7-18-2023

LIDAR Voxel Segmentation Using 3D Convolutional Neural Networks

Yuval H. Levental
yhl3051@rit.edu

LIDAR Voxel Segmentation Using 3D Convolutional Neural Networks

by

Yuval H. Levental

B.S. Michigan State University, 2013

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in the Chester F. Carlson Center for Imaging Science
College of Science
Rochester Institute of Technology

July 18, 2023

Signature of the Author _____

Accepted by _____
Coordinator, M.S. Degree Program Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE

COLLEGE OF SCIENCE

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Yuval H. Levental
has been examined and approved by the
thesis committee as satisfactory for the
thesis required for the
M.S. degree in Imaging Science

Dr. Jan van Aardt
Thesis Advisor

Dr. John Kerekes
Committee Member

Dr. Carl Salvaggio
Committee Member

Date

Lidar Voxel Segmentation Using 3D Convolutional Neural Networks

by

Yuval H. Levental

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Master of Science Degree
at the Rochester Institute of Technology

Abstract

Light detection and ranging (lidar) forest models are important for studying forest composition in great detail, and for tracking objects in the understory. In this study we used DIRSIG, a first-principles and physics-based simulation tool, to turn the lidar data into voxels, towards classifying forest voxel types. A voxel is a 3D cube where the dimension represents a certain distance. These voxels are split into categories consisting of background, leaf, bark, ground, and object elements. Voxel content is then predicted from the provided simulated and real National Ecological Observation Network (NEON) data. The inputs are 3D neighborhood cubes which surround each voxel, which contain surrounding lidar signal and content type information. Provided simulated data are from two sources: a VLP-16 drone, which collects discrete lidar data close to the canopy, and the NEON Airborne Observation Platform (AOP), which is attached to an airplane flying 1000 m above ground level and collects both discrete and waveform lidar data. Different machine learning algorithms were implemented, with 3D CNN algorithms shown to be the most effective. The Keras library was used, since creating the layers with the sequential model was regarded as an elegant approach. The simulated VLP-16 waveform data were significantly more accurate than the simulated NEON waveform data, which was attributed to its proximity to the canopy. Leaves and branches exhibited acceptable accuracies, due to their relatively random shapes. However, ground and objects in both cases had very

high accuracy due to the high intensities and their rigid shapes, respectively. A sample of real NEON waveform lidar data was used, though the sample primarily focused on the canopy region; however, most of the voxels were correctly predicted as leaves. Additional channels were added to the input voxels in order to improve accuracy. One input parameter which proved to be very useful were the local z-values of each input array. Additionally, the Keras Tuner framework was used to obtain improved hyperparameters. The learning rate was reduced by a factor of 10, which provided slower, but steadier convergence towards accurate predictions. The resulting accuracies from the predictions are promising, but there is room for improvement. Different ML algorithms that use the point cloud should also be considered. Further segmentation of forest classes is another possibility. For example, there are different types of trees and bushes, so each tree or bush could have its own unique classes, which would make predicting the shapes much easier. Overall, discovering a method for accurate object prediction has been the most significant finding. For the ground truth models, the best object precision is approximately 99% and the best recall is 78%.

Acknowledgments

I would like to thank my family for encouraging me throughout this degree program, especially because much of my research was done throughout the COVID-19 pandemic. I would have not been able to enter this program or obtain a MS in Imaging Science without their support. I am also grateful to Lt. Col. Rob Wible and Mr. M. Grady Saunders for providing me with support as I was learning about lidar data, voxels, and running DIRSIG simulations. Additional thanks go to visiting professor Dr. Ruchi Gajjar for teaching me better ways to separate training and testing data, and for introducing me to the Keras Tuner. Thank you to all the professors in the Center that taught the courses enabling me to perform quality scientific research. Of these professors, I would like to specifically thank Dr. Chris Kanan for teaching me important machine learning principles in Deep Learning for Vision. I would also like to thank Dr. Rich Hailstone for teaching Noise and Probability with a very positive attitude and great sense of humor. Additionally, I would like to thank Dr. John Kerekes for working with me on a machine learning independent study project, which prepared me for this thesis. RIT Research Computing (RC) was very helpful in instructing me how to run machine learning algorithms on their servers. I am glad that I got to meet them in-person sometimes. Last but definitely not least, thank you, Dr. Jan van Aardt, for providing me with guidance and wisdom in regards to conducting scientific research, running machine learning simulations, and properly classifying voxels.

Contents

List of Figures

List of Tables

Introduction

CONTEXT

Forests support many ecosystem services for modern civilizations, and contribute to the health of our society and the global ecosystem. They form the habitats of countless plant and animal species, and are a critical part of climate regulation via processes such as carbon capture [1, 2]. However, tracking forest dynamics and traits in an efficient, orderly manner requires specialized monitoring and measurement methods. One technique for assessing forest structural traits is light detection and ranging (lidar). In fact, modern lidar systems have sufficiently advanced to the extent where they can form the basis of remotely sensed forest structural measurements [3, 4, 5].

Structural remote sensing of forests was initially performed via electro-optical sensing or synthetic aperture radar (SAR). SAR systems emit low-frequency pulses capable of measuring tree height or penetrating forest canopies to detect the ground and static targets [6]. However, the precision and recall of results from older or heterogeneous forests was deemed to be inadequate. For example, optical sensors could not properly measure the above ground biomass (AGB) values of re-grown forests that are older than 10-15 years [7]. Additionally, models generated from SAR backscatter returns typically are insensitive to AGB values that were equal to or greater than 60 Mg/ha [8].

Lidar, by contrast, can directly measure structural information of forests, due to high laser pulse and scanning rates, both of which enable dense 3D point cloud measurements [9]. Lidar has been utilized in this domain for at least the past 30 years. Typical structural measurements that can be analyzed via lidar systems include tree height, biomass, leaf area index (LAI), forest element shapes, and voxel fills, to name a few [3, 4, 5].

Lidar operates by emitting a pulse of light energy at a specific wavelength, and then through measuring the number of photons which return after scattering as a function of time [10]. So-called “discrete” lidar systems identify only the locations of the return peaks (x, y, z, or 3D position), which

generally correspond to solid physical objects. Full-waveform lidar (FW lidar) systems, on the other hand, digitize the entire backscattered waveform as a function of time [11], which can be used to characterize different kinds of forests, including older or heterogeneous forests. However, a comprehensive understanding of the lidar waveforms' propagation through and interaction with forest elements remain a daunting task, given the structural complexity of most forests and the challenges related to radiative transfer modeling in such environments. It is in this context that simulation could play a critical role in our furthering the knowledge of radiative transfer in forest settings.

The Rochester Institute of Technology (RIT) has created a scene simulation tool, called Digital Imaging and Remote Sensing Image Generation (DIRSIG), to perform such radiative transfer modeling for virtual 3D scenes. DIRSIG is first-principles and physics-based, meaning that the method is based directly on established scientific knowledge, from the ground-level to canopy levels. Monte-Carlo ray tracing is used to determine what fraction of photons are absorbed or scattered by forest elements. The scattering phase function determines scatter direction, followed by photon propagation [12, 13, 14]. We will use DIRSIG in this study to reconstruct input array properties from FW lidar data and simulate waveform lidar propagation through the vertical forest structures.

The main objective of this research is to predict forest voxel fills and types from scattered waveform fraction arrays. Reconstructing forest element shapes, which consist of scattered waveform fraction voxels, is important for predicting voxel fills. Ideally, there would be little background noise in the input array to enable accurate predictions, while the virtual scene needs to be a close approximation of reality (complex forest structures). The best method of voxel prediction, as current state-of-the-art, is via 3D convolutional neural networks (3D CNN). This is because there are many classifiers and layers, which are helpful in predicting voxel fills based on surrounding 3D inputs [15, 16, 17]. However, we will first provide a background to lidar sensing, simulation, and CNN approaches.

One specific objective is assessing different machine learning models. Support vector classification (SVC), single-stage CNN, and dual-stage CNN all will be compared to identify which models produce the best results. Additionally, the quality of different types of input data will be evaluated, including simulated VLP-16 drone waveform lidar data, simulated NEON waveform lidar data, and real-world waveform lidar data. Simulated VLP-16 data are far more likely to be accurate overall, given the data density (altitude and flight parameters). Finally, the best model hyperparameters will be determined by testing the model on computer servers with different capabilities, and by using a model tuner library. More powerful servers should provide the opportunity to achieve far more accurate results.

OBJECTIVES

- Objective 1: Design 3D CNN algorithms to predict voxel ground truth forest categories from simulated lidar data.
- Objective 2: Evaluate the accuracy of simulated VLP-16 waveform lidar data versus simulated NEON waveform lidar data. VLP-16 data are collected close to the canopy, and NEON data from an altitude of 1000 m, whereas the specific lidar types are also different.
- Objective 3: Evaluate whether prediction accuracies significantly can be improved by creating better input data channels, and through tuning the hyperparameters.

THESIS LAYOUT

Background

This chapter provides a more in-depth background of the history of remote sensing in the forestry domain. An overview of how CNNs work and how they are structured also is given. Basic terms and methods are identified in more detail than what would be found in subsequent chapters.

Methods

This chapter discusses the provided data and voxel algorithms in significant detail. All the data were collected from the Harvard Forest, and the category areas were measured for each voxel. The SVC and 3D CNN algorithms which were used to predict categories are examined. Sequential

modeling is used to design the CNN algorithms. Finally, there is an overview of the Keras Tuner library, which was used to pick the best hyperparameters.

Results

This chapter provides and discusses the results from all the simulations. The VLP-16 models were relatively accurate, which was attributed to the drone data being collected from right above the canopy. NEON models, however, were far less accurate due to the 1000 m height for data collection. Additionally, the NEON system is based on waveform lidar data which contains far more noise than discrete return data. Different methods for data collection were compared with each other. Overall, ground and object voxels performed extremely, likely due to their predefined, rigid shapes. Additionally, there is an example of real data, which mostly consists of the top canopy, correctly predicted to mostly consist of leaf voxels.

Summary

This chapter summarizes the work done at the time of writing, presents the current conclusions, and describes future work and improvements to be considered for continuation of the research topics.

SCIENTIFIC CONTRIBUTIONS

- We created a 3D CNN algorithm where every input contains the surrounding neighborhood of each voxel, and developed a method to reconstruct ground truth training and predicted test data.
- We demonstrated that simulated VLP-16 data are more accurate than simulated NEON data, likely due to the close range of collection, as well as other flight parameter that influence data density. For both datasets, ground and objects had high accuracies due to high intensities and predictable shapes.
- We discovered that larger input windows yielded more accurate results, which was attributed to the greater amount of available data.
- Finally, we used hyperparameter tuning to significantly lower the learning rate, resulting in greater overall accuracy through a longer time period.

Background

LIDAR BASICS

Lidar technology is based on active sensing, where a system emits the energy that is used for measurement. This is opposed to passive sensing, such as electro-optical telescopes, which measure energy from an independent source that is emitted by or backscattered from a designated target. Lidar emits laser beam pulses, and thus detects a series of backscattered returns separated by a time delta. Because the photons in each laser pulse travel at the speed of light ($3x10^8$ m/s), the distance between the sensor and the target can be calculated based on the time elapsed between emission and detection. This distance method can be utilized to generate a 3D point cloud when the location and line-of-sight from the sensor are also known [9].

The general and approximate equation for lidar range measurements from an airborne laser [10] is outlined in $Equation\ 1$:

$$R = c\frac{\Delta t}{2} \qquad (Equation\ 1)$$

where *R* is the direct measured distance from the sensor to the designated target, *c* is the speed of light in the Earth's atmosphere, and Δt is the time-period from initial emission to the return pulse arrival. The laser twice traverses the total distance in the time period, so the time period is initially halved, before being multiplied by the speed of light to determine the distance.

If the sensor's rectangular coordinates and angular direction (roll, pitch, and yaw) are known in 3D space, coordinate information can be determined for each measurement. The coordinate information can be used to generate a 3D point cloud in each coordinate reference system, where a GPS sensor typically determines the latitude, longitude, and altitude of the lidar sensor. Additionally, an inertial measurement unit (IMU) sensor is used to measure the lidar sensor's roll, pitch, and yaw [18].

The varying coordinates and look angles (see Figure 1) are indicated in the FUSION lidar processing software suite manual [19], as an example. Generating lidar point clouds using this method

requires precise and accurate timing to correctly return the data points; otherwise, the structural measurements will be inaccurate, due to several system detection inconsistencies or errors. The GPS sensor is the most reliable component, because satellite data are constantly being received from an onboard atomic clock. Therefore, this sensor is used to synchronize the time-stamps of the lidar sensors and the IMU sensor [20].

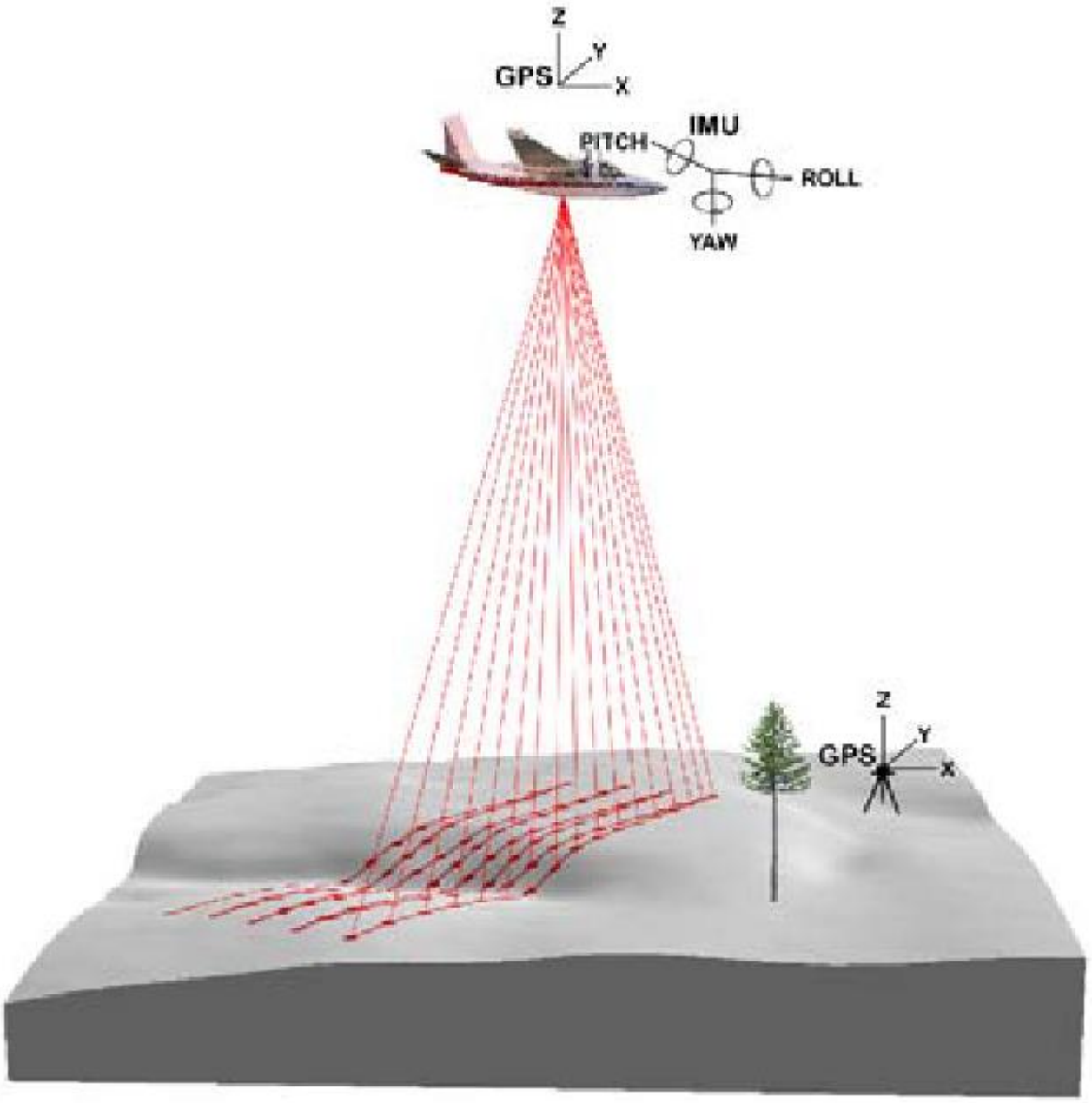

Figure 1: Diagram of different geometric properties involved in point cloud generation originating from airborne laser scanners [16].

Lidar systems have been utilized heavily by governmental, scientific, and industrial research for over 30-40 years. These systems can capture high-resolution structural information over a wide variety of distance scales. Organizations such as the National Aeronautics and Space Administration (NASA), the U.S. Military, and the National Ecological Observatory Network (NEON) all use lidar to capture

data for landscapes and forests. NEON, for example, uses lidar data to assess land use change, vegetation structure and properties, and climate change, among other phenomena [21].

In recent years, research has shown that flight parameters, such as flight pattern and overlap, have a significant impact on the quality of captured data. For instance, “flying low and slow” has been shown to be the optimal setup to obtain high-resolution point clouds [22]. Additionally, good sensor overlaps and multiple views have been shown to improve point density within a point cloud. However, there is not much knowledge currently as to how these parameters affect canopy penetration rates, point density, and occlusion zones [23, 24]. Here we intend to simulate an operational waveform lidar system to develop a CNN-based approach to voxel classification. Flight parameter optimization thus falls outside the purview of our study, but should be explored in future efforts. However, a brief discussion of how we visualize and analyze waveform lidar data is worthwhile, especially in the context of voxels.

LIDAR AND VOXEL VISUALIZATIONS

The methodology of using 3D binning to separate lidar data into high spatial resolution voxels is well-established [25, 26, 27]. There are many potential applications, some of which include studying forest volume, element classification, and forest/type area distribution. More specifically, voxel-based analysis of forests has also been utilized for tree heights, densities, and crown projection area [28]. For this study, the main objective is to use full waveform (FW) lidar to analyze forest materials. Three recent studies, using both discrete return and full-waveform lidar data, published from 2014-2017, are especially relevant.

Hagstrom et al. created voxel-based modeling methods that used discrete return lidar data for several purposes. These methods, one shown in Figure 2, include voxel transmission, data quality assessment, line-of-sight mapping, and lidar-image fusion. Discrete return values in voxel transmission calculations produced far more accurate results compared to basic hit counting from a ground truth

simulation scene. The authors showed that a 3-return discrete system accounts for 15% more backscattered waveform energy compared to the basic hit count. The voxels were classified into four categories: leaf, bark, ground, and object. Additionally, they demonstrated that a full-waveform system somewhat outperformed a three-return discrete system, which they attributed to the full-waveform data providing more information [29].

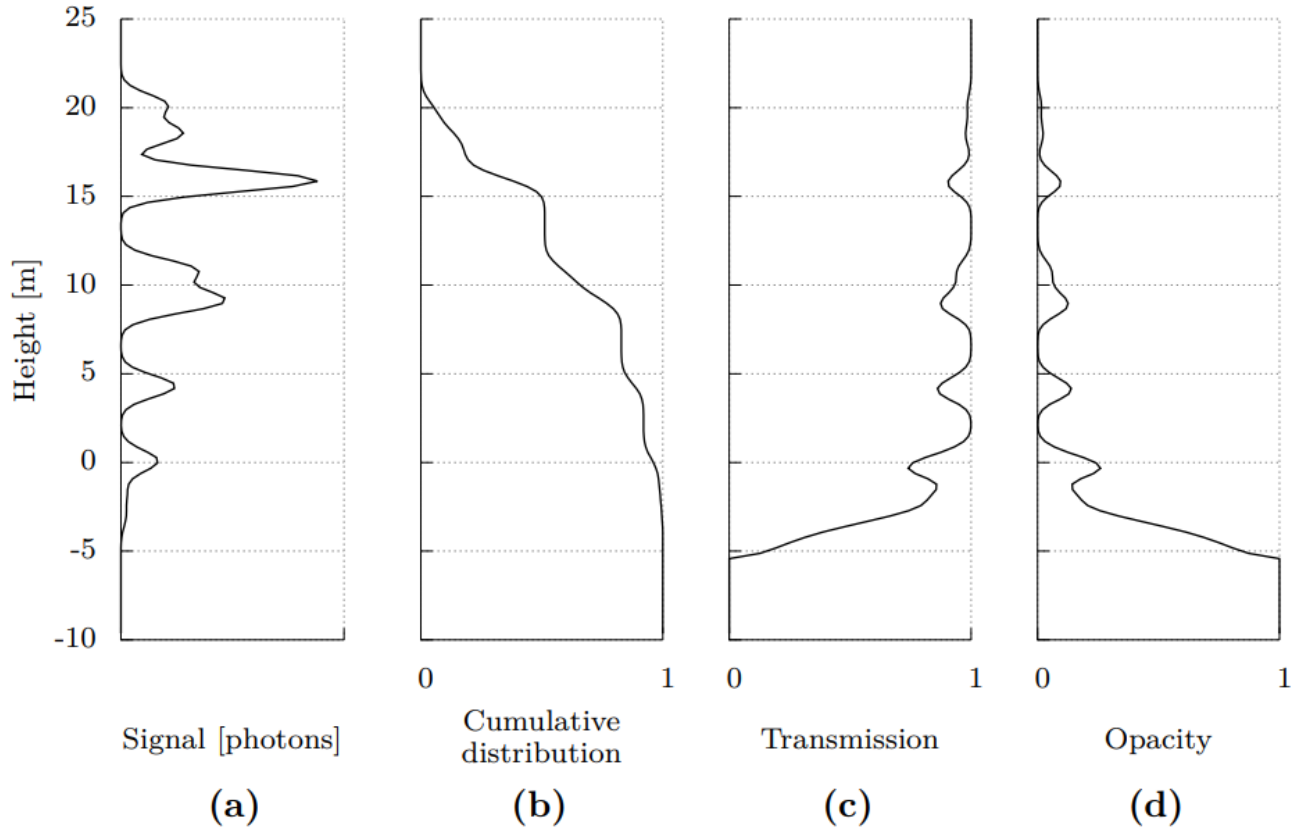

Figure 2: The main technique used by Hagstrom et al. to process waveforms [29]. The cumulative distribution is the sum of the area underneath the signal, and the transmission is equal to the current energy divided by the previous energy.

In turn, *Huang and You* introduced a 3D CNN model to label 3D point clouds using minimal prior knowledge, without requiring a segmentation step and hand-crafted features. The authors claimed that the model could handle large amounts of data. Annotated training data and the input 3D point cloud were both voxelized and fed into the 3D CNN. Voxel labels were generated and converted back into a point cloud. One experiment on a 3D urban model resulted in 75% or higher prediction rates of all objects, except for uncategorized objects, many of which were predicted as trees. The input size for

the 3D CNN in the pipeline was 20^3 voxels. Additionally, there were two consecutive sequences of convolutional layers, followed by a max-pooling layer. Finally, there is a fully-connected layer, a softmax layer, and eight object categories at the end. One of these categories is background, and the other seven are building, tree, pole, car plane, wire, and others [30].

Kukenbrink et al., on the other hand, created another voxel transversal algorithm, also from discrete return lidar data. For this algorithm, basic ray-tracing and hit counting were used, along with the four-category system. The main purpose of this paper was to assess the occlusion of canopy vegetation that was measured from airborne lidar data. High resolution terrestrial laser scanning (TLS) for a field plot was used to evaluate the lidar collection parameters. Results showed that 28% of the vegetation elements detected by the TLS system were not detected by a corresponding ALS system, mainly due to occlusion. A voxel size of 0.5 m resulted in the observed canopy being only 20% of the total canopy volume, and the occluded and unobserved canopies 40% each of the total volume. Additionally, a larger flight strip overlap was found to significantly increase observed canopy volume due to angles and pulse density [24].

Finally, *Hancock et al.* used FW lidar data to develop predicted fractional covers for each 3D voxel. This method is a significant advancement in terms of prediction accuracy, which was based on target reflectance assumptions, such as that the denoised waveform is a sum of *n* Gaussians, and that the visible target area is representative of the obscured target. The algorithm parameters were calibrated using terrestrial laser scanning (TLS) data from eight locations for various 3D vegetation structures. For the TLS data, all voxels were assumed to have a unit size of 0.5 m.

This study is considered to represent a significant step in terms of waveform voxelization approaches, since ALS data were correctly measured at 1.5 m horizontal and 50 cm vertical resolution. Additionally, understory vegetation and canopy structure were also successfully measured. Another result was that full-waveform data were shown to contain far more relevant information than discrete

return lidar or Gaussian decomposition. Finally, various structures within the closed canopy were identified, which was not possible with earlier methods. The sparse understory and pathways, obscured by trees, were mapped at a 1 m height above ground. Buildings were mapped at a 3 m height, and tree tops were at a 10 m height [31].

These studies all resulted in good predictions using different types of training data. These include single discrete returns, multiple discrete returns, and full-waveform data. However, all these methods use point-cloud data, as opposed to voxel data. The main advantage of using training voxel data is that each object is segmented into several voxels, which are cubical units. This arrangement allows the classification algorithm to better detect and categorize overlapping elements, arguably resulting in greater overall accuracy. Next, we will discuss the background and utility to a simulation-based approach to address these shortcomings.

