Fine-Tuning DeepForest for Forest Tree Detection in High-Resolution UAV Imagery

Josafat-Mattias Burmeister[1], Julian Zabbarov[1,3], Stefan Reder[2], Rico Richter[1], Jan-Peter Mund[2], Jürgen Döllner[1,3]

[1] University of Potsdam, Digital Engineering Faculty, Potsdam, Germany
(burmeister, zabbarov, rico.richter.1, doellner)@uni-potsdam.de
[2] Eberswalde University for Sustainable Development, Faculty of Forest and Environment, Eberswalde, Germany
(stefan.reder, jan-peter.mund)@hnee.de
[3] Hasso Plattner Institute, Potsdam, Germany

Keywords: UAV imagery, individual tree detection, deep learning, data labeling, forest inventory

Abstract

Forest inventories are essential for sustainable forest management and health monitoring. Image-based surveys using unmanned aerial vehicles (UAVs) are increasingly adopted for this purpose. DeepForest, a deep learning model pre-trained on large annotated datasets, enables scalable and cost-efficient tree detection in the resulting imagery. Although it is known that fine-tuning DeepForest to specific target sites improves its performance, optimal fine-tuning strategies remain unclear. In this study, we investigate the fine-tuning of DeepForest on a small dataset of UAV imagery from a temperate mixed forest. We examine the influence of four parameters on model performance: (1) the method used to label training data (manual labeling, manual correction of DeepForest predictions, and automatic labeling using 3D point cloud segmentation), (2) input image resolution, (3) the number of fine-tuning epochs, and (4) the size of the training dataset. Our results show that the out-of-the-box DeepForest model performs poorly on the target dataset but substantially improves with fine-tuning, even when using limited data and a small number of training epochs. Manual and automatic labeling achieve similarly high performance, both outperforming the manual correction approach. Although the model initially performs better on lower-resolution images, this difference diminishes after fine-tuning on higher-resolution inputs. We observe no clear benefit from increasing the training set size, possibly due to the low diversity of our test dataset. Overall, our findings confirm that DeepForest is a suitable base model for forest tree detection in high-resolution UAV imagery and can be adapted to new environments with low effort.

1. Introduction

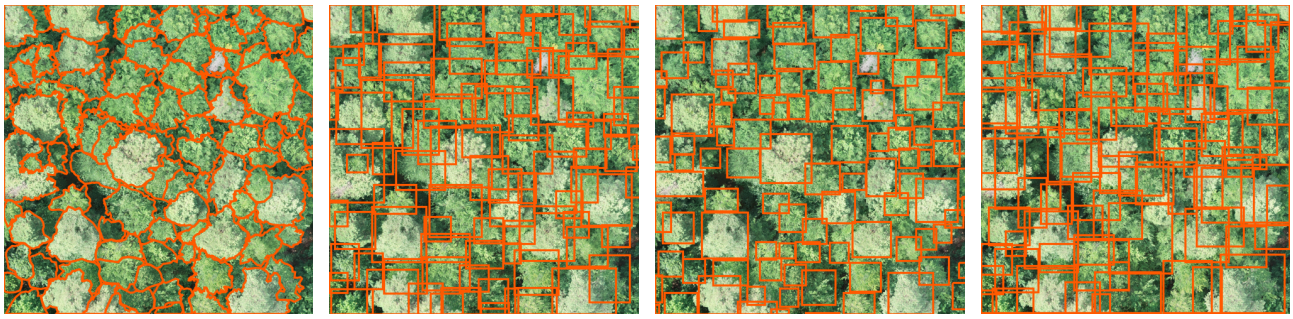
Forest inventories are essential for assessing forest health and sustainability (Lausch et al., 2017), especially since environmental stressors, such as droughts, are becoming increasingly frequent (Anderegg et al., 2020). Monitoring the state of specific tree species and counting the number of living trees in forest areas is important to implement targeted conservation measures (Bater et al., 2009). Traditionally, forest monitoring has relied on manual ground-based surveys, which are time-consuming, labor-intensive, and costly due to the need for on-site personnel (Lausch et al., 2017). This manual approach lacks scalability and limits the ability to collect data over extensive areas.

To address these challenges, many works explore the use of unmanned aerial vehicle (UAV) imagery and associated data processing techniques to automate forest monitoring. Tree detection is a crucial first step in building comprehensive and scalable forest monitoring systems. To automate tree detection, many approaches rely on Light Detection and Ranging (LiDAR) data (Reitberger et al., 2007; Jeronimo et al., 2018; Eysn et al., 2015) or photogrammetric point clouds (Nevalainen et al., 2017; Kattenborn et al., 2014). For example, Eysn et al. (2015) evaluated eight LiDAR-based tree detection approaches and found that local maxima detection within a canopy height model produced the best results, particularly in single-layered forests. Similarly, Nevalainen et al. (2017) applied a local maxima approach to photogrammetric point clouds to detect trees in boreal forests, while Kattenborn et al. (2014) used similar data to detect palm trees on a plantation in Kiribati with an overall accuracy of 86.1%.

Although 3D point clouds can provide accurate tree detection results, in particular the acquisition of LiDAR point clouds requires expensive and specialized hardware. Photogrammetric point clouds can be acquired at lower cost from RGB imagery, but their processing still requires substantial computational resources. Recent advances in deep learning, particularly in object detection tasks, have created opportunities to reduce the costs of data acquisition and processing by directly leveraging RGB images, which can be captured by consumer-grade UAVs. For example, Ball et al. (2023) presented a neural network, called Detectree2, which uses a masked R-CNN model to delineate irregular edges of tree crowns from RGB images. Santos et al. (2019) benchmark the R-CNN, YOLOv3 and RetinaNet architectures to obtain rectangular bounding boxes for particular tree species. They find that RetinaNet outperforms the other architectures tested when evaluating the models on a RGB image dataset from an urban area in Brazil.

To reduce the technical barriers to applying deep learning methods to tree detection, Weinstein et al. (2020a) developed the Python package DeepForest. In addition to providing various functions for setting up tree detection pipelines, DeepForest includes a RetinaNet model (Lin et al., 2017) that was trained on 10 000 labeled trees from 22 different sites throughout the US. It achieves an average recall of 72 % and a precision of 64 % (Weinstein et al., 2020a) on the NEON crowns benchmark dataset (Weinstein et al., 2021). Due to the additional preprocessing functionalities provided by the DeepForest package, the model can be easily applied to new data (Weinstein, 2025).

