

**Bachelor Thesis**



**Czech  
Technical  
University  
in Prague**

**F3**

**Faculty of Electrical Engineering  
Department of Cybernetics**

## **A LiDAR Data Annotation Tool with the Ability to Learn**

**Adam Pyszko**

**Supervisor: RNDr. Petr Štěpán, Ph.D.  
May 2024**



## I. Personal and study details

Student's name: **Pyszko Adam** Personal ID number: **510661**  
Faculty / Institute: **Faculty of Electrical Engineering**  
Department / Institute: **Department of Cybernetics**  
Study program: **Open Informatics**  
Specialisation: **Artificial Intelligence and Computer Science**

## II. Bachelor's thesis details

Bachelor's thesis title in English:

**A Lidar Data Annotation Tool with the Ability to Learn**

Bachelor's thesis title in Czech:

**Nástroj pro anotování lidarových dat se schopností se učení**

Guidelines:

- 1) Learn about lidar data format and existing lidar data annotation tools.
- 2) Design and implement your own program for data annotation.
- 3) For the data annotation program, design and implement several types of neural networks for learning user labeled classes of data. Integrate these neural networks into the annotation program so that the user can pre-label new data according to the output of the selected neural network.
- 4) Test the annotation program on the provided dataset and evaluate the detection success of each neural network.

Bibliography / sources:

- [1] Pan, X., Gao, L., Marinoni, A., Zhang, B., Yang, F., & Gamba, P. (2018). Semantic labeling of high resolution aerial imagery and LiDAR data with fine segmentation network. Remote Sensing, 10(5), 743
- [2] Axelsson, M., Holmberg, M., Serra, S., Ovren, H., & Tull Dahl, M. (2021). Semantic labeling of lidar point clouds for UAV applications. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4314-4321)
- [3] B. Wang, V. Wu, B. Wu and K. Keutzer, "LATTE: Accelerating LiDAR Point Cloud Annotation via Sensor Fusion, One-Click Annotation, and Tracking," 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 2019, pp. 265-272, doi: 10.1109/ITSC.2019.8916980.
- [4] Sager, C., Zschech, P., & Köhl, N. (2021). labelcloud: A lightweight domain-independent labeling tool for 3d object detection in point clouds. arXiv preprint arXiv:2103.04970.
- [5] Andersson, R., & Andersson, E. (2019). LiDAR Pedestrian Detector and Semi-Automatic Annotation Tool for Labeling of 3D Data. Master's Theses in Mathematical Sciences.

Name and workplace of bachelor's thesis supervisor:

**RNDr. Petr Štápan, Ph.D. Multi-robot Systems FEE**

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **02.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

RNDr. Petr Štápan, Ph.D.  
Supervisor's signature

prof. Dr. Ing. Jan Kybic  
Head of department's signature

prof. Mgr. Petr Páta, Ph.D.  
Dean's signature

### III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

\_\_\_\_\_  
Date of assignment receipt

\_\_\_\_\_  
Student's signature

## Acknowledgements

I would like to dedicate this thesis to my family and friends, whose support has been invaluable. I am deeply grateful to my supervisor, RNDr. Petr Štěpán, Ph.D., for his guidance, insightful critique, flexibility and unwavering support throughout the duration of this thesis. His expertise and attention to detail have significantly shaped this thesis and his encouragement has been crucial in overcoming the challenges I have faced during my work.

## Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

The following artificial intelligence (AI) tools were used during the completion of this thesis: Deepl.com and ChatGPT-4 were utilised for stylistic text editing purposes. Github Copilot was integrated into the PyCharm Integrated Development Environment (IDE).

Prague, May 15, 2024

## Abstract

This thesis addresses the challenge of efficiently annotating LiDAR data to improve the navigation capabilities of drones in forested environments. The primary objective is the development of an annotation tool that leverages the capabilities of Convolutional Neural Networks (CNNs) to assist the user in segmenting and identifying tree trunks from LiDAR-generated data more effectively and efficiently. The project's core technique is the conversion of high-dimensional LiDAR data into a two-dimensional grayscale image format that is optimal for convolutional neural networks, specifically an adapted U-Net model. This approach not only simplifies the data's complexity but also enhances the model's performance in recognising and segmenting crucial environmental features.

**Keywords:** Annotation tool, Convolutional neural network, LiDAR

**Supervisor:** RNDr. Petr Štěpán, Ph.D.

## Abstrakt

Tato práce se zabývá problematikou efektivní anotace dat z LiDARu pro zlepšení navigačních schopností dronů v lesním prostředí. Hlavním cílem je vývoj anotačního nástroje, který využívá schopnosti konvolučních neuronových sítí (CNN) a pomáhá uživateli efektivněji a účinněji segmentovat a identifikovat kmeny stromů z dat generovaných pomocí LiDARu. Základní technikou projektu je převod vysokorozměrných dat LiDAR do dvourozměrného obrazového formátu ve stupních šedi, který je optimální pro konvoluční síť, konkrétně pro přizpůsobený model U-Net. Tento přístup nejen zjednodušuje složitost dat, ale také zvyšuje výkonnost modelu při rozpoznávání a segmentaci klíčových prvků prostředí.

**Klíčová slova:** Anotační nástroj, Konvoluční síť, LiDAR

**Překlad názvu:** Nástroj pro anotování lidarových dat se schopností se učit

# Contents

<b>1 Introduction</b>	<b>1</b>	<b>7 Conclusion</b>	<b>35</b>
1.1 State of the Art of Annotating tools . . . . .	2	<b>Bibliography</b>	<b>37</b>
1.2 Decision to create our annotation tool . . . . .	2		
1.3 Chapter Summaries . . . . .	3		
<b>2 LiDAR</b>	<b>5</b>		
2.1 Introduction to LiDAR technology	5		
2.2 LiDAR Specifications . . . . .	5		
2.3 Utilization of Point Cloud Library	6		
2.4 Creating a 2D representation for Convolutional Neural Networks . . . .	6		
<b>3 Convolutional Neural Network</b>	<b>9</b>		
3.1 State of the Art of CNNs for LiDAR data . . . . .	9		
3.2 Neural Network Implementation - PyTorch . . . . .	10		
3.3 The Fundamentals of Convolutional Neural Networks . . .	10		
3.4 U-Net . . . . .	12		
3.4.1 Key Components of the U-Net Model: . . . . .	12		
3.5 Development of Neural Network	12		
3.6 Utilization of PyTorch's Dataset and DataLoader . . . . .	13		
3.7 Training Process and Results on Generated Data . . . . .	14		
<b>4 Annotation Tool</b>	<b>17</b>		
4.1 Labeler . . . . .	17		
4.2 Functionalities . . . . .	18		
<b>5 Goals and results</b>	<b>23</b>		
5.1 Defining our evaluation metrics .	23		
5.2 Data selection and split . . . . .	25		
5.3 Results on selected images . . . .	25		
5.4 Complete results . . . . .	28		
5.4.1 Reduction of time requirements . . . . .	30		
<b>6 Future improvements and work</b>	<b>33</b>		
6.1 Multiclass labeling . . . . .	33		
6.2 Exploring the Integration of Multiple Neural Network Models .	33		
6.3 The long-term objective of enabling autonomous drone navigation . . . .	34		

## Figures

2.1 Picture with unshifted data . . . . .	6
2.2 Picture after shift . . . . .	6
3.1 Computation of convolutional layer, taken from[17] . . . . .	11
3.2 U-Net, taken from[12] . . . . .	13
3.3 Loss function during training . . . . .	15
3.4 Testing Process Results . . . . .	15
4.1 Descriptive picture of the application . . . . .	18
4.2 Main display area while zoomed-in . . . . .	19
5.1 Jaccard Index (IoU), taken from[26] . . . . .	24
5.2 Data with easy to recognize trunks . . . . .	26
5.3 Data with a high density of trees . . . . .	27
5.4 Data with trees in the distance . . . . .	28
5.5 Comparison of time spent labeling images with and without neural network assistance . . . . .	30
5.6 The visualisation of the speedup gained from neural network assistance for each image, accompanied by an average speedup factor . . . . .	31

## Tables

5.1 Results achieved on images from testing set . . . . .	29
---	----





# Chapter 1

## Introduction

The integration of LiDAR (Light Detection and Ranging) technology and modern artificial intelligence methods, especially neural networks, is the basis of current recognition methods for autonomous robots. In this work, we address the problem of tree detection and recognition when a drone is flying in a forest. This technological synergy enhances the ability of vehicles to better interpret and interact with their environment. In the context of drone navigation in dense forest areas, accurate annotation of LiDAR data becomes critical to understanding complex environments.

This thesis focuses on the development of an application that uses neural networks to efficiently annotate LiDAR data, which is essential for generating high-quality annotated training images. These annotated images are fundamental for training CNNs to accurately detect and classify environmental elements, thereby improving the operational efficiency and safety of drone navigation systems in challenging forest landscapes.

