Procedural Tree Generation

Inverse modelling of 2D trees using graph neural networks

Erfan Mozafari

Supervisor(s): Elmar Eisemann, Petr Kellnhofer

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 28, 2024

Name of the student: Erfan Mozafari
Final project course: CSE3000 Research Project
Thesis committee: Elmar Eisemann, Petr Kellnhofer, Marcel Reinders

Abstract

The most established and widely used methods for analysing tree images for tasks such as geometry analysis, segmentation and classification often rely on pixels. In this paper, the applicability of analyzing tree geometry based on a graph representation rather than a pixel-based approach is pursued. To do so, 2D renders of different species of trees are converted to spatial graph structures capturing significant points on the tree skeleton. Two independent Graph Convolutional Network algorithms which learn node (coordinate) features are then applied on the obtained dataset to assess the reliability of graph based analysis. The first experiment explores a GCN for assigning correct species labels to the skeleton graph of the original tree image, demonstrating the association between geometry and tree metadata. The second experiment, an unsupervised representation learning, is conducted by using Graph Autoencoders to obtain an embedding for each skeleton graph which can be used to reconstruct partially the same graph, demonstrating the association between GCE latent representation and geometry. Promising results were found in both cases, reinforcing the reliability of the original proposition to rely on geometry as well as pixels for tree analysis tasks.

1 Introduction

Trees are inspiring creatures and they have inspired computer scientists to the extent that one of the most important data structures in the field is called a tree. In nature, they seem to grow chaotically from a structure, which makes them tricky to model. In this paper's context of two dimensional tree analysis, attempts at processing tree images to for instance discover recursive patterns or classify trees often rely on pixels of the tree as input. This approach has been proven to get the job done. However, this paper tries to picture the whys to instead model trees with a tree (graph) structure while demonstrating the applicability and the reasonableness of doing so.

The main research question explored in this paper is the following:

- **RQ:** How effective and reasonable is modelling tree geometry using graphs and Graph Convolutional Networks?

To this end, two subquestions are posed:

- **SQ1:** Can spatial graphs composed of point coordinates sampled from skeletonised tree images be used for supervised learning tasks, showing the possibility to go from geometrical representation to tree metadata ?
- **SQ2:** Can (potentially sparse) spatial tree graphs be converted to low dimensional vectors holding significant geometrical information capable of partially reconstructing the original graph ?

In the following sections papers related to our aims are mentioned; Important definitions are laid out. The details of two different experiments aiming to answer the two sub questions are explained and their results are reported and analysed.

2 Related Work

According to Okura [13], plant modelling/reconstruction techniques are categorized into 5 groups: FSPM/L-system, Graphs, Mesh, Voxels, and Point cloud. Our work paper draws inspiration from both graph and point cloud representation.

2.1 Plant modeling and reconstruction techniques

Graph Represenation

Concerning graph represenations, some (botanical) tree reconstruction methods from multiview images track the branch patterns from the root detected from a given image [19]. Also, graph and tree based structures have been used to analyze leaf veins, which is used for leaf species classification [20]. However, there are gaps in applying the same principle to e.g. classify trees in the two dimensional space, which is one of the aims of our paper.

Point Cloud Representation

PointNet++ [16] is a hierarchical neural network working recursively on a nested partitioning of the input point set while exploiting metric space distances, making it able to learn local patterns and features as the scale of the context increases. This model is adopted for the supervised classification task.

2.2 Imaging techniques

Pixel Based Approaches

Collecting input data for reconstructing realistic models of real trees through e.g. obtaining 3D point clouds using laser scanners or taking multiple photos from different angles around real trees can be too costly and time-consuming. Single-image-based tree reconstruction is a harder problem, but greatly simplifies the input requirements for recreating a realistic tree model.

Argudo et al. [4] created a pipeline which only tries to reconstruct the volumetric shape of the tree by extruding its outline, generating a passable but unfaithful reconstruction of the tree. Tan et al. [18]'s method requires the user to draw a stroke along the trunk and around the foliage, which are then fed into a growth engine to guide the generation of a tree.