While the pre-trained DeepForest model can be used out-of-the-box, Weinstein (2025) expects the model performance to



(a) Manually labeled crown polygons

(b) Bounding boxes computed from manually labeled crown polygons

(c) Manually corrected bounding boxes predicted by DeepForest

(d) Bounding boxes automatically computed using a tree instance segmentation of a 3D point cloud

Figure 1. Examples of annotations obtained with the different labeling approaches used in this work (Site 1, Plot 1.0 is shown).

improve when fine-tuning the model to the target dataset. The author found the best-performing results with five to ten fine-tuning epochs. Building on these findings, this work further investigates different configurations of the fine-tuning process of the DeepForest model. Specifically, we fine-tune the DeepForest model for a small target dataset of a temperate mixed forest located in Brandenburg, Germany. We use DeepForest's functionalities to implement a configurable fine-tuning pipeline that allows one to fine-tune the DeepForest model on custom datasets. Using this pipeline, we study the impact of four fine-tuning parameters on model performance: (1) the method used to label training data (manual labeling, manual correction of DeepForest predictions, and automatic labeling using 3D point cloud segmentation), (2) input image resolution, (3) the number of fine-tuning epochs, and (4) the size of the training dataset.

2. Methodology

In an effort to improve the performance of the DeepForest model for tree detection in high-resolution RGB UAV imagery of temperate mixed forests, we fine-tune the model on our target dataset using a small amount of labeled data. We compare different labeling approaches and experiment with different input image resolutions and numbers of fine-tuning epochs to identify a suitable fine-tuning approach.

2.1 Dataset

The target dataset for this study is a high-resolution UAV RGB imagery dataset acquired in the Sauen forest, located in Brandenburg, Germany. In the following, we will refer to this dataset as the Sauen dataset. The Sauen forest consists of several sites with different species composition and stand structure. The Sauen dataset includes images from three different sites:

Site 1. Large parts of this site are covered by a mixed stand of Scots pine (*Pinus sylvestris*), European beech (*Fagus sylvatica*), and sweet chestnut (*Castanea sativa*), with 165-year-old Scots pine as remnants in the overstory and a dense main canopy layer consisting of beech and chestnut. In the central part of the site, the closed stand is interrupted by a planted regeneration area with a loose Scots pine overstory and a dense understory of sweet chestnut trees. At the south-eastern edge of the site, a coniferous stand of green Douglas fir (*Pseudotsuga menziesii*) and coastal fir (*Abies excelsior*) is located, which is also partly included in the investigation plots.

Site 2. This site is covered by a three-layered mixed stand. The canopy layer is formed by a loose overstory of Scots pine and coastal fir, and an intermediate layer dominated by sessile oak (*Quercus petraea*) and cypress (*Chamaecyparis spec.*). The crown closure is light to open. Under it, a rich understory has established that contains northern red oak (*Quercus rubra*), Douglas fir, sessile oak, Norway maple (*Acer platanoides*), and Scots pine.

Site 3. This site is covered by a mixed stand of European beech, sessile oak, and Scots pine, which form a dense canopy layer. The stand has a loose understory of European beech which origins from natural rejuvenation.

UAV RGB imagery of all sites was collected in summer 2023 (July-August) using a DJI Phantom 4 RTK. A double-grid flight pattern with a flight altitude of $70\,\mathrm{m}$ to $80\,\mathrm{m}$ was used. The images were taken with $85\,\%$ front and side overlap and a camera inclination of $-75°$. From the UAV images, high-resolution orthophotos (pixel size of $1.6\,\mathrm{cm}$ to $1.8\,\mathrm{cm}$) and 3D point clouds were reconstructed photogrammetrically using the software Agisoft Metashape.[1] In addition to the UAV-based survey, terrestrial LiDAR scans of some areas were collected using a GeoSLAM ZEB Horizon scanner, which is a handheld personal laser scanner (PLS). The UAV-borne data and the PLS point clouds were manually aligned using markers that were placed in the field, and the manual alignment was further refined by co-registering the point clouds using the iterative closest point (ICP) algorithm (Arun et al., 1987).

2.2 Data Labeling

To create reference bounding boxes for fine-tuning the DeepForest model, three labeling approaches are tested in this work:

Manual labeling. In this approach, the boundaries of the tree crowns were manually annotated as polygons (Fig. 1a) in the UAV orthophotos using the QGIS software.[2] Since the DeepForest model works with bounding box labels, the axis-aligned bounding boxes of the polygon labels were computed and used as labels for model training and evaluation. Since, in the labeling of crown polygons, subdominant trees whose crowns are only partially visible in aerial images were also included, the resulting bounding boxes show a high degree of overlap (Fig. 1b). For some of the manually annotated image areas, reference data from field surveys (tree location, trunk diameter

[1] Agisoft Metashape software: `https://www.agisoft.com/`
[2] QGIS software: `https://www.qgis.org`

Site	Plot[(a)]	Size (m^2)	Labeling	Trees	Usage
Site 1	Plot 1.0	50 × 50	ML	98	train (small)
Site 1	Plot 1.0	50 × 50	MC	108	train (small)
Site 1	Plot 1.0	50 × 50	AL	90	train (small)
Site 1	Plot 1.1	100 × 100	ML	289	train (ext.)
Site 1	Plot 1.2	120 × 80	MC	362	train (ext.)
Site 1	Plot 2	120 × 120	MC	377	train (ext.)
Site 2	Plot 1	120 × 120	MC	435	train (ext.)
Site 1	Plot 1.3	200 × 100	AL	624	train (ext.)
Site 1	Plot 2	150 × 100	AL	311	train (ext.)
Site 1	Plot 3	100 × 100	AL	312	train (ext.)
Site 3	Plot 1	50 × 50	ML	55	test

[(a)] Plot 1.1, plot 1.2, and plot 1.3 partially overlap. Plot 1.0 is contained within these plots.

Table 1. Overview of the data used in this study (ML = manual labeling, MC = manual correction, AL = automatic labeling).

at breast height, and tree species) were available and used to verify the labels.