Segmenting tree trunks using LiDAR data is a complex problem. It requires the processing of high-dimensional data to accurately identify tree trunks amidst the varied elements of a forest. This task gains complexity due to the diverse shapes, sizes, and densities of trees. Convolutional neural networks, known for their effectiveness in image recognition and processing tasks, offer a promising approach to addressing these challenges.

This project deviates from conventional methods by transforming the range values captured by LiDAR – representing the distance from the sensor to different objects – into a grayscale image. In this format, the intensity of each pixel correlates directly with the distance measured, providing intuitive way to process and analyse LiDAR data through CNNs. This method not only simplifies the visualisation of spatial data but also enhances the ability of our neural networks to learn and recognise the relevant features for accurate object segmentation.

## 1.1 State of the Art of Annotating tools

In recent years, researchers have started to apply deep neural networks to LiDAR data for simple understanding tasks such as classification and semantic segmentation[1]. As a result, the need for efficient annotation tools has become paramount to unlock the full potential of these neural networks.

Currently, there are numerous commercial annotation tools which integrate advanced algorithms capable of handling high-resolution, 3D point cloud data efficiently. In this domain, tools such as Supervisely<sup>1</sup>, Pointly<sup>2</sup>, and Labelbox<sup>3</sup> have emerged as leaders, offering features such as automated object detection and semantic segmentation. These tools leverage machine learning techniques to enhance the accuracy and speed of data annotation, significantly reducing manual labour and improving the quality of annotations made.

Additionally, there are free alternatives, such as 3D-LiDAR-annotator[2], which facilitate labeling through the use of bounding boxes within the 3D environment. LabelCloud[3] works very similarly, also using the technique of bounding boxes in 3D. However, it should be noted that these tools do not possess the same capabilities for assisting users with annotation as their paid counterparts.

## 1.2 Decision to create our annotation tool

While the existing LiDAR annotation tools mentioned above offer valuable features and capabilities, they are primarily designed to annotate point cloud data directly in 3D. However, our approach of converting LiDAR range values into grayscale images necessitated the development of a bespoke annotation tool tailored to this specific methodology.

The conventional annotation tools may lack the flexibility and customisation required to fully exploit the advantages of our approach. Our method employs Convolutional Neural Networks (CNNs) for object segmentation from grayscale images generated by transforming LiDAR data. Consequently, our annotation tool incorporates features designed to facilitate the training and validation of CNN models on grayscale LiDAR images, including intuitive labeling interfaces and seamless integration with deep learning frameworks.

---

<sup>1</sup><https://supervisely.com/labeling-toolbox/3d-lidar-sensor-fusion>

<sup>2</sup><https://pointly.ai/>

<sup>3</sup><https://labelbox.com/product/annotate/>

## 1.3 Chapter Summaries

This thesis is organized into several chapters, each focusing on different aspects of developing our LiDAR data annotation tool that leverages the power of Convolutional Neural Networks (CNNs). Below is a brief overview of each chapter:

**Chapter 1: Introduction** - This chapter introduces the motivation behind the thesis, the importance of efficient LiDAR data annotation, and the role of CNNs in enhancing the interpretability of LiDAR data for autonomous drone navigation in forested environments.

Here we can also find the State of the Art which provides a comprehensive review of existing LiDAR annotation tools and methods. It discusses both commercial and open-source solutions, highlighting their capabilities and limitations.

**Chapter 2: LiDAR Technology** - Details the technical aspects of LiDAR technology, including its operational principles and applications. The chapter focuses on how LiDAR data is utilized in this project, emphasizing the conversion of LiDAR measurements into a format suitable for CNN processing.

**Chapter 3: Convolutional Neural Networks** - This chapter presents the State of the Art of the application of convolutional neural networks (CNNs) to LiDAR data. It describes the architecture of the CNN used in this project, with a particular focus on the U-Net model, which has been adapted for the analysis of LiDAR-generated 2D images.

**Chapter 5: Annotation Tool** - Discusses the development of the 'Labeler' tool, designed to facilitate the annotation of LiDAR data. It covers the tool's functionalities, user interface, and the integration of machine learning techniques to automate the annotation process.

**Chapter 6: Goals and Results** - Summarizes the objectives of the thesis and the results achieved. It evaluates the performance of the developed annotation tool and the CNN model in processing and annotating LiDAR data.

**Chapter 7: Future Improvements** - Outlines potential future enhancements for the annotation tool and CNN model. This includes exploring multi-class labeling and integrating additional neural network models to improve annotation accuracy and versatility.

**Chapter 8: Conclusion** - Concludes the thesis by reflecting on the research conducted, its implications for the field of LiDAR data annotation, and the future of autonomous navigation systems in complex environments.



## Chapter 2

### LiDAR

#### 2.1 Introduction to LiDAR technology

LiDAR, an acronym for Light Detection and Ranging, is a robot sensing technique employed in geology, forestry, urban planning etc. It emits laser light pulse towards objects and measures the time it takes for the light to reflect back. LiDAR uses reflected light, which enables faster and more accurate distance measurements with higher resolution than either radar or sonar[13]. It provides precise distance measurements and create accurate 3D representations of the target area or object.

#### 2.2 LiDAR Specifications

This thesis uses the OS0-128 LiDAR system[14], a high resolution sensor known for its exceptional data acquisition capabilities. The OS0-128 has 128 rows, providing a detailed 90-degree vertical field of view (vFoV), making it ideal for topographic surveys and 3D mapping in a variety of environments. The LiDAR system is configured in a top-mounted position on the drone, offering an unobstructed view that allows for comprehensive data collection from an elevated vantage point.

In contrast to common LiDAR data processing practices, which predominantly use XYZ coordinates, this research takes an unconventional approach by focusing primarily on the 'distance' parameter from the LiDAR data. This shift in the approach is driven by the hypothesis that analysing the raw distance measurements can provide sufficient information and streamline certain calculations, particularly in the context of our project.

### 2.3 Utilization of Point Cloud Library

The LiDAR sensor data is processed using the Point Cloud Library (PCL)[15], which is an open-source library commonly used for 3D point cloud data processing. PCL provides essential tools and functionalities that are crucial for the manipulation and analysis of point cloud data. This project focuses on the .pcd (Point Cloud Data) file format, which is commonly used for storing point cloud data. This format is highly efficient for handling large datasets, which is typical in LiDAR applications.

The data in this paper comes from a data forest that was created by the mrs group and is available on GitHub[14]. The data from the experiment are stored in rosbag format, from which they are converted to pcd files using the `pcl_ros bag_to_pcd` tool.

### 2.4 Creating a 2D representation for Convolutional Neural Networks

Our approach involves the translation of the distance measurements into a two-dimensional representation. This transformation is a strategic decision made to prepare the data for use in convolutional neural networks (CNNs). CNNs have demonstrated remarkable proficiency in processing and interpreting visual data. By converting the three-dimensional LiDAR data into a two-dimensional format, we aim to utilise the robust pattern recognition and feature extraction capabilities of CNNs. During the course of our experimentation, we encountered an unforeseen issue: the pixels in each row must be shifted by a specific amount in order to accurately represent the spatial information. The offset of each row of data points in the pcd file is determined by the internal configuration of the sensor, which is stored by the ROS system in the sensor configuration file. Figure 2.1 depicts the unshifted data, which fails to accurately represent the spatial relationships. Figure 2.2 illustrates the necessary adjustments for proper interpretation.

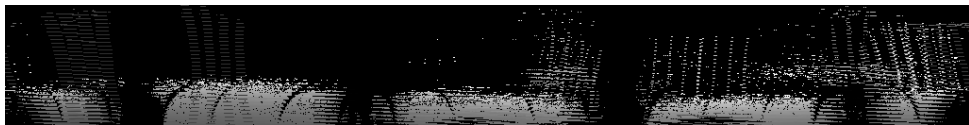


Figure 2.1: Picture with unshifted data



Figure 2.2: Picture after shift







## Chapter 3

# Convolutional Neural Network

### 3.1 State of the Art of CNNs for LiDAR data

The study[4] by researchers at Mississippi State University presents a detailed examination of using CNNs for segmenting LiDAR data in off-road environments. Their work demonstrates the application of the SqueezeSeg CNN model, particularly emphasizing its effectiveness in accurately identifying various objects and obstacles, including trees. They also compared the performance of LiDAR sensors with different beam counts, observing a marginal increase in accuracy with higher beam sensors. This research is vital for understanding the potential of CNNs in processing complex LiDAR datasets for navigation and obstacle detection in off-road or natural environment.

PointNet[5] and MultiView[6] CNNs are two prominent architectures in the field of 3D point cloud processing and analysis. PointNet revolutionized the field by providing a novel approach to directly process unordered point clouds without requiring complex pre-processing steps. It employs a symmetric function to extract features from individual points and capture global patterns, making it highly efficient for tasks like object classification and segmentation. On the other hand, MultiView CNNs leverage multiple views or perspectives of the same 3D scene to enhance the understanding and representation of complex spatial structures. By fusing information from multiple viewpoints, MultiView CNNs achieve robustness and resilience to occlusions and viewpoint variations, making them well-suited for tasks such as 3D reconstruction and scene understanding.

