed offset pattern. Following [8] we derive the edge weight w_{ij}, or affinity, between pixel i and j from the embedding vector $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ via

$$w_{ij} = 1 - \max\left(\frac{2\delta_d - \|\boldsymbol{e}_i - \boldsymbol{e}_j\|}{2\delta_d}, 0\right)^2. \tag{4}$$

Here, δ_d is the hinge from Eq. 2 and we use the L2 norm to measure the distance in the embedding space. This weight is derived from the distance term (Eq. 2) and is maximally attractive (0) when the embedding distance is zero and becomes maximally repulsive (1) for embedding distances larger than $2\delta_d$. We obtain an instance segmentation with the Mutex Watershed algorithm [13], which operates on long-range affinity graphs. We introduce long-range edges between all pixel pairs with distance 3, 9 and 27 across all dimensions. This choice yields good segmentation results empirically; potentially we could obtain even better results with this approach by determining the offset pattern via grid search.

Quantitative comparison of 4 different clustering algorithms: HDBSCAN ($min\_size = 200$), Mean-shift (with bandwidth set to $\delta_v = 0.5$), Consistency Clustering ($t_{IoU} = 0.6$) and affinity-based clustering are shown in Table 9. We report the segmentation and counting scores as well as runtimes on the CVPPP validation set. We used the embedding

networks trained using SPOCO@0.1 (i.e. 10% of randomly selected ground truth objects) and SPOCO (trained with full supervision).

Mean-shift has a high recall (correctly recovers most instances), but low precision (it tends to over-segment the image around the boundary of the objects, see Fig 3) resulting in high number of false positives and inferior counting scores. Consistency clustering significantly improves the initial mean-shift segmentation resulting in the best segmentation metric for the network trained with weak supervision (SPOCO@0.1). Affinity-based (Mutex Watershed) and density-based (HDBSCAN) methods have similar segmentation scores, with the former achieving much better counting performance in both full (SPOCO) and weak (SPOCO@0.1) supervision. The affinity-based approach has much lower runtimes compared to the other clustering methods.

Method (CVPPP)	SBD	$\lvert DiC \rvert$	t [s]
SPOCO@0.1			
Consistency	0.729 ± 0.086	2.7 ± 1.7	252.3
HDBSCAN	0.653 ± 0.077	5.7 ± 1.7	82.3
Mean-shift	0.356 ± 0.048	20.7 ± 6.2	201.2
Affinity-based	0.615 ± 0.061	2.6 ± 2.3	0.45
SPOCO			
HDBSCAN	0.834	1.6	164.7
Mean-shift	0.541	10.92	121.9
Affinity-based	0.833	0.88	0.4

Table 9. Performance and runtime comparison of the clustering methods on the CVPPP validation set. We compare the results for SPOCO@0.1, where mean $\pm$ SD are reported across 3 random samplings of the ground truth objects as well as fully supervised SPOCO (bottom), for which we report results from a single training.

Method (Ovules)	Arand error	t [s]
HDBSCAN	0.133	95.036
Mean-shift	0.102	279.202
Affinity-based	0.086	0.955

Table 10. Performance (Adapted Rand Error) and runtime comparison of the clustering methods on the ovules test set. Embedding network trained with fully supervised SPOCO.

Similarly, Table 10 shows comparison of 3 clustering algorithms: HDBSCAN ($min\_size = 600$), Mean-shift ($bandwidth = \delta_v$) and affinity-based on the Ovules test set. We skip the consistency clustering, since the embedding network is trained in the full supervision setting. We notice that for the dense tissue segmentation problems, HDBSCAN classifies the low density areas between the cells as noise, and additional post-processing is required in order to fill the empty space. Results reported in Tab 3 and Fig 5 are based on the watershed post-processing. Here, for fair comparison with other methods we don't use the watershed post-processing on the HDBSCAN clustering results (see Fig 5 bottom). Overall, the parameter-free, affinity-based clustering is much faster (3 orders of magnitude faster) than other methods under consideration and provides the best performance-runtime ratio. The downside of HDBSCAN is its sensitivity to the *min_size* hyperparameter, longer running times and the need for additional post-processing for dense tissue segmentation problems.

Qualitative results on samples from the CVPPP and Ovules datasets are illustrated in Figure 3.