DIRSIG AND PHYSICS-BASED MODELING

There are many software tools available to simulate plausible digital imagery and other data products of virtual 3D scenes. These tools typically use physics-based algorithms to calculate the spectral radiance that reaches an imaging system or other photon-detecting device, after interaction with a virtual representation of a real-world scene. The fidelity of the overall simulation is heavily dependent on the fidelity of the virtual scene, regardless of the software tool. Therefore, to represent forest elements for lidar simulations, both the geometry as well as the reflectance, transmission, and absorption parameters of the forest canopy should all be accurately modeled. These parameters should be modeled in the visible and near-infrared regions of the electromagnetic spectrum, since these regions match the wavelength range of operational lidar sensors [32].

We used the Digital Imaging and Remote Sensing Generation (DIRSIG; V5) tool for this research. DIRSIG has been continuously developed over the last 30 years at the Rochester Institute of Technology (RIT) Chester F. Carlson Center for Imaging Science (CIS). This tool is first-principles

and physics-based for the domain of unpolarized electro-optical radiometric transport [25]. I.e., the simulation methods do not originate from derived high-level approximations, but rather from fundamental physical principles in the realm of geometric optics and visible-to-infrared electromagnetism. DIRSIG can produce a variety of synthetic data products, including multispectral imagery, hyperspectral imagery, and both discrete and waveform lidar returns. Additionally, DIRSIG has been referenced in over 1100 research papers that cover a wide variety of applications, some of which are listed below.

DIRSIG uses a Monte-Carlo path tracing method to determine the photon absorption and scatter percentages. In other words, the algorithm integrates over all the incident radiance at each surface point by constructing random transport paths. These paths are distributed in accordance with the bidirectional scattering distribution functions (BSDFs) of the attributed material properties.

The main categories needed for the DIRSIG simulations (Figure 3) include scene geometric properties (describing the shapes of the objects), scene optical properties (describing the reflectance, transmission, and absorption of the object surfaces), and instrument properties (sensor type, location, and detailed parameters). Atmospheric properties are defined by the MODerate resolution atmospheric TRANsmission (MODTRAN), a software package, or manual inputs [33]. Instrument forward propagation and scanning are simulated using sensor location, orientation, and temporal sampling information.

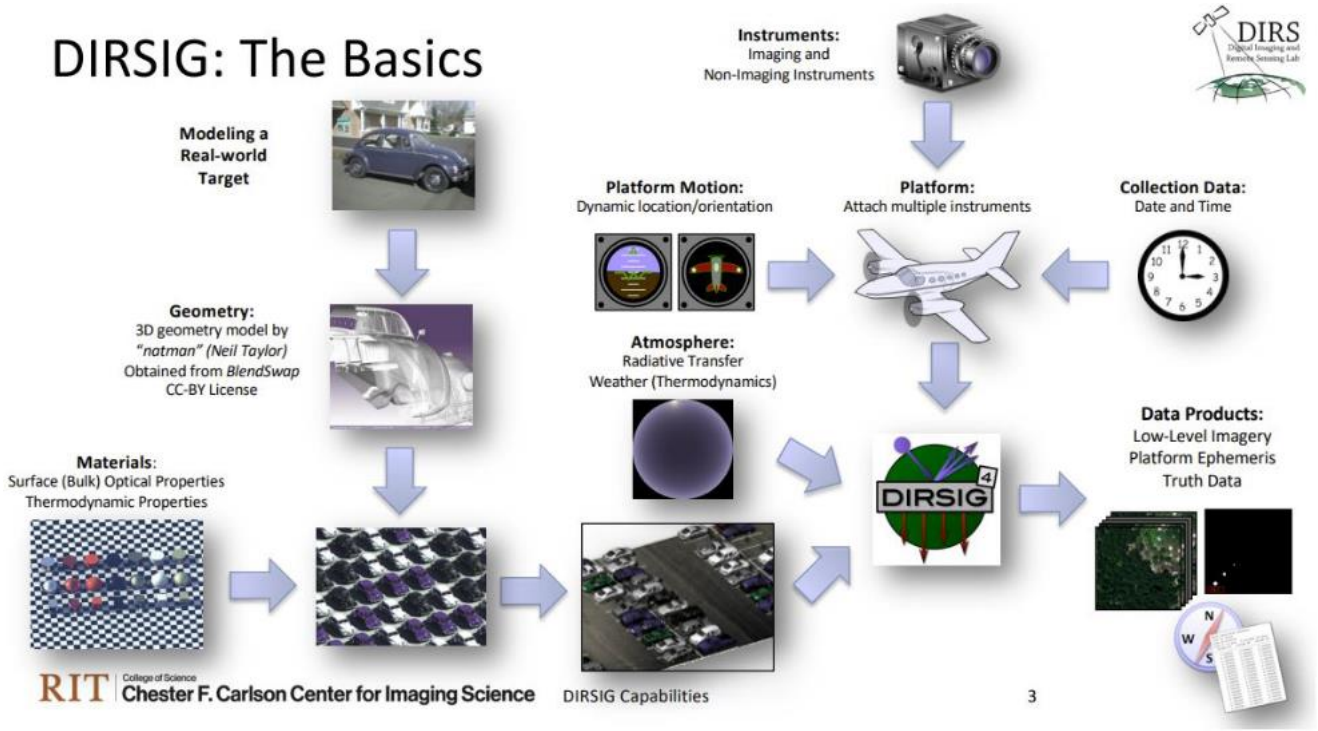

Figure 3: Diagram of DIRSIG integration for the forest scene. The scene geometry is integrated with the optical properties and system specifics. This is to render the scene for different sensing modalities in a first-principles environment.

DIRSIG has been used for a range of research applications, including assessing waveform deconvolution and preprocessing [34], evaluating the level of detail that can be extracted using a 1064 nm wavelength waveform lidar system [11], optimizing in-field leaf area index measurements (LAI) [13], and detecting lidar backscatter phenomenology [14]. This approach has been shown to provide accurate representation of observed lidar phenomena. For example, "below ground" system returns were initially thought to occur as a result of background noise and are often omitted in analyses. However, further research demonstrated that this phenomenon might result from near-ground dense grass and coarse woody biomass, while similar delayed returns were observed in operational waveform lidar data [35]. We will next detail our proposed to voxel classification, namely the convolutional neural network (CNN).

CNN ALGORITHMS

In the field of computer vision, a CNN is an analytical model which classifies images based on 2D convolutions and other methods of feature learning [36]. A 2D convolution takes a square moving filter (Figure 4), and moves the center kernel pixel across every image pixel; note that padding usually is needed for the edges to account for shape mismatch. Then, the points are replaced by values which are calculated using a convolution operation from the filter. For the process of convolution, element-wise multiplication is first performed by multiplying the image pixel and its surrounding values with the corresponding filter values. The initial pixel value is then replaced by the sum of the multiplied values.

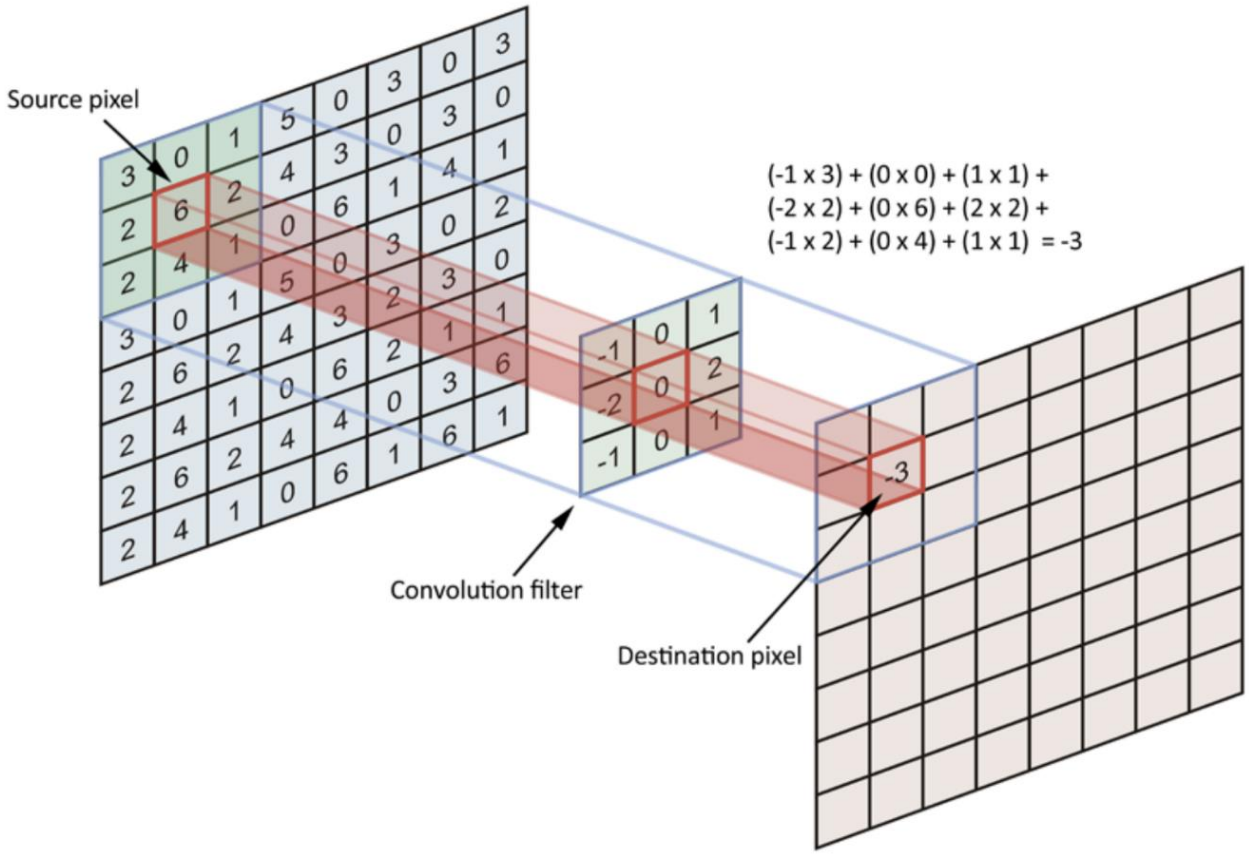

Figure 4: Example diagram demonstrating the process of 2D convolution. Each pixel in the filter is multiplied elementwise, then added together to form the new sum [37].

The resulting array is known as a feature map, and there is one map for every filter which is used. These feature maps are passed through a chosen activation function that determines whether

certain image features appear, based on certain conditions. After this step is completed, one option would be to add more layered filters, which would increase the depth of a typical CNN. Additionally, maximum pooling layers could also be implemented, where the largest values from each array region are selected and returned in a smaller array [37].

Three-dimensional CNNs operate similarly (Figure 5), with the difference being that the input is a 3D cube and the filters and other layers are also cubes. Note that the 3D version of a pixel is a voxel, or a "volumetric pixel". 3D CNNs are mostly used on 3D image data, such as magnetic resonance imaging (MRI), computed tomography (CT) scans, and videos. A video is a sequence of image frames in a row, implying that they contain useful spatial features when stacked in the third dimension. Generally, 3D CNNs produce more accurate results than 2D CNNs, because of the extra information per voxel and different layering that can be analyzed [15].

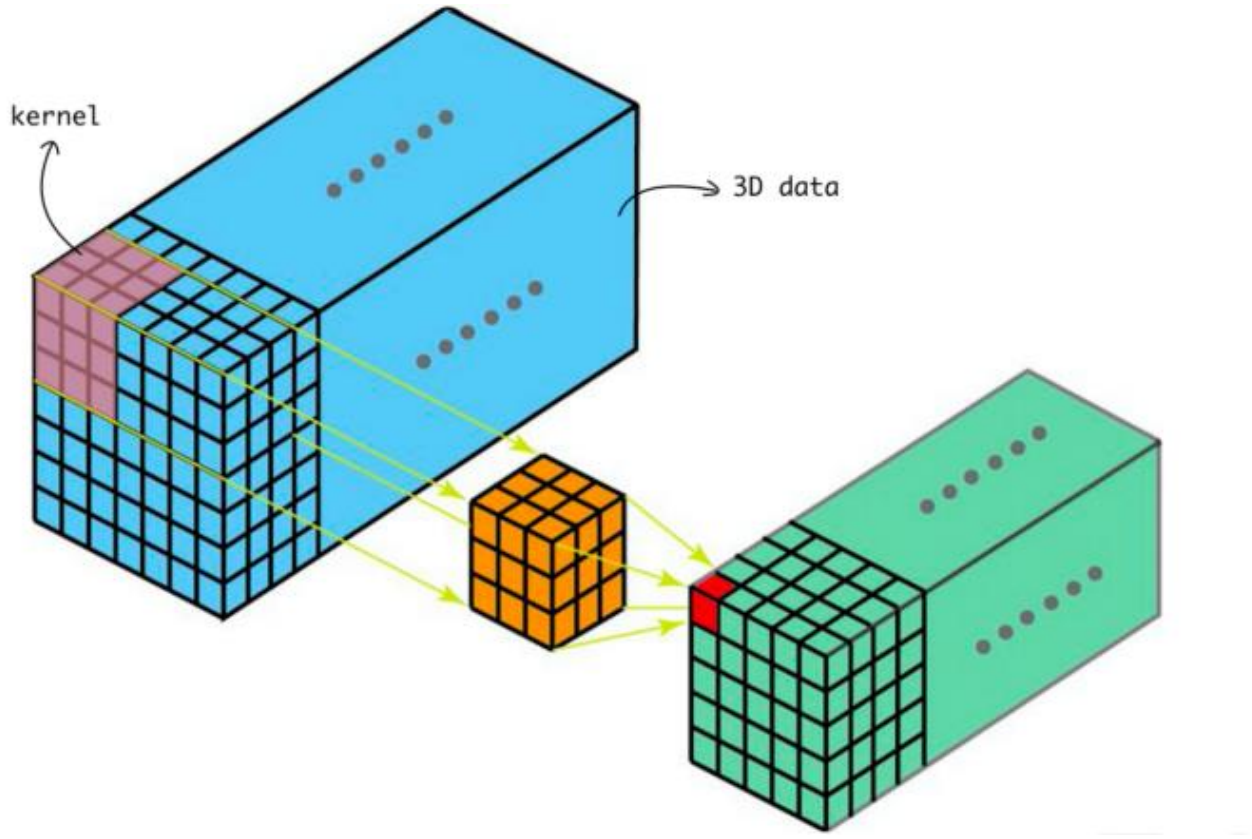

Figure 5: Example diagram demonstrating the process of 3D convolution. In this instance, both input and filter must be 3D for elementwise multiplication [15].

3D CNNs also have been used for deep learning to classify lidar sensor data. One method developed for this purpose, namely the point-voxel CNN, represents the input data in *points* and performs the convolutions in *voxels*. This was done to make predictions efficient in terms of memory usage and computation [16]. Another example of a 3D CNN model that was developed is VoxNet, which uses an occupancy grid representation to continuously detect objects from a moving camera [17]. The 3D CNN for that model is supervised.

Next, we will detail the methods and approaches in this study. The site study and data, along with DIRSIG lidar point cloud preprocessing are examined. Voxel algorithms which were studied include Support Vector Machines and 3D CNN methods. Additional aspects of these algorithms are also discussed.

Methods

SITE STUDY AND DATA COLLECTION

The research site for this study is a sizable portion of the 16.187 km^2 Harvard Forest. The Harvard Forest is owned and managed by Harvard University and is in Petersham, Massachusetts, USA. Harvard Forest research is funded by the National Science Foundation and the Department of Energy. The research site, seen in Figure 6, is called the "Mega Plot" and is located on a part of the Harvard Forest named Prospect Hill. The dimensions of the Mega Plot are roughly 700 m x 500 m. One of the research projects on this plot was conducted by the Smithsonian Institute's Forest Global Earth Observatory (ForestGEO), and took place from 2010 to 2014. During this project, scientists and students from ForestGEO measured, tagged, and geo-located all woody stems greater than 1 cm diameter-at-breast height (DBH) [38].

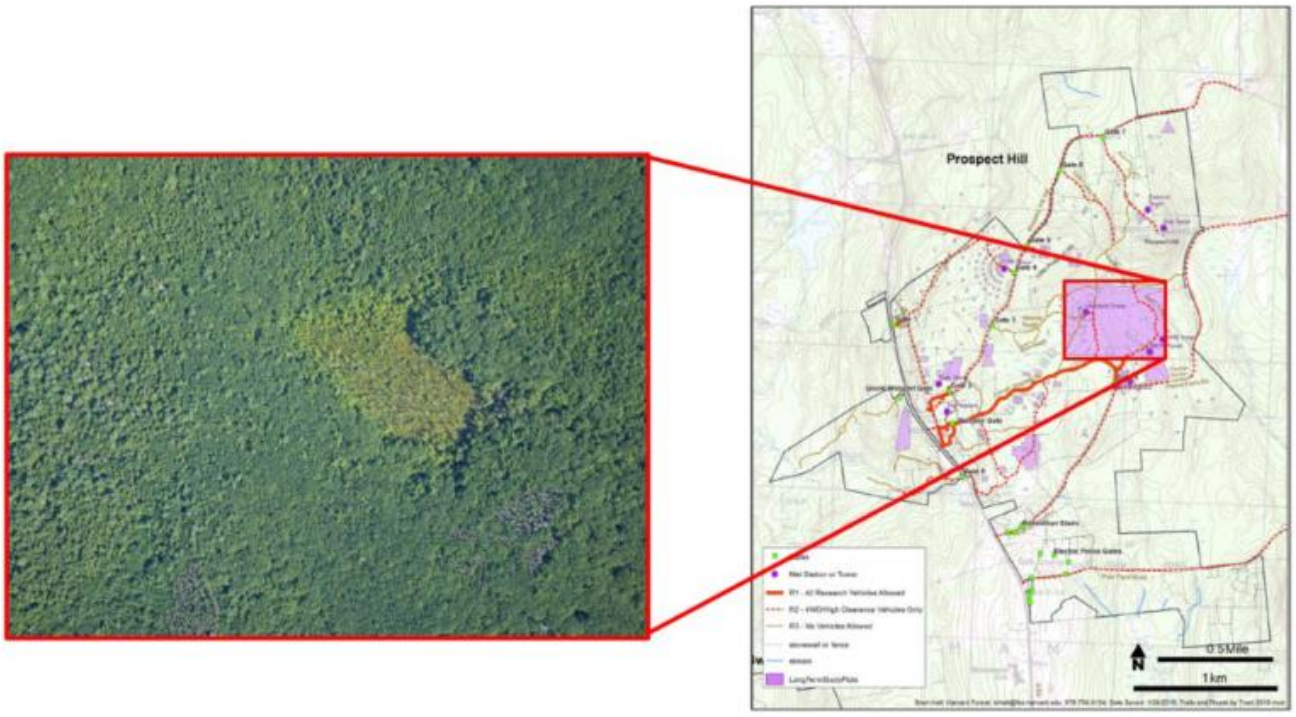

Figure 6: High-definition view of the Mega Plot (left), shown as being part of the entire Prospect Hill site (right). The wetlands in the center area lighter shade of green and stand above the rest of the forest.

POINT CLOUD PREPROCESSING

Simulated waveform lidar data were generated using Digital Imaging and Remote Sensing Image Generation (DIRSIG) sensor data parameters. The simulated data were based on RIT's simulated VLP-16 drone, in waveform lidar and not discrete (typical) lidar type, and NEON's Optech Gemini airborne platforms, which were validated based on point density, footprint, and system specs. The simulated VLP-16 data generally have more background noise, but contain more three-dimensional detailed information about the forest. In contrast, the simulated NEON data have very little background noise due to the high altitude, though most of the data only consist of the main forest structural components and the ground.

The VLP-16 drone (Figure 7) typically is flown at an altitude of 88 m above ground-level (AGL) and at a speed of 4-5 m/s. On the drone, 16 infrared (IR) lasers, paired with IR detectors, are used to measure distances between the drone and the object. The laser-detector pairs constitute an array which scans the environment for the entire 360^{o} through rapid spinning. Each laser fires at a frequency of 18.08 kHz and can register two returns per pulse. The scanner has a forward-moving push broom design, because the lasers collect all the data at the front (Table 1). This drone model can detect street signs, license plates, and lane markings. In single-return mode, approximately 300,000 points/second are detected and returned [39].

Table 1: Specifications of the VLP-16 lidar drone. The resolution is limited, but there are 16 laser channels.

VLP-16 lidar Specifications	
Channels	16 lasers
Range	Up to 100 m
Range accuracy	Up to ±3 cm
FOV (Vertical)	-15^{o} to +15^{o}
Angular Resolution (Vertical)	2^{o}
FOV (Horizontal)	360^{o}
Angular Resolution (Horizontal)	0.1^{o}-0.4^{o}

Figure 7: The first set of simulated data is based on truth data from the Velodyne VLP-16 lidar sensor, which is integrated on a Matrice 600 unmanned aerial system (UAS). The drone is a MX-1 sUAS remote sensing platform, developed by CIS faculty at RIT. Additionally, the drone contains a Mako G-419 RGB camera and ballast weight in place of the hyper-spectral camera [40].

The reconstructed data, based on an earlier version of the drone, use a 21.7 kHz laser frequency. Therefore, there are many excessive simulated points. One issue is that, due to errors with the sensor and post-processing techniques, some points do not land in the correct voxel, so they are incorrectly categorized. The best way to reduce the margin of error, which is +/- 0.05 m, is to use a smaller voxel size [41].

By contrast, the NEON Optech Gemini system (Figure 8) is flown at an AGL height of around 1000 m at a speed of 50 knots, or approximately 92 km/hr. The Gemini instruments are mounted in a rigid frame, which is installed in a Twin Otter aircraft, a type of light airplane. Due to the high altitude, there is only one pixel, with a FOV of +/- 25º. However, the laser fires at a rate of 100 kHz and can receive up to four returns/pulse (Table 2). The scanner has a side-to-side whiskbroom design which uses more power, but adequately compensates for the high operational altitude [21].

Table 2: Specifications of the NEON lidar system. The Gemini waveform lidar system is operated at a much higher altitude than the UAS-based VLP-16 system.

NEON lidar Specifications	
Operating altitude	1000 m
Wavelength	1064 nm
Pulse repetition	100 kHz
Scan frequency	50 Hz
Beam divergence	0.8 mrad
Scan angle range	±18.5º
Footprint size	0.8 m

Figure 8: Rendering of the NEON Optech Gemini system. Because of the high level of laser power, the apparatus needs to be flown in a light aircraft at a high altitude [42].

Voxelized representations of the truth data were generated using the VoxelizeHDF command line tool, which was created by *Saunders et al*. in order to generate reference voxel data from DIRSIG scenes for this project [43]. The coordinate boundaries of the VLP-16 dataset plots were taken from

the filtered version of Plot B, Dataset 3, which contains all the material types. The probability of object detection is the highest for this plot, and there is a mixture of conifers and deciduous trees. Additionally, there are low LAI values due to canopy structure, which translates into the prevalence of dead wood, and multiple sub-canopy objects. Each truth voxel contains the estimated background, leaf, bark, ground, and object area in m^2 units. The minimum (x, y, z) coordinates were (594.44, 50.51, -12.78) m and the maximum coordinates were (647.6, 117.05, 14.41) m.

The coordinate boundaries of the NEON plots were selected so that a truck and tent were included. The minimum coordinates were (600, 75, -12.4) m and the maximum coordinates were (650, 125, 20.08) m. A second NEON plot of size 150 m x 150 m was created to test whether more data (voxels, structural variability) would yield better results. This time, as shown in Figure 9, cars which were rotated at different angles, along with a tower, were included. The minimum coordinates were (550, 0, -13.43) m and the maximum coordinates were (700, 150, 35.7) m.

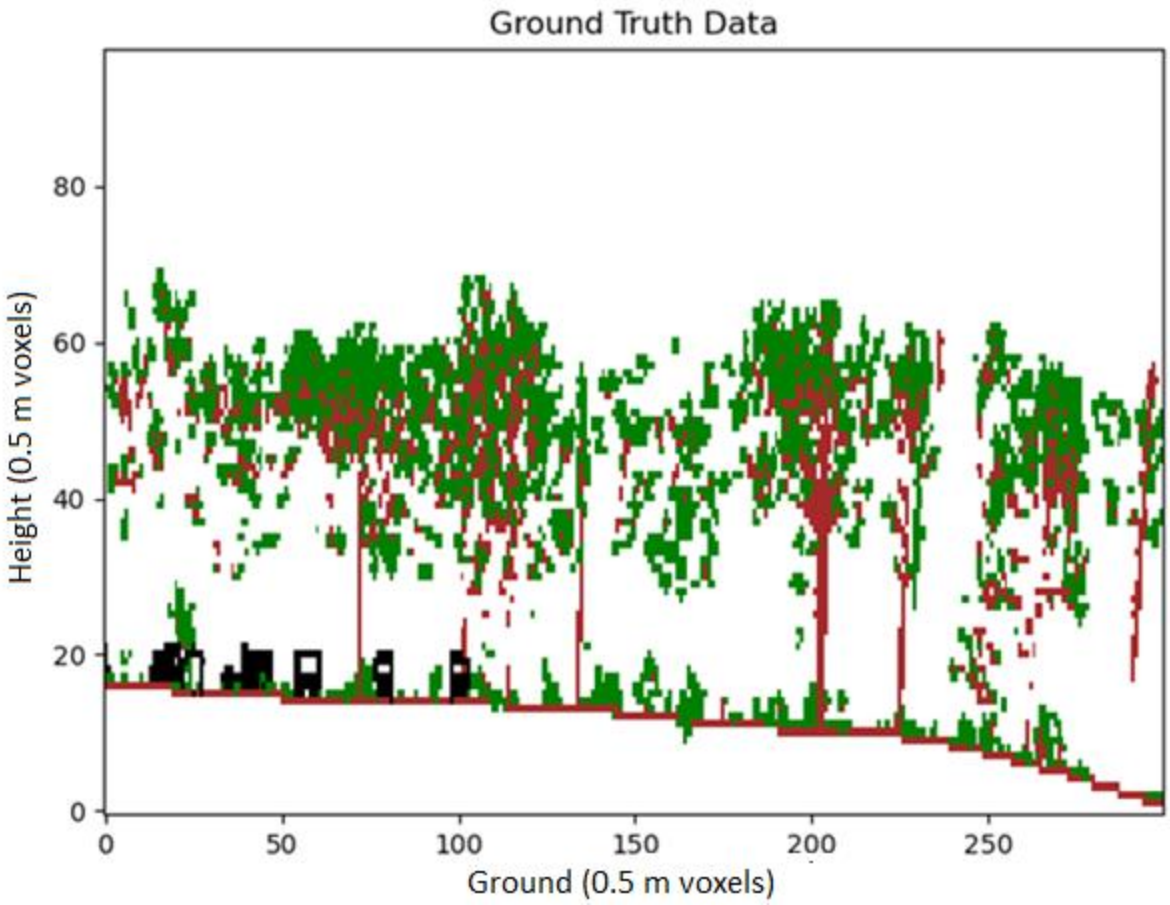

Figure 9: Slice of the enlarged 150 m x 150 m NEON plot. In this region, several cars (shown in black) are rotated at different angles for greater variety in truth data.