Manual correction of DeepForest predictions. In order to reduce the effort involved in fully manual labeling, this approach used the out-of-the-box DeepForest model (without fine-tuning) to predict the initial tree bounding boxes. The predicted bounding boxes were then manually inspected and corrected by a human annotator using the RectLabel software.[3] As shown in Fig. 1c, the labels from the manual correction of the DeepForest predictions contain fewer and less overlapping bounding boxes than the labels from the fully manual labeling. This can be attributed to the fact that the out-of-the-box DeepForest model mainly detects dominant trees, and the human annotator only corrected clearly visible errors. The fully manual labeling and manual correction of the DeepForest labels were performed by different people. In the following, we refer to this approach as 'manual correction'.

Automatic labeling using 3D point clouds. This approach employs SegmentAnyTree (Wielgosz et al., 2024), a deep learning method for tree instance segmentation in 3D point clouds, for fully automatic labeling of the aerial imagery. Specifically, the PLS point clouds and the photogrammetric UAV point clouds of the study sites were fused and processed using the SegmentAnyTree model. The model segments 3D point clouds into tree and non-tree points and outputs tree instance IDs for each tree point (Fig. 2a). To project these per-point labels onto the aerial imagery, the points were binned into a horizontal 2D grid with a grid size of $0.2\,\mathrm{m}$ based on their XY-coordinates. Each grid cell was assigned the tree instance ID of the highest tree point within the grid cell. In this context, the height of a point was defined as the distance from a digital terrain model, which was reconstructed from the 3D point cloud using the cloth simulation filtering algorithm (Zhang et al., 2016). Grid cells that did not contain any tree point were marked as background. The result of this projection is a 1-channel label image that contains the label of each pixel, i.e., the ID of the tree instance or the background label (Fig. 2b). To remove noise from the labels, the label image was post-processed by assigning grid cells to the background if the height of the corresponding point was less than $10\,\mathrm{m}$ above the ground. This threshold value was chosen because all overstory trees in the study plots are larger than $10\,\mathrm{m}$. Furthermore, a modal filter with a quadratic 3×3 kernel was applied to the label image to reduce noise resulting from small gaps in the canopy. Finally, the pixel-wise labels were converted into bounding-box labels by computing the axis-aligned bounding box of the pixel mask of each tree instance. Bounding boxes with a width or height of less than $2\,\mathrm{m}$ were discarded, as these are typically false positive detections of the SegmentAnyTree model. As shown in Fig. 1d, the bounding box labels obtained by this automatic approach appear to be visually similar to the labels obtained by manual labeling, although less accurate. In the following, we refer to this approach as 'automatic labeling'.

[3] RectLabel software: `https://rectlabel.com/`

2.3 Data Partitioning

For our experiments, we divided the Sauen dataset into training and test sets. For each labeling approach, image regions of different sizes and positions were annotated, since the labeling effort of the approaches differs, and for some regions, no 3D point clouds were available for automatic labeling. Therefore, we experiment with different variants of the training set:

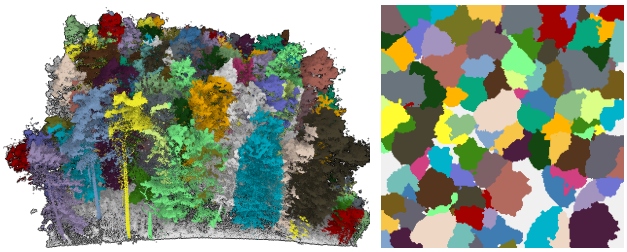
Small training set. To allow a fair comparison between the labeling strategies, we use a small training set that contains the same image region for all labeling approaches. This small training set consists of a $50 \times 50\,\mathrm{m}$ plot from site 1 (plot 1.0 in Table 1), for which labels from all three labeling approaches are available (Fig. 1).

Extended training sets. To study the benefit of additional training data, we perform additional fine-tuning runs with extended training datasets that contain all available data for the respective labeling strategy from sites 1 and 2. The size of these extended training sets differs for the different labeling strategies, ranging from $1\,\mathrm{ha}$ for manual labeling to $4.5\,\mathrm{ha}$ for automatic labeling (Table 1). There is some overlap between the extended training sets (the data from the small training set is included), but each extended training set also contains some data that are not included in the extended training sets for the other labeling approaches. For example, the extended training sets for manual and automatic labeling contain only data from site 1, while the extended training set for manual correction also contains a $120 \times 120\,\mathrm{m}$ plot from site 2. Therefore, the extended training set variants should not be used to compare the labeling methods against each other, but can be used to study the benefit of additional data compared to the respective small training set for the same labeling approach.

Test set. The same test set is used for all experiments, which consists of a $50 \times 50\,\mathrm{m}$ plot from site 3 (plot 1). To reduce data leakage, no other data from site 3 is used in any of the training set variants (the minimum distance between the plots in the training and the test set is $350\,\mathrm{m}$). Since we consider manual labeling to be the most accurate labeling approach, the test set was manually annotated, and the resulting labels were used as a reference to evaluate the other labeling approaches.

2.4 Model Architecture and Training Settings

The base model for our fine-tuning experiments is the tree detection model included in the DeepForest package (Weinstein et al., 2020a), which uses the RetinaNet architecture (Lin et al., 2017). The RetinaNet consists of two main components: a backbone for feature extraction and a model head for object detection. The backbone of the RetinaNet model in the DeepForest package is based on the ResNet50 architecture (He et al.,



(a) 3D point cloud segmented using SegmentAnyTree

(b) Label image obtained by projecting the point-wise labels to a 2D grid

Figure 2. Examples of the intermediate data involved in the automatic derivation of 2D bounding box labels from point-wise tree instance segmentation labels of a 3D point cloud (different colors represent different tree instance IDs and points or pixels labeled as background are shown in gray).