The proposed method by Li et al. [10] involves training neural networks to predict the 3D structure of a tree from a single image. It segments the branches and leaves from the rest of the picture and generates disc-shaped bounding boxes and classifies the tree species, after which a procedural algorithm generates a more faithful 3D model of the tree than the aforementioned two papers. Another recent paper uses a Generative Adversarial Network (GAN) to infer the 3D skeleton of a tree from a single image and generate a realistic tree model [11].

On the topic of tree classification based on 2D images, several papers look at tree top classifications based on remote sensing data and drone imagery, in both RGB and hyperspectral images [7] [12] [21]. No papers were found solely on the topic image classification of trees based on 2D images, since that would be too elementary a task for a research paper.

GCN Based Approaches For Image Classification

[9] proposes some good practices for designing GCNs for image classification to compensate for the lack of domain knowledge that is hardcoded into CNNs, such as spatially oriented translation invariant filters.

3 Preliminaries

3.1 Graph Convolutional Networks

Graph Neural Networks (GNNs) are neural models that capture the dependence of graphs via message passing between the nodes of graphs. In recent years, variants of GNNs such as the Graph Convolutional Network (GCN), Graph Attention Network (GAT) and Graph Recurrent Network (GRN) have demonstrated ground-breaking performances on many deep learning tasks. [22]

According to Kipf [8], Most graph neural network models have a somewhat universal architecture in common which we will refer to as Graph Convolutional Networks.

For these models, the goal is then to learn a function of signals/features on a graph which takes as input: A feature description x_i for every node i summarized in a $N \times D$ feature matrix X, where N is the number of nodes and D the number of input features, in our case we would be having two features (x and y coordinates) for each node. A representative description of the graph structure is a matrix typically in the form of an adjacency matrix A and produces a node-level output Z (an $N \times F$ feature matrix, where F is the number of output features per node). These individual node-level outputs can be modeled by introducing some form of pooling operation arriving at a graph-level output.

Every neural network layer can then be written as a non-linear function $H^{l+1} = f(H^l, A)$, with $H^0 = X$ and $H^L = Z$ (or z for graph-level outputs), L being the number of layers. The specific models then differ only in how $f(\cdot, \cdot)$ is chosen and parameterized.

3.2 (Variational) Graph Autoencoders

Variational graph autoencoders (VGAE) proposed by Kipf [8] apply the idea of VAEs on graph-structured data. They have been successful on a number of citation network datasets such as Cora and Citeseer [17].

The encoder (inference model) of GAEs consists of Graph Convolutional Networks (GCNs). It takes an adjacency matrix A and a feature matrix X as inputs and generates the latent variable Z as output. The first GCN layer generates a lower-dimensional feature matrix. The decoder (generative model) is defined by an inner product between the latent variable Z and the input. After encoding a node to Z in the latent space, the similarity of each node embedding in that space is computed to generate the output adjacency matrix.

4 Method

4.1 Dataset

120 images of trees all along the same camera angle were rendered using the software Tree-it [3]. The dimensions of each image were 512 by 512 pixels. It should be noted the dataset was obtained independently. The 120 trees comprised of 6 different species, more specifically: 50 Maple trees, 30 Acacia trees, 10 Beech trees, 10 Oak trees, 10 Pine trees, and 10 Willow trees.

Skeletonization

Prior to skeletonization of each image, due to bugs in the software used, manual alpha blending had to be conducted on each image in order to be able to process them. After applying a threshold on the grayscale image equal to the mean value of the same image, sci-kit's [14] skeletonize function is used to obtain the skeleton of the tree. The skeleton object is then passed to a chain of dilation, erosion and closing.

