Supplementary:
CellTypeGraph: A New Geometric Computer Vision Benchmark

Lorenzo Cerrone[1] Athul Vijayan[3] Tejasvinee Mody[2] Kay Schneitz[2]

Fred A. Hamprecht[1]

[1]HCI, Heidelberg University, Germany
[2]School of Life, Technical University of Munich, Germany
[3] Max Planck Institute for Plant Breeding Research, Germany
lorenzo.cerrone@iwr.uni-heidelberg.de

1. Global reference axis

Our landmark-based global reference systems are defined by fixing an origin and an orthogonal 3D vectors basis. A good land-mark should be retrievable from the ground truth labels at training time, but should also be easy to identify by a human without ground truth labels at test time.

We choose the following four approaches:

Label Surf: The origin is defined as the center of mass of the *L1* tissue. The main axis is found by approximately determining the integument tissue symmetry axis. This is achieved by finding for each tissue in the integument (*L2, L3, L4, es, nu*) the respective center of mass. Then, we use least-square linear regression to find the best line interpolating the integument tissue. The second axis is found by finding the line passing through *fu* tissue center of mass and perpendicular to the main axis. The third axis is computed by taking the cross-product between the first and second axes.

Label Fu: The origin is defined as the center of mass of the *fu* tissue. The global axes are computed as in the *Label Surf*.

Es trivial: Here we use the *Es* tissue to fix an origin for the reference system. It is usually easily identifiable by its large size and central position. We set the reference system origin to the *Es* center of mass. While for the axis, we use the original orientation as acquired by the microscope.

Es PCA: The origin is the same as *Es Trivial*. But the system axes are set to the PCA axes of the whole ovule.

Our python implementation can be found at: https://github.com/hci-unihd/plant-celltype/blob/main/plantcelltype/features/cell_vector_features.py.

2. Growth and surface axis

To compute the local reference system, we designed two simple heuristics. We estimate the surface axis of a cell by averaging over directions corresponding to the edges connecting the cell to all neighbors which are closer to the surface. These directions are defined as the edge surface normal direction.
The growth axes are found by looking for each cell the most co-linear pair of neighboring cells.

The algorithms used to compute the surface axis and growth axis are described in more detail respectively in Algorithm 1 and Algorithm 2. Our python implementation can be found at: https://github.com/hci-unihd/plant-celltype/blob/main/plantcelltype/utils/axis_transforms.py.

3. Features

We here report the results of additional experiments conducted to identify the best feature homogenization strategy. In Fig. 1, we tested the difference between different vector representations. In Fig. 3 and Fig. 2, we tested the normalizations for the graph features and the maximum value to be used in our hops to surface feature. In Fig. 4 and Fig. 5 we tested different normalization applied to respectively morphological and angles features. Lastly, in Fig. 6 we tested the impact of using lengths measured in several directions as features, similarly to [9]. An overview of all

Algorithm 1 Compute Surface Axis

Require: node, edges ▷ Vectors containing: nodes ids, edges ids.
Require: hops ▷ Vectors containing: number of hops from each node to the surface.
Require: directions, bg ▷ Vector containing: edges directions (surface normal), background node id.
  N ← *len*(node)
  surface-axis ← *zeros*(N, 3) ▷ Initialize an array full of zeros.
  for ($i = 0$, $i = \text{N}$, i++) **do**
    if $\text{hops}_i = 1$ **then** ▷ I.e. node is on the organ surface.
      e ← *find-edge*(node_i, bg, edges) ▷ Find edge id between node_i/node_j.
      $\text{surface-axis}_i \leftarrow \text{directions}_e$
    else
      $\text{neighbors}_i \leftarrow$ *find-neighbors*(node_i, edges) ▷ Find all nodes neighbors of node_i.
      N_i, count ← *len*(neighbors_i), 0
      for $j = 0$, $j = \text{N}_i$, j++ **do** ▷ Loop over the neighborhood.
        if $\text{hops}_j < \text{hops}_i$ **then** ▷ If neighbor is closer to surface.
          e ← *find-edge*(node_i, node_j, edges) ▷ Find edge id between node_i/node_j.
          $\text{surface-axis}_i \leftarrow \text{surface-axis}_i + \text{directions}_e$ ▷ Average edges normal.
          count = count + 1
        end if
        $\text{surface-axis}_i \leftarrow \text{surface-axis}_i/\text{count}$
      end for
    end if
  end for