A.5. Training with limited annotation budget

Choosing a fixed annotation budget of N ground truth instances we can objectively compare the weakly supervised training with the dense, fully supervised one. We set $N = 16$, which corresponds to roughly 1% of the objects from the CVPPP training set containing 1683 objects spread across 103 files (we use train/val script described in Sec A.3). In the *dense* setup we randomly choose a single groundtruth file with 16 objects and dense labeling (including the background label), whereas in the *sparse* setting we randomly sample 16 objects from the whole training set, resulting in 16 files, each with only one object labeled. We train fully supervised SPOCO using the densely labeled image and weakly supervised SPOCO using the sparsely labeled images.

Segmentation metrics and embeddings emerging the two training schemes are shown in Tab 11. The network trained from dense annotations is prone to over-fitting and result in visible artifacts in the embedding space. On the other hand, exposing the network to a much more varied training set in the sparse setting and the presence of a strong consistency regularizer results in a feature space of much better quality. Quantitative comparison confirms that the *sparse* significantly outperforms the *dense* setting in terms of segmentation and counting scores.

A.6. Momentum coefficient (m) exploration

In this experiment, we explore the effect of the momentum coefficient m used in the momentum update of the g-network parameters (see Sec. 3.2). We use the train/val split of the CVPPP training set described in Sec. A.3. Similar to [5] we show in Table 12 that the large momentum ($m = 0.999$) performs best. We hypothesize that using slowly moving g moving acts as a strong regularizer which prevents the embedding network f to adapt too quickly to the spare ground truth signal.

A.7. Kernel threshold (t) exploration

Figure 4 illustrates the effect of the kernel threshold parameter t (Eq. 3 in Sec. 3.1) on the SPOCO model perfor-

