Improving Pixel Embedding Learning through Intermediate Distance Regression Supervision for Instance Segmentation

Yuli Wu, Long Chen, and Dorit Merhof

Institute of Imaging and Computer Vision, RWTH Aachen, Germany
yuli.wu@rwth-aachen.de, {long.chen, dorit.merhof}@lfb.rwth-aachen.de

Abstract. As a proposal-free approach, instance segmentation through pixel embedding learning and clustering is gaining more emphasis. Compared with bounding box refinement approaches, such as Mask R-CNN, it has potential advantages in handling complex shapes and dense objects. In this work, we propose a simple, yet highly effective, architecture for object-aware embedding learning. A distance regression module is incorporated into our architecture to generate seeds for fast clustering. At the same time, we show that the features learned by the distance regression module are able to promote the accuracy of learned object-aware embeddings significantly. By simply concatenating features of the distance regression module to the images as inputs of the embedding module, the mSBD scores on the CVPPP Leaf Segmentation Challenge can be further improved by more than 8% compared to the identical set-up without concatenation, yielding the best overall result amongst the leaderboard at CodaLab.

Keywords: Instance Segmentation, Pixel Embedding, Distance Regression

1 Introduction

Instance segmentation aims to label each individual object, which is critical to many biological and medical applications, such as plant phenotyping and cell quantification. Learning object-aware pixel embeddings is one of the trends in the field of instance segmentation. The embedding is essentially a high-dimensional representation of each pixel. To achieve instance segmentation, pixel embeddings of the same object should be located relatively close in the learned embedding space, while those of different objects should be discriminable.

The loss usually consists of two terms: the between-instance loss term $\mathcal{L}_{inter}$ and the within-instance loss term $\mathcal{L}_{intra}$. The former term $\mathcal{L}_{inter}$ encourages different-instance embeddings to be located far away from each other, while the latter term $\mathcal{L}_{intra}$ encourages same-instance embeddings to stay together. Two most popular metrics used to describe the similarities of embeddings are Euclidean distance and cosine distance. Although the pixel embedding approaches

have gained success in many datasets including CVPPP Leaf Segmentation Challenge [4, 5, 12, 16], the trained embedding space is far from optimal.

Our idea was indirectly inspired by the "easy task first" concept behind curriculum learning [1]. Distance regression predicts the distance from a pixel to the object boundary and is used in [4, 20], for example, as an auxiliary module. We have empirically found that the distance regression module is relatively easy to train on many datasets. Considering that the learned features by the distance regression module should be already recognizable for distinguishing instances, we prefix the embedding module with a distance regression module to promote the embedding learning process.

The main contributions of this paper are summarized as follows:

1. We propose an architecture to promote the pixel embedding learning by utilizing features learned from the distance regression module, which significantly improves the performance in the CVPPP Leaf Segmentation Challenge [19]. Our overall mean Symmetric Best Dice (mSBD) score is at the top position of the leaderboard with 0.879 by paper submission. Furthermore, the average of mSBD scores on Arabidopsis images (testing sets A1, A2, A4) outperforms the second best results from three different teams by over 3%, namely from 0.883 to 0.917;
2. We conduct a number of ablation experiments in terms of the stacked U-Net architecture, different types of concatenative layers and varied loss formats, to validate our architecture and also supplement some experimental vacancies in this field.

2 Related Work

We roughly categorize approaches of instance segmentation into two groups with respect to the overall pipeline: *instance-first* approaches and *one-stage* approaches. *Instance-first* approaches exploit the instance-level bounding boxes from the first-stage object detector. For example, Mask R-CNN [7] uses RPN [18], and recent methods like BlenderMask [3] and CenterMask [10] are based on the anchor-free detector FCOS [21]. Pixel-level segmentations are then produced through subjoined refinement modules. Mask R-CNN [7] constructs a lightweight segmentation network with consecutive convolutional layers, while the Blender Module and Spatial Attention-Guided Mask (SAG-Mask) are proposed in [3] and [10], respectively, for a more accurate segmentation.

In contrast, *one-stage* approaches predict the existence (*object-ness*) and mask of objects all at once. Masks are represented as polar coordinates in [20, 25]. Specifically, the model regresses the distances to the boundary along a set of fixed directions at each location. To describe more complex shapes, masks are encoded with a linear projection in [27].

Furthermore, the approaches based on pixel embedding learning, which also belong to *one-stage* approaches, are becoming a new trend. They share the general pipeline of *embedding and clustering*. Each pixel of input images is mapped to a high-dimensional vector (embedding), in which pixels of the same object

