Learning to Infer Parameterized Representations of Plants from 3D Scans

Supplementary Material

1. Implementation Details

We implemented our method using Pytorch on Quadro RTX 5000 GPU. We learn a latent space $\mathcal{S}$ of dimension $dim_{\mathcal{S}} = 64$ with the recursive encoder-decoder pair architecture in Fig. 4 in the main paper, of hidden layer size $h_{recur} = 128$, and the classifiers with a hidden layer $h_{class} = 128$. We optimize the weights of the network using Adam optimizer with a learning rate of 0.001, decaying by factor of 0.5 every 50 epochs, on batches of size 32. The point cloud encoder is a PointNet point regression network that has the architecture shown in Fig. 2 in [7] without the optional feature transformation and with reduced dimensions for all layers. We train it with Adam optimiser on a learning rate of 0.001 on batches of size 32 without scheduling. The weights of the layers of all the trained networks are initialized using Xavier uniform initialization.

Our method works on binary trees, while the L-String trees in our dataset do not have a binary tree structure in general. We summarize the L-Strings into binary trees, by combining the modules that always occur together as individual nodes with concatenated parameter sets. In particular, we combine the two cotyledon modules into one *Cotyledons* node with 6 parameters. Stems are always followed by a petiole and a leaf, therefore we combine them in one node called *Stem* that has 13 parameters. In case of a branching, the branch module is combined with the stem, petiole and leaf modules to form a *Branch* node of 15 parameters. Finally, the first stem module of the tree forms a node on its own called *Root* of 5 parameters. Parameters that are drawn from a Gaussian distribution do not correlate with their parent in the tree structure and cannot be learned by the recursive network, thus they are set to the mean during the training and are then optimized with other parameters in the test-time optimization phase.

For the segmentation experiments where we use the k nearest neighbors algorithm, we set $k = 10$ for our results, and $k = 3$ for PSegNet results. As for the reconstruction experiments, when using SIREN we need to input the point normals along with the point cloud, for this we use a normal estimation PCA-based method, which fits local planes to each point's neighborhood.

2. Additional evaluations

2.1. Semantic Segmentation

We quantitatively evaluate our method's performance on the semantic segmentation task applied to our test sets using standard classification metrics precision, recall, F1 score, and Intersection over Union (IoU). For each class, precision is defined as the ratio of true positives to the sum of true positives and false positives, while recall is the ratio of true positives to the sum of true positives and false negatives. The F1 score is the harmonic mean of precision and recall. Finally, IoU is computed as the ratio of the intersection to the union of the predicted and ground truth point sets for a given class. Table 1 shows the results on all test sets. Our method is robust to noise and performs overall on-par with the strong baselines PlantNet [4] and PSegNet [3] on each individual task, while performing all the tasks at once.

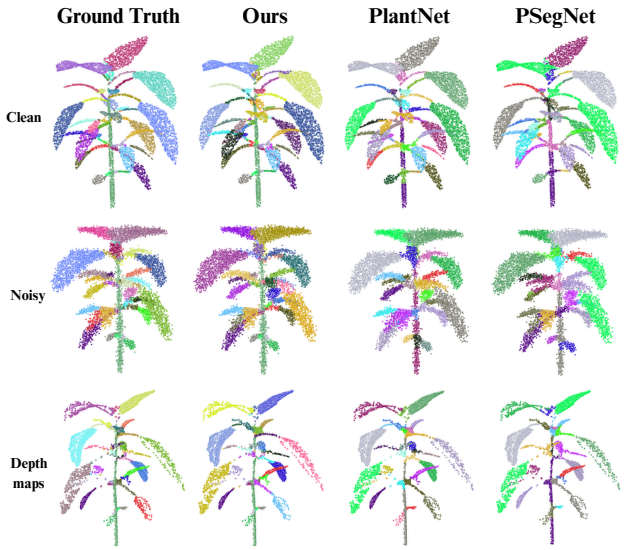

Figure 1. Visual comparison between our method, PlantNet and PSegNet for the instance segmentation task. Different colors are assigned to segmented instances without correspondence to the ground truth.