Minimal Skeleton

The minimal skeleton in this paper refers to a graph structure composed of nodes representing important parts of the original skeleton, mainly split points and segment tails. Edges in the graph represent connecting segments between the two points in the original image. These minimal graph skeletons form the dataset used for the autoencoder experiment. In order to locate the most important points on the obtained skeleton object, the skeleton

is converted to a graph object whose independent connected paths are traversed in order to obtain different segments. The 25 longest segments are then identified and their start and end points are sampled, resulting in 50 points, most of which represent split points on the trunk or the end point of the branches. Creating a graph only with edges connecting these nodes often results in disconnected graphs. Hence, a minimum spanning tree connecting all the obtained points is formed to arrive at a final representative graph structure. Each node has one attribute called pos which is a tuple of normalized coordinates.

Figure 1: Data generation process. On the left is the input image of a synthetic maple tree. In the middle is the generated skeleton. On the right, the skeleton graph.

Dense Skeleton

The graph classification experiment requires graphs with more sampled points. Therefore this time only one third of the points in the skeleton object were sampled randomly. In this case, the input of the model are purely point clouds. The model to be used itself forms edges between the points. Both datasets, dense and minimal, loaded from node-link format are then converted into PyTorch custom datasets and batched and split for training and testing.

4.2 Experiments

GCN-Based Skeleton Classification

In this experiment, isolated nodes from the disjoint dense skeleton dataset (120 graphs) are fed into the PointNet++ neural network [16], originally proposed for 3D point cloud classification/segmentation, which tries to capture meaningful local geometric patterns to be able to classify the entire graph. We only had to manually reshape the *pos* coordinates of the nodes to have a 0.0 third dimension as well for it to work. In PointNet++, the nodes are iteratively processed by following a simple grouping, neighborhood aggregation and downsampling scheme:

The grouping phase in our case uses a k-NN algorithm to construct a graph in which nearby points are connected. The constructed graph is then processed in a Graph Neural Network layer by aggregating coordinate values for each node from their direct neighbors. This layer is what allows PointNet++ to capture local structures at different scales. Finally, in the downsampling phase, a pooling scheme which summarizes local structures containing aggregated skeleton coordinate positions is adopted.

This experiment is the result of applying the PyG implementation of this model [1] to our dense skeleton dataset. As a variation of this experiment, the minimal skeleton dataset is also fed into the model. 80% of the graphs in both cases were used for training while 20% of them for testing.

Concerning implementation details, as visible in the provided code by PyG [1], a (k=16)-Nearest Neighbor graph based on the position *pos* of nodes is generated on which two graph-based convolutional operators are applied and enhanced by ReLU non-linearities. The first operator takes in 3 input features (the positions of nodes) and maps them to 32 output features. The second operator is applied on the 32 dimension output from the previous step remaining in the same dimension. In the end a global graph readout function, *global-max-pool*, which takes the maximum value along the node dimension for each skeleton graph is applied which is then fed to a linear classifier mapping it to 6 different classes (1:Oak, 2:Maple, 3:Acacia, 4:Pine, 5:Beech, 6:Willow respectively). Three variations of this experiment, namely experimenting with different values for k, an additional convolutional middle layer and running the model on the minimal skeleton dataset as well as the dense one are also conducted.

GCN-Based Latent Representation With Graph Autoencoders

The vector encodings associated with the skeleton graphs in the previous experiment lack one important characteristic. There are no guarantees the vectors are representative enough to be able to reconstruct the original skeleton. Hence, in this experiment, a Graph Autoencoder is utilised in which the vectors produced by the encoder layer are made in such a manner that they are able to reconstruct the adjacency matrix in the original graph, which is done by minimizing the inner product between each node and its embedding during the process). For this experiment, the more challenging dataset containing the minimal skeleton graphs is used. This means the Graph AutoEncoder will learn to take a skeleton graph and compress the node positions and the edge connections into a low dimensional vector capable of being used to reconstruct the edges between different points on the skeletons, i.e constructing the links of the original graph. Doing so would mean given the original nodes in the skeleton, the skeleton tree can be accurately constructed, demonstrating geometrical insight hidden in the latent space.