In the paper[7], the authors compared their results with both PointNet and MultiView CNNs. Their main objective was to correctly classify and then remove phantom effects caused by objects moving in parallel with the scanning platform. They also developed their own 3D point cloud annotation tool, which they used to create their dataset called SZTAKI CityMLS[8]. On this dataset they achieved better results than both PointNet and MultiView CNNs.

The researchers from Tokyo Institute of Technology do something very similar to our work in their paper[9]. They used different neural network models to correctly segment human bodies and managed to create an automatic annotator for LiDAR data containing humans. Their results showed that

Fully Convolutional Network (FCN) performed better than both PointNet and U-Net. Only Fully Convolutional DenseNets (FCDN) outperformed FCN in terms of recall.

The paper[10] presents an innovative method to automatically label 3D LiDAR data to generate training data for LiDAR-based Moving Object Segmentation (MOS). By processing data offline in batches, dynamic objects are identified using occupancy-based dynamic object removal, followed by instance segmentation and tracking using a Kalman filter<sup>1</sup>. The method labels moving objects such as pedestrians and cars, and static objects such as parked cars and buildings. Unfortunately, the offline labeling meant that it was not possible to segment real-time data while the vehicle was moving. Performance evaluations of multiple neural networks conducted on of the most well-known datasets, KITTI[11], show that the automatically labeled data achieves comparable and in some cases superior performance to manually labeled data.

All of these different results may suggest that there isn't a single best neural network model for all of the tasks associated with the processing of LiDAR data.

## 3.2 Neural Network Implementation - PyTorch

PyTorch is a widely used deep learning framework that is known for its flexibility, ease of use, and strong community support. It was developed by Facebook's AI Research lab and provides an intuitive interface for building and training deep learning models. Its dynamic computational graph is a particularly popular feature, as it allows for easier debugging and experimentation with models.[18] Therefore, it is ideal for rapid prototyping and experimentation in research, making it a valuable tool for this project.

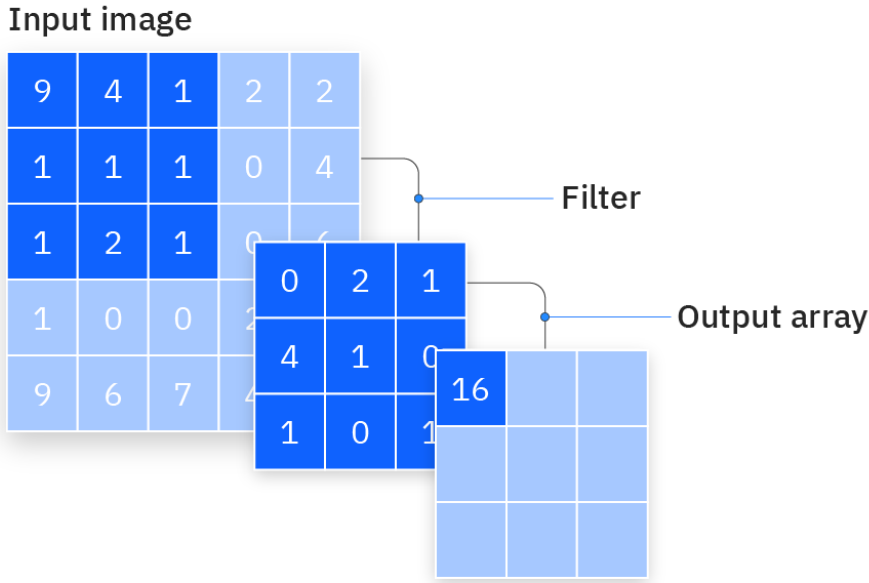
## 3.3 The Fundamentals of Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are specifically designed to recognise patterns in multidimensional data, making them highly effective for image analysis. The convolutional layer is the core component of a CNN, where most of the computation takes place. This layer utilises a filter, or kernel, to scan the input image, which usually has three dimensions corresponding to RGB colour channels. The convolution operation is executed by the kernel as it traverses the image, computing the dot product between the filter weights and the input pixels to generate a feature map.

This process highlights local dependencies in the image, such as edges and textures. The size of the kernel, typically a 3x3 matrix, and the stride, which

---

<sup>1</sup>Kalman filtering is an algorithm that uses a series of measurements observed over time, including statistical noise and other inaccuracies, and produces estimates of unknown variables



**Figure 3.1:** Computation of convolutional layer, taken from[17]

is the step size with which the kernel moves across the image, are crucial hyperparameters that impact the output volume.

After each convolution operation, the CNN applies a Rectified Linear Unit (ReLU) transformation to introduce nonlinearity. This enables the network to learn more complex patterns.[17]

The equation 3.1 demonstrates how to calculate convolution with a stride of 1 and no padding. Figure 3.1 provides an example of this operation.

$$\text{Output}[i][j] = \sum_{u=0}^{H-1} \sum_{v=0}^{W-1} \text{Input}[i+u][j+v] \times \text{Filter}[u][v] \quad (3.1)$$

where:

- $i, j$  are the row and column indices of the output feature map,
- $H, W$  are the height and width of the filter,
- $\text{Input}[i+u][j+v]$  is the value of the input image at the position that corresponds to the filter's current location,
- $\text{Filter}[u][v]$  is the value of the filter at position  $u, v$ .

## 3.4 U-Net

To analyze 2D LiDAR data representations we used a U-Net model designed with PyTorch. The U-Net architecture, a convolutional network originally designed for biomedical image segmentation. Developed by Olaf Ronneberger, Philipp Fischer, and Thomas Brox[12], U-Net is known for its effectiveness in training with a limited number of annotated samples. The architecture is characterised by a contraction path for context capture and a symmetric expansion path for precise localisation. While this structure is tailored to biomedical images, it has similarities to the SqueezeSeg[4] model, particularly in how it handles complex segmentation tasks.

The U-Net model architecture, consisting of downscaling and upscaling blocks, is well-suited for tasks that require precise localization.

### 3.4.1 Key Components of the U-Net Model:

#### ■ Double Convolution Block (DoubleConv)

This fundamental component of the model comprises two consecutive convolution operations, each followed by batch normalization and a ReLU activation function. This design is crucial for enhancing the model's ability to learn complex features from the input data.

#### ■ Downscaling Blocks (Down)

These blocks are responsible for reducing the spatial dimensions of the input data. Each downscaling block uses max pooling followed by a double convolution process, progressively increasing the number of features while decreasing the size of the feature maps.

#### ■ Upscaling Blocks (Up)

For upscaling, the model uses transposed convolutions[19]. This method effectively increases the spatial resolution of the feature maps, ensuring that the network learns from both detailed and broader spatial features.

## 3.5 Development of Neural Network

Prior to the selection of the convolutional U-net model, we explored a variety of neural network architectures. This exploration was crucial in comprehending the strengths and limitations of different approaches when applied to our task. These preliminary experiments ranged from basic feedforward neural networks to more complex architectures. The knowledge acquired from these trials was essential in guiding us towards the adoption of CNNs, in particular U-Net.

During this stage of development, we focused on working with randomly generated 'images' representing distance measurements from the LiDAR sensor. This approach was selected for its simplicity and ease of annotation.

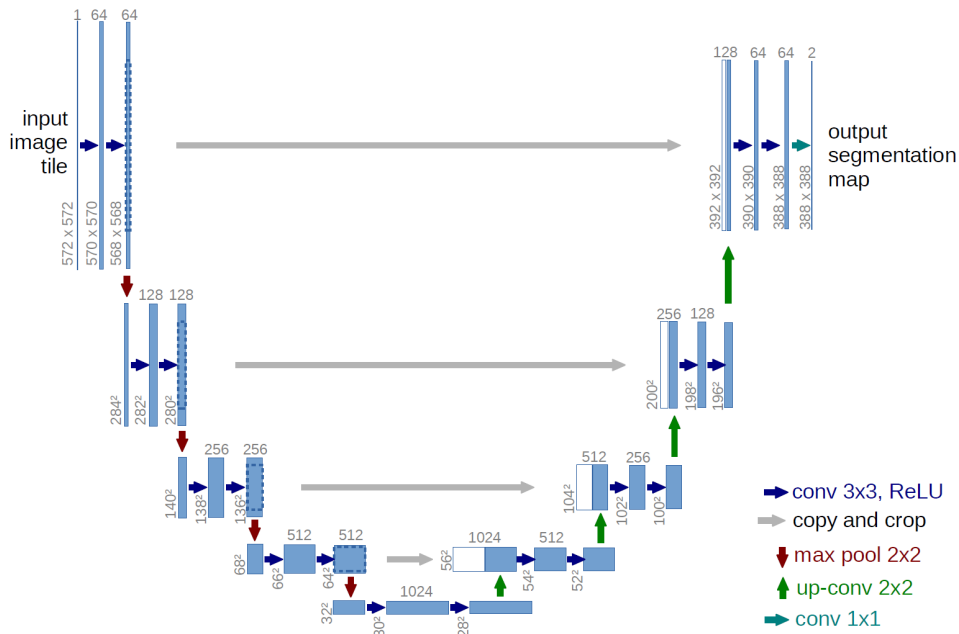


Figure 3.2: U-Net, taken from[12]

Starting with synthetic data allowed us to better control and comprehend the variables involved, making it easier to develop and refine our data processing and analysis methods. These artificially created images encode distance information and provide a straightforward and effective medium for testing and calibrating our deep learning models before applying them to more complex real-world LiDAR data.