Algorithm 2 Compute Growth Axis

Require: node, edges ▷ Vectors containing: nodes ids, edges ids.
Require: coms, hops ▷ Vectors containing: nodes center of mass, number of hops from each node to the surface.
  N ← *len*(node)
  growth-axis ← *zeros*(N, 3) ▷ Initialize an array full of zeros.
  for ($i = 0$, $i = \text{N}$, i++) **do**
    $\Theta_{min} \leftarrow 1$
    $\text{neighbors}_i \leftarrow$ *find-neighbors*(node_i, edges) ▷ Find all nodes neighbors of node_i.
    $\text{N}_i \leftarrow$ *len*(neighbors_i)
    for $j = 0$, $j = \text{N}_i$, j++ **do** ▷ Loop over all tuple (node_j, node_j) of distinct neighbors of node_i.
      vector_{ij} = *get-vector*(node_i, node_j, coms) ▷ Find vector connecting node_i/node_j center of masses.
      for $k = j + 1$, $k = \text{N}_i$, k++ **do**
        vector_{ik} = *get-vector*(node_i, node_k, coms)
        Θ = *get-angle*(vector_{ij}, vector_{ik}) ▷ Angle is measured as the normalized dot product between the two vectors.
        if $\Theta < \Theta_{min}$ **and** $\text{hops}_i = \text{hops}_j = \text{hops}_k$ **then** ▷ When the angle is minimum the cell are the most co-linear.
          $\Theta_{min} \leftarrow \Theta$
          $\text{growth-axis}_i \leftarrow \text{vector}_{ij}$
        end if
      end for
    end for
  end for

features available in the CellTypeGraph benchmark is reported in Tab. 1, Tab. 2. Our python implementation can be found at: https://github.com/hci-unihd/plant-celltype/tree/main/plantcelltype/features

4. Grid search complete results

Experiment parameters setup is reported in Tab. 3. The configuration files used to run our experiments can be fount at: https://github.com/hci-unihd/plant-celltype/tree/main/experiments/

Feature Name	Invariant	Default	Size	Description
Center of mass		✓	3	Cell center of mass represented in the global reference system, expressed in μm.
Center of mass / GRS Proj.			3	Angles between the cell center of mass and the global reference system.
LRS axis			9	Growth axis, surface axis and third perpendicular axis.
LRS orientations		✓	18	Direction invariant growth axis, surface axis and third perpendicular axis.
LRS / GRS Proj.	✓	✓	9	Angles between the LRS axis and the global reference system.
Growth/Surface axis angle	✓	✓	1	Angle between growth and surface axis.
Growth axis alignment	✓	✓	1	Angle between periclinial cell walls along predicted growth direction, measure how good is the fit is.
Length LRS	✓	✓	3	Cell length along the LRS directions.
PCA axis			9	Principal component analysis axis.
PCA orientations		✓	18	Direction invariant principal component analysis.
PCA / GRS Proj.	✓	✓	9	Angles between the PCA axis and the global reference system.
PCA explained variance	✓	✓	3	PCA axis explained variance.
Surface	✓	✓	1	Cell surface area in μm.
Volume	✓	✓	1	Cell volume in μm.
Lengths uniform samples			64	Cell length in uniform directions.
Hops to Surface	✓	✓	1	Shortest path length on the graph between a cell and the surface. For this measure we ignore the geo-localization of nodes, and consider all neighbors one hop distant.
Degree Centrality	✓	✓	1	Degree centrality.
CFC centrality	✓	✓	1	Current-flow closeness centrality.

Table 1. Complete list of node features pre-computed in the CellTypeGraph.

Feature Name	Invariant	Default	Size	Description
Center of mass surface	✓	✓	3	Cell Surface (edge) center of mass represented in the global reference system, expressed in μm.
Center of mass distance	✓	✓	1	Distance between two adjacent cell center of mass.
Center of mass /GRS Proj.		✓	3	Angles between two adjacent cell center of mass direction and the global reference system.
LRS Proj.	✓	✓	3	Angles between local reference system in adjacent cells.
Surface	✓	✓	1	Edge surface area in μm.

