Dissertation

submitted to the

Combined Faculty of Natural Sciences and Mathematics

of Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

M.Sc. Lorenzo Cerrone

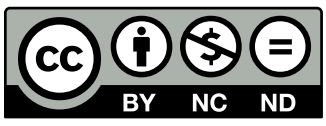
born in: Sora, Italy

Oral examination: 10.05.2023

Designing Deep Learning Frameworks for Plant Biology

Referees: Prof. Dr. Fred A. Hamprecht

Prof. Dr. Ulrich S. Schwarz



Abstract

In recent years the parallel progress in high-throughput microscopy and deep learning drastically widened the landscape of possible research avenues in life sciences. In particular, combining high-resolution microscopic images and automated imaging pipelines powered by deep learning dramatically reduced the manual annotation work required for quantitative analysis. In this work, we will present two deep learning frameworks tailored to the needs of life scientists in the context of plant biology.

First, we will introduce PlantSeg, a software for 2D and 3D instance segmentation. The PlantSeg pipeline contains several pre-trained models for different microscopy modalities and multiple popular graph-based instance segmentation algorithms.

In the second part, we will present CellTypeGraph, a benchmark for quantitatively evaluating graph neural networks. The benchmark is designed to test the ability of machine learning methods to classify the types of cells in an *Arabidopsis thaliana* ovules. CellTypeGraph's prime aim is to give a valuable tool to the geometric learning community, but at the same time it also offers a framework for plant biologists to perform fast and accurate cell type inference on new data.

Zusammenfassung

In den letzten Jahren haben die parallelen Fortschritte in der Hochdurchsatzmikroskopie und im Deep Learning die Landschaft der möglichen Forschungsrichtungen in den Biowissenschaften drastisch erweitert.

Insbesondere die Kombination von hochauflösenden mikroskopischen Bildern und automatisierten Bildgebungspipelines, die durch Deep Learning unterstützt werden, hat die manuelle Annotationsarbeit, die für die quantitative Analyse erforderlich ist, drastisch reduziert. In dieser Arbeit werden wir zwei Deep-Learning-Frameworks vorstellen, die auf die Bedürfnisse von Biowissenschaftlern im Kontext der Pflanzenbiologie zugeschnitten sind.

Zunächst stellen wir PlantSeg vor, eine Software für die Segmentierung von 2D- und 3D-Instanzen. Die PlantSeg-Pipeline enthält mehrere vortrainierte Modelle für verschiedene Mikroskopie-Modalitäten und mehrere populäre graphbasierte Algorithmen zur Instanzsegmentierung.

Im zweiten Teil stellen wir CellTypeGraph vor, einen Benchmark-Datensatz zur quantitativen Bewertung von graphischen neuronalen Netzen. Der Datensatz wurde entwickelt, um die Güte von Methoden des maschinellen Lernens zu testen, die Zelltypen in den Samenanlagen von *Arabidopsis thaliana* klassifizieren. Das Hauptziel von CellTypeGraph ist es, der Gemeinschaft des geometrischen Lernens ein wertvolles Werkzeug an die Hand zu geben, gleichzeitig bietet es aber auch einen Rahmen für Pflanzenbiologen zur schnellen und genauen Bestimmung von Zelltypen in neuer Datensätzen.

Acknowledgements

The first person I need to thank is my Ph.D. advisor Fred Hamprecht. He allowed me to grow as a scientist by giving me the freedom to decide my research direction and trust in my ideas. Thought my journey, he always gave me invaluable ideas, guidance, and encouragement. Working with such a brilliant scientist and a wonderful person has been a great honor, and I will always cherish the time I spent in the group.

During my six years in the Image Analysis and Learning group, I had the privilege to work with some of the most brilliant and kind people I have ever met. This work would not have been possible without the countless brainstorming, lunch discussions, and coffee breaks. In particular, my gratitude goes to Roman Remme, Sebastian Damnrich, Nasim Rahman, and Alberto Bailoni. They always listened to my rumbling ideas, answered my questions, and engaged in fruitful scientific discussions. Furthermore, I thank all current group members, Enrique Fita Sanmartin, Ocima Kamboj, Peter Lipmann, and old friends Thorsten Beier, Elke Kirschbaum, Manuel Haussmann, Steffen Wolff, Contastin Pape, and Moritz Plenz. Not only they all helped me in my work, but they all made our office a welcoming and friendly environment. A special thanks go to Barbara Werner. Her kind support and masterful knowledge of the University bureaucracy were priceless.

I would also thank all my collaborators. In particular, Adrian Wolny for his priceless contributions to PlantSeg, and for his friendship. Furthermore, I need also to thank Anna Kreshuk and Athul Vijayan for their essential contributions to my scientific work.

Finally, my thanks go to my family and friends for their unconditioned support. To Riccardo and Tasha for the love they give Whisky every day and for helping to proofread this manuscript. To my Family members my father Gilberto, my mum Mirella, my sister Emanuela (and her wonderful family Virginia, Vittoria, and Ulderico), Zia Vincenza, and my grandmas Idelma and Emma, to all of you, my most profound thank you for your love and support throughout my life. A big thanks also go to my dog Whisky. His unwavering happiness and affection brighten my life every day. My biggest thanks of all go to the love of my life Alessandra. She was always there to enrich my life, encourage me to believe in myself, and push me to become a better person. This thesis would not have been possible without her.

Contents

Chapter 1

Introduction

1.1 Background

In just a few years, machine learning has progressed from a niche tool, relegated to a handful of applications, to an unavoidable instrument deeply embedded in countless modern technologies, industries, and academic research.

This exponential growth is, on the one hand, due to technological advancements. Hardware progress has made massive parallelism readily accessible, first with Graphic Processing Units (GPUs) [1,2] and later with dedicated Ai accelerators like Google Tensor Processing Units (TPUs) [3]. In parallel, the massive availability of data allowed for the assembly of larger and more diverse datasets. In the span of two decades, we progressed from: the 70 thousand handwritten digits of the MNIST [4] dataset in 1998 to the 1.3 million images in 2012 for the ImageNet [5] classifications challenge for us to arrive at 3 billion images in [6] and other similarly sized datasets [7–9] used today to train staggering computer vision models [6,9,10].

On the other hand, the modern renaissance of machine learning is synonymous with the rise of Artificial Neural Networks (ANNs). While the concept has been known since 1958 [11], in recent years, we have pushed further and further the capability of ANNs. This advancement is the result of a frantic research effort to improve training techniques [12–14], find new computational tricks [15–18], and introduce deeper and more effective architectures [19–22].

Computer vision has been one of the first fields to profit from these machine learning advancements. In particular, Convolutional Neural Networks (CNNs) have revolutionized this field by outclassing any other machine learning model in several applications such as image classification [4,19,23], semantic segmentation [24–26], and video tracking [27,28]. Only in recent years, transformer based [22] architectures such as [23] are challenging this supremacy.

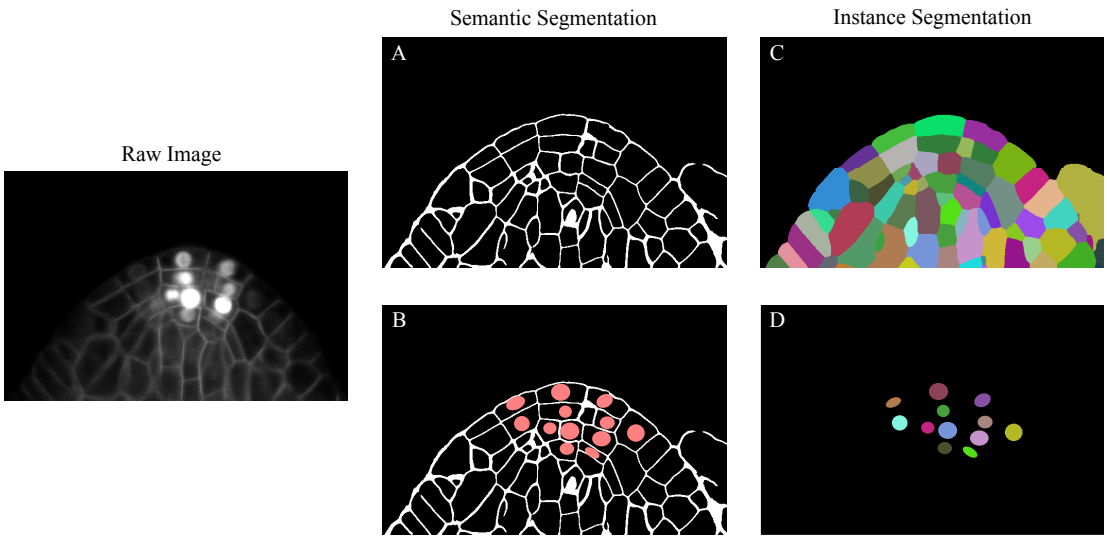


Figure 1.1. Segmentation tasks samples in a light sheet microscopy. In (A) and (B), one can see two examples of semantic segmentation, where the task is to classify each pixel into one of three different semantic classes: cell boundary, nucleus, and background. In (C) and (D), one can see two examples of instance segmentation, where the task is to assign a unique identifier to each categorical object in the scene, such as cells in (C) and nuclei in (D).

Today, the cumulative outcomes of all those innovations fall under the umbrella term of Deep Learning. The unreasonable effectiveness of this sub-field of machine learning has periodically shocked the scientific community and the general public with massive leaps in a wide spectrum of fields. From the superhuman performance in the game of Go [29], which recently lead to a breakthrough in protein folding [30] and matrix multiplication algorithms [31], and to the improvement in text and image generation [32–34] that are becoming less and less distinguishable from the creative works of humans minds.

1.2 Image segmentation

The term “image segmentation” is sometimes improperly used to describe the task of classifying pixels in a digital image and the task of grouping pixels in a digital image indiscriminately. These two tasks, although similar on the surface, are profoundly different. In this section, we will characterize their differences and cover the current state of research on this topic.

1.2.1 Semantic segmentation

We define semantic segmentation as the task of assigning each pixel in a digital image to a categorical class. An example from life sciences is distinguishing which pixels belongs to

the cell boundaries, which to the nuclei, and which to the background, Fig. 1.1 (A-B).

Early neural network approaches to semantic segmentation focused on generating dense pixel predictions by sliding a classification CNN over the image [35]. While these approaches were accurate, they were also computationally and memory inefficient. The introduction of fully convolutional networks [36] addressed these issues, and this style of neural networks has become the most prominent class of architecture for semantic segmentation.

A fully convolutional network generally consists of two parts: an encoder and a decoder. The encoder takes the source image as input and processes it by using a series of convolutional blocks and spatial pooling layers. Each block spatially condenses the low-level semantic information present in the image such as textures, boundaries, shapes etc. The decoder, instead, reverses this process and is tasked to upsample the encoder's output back to the original image resolution. A linear classifier usually maps the decoder's output to class probabilities.

Since its introduction, the most successful encoder-decoder style in life sciences applications has been the U-Net [24]. The U-Net distinguishes itself from other architectures by using skip-connections, that connect the hidden states at different resolution between the encoder and the decoder. See Fig. 1.2 for a characterization of the U-Net salient architectural components. Since its inception, several works have been built upon its original concept [37–40]. Moreover, similar encoder-decoder CNNs also found wide popularity in natural image and self-driving benchmarks [25, 41].

In chapter 2, we will see a practical application of the U-Net as a boundary predictor for plant tissue applications.

1.2.2 Instance segmentation

Instance segmentation is defined as the task of discriminating and segmenting each object (or instance in this context) in a digital image. An example of instance segmentation is shown in Fig. 1.1 (C-D), where each cell and nucleus is segmented with a unique identifier.

Instance segmentation is a fundamentally more challenging task compared to semantic segmentation. In particular, it presents two additional major obstacles: any permutation of the assignments is an equally optimal solution, and the number of instances present is not known a priori. Therefore, unlike semantic segmentation where one can use an end-to-end deep learning approach (as shown in subsection 1.2.1), for Instance segmentation a more complex pipeline is required. In recent years, we can distinguish three prominent approaches to this task.

Object detection: Object detection pipelines can be decomposed in two main steps: identify a region of interest for each object, usually a bounding box, and then complete the in-

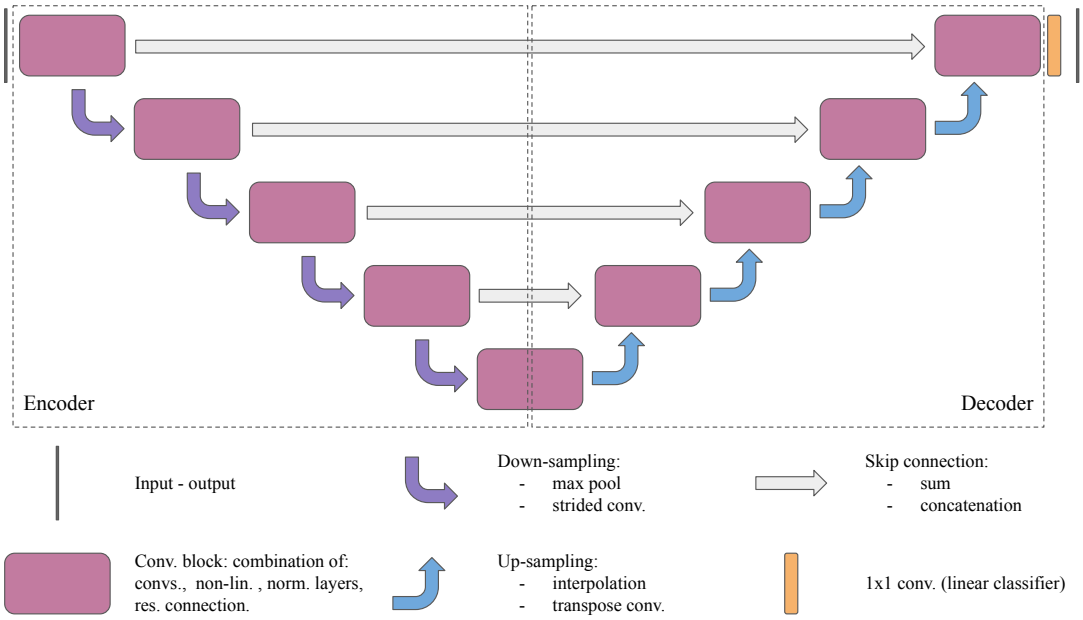


Figure 1.2. During the years since its introduction in [24], the implementation details of the U-Net style architecture have varied following the progress of deep learning research. Here is shown a diagram of a generic U-Net style architecture.

stance segmentation by computing a background-foreground mask for each detected object. This approach is the most common one for natural image applications. Among the most prolific neural network architectures belonging to this group of approaches, we can find region-based convolutional network (R-CNN) [42–44], and YOLO based models [45, 46].

For cell instance segmentation, a popular model is StartDist [47,48]. In StarDist, instead of predicting an axis-aligned bounding box for every pixel like in YOLO, we predict a star convex polygon that circumscribes the cell at that location. This approach is particularly well suited for convex cells of a regular shape. Moreover, using a polygon instead of a squared bounding box negates the need to compute a background-foreground mask, further simplifying the pipeline.

All the approaches above predict multiple detections for each instance and then refine the selection using non-maximum suppression. More recently, DETR [49], which is a transformers-based method, is emerging as an alternative joint detection and segmentation model. In DETR, the model learns implicitly to detect each instance once, thus avoiding non-maximum suppression and other computationally expensive heuristics.

Pixel embedding: The core idea is to produce a feature vector for each pixel, such that this vector embeds some relevant information for the segmentation task. A popular paradigm to produce such embedding is contrastive learning [50–52]. Where a deep learning model is trained with two goals, maximize the embedding similarity for pixels belonging to the

same instance and, at the same time, push apart embedding of pixels belonging to different instances. One can find several successful applications of contrastive learning to instance segmentation in the literature [53–56].

Another paradigm to train pixel embeddings is to regress to a handcrafted target embedding. A very successful instance segmentation pipeline is Cellpose [57], which is based on this approach. In Cellpose, the goal is to predict a diffusion flow direction to the center of the instance for each pixel. At inference time, a reverse-diffusion model is used to cluster pixels. Cellpose has a less strict shape prior compared to other models like StarDist. Thus it can work better on highly irregular and jagged cells.

Graph clustering: Modeling an image as a regular square lattice graph has been an essential fruitful paradigm in computer vision.

In Instance Segmentation, one can model each pixel in an image as a node in a square lattice and every two adjacent pixels as an edge. Moreover, one can describe the confidence that two pixels belong to the same instance by a scalar weight associated with each edge.

In this setup, the instance segmentation task is reduced to finding a robust and accurate estimator for our edge weights (usually a neural network [35, 58]) and finding a graph clustering algorithm robust to the inevitable flaws of the predicted weights. Which graph clustering algorithm should be selected depends on the quality of the predicted edge weights. If the weights are accurate, simpler clustering heuristics like hierarchical clustering [59, 60] or the Mutex Watershed [61, 62] can be used. In contrast, more complex clustering methods such as MultiCut [63–65] can be more robust to gaps in the detected boundaries. Solving the MultiCut problem is NP-hard [66], thus infeasible for even moderately small images. A common solution to mitigate this issue is to reduce the graph size by pre-computing superpixels (usually an over-segmentation of the image performed by Watershed-based pipeline [67]) and to solve the MultiCut problem approximately [68, 69].

This paradigm is particularly well suited for densely packed instance segmentation. The most prominent examples are the densely packed cells in some biological tissue, such as brain tissues or cells in plant tissues.

1.3 Geometric deep learning

So far, we have predominantly discussed computer vision applications of deep learning where the input domain is generally represented as a regular two- or three-dimensional tensor.

However, deep learning applies to a wider variety of applications and data domains such as graphs, sets, point clouds, and meshes. Geometric deep learning [70] proposes a common

framework to unify the independent development of machine learning in these domains and systematically derive the most popular neural network building blocks from first principles.

In this section, we will use the lens of geometric deep learning to introduce graph neural networks.

1.3.1 Invariance and equivariance

It is hard to overstate how robust and versatile human vision is. For example, if we are tasked to classify the subject of an image, we would all agree that shifting the image some pixels to the left or the right will not affect our ability to solve the task. Similarly, our vision is robust against many other transformations like reasonably changing the image resolution, flipping it horizontally, or slightly distorting it. These inductive biases are so intuitive to us but, at the same time, extremely hard to transfer to the digital world. In deep learning, this problem can be divided into two steps, finding the most fitting domain to represent our data and finding a suitable model that conforms to our inductive biases.

Before proceeding further, it is necessary to formalize our intuitive understanding of what it means to be robust to certain transformations of the input data. Let us define a function f as $f : \Omega \to \Omega'$, where Ω and Ω' are the domains of our model input/output, and let us also consider a group of transformations $\mathcal{G}$ whose elements $g \in \mathcal{G}$ are also functions in $g : \Omega \to \Omega$.

We define f to be invariant with respect to the grout of transforms $\mathcal{G}$ if

$$f(g(x)) = f(x) \quad \forall g \in \mathcal{G}. \tag{1.1}$$

In other words, we define invariance as the property of a function to remain unchanged under the action of a certain group of transforms on its input . For example, if we chose f as the euclidean distance between a point in $\mathbb{R}^2$ and the origin, we can see that f is invariant with respect to the rotation group $SO(2)$.

Similarly, we define f to be equivariant with respect to a group of transforms $\mathcal{G}$ if

$$f(g(x)) = g(f(x)) \quad \forall g \in \mathcal{G}. \tag{1.2}$$

For example, if f is a function that multiplies a point in $\mathbb{R}^2$ by a scalar, then f is equivariant with respect to the rotation group $SO(2)$.

For completeness of notation, equivariance is a special case of a more general property, covariance, where the elements of the transformation group act predictably on the input and output of a function, but not necessarily in the same way. More formally:

$$f(g(x)) = g'(f(x)) \tag{1.3}$$

where g and g' are two different actions of the same element of $\mathcal{G}$.

1.3.2 From image to graph data

In deep learning, inductive biases such as invariance and equivariance are crucial for a model's success. Let us, for instance, consider the task of image segmentation. In this task, the prediction of our model should be equivariant with respect to translation. Therefore, knowing that convolutions are shift equivariant [4, 71], it should not be surprising that CNNs are so successful in this task.

In recent years, several works have shown promising results in incorporating more and more inductive biases in their architecture [72–75]. Although the results are encouraging, particularly in parameter efficiency, those architectures have yet to become mainstream for computer vision applications. This slow adoption is partially due to other heuristic approaches often times being a practical and effective way to introduce inductive biases into a model without increasing the complexity of the architecture.

Simple strategies like data augmentation have shown to be very effective, particularly in image data. But, "learning" inductive biases from data requires paying a higher cost for the model size. Indeed, we need a significantly increase in number of parameters to include more and more inductive biases.

However, these heuristics are not feasible for all data domains. For example, in a graph, it is not possible to impose any intrinsic ordering of the nodes, regardless of the chosen representation. All permutations of the nodes are homeomorphic to each other. Therefore, in a graph $G(N, E)$ (where N is a set of nodes and E a set of edges between the nodes), there can be $|N|!$ equivalent permutations. The factorial growth of this problem makes it untractable even for a neural network with many parameters. Thus, for data domains like graphs, imposing strict geometric priors is necessary.

1.3.3 Graph neural networks

Graph neural networks (GNNs) are machine learning models that operate on graphs. Typical applications of these models vary from predicting molecular properties from the molecular graph [76–79], to various transductive tasks like label propagation in social network graphs or classifications in citation graphs [80–82]. For all the tasks mentioned above, the input data is represented as a graph $G(N, E)$ in which every node is associated with a feature vector x. The core assumption is that the graph structure itself stores essential information for solving the task. Thus, each GNN layer must extract not only information from the features but also from the graph structure itself.

Following the success of CNNs on image data, there have been several attempts to generalize convolutional layers for graph structures, either by using the graph laplacian spectrum [83–85] or spatial convolutions [86–88].

As shown in [89], the formalism of message passing can describe most of the proposed GNN layers. The message passing for GNNs can be formally written as:

$$x_i^k = \alpha^{(k-1)}\left(x_i^{(k-1)}, \Lambda_{j\in\mathcal{N}(i)}^{(k-1)}\beta^{(k-1)}\left(x_i^{(k-1)}, x_j^{(k-1)}\right)\right) \tag{1.4}$$

where x_i^k represents the feature vector of the node i at the layer k, β represents the *message* and can be any differentiable function, Λ can be any permutation invariant aggregation function over the neighbourhood $\mathcal{N}(i)$, and α can be any differentiable features *update* function.

As an example, the Graph Convolutional Network (GCN) [83] has been one of the most successful GNN architectures. A GCN layer is defined as

$$x_i^k = \text{Relu}\left(\sum_{j\in\mathcal{N}(i)} \frac{W\cdot x_j^{(k-1)}}{\sqrt{deg(i)\cdot deg(j)}} + b\right) \tag{1.5}$$

where the *message* function is a linear transformation W of the source node features x_j (symmetrically normalized by the square root of the degrees $deg(i)\cdot deg(j)$), the aggregation function is simply the sum operation, and the *update* function adds a learned bias b and applies the Relu non-linearity.

Inspired by the success of the GCN, several variations of graph convolutions layer have been proposed [86, 87, 90–95]. However, compared to other deep learning domains, GNN architecture search is still a very active research field. This is because current GNNs performance is hindered by a series of theoretical and experimental issues like over-smoothing [96–98], the bottleneck effect [99, 100], and more [101–103].

1.4 Deep learning applications for plant biology

Microscopy has played a foundational role in biology. Since the pioneering works of Antonie van Leeuwenhoek in the 18th century, the evolution of microscopy has been closely related to the scientific progress of biology. Today state-of-the-art light-sheet and confocal microscopes are engineering marvels that allow researchers to acquire terabytes of volumetric images of larger and larger plant and animal tissues.

Among many other areas of research, this microscopy progress immensely contributes to the study of plant morphogenesis [104–107] where biologists are interested in unveiling the underlying mechanism of how plant tissues and organs form.

However, this massive stream of raw imaging data requires complex processing before being analyzed and used for quantitative research. In this work, we will show how deep learning impacted two of these processing steps: cell instance segmentation and cell type classification.

1.4.1 Cell instance segmentation in plant tissues

Fluorescence light microscopy is widely used for the imaging of plant tissues. In this type of microscopy, the tissue undergoes a staining process, which targets specific locations (for example, cell membranes or nuclei) or specific molecules.

In chapter 2, we will introduce PlantSeg, a software framework developed to perform instance segmentation from cell membrane staining images. The tools follow the paradigm introduced in subsection 1.2.2 of the instance segmentation as graph clustering. The pipeline comprises two steps: a U-Net [24] boundaries classifier and multiple popular graph-based image clustering algorithms.

PlantSeg comes with several pre-trained models. These have been trained on datasets from different modalities (light-sheet and confocal) and at different voxels resolutions. Finally, in subsection 2.2.2, we will show the effectiveness of the PlantSeg models on a wide variety of external biological benchmarks.

1.4.2 Semantic segmentation as graph labeling

Cell type classification is a widely used analysis tool in several biological research areas. Cells can be grouped into different types depending on their properties. Cells of the same type can share a number of characteristics which are either morphological, positional, functional, and/or phenotypical.

In the context of plant morphogenesis, researchers are interested in the automatic classification of tissues for plant organs. This task, on the surface, is a straightforward semantic segmentation problem. But, approaching the problem from this perspective is extremely challenging, see Fig. 1.3.

In chapter 3, we will show how we tackled the problem from an alternative point of view. We mapped the semantic segmentation task to a geo-referenced graph labeling task. Every node in the graph represents a cell, and every neighboring cell is connected with an edge. Moreover, out of this task, we built an open benchmark for the geometric deep learning community and evaluated the performance of several state-of-the-art GNNs models on it.

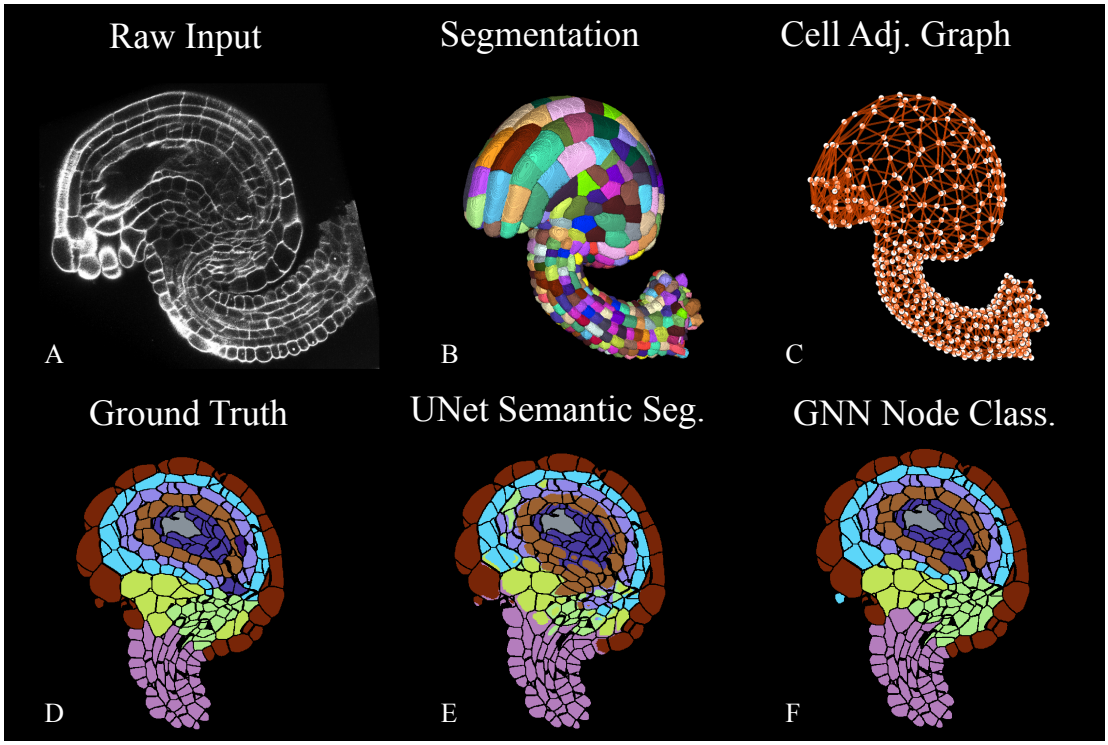


Figure 1.3. Celltype graph benchmark overview. In the top row, we can see a raw image of an *Arabidopsis thaliana* ovule (A), a cell instance segmentation for this organ is performed using PlantSeg (B), and lastly, we compute the cell adjacency graph from the segmentation (C). In the bottom row, one can see qualitatively how a U-Net model (E) struggles to predict certain cell types annotations from an expert biologist (D). Meanwhile, the GNN model (F) is able to learn the layering structure of the biological cell type.