5 Results

In this section, the results of the previously laid out two experiments are reported.

5.1 Skeleton Classification

As explained in the previous chapter, the PointNet++ [16] neural network was applied to our dataset. 82% accuracy in label prediction was observed with the first successful training and testing of the model on the dense skeleton dataset to assign a species label to each. Three separate sub-experiments were conducted to further examine and improve the prediction score, namely: applying the algorithm to the minimal skeleton dataset, restructuring the hidden layer of the model, and variations in the initial graph formation of the model. A streamline of the process looks as follows :

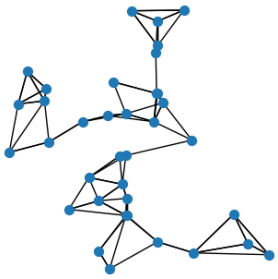

Figure 2: One input to the PointNet++

Dataset Experimentation

Following the initial success on the dense dataset, the model was also tested on the minimal skeleton dataset stripped of the edges in the graph. Quite surprisingly, as observed in Figure 3, the model classified the skeletons more accurately, jumping up to 87% accuracy in the dataset with fewer points (35 points on average for each minimal skeleton vs 761 points on average for each dense skeleton).

Hidden Convolutional Layer Experimentation

The used implementation of the model originally had placed two convolutional 32x32 layers for aggregating the node coordinates. As expected with other standard deep neural nets, the addition of an extra 32x32 layer to the process, saw the model jump 4% in the predictive power up to 95% as observed in figure X.

Input Formation Experimentation

A critical part in the implementation of PointNet++ concerns the formation of a custom k-NN graph since the model expects a point cloud. Originally the value was set to k=16. Given the two dimensional nature of this task and the special case of points representing tree skeletons, the model was

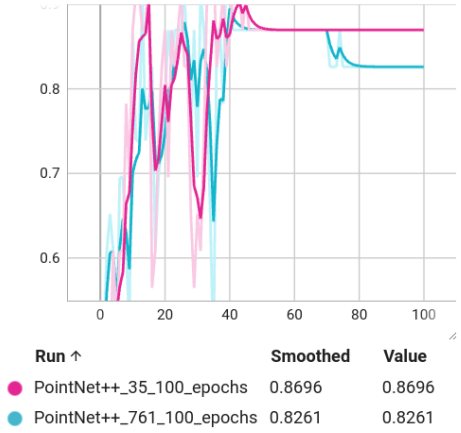

Figure 3: PointNet++ performance on dense (blue) vs the minimal dataset (pink)

Figure 4: PointNet++ accuracy with three convolution layers (red) instead of two (green)

adjusted and tested for three more cases of k=1, k=4, and k=8 both in the case of three convolution layers, and with two, resulting in eight different variations.

In both architectures, k=4 was found to produce the best results: 95% testing accuracy. In the case of 3 convolution layers the original k=16 performed as well as k=4. In both cases k=1 performs the worst. The performance for these values of k are understandable because, in the case of a minimal skeleton which this experiment was based on, message passing and association between a reasonable number of neighbours is the optimal mode of analysis as geometrical information is not isolated in each node as in the case of k=1 and not too spread out in the whole structure in the case of k=16 or k=8 with two convolution layers.

In summary, the customized PointNet++ model described above, was able to predict the original tree

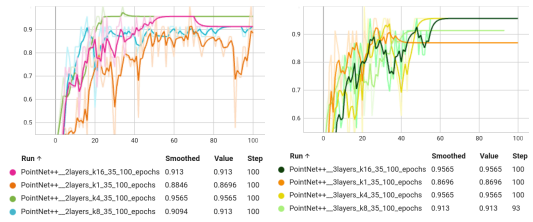

Figure 5: 20k steps of training and its performance on test sets

label with a 95% accuracy through learning to map shared and convolved geometrical patterns between the critical points in the minimal skeleton of the trees to metadata.