Table 2. Complete list of edge features pre-computed in the CellTypeGraph.

node_grid_search

Model	Optimizer	Model Params.	top-1 acc.	class-avg. acc.
GIN [14]	lr $= 10^{-2}$ wd $= 10^{-5}$	# feat $= 64$ # layers $= 2$ dropout $= 0.1$	0.714 ± 0.071	0.563 ± 0.136
GCN [6]	lr $= 10^{-2}$ wd $= 10^{-5}$	# feat $= 128$ # layers $= 2$ dropout $= 0.5$	0.762 ± 0.043	0.617 ± 0.077
GAT [13]	lr $= 10^{-3}$ wd $= 0$	# feat $= 256$ # layers $= 2$ dropout $= 0.5$ heads $= 3$ concat. $=$ True	0.824 ± 0.033	0.705 ± 0.084
GATv2 [2]	lr $= 10^{-3}$ wd $= 10^{-5}$	# feat $= 256$ # layers $= 2$ dropout $= 0.5$ heads $= 3$ concat. $=$ True	0.855 ± 0.041	0.757 ± 0.087
GraphSAGE [5]	lr $= 10^{-3}$ wd $== 10^{-5}$	# feat $= 128$ # layers $= 4$ dropout $= 0.1$	0.859 ± 0.048	0.765 ± 0.093
GCNII [5]	lr $= 10^{-2}$ wd $= 10^{-5}$	# feat $= 128$ # layers $= 4$ dropout $= 0.0$ share weight $=$ False	0.863 ± 0.050	0.772 ± 0.100
Transf. GCN [10]	lr $= 10^{-3}$ wd $= 10^{-5}$	# feat $= 128$ # layers $= 2$ dropout $= 0.5$ heads $= 3$ concat. $=$ True	0.868 ± 0.045	0.779 ± 0.098
EdgeTransf. GCN [10]	lr $= 10^{-3}$ wd $= 0$	# feat $= 128$ # layers $= 2$ dropout $= 0.5$ heads $= 5$ concat. $=$ True	0.868 ± 0.044	0.777 ± 0.098
DeeperGCN [8]	lr $= 10^{-3}$ wd $= 0$	# feat $= 128$ # layers $= 32$ dropout $= 0.0$	$\mathbf{0.877 \pm 0.050}$	$\mathbf{0.796 \pm 0.098}$
EdgesDeeperGCN [8]	lr $= 10^{-3}$ wd $= 10^{-5}$	# feat $= 128$ # layers $= 16$ dropout $= 0.0$	$\mathbf{0.878 \pm 0.047}$	$\mathbf{0.797 \pm 0.095}$

Table 3. Best performing optimizer and model parameters according to the class-avg. accuracy.

5. Expert agreement

In order to highlights the most challenging aspect of our CellTypeGraph Benchmark, we here report further analysis of the expert biologist performance, see Fig. 7 and Fig. 8.

6. Data augmentation

Data augmentation is commonly used in machine learning to avoid overfitting in a small dataset and improve generalization. We tested the impact of two simple approaches

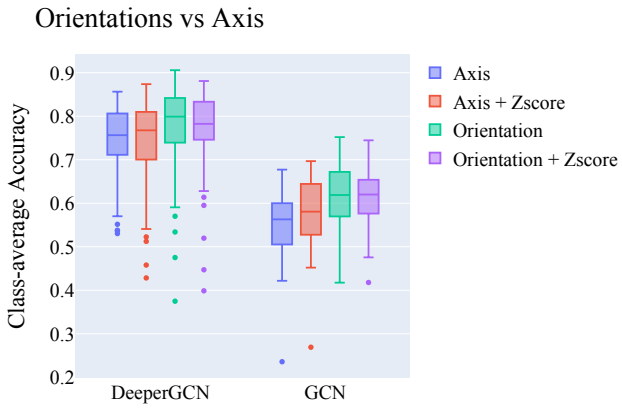

Figure 1. Class-average accuracy comparison between different vector features representations. Using the orientation instead of axis resulted in a small but consistent improvement in performance.

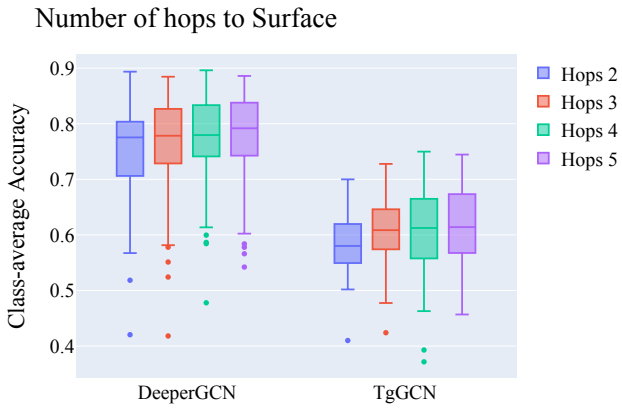

Figure 2. Class-average accuracy with varying maximum number of hops. Hops to surface are intuitively closely related to the tissue stratification *L1-L4*, but their relation loosen with depth. We tested how important this feature is by clipping its value. From the results one can see that the feature contribution saturates after three hops.

on our CellTypeGraph Benchmark. The first augmentation is to add Gaussian distributed noise to the node features, while the second approach is to use random dropout of edges in the cell adjacency graph, results are reported in Fig. 9. Our preliminary results show a small effect of data augmentation on the metrics, but more exhaustive experimentation is necessary to draw more solid conclusions.