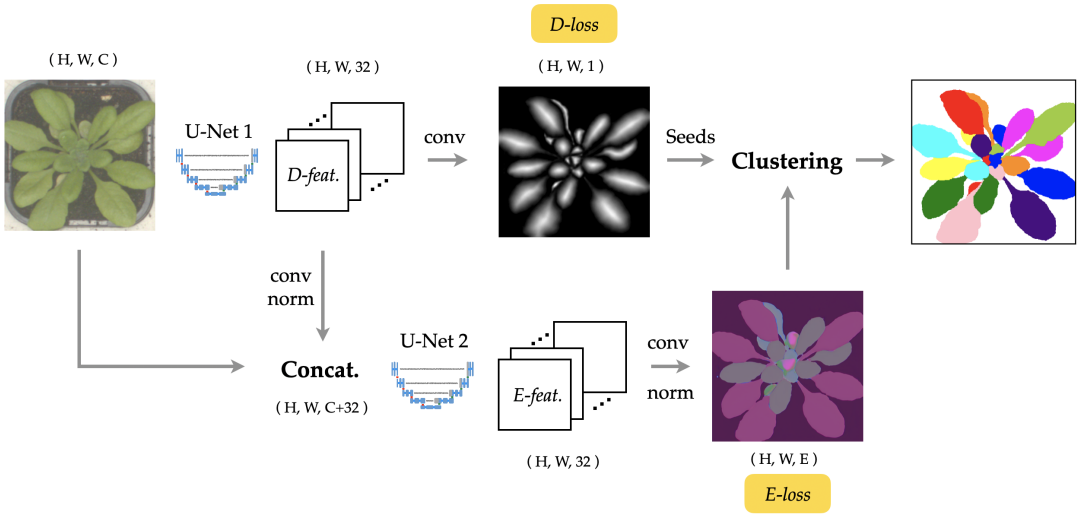


Fig. 1: Processing Pipeline. Distance regression features and distmaps are learned via distance module with U-Net 1. Concatenated distance regression features and images are fed into U-Net 2, from which the embeddings are learned. Final labels are generated based on seeds (thresholded maxima of distmaps) and embeddings via angular clustering. Denotations: H, W, C, E = Dimensions of Height, Width, Channel, Embedding.

are located closely. Then, clustering in the embedding space results in the final instance segmentation. De Brabandere and Neven [5, 12] have proposed Euclidean distance based embedding loss for instance segmentation. Payer et al. [16] have demonstrated embedding loss which utilizes cosine similarity and recurrent stacked hourglass network [13]. Chen et al. [4] have introduced a U-Net based architecture of two heads, where the embeddings are trained with cosine embedding loss and local constraints. These two heads are distance regression head and embedding head. The distance regression head aims to provide seed candidates for clustering. Our proposed method inherits the fundamental modules from this work.

For current pixel embedding based approaches, clustering is an essential step. Mean Shift [6] and HDBSCAN [2] are used in [5] and [16] respectively. In [4, 12], threshold based clustering is used with knowledge of the learned seeds.

3 Method

Our network consists of two cascaded parts (Fig. 1): the distance regression module and the embedding module. Each module uses a U-Net architecture with a 32-dimensional output feature map as the backbone network. The learned distance and embedding feature maps are denoted as *D-feat.* and *E-feat.*, respectively.

The distance regression module takes normalized images as the inputs and outputs the distance map (abbreviated as *distmap* in the following context)

through a single convolutional layer with ReLU activation. The ground truth distmap is obtained by computing the shortest distances from pixels to the object boundary and then being normalized instance-wise with respect to the maximal value. The distance regression module is trained with Mean Squared Error (MSE) loss in this work, which is illustrated as *D-loss* in Fig. 1.

Distance feature map *D-feat.* learned by the distance regression module is fed to the embedding module together with the input image by concatenation. Details of the concatenation are introduced in Section 3.2. The final embeddings are obtained through a convolutional layer with linear activation, followed by L2 normalization. The embedding module is trained with the loss based on the cosine similarity and local constraints (Section 3.1), denoted as *E-loss* in Fig. 1.

The embedding space trained with loss in Eq. 1 has a comprehensive geometric interpretation: embedding vectors of neighboring objects tend to be orthogonal, which simplifies the complexity of clustering. The fast *angular clustering* can be effortlessly performed based on angles between embedding vectors. Firstly, seeds are obtained from distmaps by fetching local maxima with a trivial threshold (selected as 70% of the global maximum in an image). After that, all neighboring pixels within the angular range δ_a of a seed are collected to form a cluster. In this work, we use $\delta_a = 45\,\text{deg}$ for all experiments. At last, the labels outside of the officially provided ground truth foreground masks are omitted.

3.1 Cosine Embedding Loss with Local Constraints

For the embedding module training, we build upon the loss format from [4]. The training loss, denoted as *E-loss* in Fig. 1, is defined based on the cosine similarity $\mathcal{S}_{cos}(\mathbf{e_1}, \mathbf{e_2}) = \mathbf{e_1}^T\mathbf{e_2}/(\|\mathbf{e_1}\|\|\mathbf{e_2}\|)$ and is formularized as:

$$\begin{aligned}
\mathcal{L}_{emb} &= \lambda \cdot \mathcal{L}_{inter} + \mathcal{L}_{intra} \\
\mathcal{L}_{inter} &= \frac{1}{C}\sum_{c_A=1}^{C}\frac{1}{|\mathbf{N}_{c_A}|}\sum_{c_B\in\mathbf{N}_{c_A}}\mathcal{S}_{cos}(\boldsymbol{\mu_{c_A}}, \boldsymbol{\mu_{c_B}}) \\
\mathcal{L}_{intra} &= \frac{1}{C}\sum_{c=1}^{C}\frac{1}{E_c}\sum_{i=1}^{E_c}\Big[1 - \mathcal{S}_{cos}(\boldsymbol{e_i}, \boldsymbol{\mu_c})\Big],
\end{aligned} \tag{1}$$