5.2 Skeleton Representation

The Graph Autoencoder model previously explained was applied to the minimum skeleton dataset consisting of MSTs of the tree skeletons. The relevant details and the results are reported as follows.

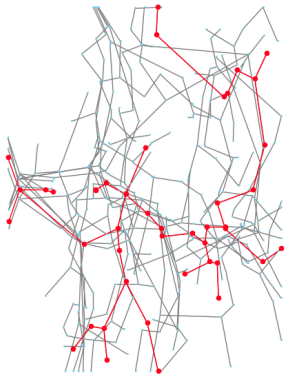

Figure 6: A subsection of the composed graph: the skeleton MSTs fall on top of each other in the normalized space but are not connected. Highlighted in red is one tree skeleton.

The Graph Auto Encoder works on one single graph so we had to compose all the different MSTs to one single object. It should be noted that no further edges were added to the composed graph. Therefore, the composed graph is essentially a forest of seperated trees. Upon the initial failure to compute node embedings out of this dataset with this model, all the coordinate positions in all skeleton were normalized. As a visual example, one tree skeleton is highlighted in the composed normalized space.

Three different sub experiments were conducted in this regard. First, the global graph was fed into the GAE model which learned to reconstruct the global adjacency matrix with different degrees of accuracy based on the dimension of the embedded space. Second, the global graph was fed into a Variational GAE model hoping for more accurate reconstruction. And lastly, due the implications of the results of the previous two sub experiments, only a small subset of the global graph containing only MST skeletons of the same species of trees were fed to the model.

Global Graph Adjacency Matrix Prediction using a Graph Auto Encoder

In this experiment, the GAE composed of the Graph Neural architecture explained in the previous section, was applied to the global normalized disconnected graph four times, each with a different dimension of the embedded space as observable in the figure below:

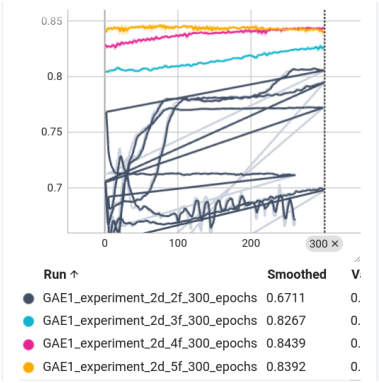

Figure 7: Our GAE Average Precision associated with two, three, four and five dimensional latent space

The GAE with the two-dimensional output space naturally performed the worst with an AP of 67% followed by three-dimensional space with an AP of 82%. However, there are no improvements observed increasing the dimension from 4d to 5d.

Global Graph Adjacency Matrix Prediction using a Variational Graph Auto Encoder

In this experiment, a standard Variational Graph Autoencoder provided by the PyG official implementation [2] was applied to the global graph to look for any improvement over the traditional GAE model. Quite surprisingly, as observable from the figure below, the VGAE performed even worse than its less complex predecessor, in all three dimensions compared.

There are a few possible reasons for this. One is the structure of the graph dataset fed into the algorithm. VGAEs were designed with the aim of learning connections between nodes with high-dimensional and sometimes irregular features such as geometrical molecular analysis. The crude convolutional layers used in this model are not optimized for aggregating spatial coordinates as pointed out by [22].

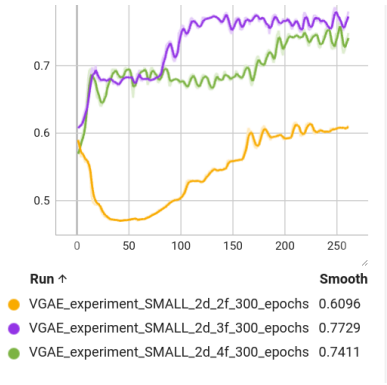

Figure 8: A subsection of the composed graph: the skeleton MSTs fall on top of each other in the normalized space but are not connected. Highlighted in red is one tree skeleton.