2016). One of the key characteristics of the RetinaNet architecture is that it constructs a multi-scale feature pyramid from the input image, allowing it to detect objects at different image resolutions. The model head of the RetinaNet architecture comprises a classification network that predicts the probability of object presence in a bounding box and a bounding-box regression network that refines the position and size of the detected bounding boxes. The RetinaNet model of the DeepForest package was trained using a two-stage semi-supervised training process (Weinstein et al., 2019, 2020a): In the first stage, the model was pre-trained using a large-scale dataset with noisy labels that were obtained by segmenting aerial point clouds using an unsupervised algorithm (Silva et al., 2016). In the second stage, the model was trained using a supervised learning approach, using a smaller set of hand-annotated RGB images for some of the sites it was fitted on in the previous stage. In total, the DeepForest model was trained on a dataset of approximately 30 million algorithmically annotated trees and 10 000 hand-annotated trees (Weinstein et al., 2020a).

In our experiments, we fine-tune the DeepForest model on the Sauen dataset using stochastic gradient descent with a learning rate of 0.0001. Following Weinstein et al. (2020a), we use focal loss (Lin et al., 2017) as the loss function for the RetinaNet classification head and l_1-loss (mean absolute error) for the regression head. When processing larger images, the DeepForest package splits these images into fixed-size tiles. Since our small training set variant and our test set each consist of a single 50×50 m plot, we choose a tile width of 50 m, although the DeepForest model was originally trained with a tile width of 40 m (Weinstein et al., 2020b). We calculate the patch size parameter of the DeepForest package (tile width in pixels) according to the input resolution. Considering the dimensions of the training plots (Table 1), the overlap between the tiles is set to 20 % to evenly distribute overlapping regions between the tiles. Before evaluation, we post-process the model predictions by applying non-maximum suppression with an Intersection over Union (IoU) threshold of 0.5. We increase the IoU threshold compared to the default value of 0.4 in the DeepForest package (Weinstein, 2025) because our dataset mainly covers dense forest stands with a closed canopy, and we expect a higher degree of overlap between the bounding boxes (Fig. 1b).

2.5 Evaluation

We evaluate the model performance using the standard metrics of precision, recall, and F_1-score. We consider a predicted bounding box to be a true positive if the IoU between the predicted and any reference bounding box is at least 0.4. For all experiments, five fine-tuning runs with different seeds (0 - 4) are performed, and the average and standard deviation of the metrics across these runs are computed.

2.6 Experimental Design

In our evaluation, we study the impact of four fine-tuning parameters on the performance of the DeepForest model on the Sauen dataset: (1) the labeling approach used to generate the training data, (2) the input image resolution, (3) the number of fine-tuning epochs, and (4) the size of the training set used for fine-tuning. We consider the three labeling approaches described in Section 2.2. We test four different input image resolutions, namely pixel sizes of 2.5 cm, 5 cm, 7.5 cm, and 10 cm, which are obtained by downsampling the original orthophotos using bilinear interpolation. To identify an appropriate number of fine-tuning epochs, we run each experiment for 20 epochs and evaluate the model after each epoch.

Initially, we compare the different labeling approaches using the small variant of the training set. Subsequently, we perform additional fine-tuning runs using the extended training sets for each labeling approach. We use a full factorial experimental design, testing all combinations of the aforementioned parameter values with five different random seeds each (3 labeling approaches $\times$ 4 input resolutions $\times$ 2 training set variants $\times$ 5 random seeds $=$ 120 fine-tuning runs in total with 20 epochs per run).

2.7 Implementation Details

All experiments were carried out on an ASUS workstation with an AMD Ryzen Threadripper PRO 5955WX 16-core 4 GHz CPU, 512 GB RAM, and a Nvidia GeForce RTX 4090 graphics card. To include the latest updates to the DeepForest package, a pre-release version of the package was retrieved from the main branch of the Github repository (as of April 10, 2025).[4] Python version 3.12, PyTorch version 2.4.1,[5] and torchvision version 0.19.1[6] were used.

3. Results and Discussion

The results of the fine-tuning runs for the small training set variants are shown in Fig. 3. Except for fine-tuning with manually corrected data, all tested fine-tuning scenarios with small training sets lead to substantial improvements over the out-of-the-box DeepForest model without fine-tuning (baseline). In Fig. 4, the results of the fine-tuning runs on the small and extended training set variants are compared. For manually and automatically labeled data, the inclusion of additional training data leads to no or only minor improvements compared to the small training sets. In contrast, the results of the fine-tuning with manually corrected data improve when additional training data are used. As a result, for the extended training sets, all fine-tuning scenarios produce an improvement over the baseline model.

[4] DeepForest GitHub repository: `https://github.com/weecology/DeepForest`
[5] PyTorch package: `https://pytorch.org/docs/2.4/`
[6] torchvision package: `https://pytorch.org/vision/0.19/`