7. Baseline implementation details

In our baseline, we benchmark eight different graph neural network architectures. We report here the most salient implementation details. For *GCN* [6], *GraphSAGE*, [5],

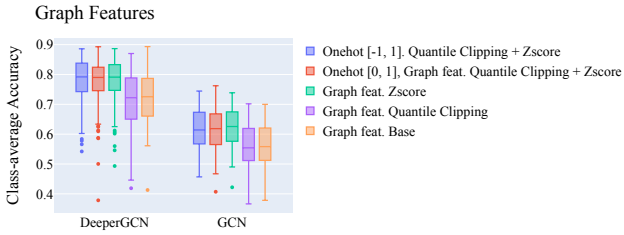

Figure 3. Class-average accuracy between different graph feature normalizations. One can see a slight accuracy improvement using z-score normalization.

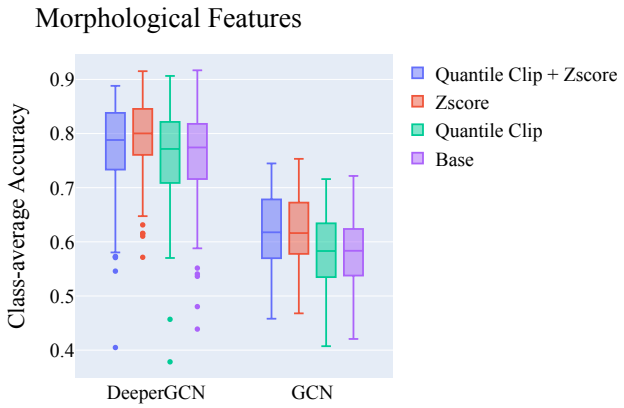

Figure 4. Class-average accuracy comparison between different morphological features normalizations. One can see a slight accuracy improvement using z-score normalization.

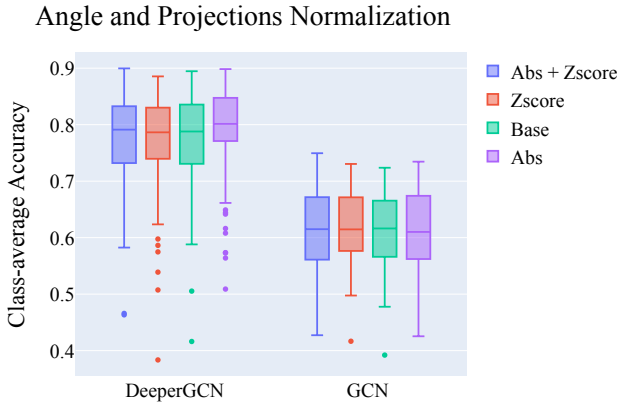

Figure 5. Class-average accuracy between different angles normalization. Angles are naturally normalized between -1 and 1, in our experiments the z-score had no significant impact.

GIN [14], *GCNII* [3], and *DeeperGCN* [7, 8]; we followed the *pytorch_geometric* implementation [4]. In all the afore-

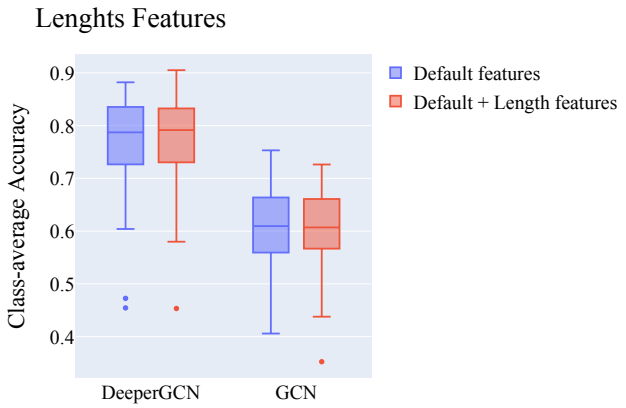

Figure 6. Class-average accuracy between: baseline features only, and baseline features plus additional lengths features. In our experiments the additional lengths showed no significant contribution to the accuracy.