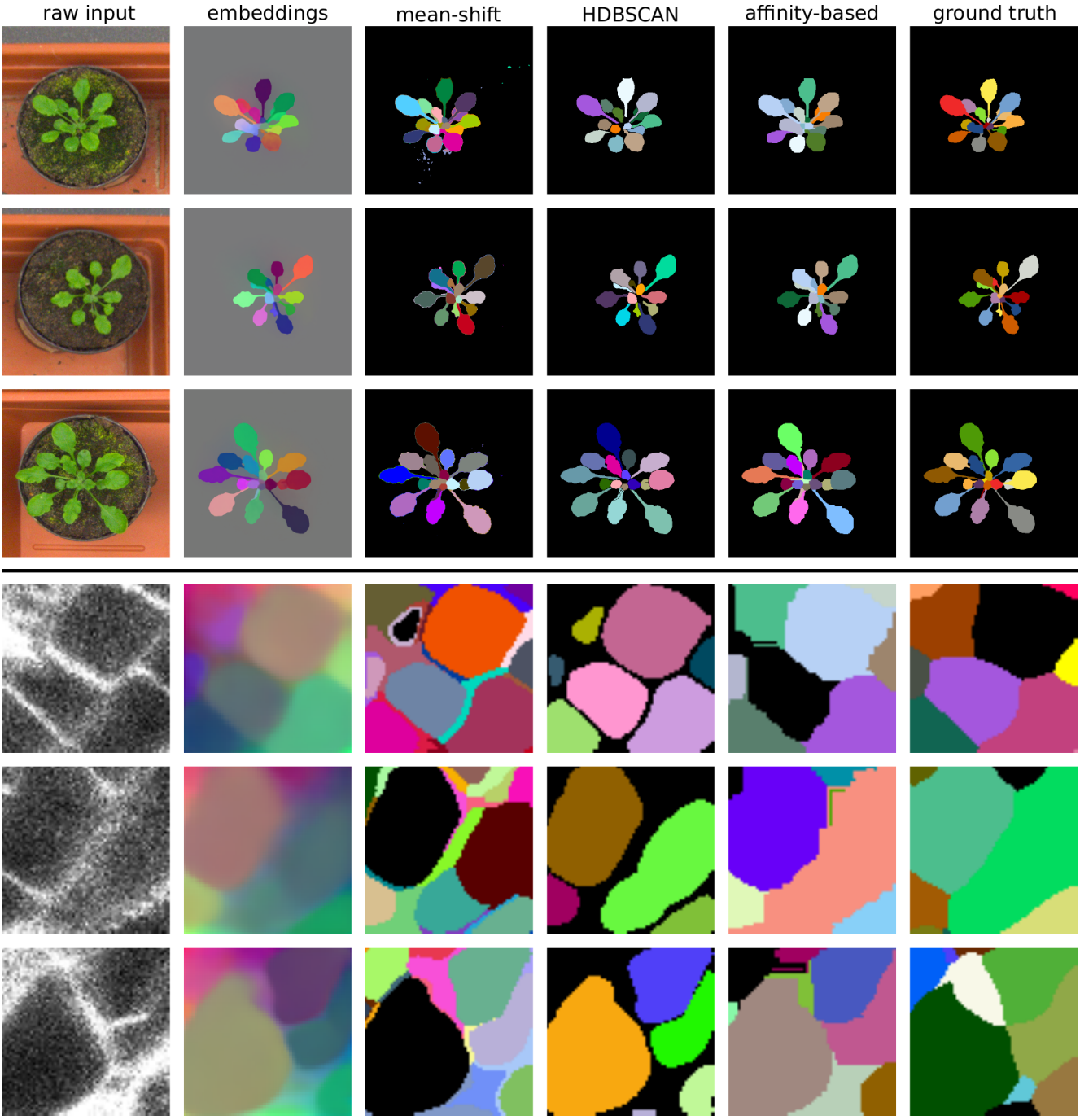

Figure 3. Qualitative comparison of different clustering schemes on the samples from the (top) CVPPP validation set and (bottom) Ovules test set. Fully supervised SPOCO was used to train embeddings.

mance. Choosing a large value (e.g. $t = 0.9$) leads to a crisper, more separable embeddings than smaller values (e.g. $t \in \{0.25, 0.5, 0.75\}$). The difference in the final segmentation performance between a small and a large value of t is especially apparent in the sparse annotation regime. Indeed, when training with only 10% (SPOCO@0.1) or 40% (SPOCO@0.4) of ground truth objects, the mean SBD score improvement between $t = 0.5$ and $t = 0.9$ is 0.044 and 0.054 respectively. Although the performance gain is less pronounced when more supervision is provided (for SPOCO@0.8 the mean SBD reaches a plateau for $t \geq 0.5$), models trained with higher values of t are more robust as

Training scheme	SBD	$\lvert DiC \rvert$
1% dense	0.380	9.8
1% sparse	**0.691**	**2.2**

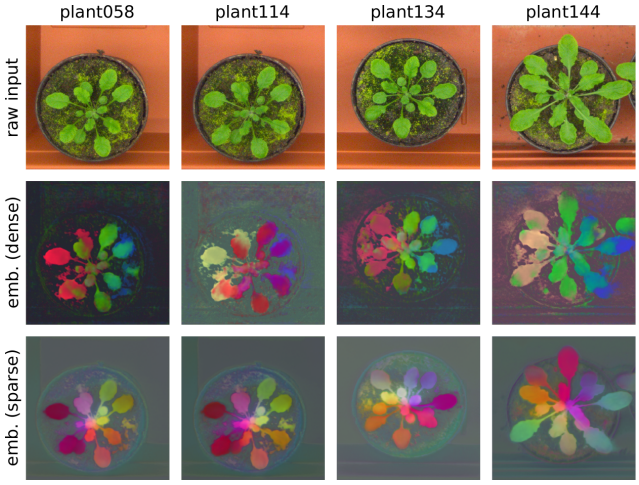

Table 11. (Top) Segmentation performance computed for the networks trained with limited annotation budget on the CVPPP validation set. Embeddings within the foreground semantic mask were clustered with mean-shift algorithm. (Bottom) Qualitative comparison of the embeddings trained in the *dense* and *sparse* setting. Four sample images where chosen from the CVPPP validation set.

Momentum coefficient	SBD	$\lvert DiC \rvert$
0.99	0.615 ± 0.042	5.2 ± 0.8
0.995	0.622 ± 0.116	5.4 ± 0.7
0.999	$\mathbf{0.649 \pm 0.045}$	$\mathbf{4.5 \pm 1.6}$

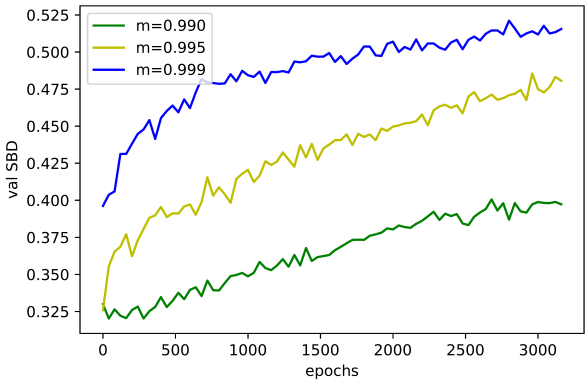

Table 12. The effect of the momentum coefficient value m on the SPOCO performance. (**top**) Segmentation and counting scores. (**bottom**) Evolution of the validation score during training. SPOCO@01 was used for training. Mean $\pm$ SD across 3 random samplings of the ground truth objects is shown.

shown by the low variance of the SBD score.
In our experiments, values of t greater than 0.95 lead to training instabilities.