1.5 List of publications

This thesis is based on the following publications:

- Accurate and versatile 3D segmentation of plant tissues at cellular resolution. Adrian Wolny[1], Lorenzo Cerrone [1], Athul Vijayan, Rachele Tofanelli, Amaya Vilches Barro, Marion Louveaux, Christian Wenzl, Sören Strauss, David Wilson-Sánchez, Rena Lymbouridou, Susanne S. Steigleder, Constantin Pape, Alberto Bailoni, Salva Duran-Nebreda, George W. Bassel, Jan U. Lohmann, Miltos Tsiantis, Fred A. Hamprecht, Kay Schneitz, Alexis Maizel, Anna Kreshuk (2020). *Elife* [108].

- CellTypeGraph: A New Geometric Computer Vision Benchmark. Lorenzo Cerrone, Athul Vijayan, Tejasvinee Mody, Kay Schneitz, Fred A. Hamprecht (2022). *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* [109].

Complete list of publications to which the author contributed:

- End-to-end learned random walker for seeded image segmentation. Lorenzo Cerrone, Alexander Zeilmann, Fred A. Hamprecht (2019). *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* [110].

- A digital 3D reference atlas reveals cellular growth patterns shaping the Arabidopsis ovule. Athul Vijayan, Rachele Tofanelli, Sören Strauss, Lorenzo Cerrone, Adrian Wolny, Joanna Strohmeier, Anna Kreshuk, Fred A. Hamprecht, Richard S. Smith, Kay Schneitz (2021), *Elife* [111].

- Microscopy-based assay for semi-quantitative detection of SARS-CoV-2 specific antibodies in human sera. Constantin Pape, Roman Remme, Adrian Wolny, Sylvia Olberg, Steffen Wolf, Lorenzo Cerrone, Mirko Cortese, Severina Klaus, Bojana Lucic, Stephanie Ullrich, Maria Anders-Össwein, Stefanie Wolf, Berati Cerikan, Christopher J. Neufeldt, Markus Ganter, Paul Schnitzler, Uta Merle, Marina Lusic, Steeve Boulant, Megan Stanifer, Ralf Bartenschlager, Fred A. Hamprecht, Anna Kreshuk, Christian Tischer, Hans-Georg Kräusslich, Barbara Mueller, Vibor Laketa (2021). *Bioessays* [112].

- Integration of Cell Growth and Asymmetric Division during Lateral Root Initiation in Arabidopsis thaliana. Lilli Marie Schütz, Marion Louveaux, Amaya Vilches Barro, Sami Bouziri, Lorenzo Cerrone, Adrian Wolny, Anna Kreshuk, Fred A. Hamprecht, Alexis Maizel (2021). *Plant and Cell Physiology* [113].

[1] Equal contribution.

1.5.1 Detailed list of contributions

This section will provide a detailed account of the author's original contributions.

Chapter 2 is derived from the publication *Accurate and versatile 3D segmentation of plant tissues at cellular resolution* [108]. I contributed equally with Adrian Wolny to the conceptualization, writing, computational experiments, and the analysis of the results described in [108]. I am responsible for the data curation, CNN training, and segmentation benchmarking of the *Arabidopsis thaliana* ovules dataset and responsible for the experiments described in section subsection 2.2.2. Adrian Wolny is responsible for the data curation, CNN training, and segmentation benchmarking of the *Arabidopsis thaliana* lateral root primordia dataset and responsible for the experiments described in subsection 2.2.3 and subsection 2.2.4.

The main contribution of chapter 2 is the software PlantSeg. Me and Adrian Wolny jointly implemented and designed the core workflow handler, the evaluation code, and the project packaging and deployment. Moreover, both authors have provided improvements, new features, and bug fixes across the entire code base. I am the main contributor to the segmentation workflows (DT-Watershed, GASP, MutexWS, MultiCut), data pre-processing and post-processing, command line interface, graphical user interface, automated and manual segmentation proofreading tools, integration with the Napari viewer, and the dynamic headless workflows. Adrian Wolny mainly contributed to the CNN implementation, training infrastructure, automated testing, and Lifted MultiCut workflow. For a more atomized breakdown of the contributions, the project's GitHub page (`https://github.com/hci-unihd/plant-seg`) contains a line-by-line history of edits.

Chapter 3 is derived from the publication *CellTypeGraph: A New Geometric Computer Vision Benchmark* [109]. My contributions lie in conceptualizing the project and methodology, dataset curation, developing all software (the benchmark, training, validation, evaluation, and additional experiments), formal analysis of the results, generating figures, writing, and supervising the co-authors in the ground truth label creation.

Chapter 2

PlantSeg: a tool for cell instance segmentation

Quantitative analysis of plant and animal morphogenesis requires accurate segmentation of individual cells in volumetric images of growing organs. In the last years, deep learning has provided robust automated algorithms that approach human performance, with applications to bio-image analysis now starting to emerge. Here, we present PlantSeg, a pipeline for volumetric segmentation of plant tissues into cells. PlantSeg employs a convolutional neural network to predict cell boundaries and graph partitioning to segment cells based on the neural network predictions. PlantSeg was trained on fixed and live plant organs imaged with confocal and light sheet microscopes. PlantSeg delivers accurate results and generalizes well across different tissues, scales, acquisition settings even on non plant samples. We present results of PlantSeg applications in diverse developmental contexts. PlantSeg is free and open-source, with both a command line and a user-friendly graphical interface (`https://github.com/hci-unihd/plant-seg`).

2.1 Introduction

Large-scale quantitative study of morphogenesis in a multicellular organism entails an accurate estimation of the shape of all cells across multiple specimen. State-of-the-art light microscopes allow for such analysis by capturing the anatomy and development of plants and animals in terabytes of high-resolution volumetric images. With such microscopes now in routine use, segmentation of the resulting images has become a major bottleneck in the downstream analysis of large-scale imaging experiments.

Here, we present PlantSeg, a deep learning-based pipeline for volumetric instance segmentation of dense plant tissues at single-cell resolution. PlantSeg processes the output from the microscope with a CNN to produce an accurate prediction of cell boundaries. Building

on the insights from previous work on cell segmentation in electron microscopy volumes of neural tissue [65, 114], the second step of the pipeline delivers an accurate segmentation by solving a graph partitioning problem. We trained PlantSeg on 3D confocal images of fixed *Arabidopsis thaliana* ovules and 3D+t light sheet microscope images of developing lateral roots, two standard imaging modalities in the studies of plant morphogenesis. We investigated a range of network architectures and graph partitioning algorithms and selected the ones which performed best with regard to extensive manually annotated ground truth. We benchmarked PlantSeg on a variety of datasets covering a range of samples and image resolutions. Overall, PlantSeg delivers excellent results on unseen data and, as we show through quantitative and qualitative evaluation, even non-plant datasets do not necessarily require network retraining. Combining the repository of accurate neural networks trained on the two common microscope modalities and going beyond just thresholding or watershed with robust graph partitioning strategies is the main strength of our package. PlantSeg is an open-source tool which contains the complete pipeline for segmenting large volumes. Each step of the pipeline can be adjusted via a convenient graphical user interface while expert users can modify configuration files and run PlantSeg from the command line. Users can also provide their own pre-trained networks for the first step of the pipeline using a popular 3D U-Net implementation (`https://github.com/wolny/pytorch-3dunet`), which was developed as a part of this project. Although PlantSeg was designed to segment 3D images, one can directly use it to segment 2D stacks. Besides the tool itself, we provide all the networks we trained for the confocal and light sheet modalities at different resolution levels and make all our training and validation data publicly available [1].

2.1.1 Related works

A few segmentation pipelines have been proposed [115, 116], but these either do not leverage recent developments in the field of computer vision or are difficult to use for non-experts.

With a few notable exceptions, such as the Brainbow experiments [117], imaging cell shape during morphogenesis relies on staining of the plasma membrane with a fluorescent marker. Segmentation of cells is then performed based on their boundary prediction. In the early days of computer vision, boundaries were usually found by edge detection algorithms [118]. More recently, a combination of edge detectors and other image filters was commonly used as input for a machine learning algorithm, trained to detect boundaries [119]. Currently, the most powerful boundary detectors are based on Convolutional Neural Networks (CNNs) [36, 120, 121]. In particular, the U-Net architecture [24] has demonstrated excellent performance on 2D biomedical images and has later been further extended to pro-

[1] All datasets used to support the findings of this study have been deposited in `https://osf.io/uzq3w`

cess volumetric data [37].

Once the boundaries are found, other pixels need to be grouped into objects delineated by the detected boundaries. For noisy, real-world microscopy data, this post-processing step still represents a challenge and has attracted a fair amount of attention from the computer vision community [61, 65, 114, 122, 123]. If centroids ("seeds") of the objects are known or can be learned, the problem can be solved by the watershed algorithm [110, 124]. For example, in [125] a 3D U-Net was trained to predict cell contours together with cell centroids as seeds for watershed in 3D confocal microscopy images. This method, however, suffers from the usual drawback of the watershed algorithm: misclassification of a single cell centroid results in sub-optimal seeding and leads to segmentation errors.

Recently an approach combining the output of two neural networks and watershed to detect individual cells showed promising results on segmentation of cells in 2D [126]. Although this method can in principle be generalized to 3D images, the necessity to train two separate networks poses additional difficulty for non-experts.

While deep learning-based methods define the state-of-the-art for all image segmentation problems, only a handful of software packages strives to make them accessible to non-expert users in biology (reviewed in [127]). Notably, the U-Net segmentation plugin for ImageJ [128] conveniently exposes U-Net predictions and computes the final segmentation from simple thresholding of the probability maps. CDeep3M [129] and DeepCell [130] enable, via the command-line, the thresholding of the probability maps given by the network, and DeepCell allows instance segmentation as described in [126]. More advanced post-processing methods are provided by the ilastik Multicut workflow [131], however, these are not integrated with CNN-based prediction.

2.2 PlantSeg

2.2.1 A pipeline for segmentation of plant tissues into cells

The segmentation algorithm we propose contains two major steps. In the first step, a fully convolutional neural network (a variant of U-Net) is trained to predict cell boundaries. Afterwards, a region adjacency graph is constructed from the pixels with edge weights computed from the boundary predictions. In the second step, the final segmentation is computed as a partitioning of this graph into an unknown number of objects (see Fig. 2.1). Our choice of graph partitioning as the second step is inspired by a body of work on segmentation for nanoscale connectomics (segmentation of cells in electron microscopy images of neural tissue), where such methods have been shown to outperform more simple post-processing of the boundary maps [65, 114, 132].

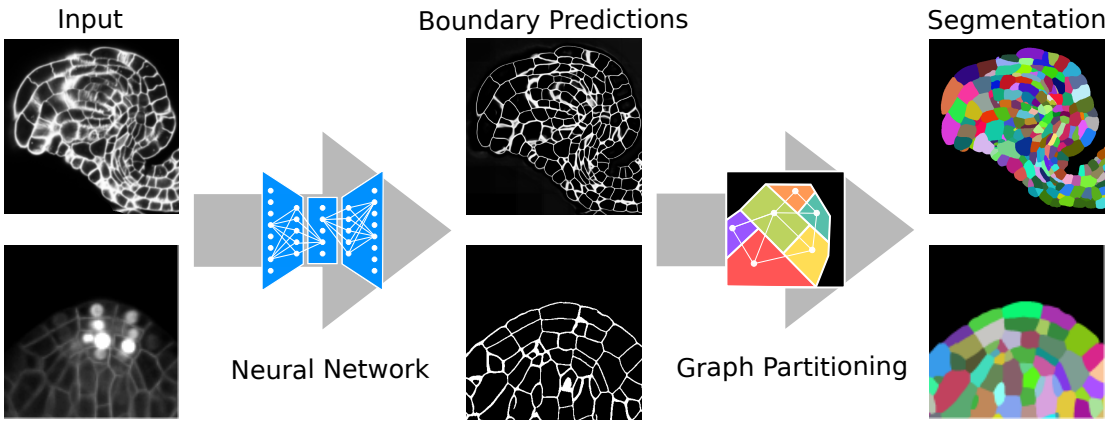


Figure 2.1. Segmentation of plant tissues into cells using PlantSeg. First, PlantSeg uses a 3D UNet neural network to predict the boundaries between cells. Second, a volume partitioning algorithm is applied to segment each cell based on the predicted boundaries. The neural networks were trained on ovules (top, confocal laser scanning microscopy) and lateral root primordia (bottom, light sheet microscopy) of *Arabidopsis thaliana*.

Step 1: cell boundary detection

Being the current state of the art in bioimage segmentation, U-Net [24] was chosen as the base model architecture for predicting the boundaries between cells. Aiming for the best performance across different microscope modalities, we explored various components of neural network design critical for improved generalization and robustness to noise, namely: the network architecture, loss function, normalization layers and size of patches used for training. For the final PlantSeg package we trained one set of CNNs for each dataset as the ovule and lateral root datasets are substantially different.

In more detail, with regard to the network architecture we compared the regular 3D U-Net as described in [37] with a Residual U-Net from [38]. We tested two loss functions commonly used for the semantic segmentation task: binary cross-entropy (BCE) ($\mathcal{L}_{BCE}$) [24] and Dice loss ($\mathcal{L}_{Dice}$) [133], as well as their linear combination ($\alpha\mathcal{L}_{BCE} + \beta\mathcal{L}_{Dice}$) termed BCE-Dice. The patch size and normalization layers were investigated jointly by comparing training on a single large patch, versus training on multiple smaller patches per network iteration. For single patch we used group normalization [134] whereas standard batch normalization [135] was used for the multiple patches.

In the ovule dataset, 39 stacks were randomly selected for training, two for validation and seven for testing. In the LRP dataset, 27 time points were randomly selected from the three videos for training, two time points were used for validation and four for testing.

The best performing CNN architectures and training procedures is illustrated by the precision/recall curves evaluated at different threshold levels of the predicted boundary proba-

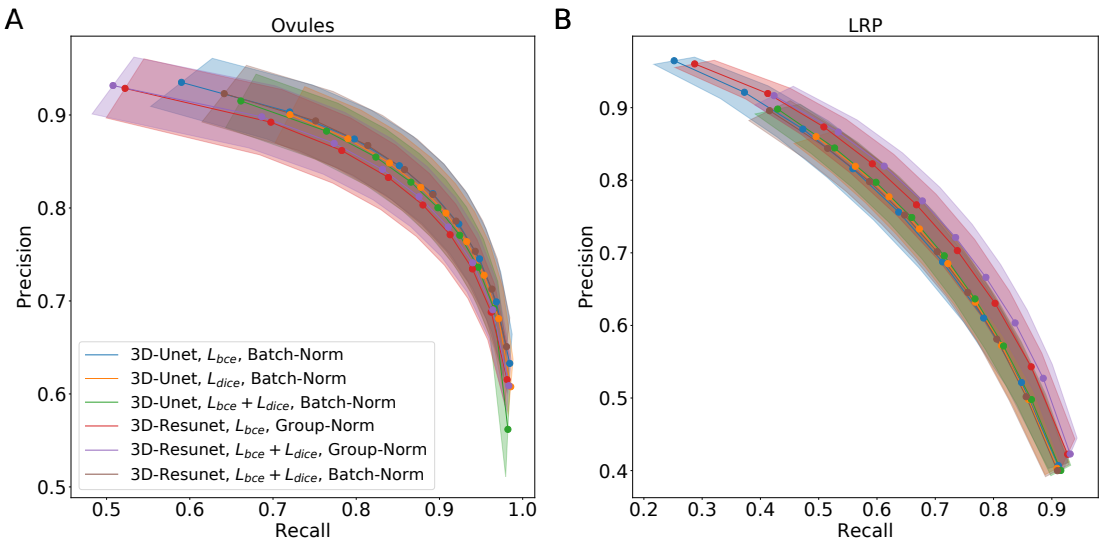


Figure 2.2. Precision-recall curves for different CNN variants on the ovule (A) and lateral root primordia (LRP) (B) datasets. Six training procedures that sample different type of architecture (3D U-Net *vs.* 3D Residual U-Net), loss function (BCE *vs.* Dice *vs.* BCE-Dice) and normalization (Group-Norm *vs.* Batch-Norm) are shown. Those variants were chosen based on the accuracy of boundary prediction task: 3 best performing models on the ovule and 3 best performing models on the lateral root datasets (see Appendix 5, Table 1 for a detailed summary). Points correspond to averages of seven (ovules) and four (LRP) values and the shaded area represent the standard error. For a detailed overview of precision-recall curves on individual stacks we refer to Appendix 4, Figure 1. Source files used to generate the plot are available in the Figure 2-source data 1.

bility maps (see Fig. 2.2). Training with a combination of binary cross-entropy and Dice loss ($L_{bce} + L_{dice}$) performed best on average across the two datasets in question contributing to 3 out of 6 best performing network variants. BCE-Dice loss also generalized well on the out of sample data described in Sec. 2.2.2. Due to the regularity of cell shapes, the networks do not benefit from broader spatial context when only cell membrane signal is present in input images. Indeed, training the networks with bigger patch sizes does not noticeably increase the performance as compared to training with smaller patches. 4 out of 6 best performing networks use smaller patches and batch normalization (*Batch-Norm*) whereas only 2 out of 6 use bigger patches and group normalization (*Group-Norm*). Residual U-Net architecture (*3D-ResUnet*) performed best on the LRP dataset (Fig. 2.2 (B)), whereas standard U-Net architecture (*3D-Unet*) was better on the ovule datasets (Fig. 2.2 (A)). For a complete overview of the performance of investigated models see also Fig. A.1 and Tab. A.2. In conclusion, choosing the right loss function and normalization layers increased the final performance on the task of boundary prediction on both microscope modalities.

Step 2: segmentation of tissues into cells using graph partitioning

After the cell boundaries are predicted, segmentation of the cells can be formulated as a generic graph partitioning problem, where the graph is built as the region adjacency graph of the image voxels. However, solving the partitioning problem directly at voxel-level is computationally expensive for volumes of biologically relevant size. To make the computation tractable, we first cluster the voxels into so-called supervoxels by running a watershed algorithm on the distance transform of the boundary map, seeded by all local maxima of the distance transform smoothed by a Gaussian blur. The region adjacency graph is then built directly on supervoxels and partitioned into an unknown number of segments to deliver a segmentation. We tested four different partitioning strategies: Multicut [63], hierarchical agglomeration as implemented in GASP average (GASP) [60], Mutex watershed (Mutex) [61] as well as the distance transform (DT) watershed [67] as a baseline since similar methods have been proposed previously [125, 126].

To quantify the accuracy of the four segmentation strategies we use Adapted Rand error (ARand) for the overall segmentation quality and two other metrics based on the variation of information [136] (see section Sec. 2.3.3), measuring the tendency to over-split (VOI_{split}) or over-merge (VOI_{merge}). GASP, Multicut and Mutex watershed consistently produced accurate segmentation on both datasets with low ARand errors and low rates of merge and split errors (Fig. 2.3A-C and Tab. A.1). As expected DT watershed tends to over-segment with higher split error and resulting higher ARand error. Multicut solves the graph partitioning problem in a globally optimal way and is therefore expected to perform better compared to greedy algorithms such as GASP and Mutex watershed. However, in our case the gain was marginal, probably due to the high quality of the boundary predictions.

The performance of PlantSeg was also assessed qualitatively by expert biologists. The segmentation quality for both datasets is very satisfactory. For example in the lateral root dataset, even in cases where the boundary appeared masked by the high brightness of the nuclear signal, the network correctly detected it and separated the two cells (Fig. 2.3D, green box). On the ovule dataset, the network is able to detect weak boundaries and correctly separate cells in regions where the human expert fails (Fig. 2.3E , green box). The main mode of error identified in the lateral root dataset is due to the ability of the network to remove the nuclear signal which can weaken or remove part of the adjacent boundary signal leading to missing or blurry cell contour. For example, the weak signal of a newly formed cell wall close to two nuclei was not detected by the network and the cells were merged (Fig. 2.3D, red box). For the ovule dataset, in rare cases of very weak boundary signal, failure to correctly separate cells could also be observed (Fig. 2.3E, red box).

Taken together, our analysis shows that segmentation of plant tissue using graph partitioning handles robustly boundary discontinuities present in plant tissue segmentation prob-

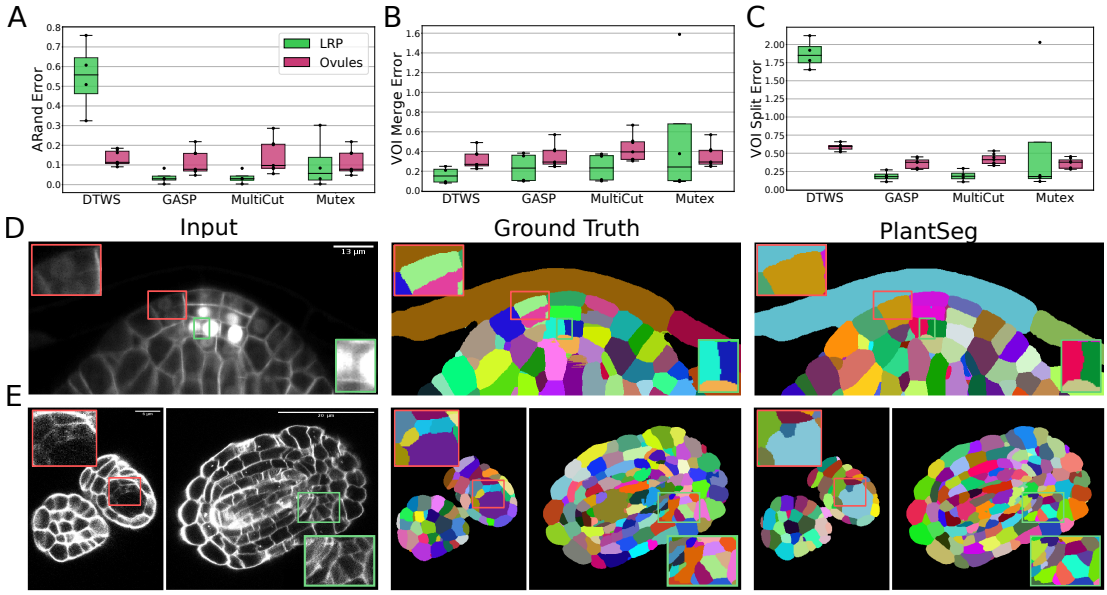


Figure 2.3. Segmentation using graph partitioning. (A-C) Quantification of segmentation produced by Multicut, GASP, Mutex watershed (Mutex) and DT watershed (DT WS) partitioning strategies. The Adapted Rand error (A) assesses the overall segmentation quality whereas $\text{VOI}_{\text{merge}}$ (B) and $\text{VOI}_{\text{split}}$ (C) assess erroneous merge and splitting events (lower is better). Box plots represent the distribution of values for seven (ovule, magenta) and four (LRP, green) samples. (D, E) Examples of segmentation obtained with PlantSeg on the lateral root (D) and ovule (E) datasets. Green boxes highlight cases where PlantSeg resolves difficult cases whereas red ones highlight errors. We obtained the boundary predictions using the *generic-confocal-3d-unet* for the ovules dataset and the *generic-lightsheet-3d-unet* for the root. All agglomerations have been performed with default parameters. 3D superpixels instead of 2D superpixels were used. Source files used to create quantitative results shown in (A-C) are available in the Figure 3-source data 1.

lems.

Datasets

To make our tool as generic as possible, we used both fixed and live samples as core datasets for design, validation and testing. Two microscope modalities common in studies of morphogenesis were employed, followed by manual and semi-automated annotation of ground truth segmentation.

The first dataset consists of fixed *Arabidopsis thaliana* ovules at all developmental stages acquired by confocal laser scanning microscopy with a voxel size of $0.075 \times 0.075 \times 0.235$ μm. 48 volumetric stacks with hand-curated ground truth segmentation were used. A complete description of the image acquisition settings and the ground truth creation protocol

is reported in [137].

The second dataset consists of three time-lapse videos showing the development of *Arabidopsis thaliana* lateral root primordia (LRP). Each recording was obtained by imaging wild-type Arabidopsis plants expressing markers for the plasma membrane and the nuclei [138] using a light sheet fluorescence microscope (LSFM). Stacks of images were acquired every 30 minutes with constant settings across movies and time points, with a voxel size of $0.1625 \times 0.1625 \times 0.250$ μm. The first movie consists of 52 time points of size 2048 $\times$ 1050 $\times$ 486 voxels. The second movie consists of 90 time points of size 1940 $\times$ 1396 $\times$ 403 voxels and the third one of 92 time points of size 2048 $\times$ 1195 $\times$ 566 voxels. The ground truth was generated for 27 images depicting different developmental stages of LRP coming from the three movies (see Sec. 2.3.4).

The two datasets were acquired on different types of microscopes and differ in image quality. To quantify the differences we used the peak signal-to-noise (PSNR) and the structural similarity index measure (SSIM) [139]. We computed both metrics using the input images and their ground truth boundary masks; higher values show better quality. The average PSNR measured on the light sheet dataset was 22.5 ± 6.5 dB (average ± SD), 3.4 dB lower than the average PSNR computed on the confocal dataset (25.9 ± 5.7). Similarly, the average SSIM is 0.53 ± 0.12 for the light sheet, 0.1 lower than 0.63 ± 0.13 value measured on the confocal images. Both datasets thus contain a significant amount of noise. LSFM images are noisier and more difficult to segment, not only because of the noise, but also due to part of nuclear labels being in the same channel as membrane staining.

In the following we describe in detail the design and performance of each of the two steps of the pipeline.

2.2.2 Performance on external plant datasets

To test the generalization capacity of PlantSeg, we assessed its performance on data for which no network training was performed. To this end, we took advantage of the two publicly available datasets: Arabidopsis 3D Digital Tissue Atlas (`https://osf.io/fzr56`) composed of eight stacks of eight different *Arabidopsis thaliana* organs with hand-curated ground truth [140], as well as the developing leaf of the Arabidopsis [141] with 3D segmentation given by the SimpleITK package [142]. The input images from the digital tissue atlas are confocal stacks of fixed tissue with stained cell contours and thus similar to the images of the Arabidopsis ovules, whereas the images of the leaf were acquired through the use of live confocal imaging. It's important to note that the latter image stacks contain highly lobed epidermal cells, which are difficult to segment with classical watershed-based algorithms. We fed the confocal stacks to PlantSeg and qualitatively assessed the resulting segmentation.