The complexity of the reconstruction algorithm was based on the number of applicable waveform paths. Initially, the reconstructed data were stored in multiple bin files and were extracted using a voxelizer Python script. The script used the d5lidar library [44], also created by *Saunders et al.*, along with an included *for* loop for extraction. Each instance of reconstructed data contains the total scattered fraction of photons, the total remaining fraction of photons, and the number of voxel waveforms. All these data instances are added to their corresponding voxels by coordinate location.

The categories from the voxelizer are total absolute facet area (m^2) per category. They are disjoint, such that the total area for all facets in the voxel is the sum of the areas of the individual categories. The area of an individual triangular facet, as seen in $Equation\ 2$, is half the magnitude of the cross product of the edge vectors.

$$A_{facet} = ||\frac{1}{2}E_1 \times E_2|| \qquad (Equation\ 2)$$

For large facets that touch multiple voxels, as seen in $Equation\ 3$, the area is distributed by randomly sampling N points on the surface of the triangle and then adding $\frac{1}{n}th$ of the facet area to the voxel containing each point [44].

$$voxelAt(P_k) \leftarrow voxelAt(P_k) + \frac{1}{n}A_{facet} \text{ for } k = 1, \ldots, n \qquad (Equation\ 3)$$

There is a specific geometric ray, also known as a line-of-sight, for each waveform. The ray starts at the platform and points downward into the scene at an angle. Therefore, segments of the waveform should be measured against the structure of the voxel grid. Not only does each ray intersect many voxels, each voxel will also be intersected by many rays. The voxelized density field is created by calculating the average scattering density for each voxel. Only the relevant portions of intersecting waveforms are used in calculating the values. The method for estimating the average scattering density is the most important portion of the overall calculations. The estimated scattering density is reconstructed on a voxel grid from the waveform data. The voxel grid is parameterized by the coordinates of its minimum and maximum corners and the number of subdivisions on the grid. The corners are chosen to cover the area of the interest, and the number of subdivisions typically is chosen to obtain approximately cubic voxels with an edge length between 0.5-1.0 meters.

For each waveform, which may originate from either a real or simulated dataset, we distribute an estimate of the scattered energy, as a function of distance, to the voxels intersected by a cone-shaped distribution of rays concentrated around the associated central line-of-sight ray. This method contrasts with discrete data, where the average intensity of all the returns is taken for each voxel. Note that if a ray intersects a voxel, it does so at exactly two points. The first intersection marks the entrance, nearer to the sensor, and the second intersection marks the exit, nearer to the ground. Each of these points corresponds to a distance from the sensor and thus an ordinate in the waveform curve. We evaluate the proportion of returned energy at the voxel in question as a ratio of waveform integrals: the ratio of the

integral of the waveform between the ray intersections to the integral of the waveform from the first intersection to the end. In other words, we evaluate the fraction of the remaining energy in the waveform which is accounted for by the voxel at hand. This calculation is consistent with a Beer-Lambert law assumption, and thus helps account for the tendency of the waveform returns to decay with distance [45]. Lastly, we average the scattering estimates at each voxel after processing all waveforms by dividing the value at each voxel by the number of rays that contributed to it, but only if this number is non-zero of course.

Reconstructed data serve as the input for the machine learning algorithm. The voxels of these data represent the relative scattering intensity as a function of spatial location, which is a unitless quantity between 0 (never scattering) and 1 (always scattering). This is so each voxel contains the fraction of scattered photons which passed through that voxel. Additionally, converting numbers in that range to a different range is facilitated. If more than one waveform passed through the voxel, the intensity fractions are divided by the number of waveforms to find the average. Finally, in order to maintain reasonable processing/computational requirements, voxels were set to 0.5 m in size.

For the reconstructed simulated VLP-16 waveform lidar data, such as the slice in Figure 10, relatively few waveform paths were used due to memory limitations. As a result, the reconstructed data exhibited significant background noise, which was slightly above zero. However, the ground line is clearly indicated by the gradient at the top of the bottom section's high intensity region. The basic canopy shapes and respective locations are also well-defined.

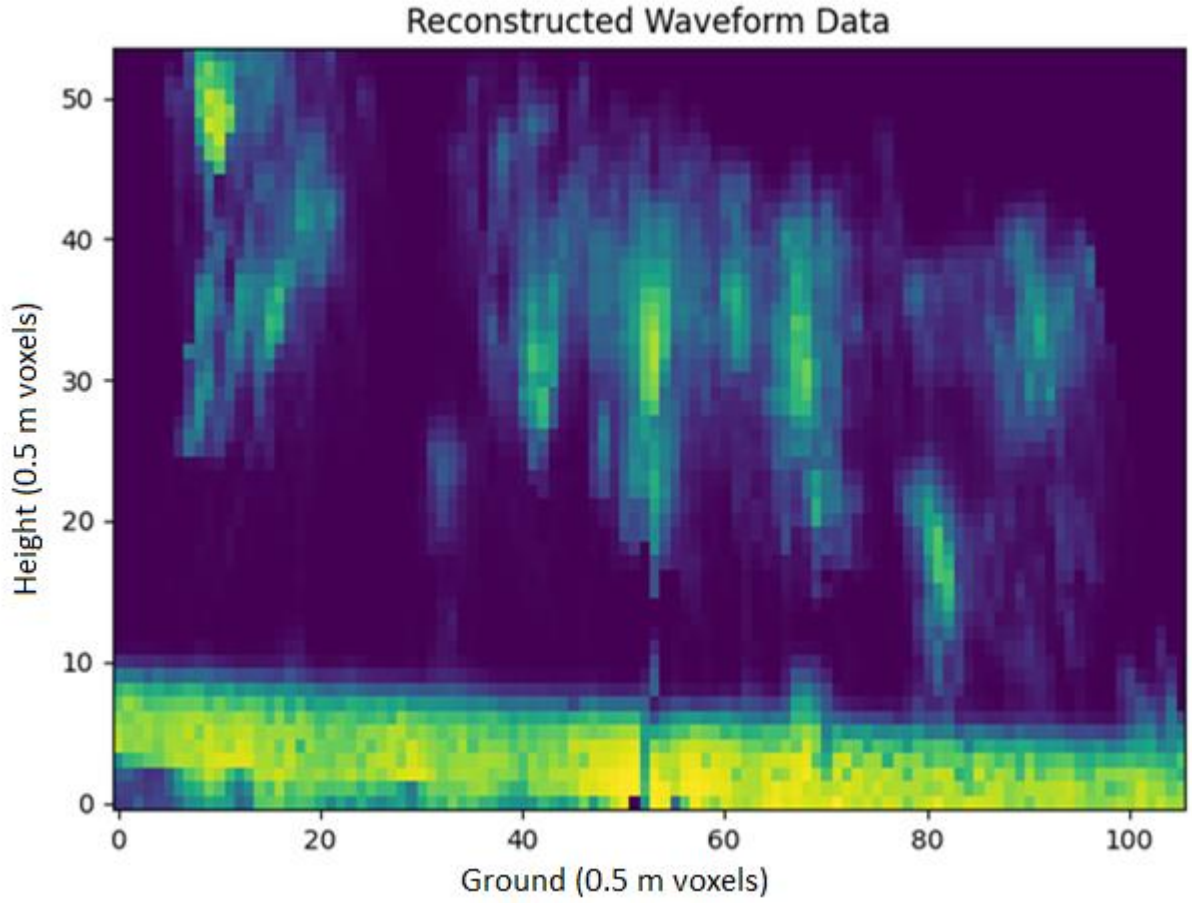

Figure 10: 2D vertical slice of simulated VLP-16 waveform lidar data, showing the fraction of reflected photons for each voxel. There is significant background noise.

For the simulated NEON waveform lidar data, such as the slice in Figure 11, there is little background noise, due to the high altitude (1000 m) data capture. Therefore, separating the forest from the background is relatively simple. Since the data are returned as limited increments due to system power limitations, the plots resemble a series of interconnected rectangles. Bark prediction is challenging, given the relatively limited data. However, like the simulated VLP-16 data, the ground line is clearly indicated by the gradient at the top of the bottom section's high intensity region. Objects are easily distinguished based on their closeness to ground level and relatively high scattering intensities.

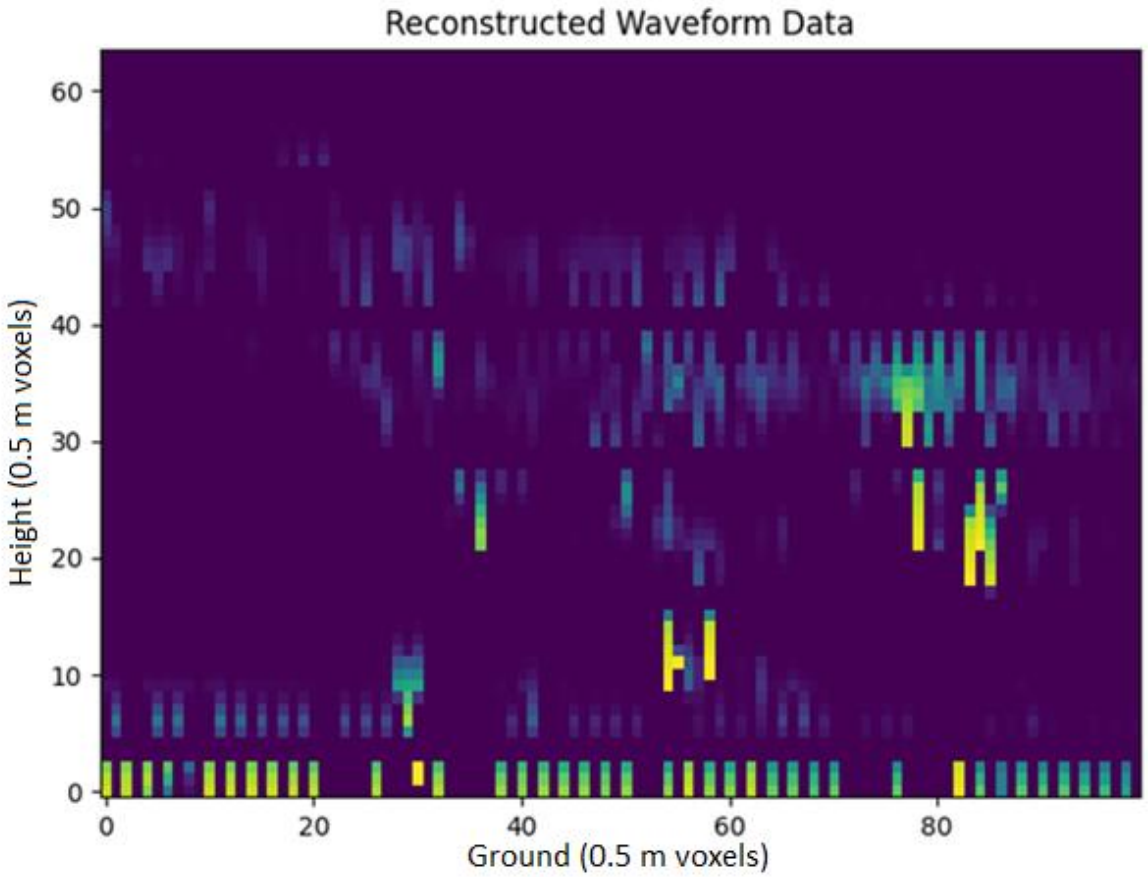

Figure 11: 2D vertical slice of simulated NEON waveform data, showing the fraction of reflected photons for each voxel. The data is highly sparse because of the relatively high altitude (1000 m AGL) of the Gemini system.

The ground truth data also were generated by DIRSIG. This geometric scene is based on actual sensor data that were collected at the Harvard Forest. The scene in Figure 12 contains over 100k geolocated plants and trees, with real world spectral and geometric features. The DIRSIG scene file format utilizes the well-known Hierarchical Data Format Version 5 (HDF5) to represent real-world objects as geometric primitives, typically collections of triangles, associated with spectral material descriptions and other metadata. A command-line program was written to process the DIRSIG HDF in order to establish the ground-truth voxelization.

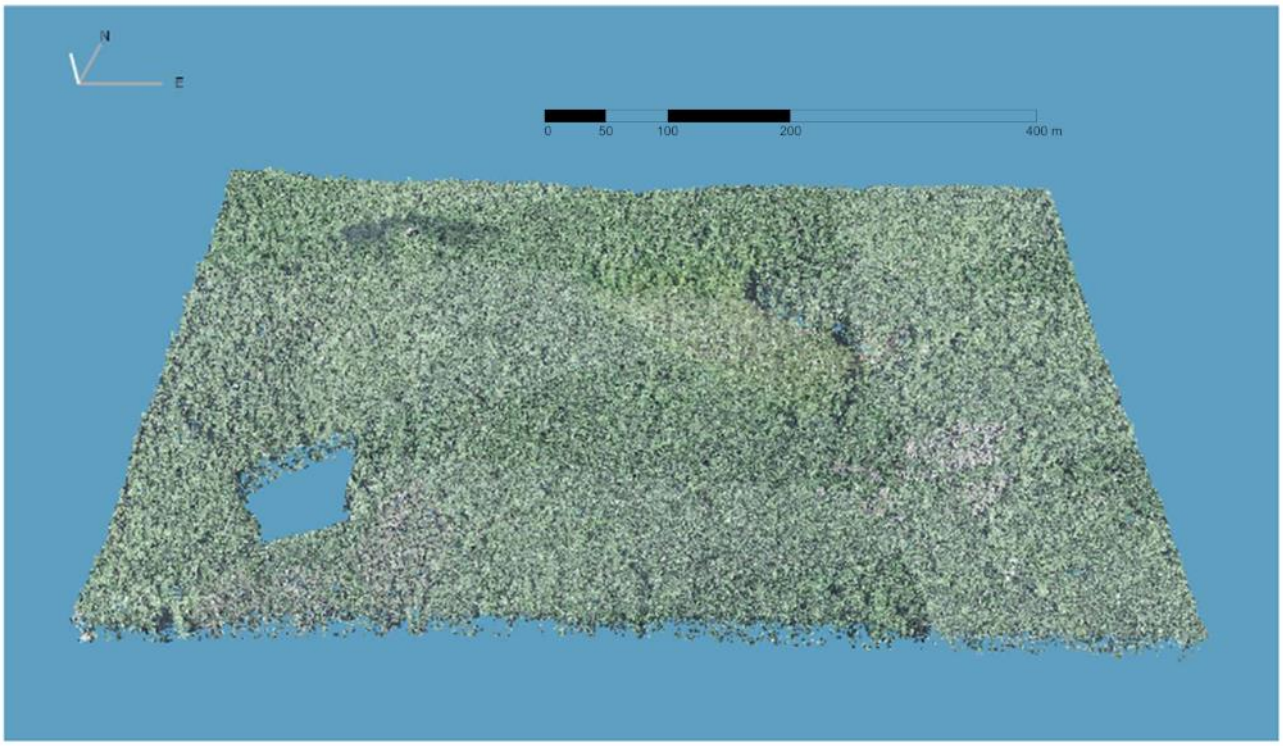

Figure 12: 3D point cloud plot of the Mega Plot, located on Prospect Hill. There are over 100k geolocated plants and trees on this plot.

The user must input a region to voxelize, such that the boundaries are determined by the maximum and minimum X and Y coordinate values. Additionally, the number of subdivisions for the array also must be specified. The program then iterates all triangles in the scene which overlap the specified region and accumulates the total one-sided surface area present in each voxel. The extraction results are stored in four binary files based on the type of forest item (class), which are distinguished by matching the material identifiers associated with the scene geometry against regular expressions. These class categories are background, foliage, bark, man-made objects, and ground voxels. A background voxel is a voxel without any surface area.

The ground truth voxels are the output data for the machine learning algorithm. Truth voxels contain five surface area values for each type of item contained within the voxel. These voxels were also set to 0.5 m in size in order to maintain consistency with the input data. The five categories are numbered in Figure 13 as background (0), leaf (1), bark (2), ground (3), and object (4). The highest

category surface area for each voxel is selected as the main fill, and the value of the voxel is set to the corresponding numbers as output labels.

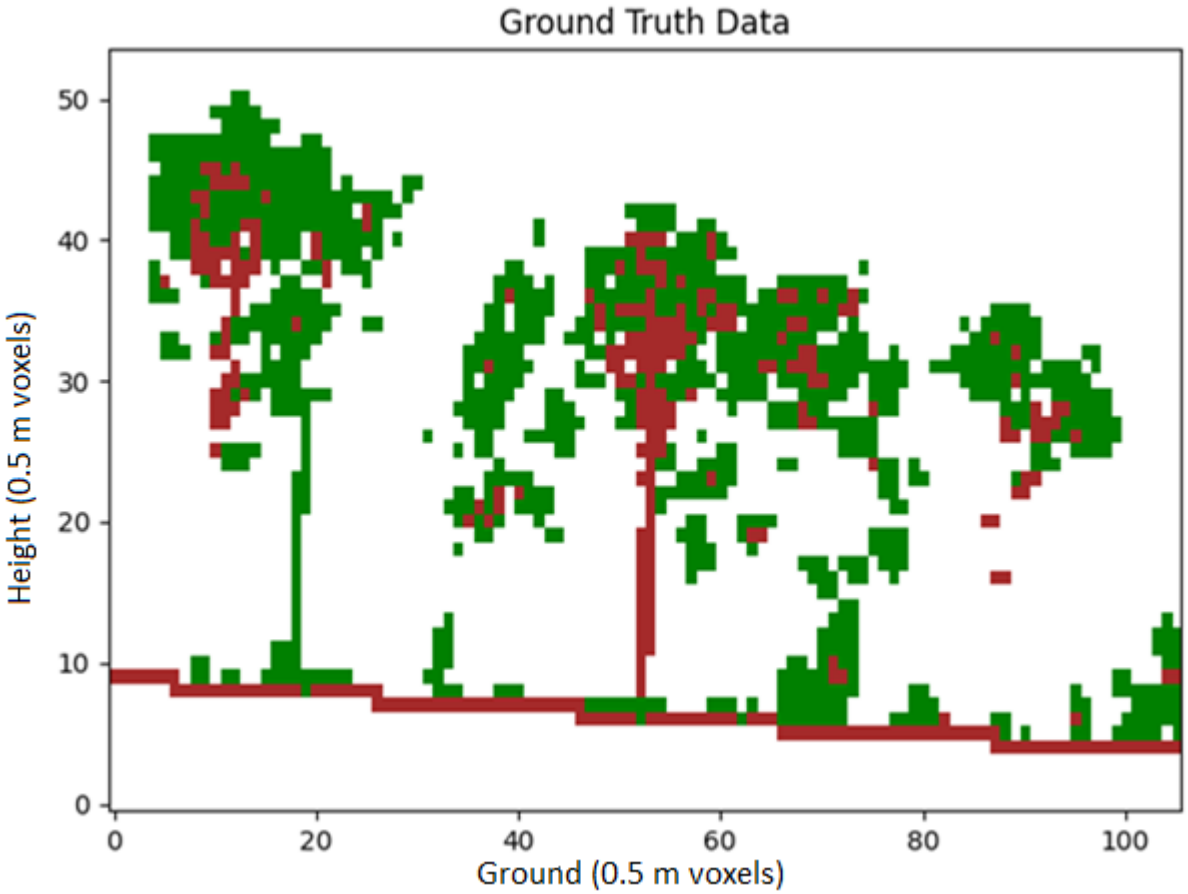

Figure 13: Vertical slice of the ground truth data. The leaves are green, and bark and ground are brown. Objects, not pictured, are black.

Using maximum surface area to create the categories implies that each voxel mostly contains a single kind of forest object type. This would be ideal if the voxel size were extremely small, so that a voxel could only contain a few small branches or a few leaves. However, some voxels may contain two or even three class areas that are close to each other. These voxels often are the most difficult to accurately predict. Typically, they are located inside the center of the tree crowns. One possible solution would be to figure out how to use smaller voxels when predicting data. Another solution would be to create more categories, where the new voxel categories would contain roughly equal areas from two different classes. After that, the categories would be classified again in another CNN model.

Finally, simulated VLP-16 or NEON waveform data were far more accurate at a lower than a higher altitude. This is because the ground and objects are sampled in a denser configuration, and are located at a lower level as a result. The point density is much higher, making simulation easier to perform due to a larger input dataset. The real data, seen in Figure 14, are also reconstructed from waveforms. However, most of the returns are from the canopy level. This is because most real NEON data cannot reach the ground level due to the extensive foliage in closed-canopy forests. This can be visualized through the attenuated raw waveforms, shown in Figure 15.

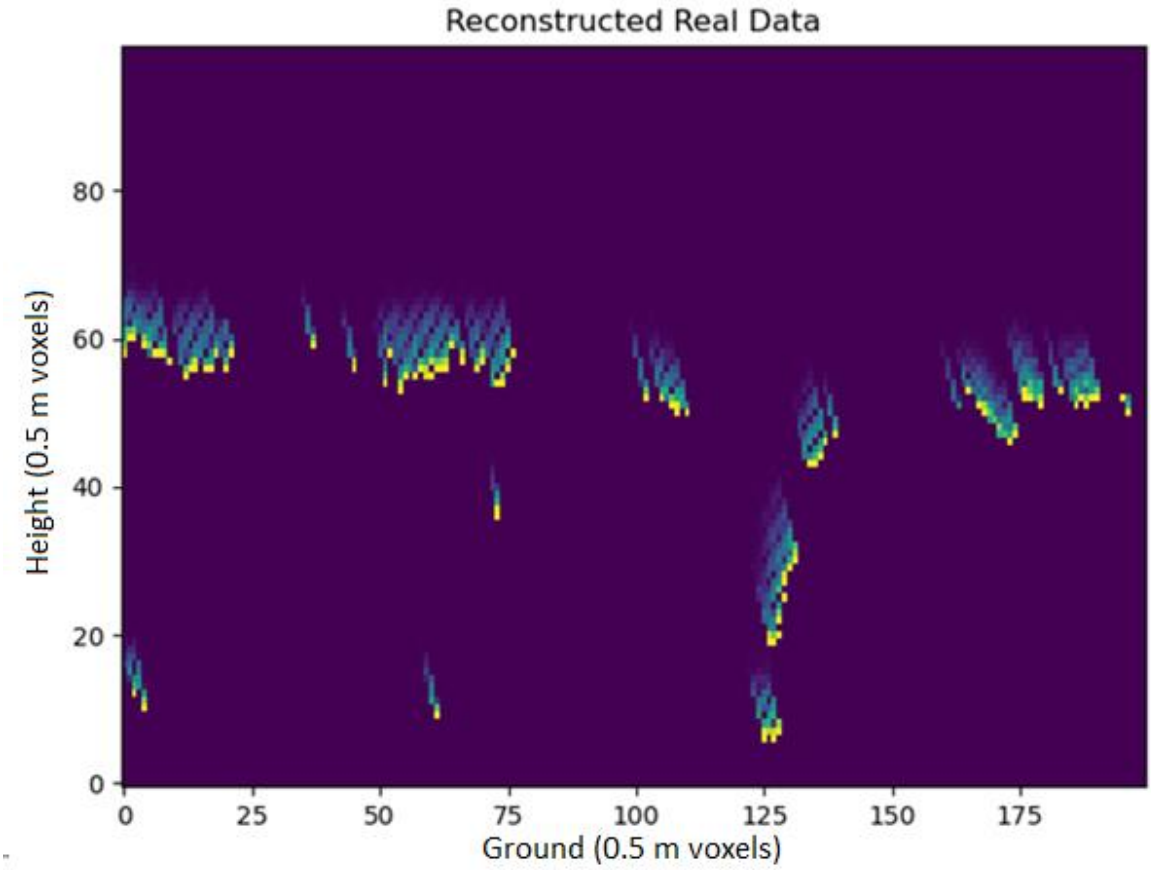

Figure 14: Vertical slice of the real NEON data, showing the fraction of reflected photons for each voxel. Most real waveforms are not able to reach the ground due to the foliage.