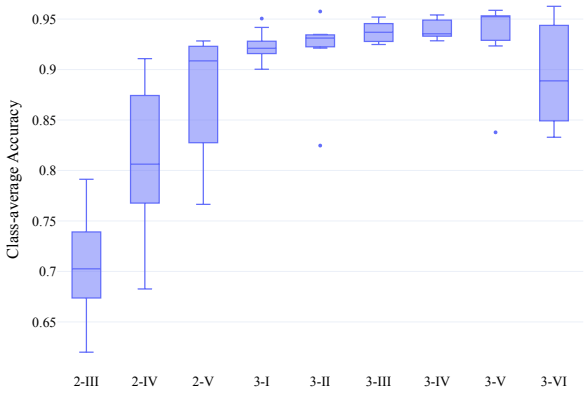

Figure 7. Class-average accuracy obtained by an expert biologist. The early stages pose the most substantial challenges, although the cause of such high variance can be attributed to the smaller number of cells for each specimen.

mentioned architectures the convolutions block are composed as follows: Graph Convolution → Relu activation [1] → normalization → Dropout [11]. The number of convolutions blocks used is an hyper parameter. In addition in *DeeperGCN* and *GCNII* and addition linear layer is added before the first graph convolution layer and after the last graph convolution layer. While for the remaining architectures *GAT* [13], *GATv2* [2], and *TransformerGCN* [10, 12], we used graph convolutions as implemented in [4] but with some minor differences in the convolution block layout: Graph Convolution → normalization → Relu activation [1] → Dropout [11]. All source implementation

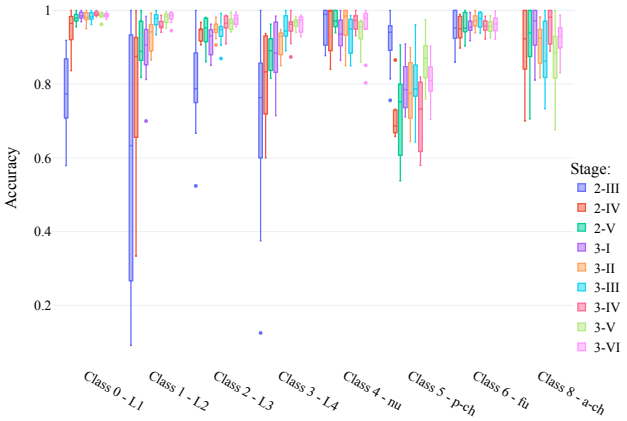

Figure 8. Per-class accuracy obtained by an expert biologist. In late stages the highest variability sources are the cell types *p-ch* and *p-ch*, while for early stages the highest variability is posed by the cell type *L1* to *L4*.

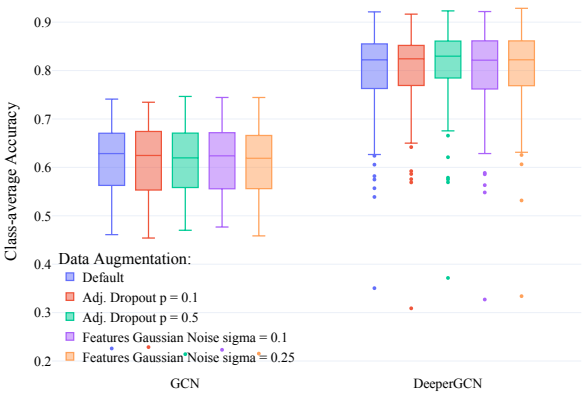

Figure 9. Comparison of the class-average accuracy for various combinations of network architecture, data augmentation technique, and parameters. The impact of data augmentation is negligible in all our experiments. Nevertheless, adjacency dropout (edge drop probability = 0.5) consistently improves accuracy for the DeeperGCN model.

are released at https://github.com/hci-unihd/plant-celltype/blob/main/plantcelltype/graphnn/graph_models.py.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 6

[2] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021. 4, 6

[3] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020. 5

[4] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 5, 6

[5] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017. 4, 5

[6] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017. 4, 5

[7] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can GCNs go as deep as CNNs? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9267–9276, 2019. 5

[8] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper GCNs. *arXiv preprint arXiv:2006.07739*, 2020. 4, 5

[9] Thorsten Schmidt, Taras Pasternak, Kun Liu, Thomas Blein, Dorothée Aubry-Hivet, Alexander Dovzhenko, Jasmin Duerr, William Teale, Franck A Ditengou, Hans Burkhardt, et al. The iRoCS Toolbox–3D analysis of the plant root apical meristem at cellular resolution. *The Plant Journal*, 77(5):806–814, 2014. 1

[10] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021. 4, 6

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 6

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 6

[13] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 4, 6

[14] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018. 4, 5