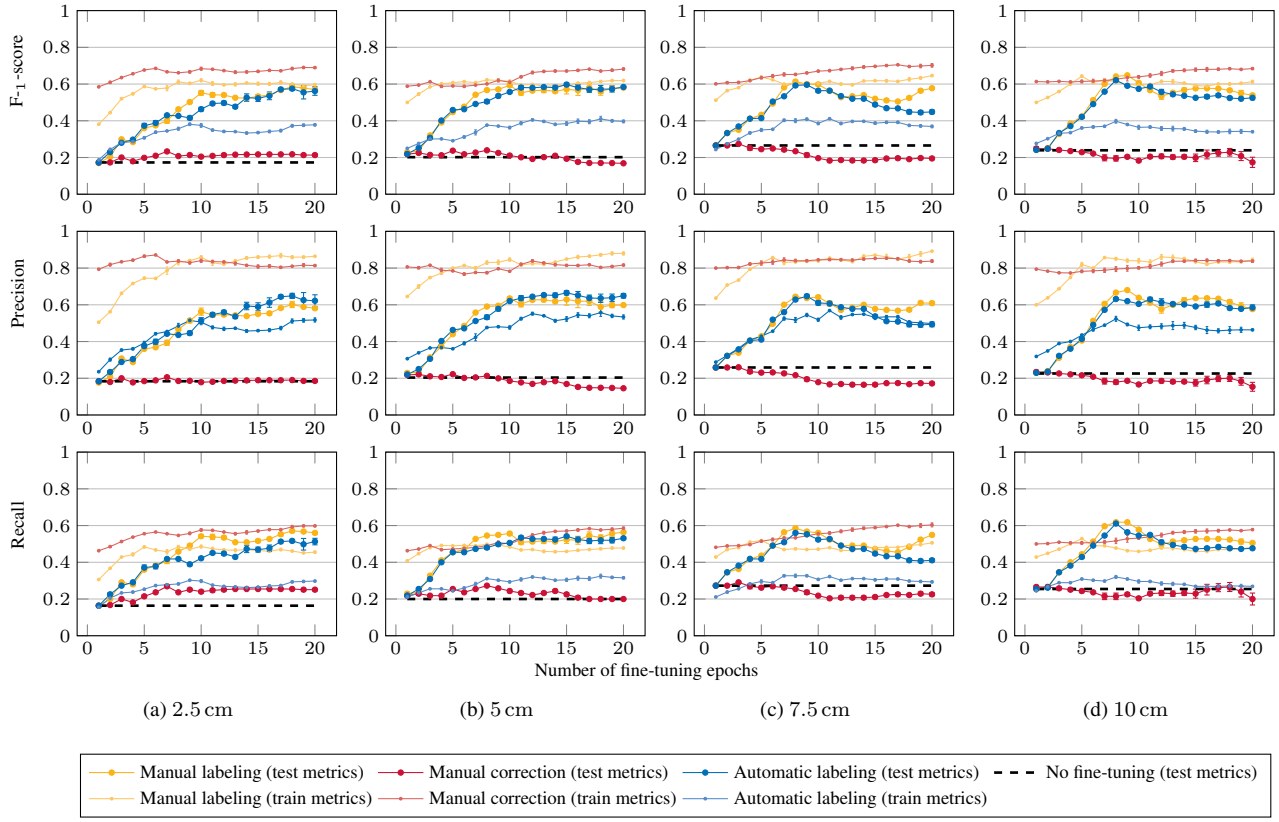


Figure 3. Performance of the fine-tuned DeepForest models after fine-tuning on the small variants of the training set. Results are shown for different approaches to label the training data (individual lines), different input resolutions (columns), and different numbers of fine-tuning epochs (x-axis).

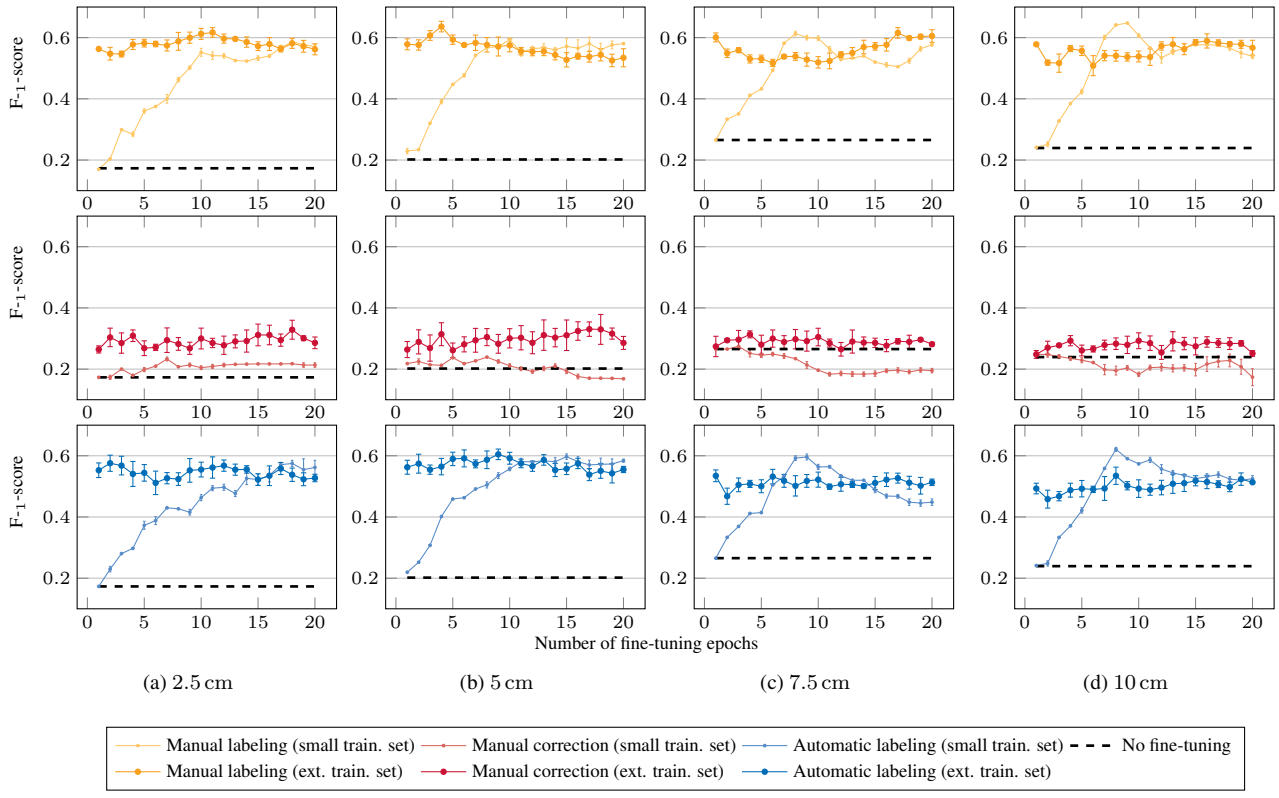
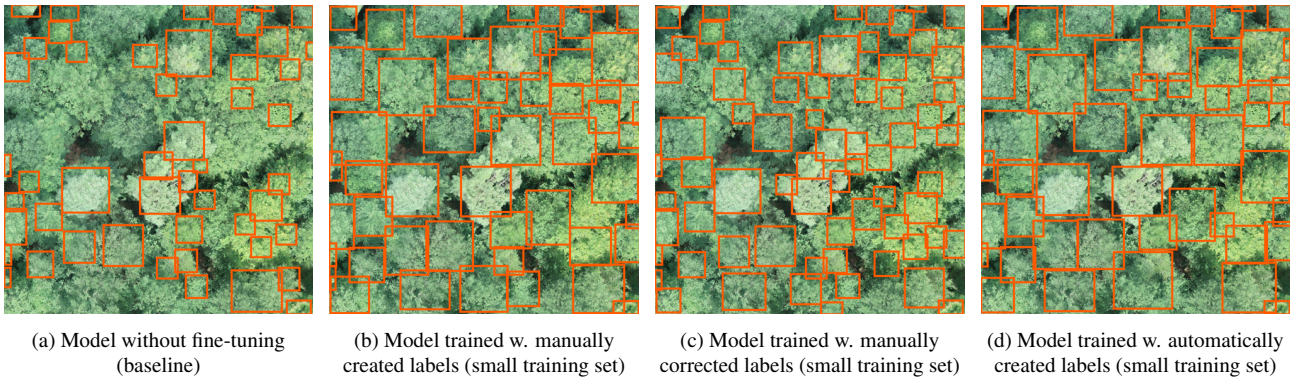
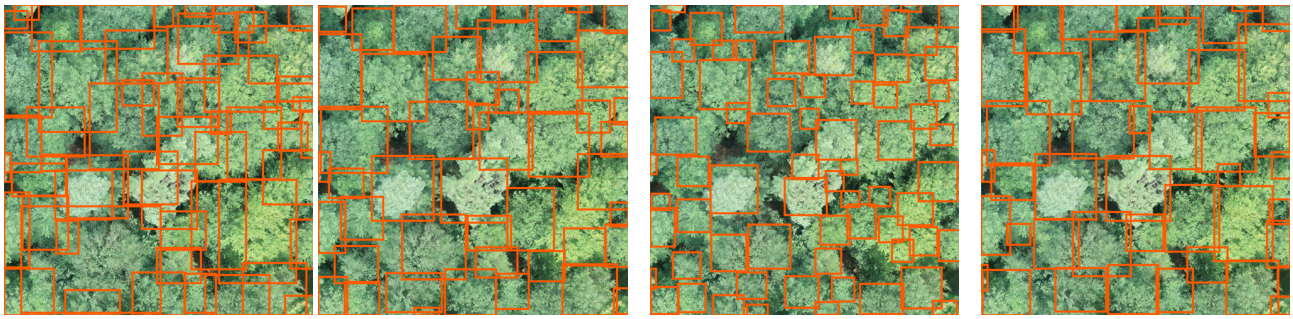