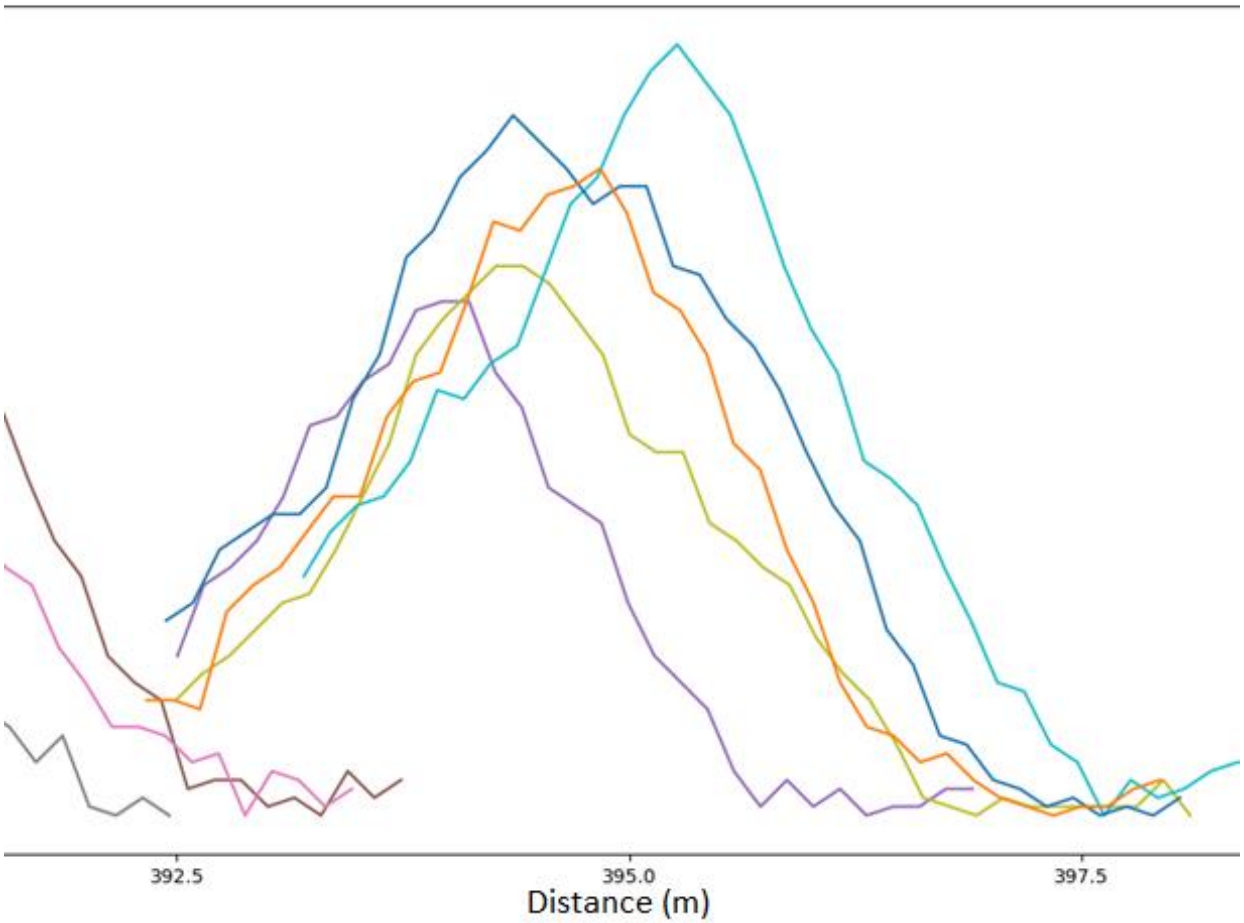

Figure 15: Sample of raw waveforms which were used to reconstruct the real NEON data. Most waveforms were truncated at relatively near distances, showing that ground could not be reached.

VOXEL ALGORITHMS

Support Vector Machines (SVM)

A simple support vector classification (SVC) algorithm initially was used for voxel prediction. A classifier has only one unit that produces the activation output, so the SVC classifier was used on the dataset to test the effectiveness of a basic machine learning algorithm [46]. SVC is a type of support vector machine (SVM), which uses a supervised learning method [47]. The software library used to implement the algorithm was *scikit-learn*, which is available as free software [48]. SVC performs multi-class classification on the dataset. The margins separating the hyperplane are maximized, and penalties are incurred when a sample is misclassified or too close to the boundary [49].

The inputs were 5x5x5 arrays surrounding each waveform voxel (zero padding was applied around the waveform voxel array for edge points). This array size was chosen because a 3x3x3 approach yielded poor accuracy, with recall and precision both less than 25%. Anything larger had distinctly slow runtime; after two hours, I stopped running the program. Classifiers only work with 1D inputs, so the inputs were all flattened to size 1x125. Additionally, the z-coordinate (height above ground) of each voxel was appended to every input, since height arguably is a significant factor in predicting the ground voxels. The five previously-mentioned voxel category labels were used as the respective outputs.

The data were split into training and testing sets, where each set contained 50% of the entire data. There were almost 300,000 total voxels used in both training and testing, so a very specific percentage is not critically important in this case [50]. Additionally, to minimize the effect of the voxel location, the data were randomly shuffled at the beginning. Voxel indices were temporarily appended to the data, so that the absolute location of each point was preserved. The SVC algorithm creates several hyperplanes, which split the inputs into several classes. Additionally, the kernel that is utilized in $Equation\ 4$ $Equation\ 4$is the radial basis function (RBF) kernel. The gamma parameter determines how much an input can influence nearby inputs. Gamma was set to 0.001 because there are many samples, so a minimal influence per input is ideal for forming well-defined predictions over many scenarios [48]. Shrinking was not needed, since most arrays surrounding each voxel have a relatively similar shape.

$$\exp(-\gamma\|x - x'\|^2) \qquad (Equation\ 4)$$

Although a convolutional neural network (CNN) utilizes far more resources than a conventional classifier, the corresponding predictions arguably are far more accurate [51]. A conventional classifier has only one unit that produces the activation output. In a CNN, there are multiple units in a single

layer, which are all classifiers. Additionally, there are multiple layers and activation functions which all interact with each other to generate a far more robust model.

3D CNN

For the 50 m x 50 m plots, 11x11x11 arrays surrounding each waveform voxel were used as inputs. Zero padding was used around the waveform voxel array for edge points. This array size was chosen because a slightly larger array size would require more memory than was available, and very small sizes do not have enough data to be filtered for the two max pooling layers. Every max pooling layer reduces the array dimension by half. Therefore, the minimum dimension size would have to be greater than $2^2 = 4$. Ideally, the array size would be 15x15x15, since much more data could be utilized. The five voxel category labels were used as the respective outputs. The prediction method used in this paper is unique, since 3D convolutional neural network (CNN) layers were used. Adding a third dimension typically improves the resulting accuracy [52]. A 3D array is equivalent to several 2D slices stacked together in a certain direction, and changes that occur throughout that direction provide additional significant patterns.

For the 150 m x 150 m simulated NEON arrays, the input parameters originally needed to be modified due to memory limits. Two methods were implemented: The input array size was reduced to 7x7x7 for the first method, and the input voxel dimension was increased to 1 m for the second method. Reducing the input array dimensions resulted in the inputs being approximately 25% of the original size. Increasing the voxel dimension from 0.5 m to 1 m resulted in the maximum number of inputs being 12.5% of the original maximum number.

However, when the program was run on the RIT Research Computing (RIT RC) Cluster [53], the 150 m x 150 m simulated NEON array could be run with 11x11x11 input arrays and the 0.5 m voxel dimension size. The computation time was significantly faster, and a greater number of objects could be added for further testing. Additionally, local z-values could be added as another filter. This information provided more detail, which made categorizing objects easier.

Initially, all the input voxels with zero intensity were discarded. This was done due to memory limitations on the local servers, and was based on the false assumption that all these voxels would be predicted as background. When these voxels were re-added, background and leaf precision noticeably declined. This is because the input voxels are not well-positioned with the output voxels. However, this led to a more accurate understanding of the CNN algorithms. The outputs seemed to be more averaged versions of the inputs in multiple directions.

The data again were split into training and testing sets in all cases, where each set contained 50% of the entire data. Additionally, to minimize the effect of the voxel location, the data were randomly shuffled at the beginning. Voxel indices were temporarily appended to the data, so the absolute location of each point was preserved. The training truth data were converted to a binary class matrix to make processing easier. A binary class matrix is a matrix of size *nxn*, where the left-to-right diagonal consists of ones, and all other values are zero. Both training and testing waveform voxel array inputs were then converted to 5D arrays, since the fifth dimension contains the values and filters.

Code for creating the fifth dimension was based on a program titled “3D CNN”, which was created by *Aggarwal* [54]. In this code, the fifth dimension was created by converting the array intensities to different (R, G, B) values of orange, by using the *ScalarMappable* mixin from Matplotlib [55]. The intensities were scaled in ascending order from light orange to dark orange. The scaling was sequential, meaning that the lightness value increases monotonically through the colormap.

The CNN workflow was programmed in Python, using components from the Keras framework for simplicity [56]. The optimizer used was *RMSProp*, which uses the root mean square. It is an extension of gradient descent, which is an algorithm used to find values of parameters that minimize the cost [57]. The scheduler used was ReduceLROnPlateau, which reduces the learning rate when relatively little improvement is detected. The CNN algorithm then was trained using the given training

data. After the algorithm was fully trained, predicted test fills were generated using the test inputs, and they were compared to the actual test fills using a confusion matrix.

The CNN architecture is based on the VGG16 architecture, which was developed in 2015 by *Simonyan and Zisserman et al.* at Oxford University. VGG16 has 16 layers, can contain up to 95 million parameters, and was trained on over 1000 classes. The maximum 2D input size is 224x224 pixels with 4096 convolutional features [58]. However, VGG16 usually works better with images that are smaller than 100x100, because large images can be computationally expensive. Because of the large number of supported parameters and trained classes, VGG16 would be suitable for 3D datasets with relatively small inputs. Overall, the VGG16 model is computationally efficient and performs well at classifying a wide variety of tasks [59]. Since the input array size is only 11x11x11, and the maximum overall array size could be 150x150x150, an architecture based on VGG16 would be ideal for this dataset.

The actual CNN, as seen in Figure 16, was built using a sequential model, meaning that each layer was simply added in sequence. First, there were two 3D convolutional layers. The first 3D convolutional layer used eight filters, and the second layer used 16 filters. For the convolution kernel, the size was set to 3x3x3. After that, a *MaxPool3D* layer was added which reduces the size of the array, returning only maximum values. This was followed by two more convolutional layers, with 32 and 64 filters, respectively. Batch normalization was added, which normalized each batch for consistent analysis. A second *MaxPool3D* layer was used to return the maximum values after all these operations.

A dropout layer with parameter 0.25 was implemented, following the convolutional and pooling layers. The inputs were then flattened to enable dense connections. A dense layer with dimensionality of 11^3 (length 1331) was added, along with *ReLU* activation, followed by a dropout layer with parameter 0.5. Another dense layer was added with dimensionality of 1024 and *ReLU* activation,

followed by another dropout layer with parameter 0.5. The last layer, a dense layer of dimensionality five, was added, since there are five classes. For this layer, the *softmax* activation function was used because the output is a probability distribution.

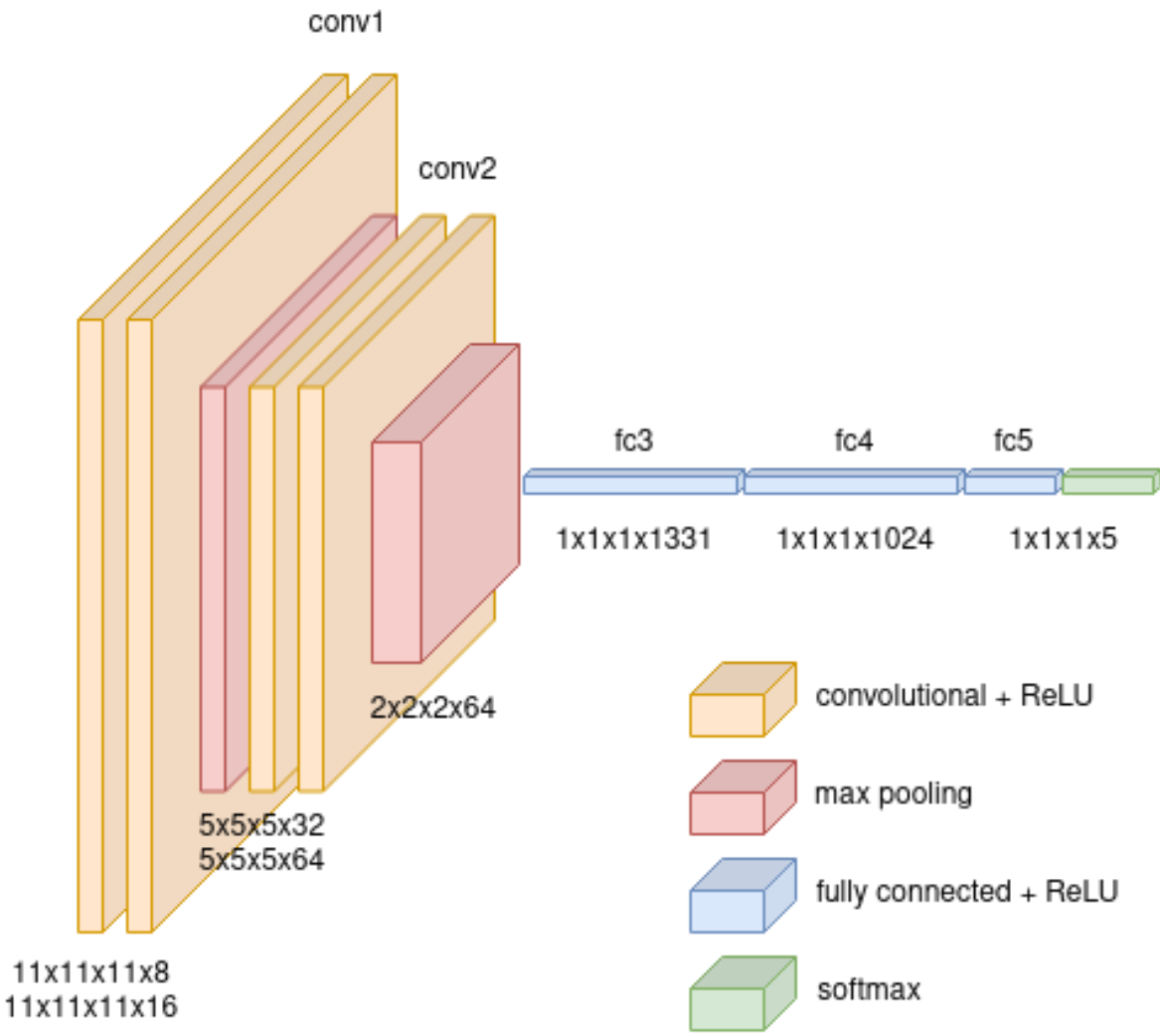

Figure 16: Visual diagram of the CNN model. There are two max pooling layers and the number of filters increases per layer for more detail. At the end, the layers are flattened and there is a softmax layer.

Right before training began, the categorical cross-entropy loss was assigned to this model. The accuracy metric was returned for tracking. Before tuning, the batch size was set to 86, which is small enough for this purpose, with the number of epochs also set to 30. There are over 1 million voxels, so the batch size should be around 100. For an average large dataset, the change in learning rate slows around 30-35 epochs. A validation set, which consisted of 15% of the training data, were also created

to tune the data parameters. The validation set size should be much smaller compared to the main training data, but should still have significant weight. Finally, the results of each epoch were displayed using the *TensorBoard* toolkit.

Analysis of the predicted results, as shown in Figure 17, indicates that many pixels that were not properly detected were located at the background edges. As a result, a two-stage model was developed (Figure 18). First, a copy of all the truth data was made, where the background value was still 0, and all the voxel values of the actual forest were set equal to 1. For the first stage, the same waveform arrays were used as the input, and the new thresholded voxels were set as the output. Since most waveform array intensities in background areas are close to zero, most of the background was properly detected and set exactly to zero.

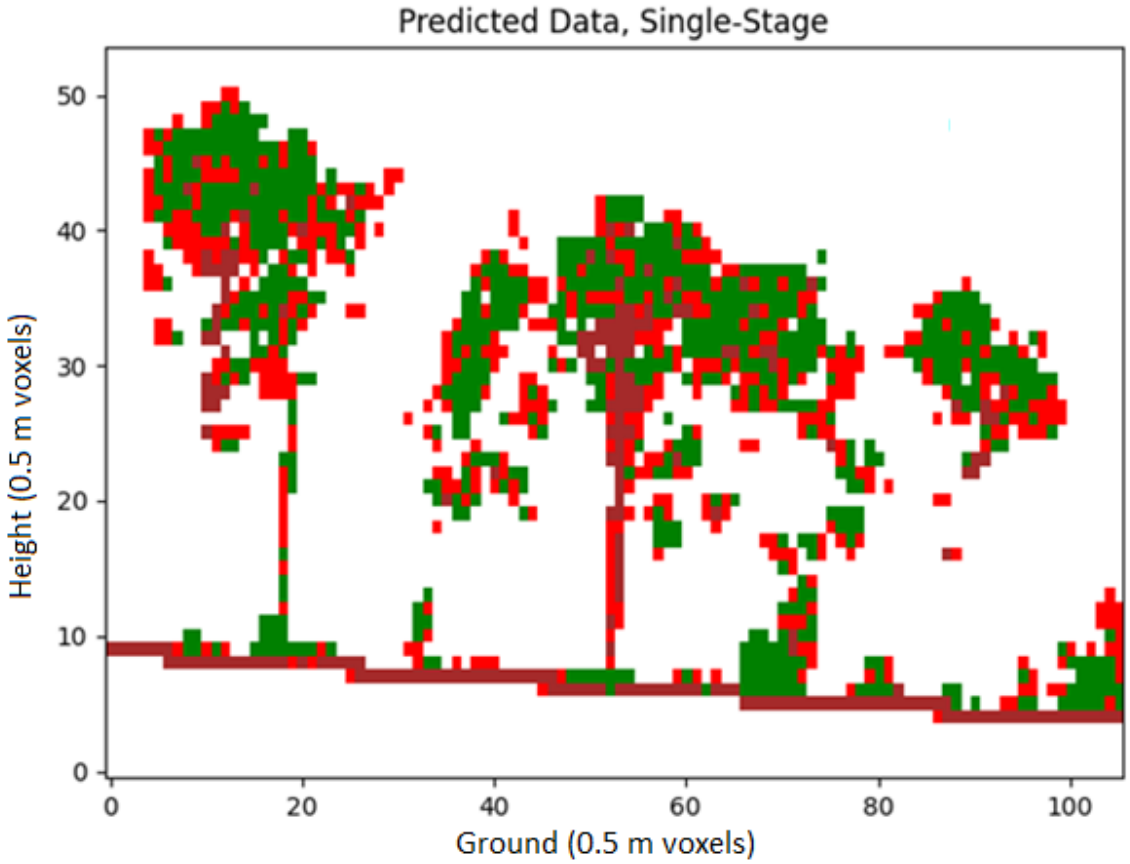

Figure 17: Slice indicating points which were incorrectly predicted. Incorrect predictions are colored red, and are close to the forest boundaries. Correct predictions are not colored red.

The initial input was multiplied by the first output to create a new output for the second stage. This removed the predicted background and retained the predicted forest voxel regions. The output was still the ground truth data. None of the voxels that were predicted as background were used in this stage, in order to simplify the algorithm. The new inputs and outputs were sent to the same CNN algorithm. Overall, the branches and tree trunks were far more likely to be correctly predicted.

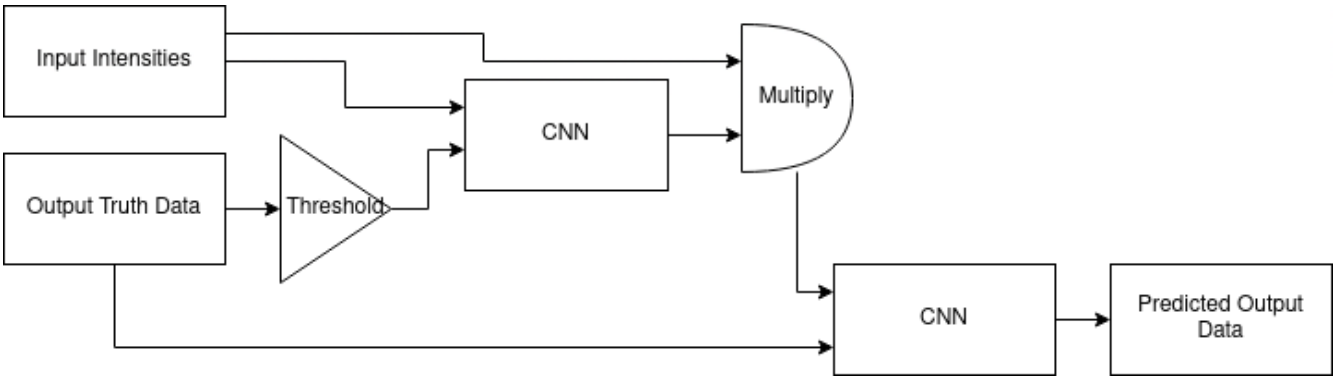

Figure 18: Workflow of the two-stage model. First, the output is thresholded, then the input's background is removed.

Originally, every flattened array was returned in order using a provided *for* loop, then fed into the *ScalarMappable* mixin to convert to (R, G, B) values. This structure is suitable for older servers, but is highly computationally inefficient for modern hardware. The dimension of the *ScalarMappable* output thus was expanded by one to store all the arrays at once, in order to remove this loop. After that, an even more computationally-efficient solution was discovered; instead of converting the array intensities to (R, G, B) values, a fifth axis was simply appended to the input array. This small change satisfied the 5D array requirement, required one-third of the original memory, and provided the same level of overall accuracy.

Finally, an additional channel was created where every positive value was thresholded to one. Segmenting the background from everything else was far easier, since the contrast for this layer is very high. As a result, the new single-stage model provided the same accuracy for the simulated VLP-16

waveform data as the previous two-stage model. Creating a more pronounced threshold helped the algorithm filter out intensities that were slightly above zero, thus removing background noise.

Z-VALUE INPUT CHANNELS – adding height above ground

A second channel was initially created, to provide greater accuracy, by appending local z-values to every input array. These local z-values range from zero to the maximum height of the input array. Z-values provide the algorithm with a measure of relative height, also called normalized height (above the ground). Object accuracy significantly increased due to the contrast of the straight bounds with the newly-added height information. Global z-values, which ranged from zero to the maximum height of the scene array, were not as effective due to the different heights and the unique properties of the tower object in the virtual scene.

However, global z-value segmentation bands, where the ground level is set to 0, proved to be a more effective solution, resulting in greater accuracy. It is worth noting that the use of the absolute z-values would create too many inputs, leading to low accuracy. The four levels were: below ground, shrubs and objects, vertical midsection of tree trunks, and tree canopies. After the classification was complete, voxels below ground were set to 0, ground level to 15 m above were set to 1, 15 - 35 m above were set to 2, and >35 m were set to 3.

A copy of the 3D array was created to create this layer, where ground was temporarily defined as the very bottom of the array. The segmentation bands of 1, 2, and 3 were defined. Additionally, a z-coordinate copy of this array was created. All the ground heights from the original array were returned in a duplicated cross-section as numbers. Missing ground values were set to *nan*, masked, and then interpolated using *scipy*. The cross-section was vertically duplicated to match the height of the original array, giving each voxel a vertical rotation value. The z-coordinates were vertically shifted by simply adding the z-coordinate array to the vertical rotation array. Finally, the new z-coordinates were set as

coordinate inputs into the segmentation band array, which were then set equal to the current segmentation values.

Since the 1-band is shifted upward, the values below are all 3, which should be set to 0. A duplicate matrix thus is created, where all values that are not 3 are set to zero. The *scipy.ndimage.label* function is used to segment the sections of value 3 into two sections. The bottom segment is equal to 1, so all locations in the original segmentation array where this is true are set to 0. This results in four unique sections, as shown in Figure 19.