where the embedding loss is defined as the weighted sum of the between-instance loss term $\mathcal{L}_{inter}$ and within-instance loss term $\mathcal{L}_{intra}$ with the weighting factor λ. $\boldsymbol{e}$ and $\boldsymbol{\mu}$ represents the pixel embedding vector and the mean embedding of an object, respectively. C denotes the number of objects, while the number of pixels of a single object c is denoted as E_c. $\mathbf{N}_{c_A}$ represents the set of neighboring objects around the object c_A and $|\mathbf{N}_{c_A}|$ is the number of neighbors.

The between-instance loss term $\mathcal{L}_{inter}$ encourages the embeddings of different object to be separated, while the within-instance loss term $\mathcal{L}_{intra}$ punishes the case where pixel embeddings of the same object diverge from the mean. In addition, the local constraints of this loss only force neighboring objects to form separable clusters in the embedding space. The benefits of local constraints and the comparison with the global constraint are demonstrated in Section 4.3.

3.2 Feature Concatenative Layer

The feature map *D-feat.* learned by the distance regression module is firstly transformed to the desired dimensions (shown with an example of 32 in Fig. 1) via a convolutional layer and then L2 normalized pixel-wise along through the feature channels before being concatenated to the images. Our experiment shows that the feature map normalization is critical to a stable training process.

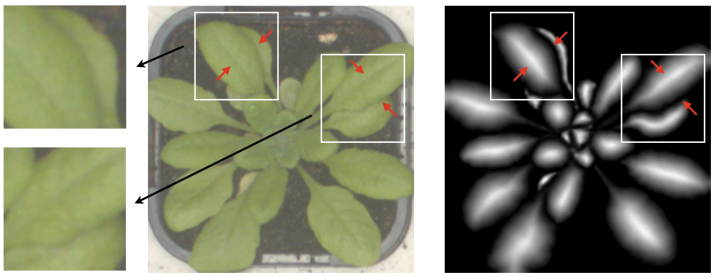
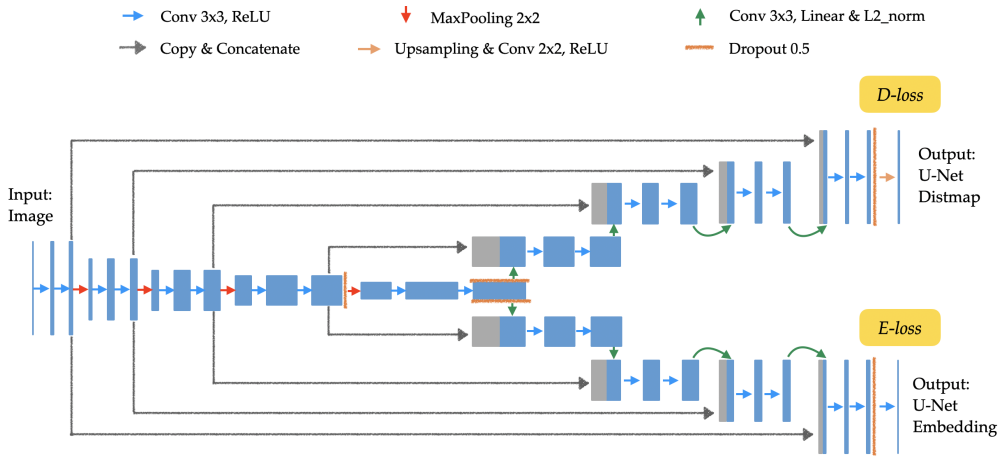


Fig. 2: Ambiguity between Leaf Boundary and Leaf Midvein. Although the embedding space learned with U-Net often fails at such locations, distmap (right) is able to distinguish them well: lower values (darker areas) indicate boundaries and higher values (brighter areas) indicate midveins.

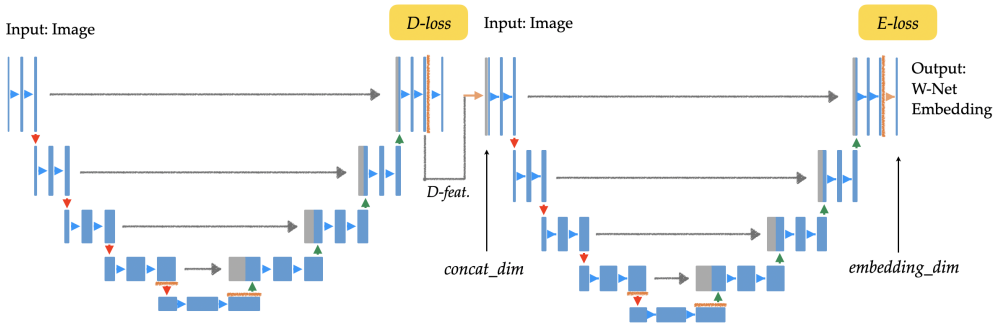
As illustrated in Fig. 2, the difference between leaf boundary and leaf midvein (primary vein) is ambiguous. The learned embeddings by the U-Net architecture [4] often fail at those locations. However, the distmaps are able to tell the difference with lower values representing leaf boundaries and higher values representing leaf midveins. From another perspective, the distmap, which gives an approximate outline of objects, can be interpreted as a *object-ness* score, the pixel-wise probability about existence of object. In addition, as proposed by [14], mixing convolutional operations with the pixel location helps constructing dense pixel embeddings that can separate object instances. From this perspective, the distance regression features can indirectly provide location information to the subsequent module.