This was not the case in the previous experiment in which a Euclidian-space-aware Graph Neural Network was used. Moreover, not only the global graph contained disjoint tree skeletons, they were skeletons of different species of trees. This is why as a last experiment, only the MST skeletons of 10 pine trees were fed into the algorithm as laid out below.

Single Type Adjacency Matrix Prediction using a Graph Auto Encoder

Restricting the scope of the dataset to only one species of trees did result in an improvement over the original mixed dataset. As it can be observed from the figure below, a 10% jump compared to figure Y is observed in the case of a two-dimensional output space.

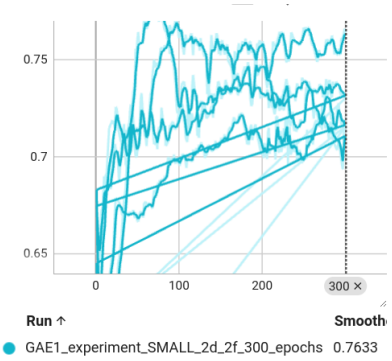

Figure 9: GAE on only one class of skeletons

6 Discussion

The experiments conducted not only show the possibility of encoding tree geometry with Graph Neural Networks, but also show the possibility of using skeleton graphs of trees as a proxy for the tree itself. Doing so is justifiable and advantageous in terms of both space and time efficiency for some tasks such as tree classification.

Concerning the tree classification experiment, an important question might be posed: It could be that such a supervised classification model could learn to cheat its way to the end by learning special easy mappings from input to labels with disregard to the geometry of the skeleton as a whole. One possible answer to this argument is that the model learns to arrive at a latent space by using only pure geometrical information (node positions) and no other node or edge features whatsoever. Even if the model learns shortcuts from the input graph, it would only do so purely based on the positions of the nodes, and not based on any other features such as colour or node degrees, since the skeletons are fed as raw point cloud data with no edges. This would actually demonstrate the model has learned key geometrical patterns specific to that species of trees. It is argued that if indeed cheat patterns are part of what the model does, they can be treated as a representative pattern of the original skeleton graph. With that said, the second GAE experiment was conducted both as a remedy to the this point and to further demonstrate the capability of GCNs to capture a geometrical profile at different scales.

Limitations and Future Work

Concerning the first experiment in classifying skeleton points, our model lacks flexibility as it takes node positions as input features, and uses a simple difference in relational Cartesian coordinates for creating messages between nodes. This means it would not generalize across rotations applied to the input points. Future work can further strengthen this experiment by fixing this issue by using the PPFNet algorithm [6].

As pointed out by Danel et. Al [5], traditional GCNs lack the ability to take into account spatial information even when they are coded into node features. Hence, a strong case for the second experiment not performing as well as the supervised classification which used a spatially aware convolution mechanism is this. Therefore, future work can take this into account by designing a pair of encoder and decoder for the GAE custom designed for spatial graphs such as the one proposed in the same paper, PointNet++ [16], or PPFNet [6].

Another area for venture exploring graph based tree geometry would be with random walk based GCNs such as a spatially aware DeepWalk algorithm and the like [15].

Another area for venture concerns using node embeddings not only to predict edges between them but to also predict new (missing) node positions demonstrating an even deeper representation of tree geometrical profile embedded in the latent space.

This area (new node generation with GCNs) was however found to be underdeveloped which makes this an even more challenging task.

Conclusion

Going back to the original RQ of this paper 1, the results of our unsupervised and supervised graph embedding experiments demonstrate the soundness of modelling trees with only a minimal skeleton spatial graph object while preserving and encoding both tree geometry and metadata using Graph Convolutional Networks. This point was reached through two independent experiments. In the first one, the PointNet++ model was used to find an 95% accurate mapping between the sparse skeleton graph objects composed of only coordinates to metadata (species label). In the second experiment, the same point was demonstrated again by utilising Graph Auto Encoders capable of encoding a substantial part of a tree's geometrical profile (i.e the segments connecting its critical points). In summary, the answer to both sub questions posed was found to be positive which declares modelling tree geometry in 2d using graphs and Graph Neural Networks is both reasonable and effective.