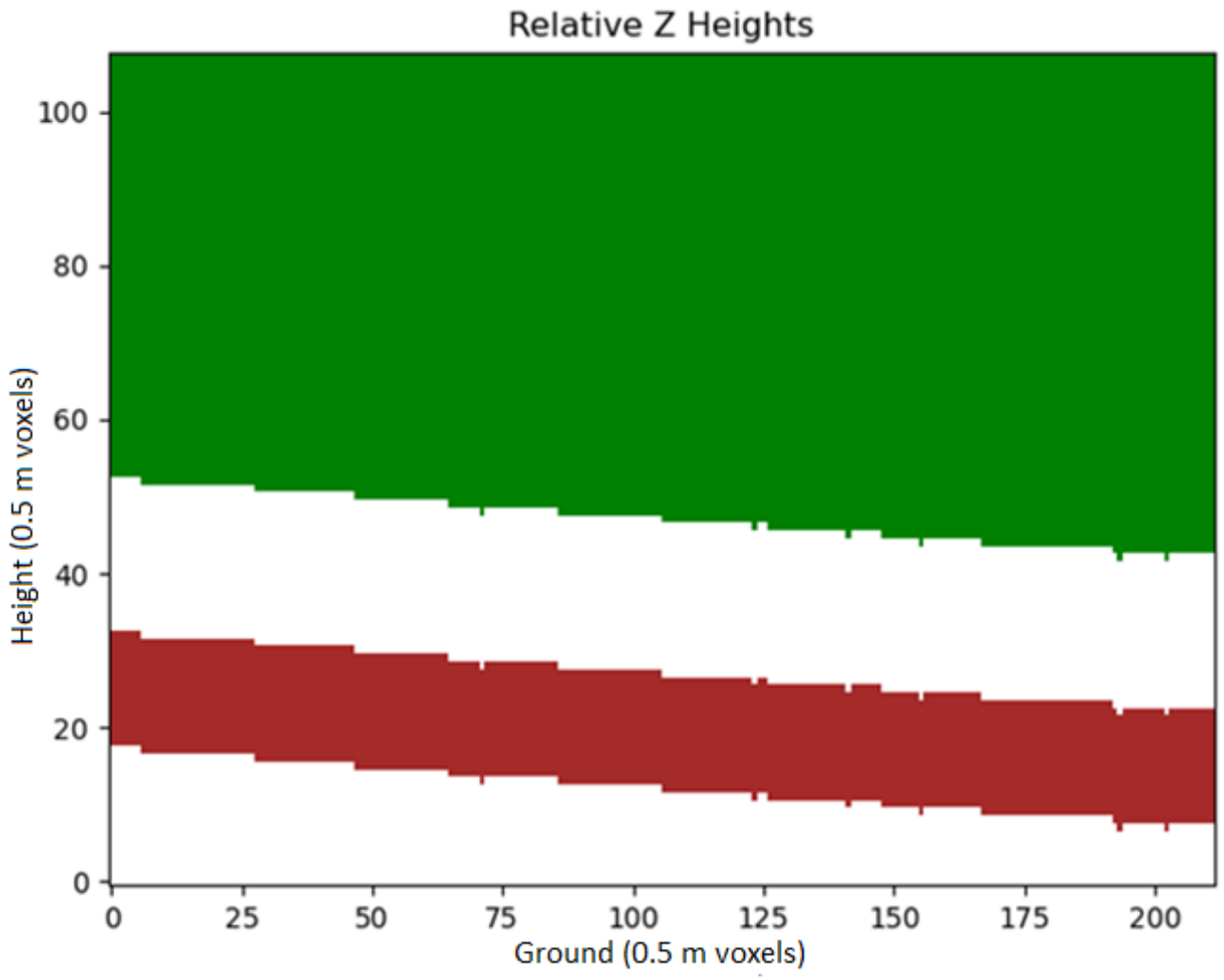

Figure 19: A plot of the relative z-heights, shifted to follow the ground's contour. Brown is 1, green is 3, and the white regions are 0 and 2.

ON THE USE OF THE KERAS TUNER

The Keras Tuner is a library that is part of the Keras API, which picks the best set of hyperparameters for a TensorFlow program [60]. Hyperparameters are numbers in the algorithm which determine the structure and flow of the components. There are two main types of hyperparameters: *Model hyperparameters*, which influence model selection such as dimensionality, and *Algorithm hyperparameters*, which influence speed and quality such as the learning rate [61].

The CNN model must be included in a *hypermodel*, the model set up for hypertuning. For this case, a model builder function is used, meaning that the CNN model is transferred to a separate function. Additionally, *model.fit* is not used, since this command is for training the model. There are several hyperparameter methods, which determine how the hyperparameters are selected. The methods which are used are *Int* for the second dense layer and *Choice* for the algorithm's learning rate. *Int* is an efficient method for creating an integer range, which requires as inputs a lower bound, an upper bound, and a step size, which separates the inputs. *Choice* simply picks the best value from a preset list [62].

More specifically, the *Hyperband tuner* is used for this algorithm, which uses a tournament bracket-style method to quickly converge on a high-performing model [63]. More detail about the algorithm is given in Figure 20. The objective is validation accuracy, and the number of maximum epochs is set to 10. There are fifteen choices for the second dense layer, and three choices for the learning rate. A small number of epochs would lead to good parameter results, which would be difficult to improve on.

Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

input : R, η (default $\eta = 3$)
initialization : $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$

for $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
  $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $r = R\eta^{-s}$
  // begin SUCCESSIVEHALVING with (n, r) inner loop
  $T =$ `get_hyperparameter_configuration`(n)
  for $i \in \{0, \ldots, s\}$ **do**
    $n_i = \lfloor n\eta^{-i} \rfloor$
    $r_i = r\eta^i$
    $L = \{$`run_then_return_val_loss`$(t, r_i) : t \in T\}$
    $T =$ `top_k`$(T, L, \lfloor n_i/\eta \rfloor)$
  end
end
return *Configuration with the smallest intermediate loss seen so far.*

Figure 20: Outline of the hyperband algorithm from the paper on Hyperband. There are two for-loops, which determine the effectiveness of the models in each bracket. The top half of the models move to the next bracket until tuning is complete.

The algorithm initially creates many models based on the dense layer and learning rate methods. These models are trained for a few epochs, and the top-performing half is then carried to the next round. Hyperband determines the number of models to train in each bracket using *Equation* 5:

$$1 + log_{factor}(max\_epochs) \qquad (Equation\ 5)$$

The possible parameters for the second dense layer were a minimum unit number of 64, a maximum unit number of 1024, and a step size of 64. For the learning rate, the possible parameters were 0.01, 0.001, and 0.0001. The program ended up selecting a unit number of 960 for the second dense layer, and a learning rate of 0.0001 for the optimizer. A very small learning rate made sense, given the large number of inputs and outputs, as a larger learning rate could easily overcompensate, making accurate results harder to obtain. Overall, accuracy significantly improved for all datasets, especially for the bark voxels.

Results and Discussion

The overall results were strongly comparable to the three seminal papers. *Huang and You* reported 78% accuracy for trees, but all object categories had 84% or higher accuracy. This is comparable to the ground truth models in this study, where leaf and bark precision was around 75%, but ground and object precision was around 99% [30]. *Kukenbrink et al.* demonstrated that smaller voxel sizes resulted in higher accuracy for the occluded and unobserved canopy. This matches with the models in this study, where smaller voxels generally resulted in higher accuracies. However, the input array size also needs to be larger [24]. Finally, *Hancock et al.* demonstrated methods to voxelize forests using ALS and TLS data. The false positives were less than 15%, and the negatives were less than 1% [31]. This is comparable to what was reconstructed with the d5lidar voxelizer [44].

0.25 m GROUND TRUTH MODELS (maximum category areas)

All models used ground truth maximum surface area values as the inputs. The training length was 50 epochs, with a batch size of 64. The training/testing ratio of voxels was 80/20. All models were trained using Plot B, Dataset 3 voxels. Global z-values were used as an input parameter to better classify different parts of the forest, improving accuracy. Different input voxel sizes and plot sections were used to compare different accuracies.

Full-plot 7x7x7 global z-values

According to Table 3, precision was very strong overall, with weaknesses in the leaf and bark elements. This is because leaf and bark elements tend to overlap, particularly in the canopy. Bark recall was quite low, because the number of leaf elements greatly outnumber the bark elements. This is despite that class weighting was used to improve accuracy. Because of the relatively small voxel size, object recall was only 60%. As shown in Figure 21, objects were slightly misshapen around the edges.

Table 3: Confusion matrix of the 0.25 m classifier model on ground truth full plot, input sizes 7x7x7. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Overall Recall	79%		Total Voxels	1218071		
Overall Precision	89%					
	Background	Leaf	Bark	Ground	Objects	Precision
Background	1.00	0.00	0.00	0.00	0.00	99%
Leaf	0.00	0.74	0.25	0.00	0.00	74%
Bark	0.00	0.26	0.74	0.00	0.00	74%
Ground	0.00	0.00	0.00	1.00	0.00	99%
Objects	0.00	0.01	0.00	0.01	0.99	99%
Recall	99%	85%	56%	94%	60%	

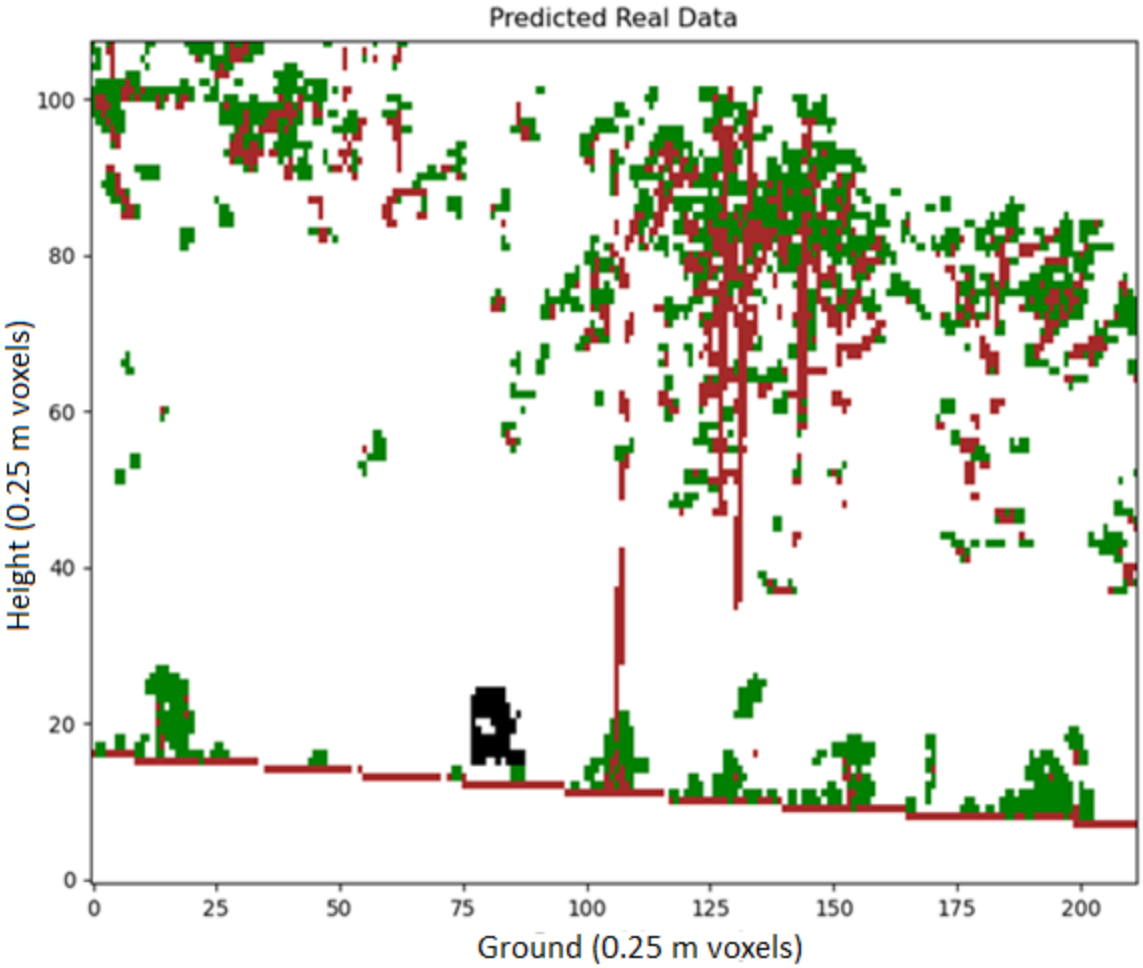

Figure 21: Predicted values of whole 0.25 m truth model with global z-value 7x7x7 inputs. Because of the relatively small input size, there is difficulty in classifying object edges, resulting in misshapen objects.

Half-plot 9x9x9 global z-values

As seen in Table 4, overall recall and precision were the best from all three 0.25 m voxel models. Leaf precision did decrease somewhat due to fewer voxels, but bark precision significantly increased. For good leaf accuracy values, large inputs are not needed, because of the smaller size of leaves. However, bark sections take up more room (volume), and would need larger inputs. The greatest improvement was in object recall, because larger voxels can capture the more rigid edges. In Figure 22, the objects were more detailed.

Table 4: Confusion matrix of the global z-value 0.25 m classifier model on ground truth half plot, input sizes 9x9x9. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Overall Recall	83%		Total Voxels	604455		
Overall Precision	89%					
	Background	Leaf	Bark	Ground	Objects	Precision
Background	1.00	0.00	0.00	0.00	0.00	99%
Leaf	0.00	0.70	0.29	0.00	0.00	70%
Bark	0.00	0.22	0.78	0.00	0.00	78%
Ground	0.00	0.00	0.00	1.00	0.00	99%
Objects	0.00	0.00	0.00	0.01	0.99	99%
Recall	99%	86%	55%	94%	78%	

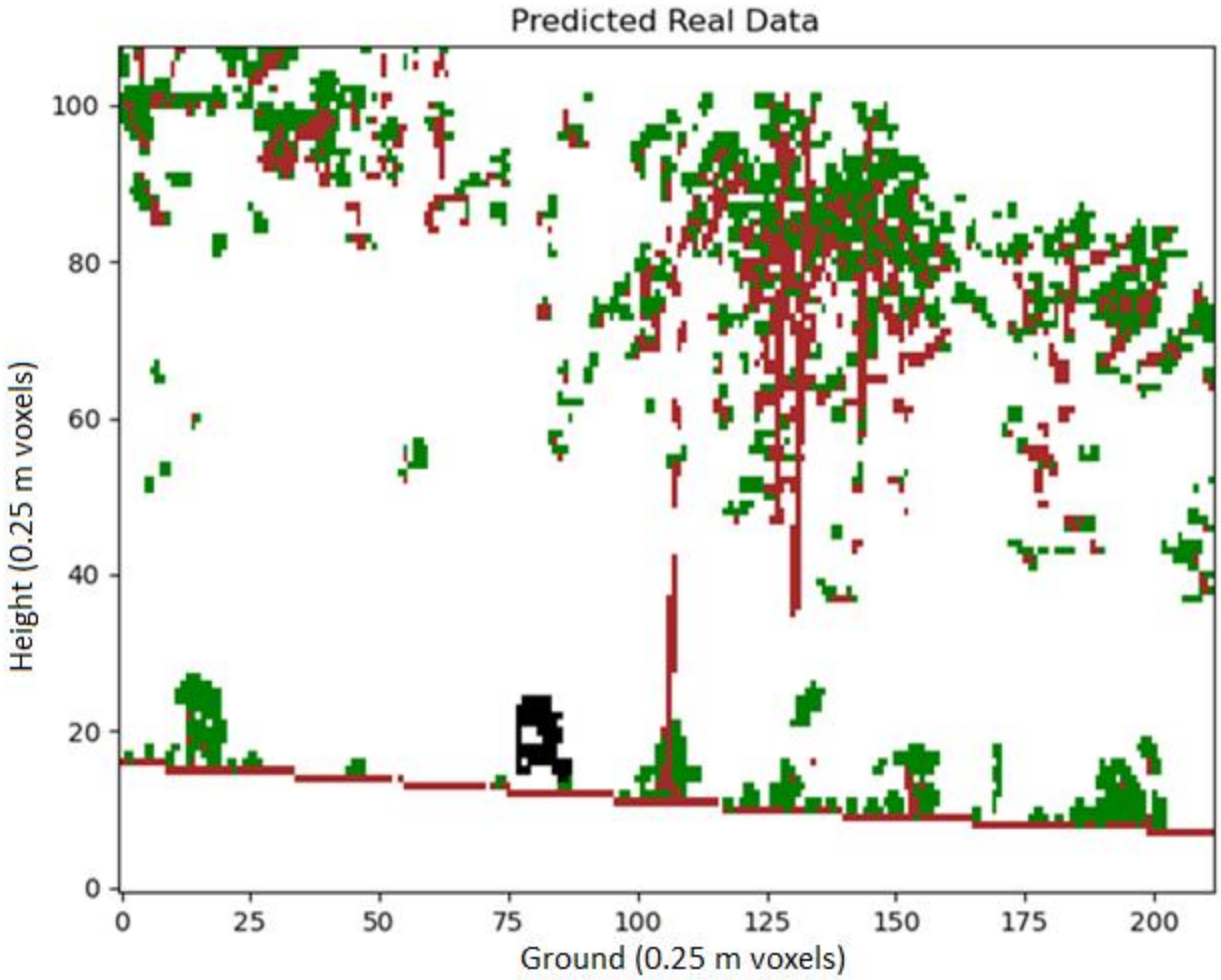

Figure 22: Predicted values of half 0.25 m truth model with global z-value 9x9x9 inputs. The input size is larger, so the object has significantly more pronounced edges.

Quarter-plot 11x11x11 global z-values

Overall recall is significantly less than the half-plot 9x9x9 values. One major reason is that in Table 5, the background precision is only 97%. This figure is due to the relatively small number of voxels, creating choppier edges, as seen in Figure 23. Leaf precision is slightly higher, and bark precision is almost 80%. In particular, the recall values of leaf and bark voxels are significantly less accurate, which was attributed to the slight decline in background precision.

Table 5: Confusion matrix of the global z-value 0.25 m classifier model on ground truth quarter plot, input sizes 11x11x11. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Overall Recall	78%		Total Voxels	302228		
Overall Precision	89%					
	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.97	0.01	0.01	0.00	0.00	97%
Leaf	0.01	0.72	0.27	0.00	0.00	72%
Bark	0.00	0.20	0.79	0.00	0.00	79%
Ground	0.00	0.00	0.00	0.99	0.00	99%
Objects	0.00	0.00	0.00	0.01	0.99	99%
Recall	99%	75%	49%	91%	75%	

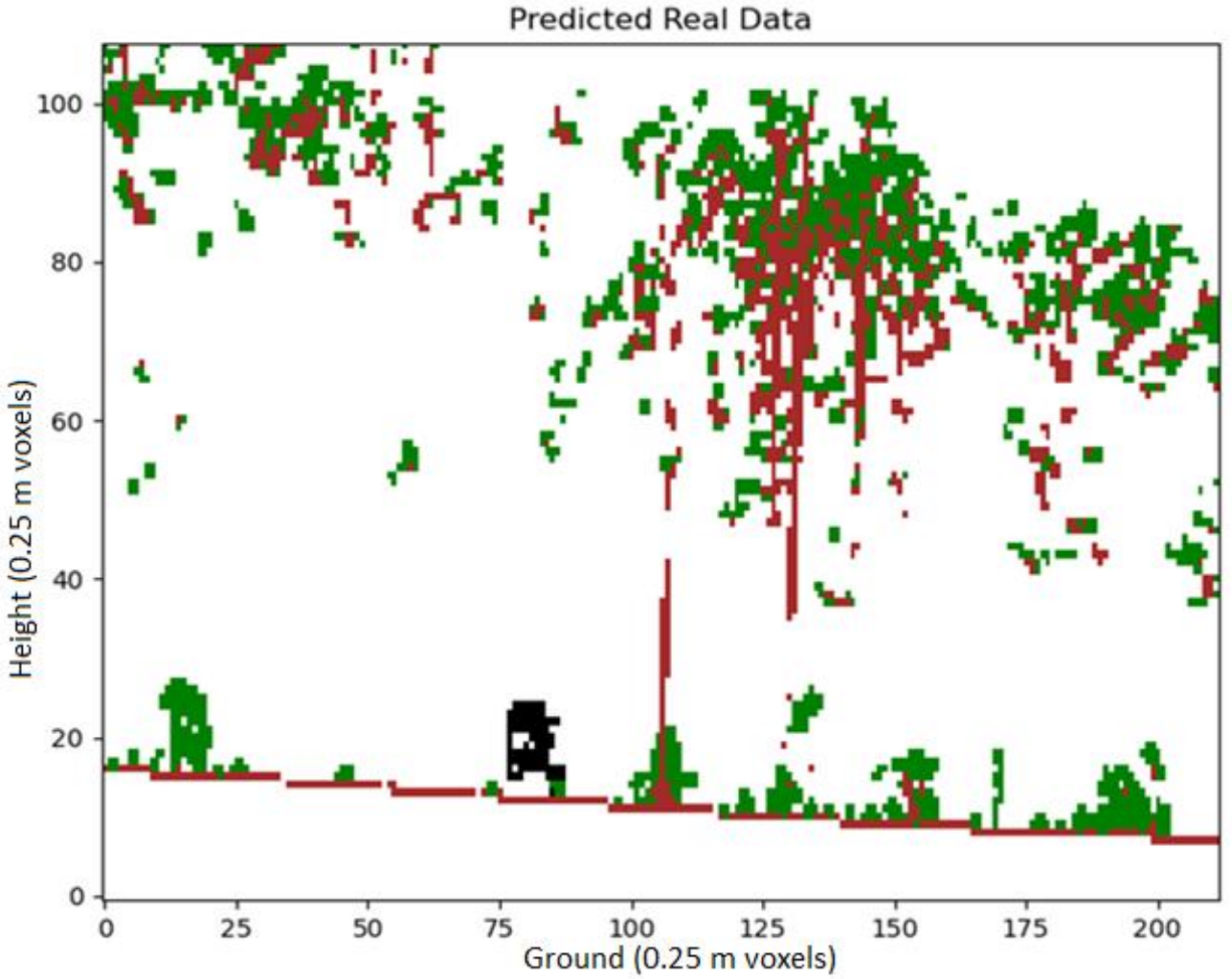

Figure 23: Predicted values of quarter 0.25 m truth model with global z-value 11x11x11 inputs. Because of the significantly low number of voxels, all the voxel edges are choppier compared to the previous two models.

VLP-16 MODELS (simulated waveform lidar data)

Classifier (SVC, only one unit per output)

The background voxels performed the best because according to Table 6, the predictions had 67% precision and 75% recall. Most of the voxels are background voxels, which are located at very large, specific sections. Leaf voxels were somewhat accurate, with 25% precision and 53% recall, as shown in Table 6. Because background voxels greatly outnumbered the leaf voxels, many leaf voxels were falsely predicted as background voxels, and not the other way around. Finally, ground voxels were more accurately predicted than leaf voxels, since the ground voxels are in a very specific region.

Table 6: Confusion matrix of the simulated VLP-16 classifier model with background voxel counts, 9x9x9 input size, with 0.5 m voxel size. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Overall Recall	75%		Total Voxels	293454		
Overall Precision	67%					
	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.97	0.03	0.00	0.00	0.00	97%
Leaf	0.73	0.25	0.00	0.01	0.00	25%
Bark	0.68	0.32	0.00	0.00	0.00	0%
Ground	0.38	0.02	0.00	0.60	0.00	60%
Objects	0.91	0.04	0.00	0.05	0.00	0%
Recall	77%	53%	0%	71%	0%	

Table 7: Confusion matrix of the simulated VLP-16 classifier model without background voxel counts, 9x9x9 input size, with 0.5 m voxel size. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

	Leaf	Bark	Ground	Objects
Leaf	0.95	0.00	0.05	0.00
Bark	0.99	0.00	0.01	0.00
Ground	0.04	0.00	0.96	0.00
Objects	0.43	0.00	0.57	0.00

However, only a small handful of bark and object voxels were correctly predicted. As seen in Figure 8, most bark voxels significantly overlap with the leaf voxels, making them hard to predict. This is because in a classifier, only one unit produces the activation output. There were also only 232

object voxels. The classifier is too simple in nature to be able to work with the rigid shapes of objects. The CNN methods were far more complex, and therefore performed much better.

The optimal objective value of the dual SVM problem was found to be -414.79, which is relatively close to zero, when compared to earlier iterations. This value measures the overall accuracy of the classifier [49]. The bias term is equal to -0.72, which means that the best classification predictions differ somewhat from most predictions. There were 429 support vectors, and 418 of them were bounded, meaning that the data were quite difficult to separate. Overall, there were 143,511 support vectors needed for classifying the voxels in the best possible way, which is a significant number, given the wide variety of data points.

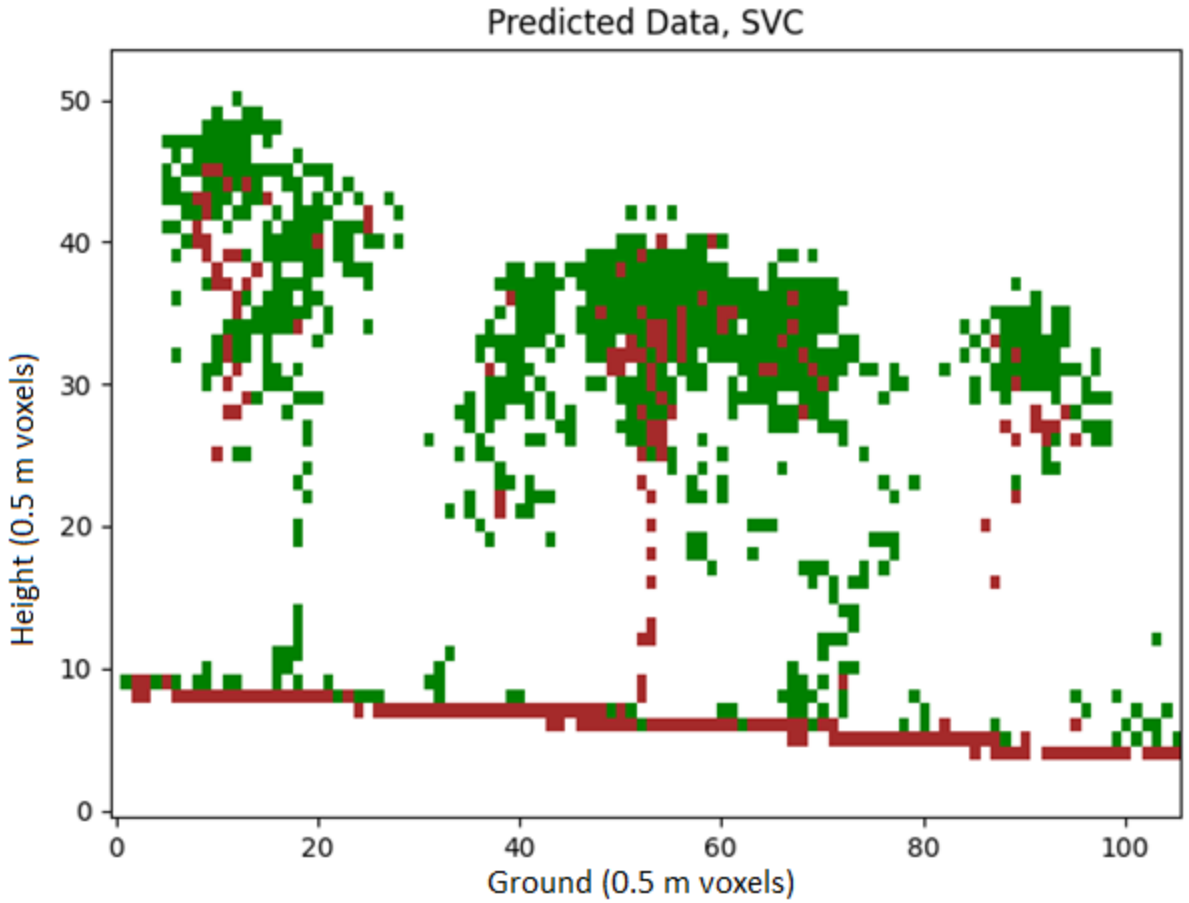

Figure 24: Predicted output of the classifier algorithm. Leaf and ground voxels performed much better than other types of voxel fills.