To this end, we construct a two-stage architecture, as depicted in Fig. 1, by forwarding the distance regression features to the embedding module. And the concatenation of the distance regression features and images can bring in best performance in the experiments. We term the distance features as concatenative layer in between the stacked U-Nets as *intermediate distance regression supervision*.

In the experiments, other different features have also been tested to forward: the 1-dimensional distmap, 8-dimensional distance features, 32-dimensional distance features, 32-dimensional embedding features, concatenated 16-dimensional distance features and 16-dimensional embedding features. Inspired by [12, 14], we have also evaluated the performance of augmenting the input image with x- and y-coordinates.



(a) U-Net with Two Heads.



(b) W-Net with Intermediate Distance Regression Supervision.

Fig. 3: Network Architectures of U-Net and W-Net.

3.3 From U-Net to W-Net

We abbreviate the proposed network as W-Net to differ from the existing U-Net with two heads, although the novelty and characteristic are not fully represented: the distance regression features as intermediate supervision and the cosine embedding loss with local constraints.

In Fig. 3, the detailed architectures of U-Net with two heads and W-Net with intermediate distance regression supervision are illustrated. The parallel distance and embedding heads of U-Net (Fig. 3a) are modified towards the serial distance and embedding modules in W-Net (Fig. 3b). Apart from the types of concatenative layer as discussed previously, we have also investigated the final dimensions of embeddings as another hyper-parameter, denoted as *embedding_dim*. The corresponding ablation experiments can be found in Section 4.4.

4 Experiments

Ablation experiments are conducted with U-Net and W-Net, as depicted in Fig. 3. The training loss is the sum of the distance regression loss (ReLU+MSE)

and the cosine embedding loss with local constraints (Eq. 1). The latest CodaLab dataset of CVPPP2017 LSC is used as training set without augmentation. Model parameters are initialized by He Normal [8] and optimized by Adam [9]. The initial learning rate is set to 0.0001 and scheduled with exponential decay, with the decay period being set to 5000 steps and the decay rate 0.9. The batch size is set to 4 in most experiments, or 2 if high embedding dimensions are used. The maximal training epochs are set to 500. We show mSBD scores of testing set from CodaLab as the evaluation metric.

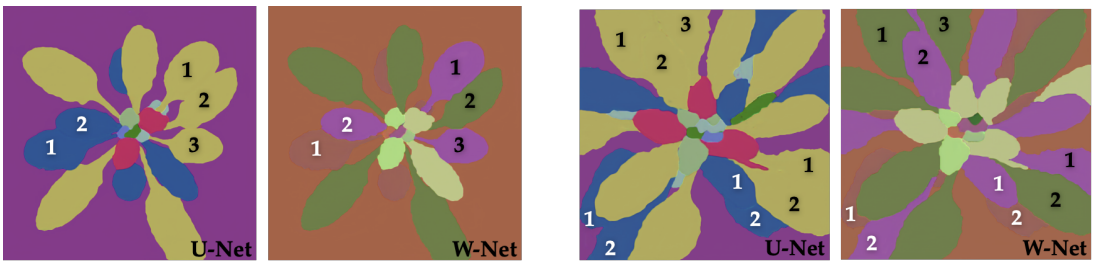


Fig. 4: Learned Embeddings with U-Net and W-Net. Numbered leaves are treated as one object by U-Net, while they are successfully separated in the embedding space learned with W-Net.

4.1 U-Net vs. W-Net

Firstly, we illustrate the performance improvement from U-Net with two heads to the proposed W-Net. In Fig. 4, two representative cases are demonstrated, where the U-Net fails to separate closely located leaves. In contrast, the W-Net has successfully distinguished the numbered leaves in Fig. 4.

Quantitatively, W-Net surpasses U-Net on overall mSBD by approximately 8% from 0.794 to 0.879 with the best set-ups for W-Net, as shown in Table 1. Under different settings of embedding dimensions (Fig. 6a) and loss weights (Fig. 6b), the performance gap between U-Net and W-Net can be continuously observed and remain about 8%.

4.2 Concatenative Layer

We compare the effects of different types of concatenative layer. Firstly, the distmap (1-dimensional) can be directly forwarded. Alternatively, the distance regression features instead of the distmap can be utilized. Before concatenation, we convert the 32-channel *D-feat.* into 8 and 32 dimensions (denoted as *dfeat.8* and *dfeat.32* in Table 1) through a single convolutional layer.