Dataset	PlantSeg (default parameters)			PlantSeg (tuned parameters)		
	ARand	VOI_{split}	VOI_{merge}	ARand	VOI_{split}	VOI_{merge}
Anther	0.328	0.778	0.688	0.167	0.787	0.399
Filament	0.576	1.001	1.378	0.171	0.687	0.487
Leaf	0.075	0.353	0.322	0.080	0.308	0.220
Pedicel	0.400	0.787	0.869	0.314	0.845	0.604
Root	0.248	0.634	0.882	0.101	0.356	0.412
Sepal	0.527	0.746	1.032	0.257	0.690	0.966
Valve	0.572	0.821	1.315	0.300	0.494	0.875

Table 2.1. Quantification of PlantSeg performance on the 3D Digital Tissue Atlas, using PlantSeg . The Adapted Rand error (ARand) assesses the overall segmentation quality whereas VOI_{merge} and VOI_{split} assess erroneous merge and splitting events. The petal images were not included in our analysis as they are very similar to the leaf and the ground truth is fragmented, making it difficult to evaluate the results from the pipeline in a reproducible way. Segmented images are computed using GASP partitioning with default parameters (left table) and fine-tuned parameters described in Appendix Appendix A.4 (right table).

Quantitative assessment was performed only for the digital tissue atlas, where the ground truth labels are available.

Qualitatively, PlantSeg performed well on both datasets, giving satisfactory results on all organs from the 3D Digital Tissue Atlas, correctly segmenting even the oval non-touching cells of the anther and leaf: a cell shape not present in the training data (Fig. 2.4). Our pipeline yielded especially good segmentation results when applied to the complex epidermal cells, visibly outperforming the results obtained using the SimpleITK framework (Fig. 2.5).

Quantitatively, the performance of PlantSeg out of the box (default parameters) on the 3D Digital Tissue Atlas is on par with the scores reported on the LRP and ovules datasets on the anther, leaf, and the root, but lower for the other tissues (Tab. 2.1, left). Default parameters have been chosen to deliver good results on most type of data, however we show that a substantial improvement can be obtained by parameter tuning (see Appendix Appendix A.3 for an overview of the pipeline's hyperparameters and Appendix A.4 for a detailed guide on empirical parameter tuning), in case of the tissue 3D Digital Tissue Atlas tuning improved segmentation by a factor of two as measured with the ARand error (Tab. 2.1, right). It should be noted that the ground truth included in the dataset was created for analysis of the cellular connectivity network, with portions of the volumes missing or having low quality ground truth (see e.g filament and sepal in Fig. 2.4). For this reason, the

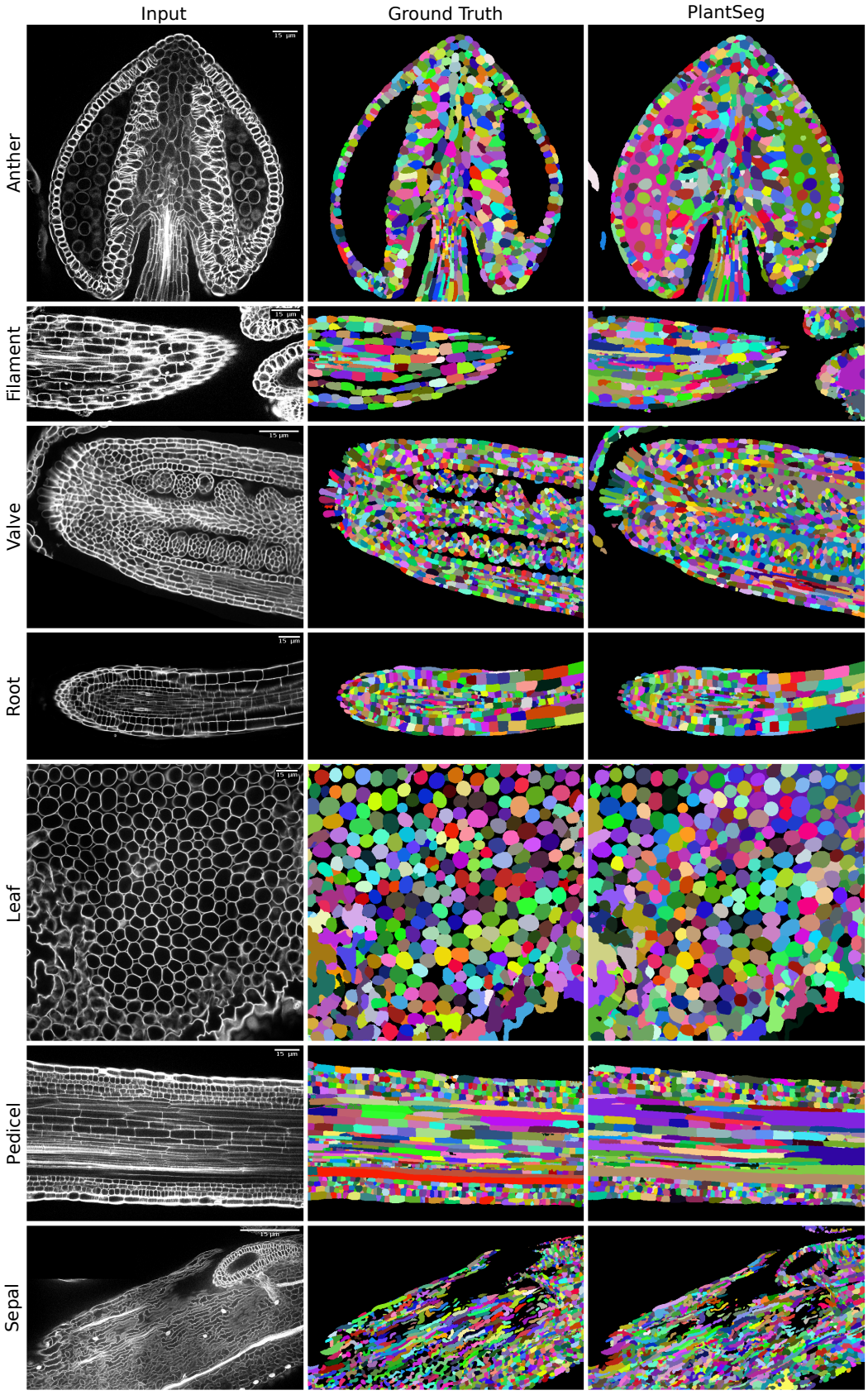


Figure 2.4. PlantSeg segmentation of different plant organs of the 3D Digital Tissue Atlas dataset, not seen in training. The input image, ground truth and segmentation results using PlantSeg are presented for each indicated organ.

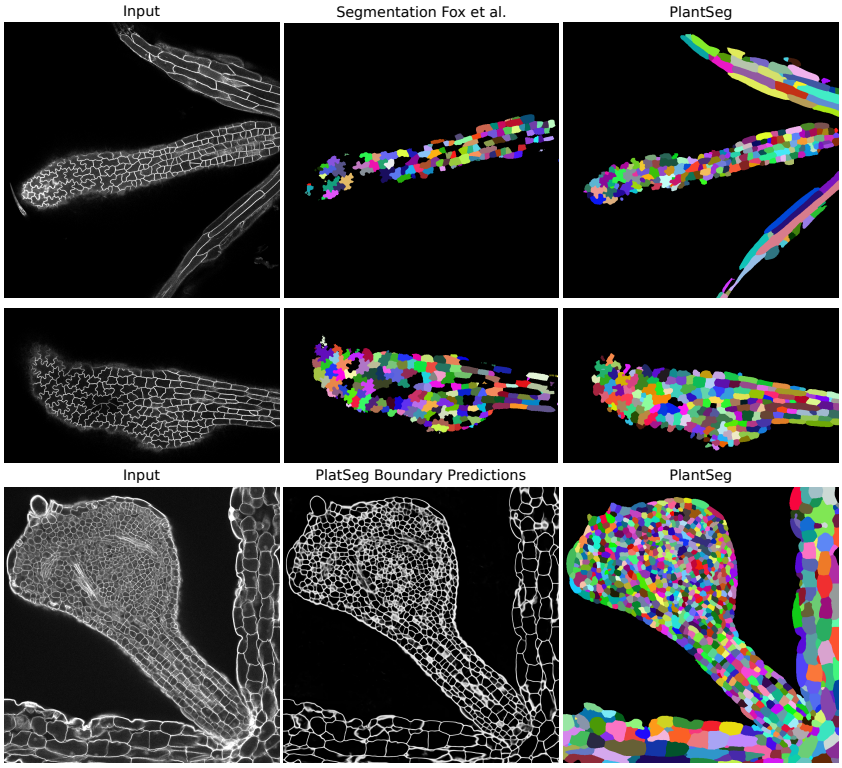


Figure 2.5. Qualitative results on the highly lobed epidermal cells from [141]. First two rows show the visual comparison between the SimpleITK (middle) and PlantSeg (right) segmentation on two different image stacks. PlantSeg's results on another sample is shown in the third row. In order to show pre-trained networks' ability to generalized to external data, we additionally depict PlantSeg's boundary predictions (third row, middle). We obtained the boundary predictions using the *generic-confocal-3d-unet* and segmented using GASP with default values. A value of 0.7 was choosen for the under/over segmentation factor.

performance of PlantSeg on these datasets may be underestimated.

Altogether, PlantSeg performed well qualitatively and quantitatively on datasets acquired by different groups, on different microscopes, and at different resolutions than the training data. This demonstrates the generalization capacity of the pre-trained models from the PlantSeg package.

2.2.3 Performance on a non-plant benchmark

For completeness, we compared PlantSeg performance with state-of-the-art methods on a non-plant open benchmark consisting of epithelial cells of the *Drosophila* wing disc [143]. Visually, the benchmark images are quite similar to the ovules dataset: membrane staining is used along with a confocal microscope, and the cell shapes are compact and relatively regular. Although we did not train the neural networks on the benchmark datasets and used

only the ovule pre-trained models provided with the PlantSeg package.

The dataset consists of 2D+t videos of membrane-stained developing *Drosophila* epithelial cells [144]. The benchmark dataset and the results of four state-of-the-art segmentation pipelines are reported in [143]. Treating the movie sequence as 3D volumetric images not only resembles the plant cell images shown in our study, but also allows to pose the 2D+t segmentation as a standard 3D segmentation problem.

We compared the performance of PlantSeg on the 8 movies of this dataset to the four reported pipelines: MALA [145], Flood Filling Networks (FFN) [146], Moral Lineage Tracing (MLT) [147, 148] and Tissue Analyzer (TA) [143]. On our side, we evaluate two PlantSeg predictions: for the first one, we use the boundary detection network trained on the *Arabidopsis thaliana* ovules. This experiment gives an estimate of how well our pre-trained networks generalize to non-plant tissues. For the second evaluation, we retrain the network on the training data of the benchmark and obtain an estimate of the overall PlantSeg approach accuracy on non-plant data. Note that unlike other methods reported in the benchmark, we do not introduce any changes to account for the data being 2D+t rather than 3D, i.e. we do not enforce the lineages to be moral as the authors of [143] did with their segmentation methods.

For the first experiment, peripodial cells were segmented using the 3D U-Net trained on the ovule dataset together with GASP segmentation, whereas proper disc cells were segmented with 2D U-Net trained on the ovule dataset in combination with Multicut algorithm. Both networks are part of the PlantSeg package. Qualitative results of our pipeline are shown in Fig. 2.6: PlantSeg produces very good segmentations on both the peripodial and proper imaginal disc cells. A few over-segmentation (peripodial cells) and under-segmentation (proper disc) errors are marked in the figure. This impression is confirmed by quantitative benchmark results in Tab. 2.2.

For the second experiment, we trained the network on the ground truth labels included in the benchmark (*PlantSeg (trained)*). Here, our pipeline is comparable to state-of-the-art. The difference in SEG metric between 'vanilla' PlantSeg and *PlantSeg (trained)* is 6.9 percent points on average, which suggests that for datasets sufficiently different from the ones PlantSeg networks were trained on, re-training the models might be necessary. Looking at the average run-times of the methods reported in the benchmark shows that PlantSeg pipeline is clearly the fastest approach with the average run-time of 3 min per movie when run on a server with a modern GPU versus 35 min (MALA), 42 min (MLT) and 90 min (FFN).

Thus, PlantSeg achieves results which are competitive with other proven methods in terms of accuracy, without explicitly training the boundary detection networks on the epithelial cell ground truth or accounting for the 2d+t nature of the data. PlantSeg outperforms

Method	PERIPODIAL	PROPER DISC
MALA	0.907 ± 0.029	0.817 ± 0.009
FFN	0.879 ± 0.035	0.796 ± 0.013
MLT-GLA	0.904 ± 0.026	0.818 ± 0.010
TA	-	0.758 ± 0.009
PlantSeg	0.787 ± 0.063	0.761 ± 0.035
PlantSeg (trained)	0.885 ± 0.056	0.800 ± 0.015

Table 2.2. Epithelial Cell Benchmark results. We compare PlantSeg to four other methods using the standard SEG metric [149] calculated as the mean of the Jaccard indices between the reference and the segmented cells in a given movie (higher is better). Mean and standard deviation of the SEG score are reported for peripodial (3 movies) and proper disc (5 movies) cells. Additionally we report the scores of PlantSeg pipeline executed with a network trained explicitly on the epithelial cell dataset (last row).

all methods in term of computing time. We argue that the large selection of pre-trained networks and graph partitioning algorithms make PlantSeg versatile enough to work well on wide variety of membrane stained tissues, beyond plant samples.

2.2.4 Exploiting nuclei staining to improve cells segmentation

The lateral root dataset contains a nuclei marker in a separate channel. In such cases we can take advantage of the fact that a cell contains only one nucleus and use this information as an additional clue during segmentation.

For this, we first segmented the nuclei using a simple and accurate segmentation pipeline that consists of a 3D U-Net trained to predict the binary nuclei mask followed by thresholding of the probability maps and connected components. We then incorporated this additional information into the multicut formulation, called lifted multicut [150,151], where additional repulsive edges are introduced between the nodes in the graph corresponding to the different nuclei segmented from the second channel.

We compared the scores of this lifted multicut algorithm to the scores for GASP, multicut and mutex watershed (see Tab. A.1). We see that lifted multicut performs competitively not only the standard multicut, but also all the other algorithms. This is because lifted multicut is able to separate two cells incorrectly merged into one region by the segmentation algorithm, as long as the region contains the two different nuclei instances corresponding to the merged cells.

A 3D U-Net trained to predict nuclei mask is available in the PlantSeg package. Lifted multicut segmentation can be executed via the PlantSeg's command line interface. We refer

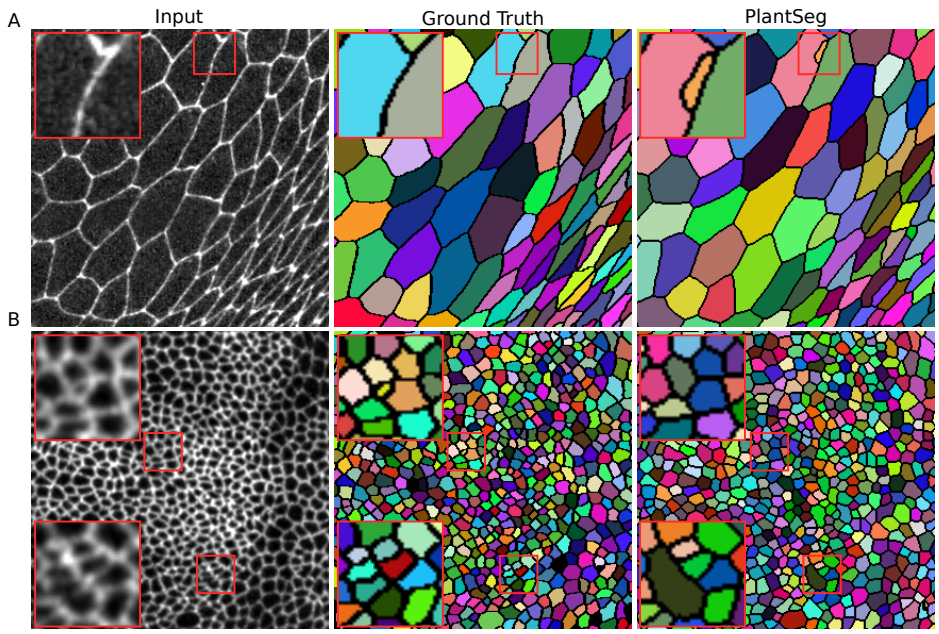


Figure 2.6. Qualitative results on the Epithelial Cell Benchmark. From top to bottom: Peripodial cells (A), Proper disc cells (B). From left to right: raw data, groundtruth segmentation, PlantSeg segmentation results. PlantSeg provides accurate segmentation of both tissue types using only the networks pre-trained on the *Arabidopsis* ovules dataset. Red rectangles show sample over-segmentation (A) and under-segmentation (B) errors. Boundaries between segmented regions are introduced for clarity and they are not present in the pipeline output.

to the project's GitHub page for instructions.

2.2.5 Automating proofreading from nuclei segmentation

Similarly to the lifted multicut workflow discussed in Sec. 2.2.4, PlantSeg implements an alternative workflow for automatically correcting the cell segmentation from a trusted nuclei segmentation. The workflow will adjust the cell segmentation to resolve conflicts with the respective nuclei segmentation. Therefore for this workflow, the accuracy of the nuclei is extremely important. Mistakes in the nuclei segmentation will propagate in the cell segmentation. The workflow comprises two subroutines: one for fixing the split mistakes in the cell segmentation and one for fixing the merge mistakes. The split routine checks for every cell if two or more nuclei respectively overlap (measured in percentage over the total cell volume) with the cell segmentation for more than a user-defined *threshold-split*. If the overlap exceeds the threshold, the script will use the nuclei segmentation as a seed and split the cell using the seeded watershed algorithm. The merge routine checks for every nucleus if two or more cells respectively overlap (measured in percentage over the total nucleus volume) a

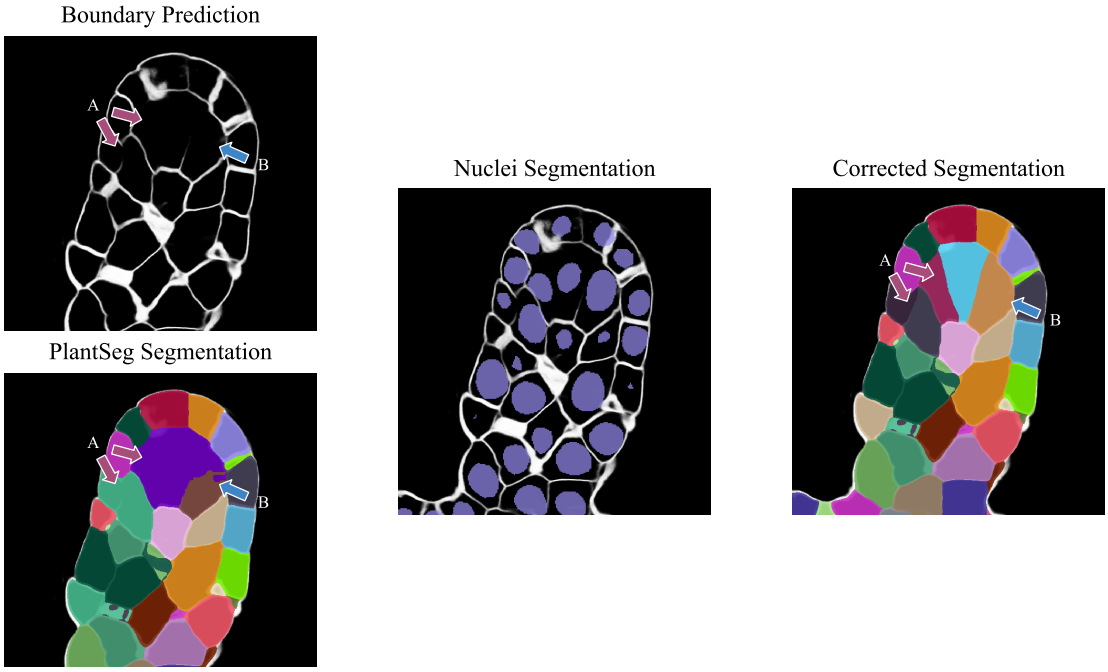


Figure 2.7. Overview of the automated proofreading from nuclei widget. This widget is particularly useful when the user poses a trusted nuclei segmentation and wants to use it to fix segmentation errors automatically. In (A), one can see how the tool can fix under-segmentation mistakes using the nuclei prior, while in (B), the tool fixes a case of over-segmentation.

single nucleus segmentation for more than a user-defined *threshold-merge*. The script will merge the cells if the overlap is above the threshold. The default thresholds provided are 66% for the *threshold-split* and 33% for the *threshold-merge*.

2.2.6 A package for segmentation and benchmarking

We release PlantSeg as an open-source software for the 2D and 3D segmentation of cells with cell contour staining. PlantSeg allows for data pre-processing, boundary prediction with neural networks and segmentation of the network output with a choice of four partitioning methods: Multicut, GASP, Mutex watershed and DT watershed.

PlantSeg can be executed via a simple graphical user interface, via a Napari plugin [152], or via the command line. The critical parameters of the pipeline are specified in a configuration file. Running PlantSeg from the graphical user interface or from the Napari plugin is well suited for the processing of single experiments, while the use of the command line utility enables large scale batch processing and remote execution. Our software can export both the segmentation results and the boundary probability maps in Hierarchical Data Format (HDF5) or Tagged Image File Format (TIFF). Both file formats are widely supported,

and other bioimage analysis tools, such as ilastik, MorphographX, or Fiji can further process the results. In particular: the final segmentation is exported as a labeled volume where all pixels of each segmented cell is assigned the same integer value. It is best viewed with a random lookup table, such as ”glasbey” in Fiji. In exported boundary probability maps each pixel has a floating point number between 0 and 1, reflecting the probability of that pixel belonging to a cell boundary. PlantSeg comes with several 2D and 3D networks pre-trained on different voxel sizes of the Arabidopsis ovule and LRP datasets. Users can select from the available set of pre-trained networks the one with features most similar to their datasets. Alternatively, users can let PlantSeg select the pre-trained network based on the microscope modality (light sheet or confocal) and voxel size. PlantSeg also provides a command-line tool for training the network on new data when none of the pre-trained networks is suitable to the user's needs.

PlantSeg is publicly available `https://github.com/hci-unihd/plant-seg`. The repository includes a complete user guide and the evaluation scripts used for quantitative analysis. Besides the source code, we provide a multiplatform conda package. The software is written in Python, the neural networks use the PyTorch framework [153]. We also make available the raw microscopic images and the ground truth used for training, validation, and testing.

Creating dynamical workflows

While the two-step pipeline, as described in Sec. 2.2.1 is the main workflow of PlantSeg, this workflow can be too restrictive for a challenging dataset. To allow users to access the full potential of our package, we allow users to create their workflows dynamically using our PlantSeg-Napari plugin. This allows users to execute all steps available in PlantSeg independently and in a customized order. In Fig. 2.9, we present a use case for a dynamic workflow scenario on a difficult specimen. Here, we show how one can drastically improve the workflow by performing multiple independent boundary predictions with different models and then merging them in a single improved boundary image.

Once the user is satisfied, the current workflow can be exported for future headless batch processing.

Interactive segmentation proofreading

While the primary goal of PlantSeg is to ensure accurate automatic segmentation, proofreading is often necessary to ensure 100% accuracy. Since version 1.5, PlantSeg has directly incorporated a widget for manual proofreading in the PlantSeg-Napari viewer. The tool's functionalities are presented in Fig. 2.10. Moreover, in Sec. 2.3.4, we will further describe

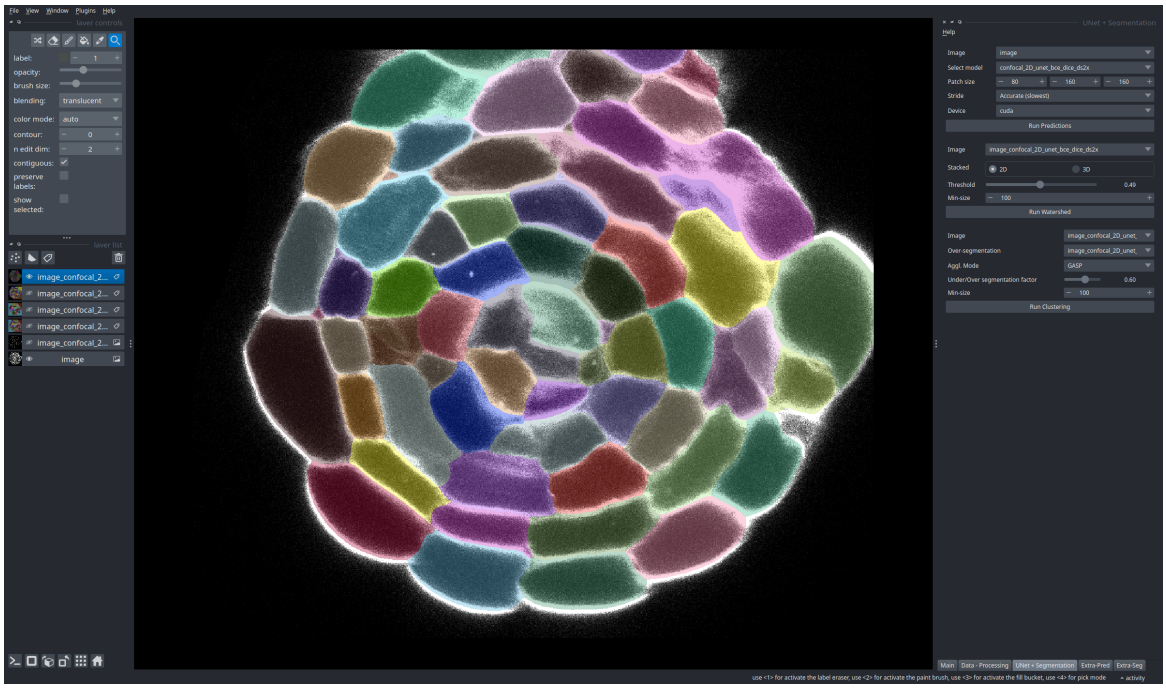


Figure 2.8. Screen capture of the PlantSeg-Napari viewer. The interface allows running independently all steps of the segmentation pipeline, such as: selecting the neural network model and specifying hyperparameters of the partitioning algorithm. In addition to the basic PlantSeg workflow, The interface allows for semi-automated proofreading from scribbles. Moreover, the user can export the dynamically created workflow for headless batch processing.