### 3.6 Utilization of PyTorch's Dataset and DataLoader

To handle this data efficiently, we utilized PyTorch's Dataset and DataLoader classes. The Dataset class allowed us to access our data in a standardized way, integrating with PyTorch's deep learning models. We defined how the data was loaded and transformed through a custom Dataset subclass, ensuring that the 'pictures' were correctly formatted and pre-processed for input into our U-Net model.

Each image in our dataset is treated as an individual sample and contains distance values as pixel intensities. The custom Dataset class handles the loading of these images, applies necessary transformations, and converts them into tensor[20] format suitable for PyTorch models.

The DataLoader class is essential for improving the efficiency of the training process by providing batching, shuffling, and parallel loading of data. The U-Net model was trained using batches of distance images fed through the DataLoader. This optimized resource utilization and enhanced the model's training effectiveness.

## 3.7 Training Process and Results on Generated Data

During the training phase of our customised U-Net model, we utilised the Binary Cross-Entropy with Logits Loss[22] (BCEWithLogitsLoss) as our criterion. This loss function is particularly effective for binary classification tasks as it combines a sigmoid layer with the BCE loss in a single step, which is computationally more stable. For the optimization of the model parameters, we chose the Adam optimizer[21] (Adaptive Moment Estimation), with a learning rate set to 0.001. Adam optimizer is known for its efficiency in handling sparse gradients and adapting the learning rate during training, making it a suitable choice for our application.

The training results on the randomly generated distance images were highly encouraging. The model demonstrated significant ability to accurately interpret and classify the data, as evidenced by its performance metrics. The model's robustness was further validated when random noise was introduced to the images. Despite the added complexity and potential for confusion, the U-Net model maintained a high level of accuracy in its predictions. The effectiveness of our neural network architecture is highlighted by its resilience to noise in the data. This suggests that the model could perform reliably even under less-than-ideal real-world conditions where noise and data irregularities are common.

These results on the synthetic data provide a strong foundation for the next phase of our research, which involves applying the model to real-world LiDAR data. The model's ability to handle noise and its high accuracy in classification tasks give us confidence in its potential applicability and effectiveness in more complex and realistic scenarios. In figure 3.3 we can see the training process over 10 epochs, the vertical axis represents validation and training loss. Figure 3.4 shows the model's performance on one of the testing images after training.

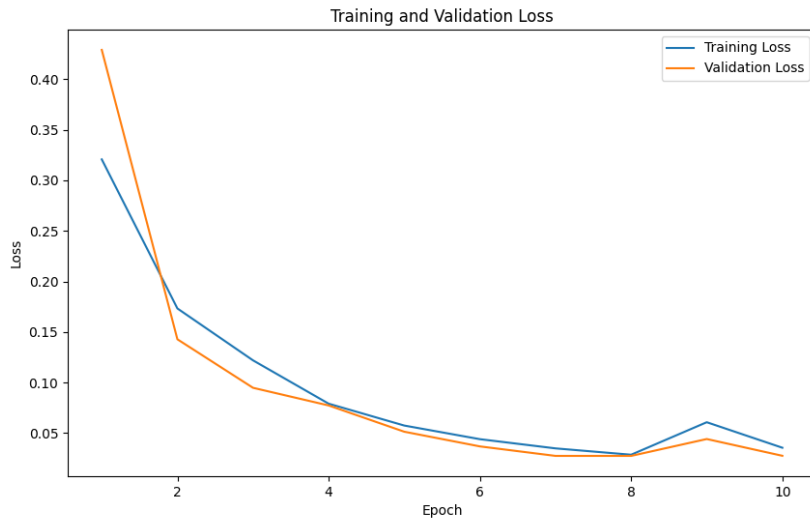
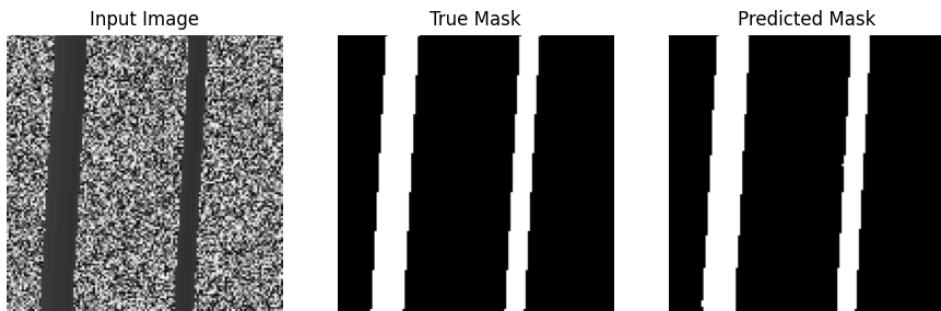


Figure 3.3: Loss function during training



Accuracy of the prediction: 99.82%

Figure 3.4: Testing Process Results





## Chapter 4

### Annotation Tool

One of the major hurdles in processing LiDAR data is the challenge of manual annotation of huge datasets. LiDAR data, characterised by its high-dimensional nature, presents a complex array of information that is often dense and intricate. Each data point in a LiDAR dataset represents not only spatial coordinates, but also distance and additional attributes such as light intensity, reflectivity, etc., which together create a highly detailed representation of the physical environment. Manually annotating this data to train machine learning models is a difficult task, largely due to the volume itself and complexity of the data.

The necessity for precise labeling of each data point or segment adds to the difficulty of the task. In real-world environments, LiDAR sensors capture a quantity of objects, ranging from simple structures to complex natural forms. The level of precision required demands a significant amount of time and effort, making manual annotation impractical and nearly impossible to accomplish with the required accuracy and efficiency.

To address this challenge, we have developed an application, called Labeler, designed to facilitate and expedite the annotation process for LiDAR data. This software employs machine learning and computer vision methodologies to enhance the efficiency and accuracy of data annotation. By automating key aspects of the annotation workflow, our application aims to optimise the processing of LiDAR data, enabling users to annotate datasets with increased speed and precision.

#### 4.1 Labeler

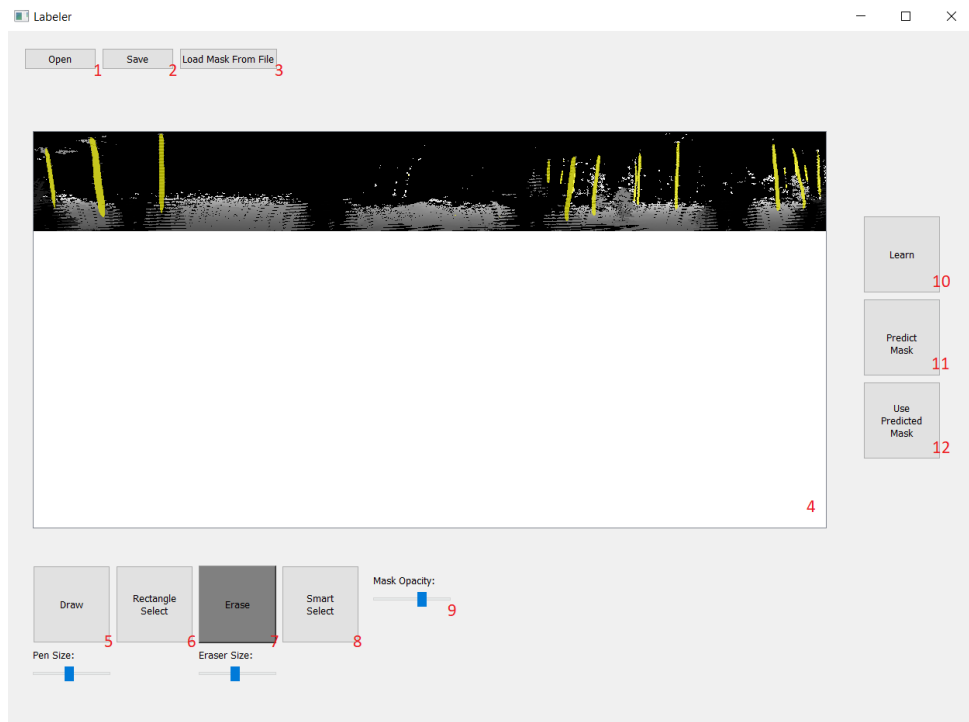
The Labeler application offers a comprehensive set of instruments for the precise and fast labeling of LiDAR images. Written using Python and leveraging the powerful PyQt5 framework[23], this software enables users to draw, delete, and modify annotations directly on the visual representation of LiDAR data. This allows for the detailed and exact labeling of a wide range of environmental characteristics. Enhanced by features such as smart selection, which utilizes color similarity algorithms to automatically identify and label contiguous regions with similar characteristics, the application provides a robust toolkit for efficient data processing.