Meanwhile, the case of using embedding loss as the intermediate supervision (*efeat.32*) has also been tested. Specifically, the embedding features from the first U-Net are concatenated with the images as the inputs of the second embedding

Table 1: Comparison of Different Types of Concatenative Layers. Denotation: dfeat.16+efeat.16 = concatenated distance features of 16 dim and embedding features of 16 dim. Others can be analogously educed.

Concatenative Layer	Net	mSBD
none (baseline)	U-Net	.794
coordinate	U-Net	.798
distmap	W-Net	.824
dfeat.8	W-Net	.864
dfeat.32	W-Net	**.879**
efeat.32	W-Net	.847
dfeat.16+efeat.16	W-Net	.873

Table 2: Comparison of Local/Global Constraints, Network and Clustering. Denotations: Local = local constraints, otherwise global; 64d = 64 dim embeddings, otherwise 8 dim; AC = Angular clustering; MWS = Mutex Watershed [24].

Local	Net	Clustering	mSBD
✓	W-Net	AC	**.879**
✓	W-Net 64d	AC	.854
	W-Net	AC	.835
	W-Net 64d	AC	.823
✓	U-Net	MWS	.719
✓	W-Net	MWS	.771
✓	U-Net	MeanShift	.679
✓	W-Net	MeanShift	.733
✓	U-Net	HDBSCAN	.631
✓	W-Net	HDBSCAN	.681

module. Furthermore, the concatenated distance regression features and embedding features (*dfeat.16+efeat.16*) are also investigated. At last, augmenting the input image with coordinates is tested. As proposed in [14], constructing dense object-aware pixel embeddings cannot be easily achieved using convolutions and the situation can be improved by incorporating information about the pixel location. In this work, we augment the input image with two coordinate channels for the normalized x- and y-coordinates, respectively.

Experimental results are summarized in Table 1. First of all, forwarding distmaps is not as effective as forwarding feature maps, including the distance regression features and the embedding features. The embedding features (*efeat.32*) can also boost the performance, but not as significantly as the distance regression features. This is verified by the fact that *efeat.32* is worse than *dfeat.32* and the mixed feature map *dfeat.16+efeat.16*. For the distance regression feature itself, higher dimensions of 32 are preferred. Finally, augmenting images with coordinates does not show apparent differences in our experiments. The effects could be further studied. For example, augmenting each intermediate feature map with coordinates is also worth being investigated.

4.3 Local vs. Global Constraints

Local constraints make it possible to exploit lower-dimensional embedding space more efficiently, as in this case, different labels only have to be distributed to the neighboring objects. In contrast, the global constraints have to thoroughly give each single object in the images a different label, which requires larger receptive fields and more redundant embedding space. The combination of local

constraints and cosine embeddings utilizes the embedding space further comprehensively, as the push force imposed by loss expects orthogonal embedding clusters for neighboring instances.

This is confirmed qualitatively by examples showcased in Fig. 5. In Fig. 5c, 8-dimensional embeddings are trained with global constraints. Not surprisingly, there are exactly 8 colors in the image, indicating 8 orthogonal clusters in the embedding space. Apparently, the global constraint will fail when the embedding dimensions are fewer than the number of objects. In contrast, the local constraints (Fig. 5a - 5b) can distribute labels alternately between objects, with the same labels appearing multiple times for non-adjacent objects. This makes it possible to utilize a lower-dimensional embedding space. Quantitatively, the W-Net trained with local constraints surpasses the one trained with global constraints by more than 4% on overall mSBD, as listed in Table 2.

Intuitively, a higher-dimensional embedding space is able to provide a higher degree of freedom, i.e. we could simply use higher-dimensional embeddings to alleviate the problem of global constraints. At least the embedding vector does not have to be restricted to low dimensions. However, from the results in Fig. 6a, we find that higher-dimensional embeddings produce worse results. This makes the capability of using lower-dimensional embedding space particularly important.

(a) local 8 (b) local 64 (c) global 8 (d) global 64

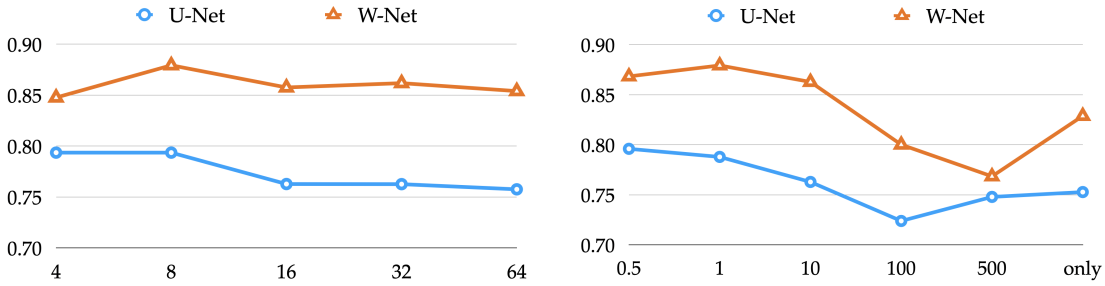
Fig. 5: Learned Embeddings for Combined Cases of Local/Global Constraints and 8/64-dimensional Embeddings. (a,b) vs. (c,d): Local constraints ensure the effective utilization of embedding space, as same embeddings appear alternately for non-adjacent objects. (a) vs. (b): Higher-dimensional embeddings are redundant in the local constraint case. (c) vs. (d): Lower-dimensional embeddings with global constraints are not sufficient to distinguish all objects. This problem is slightly mitigated via higher-dimensional embeddings, still not as effective as local constraints.