Single-stage (intensity-only input, no additional segmentation)

For the VLP-16 single-stage model, the 74% recall result was slightly better than the 66% precision result in Table 8. This is because the background takes up most space, according to Figure 25. Most of the background voxels were correctly predicted with 94% precision and 85% recall, but other voxel classes also were predicted as background. Leaf voxels exhibited average 53% precision and 60% recall rates, because they overlap with the other elements. Bark voxels showed a low 10% precision rate. Much of the bark overlaps with the leaves, and there are far more leaf voxels than bark voxels.

Table 8: Confusion matrix of the simulated VLP-16 single-stage model with background voxel counts, 9x9x9 input size, with 0.5 m voxel size. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Overall Recall	74%		Total Voxels	296493		
Overall Precision	66%					
Class	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.94	0.05	0.00	0.00	0.00	94%
Leaf	0.44	0.53	0.01	0.02	0.00	53%
Bark	0.40	0.50	0.10	0.00	0.00	10%
Ground	0.03	0.01	0.00	0.96	0.00	96%
Objects	0.10	0.08	0.00	0.06	0.76	76%
Recall	85%	60%	61%	83%	82%	

Table 9: Confusion matrix of the simulated VLP-16 single-stage model without background voxel counts, 9x9x9 input size, with 0.5 m voxel size. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Class	Leaf	Bark	Ground	Objects
Leaf	0.95	0.03	0.03	0.00
Bark	0.83	0.17	0.00	0.00
Ground	0.01	0.00	0.99	0.00
Objects	0.09	0.00	0.06	0.84

The ground voxels were predicted very accurately, because all these voxels are located at a very specific height. Additionally, the surrounding shape of these voxels is very specific, which looks almost like a flat plane. Although there were less than 500 object voxels, the rates of both precision and recall were relatively high. One possible explanation might be that, because these are manmade objects, they have a defined rigid shape, as opposed to the far greater randomness of the forest leaves and bark.

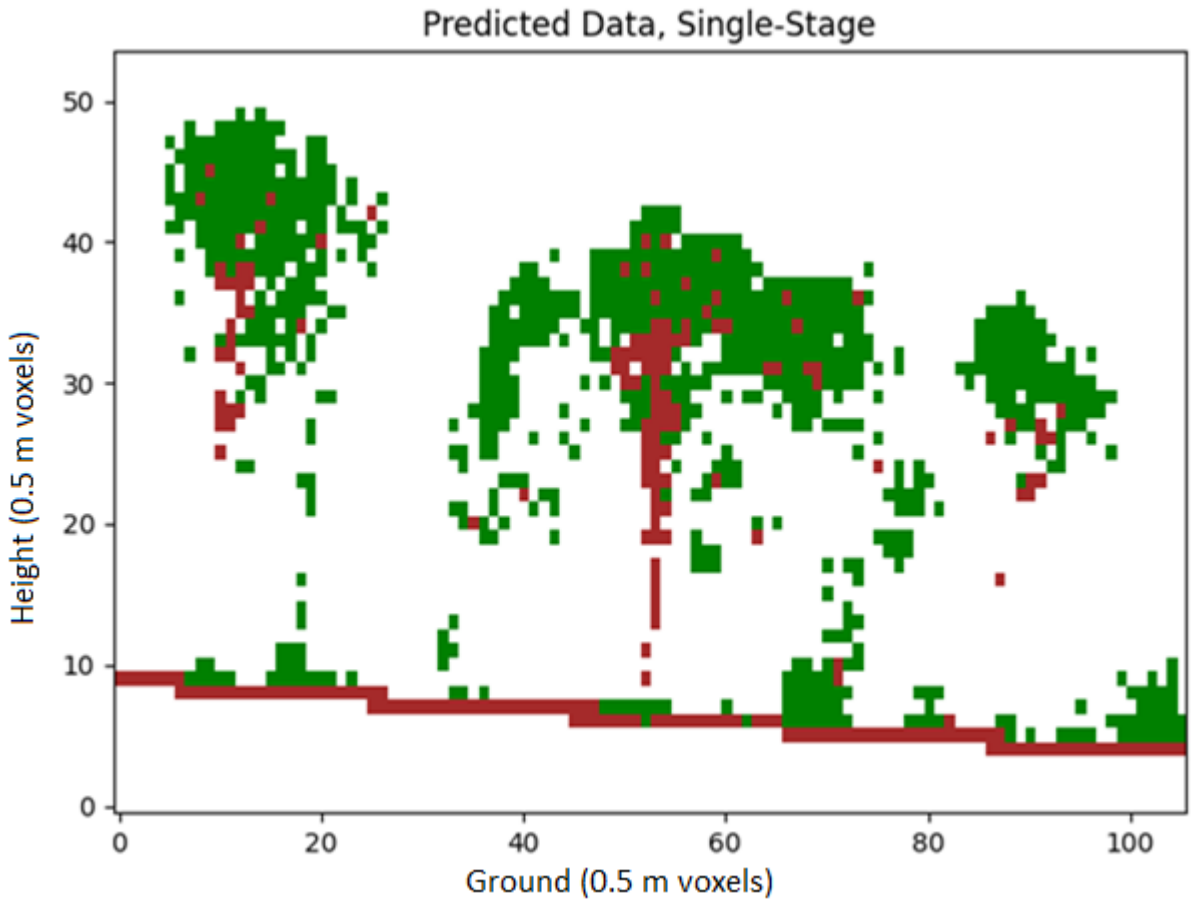

Figure 25: Predicted output of the VLP-16 single-stage model. Leaf predictions were much better when compared to the classifier algorithm, and objects (not pictured here) also performed much better.

Two-stage (pre-segmentation of background, then all other categories)

The 92% background precision was strong for the first stage of the two-stage model (Table 10). This was because the background comprises most of the voxels, especially at the edge regions of the array. The 70% forest precision value was decent, because forest leaf structure is mostly random, and the boundary between forest and background is significantly blurry in the input array. Regardless of these issues, however, bark detection in the second stage of the model was much better than the single-stage model.

Table 10: Confusion matrix of the first stage of the two-stage model. Values are normalized so that the sum of every row is equal to 1.

Class	Background	Forest
Background	0.92	0.08
Forest	0.3	0.7

For the second stage of the two-stage model (Table 11), the voxel types were very similar, resulting in approximately 66% precision and recall. Most voxels in this stage were leaf and bark, which are more random in nature. Since most of the background was eliminated, there were relatively few background voxels which were randomly scattered. As a result, the rates were less than 50% for precision and recall. The 86% precision and 72% recall for the leaf voxels were significantly better this time. This is likely because most leaf voxels are grouped (clumped) in near-ground surface bushes or as part of the tree canopies. The locations were more specific for this CNN model, which made detection easier.

Table 11: Confusion matrix of the second stage of two-stage model with background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1. Simulated VLP-16 data is used, 9x9x9 input size, with 0.5 m voxel size.

Overall Recall	67%		Total Voxels	82633		
Overall Precision	66%					
Class	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.11	0.74	0.15	0.01	0.00	11%
Leaf	0.02	0.86	0.10	0.01	0.00	86%
Bark	0.02	0.56	0.42	0.00	0.00	42%
Ground	0.00	0.01	0.00	0.98	0.00	98%
Objects	0.01	0.02	0.00	0.02	0.95	95%
Recall	41%	72%	54%	87%	80%	

Table 12: Confusion matrix of the second stage of two-stage model without background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1. Simulated VLP-16 data is used, 9x9x9 input size, with 0.5 m voxel size.

Class	Leaf	Bark	Ground	Objects
Leaf	0.88	0.10	0.02	0.00
Bark	0.57	0.43	0.00	0.00
Ground	0.01	0.00	0.99	0.00
Objects	0.02	0.00	0.02	0.95

Bark detection performance had 42% precision and 54% recall rates, because there were far fewer bark voxels, which were also mostly surrounded by leaves. Bark predictions are far more visible in Figure 26. Again, the ground exhibited very high 98% precision and 87% recall rates, because the ground is located at a specific location and has a relatively specific shape. Finally, the objects had 95% precision, because the overall object location is very specific and they have a defined rigid shape.

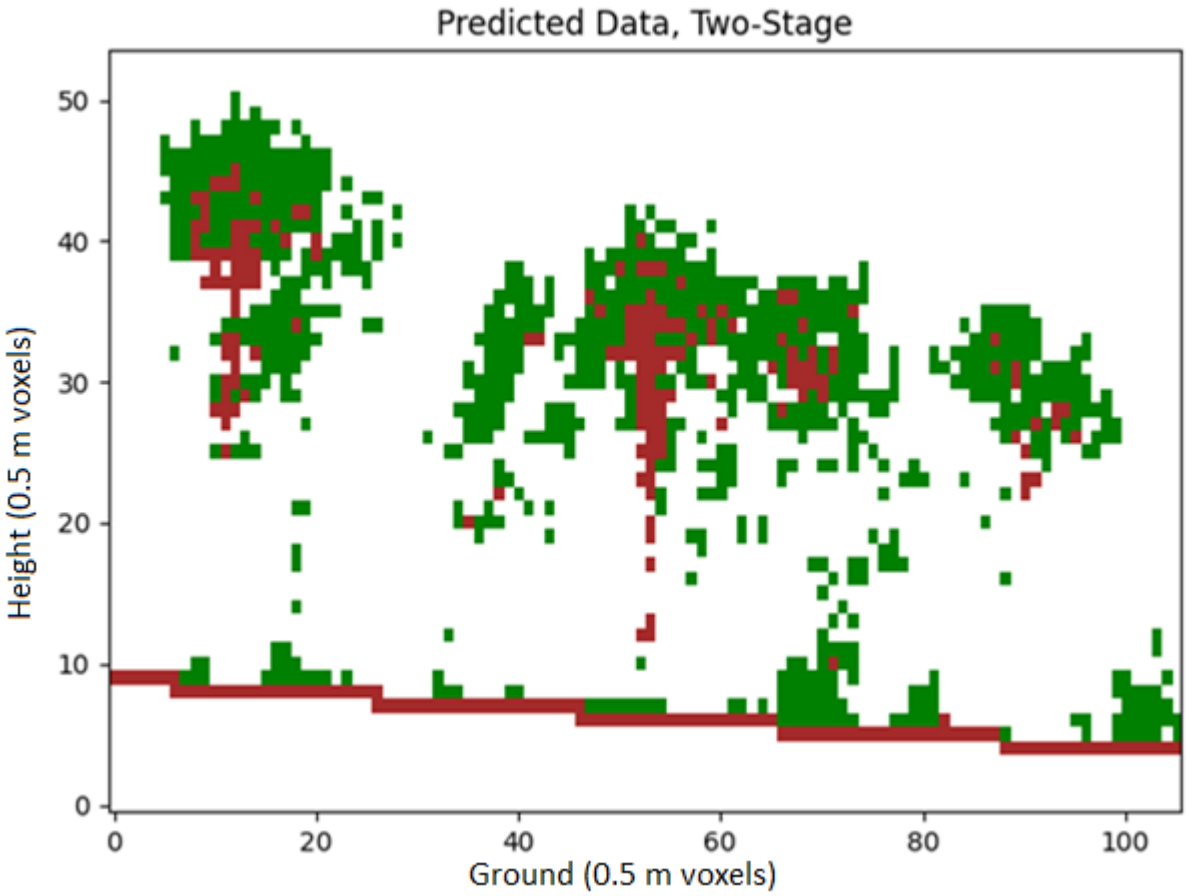

Figure 26: Predicted output of the two-stage model. Leaves and bark were classified much more accurately when compared to the single-stage model. Objects (not pictured here) were classified extremely well.

Three-channel (intensity, background threshold, and local z-values)

The 76% recall result was very close to the 77% precision result for the VLP-16 three-channel model in Table 13. This is because the background occupies most of the space, as seen in Figure 27. Most of the background voxels were correctly predicted, with 93% precision and 91% recall rates. Leaf voxels exhibited good (71%) precision and recall (64%) rates, because the edges are easily detectable, though they overlap with the other elements. The thresholded channel likely helps filter out background noise. Again, the bark voxels had a low 28% precision rate. Bark is the element that is most "buried" in the scene, so VLP-16 sensors had the greatest difficulty distinguishing bark voxels. The bark recall rate is surprisingly high at 62%, but that is most likely due to very few voxels being predicted as bark.

Table 13: Confusion matrix of the simulated VLP-16 three-channel model (local z-value and thresholded background channels) with background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1. A 9x9x9 input size was used, with 0.5 m voxel size.

Overall Recall	76%		Total Voxels	124836		
Overall Precision	77%					
Class	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.93	0.06	0.01	0.00	0.00	93%
Leaf	0.25	0.71	0.03	0.01	0.00	71%
Bark	0.20	0.51	0.28	0.00	0.00	28%
Ground	0.02	0.02	0.00	0.97	0.00	97%
Objects	0.01	0.00	0.00	0.02	0.96	96%
Recall	91%	64%	62%	85%	78%	

Table 14: Confusion matrix of the simulated VLP-16 three-channel model (local z-value and thresholded background channels) without background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1. A 9x9x9 input size was used, with 0.5 m voxel size.

Class	Leaf	Bark	Ground	Objects
Leaf	0.94	0.04	0.02	0.00
Bark	0.64	0.35	0.00	0.00
Ground	0.02	0.00	0.98	0.00
Objects	0.00	0.00	0.02	0.98

The ground voxels were predicted very accurately, since, as stated before, these voxels are located at a specific height and have high intensities. The shape is almost perfectly contoured to the z-value channels. Also, the surrounding shape of these voxels is very specific, which looks almost like a flat plane. Although there were fewer than 200 object voxels, the rates of both precision and recall were around 96% and 78%, respectively.

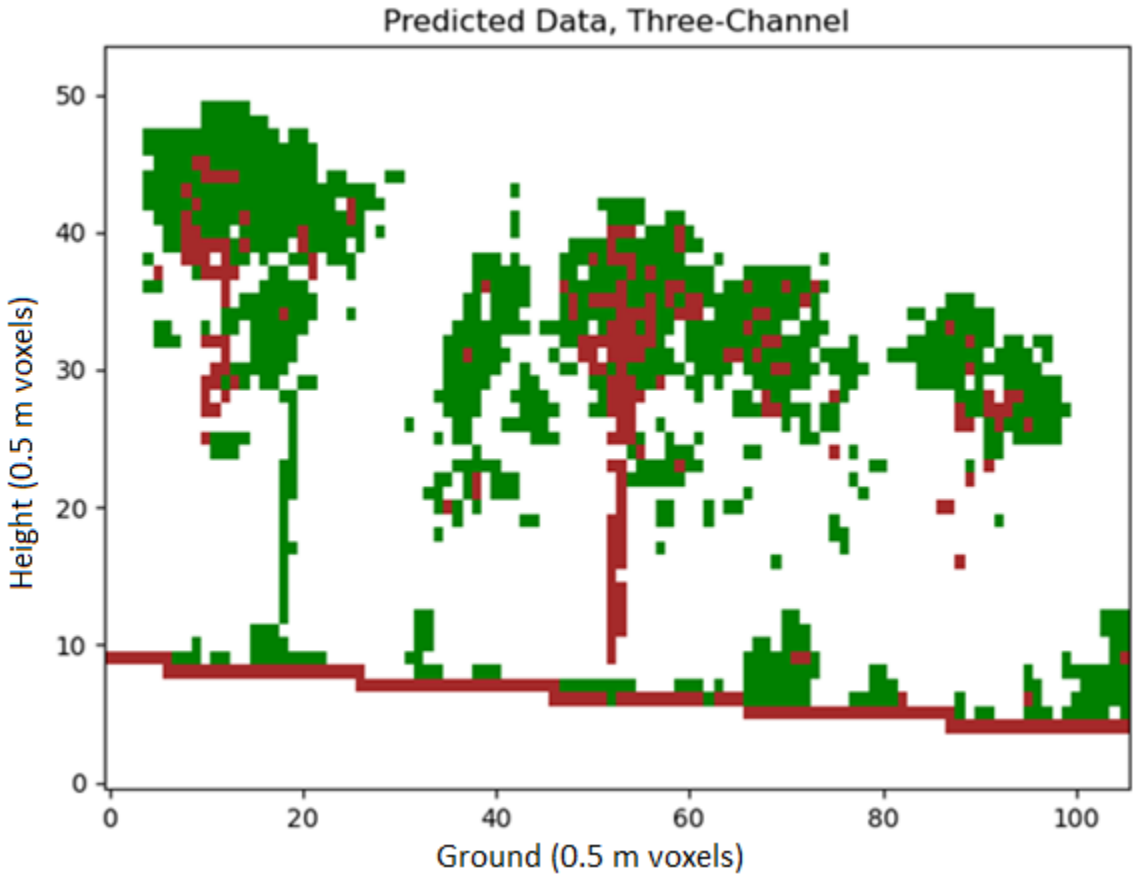

Figure 27: Predicted output of the VLP-16 three-channel model (local z-value and thresholded background channels). Leaf predictions were good because most of the forest elements are leaves. Both ground and object predictions were excellent because the input channels included the simulated VLP-16 intensities, local z-values, and thresholded background.

On use of the Keras Tuner

The 74% recall result was very close to the 73% precision result for the VLP-16 three-channel model in Table 15. This is because the background occupies most of the space according to Figure 28. Most of the background voxels were correctly predicted with 93% precision and 89% recall. Leaf voxels exhibited good (61%) precision and recall (66%) rates, because the edges are easily detectable, though they overlap with the other elements. Bark voxels had a much better 36% precision rate, since the tuner resulted in a far lower learning rate. This result makes sense, because there are many voxels overall, so a lower learning rate is best for tracking hidden voxels. The bark recall rate is surprisingly high at 66%, but that is most likely due to very few voxels being predicted as bark.

Table 15: Confusion matrix of the simulated VLP-16 Keras Tuner model with background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1. A 9x9x9 input size was used, with 0.5 m voxel size.

Overall Recall	74%		Total Voxels	312090		
Overall Precision	73%					
Class	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.93	0.05	0.01	0.00	0.00	93%
Leaf	0.32	0.61	0.06	0.01	0.00	61%
Bark	0.26	0.38	0.36	0.00	0.00	36%
Ground	0.01	0.02	0.00	0.97	0.00	97%
Objects	0.14	0.02	0.02	0.02	0.80	80%
Recall	89%	66%	54%	84%	76%	

Table 16: Confusion matrix of the simulated VLP-16 Keras Tuner model without background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1. A 9x9x9 input size was used, with 0.5 m voxel size.

Class	Leaf	Bark	Ground	Objects
Leaf	0.89	0.09	0.02	0.00
Bark	0.51	0.49	0.00	0.00
Ground	0.02	0.00	0.98	0.00
Objects	0.02	0.02	0.02	0.93

The predictions for the ground voxels again were good, for the same reasons mentioned before. The shape generally fits the layout and direction of the z-value channels. The ground voxels again almost look like a flat plane. Even though there were fewer than 200 object voxels, their rates of both precision and recall were around 80% and 76%, respectively.

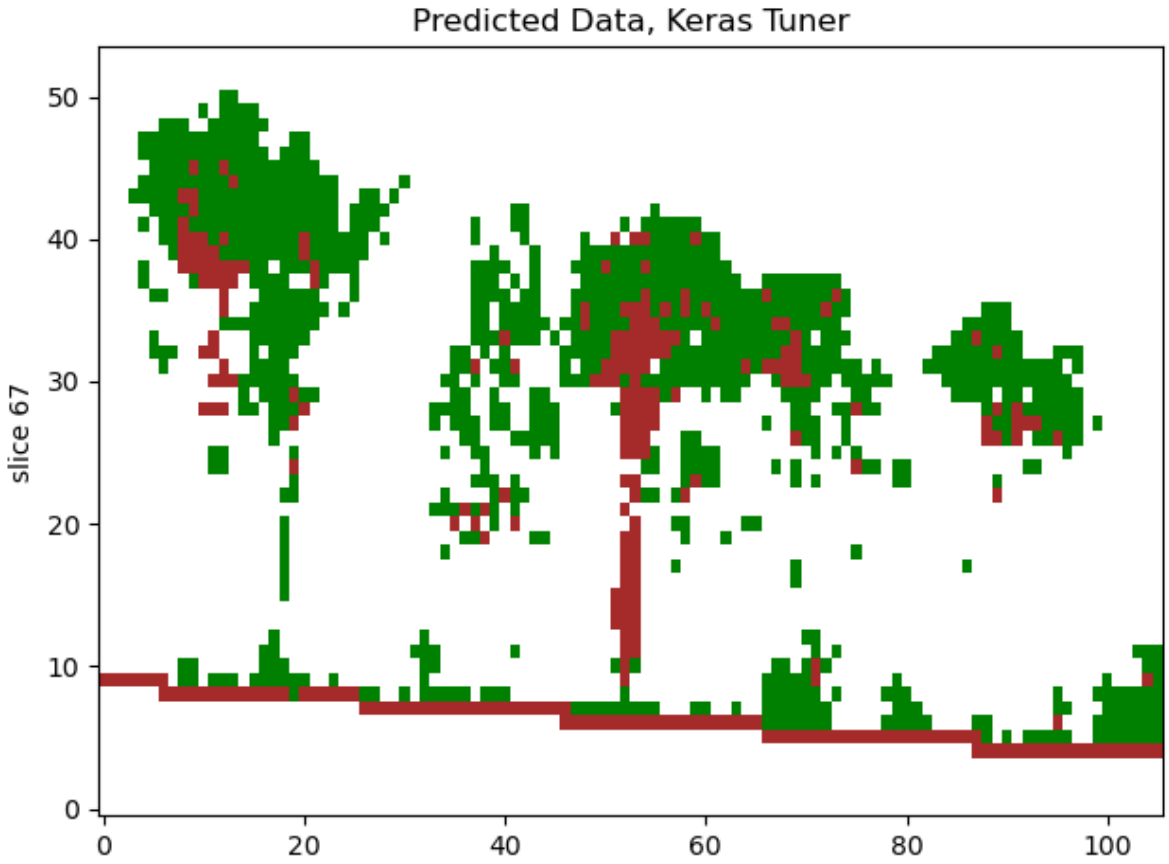

Figure 28: Predicted output of the VLP-16 three-channel model (local z-value and thresholded background channels). Leaf predictions were good because most of the forest elements are leaves. Both ground and object predictions were excellent because the input channels included the simulated VLP-16 waveform intensities, local z-values, and thresholded background.

NEON MODELS (simulated waveform lidar data)

50 m x 50 m (single-stage model, 0.5 m voxels)

For the NEON (50 m x 50 m) single-stage model, the 67% recall result was around the same as the 63% precision result in Table 17. This again is because the background occupies most of the space, as seen in Figure 29. Most of the background voxels were correctly predicted with 92% precision and 87% recall, but other voxel classes also were predicted as background. Leaf voxels exhibited average 64% precision and 62% recall rates, because they overlap with everything else. Again, the bark voxels had a low 11% precision rate. Bark is the element that is most buried (or occluded) in the scene, so NEON sensors have the greatest difficulty distinguishing bark voxels.

Table 17: Confusion matrix of the simulated NEON model (50 m x 50 m) with background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1. An 11x11x11 input size was used, with 0.5 m voxel size.

Overall Recall	67%		Total Voxels	110623		
Overall Precision	63%					
Class	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.92	0.07	0.01	0.00	0.00	92%
Leaf	0.31	0.65	0.03	0.01	0.00	64%
Bark	0.38	0.50	0.11	0.00	0.00	11%
Ground	0.05	0.03	0.00	0.91	0.00	91%
Objects	0.09	0.22	0.00	0.14	0.55	54%
Recall	87%	62%	38%	80%	67%	

Table 18: Confusion matrix of the simulated NEON model (50 m x 50 m) without background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1. An 11x11x11 input size was used, with 0.5 m voxel size.

Class	Leaf	Bark	Ground	Objects
Leaf	0.94	0.05	0.02	0.00
Bark	0.81	0.18	0.00	0.00
Ground	0.04	0.00	0.96	0.00
Objects	0.24	0.00	0.16	0.60

The ground voxels were predicted very accurately, because all these voxels are located at a specific height and have high intensities. Additionally, the surrounding shape of these voxels is specific, which looks almost like a flat plane. The rates of both precision and recall for object voxels again were decent even from a very high altitude, even given that there were fewer than 200 such voxels; this was attributed to the defined shape of these objects.