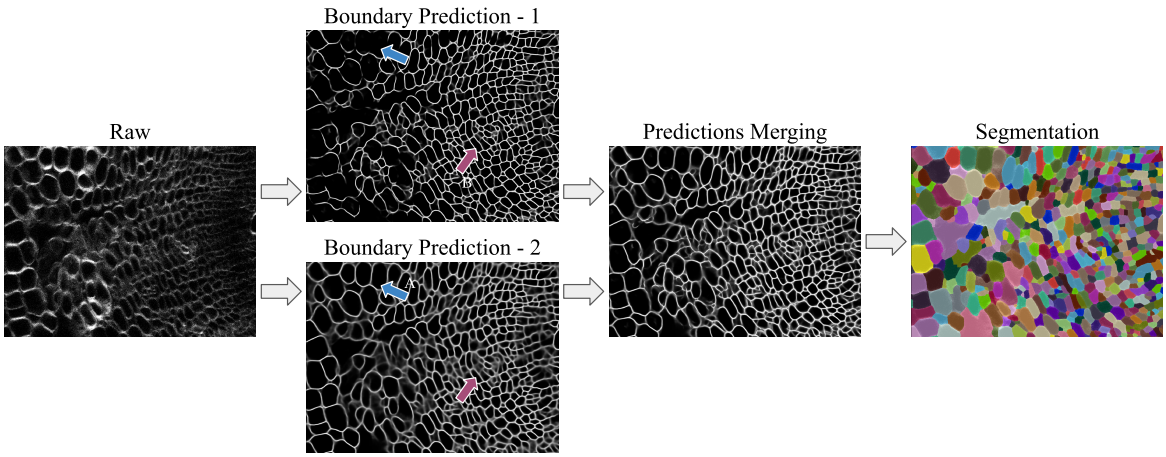


Figure 2.9. Example of customized Workflow. None of our pre-trained models can accurately process the raw input in this example. But, one can see that one of the models excels in some regions (Boundary predictions 1, region B), while the other performs better in a different area (Boundary predictions 2, region A). In this challenging scenario, merging the two models' output allows us to obtain a higher quality segmentation.

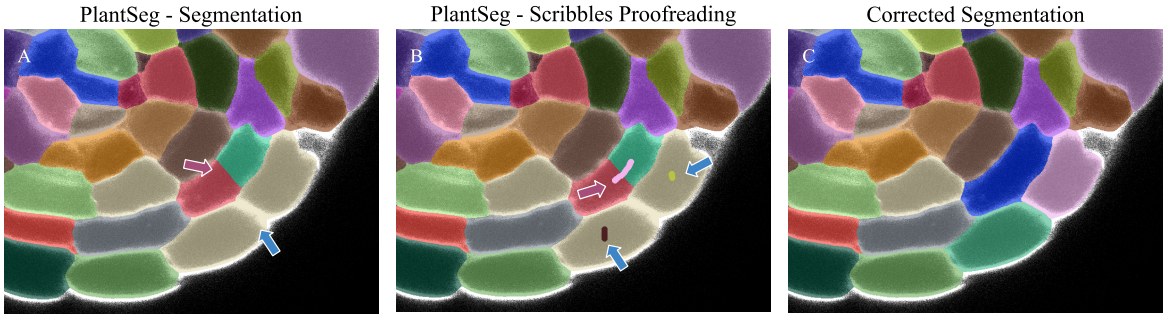


Figure 2.10. PlantSeg-Napari proofreading widget allows the user to perform semi-automated proofreading from scribbles. In (A), one can see a segmentation with two noticeable mistakes, an over-segmentation (red arrow) and an under-segmentation (blue arrow). To fix the over-segmentation, the user can draw a single scribble connecting the two segments (B, red arrow). To fix the under-segmentation, the user can use scribbles to create watershed seeds (B, blue arrows). In (C), one can see the widget output.

a protocol for using the PlantSeg-Napari proofreading capability for ground truth creation.

2.3 Methods

2.3.1 Neural network training and inference

Training

2D and 3D U-Nets were trained to predict the binary mask of cell boundaries. Ground truth cell contours where obtained by taking the ground truth label volume, finding a 2 voxels-thick boundaries between labeled regions ($find\_boundaries(\cdot)$ function from the Scikit-image package [154]) and applying a Gaussian blur on the resulting boundary image. Gaussian smoothing reduces the high frequency components in the boundary image, which helps prevent over-fitting and makes the boundaries thicker, increasing the amount of foreground signal during training. Transforming the label image $\mathcal{S}_{\mathbf{x}}$ into the boundary image $\mathcal{I}_{\mathbf{x}}$ is depicted in Eq. (2.1).

$$\mathcal{I}_{\mathbf{x}} = \begin{cases} 1 & \text{if } \Phi(\mathcal{S}_{\mathbf{x}}) * G_{\sigma} > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Where $\Phi(\cdot)$ transforms the labeled volume into the boundary image, G_{σ} is the isotropic Gaussian kernel and $*$ denotes a convolution operator. We use $\sigma = 1.0$ in our experiments. Both standard and residual U-Net architectures were trained using Adam optimizer [12] with $\beta_1 = 0.9, \beta_2 = 0.999$, L2 penalty of 0.00001 and initial learning rate $\epsilon = 0.0002$. Networks

were trained until convergence for 150K iterations, using the PyTorch framework [153] on 8 NVIDIA GeForce RTX 2080 Ti GPUs. For validation during training, we used the adjusted Rand error computed between the ground truth segmentation and segmentation obtained by thresholding the probability maps predicted by the network and running the connected components algorithm. The learning rate was being reduced by a factor of 2 once the learning stagnated during training, i.e no improvements were observed on the validation set for a given number of iterations. We choose as best network the one with lowest Arand error values. For training with small patch sizes we used 4 patches of shape $100 \times 100 \times 80$ and batch normalization [135] per network iteration. When training with a single large patch (size $170 \times 170 \times 80$), we used group normalization layers [134] instead of batch normalization. The reason is that batch normalization with a single patch per iteration becomes an instance normalization [155] and makes the estimated batch statistics weaker. All networks use the same layer ordering where the normalization layer is followed by the 3D convolution and a rectified linear unit (ReLU) activation. This order of layers consistently performed better than alternative orderings. During training and inference, input images were standardized by subtracting mean intensity and dividing by the standard deviation. We used random horizontal and vertical flips, random rotations in the XY-plane, elastic deformations [24] and noise augmentations (additive Gaussian, additive Poisson) of the input image during training in order to increase network generalization on unseen data. The performance of CNNs is sensitive to changes in voxel size and object sizes between training and test images [156]. We thus also trained the networks using the original datasets downscaled by a factor of 2 and 3 in the XY dimension.

3D U-Nets trained at different scales of our two core datasets (light-sheet lateral root, confocal ovules) are made available as part of the PlantSeg package. All released networks were trained according to the procedure described above using a combination of binary cross-entropy and Dice loss:

$$\mathcal{L} = \alpha \mathcal{L}_{BCE} + \beta \mathcal{L}_{Dice} \tag{2.2}$$

(we set $\alpha = 1$, $\beta = 1$ in our experiments) and follow the standard U-Net architecture [24] with two minor modifications: batch normalization [135] is replaced by group normalization [134] and same convolutions are used instead of valid convolutions. For completeness we also publish 2D U-Nets trained using the Z-slices from the original 3D stacks, enabling segmentation of 2D images with PlantSeg.

Inference

During inference we used mirror padding on the input image to improve the prediction at the boundaries of the volume. We kept the same patch sizes as during training since increasing it during inference might lead to lower quality of the predictions, especially on the borders of

the patch. We also parse the volume patch-by-patch with a 50% overlap between consecutive tiles and average the probability maps. This strategy prevents checkerboard artifacts and reduces noise in the final prediction.

The code used for training and inference can be found at `https://github.com/wolny/pytorch-3dunet`.

2.3.2 Segmentation using graph partitioning

The boundary predictions produced by the CNN are treated as a graph $G(V, E)$, where nodes V are represented by the image voxels, and the edges E connect adjacent voxels. The weight $w \in R^+$ of each edge is derived from the boundary probability maps. On this graph we first performed an over-segmentation by running the DT watershed [67]. For this, we threshold the boundary probability maps at a given value δ to get a binary image ($\delta = 0.4$ was chosen empirically in our experiments). Then we compute the distance transform from the binary boundary image, apply a Gaussian smoothing ($sigma = 2.0$) and assign a seed to every local minimum in the resulting distance transform map. Finally we remove small regions (< 50 voxels). Standalone DT watershed already delivers accurate segmentation and can be used as is in simple cases when, for example noise is low and/or boundaries are sharp.

For Multicut [63], GASP [60], and Mutex watershed [61] algorithms, we used the DT watershed as an input. Although all three algorithms could be run directly on the boundary predictions produced by the CNN (voxel level), we choose to run them on a region adjacency graph (RAG) derived from the DT watershed to reduce the computation time. In the region adjacency graph each node represents a region and edges connect adjacent regions. We compute edge weights by using the mean value of the probabilities maps along the boundary. We then run Multicut, GASP or Mutex watershed with a hyperparameter $beta = 0.6$ that balances over- and under-segmentation (with higher β tending to over-segment). As a general guideline for choosing the partitioning strategy on a new data is to start with GASP algorithm, which is the most generic. If needed, one may try to improve the results with multicut or mutex watershed. If none of the three strategies give satisfactory segmentation results we recommend to over-segment provided stack using the distance transform watershed and proofread the result manually using Paintera software [157].

A detailed overview of the parameters exposed via the PlantSeg's UI can be found on the project's GitHub page `https://github.com/hci-unihd/plant-seg` as well as in Appendix 6, Table 3.

2.3.3 Metrics used for evaluation

For the boundary predictions we used precision (number of pixels positively predicted as boundary divided by the number of boundary pixels in the ground truth), recall (number of positively predicted boundary pixels divided by the sum of positively and negatively predicted boundary pixels) and F1 score

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{2.3}$$

For the final segmentation, we used the inverse of the Adjusted Rand Index (AdjRand) [158] defined as ARand error $= 1 -$ AdjRand [159] which measures the distance between two clustering as global measure of accuracy between PlantSeg prediction and ground truth. An ARand error of 0 means that the PlantSeg results are identical to the ground truth, whereas 1 shows no correlation between the ground truth and the segmentation results. To quantify the rate of merge and split errors, we used the Variation of Information (VOI) which is an entropy based measure of clustering quality [160]. It is defined as:

$$\text{VOI} = H\left(\text{seg}|\text{GT}\right) + H\left(\text{GT}|\text{seg}\right), \tag{2.4}$$

where H is the conditional entropy function and the Seg and GT the predicted segmentation and ground truth segmentation respectively. $H\left(\text{seg}|\text{GT}\right)$ defines the split mistakes ($\text{VOI}_{\text{split}}$) whereas $H\left(\text{GT}|\text{Seg}\right)$ corresponds to the merge mistakes ($\text{VOI}_{\text{merge}}$).

2.3.4 Groundtruth creation

Training state-of-the-art deep neural network for semantic segmentation requires a large amount of densely annotated samples. Groundtruth creation for the ovule dataset has been described previously [137], we briefly describe here how the dense groundtruth labeling of cells for the lateral root was generated.

We bootstrapped the process using the Autocontext Workflow [161] of the open-source ilastik software [131] which is used to segment cell boundaries from sparse user input (scribbles). It is followed by the ilastik's multicut workflow [65] which takes the boundary segmentation image and produces the cell instance segmentation. These initial segmentation results were iteratively refined (see Fig. 2.11). First, the segmentation is manually proofread in a few selected regions of interest. Second, a state-of-the-art neural network is trained for boundary detection on the manually corrected regions. Third, PlantSeg framework consisting of neural network prediction and image partitioning algorithm is applied to the entire dataset resulting in a more refined segmentation. The 3-step iterative process was repeated until an instance segmentation of satisfactory quality was reached. A final round of manual proofreading is performed to finalize the groundtruth. The open-source software

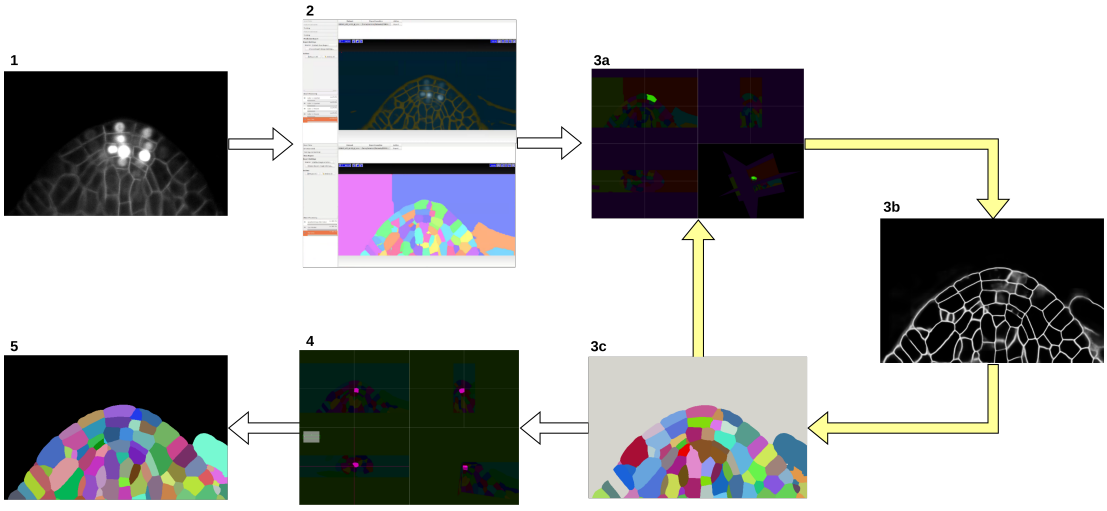


Figure 2.11. Later root groundtruth creation process. Starting from the input image (1), an initial segmentation is obtained using ilastik Autocontext followed by the ilastik multicut workflow (2). Paintera is used to proofread the segmentation (3a) which is used for training a 3D UNet for boundary detection (3b). A graph partitioning algorithm is used to segment the volume (3c). Steps 3a, 3b and 3c are iterated until a final round of proofreading (4) and the generation of satisfactory final groundtruth labels (5).

Paintera [157] has been used for the manual proofreading of the lateral root. Since version 1.5, similar proofreading capabilities are built in PlantSeg (see Sec. 2.2.6).

This protocol can be used to generate new ground truth datasets for different plant tissues and microscopy modalities. The initial bootstrapping can be either performed with ilastik Autocontext Workflow or using PlantSeg build-in models.

2.4 Conclusion

Taking advantage of the latest developments in machine learning and computer vision we created PlantSeg, a simple, powerful, and versatile tool for plant cell segmentation. Internally, it implements a two-step algorithm: the images are first passed through a state-of-the-art convolutional neural network to detect cell boundaries. In the second step, the detected boundaries are used to over-segment the image using the distance transform watershed and then a region adjacency graph of the image superpixels is constructed. The graph is partitioned to deliver accurate segmentation even for noisy live imaging of dense plant tissue.

PlantSeg was trained on confocal images of *Arabidopsis thaliana* ovules and light sheet images of the lateral root primordia and delivers high-quality segmentation on images from these datasets never seen during training as attested by both qualitative and quantitative benchmarks. We experimented with different U-Net designs and hyperparameters, as well as

with different graph partitioning algorithms, to equip PlantSeg with the ones that generalize the best. This is illustrated by the excellent performance of PlantSeg without retraining of the CNNs on a variety of plant tissues and organs imaged using confocal microscopy (3D Cell Atlas Dataset) including the highly lobed epidermal cells [141]. This feature underlines the versatility of our approach for images presenting similar features to the ones used in training. In addition, PlantSeg comes with scripts to train a CNN on a new set of images and evaluate its performance. Besides the plant data, we compared PlantSeg to the state-of-the-art on an open benchmark for the segmentation of epithelial cells in the *Drosophila* wing disc. Using only the pre-trained networks, PlantSeg performance was shown to be close to the benchmark leaders, while additional training on challenge data has narrowed the gap even further.

Unlike intensity-based segmentation methods used, for example, to extract DAPI-stained cell nuclei, our approach relies purely on boundary information derived from cell contour detection. While this approach grants access to the cell morphology and cell-cell interactions, it brings additional challenges to the segmentation problem. Blurry or barely detectable boundaries lead to discontinuities in the membrane structure predicted by the network, which in turn might cause cells to be under-segmented. The segmentation results produced by PlantSeg on new datasets are not fully perfect and still require proof-reading to reach 100% accuracy. Importantly the newly proof-read results can then be used to train a better network that can be applied to this type of data in the future (see Sec. 2.2.6 for an overview of this process). If nuclei are imaged along with cell contours, nuclear signal can be leveraged for improving the segmentation as we have explored in [150] (see Sec. 2.2.4 for detailed procedure). In future work, we envision developing new semi-supervised approaches that would exploit the vast amounts of unlabeled data available in the plant imaging community.

During the development of PlantSeg, we realised that very few benchmark datasets were available to the community for plant cell segmentation tasks, a notable exception being the 3D Tissue Atlas [140]. To address this gap, we publicly release our sets of images and the corresponding hand-curated ground truth in the hope to catalyse future development and evaluation of cell instance segmentation algorithms.

Chapter 3

CellTypeGraph: a geometric computer vision benchmark

Classifying all cells in an organ is a relevant and difficult problem from plant developmental biology. We here abstract the problem into a new benchmark for node classification in a geo-referenced graph. Solving it requires learning the spatial layout of the organ including symmetries. To allow the convenient testing of new geometrical learning methods, the benchmark of *Arabidopsis thaliana* ovules is made available as a PyTorch data loader, along with a large number of precomputed features. Finally, we benchmark eight recent graph neural network architectures, finding that *DeeperGCN* currently works best on this problem.

3.1 Introduction

Understanding morphogenesis, the generation of form, remains a major challenge in biology. It requires a detailed quantitative description of the molecular and cellular processes of the underlying mechanism. 3D digital organs with cellular spatial resolution promise to help decipher the morphogenesis of complex organs in plants [104–107]. They can be generated by 3D microscopic imaging followed by cell instance segmentation and tissue annotation. Plant organs lend themselves to this approach as they exhibit a relatively well-structured, layered organization and the tissues can be identified based on their positions and morphology. Thus, plant developmental biology is profiting from a growing body of 3D digital organs. However, automatic annotation [104, 162, 163] of different tissue types within an organ remains a major problem in the field, particularly for plant organs of complex cellular architecture and shape such as ovules of *Arabidopsis thaliana* [111].

In medical image analysis, the analogous problem of segmenting whole-body scans has been addressed successfully using 3D encoder-decoder CNN architectures [164, 165]. In

the images of interest to plant morphogenesis, this straightforward approach does not work as well: here, the semantic classes lack distinctive local and texture features helpful for convolution based architectures [166]. Instead, the task requires very long-range spatial awareness and good geometrical reasoning. In response, here we cast the problem as a node classification task. To succeed, any approach requires informative input features, and indeed we show that a cell-adjacency graph with cell-level features is a powerful representation.

That said, our primary intent is twofold: first, to create a testing ground for machine learning on highly structured input. Indeed, it uniquely combines node classification typical of popular datasets [80–82] with the requirement to generalize across geo-referenced graphs like in Quantum Chemistry [76–79]. Our second intent is, by allowing computer vision and machine learning scientists to work on this problem without having to deal with data-processing issues that actually take the most time in many applied projects, to channel the creativity and resourcefulness of our community to help solve a fascinating biological problem.

As a starting point, we present extensive experiments evaluating the performance of state-of-the-art and popular models.

In summary, we make the following contributions:

1. We propose a new benchmark for node classification in a geo-referenced graph. We release the benchmark dataset together with a ready-to-use PyTorch [167] data loader. The source-code and usage instructions are available at:
   `https://github.com/hci-unihd/celltype-graph-benchmark`.

2. We run comparative experiments using state-of-the-art graph neural networks (GNNs) and offer a study of feature relevance.

3. We provide an extensive set of precomputed features and an additional set of ground truth labels.

3.1.1 Related work

Arguably the most popular type of dataset for node predictions is the family of citation datasets, such as Cora [80], CiteSeer [81] or PubMed [82]. Although the task is nominally identical, the learning task is here transductive, and the underlying topologies are extremely different. In our CellTypeGraph, the task is inductive, and nodes are geo-referenced with a limited variance in node degrees and no shortcuts between distant nodes. This is not typically the case in citation datasets. More closely related are other natural science datasets such as QM9 [76, 77] or ZINC [78], MoleculeNet [79]. Nodes here represent atoms and

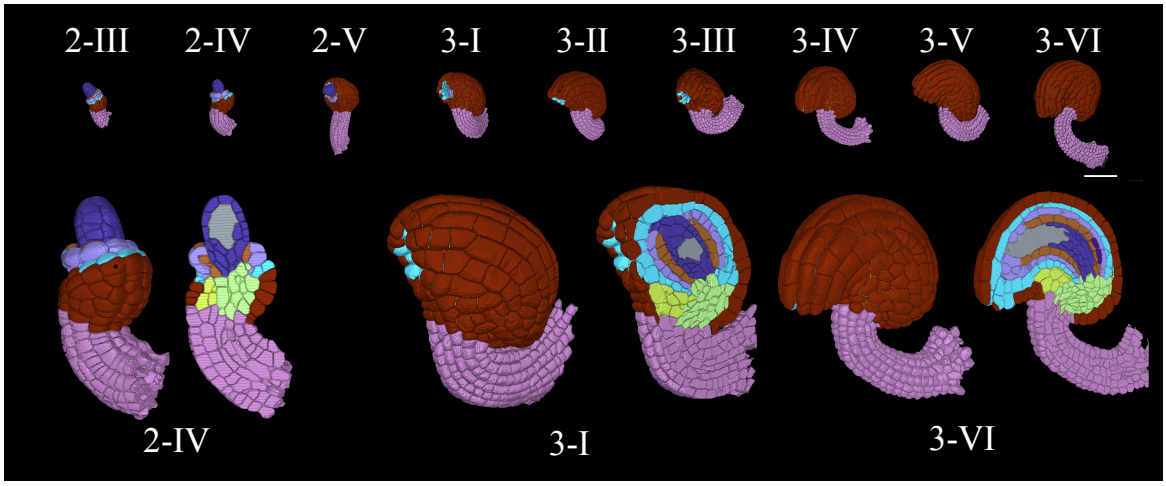


Figure 3.1. 3D surface view of a small subset of specimens from the CellTypeGraph Benchmark. Different developmental stages are indicated. From left to right the tissue complexity increases with organ growth. Scale bar $50\mu m$. Bottom: three stages are represented with their 3D view and a 2D section displaying the internal tissue architecture. Colors show ground truth cell types.

are geo-referenced. Nevertheless, they are, for the most part, interested in regression and classification of global graph properties.

The Open Graph Benchmark [168] contains a vast collection of datasets at different scales and also releases easy to use data loader and evaluation tools. As pointed out in [168], on many datasets the limited size, lack of agreed-upon train/val/test splits, and use of different metrics make it hard to measure progress in the field. In CellTypeGraph, we also release all tools required to ensure simple and reproducible experimentation. Moreover, we propose cross-validation [169] rather than a simple train/val/test split to further reducing sampling biases.

Motivated by the success of convolution neural networks, several GNN architectures attempt to generalize the same concept to non-grid structured data [83, 170, 171]. Those architectures and many more [86,87,90–95] can be modeled as message passing [89], where messages built at the node level can be shared over the local neighborhood. In case of geo-referenced or spatial graphs, the node positions can be used to model the messages aggregations over the graph adjacency [172, 173]. Alternatively, others [174, 175], have obtained competitive results on molecular tasks by using only node positions and ignoring the graph adjacency.

3.2 CellTypeGraph benchmark

3.2.1 Overview

We introduce the CellTypeGraph Benchmark, aiming to give a valuable tool to the geometric learning community. To do so, we distilled a publicly available biological dataset [111] into a ready-to-use machine learning benchmark. In particular, we would like to focus the community's attention on two primary objectives:

- Finding better GNN architectures and methodologies. In this case, we encourage using our pre-computed features, thus allowing direct comparison with our baseline, see Sec. 3.4.
- Finding more expressive features or, alternatively, replace hand crafted features by end-to-end learning. We will expand on this point in Sec. 3.3.2.

Progressing in either of the two challenges is an equally valid contribution to achieving the final goal of improving automated tools for plant-biology.

The statistics of the dataset are summarized in Tab. 3.1.

3.2.2 Raw data

The benchmark dataset is obtained from 3D confocal microscopic images of Arabidopsis ovule [111]. Ovules are the female gametophyte in higher plants that eventually gives rise to seeds upon successful fertilization.

The 3D digital ovule atlas is a composite of raw cell boundary images and their respective cell segmentation which are further tissue annotated. Fig. 3.1 shows a sample ovule from each developmental stage. Ovules at early stages (2-III to 2-V) protrude from the placental surface as a simple three layered dome like structure. Ovules form a complex 3D tissue structure during integument initiation and outgrowth. Four layers of integument tissues surround the core of the organ (cell labels *L1* to *L4*). Integument tissues undergo growth conflicts that result in their final shape with a hood-like outer structure at maturity. Inner tissue layers of integuments are arranged as a bent tube like shape. Overall, the organ is moulded by complex arrangements of cells in 3D that partly follow a stratified arrangement of cells in the medial and upper half of the organ and it also forms a partial radial and bilateral symmetry.

The ground truth cell type labels for the benchmark dataset were obtained semi-automatically by first using a modified detect layer stack process in MorphographX [176]; and then proof-reading the results from these predictions manually. Essentially, the layered cells in the upper medial half of the organ were labeled by the cell layer detection method and then

Number of specimens	84
Number of developmental stages	9
Total number of cells/nodes	95757
Total number of edges	632443
Average number of nodes per specimen	1140
Average number of edges per specimen	7529
Number of semantic classes	9
Number of node features	78
Number of edge features	11

Table 3.1. Salient statistics of the CellTypeGraph benchmark.

manually corrected, while the rest of the cells from the organ were labeled manually. This required extensive human input, totaling in the order of 60 human hours for the dataset in this study.

Tissue annotations allowed in extensive biological analysis of the cellular basis of growth in different tissues and at different developmental stages [111].

Pre-processing: The segmentation images as published in [111] have been further manually curated. We have ensured that for each specimen, cells form a single connected component. In case of multiple specimen imaged together, they have been split in separate stacks. Lastly, cell type annotations have been added for missing cells. Although we did our best to hand curate the data, minor imperfections might still be present due to the variability of the organs. Since the *L5* cell type is rarely present in the later stages, we slightly simplified the benchmark task by merging the cell types *L4* and *L5*.

3.2.3 Evaluation

Metrics: The ovules cell types are imbalanced. This not only impacts the learning procedure but also requires attention when discussing the results quantitatively. An effective way to account for imbalance is to evaluate the model performance for each class independently and then report as final score the average of the single class results. In this work, we use two metrics, the simple global *top-1 accuracy* and the *class-average accuracy*.