4.4 Dimensions of Embeddings

As discussed previously, the local constraints make the use of lower-dimensional embedding possible. It is thus worth investigating the influence of different embedding dimensions on the overall performance. The mSBD scores of both U-Net

and W-Net for {4, 8, 16, 32, 64}-dimensional embeddings are plotted in Fig. 6a. For 32 and 64 dimensions, the batch size is set to 2, instead of 4 as in other cases, to fit the memory of a single GPU.

Our experiments show that the 8-dimensional embedding brings in the best result. First of all, merely 4 dimensions are incompetent to separate all adjacent objects, since it is common that one object has more than 4 neighbors. Although higher dimensions may not bring in more labels under local constraints, comparing Fig. 5a to 5b, increasing the embedding dimensions should not degrade the performance hypothetically. However, the mSBD score decreases slightly as the dimensions increase. Therefore we believe, under the premise that the dimensions are sufficient for all objects to fulfill the local constraints, higher-dimensional embedding space is more difficult to train.



(a) Dimensions of Embeddings

(b) Loss Weights λ

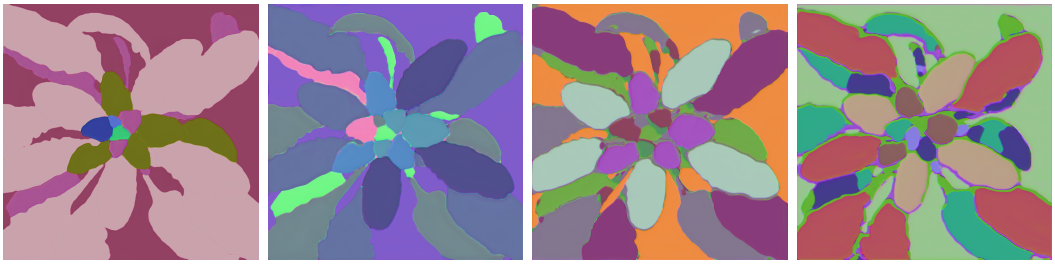
Fig. 6: mSBD w.r.t. Dimensions of Embeddings and Loss Weight λ in $\mathcal{L}_{emb} = \lambda \cdot \mathcal{L}_{inter} + \mathcal{L}_{intra}$ using U-Net and W-Net. In (b), *only* denotes $\mathcal{L}_{emb} = \mathcal{L}_{inter}$. W-Net surpasses U-Net generally. Best overall performance of W-Net can be obtained with 8-dimensional embeddings and $\lambda = 1$.

4.5 Loss Weights

During the experiments, we find that the values of between-instance loss term $\mathcal{L}_{inter}$ are approximately 10 times greater than the values of within-instance loss term $\mathcal{L}_{intra}$. This is consistent with the fact that pixel embeddings of the same object converge tightly, but adjacent objects are not correctly segmented occasionally. The larger weighting factor λ of between-instance loss term $\mathcal{L}_{inter}$ might be helpful to emphasize the significance of it by amplification of its gradient. We set λ as $\{0.5, 1, 10, 100, 500\}$, and moreover, we omit the within-instance loss, denoted as *only* in Fig. 6b. The experiments are preformed for both U-Net and W-Net under identical main set-ups: 32-dimensional distance features as concatenative layer, local constraints and 8-dimensional embeddings.

From the experiments, we find that larger weighting factor of the between-instance loss term does not further help to encourage the network to separate the confused objects when λ is larger than 1, but reduces the consistency of

embeddings in the same object. Fig.7 showcases the trade-off between the discrimination of adjacent objects (larger λ) and the consistency of individual object (smaller λ). The experiments show that $\lambda = 1$ brings in best overall performance, as shown in Fig. 6b. Besides, one surprising conclusion is that training the network with just the between-instance loss term can also, to some extent, form clusters in the embedding space (Fig. 7d).



(a) $\lambda = 0.5$ (b) $\lambda = 10$ (c) $\lambda = 100$ (d) only $\mathcal{L}_{inter}$

Fig. 7: Learned Embeddings with Different Weights λ as in $\mathcal{L}_{emb} = \lambda \cdot \mathcal{L}_{inter} + \mathcal{L}_{intra}$. With ascending λ, overall segmentation performance becomes worse (Fig. 6) with the decreased consistency of embeddings in the same object. It is worth noting that training with just the between-instance loss term can also to some extent form clusters in the embedding space.