The tool is designed with a focus on user interaction, allowing users to easily review and adjust the suggested annotations. This interaction is not only crucial for ensuring the accuracy of the data but also serves as a feedback mechanism for the models to learn and adapt. As users correct and refine the annotations, the models will use this input to fine-tune their algorithms, leading to a continuous improvement in the tool's annotation capabilities.

The complete code is available on [GitHub\[24\]](#).

## 4.2 Functionalities

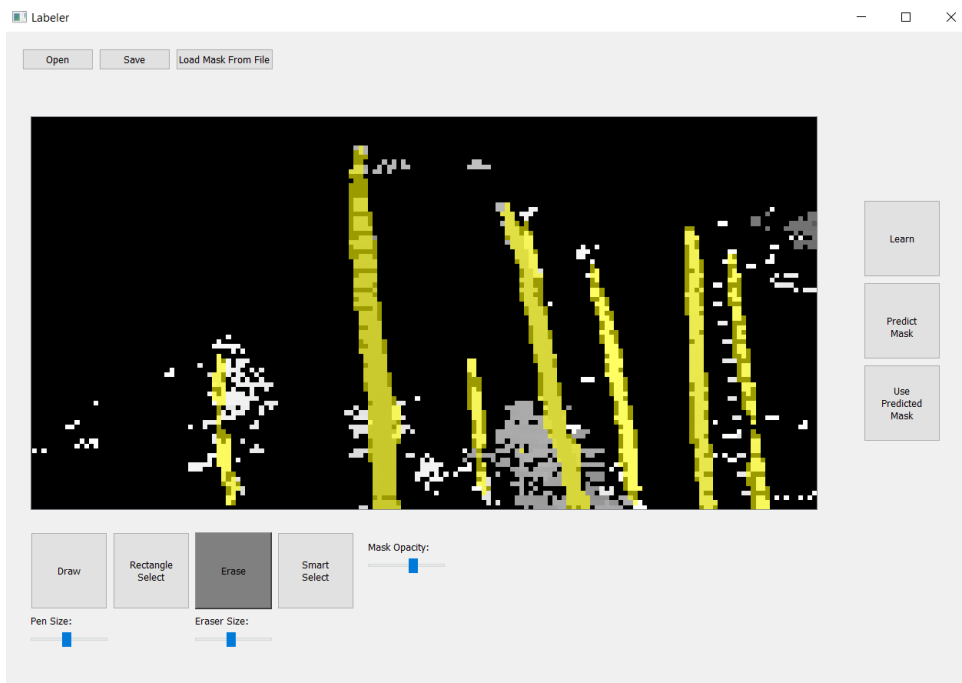
In this section we present detailed discussion of the application's functionality.



**Figure 4.1:** Descriptive picture of the application

1. The "Open" button allows users to open a PCD file for editing within the application. Upon clicking the button, a file dialog is triggered, prompting the user to select a file. Once the file has been chosen, the PCD file is transformed into a grayscale image with correctly shifted pixels, which is then displayed.
2. The "Save" button enables users to save their work, including combined images of the background image and mask data. Clicking the button initiates the save process for the currently opened project in the labeler. The mask is saved as a NumPy binary file, and the values of each pixel of the background image are saved as well.

3. The "Load Mask from File" button enables the user to import a previously saved mask into the current labeling session. This eliminates the need to start from scratch, allowing the user to fix previously saved masks without the need to start over.
4. The main display area is a large area that displays the image currently being worked on. The user is able to draw and erase the mask within this area. The image can be zoomed in and out using the scrolling wheel. While holding the right mouse button pressed, the user is able to move and drag the image. In order to guarantee a smooth and proportional transition across varying zoom levels, the delta of mouse movement is adjusted in accordance with the current zoom scale. While annotating users will, for the most part, employ a zoomed-in perspective, Figure 4.2 illustrates the zoomed-in picture.



**Figure 4.2:** Main display area while zoomed-in

5. Upon pressing the "Draw" button, the application enters drawing mode, capturing the user's mouse movements over the main display area to draw lines. The slider located below the button allows the user to change the thickness of the line.
6. The "Rectangle Select" tool is employed to delineate areas of the image in a rectangular configuration. A provisional mask is generated to illustrate the outlines of the rectangle, assisting the user in the accurate creation of the desired object.
7. "Erase" button sets the application into erase mode, basically doing the opposite of drawing. Slider to change the size of eraser is also present.

8. The "Smart Select" tool enables users to select contiguous areas within an image based on colour similarity around a specified point  $(x, y)$ . This tool is particularly useful for isolating regions with consistent colour patterns in complex visual data. The depth-first search (DFS) approach is employed for selecting the region, where a stack is initiated with the starting point. The function then iterates over the stack until it is empty:
  - The coordinates of each pixel are validated against the image boundaries and the visited array in order to avoid redundant checks.
  - If the colour of the current pixel is similar to the target colour, the pixel is added to the mask.
  - Subsequently, neighbouring pixels (up, down, left, right) are added to the stack for further exploration.
9. The "Mask Opacity" slider enables the user to adjust the opacity of the masked areas, thus allowing them to view the underlying image to varying degrees. This is of particular utility when correcting the mask generated by a neural network.
10. The "Learn" button initiates a machine learning process. All previously labeled images are used to train the neural network. If this is the first time the user is training the model, a new dictionary with weights will be created. Otherwise, the dictionary is loaded and the relearning process begins with a slightly lowered learning rate. This is done on a new thread to allow users to continue labeling while the neural network is learning.
11. The "Predict Mask" button utilises the trained model to predict the mask of the currently selected PCD file.
12. The "Use Predicted Mask" button is used to apply the generated mask to the image.

In addition, the software incorporates a series of hotkeys:

- The "Save" (Ctrl+S) shortcut enables users to quickly save their current work without the necessity of clicking on the GUI's save button. When the Ctrl+S keys are pressed simultaneously, the save procedure is initiated.
- Undo (Ctrl+Z) and Redo (Ctrl+Y):

The class `HistoryQueue` serves as a management system for undo and redo operations within our application. This is a particularly useful feature because it is anticipated that users will commonly make mistakes. It works by utilising two stacks: the `undoStack` and the `redoStack`. Upon initialization, the class sets a default capacity to limit memory usage, allowing it to store a specified number of states. When a new state is generated, it is pushed onto the `undoStack`. If this stack reaches its capacity, the oldest state is removed to make room for new entries. This addition prompts the clearing of the `redoStack` to maintain the integrity

of the action sequence, preventing users from redoing actions that no longer align with the latest state. When the user performs an undo operation, the most recent state is transferred to the redoStack and the previous state is reinstated. Conversely, a redo operation moves the state from the redoStack back to the undoStack, effectively reapplying an action that was undone. This dual-stack approach allows for a clear and logical sequence of user actions, providing a robust mechanism for navigating through changes within the application.





## Chapter 5

### Goals and results

This chapter sets out the primary objectives of this thesis and presents the results achieved towards these objectives. The overall objective was to improve the efficiency and effectiveness of LiDAR data annotation for tree trunk segmentation, which is critical for improving autonomous drone navigation in forested environments. In particular, this project aimed to develop a user-friendly annotation tool that reduces the time and effort required for manual annotation. This meant that we aimed to train a robust convolutional neural network (CNN) capable of accurately detecting tree trunks from LiDAR data to assist the user while annotating. We hope that the trained model could be then used for drone navigation in real-world settings.

The successful achievement of these goals would not only streamline the annotation process, but also allow us to improve the performance of autonomous navigation systems by enabling more accurate and reliable environmental perception that could be performed while flying.



#### 5.1 Defining our evaluation metrics

In this thesis, we employed several evaluation metrics to assess the performance of our neural network model in detecting tree trunks from LiDAR data. Accuracy<sup>5.1</sup> was used to measure the overall correctness of the model, defined as the ratio of correctly predicted observations to the total observations. Precision<sup>5.2</sup> was employed to evaluate the model's performance in terms of the proportion of positive identifications that were actually correct, indicating the model's ability to avoid false positives.