Figure 4. Comparison of the performance of the fine-tuned DeepForest models after fine-tuning on the small vs. the extended variants of the training set. Results are shown for different approaches to label the training data (individual lines), different input resolutions (columns), and different numbers of fine-tuning epochs (x-axis). The metrics shown are those for the test set of the Sauen dataset.

This contribution has been peer-reviewed.
https://doi.org/10.5194/isprs-archives-XLVIII-4-W15-2025-39-2025



(a) Model without fine-tuning (baseline)
(b) Model trained w. manually created labels (small training set)
(c) Model trained w. manually corrected labels (small training set)
(d) Model trained w. automatically created labels (small training set)



(e) Reference labels (manual labeling)
(f) Model trained w. manually created labels (ext. training set)
(g) Model trained w. manually corrected labels (ext. training set)
(h) Model trained w. automatically created labels (ext. training set)

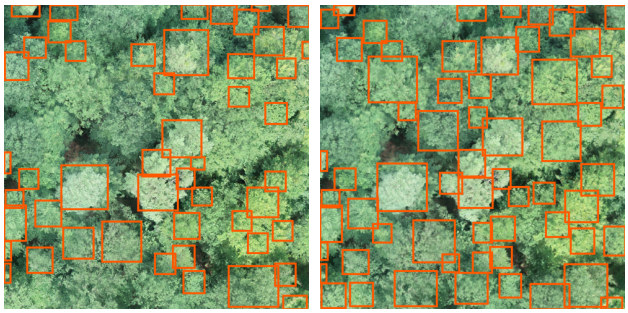
Figure 5. Predictions of the DeepForest models fine-tuned with different training set variants for the test set of the Sauen dataset. For each training set variant, we show the predictions of the model corresponding to the epoch and random seed that achieved the highest test F_1-score, when training on the highest input resolution (2.5 cm pixel size).

Labeling approaches. Comparing the different labeling approaches on the small training set variant shows that manual labeling and automatic labeling produce very similar results, clearly outperforming manual correction in terms of the test metrics (max. test F_1-score of 0.65 for manual labeling, 0.62 for automatic labeling, 0.27 for manual correction). In most cases, fine-tuning on the manually corrected data even leads to a performance degradation compared to the baseline. This can be attributed to overfitting of the models, as fine-tuning with manually corrected data results in the highest training metrics (max. train F_1-score of 0.7 for manual correction, 0.65 for manual labeling, 0.41 for automatic labeling).

Visual inspection of the predictions shows that the models fine-tuned with manually corrected data tend to predict too small bounding boxes (Fig. 5c and Fig. 5g). This is most likely because the manual correction of the DeepForest predictions only corrected clearly visible errors, so that the resulting labels consist of smaller and less overlapping bounding boxes compared to fully manual labeling (Fig. 1). The fine-tuning runs with manually and automatically labeled data lead to very similar test results, with manual labeling achieving only slightly higher test metrics for image resolutions of 7.5 cm and 10 cm. Visually, the results also appear similar (Fig. 5). This is surprising, since the automatically generated labels contain visible segmentation errors (Fig. 2a). However, the similar performance of both approaches is probably due to the following factors: First, the fine-tuning on the automatically generated labels is less affected by overfitting, with the training metrics being even lower than the test metrics. Second, the fine-tuning seems to mainly fine-tune the general distribution (expected size and overlap) of the bounding boxes, and these are similar in the manually and automatically generated labels (Fig. 1).