Ethical Considerations

There are no ethical considerations pertaining to this paper as the dataset was generated independently and no other stakeholders or parties were involved.

A Appendix

References

[1] Google colaboratory. https://colab.research.google.com/drive/1D45E5bUK3gQ40YpZo65ozs7hg5l-eo_U?usp=sharing. Accessed: 2024-1-28.

[2] Google colaboratory. https://colab.research.google.com/github/AntonioLonga/PytorchGeometricTutorial/blob/main/Tutorial6/Tutorial6.ipynb. Accessed: 2024-1-28.

[3] TreeIt — Tree Generator — evolved-software.com. evolved-software.com/treeit/treeit. [Accessed 04-01-2024].

[4] Oscar Argudo, Antonio Chica, and Carlos Andujar. Single-picture reconstruction and rendering of trees for plausible vegetation synthesis. *Computers & Graphics*, 57:55–67, 2016.

[5] Tomasz Danel, Przemysław Spurek, Jacek Tabor, Marek Śmieja, Łukasz Struski, Agnieszka Słowik, and Łukasz Maziarka. Spatial graph convolutional networks, 2020.

[6] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching, 2018.

[7] Fabian Ewald Fassnacht, Hooman Latifi, Krzysztof Stereńczak, Aneta Modzelewska, Michael Lefsky, Lars T Waser, Christoph Straub, and Aniruddha Ghosh. Review of studies on tree species classification from remotely sensed data. *Remote sensing of environment*, 186:64–87, 2016.

[8] Thomas Kipf. Graph convolutional neural networks, 2016.

[9] Boris Knyazev, Xiao Lin, Mohamed R. Amer, and Graham W. Taylor. Image classification with hierarchical multigraph networks, 2019.

[10] Bosheng Li, Jacek Kałużny, Jonathan Klein, Dominik L Michels, Wojtek Pałubicki, Bedrich Benes, and Sören Pirk. Learning to reconstruct botanical trees from single images. *ACM Transactions on Graphics (TOG)*, 40(6):1–15, 2021.

[11] Zhihao Liu, Kai Wu, Jianwei Guo, Yunhai Wang, Oliver Deussen, and Zhanglin Cheng. Single image tree reconstruction via adversarial network. *Graphical Models*, 117:101115, 2021.

[12] Somayeh Nezami, Ehsan Khoramshahi, Olli Nevalainen, Ilkka Pölönen, and Eija Honkavaara. Tree species classification of drone hyperspectral and rgb imagery with deep learning convolutional neural networks. *Remote Sensing*, 12(7):1070, 2020.

[13] Fumio Okura. 3d modeling and reconstruction of plants and trees: A cross-cutting review across computer graphics, vision, and plant phenotyping. *Breeding Science*, 72(1):31–47, 2022.

[14] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14. ACM, August 2014.

[16] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017.

[17] Hongyu Shan, Di Jin, Pengfei Jiao, Ziyang Liu, Bingyi Li, and Yuxiao Huang. Nf-vga: Incorporating normalizing flows into

graph variational autoencoder for embedding attribute networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1244–1249. IEEE, 2020.

[18] Ping Tan, Tian Fang, Jianxiong Xiao, Peng Zhao, and Long Quan. Single image tree modeling. *ACM Transactions on Graphics (TOG)*, 27(5):1–7, 2008.

[19] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. In *ACM SIGGRAPH 2007 papers*, pages 87–es. 2007.

[20] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[21] Chen Zhang, Kai Xia, Hailin Feng, Yinhui Yang, and Xiaochen Du. Tree species classification using deep learning and rgb optical images obtained by an unmanned aerial vehicle. *Journal of Forestry Research*, 32(5):1879–1888, 2021.

[22] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.