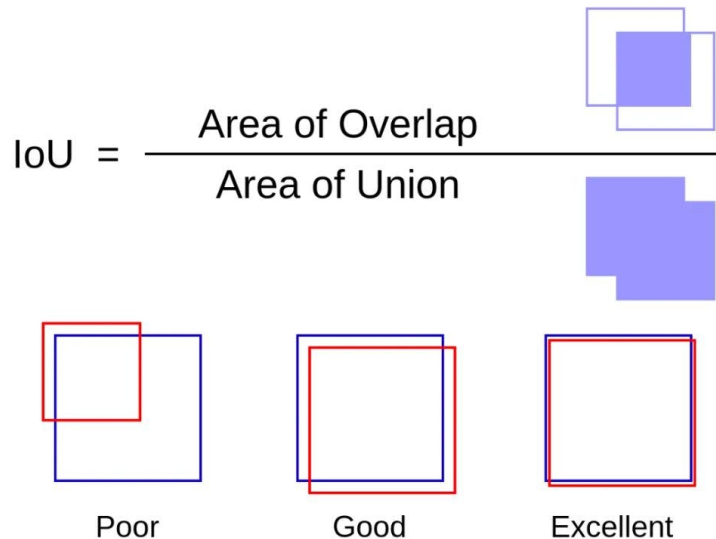
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.1)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5.2)$$

where:

- *TP* (True Positives) is the number of positive instances correctly identified by the classifier,
- *TN* (True Negatives) is the number of negative instances correctly identified by the classifier,
- *FP* (False Positives) is the number of negative instances incorrectly identified as positive by the classifier,
- *FN* (False Negatives) is the number of positive instances incorrectly identified as negative by the classifier.

However, due to the nature of our data, the accuracy numbers were consistently high and to some extent the precision as well, even if the prediction was not optimal. Therefore, we employed the Jaccard Index (see fig.5.1), which is commonly referred to as the Intersection over Union (IoU). This metric measures the similarity of samples[25]. The IoU is calculated by dividing the area of overlap between the predicted segmentation and the ground truth by the area of their union. This metric is particularly valuable for rating the quality of object segmentation tasks in computer vision, as it provides a robust indication of the model's ability to correctly mark out the target. From this point onwards, we will use the IoU as the primary metric for evaluating the models' performance.



**Figure 5.1:** Jaccard Index (IoU), taken from[26]



## 5.2 Data selection and split

A total of 50 pcd files were selected from the CTU-MRS dataset, which captures various forest scenarios via LiDAR. These files were manually annotated using our labeling application for the purposes of testing the neural networks' capabilities. Given the constraints in data volume, the images were divided into two subsets: 40 for training and 10 for testing. This split was designed to optimise the learning process while ensuring a sufficient level of model evaluation.

The dataset can be divided into three distinct groups based on the complexity and characteristics of the tree trunks depicted. The first group comprises images with easily recognisable trunks, where the tree trunks are clearly defined and distinguishable. The second group features images with a high density of trees, presenting a challenging clutter of overlapping signals. The third group captures trees at a distance, where the trunks are less distinct and more difficult to discern. This diversity ensures comprehensive learning and testing, thereby forcing the model to adapt to the various levels of segmentation challenges inherent in forested environments.

In the testing phase, the objective was to simulate realistic user interactions with the annotation tool, thereby providing the neural network with incremental learning opportunities. The model was systematically trained using different training set sizes to evaluate its learning capabilities and adaptability. Specifically, we conducted training sessions with sets consisting of 10, 20, 30, and finally, all 40 training images, with each stage using the same set of 10 images for testing. This approach allowed us to assess the model's performance and improvements at each level of training exposure. It was crucial to maintain a consistent test set across all training levels to ensure that the performance metrics were comparable, thereby providing clear insights into the gains achieved through incremental learning. This methodology not only tested the robustness of the model under varying degrees of training data but also mirrored potential real-world usage scenarios where the model progressively receives more data as it is used over time.

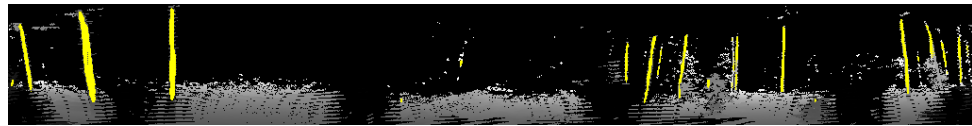
## 5.3 Results on selected images

This section presents a comparative analysis of the labels generated by the neural network when trained with varying sizes of training sets. Additionally, it contrasts these results with those obtained from human annotators. For a more nuanced assessment, we included two scenarios for human annotation: one in which the annotator was assisted by the neural network's predictions, and one where the annotator worked without such assistance. In order to ensure a comprehensive evaluation across different types of data, one image from each of the defined categories – easily recognisable trunks, high-density tree areas, and trees in the distance – was selected for this comparison. This approach allows us to observe the effectiveness of the neural network against

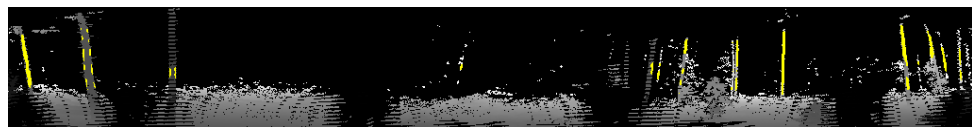
the precision and insight provided by human annotators, under both assisted and unassisted conditions. The repository at GitHub[27] contains exemplary data.



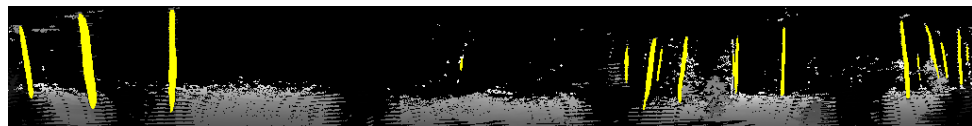
True Label



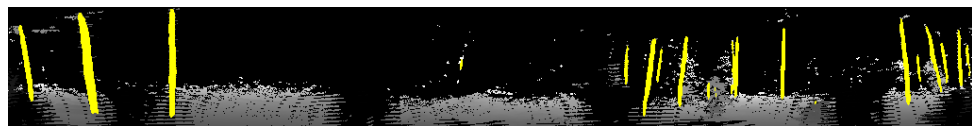
NN - 10 pictures training set (IoU: 0.54)



NN - 20 pictures training set (IoU: 0.21)



NN - 30 pictures training set (IoU: 0.78)



NN - 40 pictures training set (IoU: 0.88)

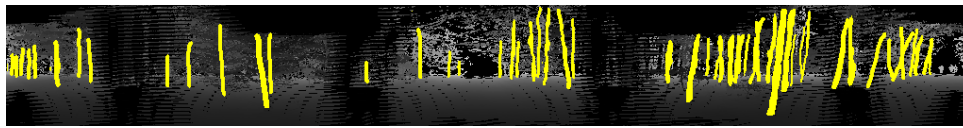


Human without NN assistance (IoU: 0.60)

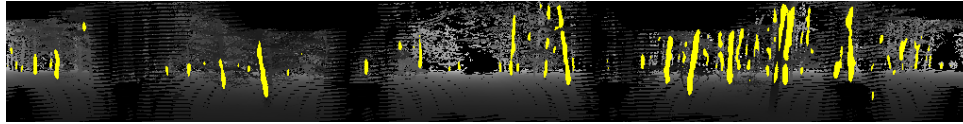


Human with NN assistance (IoU: 0.84)

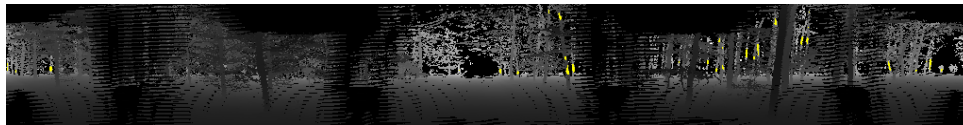
**Figure 5.2:** Data with easy to recognize trunks



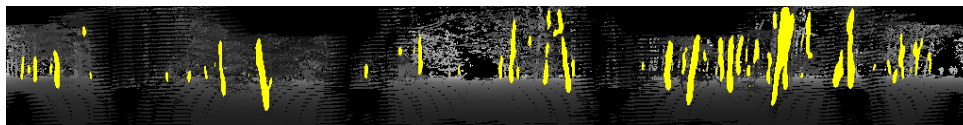
**True Label**



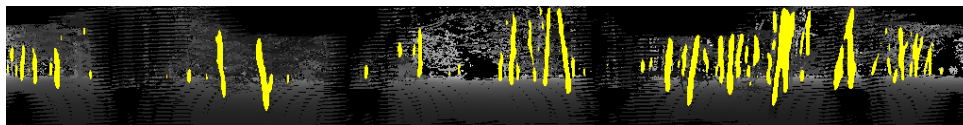
**NN - 10 pictures training set (IoU: 0.38)**



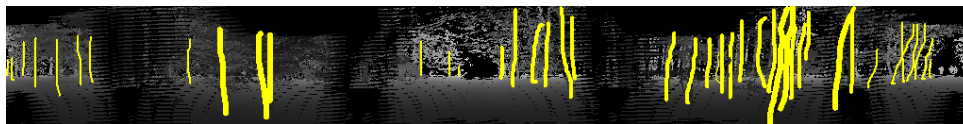
**NN - 20 pictures training set (IoU: 0.02)**