In order to further stabilize the *class-average accuracy*, we ignored cell label 7, see Fig. 3.2. Cell label 7 represents the “embryo sac”. This tissue is not a cell in itself but is an ensemble of non-segmentable cells in the inner part of later-stage ovules. This region is unique and distinct, being the largest and highest degree node in the cell graph. Nevertheless, we kept the “embryo sac” in the benchmark for training purposes because of its crucial role

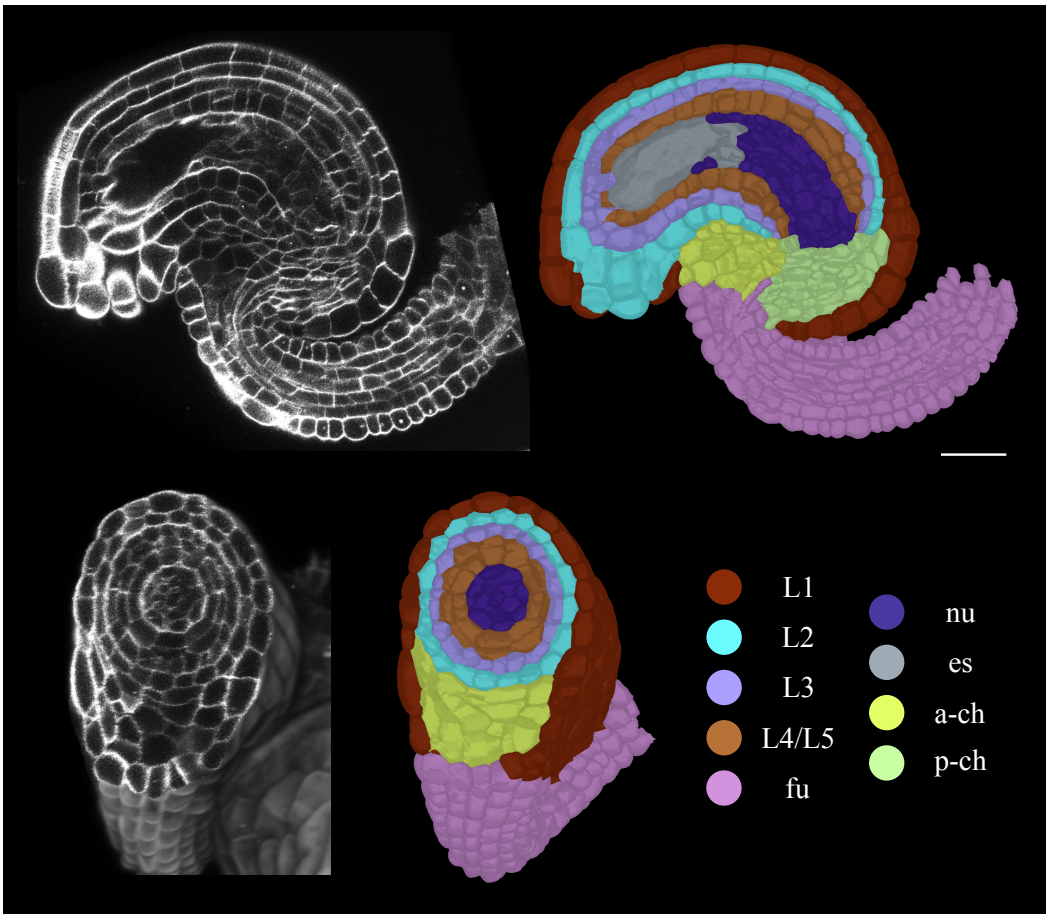


Figure 3.2. 2D section view of a mature *Arabidopsis* ovule displaying the raw cell boundary image and the respective CellTypeGraph ground truth labels manually annotated for the benchmark dataset. Different colors indicate different tissue labels annotated to the 3D instance cell segmentation. Abbreviations *es*: embryo sac, *nu*: nucellus, *L1*: outer layer of outer integument, *L2*: inner layer of outer integument, *L3*: outer layer of inner integument, *L4*/*L5*: inner layer of inner integument, *fu*: funiculus, *a-ch*: anterior chalaza, *p-ch*: posterior chalaza. Scale bar $20\mu m$.

in graph connectivity.

Moreover, if any cell type is not present in a specific specimen, that cell type will not be counted in the *class-average accuracy*, i.e.,

$$\textit{class-average accuracy} = \frac{1}{N_s} \sum_{s=1}^{N_s} \frac{\sum_c a_c \cdot \mathbb{1}_{s,c}}{\sum_c \mathbb{1}_{s,c}} \tag{3.1}$$

where N_s is the number of specimens, a_c is the one-vs-all class accuracy, $\mathbb{1}$ is an indicator function valued 1 if the class c is present in the specimen s, 0 otherwise. This is particularly relevant in the early stages of development.

We release evaluation code for all the metrics introduced above bundled with our benchmark.

Experts consensus The cell types are identified based on patterns of gene expression, cell position in space, and context in terms of tissue layers. The consensus is strong for layered cell types (*L1* to *L4*); here, segmentation quality is the dominant source of ambiguity. But variability can arise in regions where the boundaries of the tissue groups are hard to determine from the cell position itself; for example, the boundary between *fu*, *ch*, and *nu*. In order to assess this variability quantitatively and assess a reference performance for an expert biologist, we produced an independent second set of labels. The overall results are reported in Tab. 3.2, while an additional breakdown of expert performance at different developmental stages and for different classes is reported in Fig. 3.3.

Training and evaluation splits: We release our CellTypeGraph data loader in two different modes: a standardized train-validation-test split, and a five fold cross-validation split.

Although train-validation-test splits are standard practice in popular machine learning benchmarks, this approach makes evaluations susceptible to sampling biases [177]. In our CellTypeGraph, we have high variability between specimens both intra-stage and inter-stage, see Fig. 3.1. This, combined with the relatively small number of specimens, makes the splits highly susceptible to sampling bias. We address the stage variability by sampling the same number of specimen from each stage.

However, to further unbias our experiments, we preferred to focus on a five-fold cross-validation approach. Cross-validation mitigates the noise coming from the random sampling at the expense of a more costly training. All presented experiments have been evaluated using cross-validation.

3.3 Features

This section details the node and edge features included in the CellTypeGraph benchmark. However, before motivating their choice, it is necessary to discuss the reference system used to express position and orientation dependent features.

3.3.1 Reference systems

Global reference system: Ovule images acquired by the microscope are not always oriented in the same direction. This inconsistency in specimen orientation could cause poor generalization of trained models. Possible solutions are to systematically use rotation and translation equivariant methods or to fix a landmark-based global reference system. The first solution is more general and shows great potential in terms of generalization and parameter efficiency [178–180]. Equivaraint methods have a clear advatage when no standardized

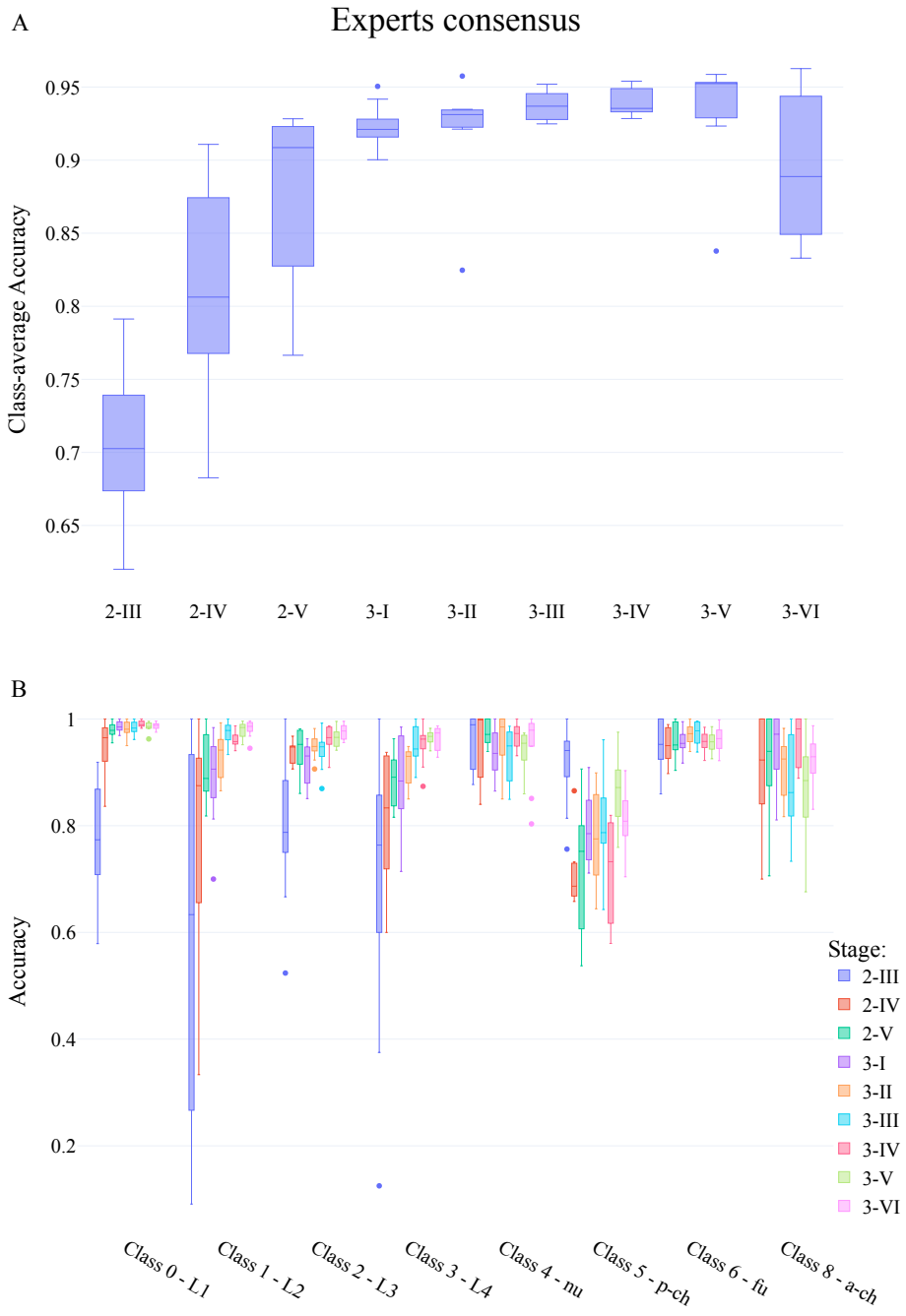


Figure 3.3. Class-average accuracy obtained by an expert biologist (A). The early stages pose the most substantial challenges, although the cause of such high variance can be attributed to the smaller number of cells for each specimen. Per-class accuracy obtained by an expert biologist (B). In late stages 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*.

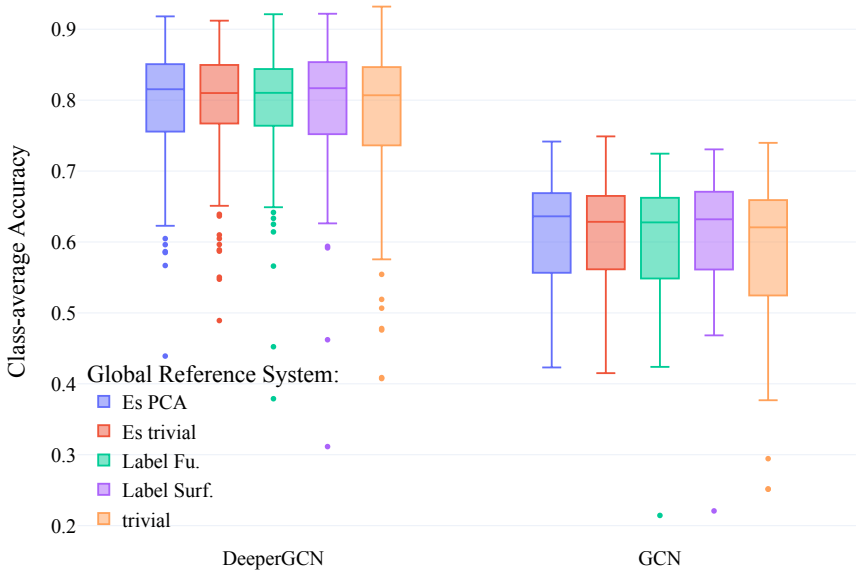


Figure 3.4. All reference systems are equally good. We trained both *GCN* and *DeeperGCN* on five distinct reference systems. There is no statistically significant difference in using either reference system. The only difference we could observe was in the spread of outliers.

orientation can be formed, as in general chemical problems. In our case, an unambiguous standard pose can be defined and so for simplicity we opt for a landmark-based approach.

More specifically, we evaluate four landmark-based approaches, both supervised (*Label Surf*, *Label Fu*) and unsupervised (*ES trivial*, *ES PCA*). Detailed descriptions can be found in Appendix B.1. All coordinate systems yield similar accuracy, see Sec. 3.4.2. Nevertheless, the local reference *Label Surf* is slightly more consistent and less prone to outliers compared with the others, see Fig. 3.4. Hence *Label Surf* was used as default orientation system for all of our experiments.

Local reference system: Ovule cells have no distinctive anisotropy that can be used to define a local orientation. However, preferential direction of growth and divisions along the central curved axis of the organ result in filar arrangement of cells within integument tissue layers (*L1* to *L4*). This regular filar arrangement of cells can be used to identify integument tissues and separate them from one another. Integument tissues divide anticlinally to maintain their sheet like structure. Cell divisions happen mainly along the transverse anticlinal direction allowing the tissue to grow in length. Division planes can be further mapped to faces of cell wall extracted from a 3D instance cell segmentation. This includes longitudinal anticlinal wall, transverse anticlinal wall and periclinal wall. In order to exploit this prior, we construct a heuristic algorithm to identify transverse anticlinal walls and define a *growth*

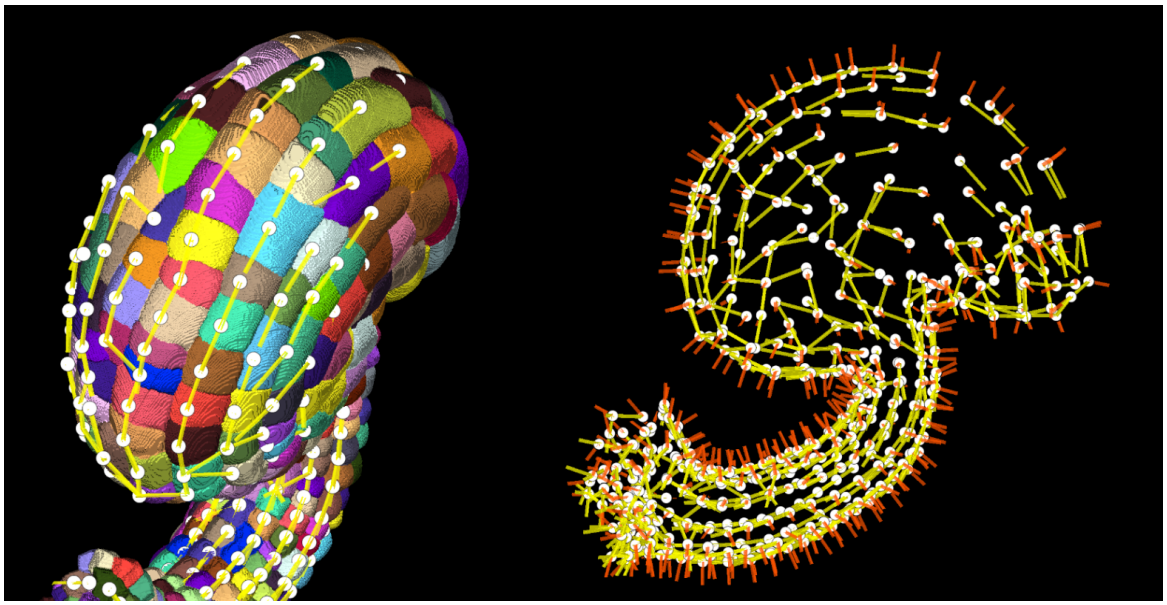


Figure 3.5. Automatically extracted local orientation features: the growth and surface directions are represented by the solid yellow lines and solid red lines respectively. Left: a 3D surface view of a mature Arabidopsis ovule overlaid with the predicted growth directions. The axis predictions align well with the filar arrangement of cells on the organ surface. Right: surface and growth orientations from the same ovule. Although the orientation of the growth axis follows the regular structure of the organ, the direction of the axis is arbitrary.

axis based on their connectivity.

Moreover, these tissues (*L1* to *L4*), follow a stratified pattern as they continue dividing anticlinally. In this case, we propose another simple heuristic to construct a *surface axis*, parallel to the stratification direction. Both algorithms are described in details in Appendix B.2. The approximate *growth axis* and *surface axis* we find in this way have two significant limitation: the axes are not guaranteed to be orthogonal, and only their orientation is defined, but not their direction. Lastly, in order to have a complete 3D basis for our local reference system, we simply compute a third axis orthogonal to the first two. An illustration of the predicted *growth* and *surface axes* can be found in Fig. 3.5.

3.3.2 Feature extraction

As a first step, we compute the Cell Adjacency Graph $G(\mathcal{N}, \mathcal{E})$, where every cell is represented as node $n_i \in \mathcal{N}$ and an edge $e_{i,j} \in \mathcal{E}$ connects every pair of touching cells. Secondly, we sample a fixed number of points from the surface of each cell. To obtain equally distributed points, we use the farthest points sampling algorithm [181] over the cell surfaces. These sampled points are used to compute downstream features more efficiently, when necessary. However, the same surface samples could potentially be used to learn features end-

to-end, using ideas from equivariant point cloud architectures. Although this direction is enticing, it would result in a much more complex pipeline. We also sample points from the boundary shared by any two touching cells, using the same sampling strategy. In each case, we sample 500 points per node/per edge.

We can divide all features into two categories: *invariant* and *covariant features*.

Invariant features are independent from the reference system, and do not change under rotation or translation of the specimen. These features are either morphological — such as the cell volume, surface area, lengths along the local reference axis, and the angle formed between the surface and growth axis — or they are derived from the Cell Adjacency Graph, such as the shortest path from each node to the ovule surface, current-flow closeness centrality [182], and degree centrality. In Fig. 3.6 we show in false colour a small selection of invariant features.

Covariant features instead transform with the reference system under rotation and translation. In this category we include: the cell center of mass, the local reference axes, and the cell PCA axes. In addition we also include: the angles formed between the local and the global reference system, and the angles between PCA axes and the global reference system. Although angles are not formally covariant, they are derived from covariant features and transform predictably under rotation and translation.

Edge features are also pre-computed in our benchmark, in particular: shared cell boundary surface, the distance between adjacent centers of mass, and the angles between the local reference axes of adjacent cells. But as discussed in Sec. 3.4.1, these brought no significant improvement in the architectures we tested.

3.3.3 Feature homogenization

Appropriate feature processing has a great impact on the training dynamics. It is extremely important to properly normalize and homogenize incommensurable features before concatenating them.

To provide the orientation of PCA and local reference system axes to the network, we used the following transformation:

$$f : \mathbb{R}^3 \to \mathbb{R}^6, (x, y, z) \to \left(x^2, y^2, z^2, xy, xz, yz\right),$$

That is, we removed the ambiguity in choosing direction and not just orientation. This yields an embedding of the manifold of orientations $\mathbb{R}P^2$ in $\mathbb{R}^6$, whose existence is guaranteed by the Whitney embedding theorem [183, 184].

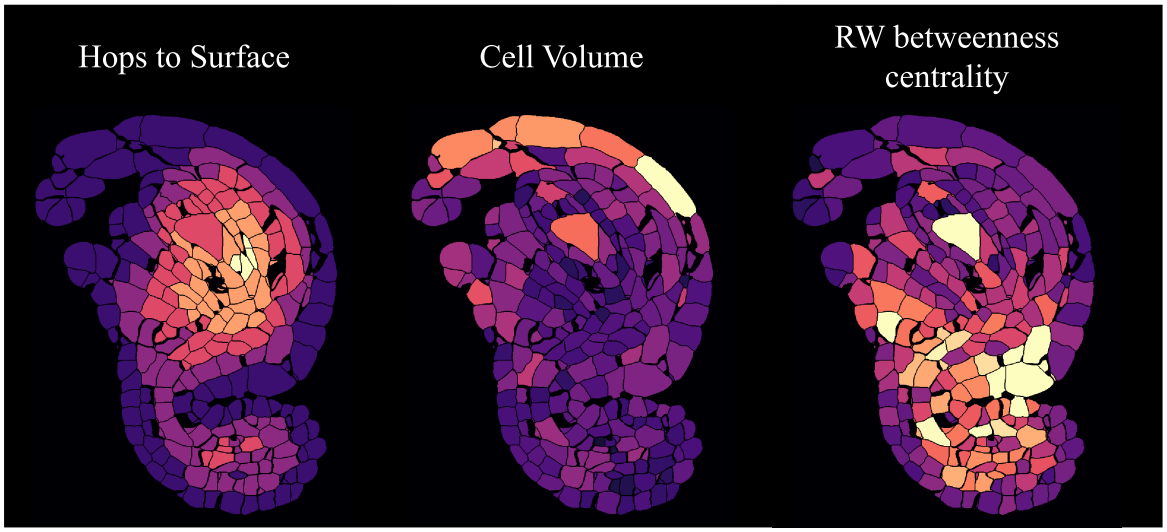


Figure 3.6. Selection of invariant features. From left to right visualized in false color one can see: The number of shortest path from each node to the ovule surface, the cell volume and the current-flow closeness centrality (also called random walker betweenness centrality).

All scalar features (except angles) are normalized to zero mean and unit variance. Furthermore, we encode categorical features as a one-hot vector and scale vector features to unit norm.

This protocol has been established by testing different preprocessing and normalization strategies for each group of features. Results are shown in Appendix B.3.

We made a best effort to ensure that the features we selected are expressive. However, our data loader can be easily extended to add new features or change the processing of the current features. The features discussed so far are all included in the loader by default, but in Appendix B.3 we list additional features. All raw segmentations, annotations and precomputed features are available at `https://zenodo.org/record/6374104`.

3.4 Benchmarking graph neural networks

To show the accuracy of existing methods on this new benchmark, we experiment with a large variety of models. We tried to represent the most prevalent GNNs paradigms. In particular, we tested graph convolutional networks, attention or transformer based architectures as well as message-passing based architectures.

Architectures: We tested the following GNN architectures: *GCN* [83], *GraphSAGE*, [87], *GIN* [90], *GCNII* [92], and *DeeperGCN* [93,94]; all as implemented in [185]. Moreover, we tested a two layer re-implementation of the graph attention network *GAT* [86] architecture, a

variation of the same architecture *GATv2* using the graph convolution operation introduced in [95], and a further variation *TransformerGCN* utilizing a transformer [22] based graph convolution from [186].

Two of the architectures tested can take into account edge features, the *TransformerGCN* and the *DeeperGCN*. When trained with edge features we will refer to them as *EdgeTransformerGCN* and *EdgeDeeperGCN*. All models used softmax as last layer activation.

Training: All models have been trained with the ADAM optimizer [187] and cross entropy loss with $\mathbb{L}_2$ weight penalty. Learning rate, weight regularization and model specific hyperparameters such as: number of layers, hidden features, dropout etc. have been tuned for each model using a coarse grid search.

Every training instance uses a single GPU and less than 2GB of VRAM. A complete five fold cross validation of the *DeeperGCN* completes in under one hour using a single Nvidia RTX6000 GPU. For all experiments, source code is provided at `https://github.com/hci-unihd/plant-celltype`.

Model		
Node Classification	top-1 acc.	class-avg. acc.
GIN [90]	0.714 ± 0.071	0.563 ± 0.136
GCN [83]	0.762 ± 0.043	0.617 ± 0.077
GAT [86]	0.824 ± 0.033	0.705 ± 0.084
GATv2 [95]	0.855 ± 0.041	0.757 ± 0.087
GraphSAGE [87]	0.859 ± 0.048	0.765 ± 0.093
GCNII [92]	0.863 ± 0.050	0.772 ± 0.100
Transf. GCN [186]	0.868 ± 0.045	0.779 ± 0.098
EdgeTransf. GCN [186]	0.868 ± 0.044	0.777 ± 0.098
DeeperGCN [94]	$\mathbf{0.877 \pm 0.050}$	$\mathbf{0.796 \pm 0.098}$
EdgeDeeperGCN [94]	$\mathbf{0.878 \pm 0.047}$	$\mathbf{0.797 \pm 0.095}$
Expert Biologist	0.932 ± 0.025	0.909 ± 0.049

Table 3.2. Top-1 accuracy and class-average accuracy obtained by different architecture on the node classification task. *DeeperGCN* is consistently the best performing architecture. Nevertheless, *DeeperGCN* results are still weaker than human expert. *EdgeDeeperGCN* and *EdgeTransformerGCN* have been trained with additional edge features, but their impact on our metrics was not significant. Uncertainty is defined as the standard deviation over all specimens and cross-validation folds.

3.4.1 Baseline results

In Tab. 3.2, we present the best performing models over the five-fold cross-validation splits according to the class-average accuracy. *DeeperGCN* is consistently the best performing architecture, although it does not match yet the expert biologist performance.

Overall, all attention mechanism methods performed relatively well, while simpler models like *GCN* and *GIN* struggled the most. For both high- and low-end models, using edge features did not increase accuracy. Fig. 3.8 shows the class-average accuracy broken down by developmental stages. While *DeeperGCN* has a decisive advantage in the later stages compared to *GCN*, we can observe a much narrower spread in the early stages (2-III to 2-V). Fig. 3.7 shows the average accuracy by cell type. One can easily identify the categories *L2*, *L4* and *ac* to be the most challenging for both *GCN* and *DeeperGCN*.

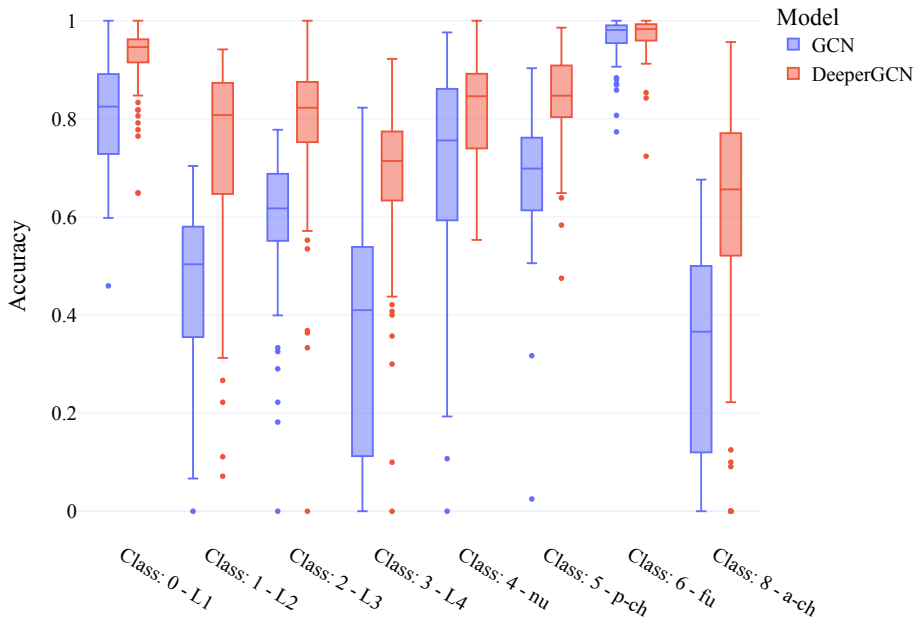


Figure 3.7. Top-1 accuracy divided by ground truth category. Accuracy varies drastically across classes. In particular, the *GCN* accuracy plummets for *L2*, *L4*, and *a-ch* tissues.

3.4.2 Additional experiments

The same standardized setup was used for all our additional experiments. We limited ourselves to two models, the *GCN* [83] and the *DeeperGCN* [94], and used optimal hyperparameters from Sec. 3.4, see Appendix B.4 for each.

Importance of the global reference system

As discussed in Sec. 3.3.1, some of the features are sensitive to the orientation of the specimen. We tested and compared four landmark-based orientations plus the trivial orientation, i.e., center of mass in the origin and original orientation as acquired by the microscope. From Fig. 3.4 we can observe no significant difference between different orientations. The only relevant difference is that the reference system found using the labels exhibits an overall smaller variance across specimens. On the other end, the trivial representation manifests the most significant number of outliers. Another interesting question for practitioners is the ability to generalize if the reference system changes. We tested that by training our pipeline with three different orientations and always testing on the same. As shown in Tab. 3.4, at test time, there is a significant drop in accuracy when evaluating on a different orientation. However, the experiment also shows that this can be mitigated by training on multiple orientations at the same time.