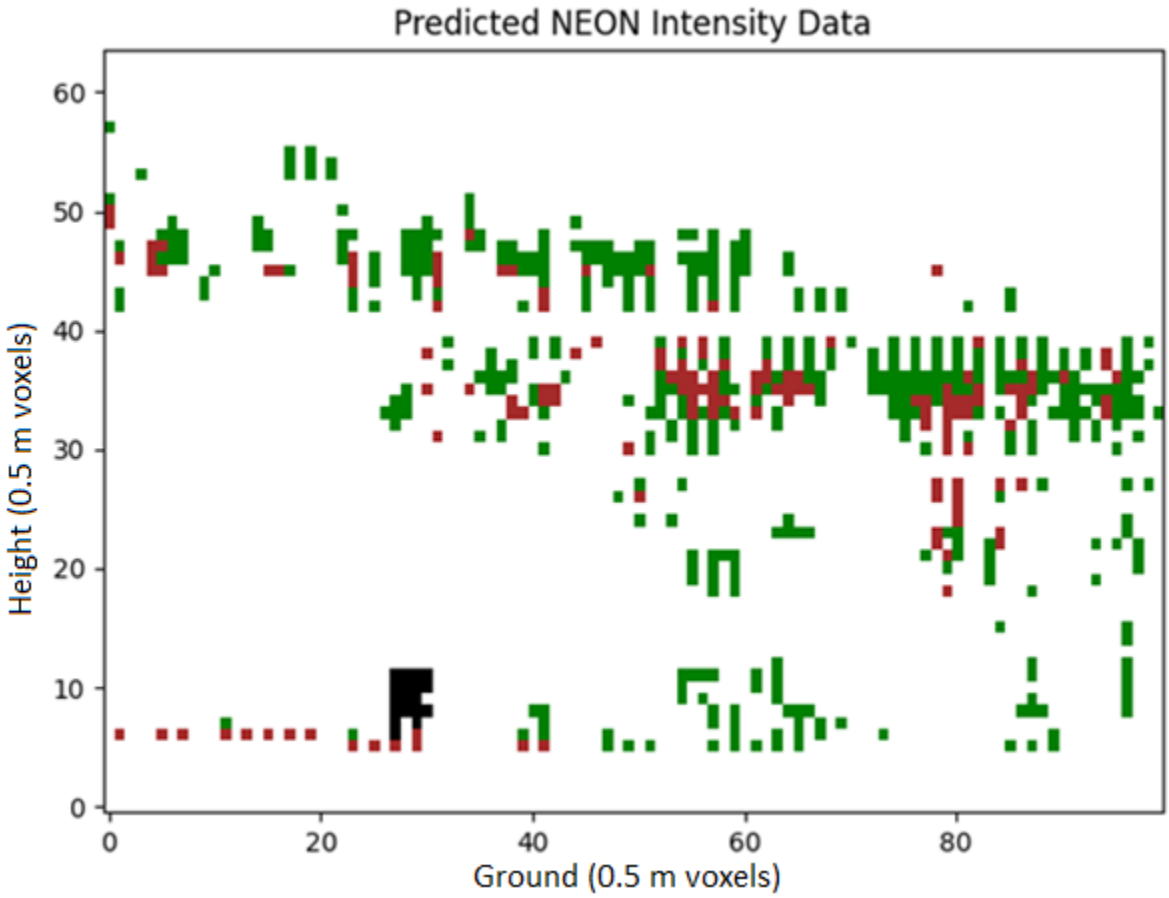

Figure 29: Predicted output of the NEON model (50 m x 50 m). Leaf predictions were decent because most of the forest elements are leaves. Object predictions were also decent because of the planned shapes. The ground performed the best because the density is the highest.

150 m x 150 m 7x7x7 (single-stage, 0.5 m voxel size)

For the NEON (150 m x 150 m, 7x7x7) single-stage model, the 68% recall result was relatively close to the 61% precision result, shown in Table 19. The reasons are similar to those mentioned before: i) the background occupies up the majority of space (Figure 30); most of the background voxels were correctly predicted with 94% precision and 88% recall, but other voxel classes also were predicted as background; ii) leaf voxels exhibited fairly average 69% precision and 64% recall rates, due to overlap with the other elements; iii) the bark voxels again had a low (1%) precision rate; bark is the element that is most hidden, so to speak, so NEON sensors have the greatest difficulty distinguishing bark voxels; however, the bark recall rate is surprisingly high, but that is most likely due to very few voxels being predicted as bark; iv) ground voxels were predicted accurately, because all of

these voxels are located at a specific height and have high intensities; also, the surrounding shape of these voxels is specific (a flat plane); and v) although there were fewer than 2500 object voxels, the rates of both precision and recall were, respectively, 50% and 60%, even from a high altitude; this was attributed to the defined shape of these objects.

Table 19: Confusion matrix of the simulated NEON model (150 m x 150 m, 7x7x7, 0.5 m voxel size) with background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Overall Recall	68%		Total Voxels	1241372		
Overall Precision	61%					
Class	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.94	0.05	0.00	0.00	0.00	94%
Leaf	0.30	0.69	0.00	0.01	0.00	69%
Bark	0.43	0.56	0.01	0.00	0.00	1%
Ground	0.04	0.04	0.00	0.92	0.00	92%
Objects	0.29	0.18	0.00	0.03	0.50	50%
Recall	88%	64%	49%	77%	60%	

Table 20: Confusion matrix of the simulated NEON model (150 m x 150 m, 7x7x7, 0.5 m voxel size) without background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Class	Leaf	Bark	Ground	Objects
Leaf	0.98	0.00	0.02	0.00
Bark	0.98	0.01	0.00	0.00
Ground	0.04	0.00	0.96	0.00
Objects	0.26	0.00	0.04	0.71

Overall, the results were like the 50 m x 50 m plot. When this plot was fed into the 3D CNN, three times the amount of graphics card memory was used due to the larger size. However, the input array dimension size was reduced from 11 to seven voxels, resulting in the volume being approximately 25% of the original volume. These two factors combined have led to only a relatively small difference in results.

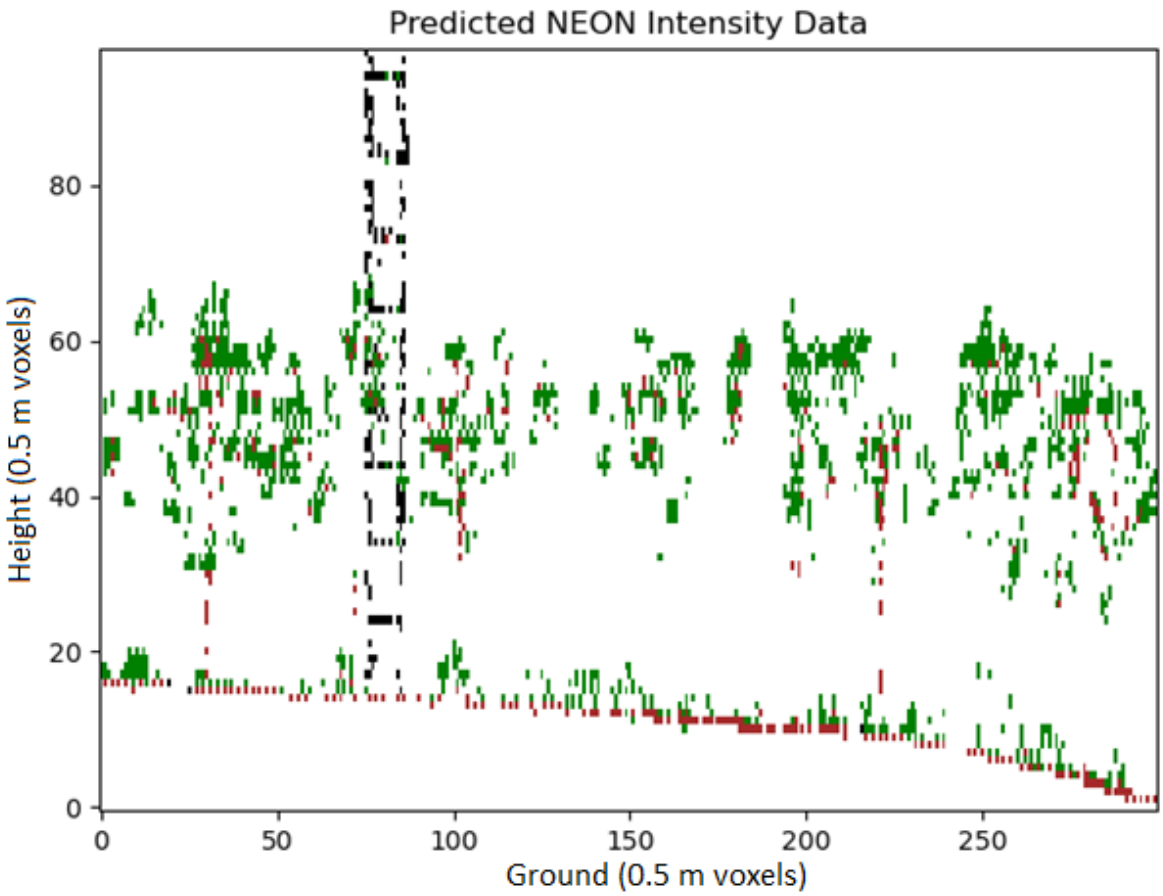

Figure 30: Predicted output of the NEON model (150 m x 150 m, 7x7x7). Leaf predictions were decent because most of the forest elements are leaves. Object predictions were also decent because of the defined shapes. The ground performed the best because the intensities were the highest.

150 m x 150 m 1 m (single-stage, other models have 0.5 m voxel size)

For the NEON (150 m x 150 m, 1 m voxels) single-stage model, the 72% recall result was very close to the 71% precision result in Table 21. The reasons again are like those mentioned before; see Figure 31 as an example of how much space is occupied by the background voxels. These voxels were correctly predicted with 90% precision and 87% recall. Leaf voxels exhibited good 79% precision and 69% recall rates, because the edges are easily detectable, though they overlap with the other elements. Again, the bark voxels had a low 19% precision rate, for the same reasons as before. The bark recall rate is surprisingly high at 52%, which was attributed the relatively low number of voxels being predicted as bark. Finally, ground voxel accuracy was attributed to their consistent location and profile,

while the fewer than 1000 object voxels exhibited rates of both precision and recall around 75% even, from a very high altitude.

Table 21: Confusion matrix of the simulated NEON model (150 m x 150 m, 7x7x7 inputs, 1 m voxels) with background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Overall Recall	72%		Total Voxels	309369		
Overall Precision	71%					
Class	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.90	0.09	0.01	0.00	0.00	90%
Leaf	0.15	0.79	0.04	0.02	0.00	79%
Bark	0.23	0.58	0.19	0.00	0.00	19%
Ground	0.02	0.06	0.00	0.92	0.00	92%
Objects	0.09	0.12	0.00	0.02	0.77	77%
Recall	87%	69%	52%	80%	72%	

Table 22: Confusion matrix of the simulated NEON model (150 m x 150 m, 7x7x7 inputs, 1 m voxels) without background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Class	Leaf	Bark	Ground	Objects
Leaf	0.94	0.04	0.02	0.00
Bark	0.75	0.25	0.00	0.00
Ground	0.06	0.00	0.94	0.00
Objects	0.13	0.00	0.03	0.85

Overall, the predicted results were much better than those for the 50 m x 50 m plot. When the input arrays were fed into the 3D CNN, nine times the amount of graphics card memory was used due to the larger size. Additionally, the increase in input voxel size made all the individual categories far

more relevant than they normally would have been. However, larger voxel size is indicative of significantly less detail.

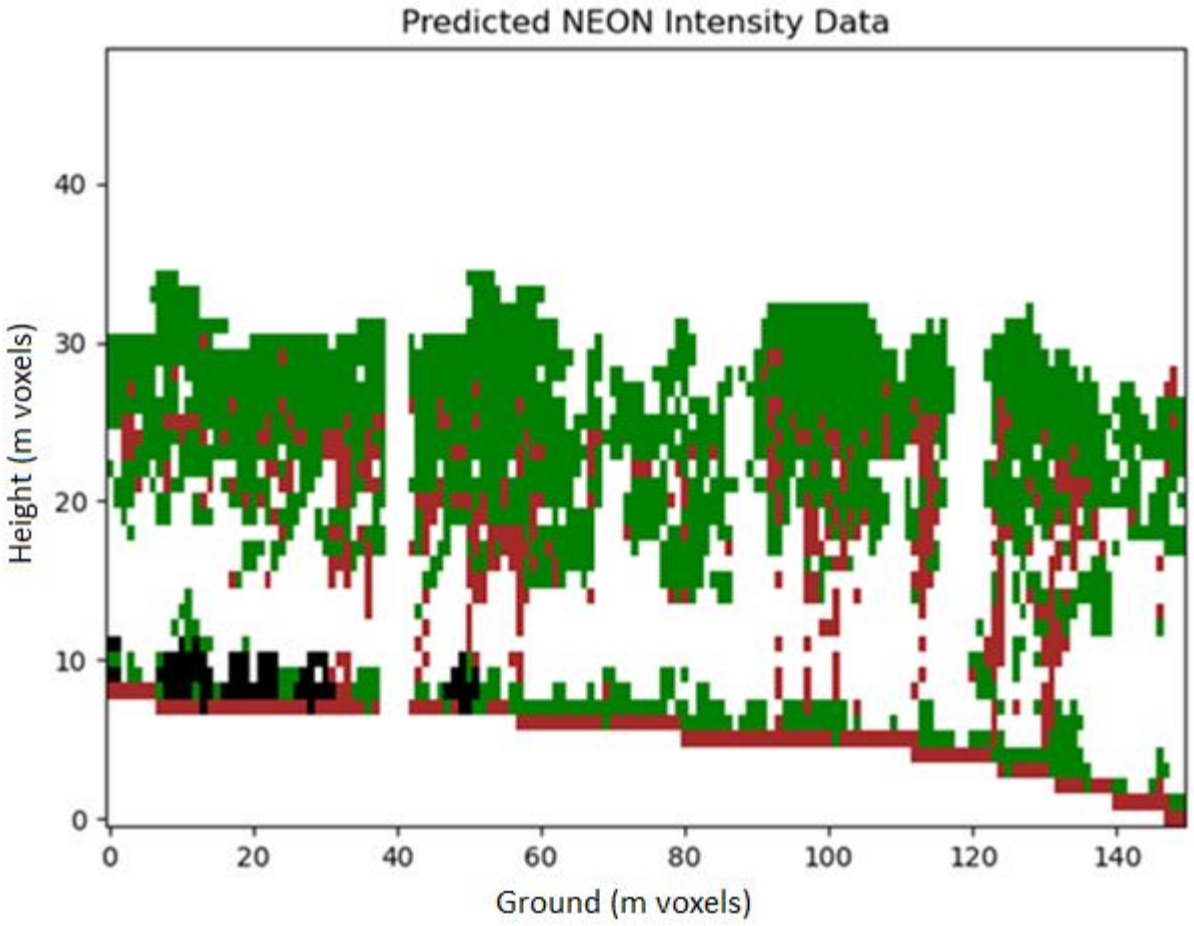

Figure 31: Predicted output of the NEON model (150 m x 150 m, 1 m voxels). Leaf predictions were good because most of the forest elements are leaves. Object predictions were also good because of the defined shapes. Because of the larger voxels however, the objects were not as detailed.

150 m x 150 m 7x7x7 three-channel (intensity, background threshold, local z-values)

The 70% recall result for the NEON (150 m x 150 m, local z-values) single-stage model was very close to the 67% precision result in Table 23. At risk of repeating several conclusions, the results were attributed to similar reasons as those mentioned earlier; also see Figure 31. Most of the background voxels were correctly predicted with 94% precision and 88% recall. Leaf voxels exhibited good (69%) precision and recall (64%) rates, while the bark voxels had a low 3% precision rate. The bark recall rate is surprisingly high at 53%, but that again is most likely due to very few voxels being

predicted as bark. Finally, the ground voxels were predicted accurately and the fewer than 2000 object voxels exhibited precision and recall rates of 71% and 63%, respectively, even from a high altitude with 0.5 m voxel dimension.

Table 23: Confusion matrix of the simulated NEON model (150 m x 150 m, 7x7x7 inputs, 0.5 m voxels, local z-values) with background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Overall Recall	70%		Total Voxels	1241372		
Overall Precision	67%					
Class	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.94	0.05	0.00	0.00	0.00	94%
Leaf	0.29	0.69	0.00	0.01	0.00	69%
Bark	0.40	0.57	0.03	0.00	0.00	3%
Ground	0.03	0.02	0.00	0.95	0.00	95%
Objects	0.20	0.08	0.00	0.02	0.71	71%
Recall	88%	64%	53%	82%	63%	

Table 24: Confusion matrix of the simulated NEON model (150 m x 150 m, 7x7x7 inputs, 0.5 m voxels, local z-values) without background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Class	Leaf	Bark	Ground	Objects
Leaf	0.98	0.01	0.02	0.00
Bark	0.94	0.05	0.00	0.00
Ground	0.02	0.00	0.98	0.00
Objects	0.10	0.00	0.02	0.88

The model was trained on the RIT RC Cluster [53], given that the input size and amounts are both much larger. Because of the z-value channel, the predicted results were much better than those from the previous 150 m x 150 m plots, where only intensities were used. Objects and ground performed very well, since their borders are straight, easily forming a contrasting linear pattern with the z-value channels.

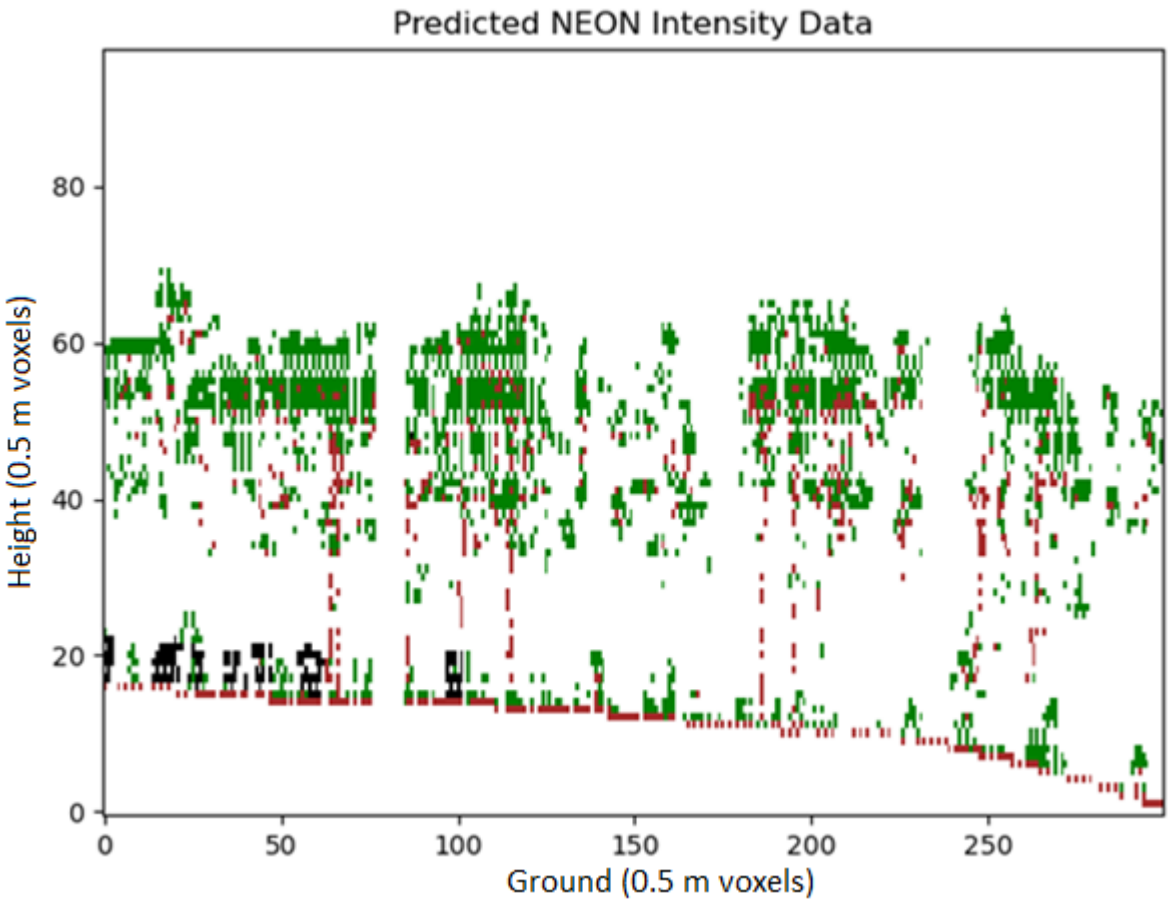

Figure 32: Predicted output of the NEON model (150 m x 150 m, local z-values). Leaf predictions were decent because most of the forest elements are leaves. Both ground and object predictions were excellent because the input channels included both the simulated NEON intensities and local z-values.

150 m x 150 m 7x7x7 three-channel Keras Tuner (0.5 m voxels)

The final simulation model, namely that for the NEON (150 m x 150 m, local z-values) single-stage model, exhibited a 68% recall result, which was close to the 69% precision result in Table 25. Most of the background voxels were correctly predicted with 93% precision and 90% recall. Leaf voxels exhibited good 70% precision and 64% recall rates, and bark voxels had a much better 12% precision rate, since the tuner resulted in a far lower learning rate. This result makes sense because there are many voxels overall, so a lower learning rate is best for tracking hidden voxels and small voxel classes. The bark recall rate is surprisingly high at 53%, which, unsurprisingly, is most likely due

to very few voxels being predicted as bark. Ground voxel predictions also experienced a significant improvement with 78% precision and 59% recall.

Table 25: Confusion matrix of the simulated NEON model (150 m x 150 m, 7x7x7 inputs, 0.5 m voxels, Keras Tuner) with background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Overall Recall	68%		Total Voxels	1250756		
Overall Precision	69%					
Class	Background	Leaf	Bark	Ground	Objects	Precision
Background	0.93	0.06	0.01	0.00	0.00	93%
Leaf	0.25	0.70	0.03	0.01	0.00	70%
Bark	0.33	0.54	0.12	0.00	0.00	12%
Ground	0.07	0.01	0.00	0.92	0.00	92%
Objects	0.15	0.05	0.00	0.02	0.78	78%
Recall	90%	64%	45%	81%	59%	

Table 26: Confusion matrix of the simulated NEON model (150 m x 150 m, 7x7x7 inputs, 0.5 m voxels, Keras Tuner) without background voxel counts. Each row represents the instances in an actual class, and each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1.

Class	Leaf	Bark	Ground	Objects
Leaf	0.94	0.04	0.02	0.00
Bark	0.81	0.18	0.00	0.00
Ground	0.01	0.00	0.99	0.00
Objects	0.05	0.00	0.02	0.92

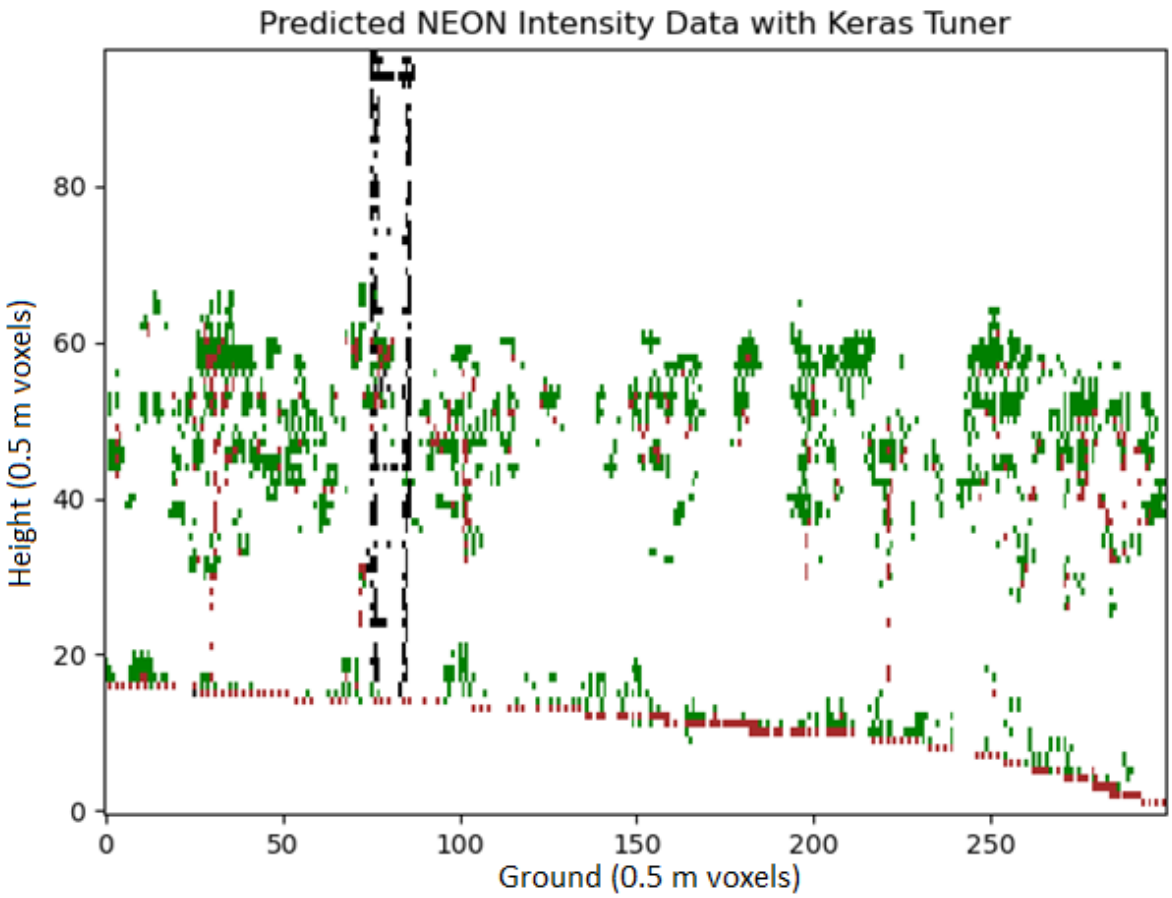

Figure 33: Predicted output of the NEON model (150 m x 150 m, local z-values). Leaf predictions were decent because most of the forest elements are leaves. Both ground and object predictions were excellent because the input channels included both the simulated NEON intensities and local z-values.