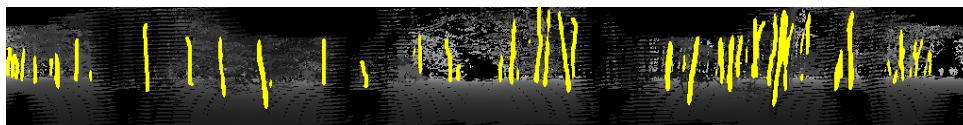
**NN - 30 pictures training set (IoU: 0.47)**



**NN - 40 pictures training set (IoU: 0.53)**

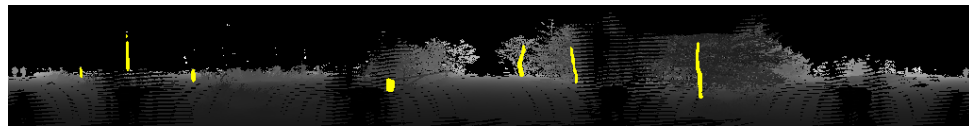


**Human without NN assistance (IoU: 0.41)**

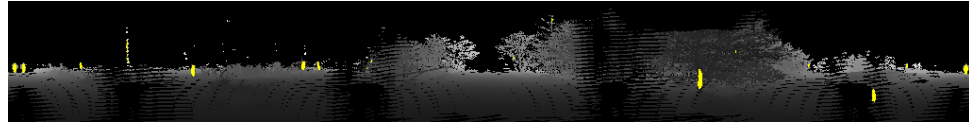


**Human with NN assistance (IoU: 0.49)**

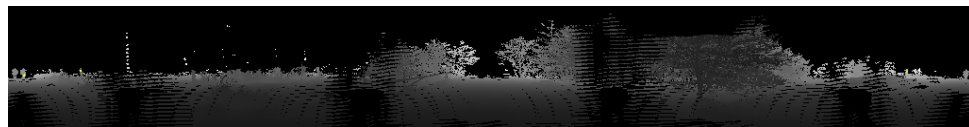
**Figure 5.3:** Data with a high density of trees



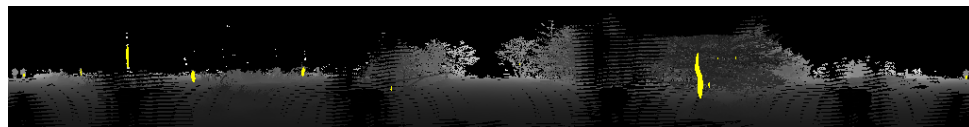
**True Label**



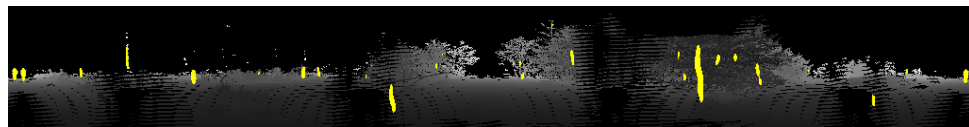
**NN - 10 pictures training set (IoU: 0.14)**



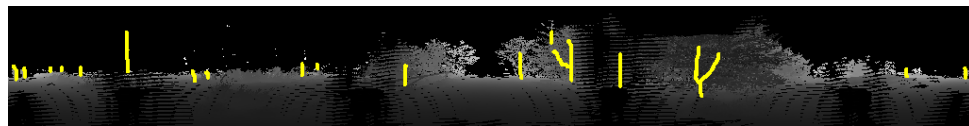
**NN - 20 pictures training set (IoU: 0.00)**



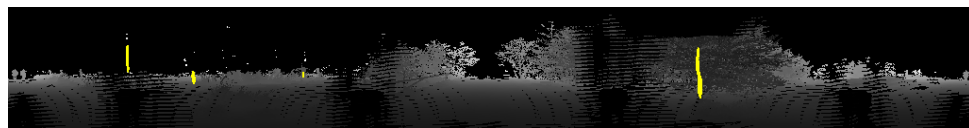
**NN - 30 pictures training set (IoU: 0.33)**



**NN - 40 pictures training set (IoU: 0.48)**



**Human without NN assistance (IoU: 0.22)**



**Human with NN assistance (IoU: 0.38)**

**Figure 5.4:** Data with trees in the distance

## 5.4 Complete results

The table 5.1 below presents the average results achieved by each of our neural network models, trained on varying sizes of training data, on the testing set, along with their respective standard deviation values ( $\sigma$ )<sup>1</sup>. This data provides

<sup>1</sup><https://www.dummies.com/article/academics-the-arts/math/statistics/why-standard-deviation-is-an-important-statistic-169731/>

insight into how the quantity of training data impacts the model’s ability to accurately segment and annotate tree trunks. Alongside these metrics, we also compare these results with those achieved by human annotators, both with and without the assistance of the neural network. By incrementally increasing the training set size, we can observe the corresponding changes in performance metrics across the models and how they stack up against human accuracy under varying conditions. This analysis highlights the evolution of the neural network’s proficiency as it processes more data and offers a clear depiction of the benefits of neural network assistance in manual annotation tasks.

**Table 5.1:** Results achieved on images from testing set

Results	IoU	$\sigma_{\text{IoU}}$	max IoU	Prec	$\sigma_{\text{Prec}}$	Acc	$\sigma_{\text{Acc}}$
10 images NN	0.37	0.11	0.54	0.75	0.19	0.97	0.02
20 images NN	0.08	0.09	0.25	0.80	0.33	0.96	0.02
30 images NN	0.55	0.15	0.78	<u>0.84</u>	0.09	0.98	0.01
40 images NN	<u>0.61</u>	0.16	<u>0.89</u>	0.79	0.11	0.98	0.01
Human	0.42	0.15	0.65	0.55	0.17	0.97	0.02
Human + NN	0.55	0.17	0.84	0.80	0.14	0.98	0.02

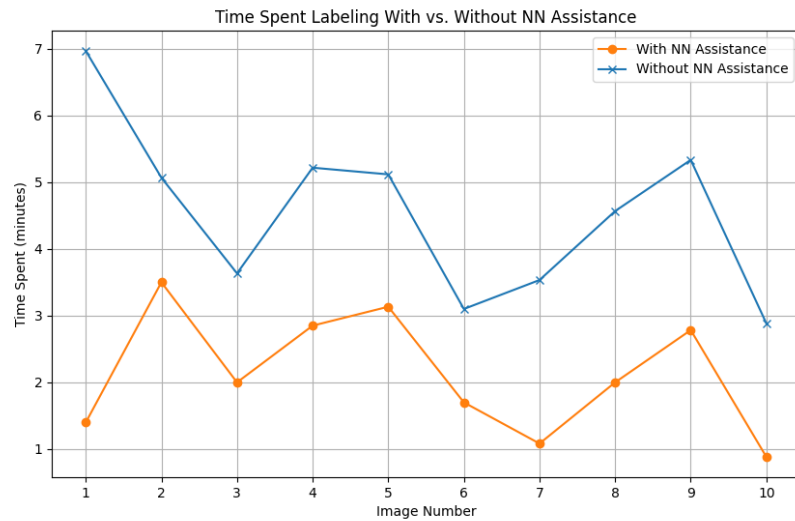
The data suggest that the neural network, when used as an assisting tool, can effectively guide the user towards achieving labels that closely match the true labels. Notably, the average IoU scores achieved by human annotators with neural network assistance are consistently higher compared to those achieved without it. Nevertheless, the relatively high standard deviation values indicate that the neural network is still experiencing difficulties in identifying tree trunks that are more challenging to recognize. However, it is able to accurately recognise tree trunks in the simpler data, as evidenced by the maximum IoU, which explains the significant difference in standard deviation.

This improvement underscores the practical benefits of integrating advanced machine learning models into traditional annotation workflows. It not only enhances the accuracy of the annotations but also aids in maintaining consistency across different sets of data and especially between different annotators. These findings reinforce the value of neural network assistance in significantly boosting the precision of human-driven data labeling efforts.