Invariant vs covariant features

To evaluate the respective contribution of invariant and covariant features to the accuracy, we trained our models using solely one of the two. Tab. 3.3 shows that the invariant features are unquestionably what the neural networks rely on most.

To further understand the difference, in Fig. 3.8 we report our results for each development stage separately. One can see a distinction between the two feature groups: the invariant features have a stronger impact on the accuracy in the later stages (3-I to 3-VI), while the covariant features have a more prominent role in the early stages (2-III to 2-V).

Local graph features alone

To emphasize the importance of the Cell Adjacency Graph structure for solving this task, we also trained models using only the degree of a node and distribution of degrees of is neighbours, the so called degree profile [188]. This is an extremely unfavorable setup since the degrees are quite homogeneous across the ovule cell graph. The results, shown in Fig. 3.8, reveal that even then, the networks are still able to predict systematically better than chance. Moreover, *DeeperGCN* in later stages performed comparably to the same model trained using the covariant features.

3.4.3 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 on our

Features	GCN	DeeperGCN
	Δ class-avg. acc.	Δ class-avg. acc.
Only invariant	-0.028	-0.019
Only covariant	-0.175	-0.337

Table 3.3. Contribution of each feature group to the accuracy. The table shows the difference between the class average accuracy of the baseline model and the class average accuracy of the perturbed model. The lower the delta, the higher is the feature importance for the neural networks. For both architectures, the invariant features are substantially more important than the covariant features.

Training Data	GCN	DeeperGCN
	class-avg. acc.	class-avg. acc.
ES PCA	0.578 ± 0.089	0.754 ± 0.118
All but *Label Surf*	0.618 ± 0.086	0.787 ± 0.103
Label Surf	0.613 ± 0.080	0.790 ± 0.100

Table 3.4. Generalization under different orientations. We tested how robust our baseline is to changes in the global orientation axes. We trained our model in different orientations, and we tested it on a particular one, *Label Surf*. From the class average accuracy, we can observe a substantial accuracy drop when training on a single different reference system, *ES PCA*. However, this can be easily compensated by augmenting the training data with multiple orientations. Uncertainty is defined as the standard deviation over all specimens.

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. 3.9. Our preliminary results show a small effect of data augmentation on the metrics, but more exhaustive experimentation is necessary to draw more solid conclusions.

3.4.4 Inference on new specimens

In addition to the benchmark, we release a dedicated repository with all necessary tools to run our trained models on new specimen. The inference can be run via command line interface. While for visualizing results and computed features we implemented a Napari viewer plugin. Source code is provided at `https://github.com/hci-unihd/plant-celltype`.

3.4.5 Limitations

The CellTypeGraph Benchmark is limited by the number of specimen. Imaging, segmenting, and annotating large 3D volumes is very time-consuming, and thus the number of specimens available is smaller compared to other datasets [78, 79]. This limitation can be mitigated by data augmentation. Although not exhaustive, this route did not show visible improvements in our experiments. A description of the experiments performed and results can be found in Sec. 3.4.3.

Nevertheless, the structural complexity of the ovules makes them a perfect candidate for a benchmark. However, ovules are not a universal proxy for all organs in plant-biology, thus pre-trained models on our benchmark are likely to fail on different organs. Possible solutions are: i) retraining the models when labels are available. ii) Using self-supervised methods such as graph autoencoders [189, 190] followed by community clustering in latent space. Such a setup has proven successful in single-cell data analysis [191, 192]. iii) Speeding up semi-automated labeling by employing active learning.

3.5 Conclusion

In this chapter, we have introduced a new graph benchmark for node classification, extensively tested several relevant GNN architectures, and released tools for quick experimentation, data handling, inference, and evaluation.

Beyond the benchmark, we established that GNNs are capable of accurately classifying cell types in the context of plant biology and have the potential to be a strong ally for quantitative biology research. In particular, models like DeeperGCN [94], Transf. GCN [186],

and GCNII [92] have been consistently good performing and are easy to train, making them prime candidates for future experimentation for similar geo-referenced graph problems.

Although results from our baseline are encouraging, we are excited to discover how far the geometric learning community will push this task. As discussed in Sec. 3.4.5, tissue labeling in 3D virtual organs is a relevant and relatively unexplored domain. We also hope to see use of the benchmark outside the strict paradigms of supervised learning, such as self-supervised and active learning. Moreover, the two main sources of complexity in the benchmark are the hand-crafted features and the need to define a global orientation. End-to-end feature learning could be achieved by applying models from the point clouds geometric deep learning domain [181, 193, 194] to our surface sample, see Sec. 3.3.2. While new exciting frameworks like [75] are expediting easy access to equivariant neural network architectures.

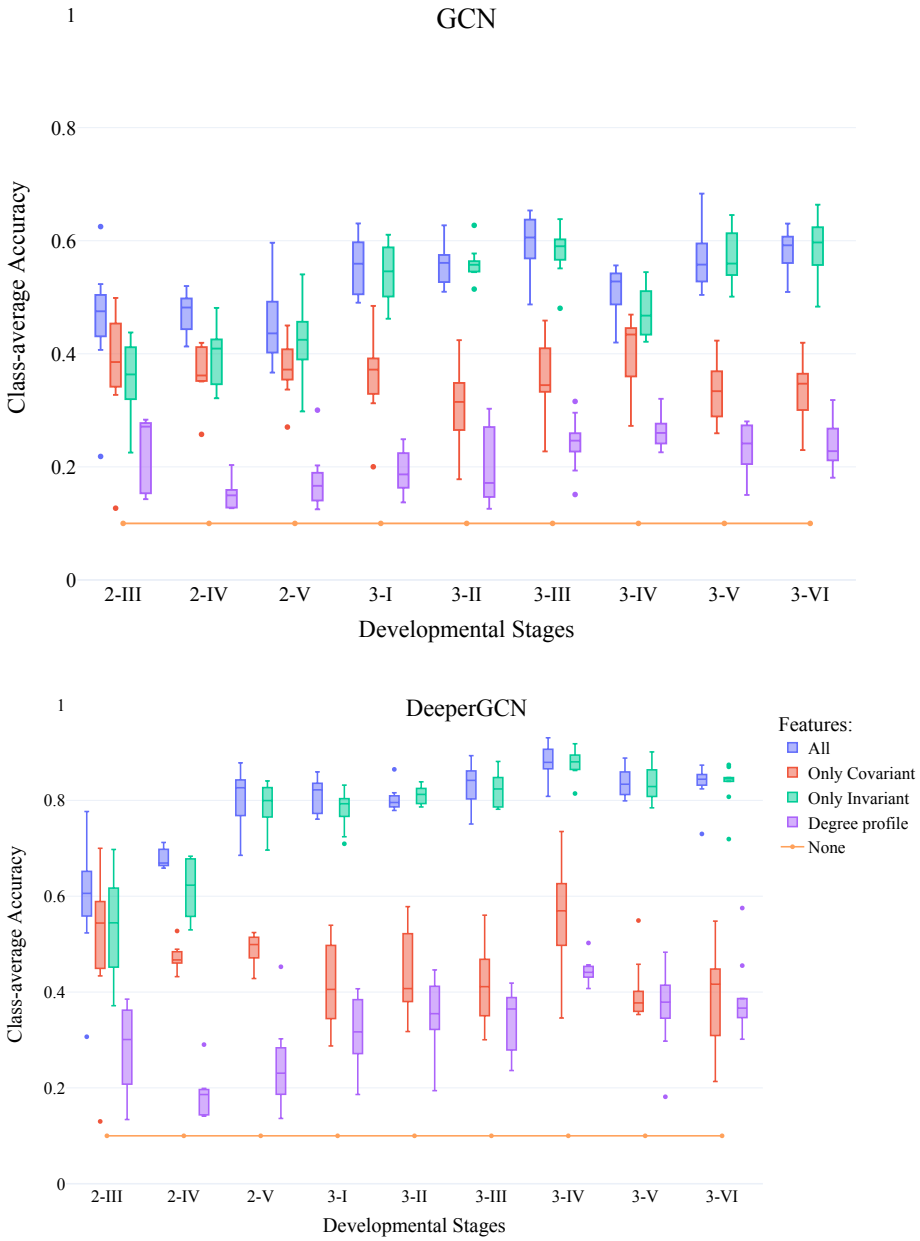


Figure 3.8. Comparison of the class average accuracy for four different feature sets. As one would expect, using all features consistently achieves the highest accuracy. Interestingly, one can observe that the covariant features mostly contribute to accuracy in the early stages (2-III to 2-V). In the later stages (3-I to 3-VI), the invariant features are the major contributors to the overall accuracy. Lastly, even when using only the local degree profile of a node as its feature, the networks are still able to make better than random predictions on the task.

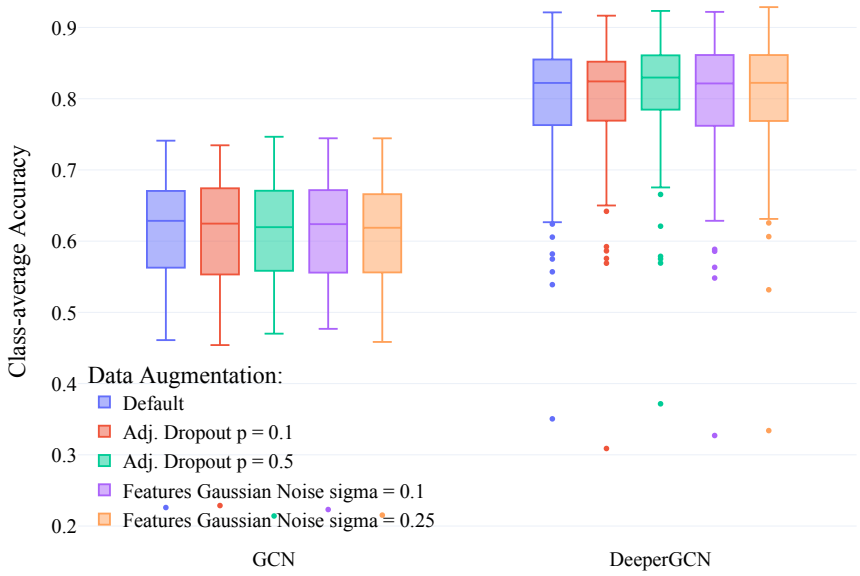


Figure 3.9. Comparison of the class-average accuracy for various combinations of network architecture, data augmentation technique, and parameters. The impact of data augmentation is negligible in all our experiments. Nevertheless, adjacency dropout (edge drop probability = 0.5) consistently improves accuracy for the DeeperGCN model.

Chapter 4

Conclusion

In this thesis, we presented two deep learning frameworks for plant biology applications PlantSeg and CellTypeGraph.

With PlantSeg we introduced a simple to use, yet powerful, software for Instance segmentation. The core pipeline comprises two main elements: a robust U-Net and multiple graph clustering-based segmentation algorithms. Out of the box, PlantSeg contains several pre-trained models for boundary detection for various plant tissues, microscopy modalities, and voxels resolutions. Furthermore, it implements proofreading and training tools for expanding the model pool. Four core segmentation algorithms (DT-Watershed, GASP, MutexWS, and MultiCut) are implemented in PlantSeg, and the robustness and accuracy of these were shown on several external benchmarks (see Sec. 2.2.2, Sec. 2.2.3) and validated by independent studies [195]. In addition, PlantSeg allows users to leverage multi-staining imaging of boundaries and nuclei by providing pre-trained nuclei detection models, the Lifted MultiCut workflow (see Sec. 2.2.4), and automated cell segmentation proofreading from nuclei (see Sec. 2.2.5). As such, PlantSeg has contributed to several biological findings [108, 113, 196–198].

In chapter 3, we introduced CellTypeGraph, a benchmark for cell type classification. The goal of this framework is two-fold, providing a unique benchmark for developing new GNN architectures and methodologies and simultaneously providing new tools for plant developmental biologists to automatize the laborious manual annotation work. To maximize the ease of adoption of CellTypeGraph, we release a ready-to-use PyTorch data loader and a large set of precomputed node and edge features. Furthermore, we provide a strong baseline using state-of-the-art GNNs and offer a study to assess the feature relevance. The positive results presented in Sec. 3.4 have shown that this approach is a viable method for cell type classification in plant organs. Nevertheless, further developments are still required to close the gap with human expert performance on this task.

PlantSeg and CellTypeGraph are contributing to bridging the gap between machine learn-

ing research and life science practitioners. We strongly believe that such type of frameworks will play a central role in future research endeavours.

Appendix A

PlantSeg

A.1 Supplemental figures

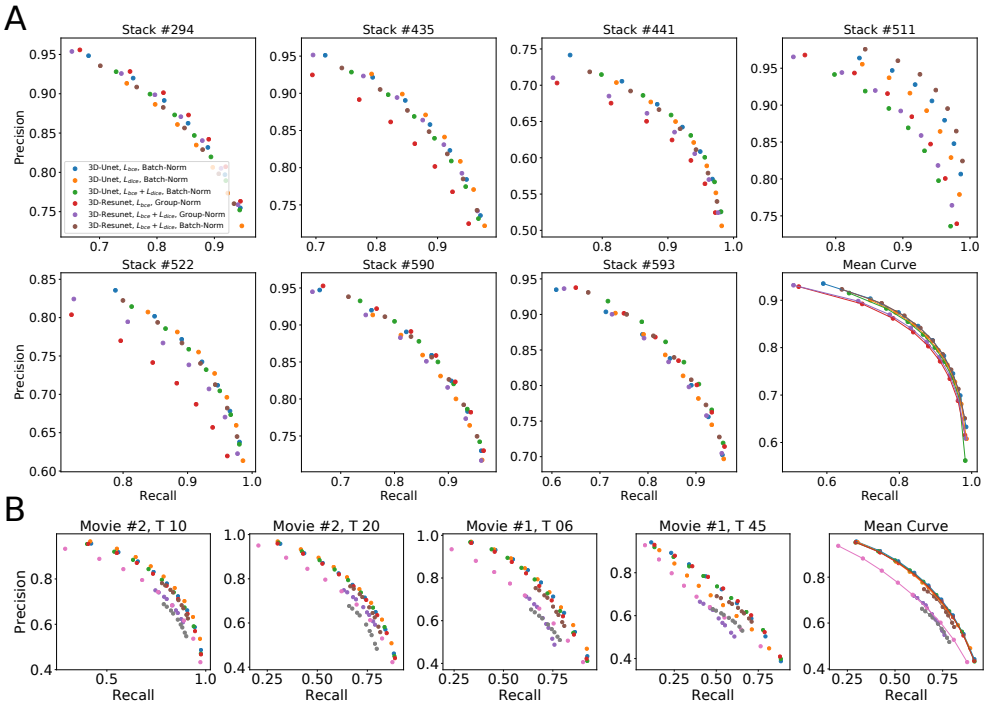


Figure A.1. Precision-recall curves on individual stacks for different CNN variants on the ovule (A) and lateral root primordia (B) datasets. Efficiency of boundary prediction was assessed for seven training procedures that sample different type of architecture (3D U-Net *vs.* 3D Residual U-Net), loss function (BCE *vs.* Dice *vs.* BCE-Dice)) and normalization (Group-Norm (GN) *vs.* Batch-Norm (BN)). The larger the area under the curve, the better the precision. Source files used to generate the precision-recall curves are available in the Appendix 4 Figure 1-source data 1.

A.2 Supplemental tables

Segmentation	ARand	VOI_{split}	VOI_{merge}
Ovules			
DTWS	0.135 ± 0.036	0.585 ± 0.042	**0.320 ± 0.089**
GASP	**0.114 ± 0.059**	**0.357 ± 0.066**	0.354 ± 0.109
MultiCut	0.145 ± 0.080	0.418 ± 0.069	0.429 ± 0.124
Mutex	0.115 ± 0.059	0.359 ± 0.066	0.354 ± 0.108
Lateral Root Primordia			
DTWS	0.550 ± 0.158	1.869 ± 0.174	**0.159 ± 0.073**
GASP	**0.037 ± 0.029**	0.183 ± 0.059	0.237 ± 0.133
MultiCut	**0.037 ± 0.029**	0.190 ± 0.067	0.236 ± 0.128
Lifted Multicut	0.040 ± 0.039	**0.162 ± 0.068**	0.287 ± 0.207
Mutex	0.105 ± 0.118	0.624 ± 0.812	0.542 ± 0.614

Table A.1. Average segmentation accuracy for different segmentation algorithms. The average is computed from a set of seven specimen for the ovules and four for the lateral root primordia (LRP), while the error is measured by standard deviation. The segmentation is produced by multicut, GASP, mutex watershed (Mutex) and DT watershed (DTWS) clustering strategies. We additionally report the scores given by the lifted multicut on the LRP dataset. The Metrics used are the Adapted Rand error to asses the overall segmentation quality, the VOI_{merge} and VOI_{split} respectively assessing erroneous merge and splitting events (lower is better for all metrics). Source files used to create the table are available in the Appendix 5 Table 2-source data 1.

Network and Resolution	Accuracy (%)	Precision	Recall	F 1
Ovules				
3D-Unet, L_{bce}, Group-Norm	97.9 ± 1.0	0.812 ± 0.083	0.884 ± 0.029	0.843 ± 0.044
3D-Unet, L_{bce}, Batch-Norm	**98.0 ± 1.1**	**0.815 ± 0.084**	0.892 ± 0.035	**0.849 ± 0.047**
3D-Unet, L_{dice}, Group-Norm	97.6 ± 1.0	0.765 ± 0.104	0.905 ± 0.023	0.824 ± 0.063
3D-Unet, L_{dice}, Batch-Norm	97.8 ± 1.1	0.794 ± 0.084	**0.908 ± 0.030**	0.844 ± 0.048
3D-Unet, $L_{bce} + L_{dice}$, Group-Norm	97.8 ± 1.1	0.793 ± 0.086	0.907 ± 0.026	0.843 ± 0.048
3D-Unet, $L_{bce} + L_{dice}$, Batch-Norm	97.9 ± 0.9	0.800 ± 0.081	0.898 ± 0.025	0.843 ± 0.041
3D-Resunet, L_{bce}, Group-Norm	97.9 ± 0.9	0.803 ± 0.090	0.880 ± 0.021	0.837 ± 0.050
3D-Resunet, L_{bce}, Batch-Norm	97.9 ± 1.0	0.811 ± 0.081	0.881 ± 0.031	0.841 ± 0.042
3D-Resunet, L_{dice}, Group-Norm	95.9 ± 2.6	0.652 ± 0.197	0.889 ± 0.016	0.730 ± 0.169
3D-Resunet, L_{dice}, Batch-Norm	97.9 ± 1.1	0.804 ± 0.087	0.894 ± 0.035	0.844 ± 0.051
3D-Resunet, $L_{bce} + L_{dice}$, Group-Norm	97.8 ± 1.1	0.812 ± 0.085	0.875 ± 0.026	0.839 ± 0.044
3D-Resunet, $L_{bce} + L_{dice}$, Batch-Norm	**98.0 ± 1.0**	**0.815 ± 0.087**	0.892 ± 0.035	0.848 ± 0.050
Lateral Root Primordia				
3D-Unet, L_{bce}, Group-Norm	97.1 ± 1.0	0.731 ± 0.027	0.648 ± 0.105	0.684 ± 0.070
3D-Unet, L_{bce}, Batch-Norm	97.2 ± 1.0	0.756 ± 0.029	0.637 ± 0.114	0.688 ± 0.080
3D-Unet, L_{dice}, Group-Norm	96.1 ± 1.1	0.587 ± 0.116	0.729 ± 0.094	0.644 ± 0.098
3D-Unet, L_{dice}, Batch-Norm	97.0 ± 0.9	0.685 ± 0.013	0.722 ± 0.103	0.700 ± 0.056
3D-Unet, $L_{bce} + L_{dice}$, Group-Norm	96.9 ± 1.0	0.682 ± 0.029	0.718 ± 0.095	0.698 ± 0.060
3D-Unet, $L_{bce} + L_{dice}$, Batch-Norm	97.0 ± 0.8	0.696 ± 0.012	0.716 ± 0.101	0.703 ± 0.055
3D-Resunet, L_{bce}, Group-Norm	**97.3 ± 1.0**	**0.766 ± 0.039**	0.668 ± 0.089	0.712 ± 0.066
3D-Resunet, L_{bce}, Batch-Norm	97.0 ± 1.1	0.751 ± 0.042	0.615 ± 0.116	0.673 ± 0.086
3D-Resunet, L_{dice}, Group-Norm	96.5 ± 0.9	0.624 ± 0.095	**0.743 ± 0.092**	0.674 ± 0.083
3D-Resunet, L_{dice}, Batch-Norm	97.0 ± 0.9	0.694 ± 0.019	0.724 ± 0.098	0.706 ± 0.055
3D-Resunet, $L_{bce} + L_{dice}$, Group-Norm	97.2 ± 1.0	0.721 ± 0.048	0.735 ± 0.076	**0.727 ± 0.059**
3D-Resunet, $L_{bce} + L_{dice}$, Batch-Norm	97.0 ± 0.9	0.702 ± 0.024	0.703 ± 0.105	0.700 ± 0.063

Table A.2. Ablation study of boundary detection accuracy. Accuracy of boundary prediction was assessed for twelve training procedures that sample different type of architecture (3D U-Net *vs.* 3D Residual U-Net), loss function (BCE *vs.* Dice *vs.* BCE-Dice) and normalization (Group-Norm *vs.* Batch-Norm). All entries are evaluated at a fix threshold of 0.5. Reported values are the means and standard deviations for a set of seven specimen for the ovules and four for the lateral root primordia. Source files used to create the table are available in the Appendix 5 Table 1-source data 1.

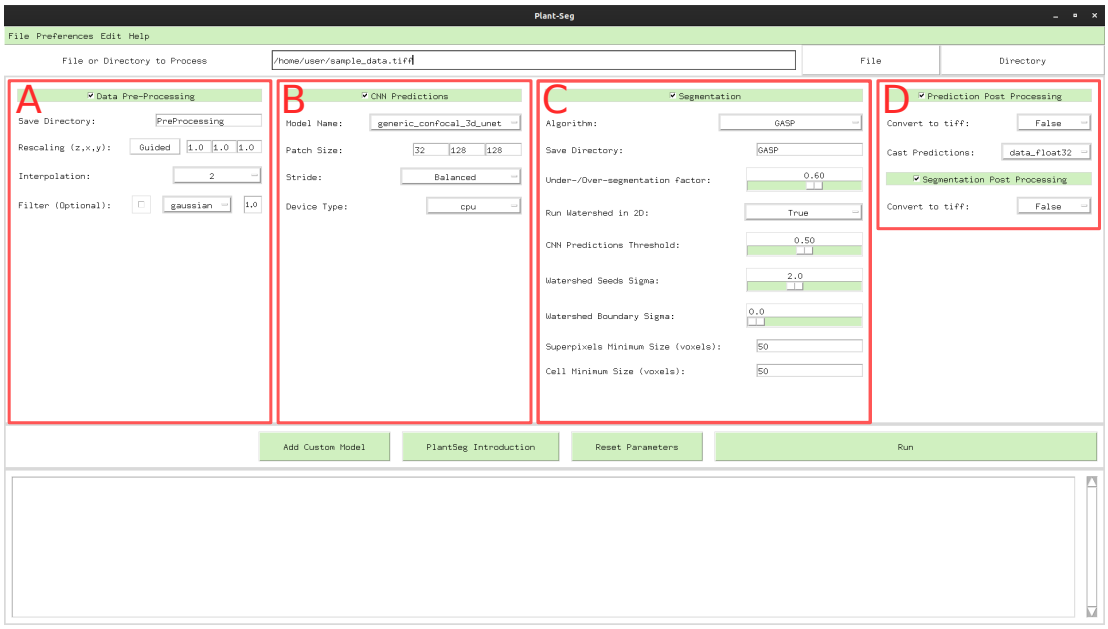


Figure A.2. PlantSeg GUI. The interface allows to configure all execution steps of the segmentation pipeline, such as: selecting the neural network model and specifying hyperparameters of the partitioning algorithm. Tab. A.3 describes the Pre-processing (A) parameters. Tab. A.4 provides parameters guide for the CNN Predictions and Post-processing (B, D). Hyperparameters for Segmentation and Post-processing (C, D) are described in Tab. A.5.

A.3 PlantSeg - parameters guide

The PlantSeg workflow can be customized and optimized by tuning the pipeline's hyperparameters. The large number of available options can be intimidating for new user, therefore we provide a short guide to explain them. For more detailed and up-to-date guidelines please visit the project's GitHub repository: `https://github.com/hci-unihd/plant-seg`.

Process Type	Parameter Name	Range	Default	Description
Data Pre-processing	Save Directory	text	”PreProcessing”	Create a new sub folder where all results will be stored.
	Rescaling (z, y, z)	tuple	$[1.0, 1.0, 1.0]$	The rescaling factor can be used to make the data resolution match the resolution of the dataset used in training. Pressing the ”Guided” button in the GUI a widget will help the user setting up the right rescaling.
	Interpolation	menu	2	Defines order of the spline interpolation. The order 0 is equivalent to nearest neighbour interpolation, 1 is equivalent to linear interpolation and 2 quadratic interpolation.
	Filter	menu	Disabled	Optional: perform Gaussian smoothing or median filtering on the input. Filter has an additional parameter that set the sigma (gaussian) or disc radius (median).

Table A.3. Parameters guide for Data Pre-processing. Menu A in Fig. A.2.

Process Type	Parameter Name	Range	Default	Description
CNN Prediction	Model Name	text	”generic_confocal...”	Trained model name. Models trained on confocal (model name: ”generic_confocal_3D_unet”) and lightsheet (model name: ”generic_confocal_3D_unet”) data as well as their multi-resolution variants are available: More info on available models and importing custom models can be found in the project repository.
	Patch Size	tuple	[32, 128, 128]	Patch size given to the network. A bigger patches cost more memory but can give a slight improvement in performance. For 2D segmentation the Patch size relative to the z axis has to be set to 1.
	Stride	menu	Balanced	Specifies the overlap between neighboring patches. The bigger the overlap the the better predictions at the cost of additional computation time. In the GUI the stride values are automatically set, the user can choose between: Accurate (50% overlap between patches), Balanced (25% overlap between patches), Draft (only 5% overlap between patches).
	Device Type	menu	”cpu”	If a CUDA capable gpu is available and setup correctly, ”cuda” should be used, otherwise one can use ”cpu” for cpu only inference (much slower).
Prediction Post-processing	Convert to tiff	bool	False	If True the prediction is exported as tiff file.
	Cast Predictions	menu	”data_float32”	Predictions stacks are generated in ”float32”. Or ”uint8” can be alternatively used to reduce the memory footprint.

Table A.4. Parameters guide for CNN Predictions and Post-processing. Menu B and D in Fig. A.2.