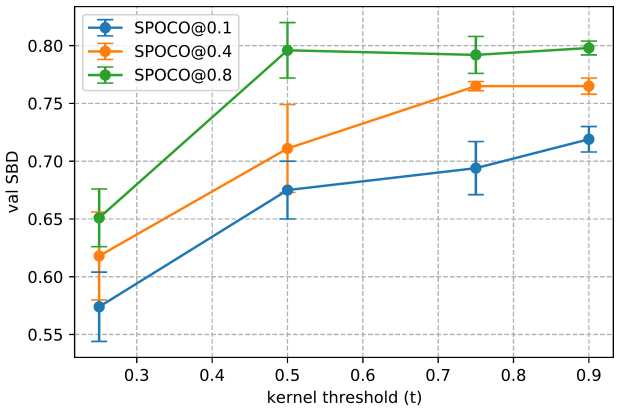

Figure 4. Effect of the kernel threshold t on the segmentation performance at different ground truth objects sampling rates $(0.1, 0.4, 0.8)$. SBD scores measured on the CVPPP validation set are shown for models trained with four values of t: $0.25, 0.5, 0.75, 0.9$. Mean $\pm$ SD are reported across 3 training runs for each (sampling rate, kernel threshold) pair. HDBSCAN ($min\_size = 200$) is used for clustering.

A.8. Cityscapes results

In Tab 13 we compare two different training setups at different object sampling ratios for the Cityscapes dataset: (1) *single-class* reported in the main text, where embedding network is trained separately on each semantic class and (2) *class-agnostic* where all objects from all classes are used to train a single embedding network. The class-agnostic training consistently outperforms the 'single-class' setup for instances of 'rider', 'car', 'motorcycle' and 'bicycle' categories at all sampling levels. The 'single-class' training is better for objects from 'truck', 'bus, and 'train' categories. We hypothesize that the class-agnostic setup learns better representation of objects from correlated classes (e.g. 'person' and 'rider', 'motorcycle' and 'bicycle'), but it is detrimental to trucks, buses and trains due to heavy class imbalance. A per-class weighting of the instance-based term could be beneficial in the class agnostic setting, which we leave for future work.

Additional qualitative results for the weakly-supervised network (SPOCO@0.4) on the Cityscapes validation set can be found in Fig 5. Apart from the segmentation results, we also show the embeddings leaned by the network trained on objects from a given semantic class. For underrepresented classes such as 'motorcycle', 'train', 'truck' the final segmentation strongly relies on the segmentation mask given by the pre-trained semantic segmentation model (DeepLabV3 [2]).

References

[1] B. D. Brabandere, D. Neven, and L. V. Gool. Semantic instance segmentation with a discriminative loss function,

Method	person	rider	car	truck	bus	train	motorcycle	bicycle	**average**
single-class@0.1	0.190	0.360	0.236	**0.438**	**0.481**	**0.490**	0.424	0.204	**0.353**
class-agnostic@0.1	**0.197**	**0.430**	**0.282**	0.243	0.276	0.167	**0.468**	**0.261**	0.291
single-class@0.4	**0.230**	0.396	0.301	**0.558**	**0.601**	**0.594**	0.405	0.214	**0.412**
class-agnostic@0.4	0.207	**0.459**	**0.332**	0.260	0.336	0.223	**0.471**	**0.266**	0.319
single-class@1.0	**0.260**	0.451	0.331	**0.604**	**0.637**	**0.656**	0.464	0.266	**0.459**
class-agnostic@1.0	0.259	**0.463**	**0.410**	0.370	0.395	0.378	**0.478**	**0.296**	0.381

Table 13. Comparison of SPOCO trained in a single-class vs class-agnostic settings at different sampling ratios. Shown are mAP@0.5 scores computed on the Cityscapes validation set.

2017.

[2] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[3] S. Gerhard, J. Funke, J. Martel, A. Cardona, and R. Fetter. Segmented anisotropic sstem dataset of neural tissue, 2013.

[4] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[5] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[8] K. Lee, R. Lu, K. Luther, and H. S. Seung. Learning and segmenting dense voxel embeddings for 3d neuron reconstruction. *IEEE Transactions on Medical Imaging*, pages 1–1, 2021.

[9] K. Lee, J. Zung, P. Li, V. Jain, and H. S. Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017.

[10] P. Luc, C. Couprie, S. Chintala, and J. Verbeek. Semantic segmentation using adversarial networks, 2016.