2.2. Instance Segmentation

We qualitatively compare our method to PlantNet [4] and PSegNet [3] for the Instance segmentation task. Fig. 1 shows the instance segmentation results on examples from all test sets, where each individual plant organ is assigned a unique color. Note that the colors are to distinguish the segmented organs without any correspondence between the results and the ground truth. Our method is on-par with the baselines for this task where it is able to segment individual organ instances and is robust to noise and partial data.

	Stem				Leaf				Petiole				Cotyledons			
	Prec.	Rec.	F1	IoU	Prec.	Rec.	F1	IoU	Prec.	Rec.	F1	IoU	Prec.	Rec.	F1	IoU
Clean																
PlantNet	85.3	88.2	86.7	76.6	98.4	99.5	98.9	97.9	82.1	71.7	76.5	62.0	98.9	94.4	96.7	93.5
PSegNet	99.1	99.3	99.2	98.5	98.4	98.9	98.6	97.3	66.6	59.6	62.9	45.9	99.7	97.0	98.6	97.3
Ours	84.8	88.0	86.2	75.9	98.0	98.4	98.2	96.5	66.7	59.4	62.2	45.9	96.7	95.8	96.2	92.7
Noisy																
PlantNet	84.2	88.4	86.2	75.8	97.7	99.1	98.4	96.9	77.6	62	69.0	52.6	96.4	93.1	94.7	89.9
PSegNet	99.3	83.6	90.7	83.1	95.9	99.9	97.9	95.9	96.2	29.6	45.3	29.3	99.6	98.1	98.9	97.7
Ours	83.0	86.3	84.5	73.3	97.9	98.3	98.1	96.2	64.2	57.4	59.9	43.7	95.8	93.9	94.8	90.2
Depth maps																
PlantNet	86.2	86.8	86.5	76.2	98.5	99.2	98.8	97.7	70.2	65.6	67.8	51.3	97.4	88.7	92.9	86.7
PSegNet	99.1	98.9	99.0	98.1	96.2	99.9	98.1	96.2	98.9	54.9	70.6	54.6	99.8	98.7	99.2	98.5
Ours	84.6	84.2	84.0	73.2	98.4	98.6	98.5	97.0	57.9	54.1	54.5	38.8	96.8	94.8	95.7	92.0

Table 1. Comparison to PlantNet [4] and PSegNet [3] for semantic segmentation on clean and noisy points test sets using standard classification measures Precision, Recall, F1-Score and Intersection over Union (IoU). All values are percentages.