4.6 Clustering

Apart from the default angular clustering used along through the experiments, other three clustering techniques have been tested based on the predicted embeddings of the best results: Mutex Watershed [24], Mean Shift [6] and HDBSCAN [2]. On the one hand, this provides a reference for the performance of different clustering methods on the embeddings trained with cosine similarity based loss. On the other hand, it can also indirectly reflect the quality of embeddings generated by U-Net and W-Net. Results are shown in Table 2.

In conclusion, the angular clustering has advantages in terms of performance and speed. Nevertheless, it should be noted that this method is only applicable to the case, where seeds are available and clusters are orthogonal in the embedding space. Additionally, all clustering approaches produce better results with embeddings predicted from the W-Net, which again confirms the improvement of our proposed method.

4.7 Comparison against State-of-the-Art

Comparison of state-of-the-art methods on the CVPPP LSC dataset is quantitatively shown in Table 3. It is clear that the learning based methods (denoted with backbones) can achieve better results than the first four classical methods.

Table 3: Comparison of Results. Abbreviations: Aug. = Data augmentation; Emb. = Metric of embedding similarity; Fg. = Ground truth foreground masks are used; syn = Synthetic images are used for training; HG = Stacked Hourglass network; Lb. = Results shown in the leaderboard of CodaLab.

Method	Backbone	Train	Aug.	Emb.	Fg.	Lb.	mSBD		
							A1	A1-3	A1-5
IPK [15, 19]	-	A1-3			✓		.791	.782	-
Nottingham [19]	-	A1-3			✓		.710	.686	-
MSU [19, 26]	-	A1-3			✓		.785	.780	-
Wageningen [19]	-	A1-3			✓		.773	.769	-
MRCNN [4, 7]	ResNet	A1-3					-	.797	-
Stardist [4, 20]	U-Net	A1-3					-	.802	-
IS-RA [17]	FCN	A1					.849	-	-
Ward [23]	ResNet	A1-4+syn	✓				.900	.740	.810
UPGen [22]	ResNet	A1-4+syn	✓				.890	**.877**	.874
DiscLoss [5]	ResNet	A1	✓	euc	✓		.842	-	-
CE-RH [16]	HG	A1	✓	cos			.845	-	-
E-LC [4]	U-Net	A1-3		cos			-	.831	.823
W-Net (ours)	U-Net	A1-4		cos	✓	✓	**.919**	.870	**.879**

The last four methods are based on pixel embedding learning. Roughly speaking, they bring in promising results. Our overall result mSBD for A1-5 outperforms all others. In the leaderboard, our overall result is at the 1. position by paper submission. Furthermore, the average of mSBD scores for Arabidopsis images (A1, A2, A4) outperforms the second best results from three different users, respectively, by over 3%, namely 0.883 to 0.917. Due to the extremely imbalanced training images on Arabidopsis (783 images) and Tobacco (27 images), our result on testing set A3 are not as good as others, with mSBD of 0.77. Compared to this, the current 1. place mSBD of A3 in the leaderboard reaches 0.89. It implies that the sufficient number of training images is critical in our proposed method. We leave this room for improvement in the future. One thing worth mentioning is that the authors tend to not submit their results to the leaderboard of CodaLab, which makes the consistent comparison and review rather difficult.

4.8 Application to Human U2OS Cells

Our method has also been tested on the image set BBBC006v1 of human U2OS cells from the Broad Bioimage Benchmark Collection [11]. Totally 754 images are randomly separated into two equally distributed training and testing set with 377 images respectively. Other set-ups are identical to previously introduced ones. We use U-Net and W-Net with distance concatenative layer to show results in mSBD and mean Average Precision with IoU=$\{0.5, 0.55, 0.6, ..., 0.9\}$ (mAP). The mSBD has increased from 0.896 to 0.915 and the mAP from 0.577 to 0.664.

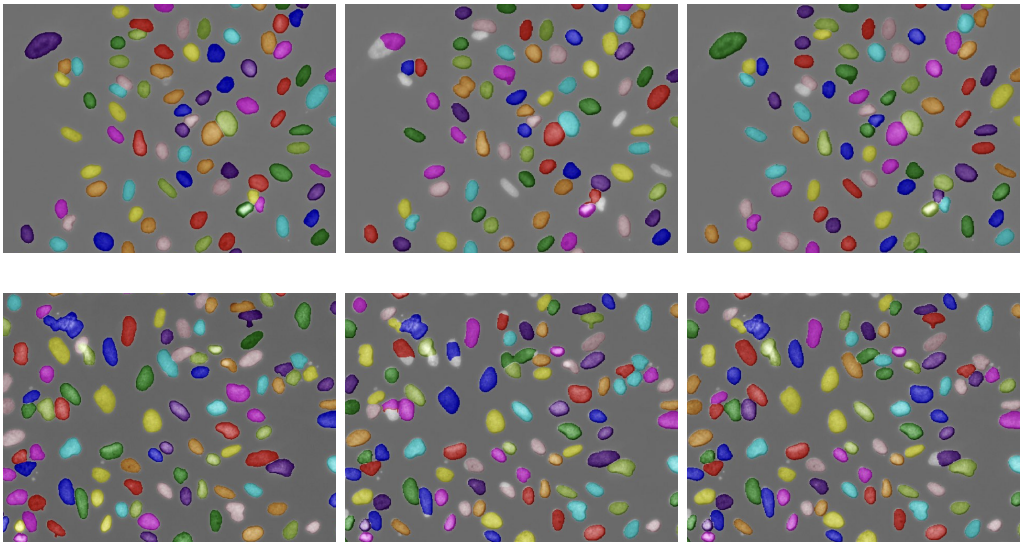


Fig. 8: Cell Segmentation Results of the BBBC006v1 Data: Ground Truth (left), U-Net (middle) and W-Net (right); Each row illustrates one example. The improvement from U-Net to W-Net is salient. The mSBD and mAP scores have increased from 0.896 to 0.915 and from 0.577 to 0.664, respectively.