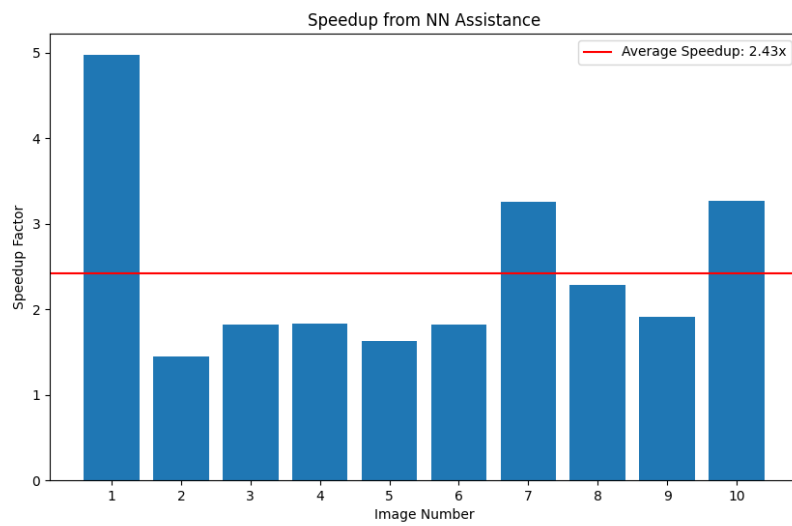
### 5.4.1 Reduction of time requirements

As a key objective of our project was to reduce the time and effort required for annotating images, the following graphs provide a clear illustration of the time needed to annotate one picture. Figure 5.5 compares the duration of annotation sessions with and without neural network assistance. Figure 5.6 displays the speedup factor for each individual image and the average speedup across all images.

By quantifying the time saved when annotators are assisted by the neural network, it is possible to evaluate the practical impact of integrating AI tools in the annotation process. This analysis is crucial for understanding how much our tool not only improves accuracy but also enhances efficiency in real-world annotation tasks.



**Figure 5.5:** Comparison of time spent labeling images with and without neural network assistance.



**Figure 5.6:** The visualisation of the speedup gained from neural network assistance for each image, accompanied by an average speedup factor.





## Chapter 6

### Future improvements and work

#### 6.1 Multiclass labeling

In the current scope of our project, the focus was on segmenting tree trunks, which required only single-class labeling to meet our specific research objectives. However, the expansion to multiclass labeling could substantially enhance the tool's applicability and utility for a broader range of users. Such an enhancement would permit the annotation tool to identify and categorise multiple types of objects within a single image, such as different kinds of vegetation, wildlife, or artificial structures. This would provide a more comprehensive understanding of forested or other environments. This capability would permit the creation of more complex autonomous navigation systems that operate in diverse settings. Implementing multiclass labeling would transform the tool into a more versatile resource, adapting to a wider variety of research needs and operational scenarios.


#### 6.2 Exploring the Integration of Multiple Neural Network Models

In addition to U-Net, we plan to investigate the potential of integrating other neural network architectures into the annotation tool. This will allow us to leverage the strengths of different models to handle various aspects of LiDAR data. For instance, some models may excel at identifying natural landscapes, while others may perform better in urban environments.

To enhance the reliability of the annotations, we might implement a model voting system. Multiple neural network models will independently analyse the LiDAR data and provide annotations. The annotations will then be aggregated and presented to the user, who will be able to select the most suitable one. This model voting approach is designed to reduce the likelihood of errors and biases associated with a single model, thereby improving the overall accuracy of the automated annotations.

### ■ 6.3 The long-term objective of enabling autonomous drone navigation

The long-term objective of integrating our trained neural network into autonomous flight systems represents a significant advance in enhancing the operational capabilities of drones, particularly in complex environments such as forests. This integration is intended to empower drones with the ability to dynamically recognise and navigate around tree trunks, thereby optimising their pathfinding capabilities and reducing the risk of collisions. Such an advancement would not only allow drones to adhere to pre-set flight paths but also to make informed, real-time decisions based on the surrounding terrain. The expected applications include environmental monitoring, search-and-rescue missions, and forest management, which demonstrate the potential for more autonomous, effective, and safe aerial operations in challenging landscapes. This goal drives our continued development and refinement of the neural network to meet the demands of real-world autonomous flight.



## Chapter 7

### Conclusion

This thesis explores LiDAR data analysis, combining LiDAR technology with convolutional neural networks (CNNs) to segment tree trunks from LiDAR data. The primary objective of this work is the development of specialised annotation tool with the ability to learn – "Labeler" – which will allow for faster and better creation of annotated datasets used for the training and testing of our neural networks.

The fundamental divergence of this project from standard methods lies in the transformation of LiDAR data into a two-dimensional format suitable for CNN processing. This methodological shift not only simplifies the visualisation of spatial data and makes them easier to work with in general, but also maximises the pattern recognition capabilities of CNNs, particularly the adapted U-Net model.

The U-Net model, which is known for its effectiveness in biomedical image segmentation, has been adapted for LiDAR data analysis. This adaptation of U-Net for the segmentation of tree trunks represents an advancement in the use of CNNs for environmental applications. It demonstrates the versatility and adaptability of these networks. The success of the project in accurately segmenting tree trunks, highlights the potential of CNNs in interpreting and analysing complex spatial data gathered from LiDAR.

The use of PyTorch as the framework for this project was crucial, particularly in the adaptation and refinement of the neural network architecture. PyTorch's dynamic and flexible environment enabled the experimentation and iterative development of the neural network, allowing for a tailored approach to handle the LiDAR data. PyTorch's Dataset and DataLoader classes greatly improved the efficiency of data management and model training. These tools simplified the handling of large datasets and optimized the training process, ensuring effective training of the neural network on the prepared datasets.

The integration of the U-Net model into our developed annotation tool represents a significant advance in the automation of the data annotation process. By reducing the time and effort required for manual annotation, this tool has proven to be a valuable asset in streamlining the workflow of LiDAR data handling. The integration of machine learning techniques has not only enhanced the precision of annotations but has also facilitated a more user-friendly interface that aids in the tedious act of manual labeling. This

has led to more effective and efficient training and validation of our CNN models.

Looking ahead, the project has laid a solid foundation for future advancements. The potential expansion to multiclass labeling and the integration of various neural network models could address a broader spectrum of environmental features, thereby enhancing the tool's utility across different research and operational contexts. Furthermore, the long-term goal of integrating this technology into autonomous drone systems highlights its potential impact on the field of self-navigating drones.

In conclusion, this work has created a tool that will hopefully be used for improving the functionality and safety of autonomous systems in different environments. It has also contributed valuable insights into the processing of LiDAR data using CNNs. The methodologies and tools developed in this thesis provide a scalable solution that could be adapted for various other applications, potentially revolutionising the way we interact with LiDAR data.



## Bibliography

- [1] SuperAnnotate. What is LiDAR Annotation. Available at: <https://www.superannotate.com/blog/what-is-lidar-annotation#uses-of-deep-learning-with-lidar-data> [Accessed 29 April, 2024].
- [2] Songan Zhang. 3D LiDAR Annotator. Available at: <https://github.com/songanz/3D-LiDAR-annotator> [Accessed April 29, 2024].
- [3] Sager, Christoph, et al. “labelCloud: A Lightweight Labeling Tool for Domain-Agnostic 3D Object Detection in Point Clouds.” *Computer-Aided Design and Applications*, vol. 19, no. 6, Mar. 2022, pp. 1191–206. DOI.org (Crossref), <https://doi.org/10.14733/cadaps.2022.1191-1206>.
- [4] X. Zhou, Y. Feng, X. Li, Z. Zhu, and Y. Hu. *Off-Road Environment Semantic Segmentation for Autonomous Vehicles Based on Multi-Scale Feature Fusion*. *World Electric Vehicle Journal*, 14, 291, 2023. <https://doi.org/10.3390/wevj14100291>
- [5] C. R. Qi, H. Su, K. Mo, L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. arXiv:1612.00593 [cs.CV] <https://arxiv.org/abs/1612.00593>.
- [6] H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. arXiv:1505.00880 [cs.CV] <https://arxiv.org/abs/1505.00880>.
- [7] Nagy, Balazs, and Csaba Benedek. “3D CNN-Based Semantic Labeling Approach for Mobile Laser Scanning Data.” *IEEE Sensors Journal*, vol. 19, no. 21, Nov. 2019, pp. 10034–45. DOI.org (Crossref), <https://doi.org/10.1109/JSEN.2019.2927269>.
- [8] SZTAKI CityMLS dataset. Available at: <http://mplab.sztaki.hu/geocomp/SZTAKI-CityMLS-DB.html>.
- [9] Kim, Wonjik, et al. “Automatic Labeled LiDAR Data Generation Based on Precise Human Model.” 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 43–49. DOI.org (Crossref), <https://doi.org/10.1109/ICRA.2019.8793916>.



- [22] PyTorch. *torch.nn.CrossEntropyLoss*. PyTorch Documentation, 2024. Available at: <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html> [Accessed 10 January, 2024].
- [23] PyQt5. Available at: <https://pypi.org/project/PyQt5/> [Accessed April 29, 2024].
- [24] Labeler application. Available at: <https://github.com/pyszkad1/lidar-tree-labeling/tree/main>
- [25] Adrian Rosebrock. Intersection over Union (IoU) for object detection. Available at: <https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>, November 7, 2016.
- [26] Ivan Popov. A Deep Dive into Semantic Segmentation Evaluation Metrics. Available at: <https://hackernoon.com/a-deep-dive-into-semantic-segmentation-evaluation-metrics>.
- [27] Exemplary data. Available at: <https://github.com/pyszkad1/lidar-tree-labeling/tree/main/data>