Process Type	Parameter Name	Range	Default	Description
Segmentation	Algorithm	menu	”GASP”	Defines which algorithm will be used for segmentation.
	Save Directory	text	”GASP”	Create a new sub folder where all results will be stored.
	Under/Over seg. fac.	(0.0...1.0)	0.6	Define the tendency of the algorithm to under of over segment the stack. Small value bias the result towards under-segmentation and large towards over-segmentation.
	Run Watersed in 2D	bool	True	If True the initial superpixels partion will be computed slice by slice, if False in the whole 3D volume at once. While slice by slice drastically improve the speed and memory consumption, the 3D is more accurate.
	CNN Prediction Threshold	(0.0...1.0)	0.5	Define the threshold used for superpixels extraction and Distance Transform Watershed. It has a crucial role for the watershed seeds extraction and can be used similarly to the ”Unde/Over segmentation factor” to bias the final result. An high value translate to less seeds being placed (more under segmentation), while with a low value more seeds are placed (more over segmentation).
	Watershed Seeds Sigma	float	2.0	Defines the amount of smoothing applied to the CNN predictions for seeds extraction. If a value of 0.0 used no smoothing is applied.
	Watershed Boundary Sigma	float	0.0	Defines the amount of Gaussian smoothing applied to the CNN predictions for the seeded watershed segmentation. If a value of 0.0 used no smoothing is applied.
	Superpixels Minimum Size	integer	50	Superpixels smaller than the threshold (voxels) will be merged with a the nearest neighbour segment.
	Cell Minimum Size	integer	50	Cells smaller than the threshold (voxels) will be merged with a the nearest neighbour cell.
Segmentation Post-processing	bool	False	Convert to tiff	If True the segmentation is exported as tiff file.

Table A.5. Parameters guide for Segmentation. Menu C and D in Fig. A.2.

A.4 Empirical example of parameter tuning

PlantSeg's default parameters have been chosen to yield the best result on our core datasets (Ovules, LRP). Furthermore, not only segmentation performances but also resource requirements have been taken into account, e.g. super-pixels are extracted in 2D by default in order to reduce the pipeline runtime and memory usage.
Nevertheless, when applying PlantSeg on a new dataset tuning parameters tuningthe default parameters can lead to improved segmentation results. Unfortunately, image stacks can vary in: imaging acquisition, voxels resolutions, noise, cell size and cell morphology. All those factors make it very hard to define generic guidelines and optimal results will always require some trial and error.

Here we present an example of empirical parameter tuning where we use a 3D Cell Atlas as a test dataset since we can quantitatively evaluate the pipeline's performance on it. We will show how we can improve on top of the results presented in Table 1 of the main manuscript. Since the number of parameters does not allow for a complete grid search we will focus only on three key aspects: model/rescaling, over/under segmentation factor and 3D vs 2D super pixels.

- *model/rescaling*: As we showed in the main text, on this particular data the default confocal CNN model already provides a solid performance. If we now take into consideration the voxel resolution, we have two possible choices. Using a CNN model trained on a more similar resolution or if this is not possible using the rescaling factor to reduce the resolution gap further. In Tab. A.6 we can see how the results vary if we take into consideration those two aspects. In particular we can observe that in this case the rescaling of voxels depth of a factor $3x$ considerably improved the overall scores.

- *over/under segmentation factor*: Now that we have tuned the network predictions we can move to tuning the segmentation performance. The main parameter we can use is the *over/under segmentation factor*, this will try to compensate the over- or under-segmentation. From the results in Tab. A.6 we can observe a strong tendency towards under-segmentation, this suggest that increasing the *over/under segmentation factor* will balance the segmentation. In table Tab. A.7 we can see the results for three different values, increasing the *over/under segmentation factor* as the desired effect and overall improved the results.

- *3D vs 2D super pixels*: Already tuning this two aspects of the pipeline drastically improved the segmentation according to our metrics. as a final tweak we can switch to 3D super pixels to further improve results. In Tab. A.8 we present the final results.

Dataset	generic confocal (Default)			ds3 confocal			ds3 confocal + rescaling		
	ARand	VOI_{split}	VOI_{merge}	ARand	VOI_{split}	VOI_{merge}	ARand	VOI_{split}	VOI_{merge}
Anther	0.328	0.778	0.688	0.344	1.407	0.735	0.265	0.748	0.650
Filament	0.576	1.001	1.378	0.563	1.559	1.244	0.232	0.608	0.601
Leaf	0.075	0.353	0.322	0.118	0.718	0.384	0.149	0.361	0.342
Pedicel	0.400	0.787	0.869	0.395	1.447	1.082	0.402	0.807	1.161
Root	0.248	0.634	0.882	0.219	1.193	0.761	0.123	0.442	0.592
Sepal	0.527	0.746	1.032	0.503	1.293	1.281	0.713	0.652	1.615
Valve	0.572	0.821	1.315	0.617	1.404	1.548	0.586	0.578	1.443
Average	0.389	0.731	0.927	0.394	1.289	1.005	**0.353**	**0.600**	**0.915**

Table A.6. Comparison between the *generic confocal* CNN model (default in PlantSeg), the closest confocal model in terms of xy plant voxels resolution *ds3 confocal* and the combination of *ds3 confocal* and rescaling (in order to mach the training data resolution a rescaling factor of $(3, 1, 1)$ zxy has been used). The later combination showed the best overall results. To be noted that *ds3 confocal* was trained on almost isotropic data, while the 3D Digital Tissue Atlas is not isotropic. Therefore poor performances without rescaling are expected. Segmentation obtained with GASP and default parameters

Overall the final improvement is roughly a factor of $\times 2$ in terms of ARand score compared to PlantSeg default.

Further fine tuning could be performed on the PlantSeg parameters to further improve the scores.

ds3 confocal + rescaling									
Dataset	over/under factor 0.5			over/under factor 0.6 (Default)			over/under factor 0.7		
	ARand	VOI_{split}	VOI_{merge}	ARand	VOI_{split}	VOI_{merge}	ARand	VOI_{split}	VOI_{merge}
Anther	0.548	0.540	1.131	0.265	0.748	0.650	0.215	1.130	0.517
Filament	0.740	0.417	1.843	0.232	0.608	0.601	0.159	0.899	0.350
Leaf	0.326	0.281	0.825	0.149	0.361	0.342	0.117	0.502	0.247
Pedicel	0.624	0.585	2.126	0.402	0.807	1.161	0.339	1.148	0.894
Root	0.244	0.334	0.972	0.123	0.442	0.592	0.113	0.672	0.485
Sepal	0.904	0.494	2.528	0.713	0.652	1.615	0.346	0.926	1.211
Valve	0.831	0.432	2.207	0.586	0.578	1.443	0.444	0.828	1.138
Average	0.602	**0.441**	1.662	0.353	0.600	0.915	**0.248**	0.872	**0.691**

Table A.7. Comparison between the results obtained with three different *over/under segmentation factor* $(0.5, 0.6, 0.7)$. The effect of tuning this parameter is mostly reflected in the VOIs scores. In this case the best result have been obtained by steering the segmentation towards the over segmentation.

ds3 confocal + rescaling								
over/under factor 0.7								
Dataset	Super Pixels 2D (Default)				Super Pixels 3D			
	ARand	VOI_{split}	VOI_{merge}	time (s)	ARand	VOI_{split}	VOI_{merge}	time (s)
Anther	0.215	1.130	0.517	600	0.167	0.787	0.399	2310
Filament	0.159	0.899	0.350	120	0.171	0.687	0.487	520
Leaf	0.117	0.502	0.247	800	0.080	0.308	0.220	3650
Pedicel	0.339	1.148	0.894	450	0.314	0.845	0.604	2120
Root	0.113	0.672	0.485	210	0.101	0.356	0.412	920
Sepal	0.346	0.926	1.211	770	0.257	0.690	0.966	3420
Valve	0.444	0.828	1.138	530	0.300	0.494	0.875	2560
Average	0.248	0.872	0.691	**500**	**0.199**	**0.595**	**0.566**	2210

Table A.8. Comparison between 2D vs 3D super pixels. From out experiments, segmentation quality is almost always improved by the usage of 3D super pixels. On the other side, the user should be aware that this improvement comes at the cost of a large slow-down of the pipeline (roughly $\times 4.5$ on our system Intel Xenon E5-2660, RAM 252Gb).

A.5 Biological material and imaging

Imaging of the *Arabidopsis thaliana* ovules was performed as described in [137]. Imaging of the shoot apical meristem was performed as previously described [199,200] with a confocal laser scanning microscope (Nikon A1, $25\times$ NA=1.1) after staining cell walls with DAPI (0.2mg/ml).

For imaging of *Arabidopsis thaliana* lateral root, seedlings of the line sC111 ($UB10_{pro}$ :: $PIP1,4{-}3{\times}GFP/GATA23_{pro}$ :: $H2B:3{\times}mCherry/DR5v2_{pro}$ :: $3{\times}YFPnls/RPS5A_{pro}$ :: $dtTomato:NLS$, described in [138]) were used at 5 day post germination. Sterilized seeds were germinated on top of 4.5 mm long Blaubrand micropipettes (Cat 708744; 100μl) that were immobilised on the bottom of a petri dish and covered with $\frac{1}{2}$MS-phytagel [201]. Before sowing, the top of the micropipettes is exposed by removing the phytagel with a razor blade and one seed is sowed per micropipette. Plates were stratified for two days and transfered to a growth incubator (23°C, 16h day light). Imaging was performed on a Luxendo MuViSPIM (`https://luxendo.eu/products/muvi-spim/`) equipped with two $10\times$ NA=0.3 for illumination and $40\times$ NA=0.8 for detection. The following settings were used for imaging: image size 2048×2048, exposure time 75ms, channel #1 illumination 488nm 10% power, detection 497-553nm band pass filter, channel #2 illumination 561nm 10% power, detection 610-628nm band pass filter. Stacks encompassing the whole volume of the root were acquired every 30 minutes. Images from the two cameras were fused using the Luxendo Image processing tool and registered to correct any 3D drift using the BigDataProcessor [202] in Fiji [203].

Appendix B

CellTypeGraph

B.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`.

B.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`.

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 ($i = 0$, $i =$ N, i++) **do**
  if 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$_j$.
    surface-axis$_i$ ← directions$_e$
  else
    neighbors$_i$ ← *find-neighbors*(node$_i$, edges) ▷ Find all nodes neighbors of node$_i$.
    N$_i$, count ← *len*(neighbors$_i$), 0
    for $j = 0$, $j =$ N$_i$, j++ **do** ▷ Loop over the neighborhood.
      if hops$_j <$ hops$_i$ **then** ▷ If neighbor is closer to surface.
        e ← *find-edge*(node$_i$, node$_j$, edges) ▷ Find edge id between node$_i$/node$_j$.
        surface-axis$_i$ ← surface-axis$_i$ + directions$_e$ ▷ Average edges normal.
        count = count + 1
      end if
      surface-axis$_i$ ← surface-axis$_i$/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 ($i = 0$, $i = \mathrm{N}$, i++) **do**

$\Theta_{min} \leftarrow 1$

neighbors$_i$ ← *find-neighbors*(node$_i$, edges) ▷ Find all nodes neighbors of node$_i$.

N$_i$ ← *len*(neighbors$_i$)

for $j = 0$, $j = \mathrm{N}_i$, 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 $k = j + 1$, $k = \mathrm{N}_i$, 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** hops$_i$ = hops$_j$ = hops$_k$ **then** ▷ When the angle is minimum the cell are the most co-linear.

$\Theta_{min} \leftarrow \Theta$

growth-axis$_i$ ← vector$_{ij}$

end if

end for

end for

end for

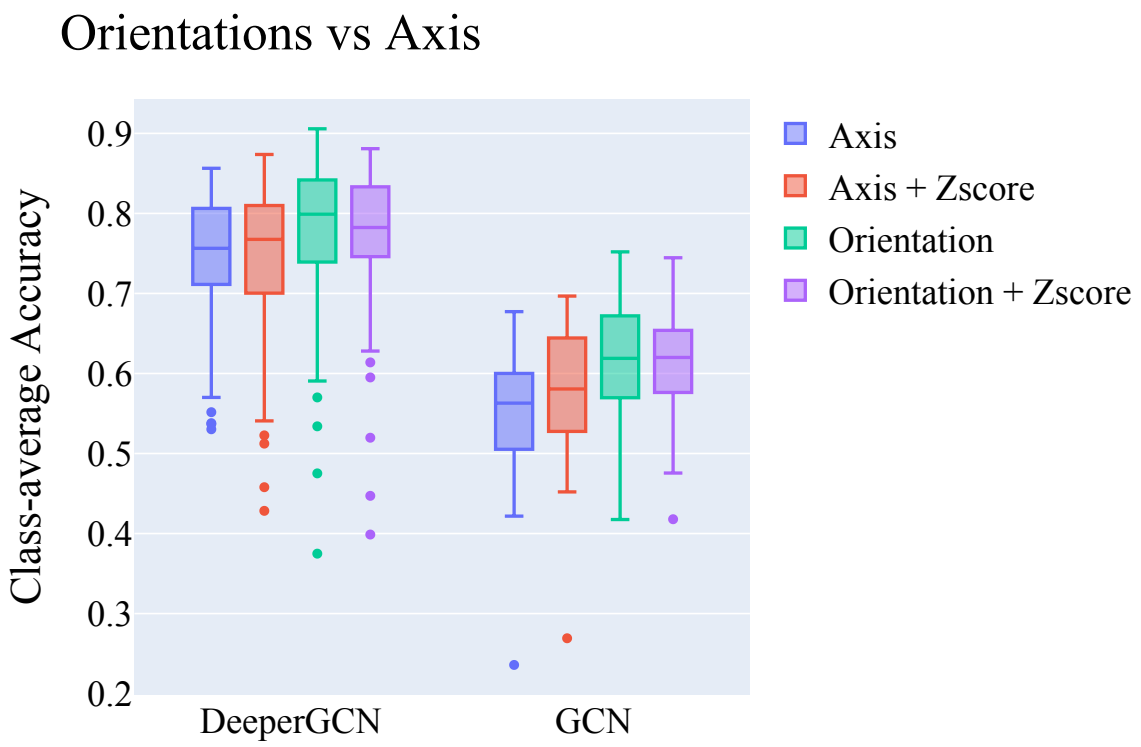


Figure B.1. Class-average accuracy comparison between different vector features representations. Using the orientation instead of axis resulted in a small but consistent improvement in performance.

B.3 Features

We here report the results of additional experiments conducted to identify the best feature homogenization strategy. In Fig. B.1, we tested the difference between different vector representations. In Fig. B.3 and Fig. B.2, we tested the normalizations for the graph features and the maximum value to be used in our hops to surface feature. In Fig. B.4 and Fig. B.5 we tested different normalization applied to respectively morphological and angles features. Lastly, in Fig. B.6 we tested the impact of using lengths measured in several directions as features, similarly to [162]. An overview of all features available in the CellTypeGraph benchmark is reported in Tab. B.1, Tab. B.2. Our python implementation can be found at: `https://github.com/hci-unihd/plant-celltype/tree/main/plantcelltype/features`

B.4 Grid search complete results

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

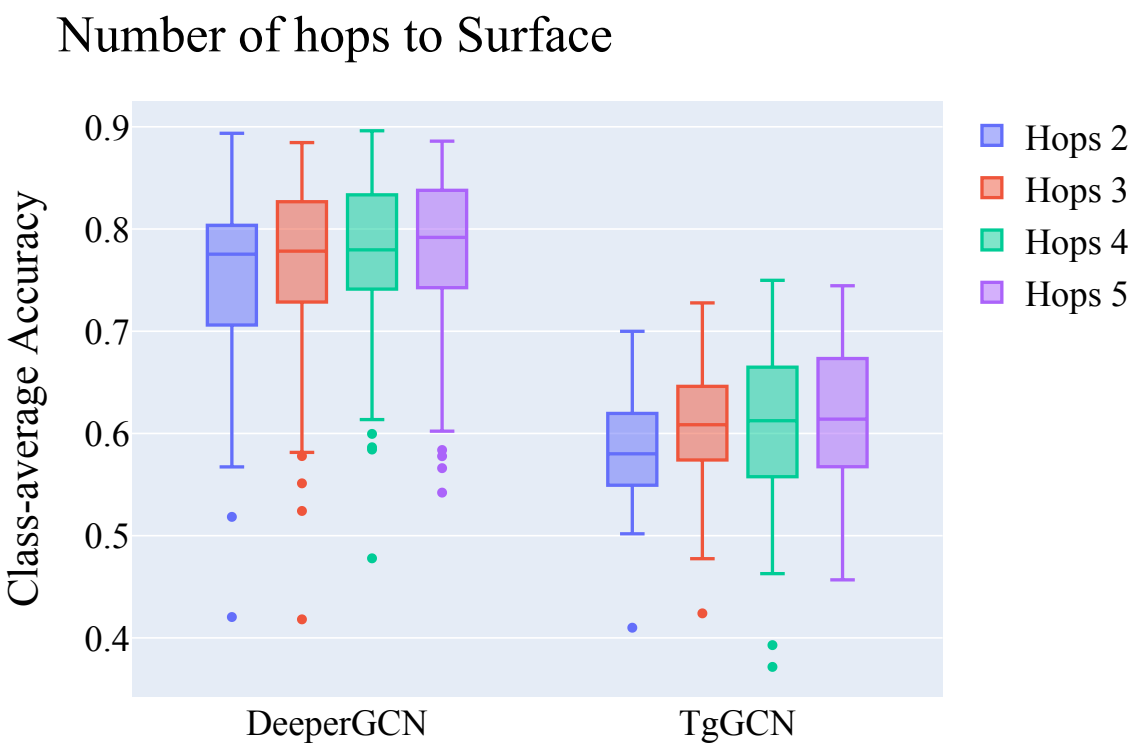


Figure B.2. Class-average accuracy with varying maximum number of hops. Hops to surface are intuitively closely related to the tissue stratification *L1-L4*, but their relation loosen with depth. We tested how important this feature is by clipping its value. From the results one can see that the feature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

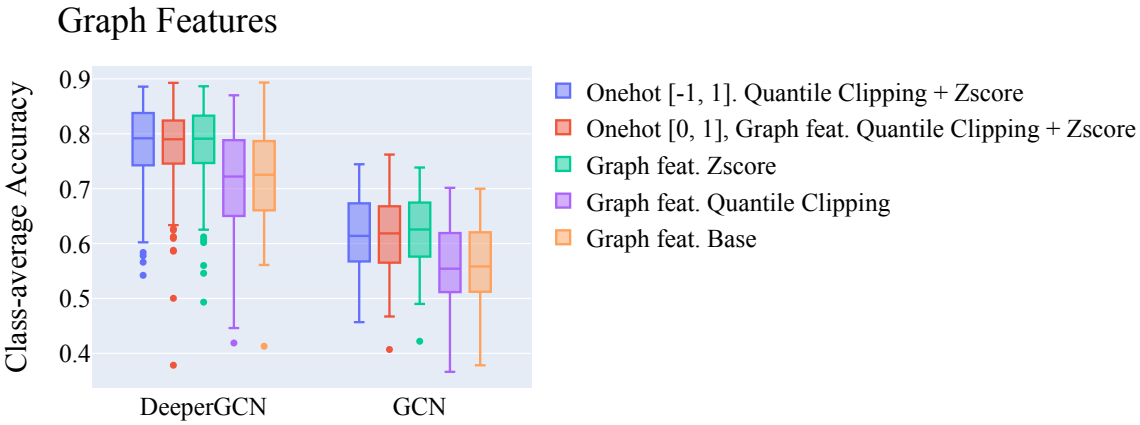


Figure B.3. Class-average accuracy between different graph feature normalizations. One can see a slight accuracy improvement using z-score normalization.

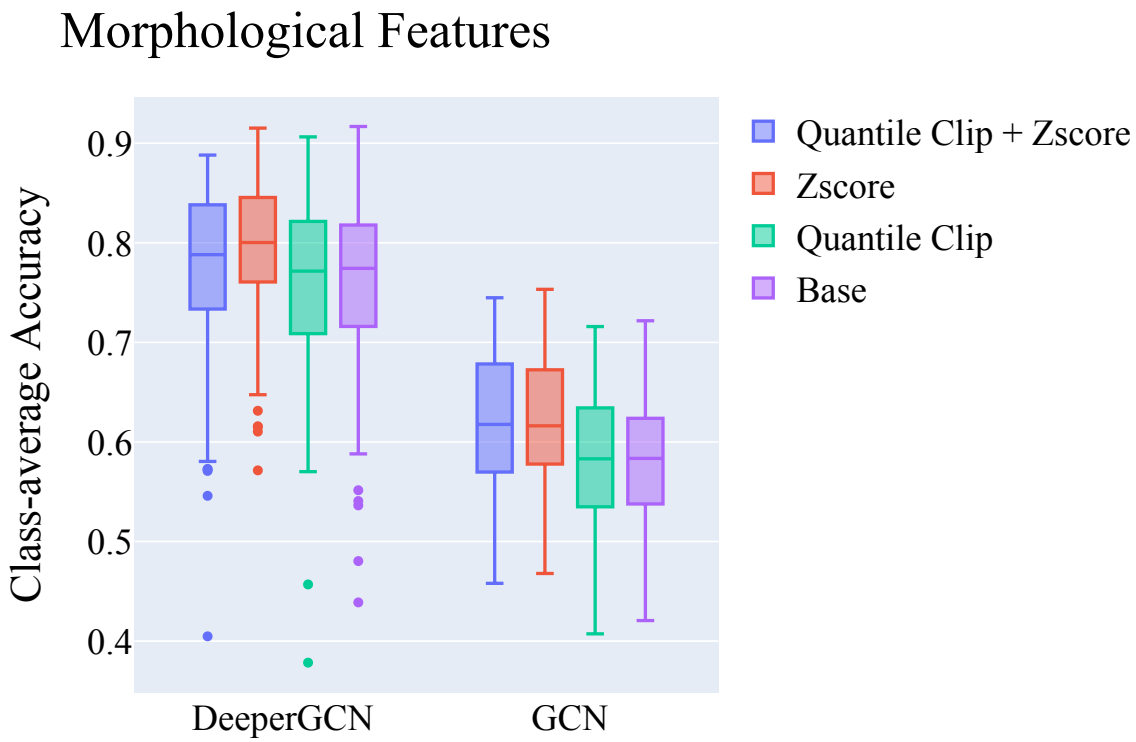


Figure B.4. Class-average accuracy comparison between different morphological features normalizations. One can see a slight accuracy improvement using z-score normalization.

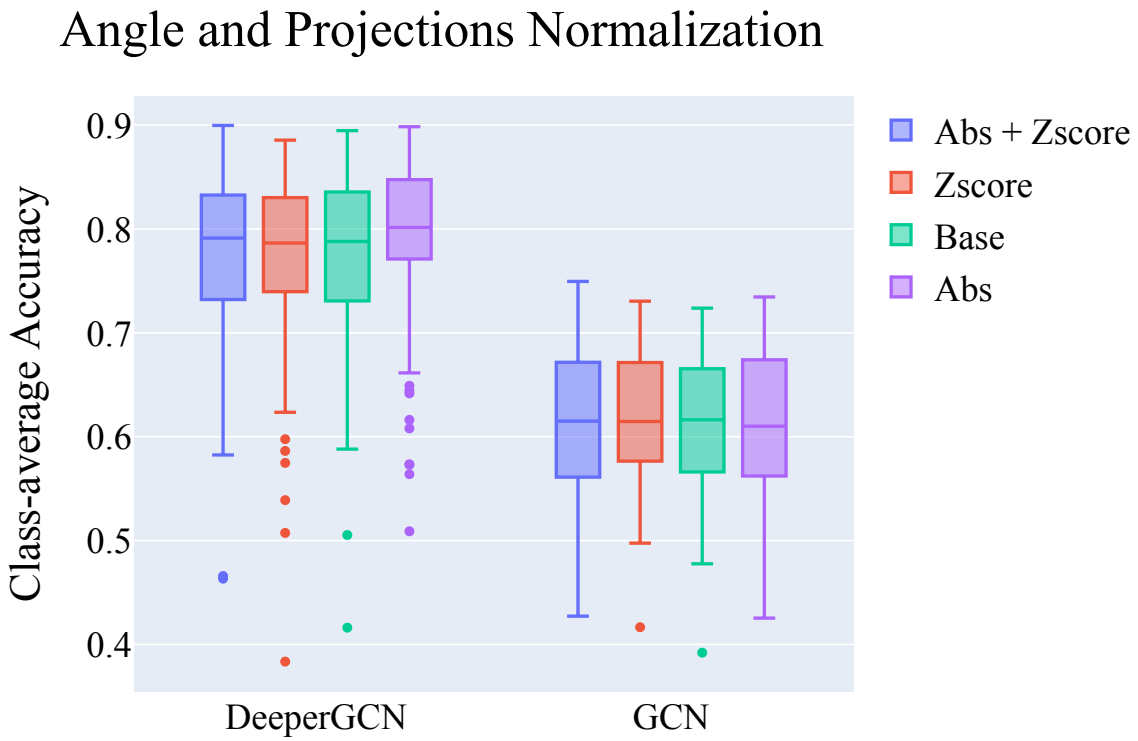


Figure B.5. Class-average accuracy between different angles normalization. Angles are naturally normalized between -1 and 1, in our experiments the z-score had no significant impact.

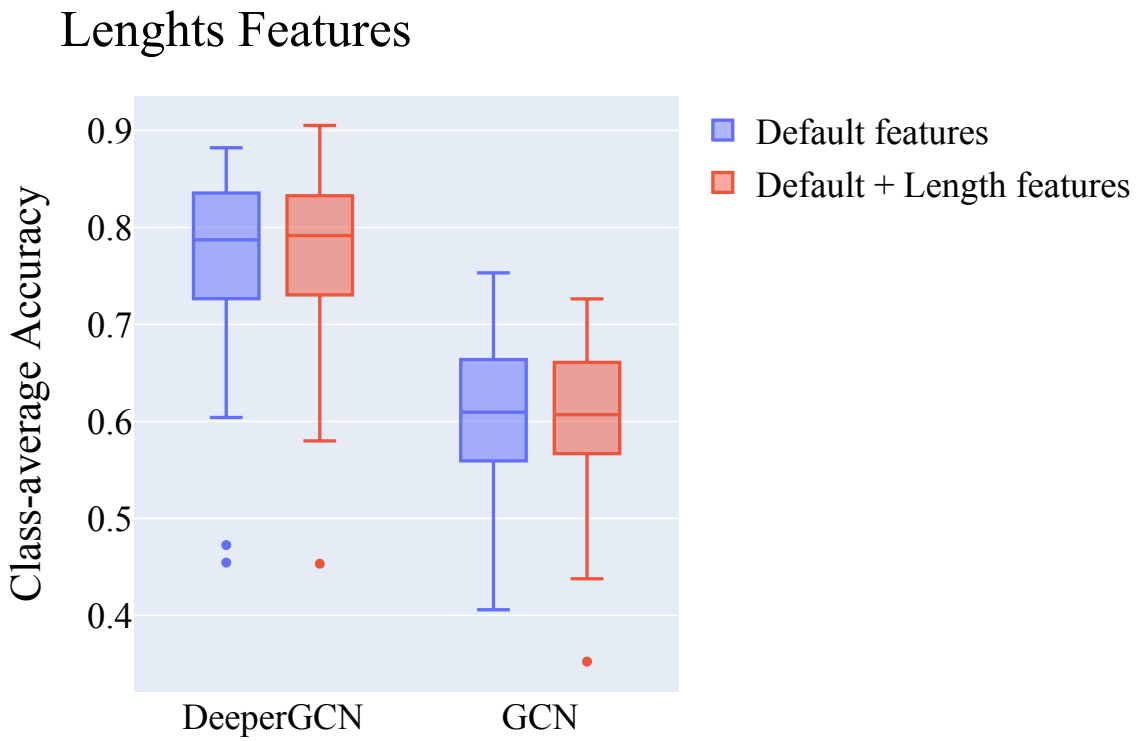


Figure B.6. Class-average accuracy between: baseline features only, and baseline features plus additional lengths features. In our experiments the additional lengths showed no significant contribution to the accuracy.