REAL DATA MODEL (NEON waveform lidar data)

50 m x 50 m (most data at canopy level)

In this case a confusion matrix cannot be provided, since the real data do not have corresponding ground truth data. Additionally, due to waveform range limitations, almost no ground was returned. Overall, there is far less information, when compared to the simulated waveform data. However, the top canopy classification seems to be accurate from a visual perspective. Most voxels are classified as leaves, and most voxels classified as bark are inside the leaves.

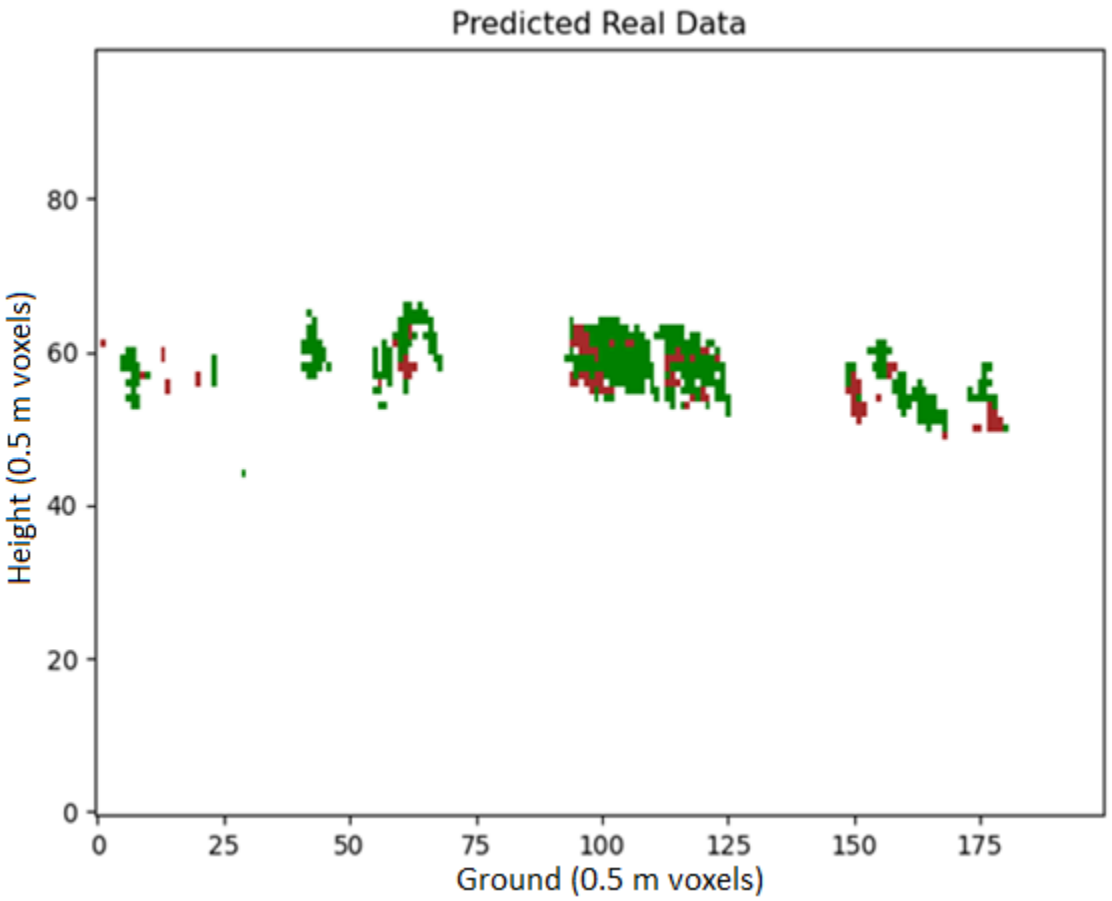

Figure 34: Predicted output of the real data model (50 m x 50 m). Leaf predictions visually appeared to be accurate, given that the top canopy is mostly composed of leaves, and the leaf distribution shapes are mostly random. Bark predictions also seem on par, since bark is usually surrounded by leaves in a very specific manner (branches).

Summary

CONCLUSIONS

The main purpose of this study was to predict forest voxel fills (classes), mainly based on simulated full waveform VLP-16 UAS and NEON lidar data. The SVC algorithm was the first method used; the inputs had to be flattened, because with SVC, only one unit produces the activation output. Predictions for the background and ground voxels were very accurate, since these are the easiest to group, given their attributes. Most voxels are background, and the ground is at a very specific height and exhibit a standard form. However, using SVC, a significant portion of leaf voxels were incorrectly predicted, and few bark and object voxels were correctly predicted. This is because only one unit produces the activation output in a classifier, resulting in limited input information. Additionally, there were only 232 object voxels. The classifier is too simple to be able to work with unique shapes of objects. Therefore, we hypothesized that a CNN may be better suited for these predictions, as there are far more classifiers and layers which can detect many different, diverse kinds of patterns [51].

Keras was used as the main library for the layers for the CNNs. More specifically, 3D convolutional and maximum pooling layers were used. The main online tutorial used to develop the CNN asserted that the intensity values had to be converted to RGB values [54], but simply adding a fifth axis to the inputs produced the same results, without any loss in accuracy. Additionally, two-thirds of the memory was freed, since a (R, G, B) conversion requires an additional dimension of length three, but adding an axis adds a dimension of only length one.

The first approach was to convert the VLP-16 dataset to RGB values, where 11x11x11 arrays surrounding each voxel were inputs, and voxel fill labels were outputs. According to the results, background, ground, and object voxel predictions were all significantly more accurate (66% vs. 36% precision, 74% vs. 40% recall). Object predictions were accurate in this case (76% precision, 82% recall), since their shape is far less random when compared to other elements. Some leaf voxels were

accurately predicted, but the overall precision of 53% was still low. This is likely because leaf voxels can be mostly random, while exhibiting significant overlap with bark voxels.

Further analysis of the results demonstrated that many edge points were not accurately detected. A two-stage plan was developed as result. For the first stage, voxels were reclassified as part of either background or forest categories. Using this method, background noise from the inputs can be filtered out more accurately. The predicted background voxels then were all set to zero in the input array. The new inputs were subsequently trained with the initial outputs to create a second model. Since the number of possibilities was reduced, leaf and bark voxels were now far more accurately predicted (86% vs. 53% leaf, 42% vs. 10% bark precision). Ground and object voxels still yielded high rates of accuracy (98% vs. 96% ground, 95% vs. 76% object precision).

After similar results were obtained by simply appending a 5D axis, two additional channels were added: local z-values (height above ground) and a background threshold. The local z-values provided additional vertical structure for each input, making object detection far more accurate (96% vs. 76% single-stage precision, 78% vs. 82% single-stage recall). The background threshold made filtering out very small values effective, thus making leaf detection far more accurate (71% vs. 53% single-stage precision, 64% vs. 60% single-stage recall). The three-channel model performed as well as the two-stage model when trained on the ground truth data categories, so the two stages were no longer needed.

When simulated NEON waveform lidar data were used as the input for the same CNN architecture, bark prediction accuracy was close to zero due to the sparsity of the data. Additionally, object prediction accuracy was around 50%, because although the input data were sparse, the objects still had a well-defined shape. Object prediction accuracy was improved to 70% by using much larger 150 m x 150 m simulated data and ground truth arrays, and by adding the local z-values for each returned input array after the RGB values were removed.

Additionally, using the Keras Tuner produced significant improvements in bark detection for both simulated VLP-16 and NEON waveform datasets (36% vs. 28% VLP-16, 12% vs. 3% NEON precision), and a measurable improvement in object detection for the NEON dataset (78% vs. 71% precision). From the Keras Tuner, the *Hyperband* tuner was used to quickly train and compare different combinations of hyperparameters to determine the best values. The units of the second dense layer and the learning rate were both selected for tuning. The initial dense layer was mostly accurate, but the learning rate was reduced by a factor of 10, from 0.001 to 0.0001, due to the wide number and variety of voxels.

One limitation of these results is that all the input data were simulated by DIRSIG. The DIRSIG parameters also included a 3D mapping of the forest, making reconstruction of the input voxels relatively straightforward. Real data could be taken from RIT's VLP-16 drone or NEON's Optech Gemini airborne platform. However, the caveat remains that it is quite challenging to generate accurate and comprehensive truth data, whereas we have 100% confidence in the simulated truth data. Most likely, the results would be less accurate in the case of real data, since the overall structure would need to be determined.

We concluded that 3D CNNs are far more suited than classifiers for this kind of problem, given the semi-random nature of bark and leaf elements. More specifically, the two-stage CNN model had an overall higher accuracy, because most of the background noise was filtered out. The strong performance of object detection is significant, as the ability to track objects is important for safety and security. More datasets with greater detail should be utilized for training in the future in order to include more natural variability and evaluate accuracies across different forest types and structures. Finally, other algorithms, such as *KPConv* and *FWNet*, should be studied for this purpose, since these approaches segment and classify data in ways which consider their intrinsic properties.

FUTURE WORK AND IMPROVEMENTS

One major area that should be improved is the overall accuracy of the voxel classification. There are several possible algorithms which involve advanced deep learning methods:

- Kernel Point Convolution (*KPConv*) is a form of convolution that is flexible and deformable for point clouds. For this algorithm, there is no intermediate representation. The convolution weights are in Euclidean space, and are applied to input points in close proximity. In this case, the focus is on scene objects, and deformable kernels improve the ability to adapt to scene object geometry. The network was tested on multiple datasets, such as S3DIS [64], which contains indoor large spaces. *KP-FCNN* is the name of the fully convolutional network which typically is used for segmentation. The architecture segments small sub-clouds contained in spheres. The spheres are picked randomly during training to maximize variety. *KPConv* was found to be useful for large and diverse datasets. The kernel combines a strong descriptive power and great learnability [64].
- The Full-Waveform Network (*FWNet*) is an architecture which uses semantic segmentation for full-waveform lidar data. This network directly handles the waveform data without any conversion process, such as projection onto a 2D grid or calculation of handcrafted features. This is a PointNet-based architecture, so local and global features of every input waveform and their geographical coordinates can be extracted. The next classifier consists of convolutional operational layers, which predict the class vector corresponding to the input waveform from the extracted local and global features. The trained *FWNet* achieved higher scores in its recall, precision, and F1 score for unseen test data, when compared to other recognized methods. Overall, this network for local and global feature extraction allows for semantic segmentation training without needing high levels of knowledge on waveform data, or translation into 2D elements [65].

- Additionally, predicted voxel data could be expanded to fill the entire 600 m x 500 m scene, which is centered on the overall 700 m x 500 m plot. There should also be scenes which contain voxels from conifer, deciduous, and mixed forest canopies. The program's algorithmic complexity could be redesigned to handle more layers, so that the inputs will be more accurate. Utilizing this method will require more computational resources, but likely will provide more robust results.
- The current plot that is being used is Plot B, Section 3. The dimensions of this plot are approximately 53 m x 67 m x 27 m, and the minimum X and Y coordinates are (594 m, 50 m), i.e., when using 0.5m voxels, this comprises ~800k total voxels. Three sensor passes were used to detect the forest. If the full scene is used, the total dataset is estimated to reach over five million voxels, though additional sensor passes will be needed. Training data for each category needs to be proportional to the category's total amount. Additionally, for the initial data, the order of the Nth Catalan number should be greater for more layer depth. However, when N is equal to 20, the number of possible paths exceeds one billion, which is extremely complex.

Overall, we created a 3D CNN algorithm used to predict forest voxel classifications based on the surrounding neighborhood of each voxel. In the future, this could be used to study forest structure remotely and to track sub-canopy man-made objects and trails.

References

[1] G. B. Bonan, "Forests and climate change: forcings, feedbacks, and the climate benefits of forests," *Science,* vol. 320, no. 5882, pp. 1444-1449, 2008.

[2] K. Zhao and R. B. Jackson, "Biophysical forcings of land-use changes from potential forestry activities in North America," *Ecol. Monogr.,* vol. 84, no. 2, pp. 329-353, 2014.

[3] L. Wallace, R. Musk and A. Lucieer, "An Assessment of the Repeatability of Automatic Forest Inventory Metrics Derived From UAV-Borne Laser Scanning Data," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 52, no. 11, p. 7160–7169, 2014.

[4] M. A. Wulder, C. W. Bater, N. C. Coops, T. Hilker and J. C. White, "The role of LiDAR in sustainable forest management," *The Forestry Chronicle,* vol. 84, no. 6, p. 807–826, 2008.

[5] J. Hyyppä, M. Holopainen and H. Olsson, "Laser Scanning in Forests," *Remote Sensing*, vol. 4, no. 10, p. 2919–2922, 2012.

[6] A. W. Doerry and F. M. Dickey, "Synthetic aperture radar," *Optics and photonics news*, vol. 15, no. 11, pp. 28-33, 2004.

[7] E. F. Moran, E. Brondizio, P. Mausel and Y. Wu, "Integrating Amazonian Vegetation, Land-Use, and Satellite Data," *BioScience,* vol. 44, no. 5, p. 329–338, 1994.

[8] M. Imhoff, "Radar backscatter and biomass saturation: ramifications for global biomass inventory," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 33, no. 2, p. 511–518, 1995.

[9] T. Raj, "A survey on LiDAR scanning mechanisms," *Electronics,* vol. 9, no. 5, p. 741, 2020.

[10] E. Baltsavias, "Airborne laser scanning: basic relations and formulas," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 54, no. 2-3, p. 199–214, 1999.

[11] P. Romanczyk, J. van Aardt, K. Cawse-Nicholson, D. Kelbe, J. McGlinchy and K. Krause, "Assessing the impact of broadleaf tree structure on airborne full waveform small-footprint LiDAR signals through simulation," *Canadian Journal of Remote Sensing,* vol. 39, no. s1, pp. S60-S72, 2013.

[12] J. Wu, J. van Aardt and G. P. Asner., "A Comparison of Signal Deconvolution Algorithms Based on Small-Footprint LiDAR Waveform Simulation," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 49, no. 6, pp. 2402-2414, 2011.

[13] W. Yao, J. van Aardt, M. van Leeuwen, D. Kelbe and P. Romanczyk, "A simulation-based approach to assess sub-pixel vegetation structural variation impacts on global imaging spectroscopy," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 56, no. 7, pp. 4149-4164, 2018.

[14] W. Yao, M. van Leeuwen, P. Romanczyk, D. Kelbe and J. van Aardt, "Towards an improved LAI collection protocol via simulated and field-based PAR sensing," *Sensors,* vol. 16, no. 7, p. 1092, 2016.

[15] S. Verma, "Understanding 1D and 3D Convolution Neural Network | Keras," Medium, 20 September 2019. [Online]. Available: https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610. [Accessed 5 June 2022].

[16] Z. Liu, H. Tang, Y. Lin and S. Han, "Point-Voxel CNN for Efficient 3D Deep Learning," in *NeurIPS 2019,* Vancouver, 2019.

[17] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, 2015.

[18] A. Salcedo-Bosch, F. Rocadenbosch, M. A. Gutiérrez-Antuñano and J. Tiana-Alsina, "Estimation of wave period from pitch and roll of a lidar buoy," *Sensors,* vol. 21, no. 4, p. 1310, 2021.

[19] R. J. McGaughey, Fusion/Ldv: Software for Lidar Data Analysis and Visualization, Forest Service, Pacific Northwest Research Station: United States Department of Agriculture, 2016.

[20] J. Li and M. Chapman, " Terrestrial mobile mapping towards real-time geospatial data collection," in *Geospatial Information Technology for Emergency Response*, CRC Press, 2008, pp. 103-123.

[21] K. Krause and T. Goulden, "NEON L0-TO-L1 DISCRETE RETURN LiDAR (ATBD)," 25 March 2022. [Online]. Available: https://data.neonscience.org/api/v0/documents/NEON.DOC.001292vB. [Accessed 5 June 2022].

[22] M. J. Campbell, P. E. Dennison, A. T. Hudak, L. M. Parham and B. W. Butler, "Quantifying understory vegetation density using small-footprint airborne lidar," *Remote sensing of environment,* vol. 215, pp. 330-342, 2018.

[23] K. N. Salvaggio, C. Salvaggio and S. Hagstrom, "A voxel-based approach for imaging voids in three-dimensional point clouds.," *Geospatial InfoFusion and Video Analytics IV; and Motion Imagery for ISR and Situational Awareness II,* vol. 9089, p. 90890E, 2014.

[24] D. Kukenbrink, F. Schneider, R. Leiterer, M. Shaepman and F. Morsdorf, "Quantification of hidden canopy volume of airborne laser scanning data using a voxel transversal algorithm.," *Remote Sensing of Environment,* vol. 194, pp. 424-436, 2017.

[25] M. Béland, D. D. Baldocchi, J. L. Widlowski, R. A. Fournier and M. M. Verstraete, "On seeing the wood from the leaves and the role of voxel size indetermining leaf area distribution of forests with terrestrial LiDAR.," *Agricultural and Forest Meteorology,* vol. 184, pp. 82-97, 2014.

[26] M. Béland, J. L. Widlowski, R. A. Fournier, J. F. Côté and M. M. Verstraete, "Estimating leaf area distribution in savanna trees from terrestrial LiDAR Measurements," *Agricultural and Forest Meteorology,* vol. 151, pp. 1252-1266, 2011.

[27] M. A. Lefsky and M. R. McHale, "Volume estimates of trees with complex architecture from terrestrial laser scanning," *Journal of Applied Remote Sensing,* vol. 2, no. 1, p. 023521, 2008.

[28] H. Weiser, L. Winiwarter, K. Anders, F. E. Fassnacht and B. Höfle, "Opaque voxel-based tree models for virtual laser scanning in forestry applications," *Remote Sensing of Environment,* vol. 265, p. 112641, 2021.

[29] S. T. Hagstrom, *Voxel-Based LIDAR Analysis and Applications,* Rochester: Rochester Institute of Technology, 2014.

[30] J. Huang and S. You, "Point cloud labeling using 3D Convolutional Neural Network," in *2016 23rd International Conference on Pattern Recognition (ICPR),* Cancun, 2016.

[31] S. Hancock, K. Anderson, M. Disney and K. J. Gaston, "Measurement of fine-spatial resolution 3D vegetation structure with airborne waveform lidar: Calibration and validation with voxelised terrestrial lidar," *Remote Sensing of Environment,* vol. 188, pp. 37-50, 2017.

[32] H. Sun, Z. Wang, Y. Chen, W. Tian, W. He, H. Wu, H. Zhang, L. Tang, C. Jiang, J. Jia, Z. Duan, H. Zhou, E. Puttonen and J. Hyyppä, "Preliminary verification of hyperspectral LiDAR covering VIS-NIR-SWIR used for objects classification," *European Journal of Remote Sensing,* vol. 55, pp. 291-303, 2022.

[33] J. van Aardt and K. Krause, "Enhanced 3D Sub-Canopy with Airborne and Spaceborne Full-Waveform LiDAR," 2019. [Online]. Available: https://www.rit.edu/dirs/research/enhanced-3d-sub-canopy-airborne-and-spaceborne-full-waveform-lidar. [Accessed 5 June 2022].

[34] J. Wu, K. Cawse-Nicholson and J. van Aardt, "3D tree reconstruction from simulated small footprint waveform lidar.," *Photogrammetric Engineering and Remote Sensing*, vol. 79, no. 12, pp. 1147-1157, 2013.

[35] J. van Aardt, M. Arthur, G. Sovkoplas and T. Swetnam, "LiDAR-based estimation of forest floor fuel loads using a novel distributional approach," in *International Conference on Lidar Applications for Assessing Forest Ecosystems*, Hobart, 2011.

[36] R. Yamashita, M. Nishio, R. K. G. Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, p. 611–629, 2018.

[37] M. Stewart, "Simple Introduction to Convolutional Neural Networks," Medium, 26 February 2019. [Online]. Available: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac. [Accessed 5 June 2022].

[38] D. Orwig, D. Foster and A. Ellison, "Harvard Forest CTFS-ForestGEO Mapped Forest Plot since 2014," Harvard Forest Data Archive: HF253, Harvard Forest, 2015.

[39] V. LiDAR, "VLP-16 User Manual," 2019. [Online]. Available: https://velodynelidar.com/wp-content/uploads/2019/12/63-9243-Rev-E-VLP-16-User-Manual.pdf. [Accessed 8 June 2022].

[40] DIRS Unmanned Aerial Systems Team, "DJI Matrice-600," [Online]. Available: https://www.rit.edu/facilities/unmanned-aerial-systems-drone-lab. [Accessed 10 June 2022].

[41] V. LiDAR, "User's Manual and Programming Guide," August 2015. [Online]. Available: https://drive.google.com/file/d/1E2UINeX1kjnsha3XUHV0DmHbiEvLXsHG/view. [Accessed 8 June 2022].

[42] O. Incorporated, "Gemini Summary Specificaion Sheet," 22 August 2011. [Online]. Available: https://www.mertind.com/argentina/lidar_aereo/Folleto%20ALTM%20Gemini.pdf. [Accessed 15 June 2022].

[43] M. G. Saunders, "VoxelizeHDF Intro," YouTube, 24 January 2022. [Online]. Available: https://www.youtube.com/watch?v=O4NxS1K4HmY.

[44] M. G. Saunders, "d5lidar Module," GitHub, 2022. [Online]. Available: https://github.com/mgradysaunders/d5lidar.

[45] E. Lindberg, K. Olofsson, J. Holmgren and H. Olsson, "Estimation of 3D vegetation structure from waveform and discrete return airborne laser scanning data," *Remote Sensing of Environment,* vol. 118, pp. 151-161, 2012.

[46] D. A. Pisner and D. M. Schnyer, "Support vector machine," in *Machine learning,* Academic Press, 2020, pp. 101-121.

[47] V. Kecman, "Support Vector Machines - An Introduction," in *Support Vector Machines: Theory and Applications,* Berlin, Springer Science & Business Media, 2005, pp. 1-49.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel and others, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research,* vol. 12, no. Oct, p. 2825–2830, 2011.

[49] T. Fletcher, "Support Vector Machines Explained," 23 December 2008. [Online]. Available: https://www.csd.uwo.ca/~xling/cs860/papers/SVM_Explained.pdf. [Accessed 3 June 2022].

[50] G. S. Handelman, H. K. Kok, R. V. Chandra, A. H. Razavi, M. J. Lee and H. Asadi, "eDoctor: machine learning and the future of medicine," *Journal of internal medicine,* vol. 284, no. 6, pp. 603-619, 2018.

[51] S. Y. Chaganti, I. Nanda, K. R. Pandi, T. G. Prudhvith and N. Kumar, "Image Classification using SVM and CNN," in *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, Gunupur, 2020.

[52] I. O. Taybi, T. Gadi and R. Alaoui, "2DSlicesNet: A 2D Slice-Based Convolutional Neural Network for 3D Object Retrieval and Classification," *IEEE Access,* vol. 9, p. 24041–24049, 2021.

[53] *Research computing services,* Rochester Institute of Technology, 2022.

[54] S. Aggarwal, "3D-MNIST Image Classification," 12 January 2018. [Online]. Available: https://medium.com/shashwats-blog/3d-mnist-b922a3d07334. [Accessed 11 June 2022].

[55] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering,* vol. 9, no. 3, pp. 90-95, 2007.

[56] F. e. a. Chollet, "Keras," 2015. [Online]. Available: https://keras.io.

[57] S. Ruder, "An overview of gradient descent optimization algorithms," 19 January 2016. [Online]. Available: https://ruder.io/optimizing-gradient-descent/. [Accessed 3 June 2022].

[58] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, San Diego, 2015.

[59] A. Kumar, "Different Types of CNN Architectures Explained: Examples," Vitalflux.com, 12 April 2022. [Online]. Available: https://vitalflux.com/different-types-of-cnn-architectures-explained-examples/. [Accessed 17 August 2022].

[60] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin and L. Invernizzi, "Keras Tuner," 2019. [Online]. Available: https://github.com/keras-team/keras-tuner. [Accessed 22 August 2022].

[61] " Introduction to the Keras Tuner," TensorFlor, 7 May 2023. [Online]. Available: https://www.tensorflow.org/tutorials/keras/keras_tuner. [Accessed 7 July 2023].

[62] "HyperParameters," Keras, [Online]. Available: https://keras.io/api/keras_tuner/hyperparameters/. [Accessed 7 July 2023].

[63] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," *Journal of Machine Learning Research*, vol. 18, pp. 1-52, 2018.

[64] H. Thomas, C. R. Qi, J. E. Deschaud, B. Marcotegui, F. Goulette and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF international conference on computer vision*, Seoul, 2019.

[65] T. Shinohara, H. Xiu and M. Matsuoka, "FWNet: Semantic segmentation for full-waveform LiDAR data using deep learning," *Sensors*, vol. 20, no. 12, p. 3568, 2020.