[11] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[12] D. Wei, Z. Lin, D. Barranco, N. Wendt, X. Liu, W. Yin, X. Huang, A. Gupta, W. Jang, X. Wang, I. Arganda-Carreras, J. Lichtman, and H. Pfister. Mitoem dataset: Large-scale 3d mitochondria instance segmentation from em images. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2020.

[13] S. Wolf, C. Pape, A. Bailoni, N. Rahaman, A. Kreshuk, U. Köthe, and F. A. Hamprecht. The mutex watershed: Efficient, parameter-free image partitioning. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV 2018*, pages 571–587, Cham, 2018. Springer International Publishing.

[14] Y. Wu and K. He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

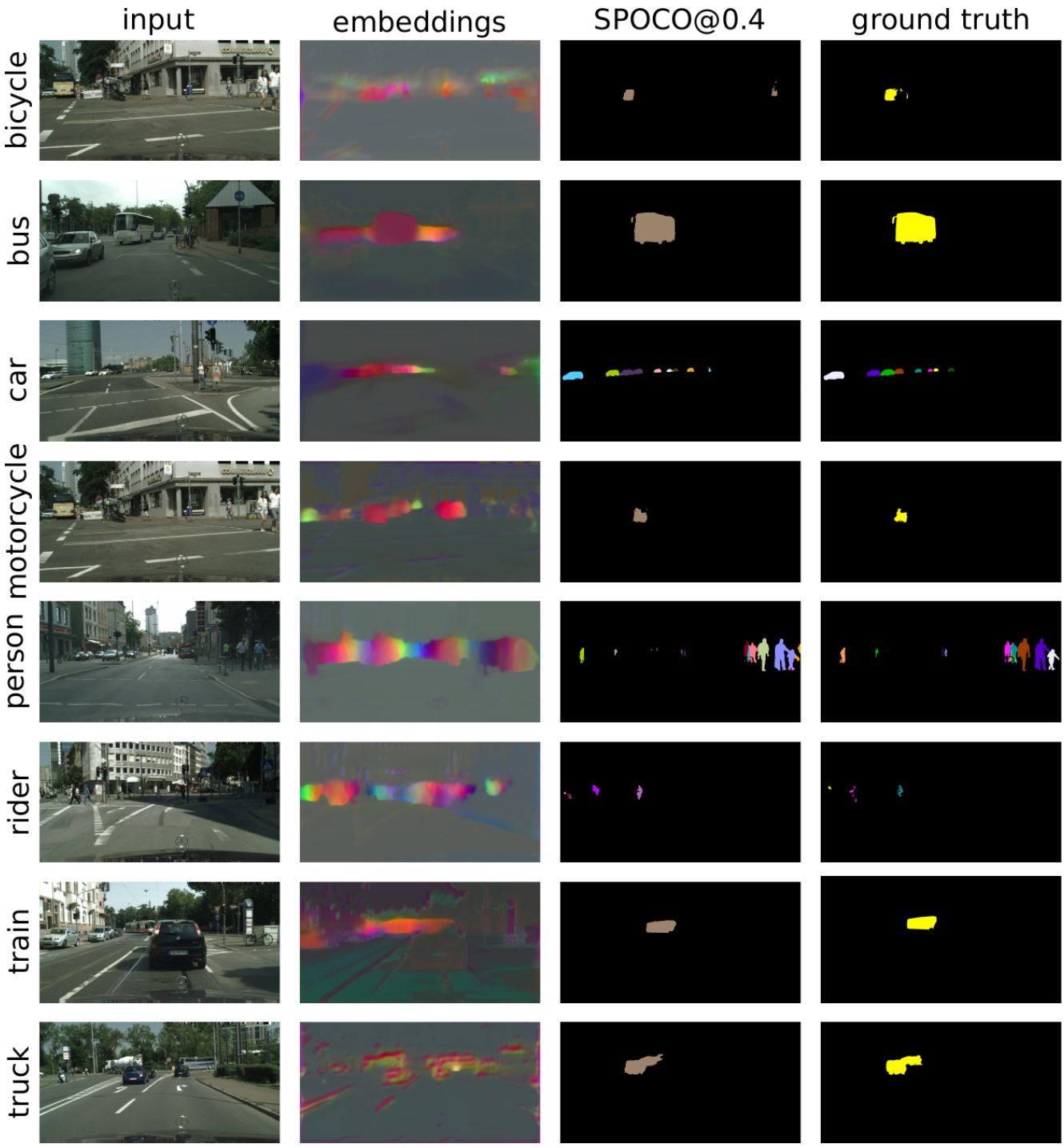

Figure 5. Qualitative results for different semantic classes on the Cityscapes validation set.