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 geolocalization of nodes, and consider all neighbors one hop distant.
Degree Centrality	✓	✓	1	Degree centrality.
CFC centrality	✓	✓	1	Current-flow closeness centrality.

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

B.5 Baseline implementation details

In our baseline, we benchmark eight different graph neural network architectures. We report here the most salient implementation details. For *GCN* [83], *GraphSAGE*, [87], *GIN* [90], *GCNII* [92], and *DeeperGCN* [93, 94]; we followed the *pytorch_geometric* implementation [185]. In all the aforementioned architectures the convolutions block are composed as follows: Graph Convolution $\rightarrow$ Relu activation [204] $\rightarrow$ normalization $\rightarrow$ Dropout [14]. 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* [86], *GATv2* [95], and *TransformerGCN* [22, 186], we used graph convolutions as implemented in [185] but with some minor differences in the convolution block layout: Graph Convolution $\rightarrow$ normalization $\rightarrow$ Relu activation [204] $\rightarrow$ Dropout [14]. All source implementation are released at `https://github.com/hci-unihd/plant-celltype/blob/main/plantcelltype/graphnn/graph_models.py`.

Model	Optimizer	Model Params.	top-1 acc.	class-avg. acc.
GIN [90]	lr $= 10^{-2}$ wd $= 10^{-5}$	# feat $= 64$ # layers $= 2$ dropout $= 0.1$	0.714 ± 0.071	0.563 ± 0.136
GCN [83]	lr $= 10^{-2}$ wd $= 10^{-5}$	# feat $= 128$ # layers $= 2$ dropout $= 0.5$	0.762 ± 0.043	0.617 ± 0.077
GAT [86]	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 [95]	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 [87]	lr $= 10^{-3}$ wd $== 10^{-5}$	# feat $= 128$ # layers $= 4$ dropout $= 0.1$	0.859 ± 0.048	0.765 ± 0.093
GCNII [87]	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 [186]	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 [186]	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 [94]	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 [94]	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 B.3. Best performing optimizer and model parameters according to the class-avg. accuracy.

Bibliography

[1] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880, 2009.

[2] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

[3] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[6] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

[7] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[8] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

[9] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

[10] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022. Featured Certification.

[11] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *Advances in neural information processing systems*, 28, 2015.

[14] 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.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[16] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[17] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[18] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[20] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[21] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[22] 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.

[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[25] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[26] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[27] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[28] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8844–8854, 2022.

[29] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[30] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[31] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

[32] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[33] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[35] Dan Ciresan, Alessandro Giusti, Luca Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in neural information processing systems*, 25, 2012.

[36] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[37] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016.

[38] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H Sebastian Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017.

[39] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.

[40] Ali Hatamizadeh, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett Landman, Holger R Roth, and Daguang Xu. Unetr: Transformers for 3d medical image segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 574–584, 2022.

[41] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[42] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[43] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[44] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6409–6418, 2019.

[45] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[46] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9157–9166, 2019.

[47] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. 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*, pages 265–273, 2018.

[48] Martin Weigert, Uwe Schmidt, Robert Haase, Ko Sugawara, and Gene Myers. Star-convex polyhedra for 3d object detection and segmentation in microscopy. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[49] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[50] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[51] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[52] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[53] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8837–8845, 2019.

[54] Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9018–9028, 2018.

[55] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[56] Adrian Wolny, Qin Yu, Constantin Pape, and Anna Kreshuk. Sparse object-level supervision for instance segmentation with pixel embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4402–4411, 2022.

[57] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *Nature methods*, 18(1):100–106, 2021.

[58] Srinivas C Turaga, Kevin L Briggman, Moritz Helmstaedter, Winfried Denk, and H Sebastian Seung. Maximin affinity learning of image segmentation. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, pages 1865–1873, 2009.

[59] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.

[60] Alberto Bailoni, Constantin Pape, Steffen Wolf, Thorsten Beier, Anna Kreshuk, and Fred A Hamprecht. A generalized framework for agglomerative clustering of signed graphs applied to instance segmentation. *arXiv preprint arXiv:1906.11713*, 2019.

[61] Steffen Wolf, Constantin Pape, Alberto Bailoni, Nasim Rahaman, Anna Kreshuk, Ullrich Kothe, and FredA Hamprecht. The mutex watershed: efficient, parameter-free image partitioning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 546–562, 2018.

[62] Steffen Wolf, Alberto Bailoni, Constantin Pape, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A Hamprecht. The mutex watershed and its objective: Efficient, parameter-free graph partitioning. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3724–3738, 2020.

[63] Jörg Hendrik Kappes, Markus Speth, Björn Andres, Gerhard Reinelt, and Christoph Schn. Globally optimal image partitioning by multicuts. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 31–44. Springer, 2011.

[64] Bjoern Andres, Jörg H Kappes, Thorsten Beier, Ullrich Köthe, and Fred A Hamprecht. Probabilistic image segmentation with closedness constraints. In *2011 International Conference on Computer Vision*, pages 2611–2618. IEEE, 2011.

[65] Thorsten Beier, Constantin Pape, Nasim Rahaman, Timo Prange, Stuart Berg, Davi D Bock, Albert Cardona, Graham W Knott, Stephen M Plaza, Louis K Scheffer, et al. Multicut brings automated neurite segmentation closer to human performance. *Nature methods*, 14(2):101, 2017.

[66] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1):89–113, 2004.

[67] Jos B.T.M. Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundam. Inf.*, 41(1,2):187–228, April 2000.

[68] Constantin Pape, Thorsten Beier, Peter Li, Viren Jain, Davi D Bock, and Anna Kreshuk. Solving large multicut problems for connectomics via domain decomposition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–10, 2017.

[69] Thorsten Beier, Thorben Kroeger, Jorg H Kappes, Ullrich Kothe, and Fred A Hamprecht. Cut, glue & cut: A fast, approximate solver for multicut partitioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, 2014.

[70] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[71] Yann Lecun and Yoshua Bengio. *Convolutional networks for images, speech, and time-series*. MIT Press, 1995.

[72] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[73] Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. *Advances in Neural Information Processing Systems*, 32, 2019.

[74] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *International conference on Machine learning*, pages 1321–1330. PMLR, 2019.

[75] Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks, 2022.

[76] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

[77] Lorenz C Blum and Jean-Louis Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database gdb-13. *Journal of the American Chemical Society*, 131(25):8732–8733, 2009.

[78] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[79] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[80] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[81] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98, 1998.

[82] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

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

[84] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[85] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, pages http–openreview, 2014.

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

[87] 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.

[88] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[89] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

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

[91] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 09–15 Jun 2019.

[92] 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.

[93] 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.

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

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

[96] Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[97] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

[98] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[99] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[100] Pablo Barceló, Egor V Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan-Pablo Silva. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations (ICLR 2020)*, 2020.

[101] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020.

[102] Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On weisfeiler-leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42–59, 2020.

[103] Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. *Advances in Neural Information Processing Systems*, 32, 2019.

[104] Thomas D Montenegro-Johnson, Petra Stamm, Soeren Strauss, Alexander T Topham, Michail Tsagris, Andrew TA Wood, Richard S Smith, and George W Bassel. Digital single-cell analysis of plant organ development using 3dcellatlas. *The Plant Cell*, 27(4):1018–1033, 2015.

[105] Lisa Willis, Yassin Refahi, Raymond Wightman, Benoit Landrein, José Teles, Kerwyn Casey Huang, Elliot M Meyerowitz, and Henrik Jönsson. Cell size and growth regulation in the arabidopsis thaliana apical stem cell niche. *Proceedings of the National Academy of Sciences*, 113(51):E8238–E8246, 2016.

[106] Yassin Refahi, Argyris Zardilis, Gaël Michelin, Raymond Wightman, Bruno Leggio, Jonathan Legrand, Emmanuel Faure, Laetitia Vachez, Alessia Armezzani, Anne-Evodie Risson, et al. A multiscale analysis of early flower development in arabidopsis provides an integrated view of molecular regulation and growth control. *Developmental Cell*, 56(4):540–556, 2021.

[107] Lilan Hong, Mathilde Dumond, Mingyuan Zhu, Satoru Tsugawa, Chun-Biu Li, Arezki Boudaoud, Olivier Hamant, and Adrienne HK Roeder. Heterogeneity and robustness in plant morphogenesis: from cells to organs. *Annual review of plant biology*, 69:469–495, 2018.

[108] Adrian Wolny, Lorenzo Cerrone, Athul Vijayan, Rachele Tofanelli, Amaya Vilches Barro, Marion Louveaux, Christian Wenzl, Sören Strauss, David Wilson-Sánchez, Rena Lymbouridou, et al. Accurate and versatile 3d segmentation of plant tissues at cellular resolution. *Elife*, 9:e57613, 2020.

[109] Lorenzo Cerrone, Athul Vijayan, Tejasvinee Mody, Kay Schneitz, and Fred A Hamprecht. Celltypegraph: A new geometric computer vision benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20897–20907, 2022.

[110] Lorenzo Cerrone, Alexander Zeilmann, and Fred A. Hamprecht. End-to-end learned random walker for seeded image segmentation. *CoRR*, abs/1905.09045, 2019.

[111] Athul Vijayan, Rachele Tofanelli, Sören Strauss, Lorenzo Cerrone, Adrian Wolny, Joanna Strohmeier, Anna Kreshuk, Fred A Hamprecht, Richard S Smith, and Kay Schneitz. A digital 3D reference atlas reveals cellular growth patterns shaping the arabidopsis ovule. *Elife*, 10:e63262, 2021.

[112] Constantin Pape, Roman Remme, Adrian Wolny, Sylvia Olberg, Steffen Wolf, Lorenzo Cerrone, Mirko Cortese, Severina Klaus, Bojana Lucic, Stephanie Ullrich, et al. Microscopy-based assay for semi-quantitative detection of sars-cov-2 specific antibodies in human sera: A semi-quantitative, high throughput, microscopy-based assay expands existing approaches to measure sars-cov-2 specific antibody levels in human sera. *Bioessays*, 43(3):2000257, 2021.

[113] Lilli Marie Schütz, Marion Louveaux, Amaya Vilches Barro, Sami Bouziri, Lorenzo Cerrone, Adrian Wolny, Anna Kreshuk, Fred A Hamprecht, and Alexis Maizel. Integration of cell growth and asymmetric division during lateral root initiation in arabidopsis thaliana. *Plant and Cell Physiology*, 62(8):1269–1279, 2021.

[114] Jan Funke, Fabian Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. Large scale image segmentation with structured loss based

deep learning for connectome reconstruction. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1669–1680, 2018.

[115] Romain Fernandez, Pradeep Das, Vincent Mirabet, Eric Moscardi, Jan Traas, Jean-Luc Verdeil, Grégoire Malandain, and Christophe Godin. Imaging plant growth in 4d: robust tissue reconstruction and lineaging at cell resolution. *Nature Methods*, 7(7):547–553, 2010.

[116] Johannes Stegmaier, Fernando Amat, William C Lemon, Katie McDole, Yinan Wan, George Teodoro, Ralf Mikut, and Philipp J Keller. Real-time three-dimensional cell segmentation in large-scale microscopy data of developing embryos. *Developmental cell*, 36(2):225–240, 2016.

[117] Tamily A. Weissman and Y. Albert Pan. Brainbow: New Resources and Emerging Biological Applications for Multicolor Genetic Labeling and Analysis. *Genetics*, 199(2):293–306, February 2015. 00056.

[118] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[119] A. Lucchi, K. Smith, R. Achanta, G. Knott, and P. Fua. Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE Transactions on Medical Imaging*, 31(2):474–486, Feb 2012.

[120] Iasonas Kokkinos. Pushing the boundaries of boundary detection using deep learning. In *ICLR 2016*, 2015.

[121] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. *CoRR*, abs/1504.06375, 2015.

[122] Srinivas C. Turaga, Joseph F. Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H. Sebastian Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 22(2):511–538, 2010.

[123] Juan Nunez-Iglesias, Ryan Kennedy, Stephen Plaza, Anirban Chakraborty, and William Katz. Graph-based active learning of agglomeration (gala): a python library to segment 2d and 3d neuroimages. *Frontiers in Neuroinformatics*, 8:34, 2014.

[124] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1384–1399, July 2011.

[125] Dennis Eschweiler, Thiago Vallin Spina, Rohan C. Choudhury, Elliot Meyerowitz, Alexandre Cunha, and Johannes Stegmaier. Cnn-based preprocessing to optimize watershed-based cell segmentation in 3d confocal microscopy images. *CoRR*, abs/1810.06933, 2018.

[126] Weikang Wang, David A. Taft, Yi-Jiun Chen, Jingyu Zhang, Callen T. Wallace, Min Xu, Simon C. Watkins, and Jianhua Xing. Learn to segment single cells with deep distance estimator and deep cell detector. *Computers in Biology and Medicine*, 108:133 – 141, 2019.

[127] Erick Moen, Dylan Bannon, Takamasa Kudo, William Graf, Markus Covert, and David Van Valen. Deep learning for cellular image analysis. *Nature Methods*, 16(12):1233–1246, 2019.

[128] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, et al. U-net: deep learning for cell counting, detection, and morphometry. *Nature methods*, 16(1):67, 2019.

[129] Matthias G. Haberl, Christopher Churas, Lucas Tindall, Daniela Boassa, Sébastien Phan, Eric A. Bushong, Matthew Madany, Raffi Akay, Thomas J. Deerinck, Steven T. Peltier, and Mark H. Ellisman. Cdeep3m–plug-and-play cloud-based deep learning for image segmentation. *Nature Methods*, 15(9):677–680, 2018.

[130] David A. Van Valen, Takamasa Kudo, Keara M. Lane, Derek N. Macklin, Nicolas T. Quach, Mialy M. DeFelice, Inbal Maayan, Yu Tanouchi, Euan A. Ashley, and Markus W. Covert. Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. *PLOS Computational Biology*, 12(11):1–24, 11 2016.

[131] Stuart Berg, Dominik Kutra, Thorben Kroeger, Christoph N. Straehle, Bernhard X. Kausler, Carsten Haubold, Martin Schiegg, Janez Ales, Thorsten Beier, Markus Rudy, Kemal Eren, Jaime I. Cervantes, Buote Xu, Fynn Beuttenmueller, Adrian Wolny, Chong Zhang, Ullrich Koethe, Fred A. Hamprecht, and Anna Kreshuk. ilastik: interactive machine learning for (bio)image analysis. *Nature Methods*, 16(12):1226–1232, December 2019.

[132] Kevin Briggman, Winfried Denk, Sebastian Seung, Moritz N Helmstaedter, and Srinivas C Turaga. Maximin affinity learning of image segmentation. In *Advances in Neural Information Processing Systems*, pages 1865–1873, 2009.

[133] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sébastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *CoRR*, abs/1707.03237, 2017.

[134] Yuxin Wu and Kaiming He. Group normalization. *CoRR*, abs/1803.08494, 2018.

[135] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[136] Marina Meilă. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98(5):873 – 895, 2007.

[137] Rachele Tofanelli, Athul Vijayan, Sebastian Scholz, and Kay Schneitz. Protocol for rapid clearing and staining of fixed arabidopsis ovules for improved imaging by confocal laser scanning microscopy. *Plant Methods*, 15(1):120, 2019.

[138] Amaya Vilches Barro, Dorothee Stöckle, Martha Thellmann, Paola Ruiz-Duarte, Lotte Bald, Marion Louveaux, Patrick von Born, Philipp Denninger, Tatsuaki Goh, Hidehiro Fukaki, Joop E. M. Vermeer, and Alexis Maizel. Cytoskeleton dynamics are necessary for early events of lateral root initiation in arabidopsis. *Current Biology*, 29(15):2443–2454.e5, 2019. 00000.

[139] A. Horé and D. Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, Aug 2010.

[140] George Bassel. Arabidopsis 3d digital tissue atlas, Feb 2019.

[141] Samantha Fox, Paul Southam, Florent Pantin, Richard Kennaway, Sarah Robinson, Giulia Castorina, Yara E. Sánchez-Corrales, Robert Sablowski, Jordi Chan, Verônica Grieneisen, Athanasius F. M. Marée, J. Andrew Bangham, and Enrico Coen. Spatiotemporal coordination of cell division and growth during organ morphogenesis. *PLOS Biology*, 16(11):1–48, 11 2018.

[142] Bradley Lowekamp, David Chen, Luis Ibanez, and Daniel Blezek. The design of simpleitk. *Frontiers in Neuroinformatics*, 7:45, 2013.

[143] J Funke, L Mais, A Champion, N Dye, and D Kainmueller. A benchmark for epithelial cell tracking. *Computer Vision – ECCV 2018 Workshops*, 2019.

[144] B. Aigouy, D. Umetsu, and S. Eaton. Segmentation and quantitative analysis of epithelial tissues. *Dahmann C. (eds) Drosophila. Methods in Molecular Biology, vol 1478. Humana Press, New York, NY*, 2016.

[145] Jan Funke, Fabian Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C. Turaga. A deep structured learning approach towards automating connectome reconstruction from 3d electron micrographs. *CoRR*, abs/1709.02974, 2017.

[146] Michal Januszewski, Jeremy Maitin-Shepard, Peter Li, Jörgen Kornfeld, Winfried Denk, and Viren Jain. Flood-filling networks. *CoRR*, abs/1611.00421, 2016.

[147] Florian Jug, Evgeny Levinkov, Corinna Blasse, Eugene W. Myers, and Bjoern Andres. Moral lineage tracing. *CoRR*, abs/1511.05512, 2015.

[148] Markus Rempfler, Jan-Hendrik Lange, Florian Jug, Corinna Blasse, Eugene W. Myers, Bjoern H. Menze, and Bjoern Andres. Efficient algorithms for moral lineage tracing. *CoRR*, abs/1702.04111, 2017.

[149] Martin Maška, Vladimír Ulman, David Svoboda, Pavel Matula, Petr Matula, Cristina Ederra, Ainhoa Urbiola, Tomás España, Subramanian Venkatesan, Deepak M.W. Balak, Pavel

Karas, Tereza Bolcková, Markéta Štreitová, Craig Carthel, Stefano Coraluppi, Nathalie Harder, Karl Rohr, Klas E. G. Magnusson, Joakim Jaldén, Helen M. Blau, Oleh Dzyubachyk, Pavel Křížek, Guy M. Hagen, David Pastor-Escuredo, Daniel Jimenez-Carretero, Maria J. Ledesma-Carbayo, Arrate Muñoz-Barrutia, Erik Meijering, Michal Kozubek, and Carlos Ortiz-de Solorzano. A benchmark for comparison of cell tracking algorithms. *Bioinformatics*, 30(11):1609–1617, 02 2014.

[150] Constantin Pape, Alex Matskevych, Adrian Wolny, Julian Hennies, Giulia Mizzon, Marion Louveaux, Jacob Musser, Alexis Maizel, Detlev Arendt, and Anna Kreshuk. Leveraging Domain Knowledge to Improve Microscopy Image Segmentation With Lifted Multicuts. *Frontiers in Computer Science*, 1, 2019. 00000.

[151] Andrea Horňáková, Jan-Hendrik Lange, and Bjoern Andres. Analysis and optimization of graph decompositions by lifted multicuts. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1539–1548. JMLR.org, 2017.

[152] Nicholas Sofroniew, Talley Lambert, Kira Evans, Juan Nunez-Iglesias, Grzegorz Bokota, Philip Winston, Gonzalo Peña-Castellanos, Kevin Yamauchi, Matthias Bussonnier, Draga Doncila Pop, Ahmet Can Solak, Ziyang Liu, Pam Wadhwa, Alister Burt, Genevieve Buckley, Andrew Sweet, Lukasz Migas, Volker Hilsenstein, Lorenzo Gaifas, Jordão Bragantini, Jaime Rodríguez-Guerra, Hector Muñoz, Jeremy Freeman, Peter Boone, Alan Lowe, Christoph Gohlke, Loic Royer, Andrea PIERRÉ, Hagai Har-Gil, and Abigail McGovern. napari: a multi-dimensional image viewer for Python, November 2022. If you use this software, please cite it using these metadata.

[153] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[154] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, and Tony and Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, June 2014.

[155] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[156] Nanne van Noord and Eric Postma. Learning scale-variant and scale-invariant features for deep image classification. *Pattern Recognition*, 61:583 – 592, 2017.

[157] Philipp Hanslovsky, Vanessa Leite, Stephan Saalfeld, Igor Pisarev, Jan Funke, Tobias Pietzsch, Ulrik Günther, John Bogovic, Uwe Schmidt, and Juan Nunez-Iglesias. saalfeldlab/paintera paintera-0.20.1, September 2019.

[158] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[159] CREMI. Cremi. miccai challenge on circuit reconstruction from electron microscopy images, 2017. `https://cremi.org`, 2017.

[160] Marina Meilă. Comparing clusterings: an axiomatic view. In *Proceedings of the 22nd international conference on Machine learning*, pages 577–584. ACM, 2005.

[161] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1744–1757, Oct 2010.

[162] 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.

[163] Thomas Montenegro-Johnson, Soeren Strauss, Matthew DB Jackson, Liam Walker, Richard S Smith, and George W Bassel. 3DCellAtlas meristem: a tool for the global cellular annotation of shoot apical meristems. *Plant methods*, 15(1):1–9, 2019.

[164] Alexander D Weston, Panagiotis Korfiatis, Timothy L Kline, Kenneth A Philbrick, Petro Kostandy, Tomas Sakinis, Motokazu Sugimoto, Naoki Takahashi, and Bradley J Erickson. Automated abdominal segmentation of CT scans for body composition analysis using deep learning. *Radiology*, 290(3):669–679, 2019.

[165] Hyunkwang Lee, Fabian M Troschel, Shahein Tajmir, Georg Fuchs, Julia Mario, Florian J Fintelmann, and Synho Do. Pixel-level deep segmentation: artificial intelligence quantifies muscle on computed tomography for body morphometric analysis. *Journal of digital imaging*, 30(4):487–498, 2017.

[166] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2018.

[167] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[168] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[169] Hastie Trevor, Tibshirani Robert, and Jerome H Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer, 2017.

[170] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[171] Mingguo He, Zhewei Wei, Hongteng Xu, et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems*, 34, 2021.

[172] Tomasz Danel, Przemysław Spurek, Jacek Tabor, Marek Śmieja, Łukasz Struski, Agnieszka Słowik, and Łukasz Maziarka. Spatial graph convolutional networks. In *International Conference on Neural Information Processing*, pages 668–675. Springer, 2020.

[173] Pim De Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. In *International Conference on Learning Representations*, 2020.

[174] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.

[175] Oliver T Unke and Markus Meuwly. Physnet: A neural network for predicting energies, forces, dipole moments, and partial charges. *Journal of chemical theory and computation*, 15(6):3678–3693, 2019.

[176] Sören Strauss, Adam Runions, Brendan Lane, Dennis Eschweiler, Namrata Bajpai, Nicola Trozzi, Anne-Lise Routier-Kierzkowska, Saiko Yoshida, Sylvia Rodrigues da Silveira, Athul Vijayan, et al. MorphoGraphX 2.0: Providing context for biological image analysis with positional information. *bioRxiv*, 2021.

[177] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[178] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR, 2020.

[179] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32:7092–7101, 2019.

[180] Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9323–9332. PMLR, 18–24 Jul 2021.

[181] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.

[182] Karen Stephenson and Marvin Zelen. Rethinking centrality: Methods and examples. *Social networks*, 11(1):1–37, 1989.

[183] Hassler Whitney. The self-intersections of a smooth n-manifold in 2n-space. *Annals of Mathematics*, pages 220–246, 1944.

[184] Masahisa Adachi. *Embeddings and immersions*. American Mathematical Soc., 2012.

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

[186] 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.

[187] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[188] Chen Cai and Yusu Wang. A simple yet effective baseline for non-attributed graph classification. *arXiv preprint arXiv:1811.03508*, 2018.

[189] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[190] Amin Salehi and Hasan Davulcu. Graph attention auto-encoders. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 989–996. IEEE, 2020.

[191] Matthew Amodio, David Van Dijk, Krishnan Srinivasan, William S Chen, Hussein Mohsen, Kevin R Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha Venkataswamy, et al. Exploring single-cell data with deep multitasking neural networks. *Nature methods*, 16(11):1139–1145, 2019.

[192] Romain Lopez, Jeffrey Regier, Michael Cole, Michael Jordan, and Nir Yosef. A deep generative model for single-cell rna sequencing with application to detecting differentially expressed genes. *arXiv preprint arXiv:1710.05086*, 2017.

[193] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[194] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

[195] Anuradha Kar, Manuel Petit, Yassin Refahi, Guillaume Cerutti, Christophe Godin, and Jan Traas. Benchmarking of deep learning algorithms for 3d instance segmentation of confocal image datasets. *PLoS computational biology*, 18(4):e1009879, 2022.

[196] Stefan Mielke, Marlene Zimmer, Mukesh Kumar Meena, René Dreos, Hagen Stellmach, Bettina Hause, Cătălin Voiniciuc, and Debora Gasperini. Jasmonate biosynthesis arising from altered cell walls is prompted by turgor-driven mechanical compression. *Science Advances*, 7(7):eabf0356, 2021.

[197] Takafumi Ichikawa, Hui Ting Zhang, Laura Panavaite, Anna Erzberger, Dimitri Fabrèges, Rene Snajder, Adrian Wolny, Ekaterina Korotkevich, Nobuko Tsuchida-Straeten, Lars Hufnagel, et al. An ex vivo system to study cellular dynamics underlying mouse peri-implantation development. *Developmental cell*, 57(3):373–386, 2022.

[198] Ravian L van Ineveld, Michiel Kleinnijenhuis, Maria Alieva, Sam de Blank, Mario Barrera Roman, Esmée J van Vliet, Clara Martínez Mir, Hannah R Johnson, Frank L Bos, Raimond Heukers, et al. Revealing the spatio-phenotypic patterning of cells in healthy and tumor tissues with mlsr-3d and stapl-3d. *Nature biotechnology*, 39(10):1239–1245, 2021.

[199] Daniel von Wangenheim, Gabor Daum, Jan U Lohmann, Ernst K Stelzer, and Alexis Maizel. Live imaging of Arabidopsis development. *Methods Mol Biol*, 1062:539–550, January 2014.

[200] Marcus G. Heisler and Carolyn Ohno. Live-Imaging of the Arabidopsis Inflorescence Meristem. In José Luis Riechmann and Frank Wellmer, editors, *Flower Development: Methods and Protocols*, Methods in Molecular Biology, pages 431–440. Springer, New York, NY, 2014.

[201] Alexis Maizel, Daniel von Wangenheim, Fernán Federici, Jim Haseloff, and Ernst HK Stelzer. High-resolution live imaging of plant growth in near physiological bright conditions using light sheet fluorescence microscopy. *The Plant Journal*, 68(2):377–385, 2011. 00134.

[202] Christian Tischer, Nils Norlin, and Rainer Pepperkok. BigDataProcessor: Fiji plugin for big image data inspection and processing, February 2019.

[203] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, Jean-Yves Tinevez, Daniel James White, Volker Hartenstein, Kevin Eliceiri, Pavel Tomancak, and

Albert Cardona. Fiji: an open-source platform for biological-image analysis. *Nature Methods*, 9(7):676–682, June 2012.

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

List of Figures

List of Tables

List of Algorithms