Input resolution. We observe that the out-of-the-box DeepForest model performs better at lower input resolutions. Specifically, the model tends to predict fewer and smaller bounding boxes at lower resolutions (Fig. 6). This is consistent with the fact that the DeepForest model was originally trained on lower-resolution images with 10 cm pixel size (Weinstein et al., 2020a). In our experiments, the best out-of-the-box performance is achieved for a pixel size of 7.5 cm (max. test F_1-score of 0.17 for 2.5 cm, 0.2 for 5 cm, 0.27 for 7.5 cm, 0.24 for 10 cm). After fine-tuning the model on the target resolution using the small training sets, the trend that DeepForest performs better at lower input resolutions is still recognizable (max. test F_1-score of 0.59 for 2.5 cm, 0.60 for 5 cm, 0.61 for 7.5 cm, 0.65 for 10 cm), but disappears when fine-tuning on the extended training datasets (max. test F_1-score of 0.62 for 2.5 cm, 0.64 for 5 cm, 0.62 for 7.5 cm, 0.59 for 10 cm). This indicates that the RetinaNet model, despite using a multi-scale feature pyramid, learns resolution-dependent features for the detection of individual trees. Thus, its out-of-the-box generalizability to image resolutions not included in the training set is limited. Given these results, it is advisable to rescale the input images to a resolution similar to the original DeepForest training data if no labeled data are available for fine-tuning. As we do not observe any clear advantage of a higher resolution, it seems reasonable to also use downsampled images for fine-tuning to increase computational efficiency. Since similar performance is achieved for all resolutions after fine-tuning, it remains unclear from our results whether the model benefits from additional contextual information that it can access at lower input resolution or from a higher level of detail that is provided at higher input resolution.

Number of fine-tuning epochs. When fine-tuning the DeepForest model on the small training sets, the largest increase in



(a) 2.5 cm (b) 10 cm

Figure 6. Predictions of the out-of-the-box DeepForest model for different resolutions of the test set of the Sauen dataset.

performance occurs in the first ten epochs of fine-tuning (Fig. 3). In later epochs, test metrics increase more slowly or even deteriorate when fine-tuning with 7.5 cm and 10 cm resolution. For the extended training set variants, an even smaller number of fine-tuning epochs is sufficient to achieve a significant improvement over the baseline (Fig. 4). This is most likely because increasing the size of the training set results in more training steps per epoch. Specifically, a significant increase in test metrics can be observed after a single training epoch when using manually or automatically generated labels, and only small improvements occur in subsequent epochs. When fine-tuning on the small training set, we observe a tendency for more epochs to be beneficial when fine-tuning on low input resolutions than on higher resolutions. This may be related to the fact that the DeepForest model was originally trained on lower resolution images, and a higher number of training steps is required to adapt the model to higher input resolutions. However, when fine-tuning on the extended datasets, this tendency is not visible.

Training set size. When comparing the results for the small and the extended training set variants, it can be observed that increasing the size of the training datasets accelerates the learning process (F_1-score of 0.24 after one training epoch on the small training set with manual labeling and 10 cm resolution, 0.58 after one training epoch on the corresponding extended training set) but does not significantly improve overall performance (max. F_1-score of 0.65 on the small training set with manual labeling and 10 cm resolution, 0.59 on the corresponding extended training set). This is surprising as the small training set only covers a 50×50 m plot and contains a very limited number of training samples (Table 1). However, the plots included in the small training set and the test set are very similar in terms of the structure of the forest stand (both are stands with a dense, closed canopy). Since some of the extended training sets cover more diverse stand structures (e.g., regeneration sites with loose canopy), their value may be reflected in the test metrics if these stand types were also represented in the test set. Nevertheless, our results demonstrate that the DeepForest model can be adapted to fairly homogeneous target datasets with a very limited amount of annotated data.

Precision vs. recall. Examining precision and recall for the fine-tuning runs on the small training set reveals that the models achieve considerably higher precision than recall on the training set. On the test set, precision and recall are more balanced, although precision remains slightly higher in most cases. This indicates that the model is more prone to omission errors than commission errors. One possible explanation is that the dataset includes labeled trees whose crowns are only partially visible in the orthophotos. The lower recall may also be due to the post-processing of the model's predictions with non-maximum suppression, which may still be too conservative, as our dataset includes highly overlapping bounding boxes.

Overall, our results demonstrate that the DeepForest model can be adapted to new environments using a small amount of training data and with low computational effort. The performance of our fine-tuned models is comparable to the results reported by Weinstein et al. (2020b) for the Eastern Deciduous site (average precision of 0.54 after cross-site training), which is characterized by a closed canopy similar to our dataset. However, the error rates of our fine-tuned models remain relatively high, indicating that further improvements are necessary to obtain reliable monitoring data. Errors are especially frequent in dense, homogeneous parts of the canopy, where identifying individual trees is inherently difficult, even for humans. Future research should focus on further enhancing accuracy in these challenging scenarios. Possible directions include exploring techniques that directly predict tree crown polygons rather than bounding boxes, as well as utilizing larger datasets for automatic training data generation or unsupervised pretraining.

4. Known Limitations

The generalizability of our findings may be limited by the small size and structural homogeneity of our test dataset (50×50 m, 55 trees). The test area closely resembles the forest structure of the smaller version of our training set. As a consequence, our results may underestimate the value of incorporating more diverse training data that represent a broader range of forest structures. Another limitation concerns inconsistencies in the annotation procedures: the fully manual labeling and the manual correction of DeepForest predictions were performed by different individuals and in different contexts. In the manual labeling process, the crown polygons were annotated with the aim of fully covering the crowns of all partially visible trees, using field inventory data as a reference when available. In contrast, the manual correction of DeepForest predictions focused only on clearly visible errors, often omitting partially visible trees and accepting tighter bounding boxes that excluded the outermost portions of the crowns. Therefore, our manual correction approach reflects a low-budget scenario where only limited time can be allocated to label correction. Unaffected by these differences, our comparison of manual and fully automated labeling provides valuable insights into the potential of data fusion approaches for automated training data generation.

5. Conclusions

In this study, we fine-tuned the DeepForest model for forest tree detection in high-resolution UAV imagery of temperate mixed forests using a target dataset from Brandenburg, Germany. Our results demonstrate that the model's out-of-the-box performance is limited in this context, particularly when applied to higher-resolution imagery than the model was originally trained on. However, we show that its performance can be substantially improved by fine-tuning it with a limited amount of labeled data and only a few training epochs. By comparing different labeling strategies, we found that fully automated labeling based on 3D point cloud segmentation achieves performance comparable to that of manual labeling. This suggests that, in scenarios where dense 3D point clouds are available along with UAV imagery, a fully automatic labeling pipeline can potentially reduce

or eliminate the need for manual labeling. However, the test dataset used in our study is relatively small and structurally similar to the training data, which may limit the generalizability of our findings. Future research should therefore validate our results using larger and more diverse test datasets that better reflect varying forest structures. Overall, our results confirm that the DeepForest model can be successfully adapted to new environments through fine-tuning, but also highlight the need to further improve accuracy in dense, homogeneous canopies.