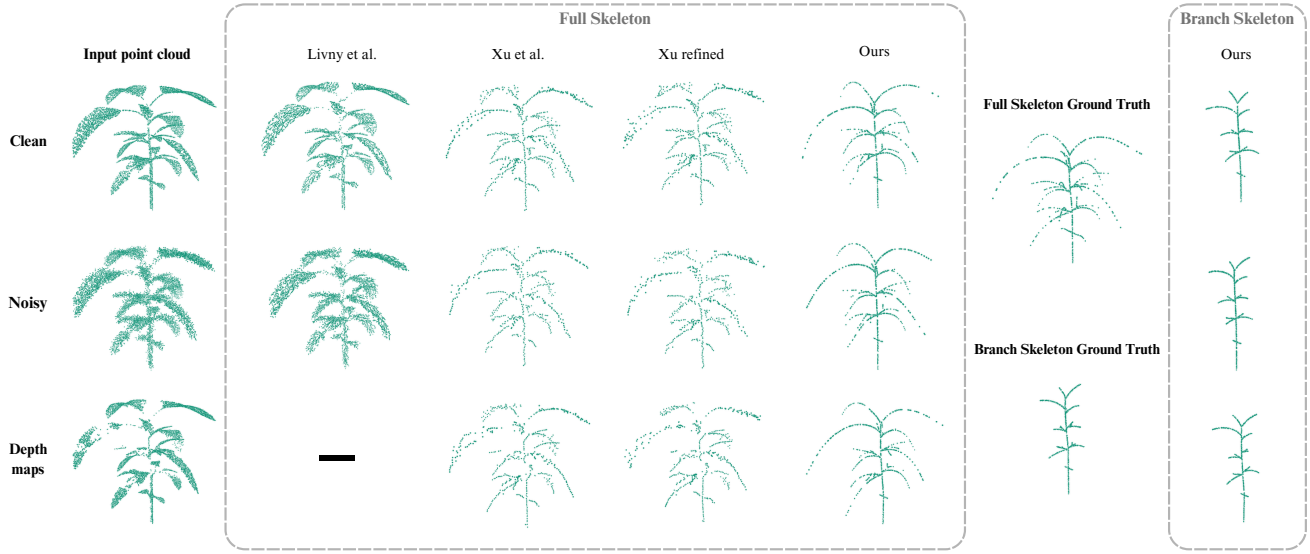

Figure 2. Comparison to Livny *et al*. [5], Xu *et al*. [8], and Chaudhury *et al*. [1] refinement on Xu skeleton for skeletonization. "-" means that the method crashed due to numerical problems. Note that ours is the only method applicable at different scales, and able to output a full skeleton or a branch skeleton, while achieving visually accurate results.

2.3. Skeletonization

To support the results shown in Table 3 in the main paper on the skeletonization task, we qualitatively compare to the baselines Livny *et al*. [5], Xu *et al*. [8] and Chaudhury *et al*. [1] on all test sets. Fig. 2 shows the visual qualitative results of all methods on extracting a full skeleton from an input point cloud, with our results of extracting the branch skeleton compared to the ground truth. Our results are close to the ground truth skeletons and are robust to noise and partial input on both scales.

2.4. Ablation Studies

Our biologically inspired network uses recursive auto-encoders, which consist of recursive encoder-decoder pairs and classifiers. This idea is implemented using the simplest individual components, namely multi-layer perceptrons with a single hidden layer for the recursive encoders E and decoders D, and the classifiers C_{node}, C_{split}. Classifiers in recursive neural networks are required as stopping criteria for the recursive decoding and to select the node type; hence, they cannot be removed for ablation. Table 2 provides a neuronal ablation where for each component C_{node}, C_{split}, and $E\&D$, 75% of all neurons are randomly deactivated. For each model, we measure the Tree-Edit Dis-

tance (TED) [2] between input and reconstructed L-Strings of the test set. TED measures the difference between two tree graphs w.r.t. the number of operations needed to translate from one tree structure to the other, with additional local cost functions corresponding to the difference between the node parameters in our case. Hence, it measures the difference in both topology and shape between plants represented as L-strings. The full model has lowest TED.

	Full model	C_{node}	C_{split}	$E\&D$
TED	0.196	0.207	0.204	0.240

Table 2. Table shows an ablation where the full model is compared to models where 75% of all neurons are deactivated in parts of the architecture. For each model, we measure tree edit distance.

2.5. Influence of Latent Dimension

We study the influence of the dimension of the learned latent space $\mathcal{S}$ on the performance of our model. For that, we evaluate the L-string latent representation stage by training the recursive auto-encoders with different latent dimensions $dim_{\mathcal{S}}$ and measuring the TED between the input L-strings and the reconstructions. Table 3 shows the average reconstruction error in TED for different dimensions, in our implementation we choose $dim_{\mathcal{S}}$ that is most compact with minimum error.

$dim_{\mathcal{S}}$	Tree Edit Distance
32	0.227
64	**0.196**
128	0.197
256	0.296

Table 3. Analysis of the influence of latent dimension on the performance of the L-string auto-encoder model w.r.t. the TED.