We showcase two examples of final labels in Fig. 8. As reported in [4], some embeddings around boundaries might be incomplete, which leads to incomplete segmentations. This problem has been mainly solved, as showcased in Fig. 8.

5 Conclusion

In this work we propose a novel W-Net, which forwards the distance regression features learned by the first-stage U-Net to the subsequent embedding learning module. The intermediate distance regression supervision effectively promotes the accuracy of learned pixel embedding space, with the mSBD score on the CVPPP LSC dataset increased by more than 8% compared to the identical set-up without supervision of distance regression features. We have also conducted a number of experiments to investigate the characteristics of the pixel embedding learning (the cosine similarity based loss), involving the embedding dimensions, the weighting factor of the within-instance loss term and the between-instance loss term. We are looking forward to applying this method to more datasets in the future.

References

1. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Annual International Conference on Machine Learning. pp. 41–48 (2009)
2. Campello, R.J., Moulavi, D., Zimek, A., Sander, J.: Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM Transactions on Knowledge Discovery from Data (TKDD) **10**(1), 1–51 (2015)
3. Chen, H., Sun, K., Tian, Z., Shen, C., Huang, Y., Yan, Y.: BlendMask: Top-down meets bottom-up for instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
4. Chen, L., Strauch, M., Merhof, D.: Instance segmentation of biomedical images with an object-aware embedding learned with local constraints. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 451–459. Springer (2019)
5. De Brabandere, B., Neven, D., Van Gool, L.: Semantic instance segmentation with a discriminative loss function. arXiv preprint arXiv:1708.02551 (2017)
6. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on information theory **21**(1), 32–40 (1975)
7. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
8. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: The IEEE International Conference on Computer Vision (ICCV) (December 2015)
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
10. Lee, Y., Park, J.: Centermask: Real-time anchor-free instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
11. Ljosa, V., Sokolnicki, K.L., Carpenter, A.E.: Annotated high-throughput microscopy image sets for validation. Nature methods **9**(7), 637–637 (2012)
12. Neven, D., Brabandere, B.D., Proesmans, M., Gool, L.V.: Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8837–8845 (2019)
13. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European conference on computer vision. pp. 483–499. Springer (2016)
14. Novotny, D., Albanie, S., Larlus, D., Vedaldi, A.: Semi-convolutional operators for instance segmentation. In: European Conference on Computer Vision (2018)
15. Pape, J.M., Klukas, C.: 3-d histogram-based segmentation and leaf detection for rosette plants. In: European Conference on Computer Vision. pp. 61–74. Springer (2014)
16. Payer, C., Štern, D., Neff, T., Bischof, H., Urschler, M.: Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 3–11. Springer (2018)
17. Ren, M., Zemel, R.S.: End-to-end instance segmentation with recurrent attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6656–6664 (2017)

18. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
19. Scharr, H., Minervini, M., French, A.P., Klukas, C., Kramer, D.M., Liu, X., Luengo, I., Pape, J.M., Polder, G., Vukadinovic, D., et al.: Leaf segmentation in plant phenotyping: a collation study. Machine vision and applications **27**(4), 585–606 (2016)
20. Schmidt, U., Weigert, M., Broaddus, C., Myers, G.: Cell detection with star-convex polygons. In: Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II. pp. 265–273 (2018)
21. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9627–9636 (2019)
22. Ward, D., Moghadam, P.: Scalable learning for bridging the species gap in image-based plant phenotyping. arXiv preprint arXiv:2003.10757 (2020)
23. Ward, D., Moghadam, P., Hudson, N.: Deep leaf segmentation using synthetic data. arXiv preprint arXiv:1807.10931 (2018)
24. Wolf, S., Pape, C., Bailoni, A., Rahaman, N., Kreshuk, A., Kothe, U., Hamprecht, F.: The mutex watershed: efficient, parameter-free image partitioning. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 546–562 (2018)
25. Xie, E., Sun, P., Song, X., Wang, W., Liu, X., Liang, D., Shen, C., Luo, P.: Polarmask: Single shot instance segmentation with polar representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
26. Yin, X., Liu, X., Chen, J., Kramer, D.M.: Multi-leaf tracking from fluorescence plant videos. In: 2014 IEEE International Conference on Image Processing (ICIP). pp. 408–412. IEEE (2014)
27. Zhang, R., Tian, Z., Shen, C., You, M., Yan, Y.: Mask encoding for single shot instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)