6. Data and Code Availability

The Python source code of this study is available on GitHub: `https://github.com/ai4trees/deepforest-finetuning`. The data is available on request.

7. Acknowledgements

We thank the anonymous reviewers for their valuable feedback. We also thank the August Bier foundation for giving us the opportunity to collect data in the Sauen forest. This work was partially funded by the Federal Ministry of Education and Research, Germany through grant 033L305 ('TreeDigitalTwins') and grant 01IS22062 (AI research group 'FFS-AI').

References

Anderegg, W. R., Trugman, A. T., Badgley, G., Konings, A. G., Shaw, J., 2020. Divergent Forest Sensitivity to Repeated Extreme Droughts. *Nature Climate Change*, 10(12), 1091–1095.

Arun, K. S., Huang, T. S., Blostein, S. D., 1987. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5), 698-700.

Ball, J. G. C., Hickman, S. H. M., Jackson, T. D., Koay, X. J., Hirst, J., Jay, W., Archer, M., Aubry-Kientz, M., Vincent, G., Coomes, D. A., 2023. Accurate Delineation of Individual Tree Crowns in Tropical Forests From Aerial RGB Imagery Using Mask R-CNN. *Remote Sensing Ecol. Conserv.*, 9(5), 641–655.

Bater, C. W., Coops, N. C., Gergel, S. E., LeMay, V., Collins, D., 2009. Estimation of Standing Dead Tree Class Distributions in Northwest Coastal Forests Using LiDAR Remote Sensing. *Canadian Journal of Forest Research*, 39(6), 1080–1091.

Eysn, L., Hollaus, M., Lindberg, E., Berger, F., Monnet, J.-M., Dalponte, M., Kobal, M., Pellegrini, M., Lingua, E., Mongus, D., Pfeifer, N., 2015. A Benchmark of Lidar-Based Single Tree Detection Methods Using Heterogeneous Forest Data From the Alpine Space. *Forests*, 6(5), 1721–1747.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 770–778.

Jeronimo, S. M. A., Kane, V. R., Churchill, D. J., McGaughey, R. J., Franklin, J. F., 2018. Applying LiDAR Individual Tree Detection to Management of Structurally Diverse Forest Landscapes. *Journal of Forestry*, 116(4), 336-346.

Kattenborn, T., Sperlich, M., Bataua, K., Koch, B., 2014. Automatic Single Tree Detection in Plantations Using UAV-Based Photogrammetric Point Clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3, 139–144.

Lausch, A., Erasmi, S., King, D. J., Magdon, P., Heurich, M., 2017. Understanding Forest Health With Remote Sensing-Part II—A Review of Approaches and Data Models. *Remote Sensing*, 9(2), 129.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection. *IEEE International Conference on Computer Vision*, IEEE, 2999–3007.

Nevalainen, O., Honkavaara, E., Tuominen, S., Viljanen, N., Hakala, T., Yu, X., Hyyppä, J., Saari, H., Pölönen, I., Imai, N., Tommaselli, A., 2017. Individual Tree Detection and Classification With UAV-Based Photogrammetric Point Clouds and Hyperspectral Imaging. *Remote Sensing*, 9(3), 185.

Reitberger, J., Heurich, M., Krzystek, P., Stilla, U., 2007. Single Tree Detection in Forest Areas With High-Density LiDAR Data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3), 139–144.

Santos, A. A. D., Marcato Junior, J., Araújo, M. S., Di Martini, D. R., Tetila, E. C., Siqueira, H. L., Aoki, C., Eltner, A., Matsubara, E. T., Pistori, H., Feitosa, R. Q., Liesenberg, V., Gonçalves, W. N., 2019. Assessment of CNN-Based Methods for Individual Tree Detection on Images Captured by RGB Cameras Attached to UAVs. *Sensors*, 19(16), 3595.

Silva, C. A., Hudak, A. T., Vierling, L. A., Loudermilk, E. L., O'Brien, J. J., Hiers, J. K., Jack, S. B., Gonzalez-Benecke, C., Lee, H., Falkowski, M. J., Khosravipour, A., 2016. Imputation of Individual Longleaf Pine (Pinus palustris Mill.) Tree Attributes From Field and LiDAR Data. *Canadian Journal of Remote Sensing*, 42(5), 554–573.

Weinstein, B., 2025. DeepForest documentation. `https://deepforest.readthedocs.io/en/v1.5.0/`. Accessed: 24 April 2025.

Weinstein, B. G., Marconi, S., Aubry-Kientz, M., Vincent, G., Senyondo, H., White, E. P., 2020a. DeepForest: A Python Package for RGB Deep Learning Tree Crown Delineation. *Methods in Ecology and Evolution*, 11(12), 1743–1751.

Weinstein, B. G., Marconi, S., Bohlman, S. A., Zare, A., Singh, A., Graves, S. J., White, E. P., 2021. A Remote Sensing Derived Data Set of 100 Million Individual Tree Crowns for the National Ecological Observatory Network. *eLife*, 10, e62922.

Weinstein, B. G., Marconi, S., Bohlman, S. A., Zare, A., White, E. P., 2020b. Cross-Site Learning in Deep Learning RGB Tree Crown Detection. *Ecological Informatics*, 56, 101061.

Weinstein, B. G., Marconi, S., Bohlman, S., Zare, A., White, E., 2019. Individual Tree-Crown Detection in RGB Imagery Using Semi-Supervised Deep Learning Neural Networks. *Remote Sensing*, 11(11), 1309.

Wielgosz, M., Puliti, S., Xiang, B., Schindler, K., Astrup, R., 2024. SegmentAnyTree: A sensor and platform agnostic deep learning model for tree segmentation using laser scanning data. *Remote Sensing of Environment*, 313, 114367.

Zhang, W., Qi, J., Wan, P., Wang, H., Xie, D., Wang, X., Yan, G., 2016. An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation. *Remote Sensing*, 8(6), 501.