2.6. Latent Space Analysis

To analyse the data distribution in the learned latent space $\mathcal{S}$, we project the latent points representing plants from the training set in Fig. 3 using UMAP [6], the uniform manifold approximation and projection non-linear technique for dimension reduction with local and global structure preservation. Different plant structures are assigned different colors in Fig. 3, one can notice that our model learns to encode plants sharing the same tree structure close by in $\mathcal{S}$.

2.7. Generalization to Complex Structures

To further assess the generalization capacity of our method, we tested how well the trained model generalizes to virtual Chenopodium album in growth stages with a structural complexity higher than that used during training. Fig. 4 shows the TED w.r.t. the complexity of the plant (using L-String size as a proxy). This indicates that our method generalizes well to virtual plants with higher complexity.

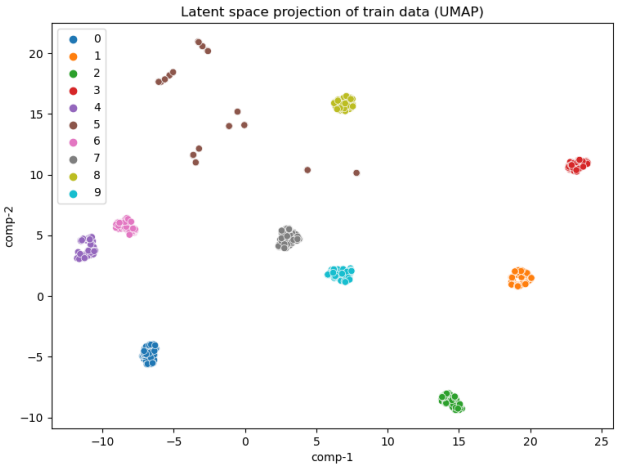

Figure 3. Analysis of the distribution of different tree structures from the training data in the latent space projected in 2D using UMAP. Each color represent a unique plant structure in the training data. Plants sharing the same structure form clusters in $\mathcal{S}$.

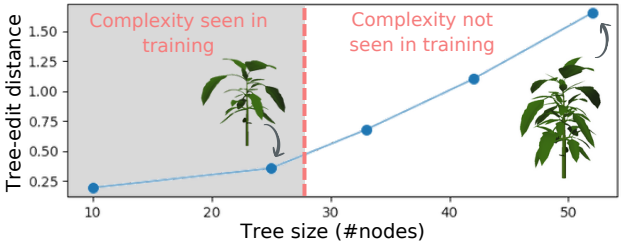

Figure 4. Tree edit distance w.r.t. the development of the plant.

References

[1] Ayan Chaudhury and Christophe Godin. Skeletonization of plant point cloud data in stochastic optimization framework. *bioRxiv*, 2020. 2

[2] Pascal Ferraro and Christophe Godin. A distance measure between plant architectures. *Annals of Forest Science*, 57(5/6): 445–461, 2000. 3

[3] Dawei Li, Jinsheng Li, Shiyu Xiang, and Anqi Pan. Psegnet: Simultaneous semantic and instance segmentation for point clouds of plants. *Plant Phenomics*, 2022, 2022. 1, 2

[4] Dawei Li, Guoliang Shi, Jinsheng Li, Yingliang Chen, Songyin Zhang, Shiyu Xiang, and Shichao Jin. Plantnet: A dual-function point cloud segmentation network for multiple plant species. *ISPRS Journal of Photogrammetry and Remote Sensing*, 184:243–263, 2022. 1, 2

[5] Yotam Livny, Feilong Yan, Matt Olson, Baoquan Chen, Hao Zhang, and Jihad El-Sana. Automatic reconstruction of tree skeletal structures from point clouds. In *ACM SIGGRAPH Asia 2010 papers*, pages 1–8. 2010. 2

[6] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. 3

[7] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and

Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. 1

[8] Hui Xu, Nathan Gossett, and Baoquan Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph.*, 26(4):19–es, 2007. 2